The Dissertation Committee for Jeremy Michael Stober

certifies that this is the approved version of the following dissertation:

Sensorimotor Embedding:
A Developmental Approach to Learning Geometry

Committee:

Risto Miikkulainen, Supervisor

Benjamin Kuipers, Co-Supervisor

Kristen Grauman

Peter Stone

Inderjit Dhillon

Sensorimotor Embedding:
A Developmental Approach to Learning Geometry

by

Jeremy Michael Stober, B.A.; M.S. Comp. Sci.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2015

For Anastasia.

Acknowledgments

I'd like to thank Risto Miikkulainen and Benjamin Kuipers, without whose supervision this thesis would not exist. I would also like to thank the members of my committee, Kristen Grauman, Peter Stone, and Inderjit Dhillon, who waited patiently while I worked out the kinks of this research. I'd like to also thank the members of The Golden Turtle Club, who reminded me that while I might have been behind, I was not alone, and extend thanks to my colleagues at Apple, who had the good grace to not ask about my progress while also making every workday a pleasure.

Anastasia and Alexander, you have kept me going as you have kept me grounded. Now that this thesis is finally moving out, we will have even more room for a new addition.

Jeremy M. Stober

The University of Texas at Austin
May 2015

Sensorimotor Embedding:
A Developmental Approach to Learning Geometry

Publication No. ____________

Jeremy Michael Stober, Ph.D.
The University of Texas at Austin, 2015

Supervisor: Risto Miikkulainen
Co-Supervisor: Benjamin Kuipers

A human infant facing the blooming, buzzing confusion of the senses grows up to be an adult with common-sense knowledge of geometry; this knowledge then allows her to describe the shapes of objects, the layouts of places, and the relative locations of things naturally and effortlessly. In robotics, such knowledge is usually built in by a human designer who needs to solve complex engineering problems of sensor calibration and inference. In contrast, this dissertation presents a model for how autonomous agents can form an understanding of geometry the same way infants do: by learning from early unstructured sensorimotor experience.

Through a framework called sensorimotor embedding, an agent reconstructs knowledge of its own sensor structure, the local geometry of the world, and the pose of objects within the world. The validity of this knowledge is demonstrated directly through Procrustes analysis and indirectly by using it to solve the mountain car task with different morphologies. The dissertation demonstrates how sensorimotor embedding can serve as a robust approach for acquiring geometric knowledge.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The doctrine that we could not perceive the world around us unless we already had the concept of space is nonsense. It is quite the other way around: We could not conceive of empty space unless we could see the ground under our feet and the sky above. Space is a myth, a ghost, a fiction for geometers. –J. J. Gibson *The Ecological Approach to Visual Perception*

If asked, you could probably describe where you are in the building, in your office, or relative to your desk. This ability to answer *where* questions as adults is natural, but if we were to rewind time and consider ourselves as newborns, this knowledge would seem impossibly out of reach. Between that time and now, we acquired knowledge of the geometry of our local space. This thesis introduces a formal, algorithmic, developmental approach to acquiring geometric knowledge called sensorimotor embedding. This approach allows artificial agents to acquire geometric knowledge from experience. In this introduction, the importance of geometric knowledge for different disciplines is discussed along with examples that motivate the development of sensorimotor embedding. This is followed by a discussion of the challenges addressed by this thesis, and an overview of the approach presented in this dissertation. The last section provides an outline of the remaining chapters.

1.1 Motivation

1.1.1 Geometry in the Liberal Arts

Not everything to do with the geometry of local space is easy even for adults. The difficulty of representing space in art points to a gap between human intuition regarding perception of geometry and our scientific understanding. For thousands of years of recorded history humans struggled to study and formalize what we all understand inherently and immediately by simply observing the world. It took an advance in scientific understanding to finally revolutionize artistic representations of space.

The very first depictions of depth in paintings appeared in the work of Filippo Brunelleschi sometime before his death in 1446. This work, as well as other later attempts at capturing perspective, such as Vredeman de Vries perspective plate number 28 shown in Figure 1.1, were influenced by early Renaissance studies of Optics (e.g. Alhazen's *Deli Aspecti* [1]).

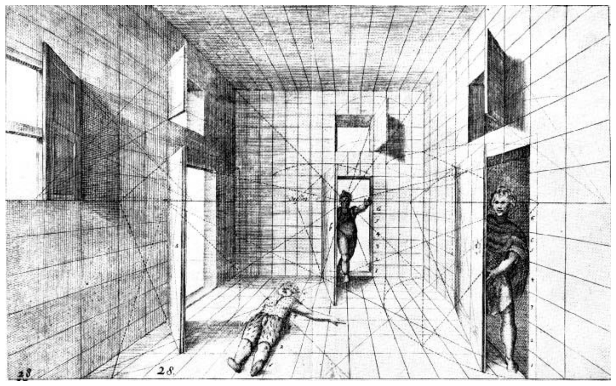


Figure 1.1: **Perspective Plate #28**. Vredeman de Vries in 1604. Early efforts to model three dimensions in art came about after advances in the study of optics.

The mathematical study of geometry preceded these artistic advancements by a number of centuries, from the first proof of the Pythagorean theorem in the seventh century BC through the discovery of non-euclidean geometries in the 19th century. Euclid's *Elements* laid out what was known about geometry in a system of axioms, theorems, and proofs. The parallel postulate is a famous example of how intuition about obvious geometric postulates can turn out to be incomplete.

> If a straight line falls on two straight lines in such a manner that the interior angles on the same side are together less than two right angles, then the straight lines, if produced indefinitely, meet on that side on which the angles are less than the two right angles.

The parallel postulate was qualitatively more complex than the other postulates. Since the postulate seemed empirically true, many geometers attempted to prove that this postulate could be derived from the proceeding four, simpler propositions. The discovery of consistent non-Euclidean geometries opened up new areas of mathematics, but also created a schism between the mathematical study of geometry and our perceptions. Since there exist multiple consistent geometries, it was no longer reasonable to assume that our knowledge of geometry could be deduced independent of experience. This distinction is important for this thesis. The methods and results discussed here are not concerned with the formal mathematical study of geometry. Sensorimotor embedding is a theory about how geometric knowledge can be acquired through perception, and reasoned about using the naive, common sense methods.

1.1.2 Geometry in Psychology

Our informal knowledge of geometry is more surprising when you consider the complex process involved in translating perceptions into mental states. A straight line in the world must first be detected by a concave set of receptor cells in the retina, transformed into neural impulses, then transmitted to the visual cortex for further processing. This complexity gave rise to several theories for grounding mental states as sensorimotor contingencies or motion based grounding [13, 46].

At some point in development, children learn geometric concepts. Children can pass reasoning tasks requiring Euclidean geometry at six years of age, during Piaget's **concrete operational phase** [63]. Piaget's theory of cognitive development provides a useful framework for thinking about how geometric knowledge might come to be during the process of cognitive development. And operational geometric thinking in children must depend on foundational knowledge acquired during the sensorimotor and pre-operational stages of development [52].

Inspired by Piaget's sensorimotor stage of development, agents using sensorimotor embedding acquire geometric knowledge directly from sensorimotor experience.

Humans are remarkably adaptable in the face of sensorimotor change. For example, in a famous series of experiments during the 1950s, Erismann and Kohler filmed themselves doing everyday tasks using vision inverting goggles [78]. An experiment shown in Figure 1.2 shows Erismann attempting to navigate an inclined plane while wearing inverting goggles. The ability to function after a period of adaptation despite a drastic change in the relationship between the visual sense and the state of the world epitomizes the advantage that humans have over even the most sophisticated robots.

These early vision inverting experiments formed the basis of a months long experiment in wearing inverting goggles undertaken by Hubert Dolezal. In *Living in a World Transformed*, Dolezal describes the stages of adaptation to his transformed visual surroundings, advancing so far as to consider the inverted world "normal" and being able to undertake daily tasks without difficulty [15]. After taking the goggles off, there was a similar, though shorter period of adaptation to normal vision.

Being able to adapt to a drastic change in the properties of a sensor is a

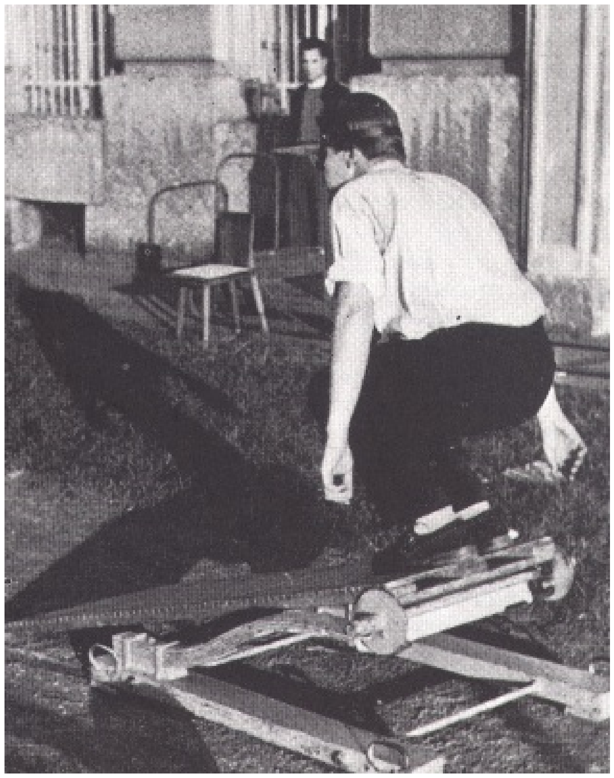


Figure 1.2: **Inverting Vision Experiments**. Inverted vision experiments show that this sensory change disrupts basic skills in the short term, but humans can eventually adapt to such changes.

significant skill. Consider attempting the same experiment on the most advanced robots today. Would a robot be able to detect the change or adapt to it over time? Given advance notice, an engineering solution could be found, yet this ability of humans to adapt to sensory changes happens naturally.

This example is especially important because it involves a change to the geometry of the visual field. Any mechanism that purports to learn geometry from experience should be able to pass the inverted goggles test. Sensorimotor embedding provides a solution as shown in Chapter 4. People have also demonstrated the ability to adapt to reduced vision or blindness. When simulating these same sensory deficits on robots, for example simulating macular degeneration or central vision loss by artificially restricting a robot's field of view, even today's most advanced robots fail in any programmed functions that depend on these senses. This sharp contrast between human adaptation and robot adaption is a key motivation for the approach in this dissertation.

1.1.3 Geometry in Robotics

The interplay of psychological and mathematical approaches to the study of geometric knowledge serve as the background for another important research area. A robot is a collection of sensors capable of perceiving the environment and effectors capable of making physical changes to the environment. These changes could involve manipulating objects or navigating a path. Typical robots in common laboratory use have camera, laser, or sonar sensors, appendages like robot arms capable of manipulating physical objects, and wheels or legs capable of moving the robot. For autonomous robots, the ability to maintain accurate internal representations of the state of the world depends crucially on the acquisition of geometric knowledge of all kinds.

For today's most advanced robotic systems, unlike their biological counterparts, the ultimate source of knowledge about how to make geometric estimates is the programmer responsible for designing the system. This lack of autonomy in the acquisition of geometric knowledge can sometimes lead to problems. For example, consider the case of the Rosetta space probe. On November 12th, 2014 European Space Agency's Rosetta spacecraft landed a probe on a comet for the first time. After a decade, and 500 million kilometers, harpoons meant to secure the probe to the comet in the low gravity environment, failed to properly deploy. According to subsequent analysis the probe bounced twice, finally coming to rest in an unknown region of the comet's surface. Fortunately, the probe was able to complete a substantial portion of its research mission, though the length of its mission was cut short since in its new orientation, the solar panels did not receive enough sunlight to supply the batteries.

A crucial question puzzled scientists after the probe came to rest on the rock – **where was it?** Answering this question from so far away was a substantial challenge. Analyzing a specific sensor on the lander, the Rosetta Lander Magnetometer and Plasma Monitor or ROMAP, gave some clues as to the probe's trajectory [2]. When a robot is so far away, our ability to reason about the robot's position is restricted to what is available from the robot's sensors. But what if it were possible to just ask the lander where it was? Some of the necessary knowledge needed for a response will be provided by sensorimotor embedding. After such an unexpected event altered the position of the probe, an adaptive, autonomous system that could learn (or relearn) to estimate its position would have been of great use to the mission team.

Autonomous robots are expected to operate independently in the environment, performing useful tasks and reacting to changes without any input from an operator. This creates several problems that must be solved by the robot's controlling routines. For example, unless operating within a tightly controlled environment, the robot must be able to adapt to many kinds of unexpected changes. An **autonomous learning robot** is a robot that, in addition to acting independently in the world, is also capable of learning new skills and concepts. This includes robots that undergo **autonomous mental development**, a process inspired by the way humans and animals develop over a lifetime [81]. The process of autonomous mental development is online, continuous, and lifelong. Sensorimotor embedding provides a method for autonomous learning of geometry from online experience.

1.2 Challenge

Autonomous mental development is clearly evident in natural systems, and autonomous learning would significantly improve the ability of artificial systems to adapt to new environments and acquire new and useful skills. In addition, computational models for artificial development may provide a deeper understanding of developmental processes in natural systems. One problem for autonomous mental development is how to build a foundation of knowledge from raw sensorimotor experience. This is the **bootstrap learning problem**.

Bootstrap Learning Problem. Create a general learning program that, when run on a robot, is able to learn useful concepts from raw sensorimotor experience.

Bootstrap learning programs that can run effectively on a wide variety of robots without customization would require less engineering effort to port between different platforms. If the specifics of the physical robot are not part of the program, but learned from experience, then the program should generalize well over a wide variety of possible robots. Even if properties of the robot architecture are built into the program, wear and tear can change the properties of the robot over time. These kinds of changes over time need to be discovered by the robot.

In bootstrapping's most stringent form, even essential facts, like the dimension of space, are not necessarily available. Discovering the number of dimensions of space from a stream of input data is a difficult task. But embedded in this challenge problem are more practical difficulties. For example, how can a robot learn the proportions of its own body? How can it properly calibrate its sensors (or know that they need calibrating)? Can the agent learn basic skills that allow it to control the input stream of data in predictable ways? These questions inspired the **geometry learning problem**.

Geometry Learning Problem. Design a developmental process that, starting only with a basic set sensor inputs and motor outputs, progresses through a period of sensorimotor development that results in knowledge of body, sensor and object location and geometry [71].

If a robot can autonomously learn about its own geometry then less effort is required by the robot designers. If this knowledge can be learned, then it can be relearned, allowing robots to adapt to physical changes.

1.3 Approach

This thesis presents a viable method for learning geometry. Sensorimotor embedding, the core algorithm in this thesis, allows an agent to learn about geometry in a way that is robust to initial conditions, will work in a variety of starting body configurations, and requires minimal prior knowledge. Sensorimotor embedding is an algorithm that allows embodied agents to discover structure in high-dimensional sensor streams through interactive experience. The algorithm was developed to address the problem of a robot trying to learn the geometric structure of its sensors, motors, and the surrounding world. Coincident with the process of learning skills, a robot can also learn about the underlying geometry of the world using sensorimotor embedding.

For many sensorimotor tasks, the actions an agent takes change the local space. An infant waving her hands is creating visual targets with varying position in the infant's egocentric frame of reference. The actions that change the position of an infant's hands directly impact the geometric relationship between the infant's eyes and hands. This leads to a simple but powerful observation, that the actions of an agent are an important source of information about these local geometric relationships. Through careful analysis of action sequences, sensorimotor embedding is able to construct features that relate to geometric properties of the external world.

1.4 Outline

This dissertation is organized as follows:

Chapter 2 presents a self-contained introduction to reinforcement learning, manifold learning, and developmental robotics. A basic understanding of reinforcement learning and manifold learning are necessary for understanding sensorimotor embedding. This chapter includes a description of several reinforcement learning and manifold learning algorithms used in this thesis. An overview of related work in the areas of developmental robotics, reinforcement learning, and manifold learning are also included. Important results in these areas are discussed and provide context for the development of sensorimotor embedding.

Chapter 3 presents sensorimotor embedding in formal detail. This chapter also includes a discussion of evaluation methods that are used in experimental work presented in later chapters. The chapter concludes with a discussion of results from psychology that motivate certain design decisions.

Chapter 4 demonstrates how sensorimotor embedding can be used to learn the structure of a foveated retina. Inspired by experiments in psychology, the robust nature of sensorimotor embedding is evaluated using both a lesion experiment and a vision inversion experiment.

Chapter 5 demonstrates how sensorimotor embedding can be used to learn robot position in both Gridworld and Roving Eye domains. Agent position is inferred using sensorimotor embedding in both these domains.

Chapter 6 demonstrates how sensorimotor embedding can be used to learn object pose. Experiments in this chapter are inspired by psychological studies that demonstrate view preference biases in human subjects. Experiments show how an agent can combine view biases with sensorimotor embedding to learn features corresponding to object pose.

Chapter 7 demonstrates how sensorimotor embedding can be used to learn depth features. These features are learned from stereo images and in simulation using a robot that supports vergence. A brief overview of vergence strategies is included, along with a discussion of human vergence models.

Chapter 8 demonstrates how sensorimotor embedding features can be used to learn a policy in the Visual Mountain Car domain. This experiment uses learned geometry to solve a control problem, and demonstrates that agents can build new skills on knowledge learned using sensorimotor embedding.

Chapter 9 includes a general discussion of all the experimental results and potential avenues for future work, including a discussion of other potential applications of sensorimotor embedding. Future work includes extending sensorimotor embedding using probabilistic policies, combining sensorimotor embedding with human vergence models, and integrating this algorithm into more general developmental programs.

Chapter 10 concludes the dissertation with a summary of contributions and final conclusions.

Chapter 2

Background and Related Work

Geometry is not true, it is advantageous. –Henri Poincaré

This chapter serves as a self-contained overview of developmental robotics, reinforcement learning, and manifold learning. The work presented in subsequent chapters will draw on techniques and ideas in each of these areas. Each section also includes a review of selected related work.

2.1 Developmental Robotics

2.1.1 Introduction

Autonomous mental development has been proposed as the solution to many of the problems of traditional robotics [81]. This approach seeks to solve the problem of programming robots for non-specific tasks in open environments. The goal of autonomous mental development is to create a developmental program that undertakes mental development similar to what is observed in biological systems. A developmental program builds up knowledge over time from raw experience. Robots that develop knowledge autonomously could potentially adapt to changes in the same way humans do. Robots programmed to develop autonomously can also serve as a platform for testing theories of biological development.

2.1.2 Bootstrap Learning

For a more formal view of development, consider an agent solving the *bootstrap learning problem.* This agent starts with an uninterpreted sensor and motor interface and needs to learn appropriate abstractions that allow the agent to function. Consider an uninterpreted sensor interface as a time sequence of vectors $z_t = (z^0, z^1, \ldots, z^n)_t$ where each $z^i \in \mathbb{R}$. A motor interface is a similarly general sequence of vectors $u_t = (u^0, u^1, \ldots, u^k)_t$ where each motor variable can be set by the agent.

The world in which the agent is embedded dictates how the agent's motor commands affect subsequent sensory signals. Note that the world as it is and the agent's perception of the world are separate entities. Let $s \in \mathcal{S}$ denote the state of the world. The function $g : \mathcal{S} \to \mathcal{Z}$ where $\mathcal{Z}$ is the space of sensory signals determines how world state results in sensor signals. The function $h : \mathcal{Z} \to \mathcal{U}$ is the agent's control function or method of choosing motor actions from the space of possible motor commands $\mathcal{U}$ given the agent's knowledge and current stimulus. The *world function* $f : \mathcal{S} \times \mathcal{U} \to \mathcal{S}$ determines how an agent's actions alter the subsequent state of the world.

The complete agent-world system has the following formal structure:

$$z_t = g(s_t) \tag{2.1}$$

$$u_t = h(z_t) \tag{2.2}$$

$$s_{t+1} = f(s_t, u_t). \tag{2.3}$$

where only h and the variables z_t and u_t are known to the agent. The goal of a bootstrapping agent is to learn a reasonable model of the external world from immediate perceptions over a long period of development. To help manage the complexity of this learning task, agents must focus first and foremost on what can be learned from direct sensorimotor experience. Developmental programs search for good abstractions and good control functions in an incremental, lifelong process for a single agent.

Autonomous mental development is potentially agnostic concerning the details of the robot platform. A robot learns basic skills and knowledge over time directly from first-hand experience. In natural systems, autonomous mental development is paired with physical development. In artificial systems, the physical platform is static. For instance, in the case of a robot, techniques that utilize knowledge already in the environment may depend on the robot having certain physical properties, even if the robot does not know it has these properties.

Evolution acting on the physical developmental process may result in certain developmental tricks that simplify mental development. As an example, consider the passive bipedal walker developed by Ikemata et al. [28]. The walker has no control code, no motors, and no sensors. It uses potential energy to walk; the algorithm controlling the movement is entirely encoded in the physical construction of the robot itself. The desired skill, walking, requires no mental development to acquire. The physical properties of the agent already provide the necessary skill.

As another example, consider that compliant materials on contact surfaces of robot grippers often provide additional robustness and adaptability in cases where the target surface may be uneven and the disposition of the object not precisely known [59]. The human hand is one example of a very capable compliant manipulator that has soft textured grip surfaces. For infants, the biological structure of the hand complements a primitive Palmar Grasp Reflex [58]. Pressure applied to the palm of an infant elicits a grasping motion, which is suppressed during later stages of development. This interplay between the physical structure and reflex actions makes the search space manageable for discovering useful early skills.

The process of development may take advantage of particular physical properties of an agent, or general physical properties of sensors, effectors, and the relationship between sensors and effectors. When considering the *bootstrap learning problem*, the platform and environment provide a great deal of structure for the learning process. Much in the same way that human development is contingent on human physiology and physiological development, a developmental program may be reasonably restricted to a constrained class of target platforms.

2.1.3 Related Work

A developmental approach to robotics requires understanding research in a wide variety of disciplines. Most developmental systems are biological, so researchers take inspiration from studies of child development, studies of great apes, and other biological systems that demonstrate developmental changes. As a robotics challenge, the algorithms and implementation depend on many areas of research in computer science. Reinforcement learning and manifold learning discussed below have applications in developmental robot systems.

One important problem in developmental robotics is grounding external world knowledge. Since developmental robots learn from experience, the robot is responsible for acquiring knowledge of the world instead of relying on knowledge encoded in advance through careful engineering. For example, Choe et al. [13] use reinforcement learning to enforce invariant properties in sensors, then associate these internally maintained invariants with external world properties. The approach is limited to grounding sensory perception using induced sensory invariants, though like sensorimotor embedding, the grounding involves properties of agent actions.

Another approach to grounding knowledge is to use *nuisance* functions as a primary method of articulating the bootstrapping problem [12]. A nuisance is an invertible, unknown adversarial function that exists between a robot's sensors and effectors and the external world. An optimal bootstrapping agent needs to be able to adapt to environments that have nuisance functions. In order to make the analysis tractable, nuisance functions are restricted to automorphisms over sets of agent senses and actions. The goal of bootstrapping can then be characterized as designing bootstrapping programs that are invariant to groups of nuisance actions and that make sense for sets of dynamical systems. Sensorimotor embedding is invariant to some nuisance functions. For example, the inverted vision experiment from Chapter 1 is a form of nuisance function. This nuisance function is applied to an agent using sensorimotor embedding in Chapter 4. The agent was able to adapt to this change.

Stronger et al [72] developed an approach to learning sensor and actuator models called Autonomous Sensor and Actuator Model Induction (ASAMI). This approach bootstraps sensor and actuator models by learning both simultaneously, and using the partially learned model of one type to aid in the estimation of the model of the other type. This technique was applied on a Sony Aibo ERS-7 robot, which was able to learn self-consistent internal action and sensor models.

Hart et al. [24] recently demonstrated a developmental learning process that involved a sequences of bootstrapping calibrations on a humanoid robot. The end result of this calibration process was an internal model of the robot's joint and sensor geometry with sufficient capabilities to make mirror recognition possible. At a high level, this approach involves calibrating a kinematics model using data collected from marker tracking on joints during a period of guided self-motion of the robot. With a learned kinematics model, the authors showed that cameras can then be calibrated using the kinematics model. This approach is inspired by the idea that infants use their own hands as tracking targets for early sensorimotor calibration.

One important element of this bootstrapping approach is that the first calibration stage, learning a kinematic model, depends on having a calibrated camera and marker tracking system. While bootstrapping camera calibration from a kinematic model is an important contribution, if learning the kinematic model requires a calibrated camera, the bootstrapping process still requires ground truth information. Tracking targets are another perceptual goal that may be useful for agents using sensorimotor embedding.

2.1.4 Conclusion

This section provided an overview of bootstrap learning, and provided some examples of related work. As demonstrated in the following chapters, sensorimotor embedding solves part of the bootstrap learning problem by learning geometric features. Sensorimotor embedding depends on the agent being able to learn policies that achieve simple perceptual goals. Even if the physical properties of the platform change, sensorimotor embedding can run unaltered to learn geometric features. This ability to adapt to change is a key advantage of the bootstrap learning approach to robot systems.

2.2 Reinforcement Learning

2.2.1 Introduction

Reinforcement learning is a formalization of how agents learn to act in complex environments. A reinforcement learning agent's goal is to find an *optimal* policy that maximizes expected agent reward (or alternatively, minimizes the cost of acting for the agent). If the environment is completely observable, these types of problems can be described as Markov Decision Processes (MDP).

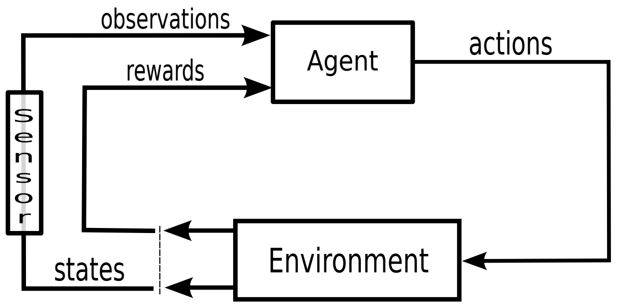


Figure 2.1: **Agent-Environment System**. An abstract agent-environment system as described by a Markov Decision Process consists of an agent that interacts with an environment that is fully observable. Actions result in changes in environmental state which are then observed by the agent.

2.2.2 Markov Decision Process

A Markov Decision Process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ consisting of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, a transition probability function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and a discount factor γ.

A policy is a function that, given a state, chooses an action for that state, e.g. $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The choice of action can be deterministic or probabilistic, in which case policies give a distribution over possible actions conditioned on the current state, $\pi : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$. In this case $\pi(a; s)$ denotes the probability of choosing action a when in state s when following policy π. A policy is considered optimal if it maximizes the expected discounted future reward. Formally, the expected discounted reward for a policy π at any state s_t is

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \tag{2.4}$$

where each subsequent state is drawn from a distribution that depends on the chosen action according to the policy and the transition probabilities and r_{t+k+1} is the reward given at state $t + k + 1$. An important related quantity is the action-value function, which gives the expected discounted future reward for state-action pairs

$$Q^{\pi}(s,a) = E_{\pi}\{R_t|s_t = s, a_t = a\} = E_{\pi}\{\sum_{k}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a\}. \tag{2.5}$$

An optimal policy solves a Markov Decision Problem by maximizing discounted, expected reward. The goal of reinforcement learning is to develop efficient algorithms for finding optimal policies.

2.2.3 Least Squares Policy Iteration

There are several different approaches to solving Markov Decision Problems. Some agents in later chapters will use Least Squares Policy Iteration (LSPI) to find an optimal policy [35]. LSPI is an actor-critic method for estimating the optimal policy over a linear feature space. It is an off-policy, model-free, sample-efficient method. LSPI depends on repeated application of Least Squares Temporal Difference Q-Learning (LSTDQ).

The state-action values for Q^{π} are the solution to a linear system of Bellman equations

$$Q^{\pi}(s,a) = \mathcal{R}(s,a) + \gamma \Sigma_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') \Sigma_{a' \in \mathcal{A}} \pi(a';s') Q^{\pi}(s',a'). \tag{2.6}$$

The system of Bellman equations can be concisely stated in matrix form

$$Q^{\pi} = \mathcal{R} + \gamma P \Pi_{\pi} Q^{\pi}, \tag{2.7}$$

where Q^{π} is a vector of size $|\mathcal{S}||\mathcal{A}|$ and its elements are expected values of discounted future rewards under policy π for each state-action pair. $\Pi_{\pi}(s,(s,a)) = \pi(a;s)$ is a matrix of size $|\mathcal{S}| \times |\mathcal{S}||\mathcal{A}|$ where $\pi(a;s)$ is the probability of taking action a in state s, and $P((s,a),s') = \mathcal{P}(s,a,s')$ represents the dynamics of the system in matrix form. P is of size $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|$.

Instead of representing an action-value function Q^{π} directly in terms of the existing state and action spaces, a mechanism of feature extraction is often introduced, where policies are represented as linear combinations of state-action features.

Consider a family of functions $\phi_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that maps states and actions to real values, and denote $\phi(s, a)$ as a vector of the form,

$$\phi(s,a) = \begin{pmatrix} \phi_0(s,a) \\ \phi_1(s,a) \\ \vdots \\ \phi_k(s,a) \end{pmatrix}. \tag{2.8}$$

Let Φ be a matrix of the form

$$\Phi = \begin{pmatrix} \phi(s_0, a_0)^T \\ \phi(s_1, a_0)^T \\ \vdots \\ \phi(s_{|\mathcal{S}|}, a_{|\mathcal{A}|})^T \end{pmatrix} \tag{2.9}$$

with size $|\mathcal{S}||\mathcal{A}| \times k$. The action-value function Q^π can be represented as the linear combination of these basis functions:

$$Q^\pi = \Phi \cdot w, \tag{2.10}$$

where w is a vector of weights of size k. Casting all the components of the Bellman equation in matrix form exposes the linear relationship between the action-value basis, the weight vector w, the rewards $\mathcal{R}$, the policy and model components Π_π and P. In matrix terms,

$$\Phi w = \mathcal{R} + \gamma P \Pi_\pi \Phi w. \tag{2.11}$$

Solving this linear system amounts to finding the fixed point solution for w given π. Collecting terms,

$$(\Phi - \gamma P \Pi_\pi \Phi) w = \mathcal{R}, \tag{2.12}$$

which is a linear system whose least squares solution is given by taking the Moore-Penrose pseudo-inverse of the matrix $\Phi - \gamma P \Pi_\pi \Phi$ [4]. This is the key to Least Squares Policy Iteration. For a given π, the algorithm approximates both the reward $\mathcal{R}$ and the matrix $\Phi - \gamma P \Pi_\pi \Phi$. The resulting linear system is then solved for w to produce an accurate action-value function in terms of the basis Φ.

With a proper choice of basis ϕ for representing the action value function, the density of matrix $\Phi - \gamma P \Pi_\pi \Phi$ can be tightly controlled. However, the matrix inverse is almost always dense. For large state spaces, inverting the matrix in Equation 2.12 would be impractical. Fortunately, it is almost never necessary to invert a matrix and the weights can be determined via a linear solver. Solvers perform quite well for sparse systems; therefore as long as the basis ϕ provides a sparse basis, the run time of LSTDQ is manageable.

Policy iteration involves an alternating process of evaluating the current policy and improving the policy. The current policy for each iteration of LSTDQ is parameterized by the current weight vector w. The next action with parameters w is given by $\pi(s) = \text{argmax}(w \cdot \phi(s, a))$. This is the action with the highest estimated value. With each new estimate of w (e.g. estimate of the approximate value function), the policy improves by choosing the maximum value relative to the new action-value estimates. So to perform policy iteration, LSTDQ only needs to be applied repeatedly until the action-value estimate converges.

It is worth noting that the choice of basis ϕ is important for the successful application of LSPI. The basis should provide a sparse representation of the value function while also being expressive enough to model the underlying MDP effectively. A key challenge in applying LSPI in developmental settings is that the input observations for MDPs are in the form of very high-dimensional sensors.

The *curse of dimensionality* refers to problems that arise as the dimension of a space increases. For example, 100 evenly spaced samples of a unit interval will have a resolution of 0.01. To achieve the sample spatial resolution from a sample set of a ten dimensional unit cube $[0, 1]^{10}$ would require 10^{20} samples [3]. Covering high-dimensional space requires the careful selection of a basis ϕ.

One effective approach is to map high-dimensional observations into a low-dimensional space using principal component analysis, and then using a radial basis function network to "featurize" the resulting principal component space, using a random subset of samples points as basis-function centers. Any method of dimensionality reduction may work. For example, sparse coding methods that mimic human visual processing [36] provide a similar framework for building basis function for representing value functions.

In Chapter 6, the policy that generates action traces for sensorimotor embedding uses principal component analysis and basis function networks to learn a policy based on simulated object images.

2.2.4 Tile Coding

Reinforcement learning in continuous domains usually requires that the learning agent implement some form of function approximation. For example, in the mountain car domain described later, the agent typically receives position and velocity information, which vary continuously. A value function $f(\text{position}, \text{velocity}) \rightarrow \text{value}$ should provide a continuous value estimate. Some form of approximation is required to model a continuous value function.

A Cerebellar Model Articulation Controller (CMAC) is one such approximation algorithm. This algorithm is sometimes referred to as tile coding. It is loosely based on the mammalian cerebellum. For inputs to a CMAC, $\text{CMAC}(x_1, \ldots, x_n)$ maps the vector $x_1, \ldots, x_n$ to k partially overlapping tiles [41]. Associated with each of these tiles is a weight, and the output of the CMAC function is the sum of all activated weights. In practice, tiles are built using a hash function specially designed to model overlapping tiles.

The tile weights can be trained using stochastic approximation, most commonly with the Robbins-Monro algorithm [4]. CMAC function approximation is often used conjunction with $SARSA(\lambda)$, an on-policy, online algorithm for solving reinforcement learning problems. In Chapter 8, the agent learns a continuous approximation of the value function using tile coding.

2.2.5 Related Work

In an area known as *representation learning*, manifold learning methods have been employed to form representations that accelerate reinforcement learning [38, 62, 19]. For example, manifolds have been used to decompose large continuous state spaces into topologically similar regions [62]. Value functions can be approximated in the image of charts, instead of over the original manifold, a change to the state space has been shown to be beneficial for learning policies.

Proto-value functions are another method of representing the state space for reinforcement learning [38]. Proto-value functions are built using Laplacian eigenmaps over the state transition graph to generate a low-dimensional basis for representing value functions during reinforcement learning. These proto-value functions make it easy for reinforcement learning to scale to very large state spaces.

Neighborhood component analysis (NCA) is another technique used to generate a low-dimensional sparse basis for reinforcement learning [19]. Maintaining a sparse representation of the state space is another approach to scaling reinforcement learning. This technique has shown promise in solving reinforcement learning problems on a mobile robot using vision [64].

Self-Organizing Distinctive State Abstractions (SODA) outlines an alternative to scaling reinforcement learning to high-dimensional, complex domains. One critical component of the SODA approach to early developmental problems or "high-diameter" task domains is the use of growing neural gas to separate the sensory experience of the agent into distinctive states. Like the earlier self-organizing map algorithm of Kohonen, Growing Neural Gas (GNG) quantizes the input space into a set of adaptive feature vectors [33, 18]. As new input is received, the GNG algorithm adapts existing feature vectors to decrease the matching error between inputs and existing feature vectors, adding new feature vectors as necessary. The result of GNG is a code book of feature vectors whose topological structure models the distribution of the observed data.

Predictive state representations (PSRs) are another method of representing state in reinforcement learning problems [60]. Predictive state representations ground state in statistics over observables, and represent state in terms of predictions of future observations. This method of representing states has been used to build models of cameras and manipulators, where PSRs allow a robotic agent to predict future depth observations based on sequences of motor commands [7].

2.2.6 Conclusion

Reinforcement learning is an important component of sensorimotor embedding. Learning policies that achieve perceptual goal states is the first step in sensorimotor embedding, and the result of sensorimotor embedding is a geometric representation of the state space. Like the related work presented here, sensorimotor embedding employs manifold learning methods to arrive at a representation of geometry.

2.3 Manifold Learning

2.3.1 Introduction

Manifold learning, also referred to as dimensionality reduction, is a form of unsupervised learning that seeks to generate low-dimensional data from high-dimensional data while preserving important properties of the data. Manifold learning has many applications, such as protein clustering, sensor localization, and face recognition [37, 5, 76]. For example, in ad hoc sensor networks, the topology and geometry of the network needs to be inferred from a large number of single-sensor observations. The goal is to construct a map of ad hoc sensors without any knowledge except the signals from the individual sensors. The aggregate sensor readings form a high-dimensional dataset, and the true coordinates of the sensors are the low-dimensional coordinates that are discoverable through manifold learning.

This section begins with a formal definition of a manifold, then continues with a detailed analysis of multidimensional scaling, which is an important component of sensorimotor embedding. The section concludes with an overview of two non-linear algorithms for manifold learning, and a review of related work.

2.3.2 Formal Manifold Definition

A manifold is a topological space that resembles Euclidean space near each point in the manifold. This idea is usually formalized using *charts*.

A *chart* for a topological space M is a homeomorphism φ from an open subset U of M to an open subset of Euclidean space. A chart is traditionally recorded as the ordered pair (U, φ). Charts are functions that map portions of manifolds to Euclidean space. These functions are sometimes called coordinate maps, since they associate coordinates with points on the manifold.

An *atlas* for a topological space M is a collection $\{(U_\alpha, \varphi_\alpha)\}$ of charts on M such that $\bigcup U_\alpha = M$. If the co-domain of each chart is the n-dimensional Euclidean space and the atlas is connected, then M is said to be an n-dimensional manifold.

For manifold learning problems, an n-dimensional manifold is usually embedded in a higher dimensional Euclidean space. The learning problem is to identify property-preserving charts that allow the manifold to be characterized (at least locally) as corresponding to a lower dimensional Euclidean space. Linear approaches, including multidimensional scaling, are described next.

2.3.3 Linear Methods

Principal component analysis (PCA) is a linear method of dimensionality reduction that projects high-dimensional data onto a basis in the order of the greatest amount of remaining data variability. PCA is often used in a wide variety of statistical, machine learning, and visualization contexts. Linear multidimensional scaling (MDS), which is similar to PCA, generates a low-dimensional dataset such that pairs of points have approximately the same metric relationships as the original high-dimensional points. The following derivation of multidimensional scaling is adapted from Krzanowski [34], and demonstrates that MDS is an optimal linear method.

For n points x_i in real p-dimensional space, let $X = [x_1\ x_2\ \dots\ x_n]$. Consider the matrix $K = X^T X$ whose entries are dot products $k_{ij} = x_i \cdot x_j$. K is known as a *Gram matrix*. Given K, X can be reconstructed through matrix decomposition. One approach is to use the singular value decomposition

$$M = U \Sigma V^T, \tag{2.13}$$

where U and V are unitary matrices and Σ is a non-negative real-valued diagonal matrix. The rows of U are called the left singular vectors of M. Similarly, the rows of V are known as the right singular vectors of M. The entries of Σ are the singular values of M.

The singular value decomposition is related to the eigen decomposition of a matrix. The left singular vectors are the eigenvectors of MM^T and the right singular vectors are the eigenvectors of $M^T M$. The singular values are the square roots of the eigenvalues of MM^T and $M^T M$. Consider the singular value decomposition of $X = U \Sigma V^T$. Then

$$K = X^T X = (V \Sigma U^T)(U \Sigma V^T) = V \Sigma^2 V^T. \tag{2.14}$$

For positive semi-definite matrices the eigen decomposition is the same as the singular value decomposition. So by finding the eigenvalue decomposition of K, it is possible to reconstruct X up to multiplication by a unitary matrix as,

$$K = V\Sigma V^T = (V\Sigma)(\Sigma V^T) = X^T X. \tag{2.15}$$

The key to multi dimensional scaling is to relate the matrix of squared inter-point distances Δ to the matrix K. Each entry d_{ij} in Δ is $\delta(x_i, x_j)$ for some distance metric δ. To do so requires the additional constraint that the points x_i have zero mean, $\sum_i x_i = 0$. This constraint can be expressed as $K1 = 0$ and $1K = 0$.

Entries of the distance matrix Δ have the form,

$$\delta_{ij} = ||x_i - x_j||^2 = x_i^T x_i - 2x_i^T x_j + x_j^T x_j = k_{ii} - 2k_{ij} + k_{jj} \tag{2.16}$$

where k_{ij} is the ij-entry in K and $||\cdot||$ is the metric space norm. Summing over the rows of Δ,

$$\sum_i \delta_{ij} = \sum_i k_{ii} - 2\sum_i k_{ij} + \sum_i k_{jj} = \text{trace}(K) + nk_{jj} \tag{2.17}$$

after applying the centering constraint $\sum_i k_{ij} = 0$. Similarly for column sums,

$$\sum_j \delta_{ij} = nk_{ii} + \text{trace}(K). \tag{2.18}$$

Finally the sum over all of the squared distances can be computed as,

$$\sum_i \sum_j \delta_{ij} = n \cdot \text{trace}(K) + n \cdot \text{trace}(K) \tag{2.19}$$

since $\sum_i \sum_j k_{ij} = 0$ by the centering constraint. For convenience $\delta_{\cdot j}$ denotes the sum over the rows of Δ. Similarly, $\delta_{i\cdot}$ and $\delta_{\cdot\cdot}$ denote the sums over the columns and all entries of Δ, respectively. From Equation 2.19,

$$\text{trace}(K) = \frac{\delta_{\cdot\cdot}}{2n}. \tag{2.20}$$

Solving for k_{ii} and k_{jj} in Equations 2.18 and 2.17 using Equation 2.20 to simplify,

$$k_{ii} = \frac{\delta_{\cdot j}}{n} - \frac{\delta_{\cdot\cdot}}{2n^2} \qquad k_{jj} = \frac{\delta_{i\cdot}}{n} - \frac{\delta_{\cdot\cdot}}{2n^2}. \tag{2.21}$$

Plugging these into Equation 2.16,

$$\delta_{ij} = \frac{\delta_{\cdot j}}{n} - \frac{\delta_{\cdot\cdot}}{2n^2} + \frac{\delta_{i\cdot}}{n} - \frac{\delta_{\cdot\cdot}}{2n^2} - 2k_{ij}. \tag{2.22}$$

Solving for k_{ij} gives

$$k_{ij} = -\frac{1}{2}(\delta_{ij} - \frac{\delta_{\cdot j}}{n} - \frac{\delta_{i\cdot}}{n} + \frac{\delta_{\cdot\cdot}}{n^2}). \tag{2.23}$$

Equation 2.23 is a recipe for constructing K in terms of the elements of Δ. Singular value decomposition reconstructs the original points x_i having the desired dimension. The use of singular value decomposition here allows for the selection of points with a fixed dimension that provides the best approximation of the original distances. This result is a consequence from the following key theorem [65].

Theorem (Eckart – Young) Consider the singular value decomposition $A = U\Sigma V^T$. Let $A_k = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T$ be the k-truncated SVD, with A_k a matrix of rank k. Among all matrices of rank k, A_k serves as the best approximation to A under the 2-norm, i.e. $||A - A_k||_2 \leq ||A - M_k||_2$ for any rank k matrix M_k.

In the case of multidimensional scaling, the amount of truncation determines the dimension of the reconstructed points, and according to the theorem, the related matrix of inter-point distances $\tilde{\Delta}$ will be the best approximation of the original matrix Δ under the 2-norm.

Multidimensional scaling is best suited to identifying low-dimensional linear representations of data given the inter-point distances of the high-dimensional data. For non-linear manifolds, however, the best linear embedding may still require many dimensions. Several non-linear methods of dimensionality reduction have been developed that generate representations of data using fewer dimensions for non-linear manifolds. Two algorithms, Isomap and Maximum Variance Unfolding (MVU)[79, 75], will be discussed next.

2.3.4 Isomap

The key insight that allows Isomap to find low-dimensional embeddings of non-linear manifolds is that distances between points are computed using a standard distance metric if points are close, and using shortest paths along the resulting graph structure if points are distant [75]. Computing distances between points “along the manifold” instead of using the distances inherited from the ambient space in which the manifold is embedded results in a lower dimensional representation of the data (Figure 2.3). Using specially constructed distances, rather than distances through the ambient space, is a key idea that motivates many approaches to manifold learning, including sensorimotor embedding.

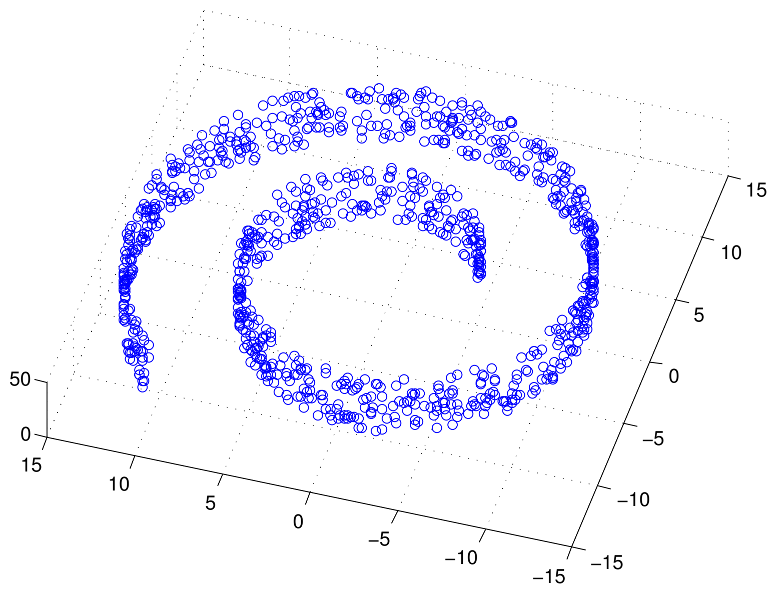


Figure 2.2: **Swiss Roll Example**. The data lies on a two dimensional nonlinear surface in three dimensional space. Non-linear manifold learning methods are needed to identify the two dimensional surface.

A great deal of the flexibility of Isomap stems from the fact that metrics for comparing data locally are often easier to specify than metrics that are valid over

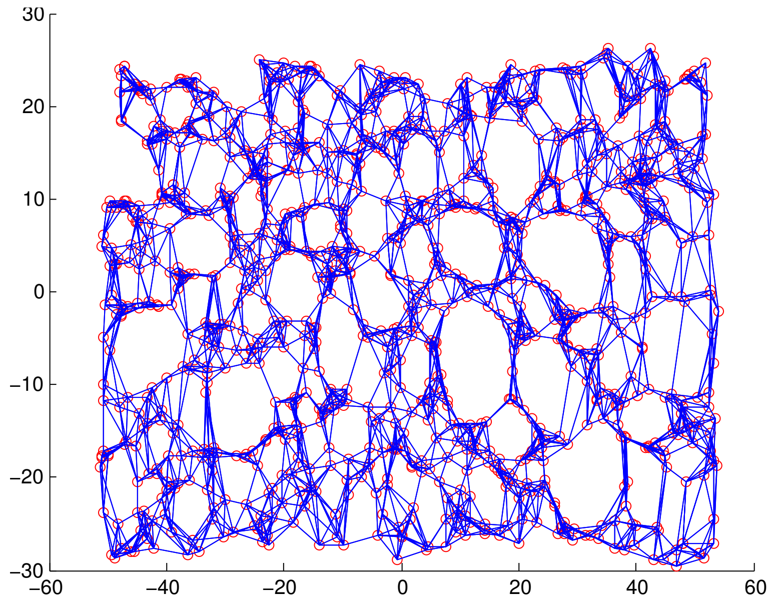


Figure 2.3: **Isomap Example**. This image shows the result of applying Isomap to the Swiss roll dataset, along with the neighborhood graph used to compute shortest paths. Isomap is able to expose the two-dimensional structure of the original data.

the whole data set. Local measurements are all that are required for generating low-dimensional representations of manifold data. As a practical matter, when deciding how to apply Isomap to any particular dataset, it is the responsibility of the designer to choose a method of computing the distance between any two points in the data set. If $\{x_i\}$ is the high-dimensional dataset, the desired distance function is given by $d(x_i, x_j)$. Local neighborhoods are then chosen either by specifying a limit ϵ such that any two points x_i and x_j are in the same neighborhood if $d(x_i, x_j) < \epsilon$, or by taking the k-nearest neighbors around every data point.

The missing entries in the resulting partial distance matrix Δ are then computed using shortest paths along the neighborhood graph. The produces a complete matrix of isometric inter-point distances Δ_{iso} which is then processed using MDS as described above. An embodied form of Isomap, that uses actions to identify local distances will be dicussed in related work and compared to sensorimotor embedding in Chapter 5.

2.3.5 Maximum Variance Unfolding

Maximum Variance Unfolding (MVU) takes a different approach to handling non-linear data [80]. The intuition is that maximizing the variance of the data subject to local constraints implicitly reduces the intrinsic dimension of the data. For example, a crumpled piece of paper is highly non-linear, but spreading the piece of paper out flat increases the variance of the data while preserving local distances along the paper's surface.

The first step in maximum variance unfolding is to form a semi-definite program that maximizes the variance of the data subject to the local distance constraints. Note that $\text{trace}(K) = \sum_i k_{ii} = \sum_i x_i^T x_i = ||x_i||^2$. So maximizing $\text{trace}(K)$ will produce an embedding that "pushes" all the points away from the origin. Maximizing the trace requires solving a semi-definite programming problem where the local measurements δ_{ij} are preserved [77]. As with Isomap, these local measurements can be chosen using an ϵ parameter or k-nearest neighbors. The semidefinite program for MVU is:

Maximize $\text{trace}(K)$ subject to

- $K \succeq 0$
- $\sum_{ij} k_{ij} = 0$
- $k_{ii} - 2k_{ij} + k_{jj} = \delta_{ij}$.

This semi-definite program can be optimized using any of the publicly available solvers [8, 73]. Eigenvectors for the resulting *Gram matrix* K provide the low-dimensional embedding of the original data points. Actual low-dimensional points are then generated through matrix decomposition as in multidimensional scaling.

MVU forms the basis of Action Respecting Embedding (ARE), an approach to learning geometry that incorporates action-based constrains into MVU. ARE will be discussed in related work.

2.3.6 Distance Functions

All manifold learning methods discussed above depend on identifying and preserving certain properties that relate data points, with the distance between data points as one such key property.

For formal manifolds, where the manifold is the domain of a set of chart functions, the distance between nearby points on the surface of the manifold can be determined using distance between the points in the image of the chart functions. Many innovations in manifold learning depend on modifying this mechanism. For example, Isomap is multidimensional scaling with a carefully chosen method of determining distance that uses the local graph structure of the data.

```python
def edit_distance(s, t):
    n = len(s) + 1
    m = len(t) + 1
    d = init_costs(n, m)  # initialize cost matrix
    for i in range(m):
        for j in range(n):
            if t[i-1] == s[j-1]:
                d[i][j] = d[i-1][j-1]
            else:
                d[i][j] = min(d[i-1][j] + 1, d[i][j-1] + 1, d[i-1][j-1] + 1)
    return d[m-1][n-1]  # return minimum cost
```

Algorithm 1: **Edit Distance**. The edit distance algorithm is a dynamic programming algorithm that determines the minimum number of edits required to transform one string into another.

For sensorimotor embedding, where sequences of actions are compared to determine the distance between anchor states, the method of comparison is important. For discrete action spaces, edit distance is a dynamic programming algorithm that can determine the distance between sequences of discrete actions. An implementation of edit distance is found in Algorithm 1. A useful feature of edit distance is that distances between individual actions do not need to be defined, only the cost of edits. This allows agents to compare sequences of discrete actions, even if the sequence length differs and the semantics of individual actions do not lend themselves to obvious methods of comparison.

2.3.7 Related Work

Manifold learning has long been considered a useful tool for understanding, processing, and simplifying complex state spaces prior to applying other learning methods. Manifold learning can also be used directly to acquire knowledge about the environment. For example, Action Respecting Embedding (ARE) is a manifold learning approach that maps raw sensory experience to a low-dimensional manifold in a way that obeys the constraints imposed by agent actions [9]. This method uses a variation of Maximum Variance Unfolding (MVU) that adds additional convex constraints to the semi-definite program that lies at the heart of MVU.

In action respecting embedding, data points that are connected by an action must be neighbors. Moreover, the distance (in sensory space) between any two points connected by an action defines a non-uniform neighborhood. The parameter for adjusting neighborhood size in action respecting embedding is the number of actions T that connect any two data points. In typical applications of MVU, neighborhood size is either set to some constant distance ϵ or k nearest-neighbors.

Formally, given an action window of size T the neighborhood adjacency matrix η is constructed such that

$$\begin{aligned} \eta_{ij} = 1 \iff & \exists k, l \text{ such that} \\ & |k - i| < T, |l - j| < T, \\ & ||x_i - x_k|| > ||x_i - x_j|| \text{ and} \\ & ||x_j - x_l|| > ||x_i - x_j||. \end{aligned}$$

In other words, x_i and x_j are neighbors if $||x_i - x_j||$ is contained within the largest distance between x_i and x_j and any of the data points within T actions of either.

In addition to non-uniform neighborhood sizes inferred from action relationships between data points, action respecting embedding encodes the intuitive idea that if at points i and j an agent takes the same action a, then that action will not affect the resulting distance between subsequent points. In other words, $||x_i - x_j|| \approx ||x_{i+1} - x_{j+1}||$. Encoding this intuition as a constraint for a semi-definite program requires that

$$\forall i, j \text{ where } a_i = a_j \text{ then } k_{(i+1)(i+1)} - 2k_{(i+1)(j+1)} + k_{(j+1)(j+1)} = k_{ii} - 2k_{ij} + k_{jj}.$$

The ARE algorithm then proceeds by using convex optimization to solve a semi-definite program of the following form:

Maximize trace(K) subject to

- $K \succeq 0$
- $\sum_{ij} k_{ij} = 0$
- $k_{ii} - 2k_{ij} + k_{jj} \leq ||x_i - x_j|| \ \forall i, j$ such that $n_{ij} > 0$
- $\forall i, j$ where $a_i = a_j$ then $k_{(i+1)(i+1)} - 2k_{(i+1)(j+1)} + k_{(j+1)(j+1)} = k_{ii} - 2k_{ij} + k_{jj}$.

The resulting matrix K can be decomposed into eigenvectors that form the basis of the low-dimensional representation of data points as in MDS. Figure 2.4 shows the results of action-respecting embedding applied to a simple *roving eye* domain. This domain is an important target for both action respecting embedding and sensorimotor embedding. The domain simulates a viewing window traversing a much larger image. The agent actions move the viewing window across the image, and the agent can perceive only the part of the underlying image that is currently inside the viewing window. This approach to embodied dimensionality reduction assumes a discrete (and small) set of available actions. In the sensorimotor embedding action spaces are not required to be discrete.

Another application area of manifold learning in developmental robotics is in learning the organization of sensors. For example, manifold learning has been used

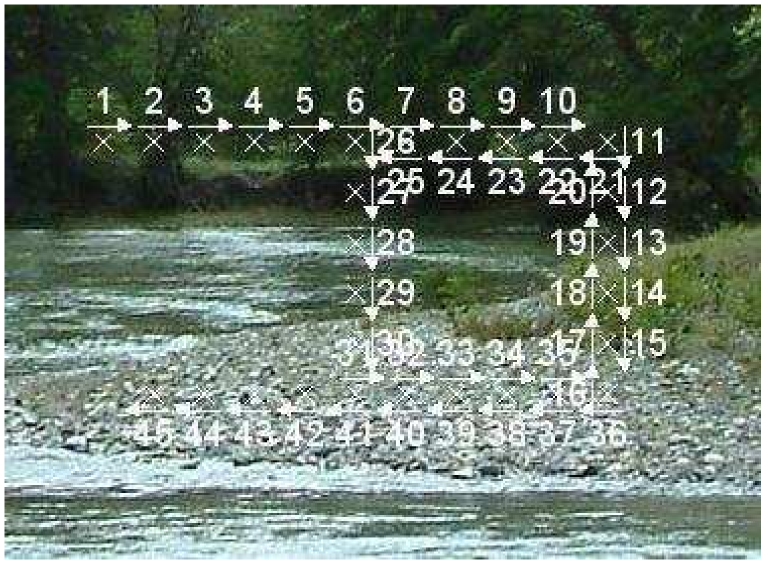


Figure 2.4: **Action Respecting Embedding**. Action respecting embedding can reconstruct paths in a “roving eye” visual navigation domain [10]. The sensorimotor embedding approach to this problem is presented in Chapter 5.

to organize sense elements and learn an abstract low-dimensional motor interface [54]. After sensor and motor geometry were learned, the agent tracked statistical features to construct reasonable primitive sensorimotor behaviors. This work was later expanded to mobile robots with vision [45]. Sensor organization was later explored using non-linear manifold learning methods [42]. Procrustes analysis (see Chapter 3) applied to non-linear dimensionality reduction methods in a variety of embodied and sensor reconstruction domains was used for comparing the quality of different approaches [69, 16]. That work included one of the only explorations of the effect of control policies on manifold learning in embodied domains.

Manifold learning has also been applied to both sensor and action spaces in a way that attempts to take advantage of the interaction between action and sensor spaces [50, 51]. A modified form of Isomap was developed that computes distances between sensory signals using either

- $d(s_i, s_j) = ||a||$ where a is an action that takes the agent from s_i to s_j or

- the shortest path along the resulting neighborhood graph.

Embodied Isomap does not always successfully represent sensorimotor data using as few dimensions as the sensorimotor embedding approach (Chapter 5).

2.3.8 Conclusion

Sensorimotor embedding is also a manifold learning algorithm, whose input is the high-dimensional sensory experience of a developing robot and whose output is a low dimensional representation of state geometry. Action respecting embedding and embodied Isomap are two competing approaches to sensorimotor embedding. Like sensorimotor embedding, these methods use information about agent actions to augment existing methods of manifold learning. Sensorimotor embedding has better run time properties than action respecting embedding, and generates more faithful representations of geometry than either action respecting embedding or embodied Isomap in some domains (Chapter 5).

2.4 Conclusion

The research areas and related work in this chapter span developmental robotics, manifold learning, and reinforcement learning. Like sensorimotor embedding, all of the results mentioned tackle some aspect "blooming buzzing confusion" of the senses through learning and development. The next chapter will describe sensorimotor embedding in detail. Subsequent chapters will evaluate sensorimotor embedding in several different domains.

Chapter 3

Sensorimotor Embedding

"...the apodeictic certainty of all geometrical propositions, and the possibility of their a priori construction, is grounded in this a priori necessity of space." –I. Kant *Critique of Pure Reason*

Sensorimotor embedding is an algorithm for learning geometry. Agents that sense and act can use sensorimotor embedding to learn about the geometry of the environment. Sensorimotor embedding is a developmental algorithm that facilitates open-ended development for diverse agent architectures. In this chapter, sensorimotor embedding will be described in detail along with methods for evaluating sensorimotor embedding.

3.1 Formal Definition

Sensorimotor embedding is a *manifold learning algorithm.* For an overview of manifold learning, see Section 2.3 in Chapter 2. To review, a manifold learning algorithm transforms high-dimensional data into low-dimensional data. Sensorimotor embedding transforms high-dimensional sensory or state data into low-dimensional data.

Sensorimotor embedding allows an agent to *learn geometry.* To learn geometry, an agent must learn a set of coordinates M and a metric defined over that set $\delta : M \times M \to \mathbb{R}$. The pair (M, δ) is a metric space if δ satisfies the following properties:

- $\delta(x, y) \geq 0$ (non-negative);
- $\delta(x, y) = 0$ if and only if $x = y$ (identity of indiscernibles);
- $\delta(x, y) = \delta(y, x)$ (symmetry);
- $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$ (triangle inequality).

The geometry the agent learns must be *grounded.* A geometry is grounded if there exists an external world property that can be described as a metric space (Y, δ') such that, for external world states y_i and y_j, the agent knows of corresponding learned states x_i and x_j such that $\delta(x_i, x_j) = \delta'(y_i, y_j)$. This grounding must be reasonable, in that the external world property described by (Y, δ') is related in some way to the metric space learned by the agent. If so, then the agent has learned a *grounded geometry.* In practice, the nature of this relationship is known to the experimenter beforehand. In nature, these grounded external world properties are how an agent can construct knowledge of the world.

To learn geometry, sensorimotor embedding proceeds in four steps:

1. The agent first learns an optimal policy.
2. For each start state, the agent applies the learned policy and records the actions as a single *action trace.*
3. The agent computes the distances between each of the recorded *action traces* forming a distance matrix.
4. The agent performs multidimensional scaling on the distance matrix, generating a set of real-valued, low-dimensional points corresponding to each start state.

3.1.1 Learning a Policy

Learning a policy is the first step of sensorimotor embedding. As described in Chapter 2 Section 2.2, *policies* are functions from a set of states to a set of actions,

$$\pi : \mathcal{S} \rightarrow \mathcal{A}.$$

Policies are functions that describe an agent's moment-by-moment decision making and are trained with respect to a reward function. For the learning tasks considered in this thesis, natural perceptual goals are present as part of the task. The relevant geometry is best understood using action traces that drive the agent to these goal states.

The reinforcement learning problems that an agent seeks to solve at this stage are not general reinforcement problems with arbitrary reward signals. In a typical application, the agent receives reward only when a perceptual goal state is reached. Chapter 9 will include some areas of future work that apply sensorimotor embedding in more general settings for the purposes of visualization and option discovery.

3.1.2 Action Traces

Action traces are sequences of actions that are the result of an agent applying a policy. In formal terms, for a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ from states to actions, and a deterministic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, an action trace is the sequence of actions produced by iteratively applying the policy and the transition function. For example, one application results in $\pi(s_1) = a_1$ and $T(s_1, a_1) = s_2$ and a subsequent application $\pi(s_2) = a_2$ and $T(s_2, a_2) = s_3$. Repeated application of the policy and transition function results in sequence $\{s_t, a_t\}_{t=1}^{\infty}$. The action component of this sequence, $\{a_t\}_{t=1}^{\infty}$, is the action trace. For notational convenience, the index term will be omitted when not needed, so $\{a_t\}_{t=1}^{n}$ will be written as $\{a\}_1^n$.

The agent associates the initial start state with the action trace that follows, e.g. $s_1 \leftrightarrow \{a\}_1^{\infty}$. Under this general definition, action traces do not necessarily terminate. The experimental work in the following chapters uses finite action traces. A finite action trace is the result of a terminating policy that ends when the agent transitions to a terminal or goal state.

The transition function presented here is deterministic. Chapter 9 will include a discussion of how to extend these results to non-deterministic transition functions and non-deterministic policies. In brief, the move to non-determinism means that policies with the same initial conditions can generate different action traces. Future work will explore using the sample mode over generated action traces for action trace comparisons.

3.1.3 Comparing Action Traces

To apply sensorimotor embedding, action traces must be comparable. Many common actions can be parameterized using real valued vectors. For example, the motors on a robot can be controlled by specifying sequences of motor torques. If higher level controllers are available, then the action space can be parameterized using velocities or positions. In each of these cases, the individual action parameters are real valued vectors that are part of a Euclidean metric space. If the action space $\mathcal{A}$ is a metric space with a metric d, then action traces $\{a\}_1^n$ and $\{b\}_1^m$ where $n > m$ can be compared by first appending the shorter of the two traces with zero actions so they are of equal length. Then the distance between the traces can be calculated using

$$\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) = \sum_{t=1}^{m} d(a_t, b_t) + \sum_{t=m+1}^{n} d(a_t, 0). \qquad (3.1)$$

With this definition, δ_{trace} is a metric since it satisfies all the properties of a metric. δ_{trace} is non-negative since each component in the sum is non-negative. $\delta_{\text{trace}}(\{a\}_1^n, \{a\}_1^n) = 0$ for two identical traces since $d(a_t, a_t) = 0$ for each component of Equation 3.1. $\delta_{\text{trace}}(\{a\}_1^n, \{b\}_1^m) > 0$ when traces $\{a\}_1^n$ and $\{b\}_1^m$ differ since

$a_t \neq b_t$ for at least one value of t after zero-padding and therefore $d(a_t, b_t) > 0$ for at least one component in Equation 3.1. The function δ_{trace} is also symmetric since each component in the sum is symmetric.

δ_{trace} satisfies the triangle equality since for three traces $\{a\}_1^m$, $\{b\}_1^n$, $\{c\}_1^l$, where without loss of generality $m \leq n \leq l$, the sum $\delta_{\text{trace}}(\{a\}_1^m, \{b\}_1^n) + \delta_{\text{trace}}(\{b\}_1^n, \{c\}_1^l)$ can be written as

$$\sum_{t=1}^{m} d(a_t, b_t) + \sum_{t=m+1}^{n} d(0, b_t) + \sum_{t=1}^{n} d(b_t, c_t) + \sum_{t=n+1}^{l} d(0, c_t)$$

which after reordering terms can be written as

$$\sum_{t=1}^{m}(d(a_t, b_t) + d(b_t, c_t)) + \sum_{t=m+1}^{n} (d(0, b_t) + d(b_t, c_t)) + \sum_{t=n+1}^{l} d(0, c_t) \tag{3.2}$$

By the properties of d, $d(a_t, b_t) + d(b_t, c_t) \geq d(a_t, c_t)$ and $d(0, b_t) + d(b_t, c_t) \geq d(0, c_t)$. Since every term in $\delta_{\text{trace}}(\{a\}_1^m, \{c\}_1^l)$ is less than or equal to the corresponding term in (3.2), the sum of all the terms is less than or equal to $\delta_{\text{trace}}(\{a\}_1^m, \{b\}_1^n) + \delta_{\text{trace}}(\{b\}_1^n, \{c\}_1^l)$ and δ_{trace} satisfies the triangle inequality.

Zero actions can also be prepended instead of appended to action traces. The metric properties of this alternate distance function remain unchanged. Prepending zero actions may bring similar actions into better alignment in certain domains. Dynamic time warping provides another potential approach for comparing sequences of continuous actions, though with standard dynamic time warping, the metric properties of the trace comparison are lost [43].

If the space of actions is discrete, then sequences of actions can be compared using edit distance. Edit distance is described in detail in Section 2.3.6. Action traces where each action comes from a discrete action space can be represented as a sequence of symbols. Edit distance will find the smallest number of edits required to transform one sequence of symbols into another sequence of symbols. If turning one sequence of actions into another one requires few edits, those sequences will have a small edit distance. If turning one sequence into another requires many edits, the two sequences will have higher edit distance. Like δ_{trace} described above, edit distance is a metric over symbol sequences [44].

3.1.4 Applying Multidimensional Scaling

For a finite set of action traces, an agent computes all distances between action traces. Since each action trace is associated with a start state, $s_1 \leftrightarrow \{a\}_1^n$, the agent can infer that distances between action traces are also distances between start states. Distances between action traces can be computed using Equation 3.1, or if $\mathcal{A}$ is discrete, using edit distance. Then a set of low-dimensional points can be generated using multidimensional scaling that have approximately the same distance relationships. In the final step of the algorithm, the agent associates the original start states with these low-dimensional points, and infers that the distance between start states is the distance between these low-dimensional points. These together provide a set of euclidean coordinates, e.g. $x, y \in M$, associated with the set of start states. For a given policy π, the distance between start states is equivalent to the distance between the resulting action traces. This equivalence has the form:

$$\delta_\pi(s,t) \equiv \delta_{\text{trace}}(\{a\}_1^m, \{b\}_1^n) \approx ||x - y||$$

where start states s and s' typically have very high dimension, $\{a\}$ and $\{b\}$ are action traces that result from applying a policy π, and x and y are low-dimensional coordinates produced by multidimensional scaling. Note that δ_π inherits the following properties from the action trace metric δ_{trace}:

- $\delta_\pi(s,t) \geq 0$ (non-negativity);
- $\delta_\pi(s,s) = 0$ if $s = s$;
- $\delta_\pi(s,t) = \delta_\pi(t,s)$ (symmetry);

- $\delta_\pi(s,u) \leq \delta_\pi(s,t) + \delta_\pi(t,u)$ (triangle inequality).

Unlike a true metric, $\delta_\pi(s,t)$ may be zero for some sensory states s and t that differ but result in the same sequence of actions, so δ_π is a pseudo or semi-metric [11]. Chapter 7 contains an example of aliasing caused by identical action traces. In cases where π produces unique action traces for each start state, δ_π is a metric.

The central claim of this thesis, is that these low-dimensional points and the distances between them can be used to represent external world geometric properties of the corresponding sensor states. This claim will be evaluated experimentally over several chapters using the evaluation methods discussed in the next section.

3.2 Evaluating Sensorimotor Embedding

As shown above, sensorimotor embedding learns geometry, since the result of sensorimotor embedding is a set of points with a metric. For sensorimotor embedding to be useful, what is learned has to also be grounded, meaning that it has to model useful external world properties. The central claim of this thesis is that sensorimotor embedding generates low-dimensional points and a distance function that reflects important geometric properties of the external world. Several approaches to empirical evaluation of this claim are presented in this section.

3.2.1 Procrustes Analysis

A robust comparison of geometry in the environment with the result of sensorimotor embedding is possible using *Procrustes analysis.* Procrustes analysis compares two sets of sampled points, by normalizing all points in two sets so that the mean of the points is zero, scaling all points so that the root mean squared distance of all the points from the origin is one, and removing any difference in rotation between two point sets. The result of this analysis is a measure of the distance between the two point sets, computed as the residual error after correcting for translations, scale changes, and rotations.

Centering each point set removes differences in translation. The first step of Procrustes analysis computes the mean of two sets of corresponding points, $((x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k))$ where x_i and y_i are points int $\mathbb{R}^n$. The mean of these points is $(\bar{x}, \bar{y})$ where

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_k}{k}, \quad \bar{y} = \frac{y_1 + y_2 + \cdots + y_k}{k}. \tag{3.3}$$

These points are translated so that their mean is at the origin, $(x_i, y_i) \to (x_i - \bar{x}, y_i - \bar{y})$. For two corresponding point sets let,

$$\bar{X} = \begin{bmatrix} x_1 - \bar{x} \\ x_2 - \bar{x} \\ \vdots \\ x_k - \bar{x} \end{bmatrix}, \bar{Y} = \begin{bmatrix} y_1 - \bar{y} \\ y_2 - \bar{y} \\ \vdots \\ y_k - \bar{y} \end{bmatrix}. \tag{3.4}$$

The Frobenius norm is given by,

$$||A||_F = \sqrt{\text{trace}(A^* A)} = \sqrt{\sum_{i=1}^{k} \sum_{j=1}^{n} |a_{ij}|^2}, \tag{3.5}$$

where A^* denotes the conjugate transpose of A. Next the scale component is removed so that the Frobenius norm for X and Y are both one. The scale is given by,

$$s_x = ||\bar{X}||_F \tag{3.6}$$

$$s_y = ||\bar{Y}||_F. \tag{3.7}$$

The point coordinates are divided element-wise by their initial scale

$$(x_i, y_i) \to ((x_i - \bar{x})/s_x, (y_i - \bar{y})/s_y)). \tag{3.8}$$

The next step of Procrustes analysis finds the rotation of one point set that minimizes the sum of squared distance between the corresponding points. For the translated and scaled point sets, let

$$X = \begin{bmatrix} (x_1 - \bar{x})/s_x \\ (x_2 - \bar{x})/s_x \\ \vdots \\ (x_k - \bar{x})/s_x \end{bmatrix}, Y = \begin{bmatrix} (y_1 - \bar{y})/s_y \\ (y_2 - \bar{y})/s_y \\ \vdots \\ (y_k - \bar{y})/s_y \end{bmatrix}. \tag{3.9}$$

To find the best rotation matrix requires solving the orthogonal Procrustes problem, $R = \text{argmin}_\Omega \; ||X\Omega - Y||_F$, subject to the constraint $\Omega^T\Omega = I$ [22]. The optimal solution can be found by applying singular value decomposition (SVD) to X^TY. So if $M = X^TY$ then applying SVD to M gives $M = U\Sigma V^*$. And the solution for R is UV^*. This result is proved in [57].

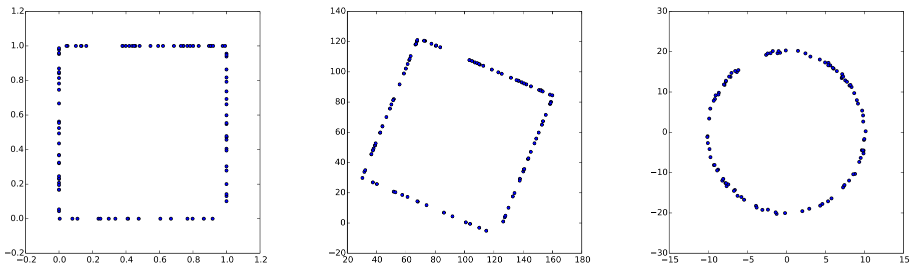
After translation, scaling, and rotating the corresponding point sets, the remaining sum of squared distance between the point sets is given by

$$\text{Procrustes Error} = ||XR - Y||_F. \tag{3.10}$$

This quantity is also known as Procrustes distance, and measures any remaining differences between the corresponding point sets that cannot be explained by translation, scale, or rotation.

Comparing shape is an important problem in many fields, and comparing shapes from sampled points is a critical issue in fields as diverse as biology and anthropology. Procrustes analysis, named after the Greek innkeeper who guaranteed the perfect bed for every guest, not by changing the size of the bed but by (ruthlessly) changing the size of the guest, is a generally accepted statistical method of performing this analysis.

As is evident in Figure 3.1, Procrustes error is low when the difference between point sets is explained by changes in translation, scale and rotation. Nonlinear changes and noise increase Procrustes error. This method can be used to evaluate manifold learning methods, including sensorimotor embedding, by comparing a ground truth sample set of points to an associated set of points generated via



(a) Shape Template (b) Linear Change (c) Non-Linear Change

Figure 3.1: **Procrustes Error Example**. On the left sample points are taken from a basic shape. The samples are then linearly transformed in the center image. In the right image, a non-linear transformation is applied to the sample points, transforming the points into a circle. Procrustes error is measured between the transformed and template samples. The Procrustes error for the center image is less than 6e-6. The Procrustes error for the right image is 0.12. Procrustes error increases if the transformation of the underlying points is non-linear. A low Procrustes error between ground truth samples and transformed samples generated via manifold learning indicates that the manifold learning method preserves linear relationships among the sample points.

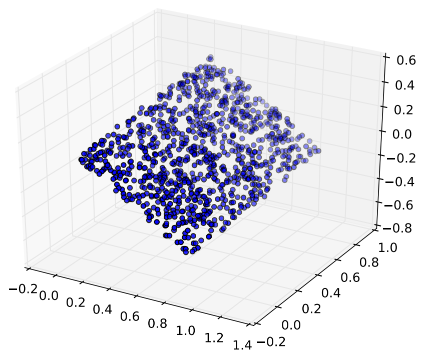
manifold learning [71]. If the point comparisons are faithful up to a linear transformation, then the Procrustes error will be low. Intuitively, ground truth shapes in the environment will correspond to the same shapes in the point set generated by manifold learning.

3.2.2 Feature Utility

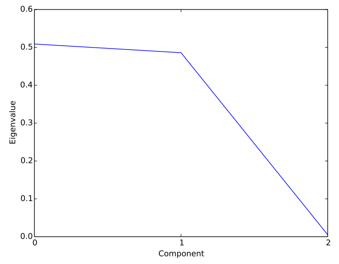
Another approach to evaluating sensorimotor embedding is to attempt to use the output as a feature for learning a task. If sensorimotor embedding learns a grounded geometry, then tasks which depend on knowing that grounded geometry should be easy to learn. Using this method of validation requires an agent learn the task using ground truth information, followed by an agent learning the task using features based on sensorimotor embedding. If the agent with access to ground truth information can learn the task, and if the agent using sensorimotor embedding can also learn the task, then the features that result from applying sensorimotor embedding enable learning in cases where ground truth information is not available to the agent. This method of validation also indicates that, as a component of a developmental program, the output of sensorimotor embedding can support other learning tasks.

3.2.3 Distribution of Eigenvalues

The distribution of eigenvalues is another method of evaluation that is widely used in manifold learning literature. For methods of manifold learning that involve eigendecomposition of a matrix, the resulting eigenvalue magnitudes represent the amount of variation explained by each eigenvector in the decomposition. For example, if the original data lies on a two dimensional plane in higher dimensional space, then the data only varies in two dimensions and principal component analysis should result in two non-zero eigenvalues corresponding to the two dimensions of variation in the original dataset (Figure 3.2).



(a) A Plane in 3D



(b) A Scree Plot

Figure 3.2: **Scree Plot**. The plot on the left shows a set of points that lie on a plane in three dimensional space. Using PCA, these points can be transformed into points in a two dimensional space. The magnitude of eigenvalues associated with each component indicate how much variation is explained by each component. Manifold learning methods seek to explain data in the fewest number of component dimensions. In cases where eigenvalues are generated, these provide an indication of how many components are needed to model the data.

Manifold learning methods seek to explain the data in the fewest number of component dimensions. When eigenvalues are generated, the magnitudes indicate how many components are needed to model the original data. Having a small number of concentrated eigenvalues is so important that some approaches to manifold learning, like maximum variance unfolding, attempt to minimize the number of non-zero eigenvalues directly (see Chapter 2 Section 2.3.5 for more details). A plot of the distribution of eigenvalues is known as a scree plot. Scree plots provide visual evidence that the distribution over eigenvalues is concentrated.

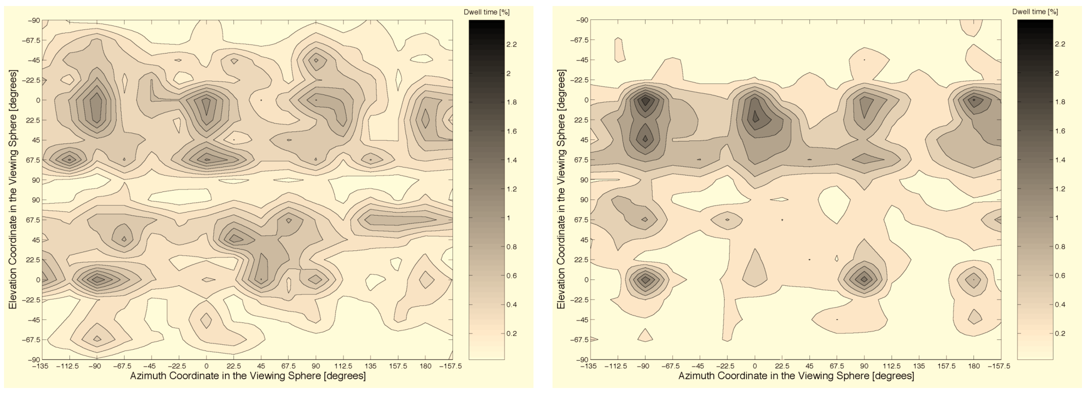
3.3 Discussion

Learning a policy is a crucial component of sensorimotor embedding. As mentioned previously, policies are functions from states to actions. Maximization of expected future reward is the criterion used to select policies, and policies are trained with respect to a reward function. In the experiments presented in subsequent chapters, the reward function is based on simple perceptual goals.

Studies show that humans have preferences that can be interpreted as perceptual goals. For human subjects not all perceptual states are considered equal, and people behave so as to bring about preferred perceptual states. For example, humans prefer certain perspectives on objects as shown in studies of adults [31] and infants [49]. Self-generated view preferences include:

- planar views over 3/4 views,
- flat surfaces aligned with line of sight, and
- upright orientation of objects with respect to gravity.

Not only are self-generated object views in adults biased towards planar views, this bias actually results in faster object recognition [23, 31]. Passive viewing of



(a) 12-18 Months

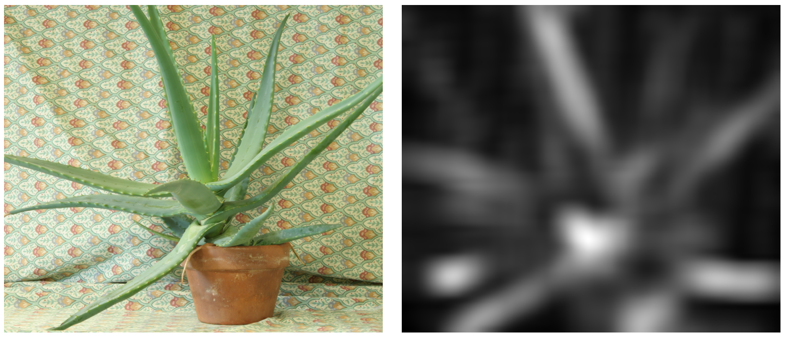
(b) 30-26 Months

Figure 3.3: **Development of Viewpoint Bias**. The left plot shows dwell times as a function of orientation for infants ages 12-18 months. The right plot shows dwell times as a function of orientation for infants ages 30-36 months. These plots demonstrate a perceptual biase in preferred orientation that strengthens with age [49].

objects where the views are biased significantly decreases in recognition time over unbiased passive viewing, and the ability to generate views by manipulating objects results in even faster object recognition.

Studies of dwell times for infants of increasing age and show that the bias exists in the youngest infants tested but increases with age (Figure 3.3) [49]. Dwell times are the amount of time subjects spend observing an object from a particular perspective. Non-biased self-generated views would result in uniformly distributed dwell times and uniform dwell time plots (which average over multiple subjects). Data shows that both infants and adults spend more time on particular perspectives. The evidence indicates that in humans, certain perceptual goal states are preferred, and manipulation policies favor these perceptual states.

Another source of perceptual goals is visual saliency. A basic saliency model



(a) Normal Image (b) Saliency Map

Figure 3.4: **Visual Saliency Example**. A saliency map of a plant using the method of Itti and Koch [30]. This uses only features present in the image to drive visual attention. Such bottom up features have proven to be surprisingly effective at modeling visual attention. The effectiveness of saliency maps provides evidence of perceptual goal states that can be formed using only bottom-up features.

built using only "bottom up" features was found to be very predictive of visual attention [29]. An example of these saliency maps is shown in Figure 3.4. The Itti and Koch model of saliency only used features present in the image, not any specific task indicators, to develop an accurate model of visual attention for human subjects.

One unifying theme in these psychological findings is that the perceptual goals that drive actions can be based on simple low level sensory features, indicating that simple perceptual goals tend to serve a role very similar to reflex actions in human development. Since both full sensory understanding and motor control systems are very complex, a developmental approach bootstraps using simple reflexes and simple perceptions. Some of these perceptual targets may be available very early on in development. For example, infants as young as two days old have shown a preference for mutual gaze, indicating that infants at this early age are already able to identify faces as perceptual targets. For this thesis, the important point is that perceptual goal states based on bottom-up features exist in natural systems and so are a reasonable reasonable prerequisite for sensorimotor embedding.

For reward functions that provide positive reward only at perceptual goal states, there is a useful special case of policies called *ballistic policies*. A policy, π, is a ballistic policy if $\pi(s)$ results in an action that takes an agent immediately to a perceptual goal state. In other words, each action trace has length one. If the agent can learn a ballistic policy, then the agent can associate with each sensor signal an action space coordinate corresponding to the ballistic policy. The agent can then infer geometry in the action space directly.

3.4 Conclusion

Sensorimotor embedding uses a learned policy to associate an action trace with each start state. These action traces are then compared using a metric. Dimensionality reduction is applied to a set of action trace comparisons. The result of dimensionality reduction is a set of low-dimensional points that represent, for certain tasks, the geometric relationships among the start states of each action trace. The quality of this representation can be evaluated using Procrustes analysis, feature utility, and by examining the distribution of eigenvalues. The following chapters will show how sensorimotor embedding can be applied to learn geometric features in several different domains.

Chapter 4

Learning the Geometry of the Foveated Retina

The human vision system has two important properties: Retinas are foveated and eyes have the ability to saccade. These two properties are sufficient to simultaneously learn the structure of receptive fields in the retina and a saccade policy that centers the fovea on points of interest in a scene. *Sensorimotor embedding* is applied in this domain to learn the receptive field structure. The results are evaluated using a roving eye robot on synthetic and natural scenes, and physical pan/tilt camera. In each case learned geometry is compared to actual geometry, and the learned motor policy is compared to the optimal motor policy. In both the simulated roving eye experiments and the physical pan/tilt camera, *sensorimotor embedding* is able to learn both an approximate sensor map and an effective saccade policy.

4.1 Motivation

In the human eye, the retina is a non-uniform array of photoreceptive rod and cone cells. The human retina has a foveal pit, a single region of maximum density of cone photoreceptors. In addition, a human can change the location of the retina relative to a scene through ballistic actions known as saccades [48]. The combination of a small, high-resolution fovea with the ability to saccade to regions of interest is an economical strategy for both humans and robots to achieve high-resolution vision across large fields of view.

Gathering and interpreting visual information requires a *motor map* and a *sensor map* of the retina. The motor map encodes the motor commands necessary to move the eye to new locations in the visual scene and is used to generate saccades. The sensor map represents the geometric structure of the retina, specifically the positions of sense elements within the sensor array, and can be used to perform geometric operations on the visual signal such as edge detection. By exploiting the relationship between motor commands and sensor geometry, an autonomous agent with foveated vision can *simultaneously* learn both the motor and sensor maps.

For simple sensors, these maps can be manually specified, but as sensors become more complex and adaptive, learning approaches are of increasing value to robotics. In addition, as lifetimes of autonomous robots increase, the robust nature of this developmental approach will allow robots to adapt to changing sensors and motors. In previous work on learning motor maps for saccades, the learning was driven by the two-dimensional difference between the pre-saccadic and post-saccadic position of a target on the retina. These models assume that the structure of the retina is known when learning the motor map, allowing calculation of the distance between a target and the fovea [47, 55].

Sensorimotor embedding is appropriate for cases with an easily identifiable reward signal (e.g. activation), linear ballistic motor commands, and a high number of sense elements. The algorithm exploits the structure of the sensorimotor domain to produce an explicit mapping between motor commands and sensor features. This map has two interpretations: as a primitive behavior that maximizes reward (the policy interpretation), and as a structure for the sensor array (the geometric interpretation).

This work presented here demonstrates that sensorimotor embedding can learn sensor structure from sensorimotor experience. The developmental nature of sensorimotor embedding allows an agent to simultaneously adapt both geometry and policy to changes in the physical model of the retina. The agent demonstrates adaption in the case of retinal lesioning and vision reversal.

4.2 A Foveated Retina Model

The abstract model of the foveated retina is inspired by the anatomy of the human retina. In this model, a retina is a collection of receptive fields, or sense elements, with fixed geometry arrayed across a two dimensional surface. Each receptive field responds to sensory input from a portion of an image or scene according to its own activation function. The learning rule requires that the distribution of activations across the retina be non-uniform and achieve a single maximum at the fovea. In addition, under this model, ballistic motions instantaneously change the location of the retina in an image or scene.

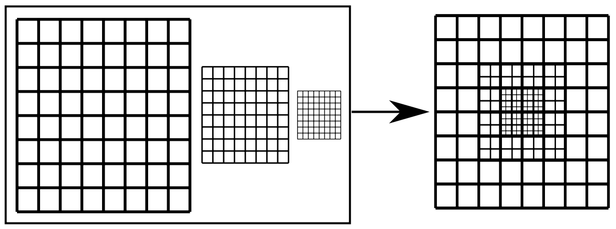


Figure 4.1: **Fovea Model**. This implementation of the fovea consists of overlapping layers of receptive fields. As the layer resolution increases, the extent of each receptive field decreases, and the number of bits necessary to describe the layer state remains constant. This model is useful since the fovea, as a region of high resolution, provides a natural perceptual target for the agent.

In this implementation, the learning agent has a foveated retina with N layers of receptive fields (Figure 4.1). Each layer has receptive fields of uniform extent and resolution. Layers with higher resolution and smaller extent overlap layers with lower resolution in the center of the retinal field of view. The fovea is the region with the highest concentration of overlapping receptive fields, and is also the region of maximal activation, so this implementation satisfies the model assumptions specified above. Alternative implementations satisfying the model assumptions should behave similarly.

The implementation of each individual receptive field may also vary. In this case, each receptive field must map a patch of underlying pixel or sensor values to an activation level. Let I_k denote the image patch that affects the state of the k^{th} receptive field. Let $\mathcal{I}$ denote the set of all such patches.

In addition to the image patch associated with each receptive field, the activation depends on the global state of the entire retina. In the case of a pan/tilt camera, the retina state can be described using the horizontal and vertical angle of the camera lens (θ, ϕ). In the case of the roving eye, the state of the retina can be described in terms of the horizontal and vertical offsets (u, v) that describe the position of the retina in the larger image. However the state space is parameterized, $\mathcal{S}$ denotes the set of all states.

Receptive fields implement an activation function $\delta : \mathcal{I} \times \mathcal{S} \to [0, 1]$. Thus, $\delta(I_k, s)$ is the total activation of the pixels in the image patch I_k given the current retina state s, normalized to $[0, 1]$ as a fraction of the maximum possible activation.

The activation over the entire retina is the sum of the activations for each receptive field for the current retina state, i.e.

$$R_{\mathcal{I}}(s) = \sum_{I_k \in \mathcal{I}} \delta(I_k, s). \tag{4.1}$$

4.3 Learning Saccades

Saccades result in 2D displacements of the image on the retina or pan/tilt changes for a physical camera. Each action or saccade $a : \mathcal{S} \rightarrow \mathcal{S}$ is described by a two-element vector denoting horizontal and vertical motion and results in a single globally rigid transformation of the image or scene.

If the receptive fields in the retina are of uniform size and distribution, and they are exposed to input consisting of a small spot of light against a uniform background, then $R_{\mathcal{I}}(s)$ would be approximately constant for all retinal states s, regardless of where the spot of light falls. However, with a *foveated* retina, $R_{\mathcal{I}}(s)$ will have a dramatic maximum for retina states that cause the spot of light to fall on the fovea, due to the larger density of receptive fields there.

Using the total activation of all the receptive fields for the current retina state, $R_{\mathcal{I}}(s)$ in Equation 4.1 as the reward, combined with saccade actions, defines a simple reinforcement learning problem. The goal is to find a policy, or choice of action, that maximizes retinal activation.

The global learning problem factors into an individual learning problem for each receptive field. The goal of each receptive field is to learn a policy that greedily maximizes the total retinal activation $R_{\mathcal{I}}(s)$,

$$\pi_k(s) = \arg_a \max R_{\mathcal{I}}(a(s)). \tag{4.2}$$

The problem is episodic and spans a pre- and post-saccade state. The collective policy π^* for the entire retina is the weighted average of the actions preferred by the individual receptive fields,

$$\pi^*(s) = \frac{1}{\mathcal{R}_{\mathcal{I}}(s)} \sum_{I_k \in \mathcal{I}} \delta(I_k, s) \cdot \pi_k(s). \tag{4.3}$$

In this factored learning problem, the only information a receptive field has about the state of the retina is the intensity level for that receptive field's visible patch I_k. If the intensity is high (i.e. $\delta(I_k, s)$ is close to 1), then the policy $\pi_k(s)$ will have a large impact on the global policy calculated in Equation 4.3. In this case, the policy should suggest an action $\pi_k(s) = a$ that maximizes the reward $R_{\mathcal{I}}(a(s))$. The action that accomplishes this goal takes the activation that the current receptive field sees and shifts it to the fovea, where the density of receptive fields is higher.

If the intensity is low, then the policy for that receptive field will have little impact on the policy for the entire retina since $\delta(I_k, s)$ is close to zero. As a consequence, $\pi_k(s)$ can be treated as a constant. So, in the factored problem, each receptive field only needs to estimate the optimal action and observe its own intensity level.

After sufficient training, the action specified by π_k will approximate the saccade that moves an image-point from receptive field k directly to the fovea. Consider the inverse $-\pi_k$ of the policy estimate for each receptive field. This is the action that would move an image-point from the fovea to the receptive field k. In other words, the inverse of the policy is a position for the receptive field relative to the fovea. Physically proximate receptive fields will have similar saccade policies, and hence similar learned positions. Note that the agent has not used any knowledge of the location of receptive fields within the fovea. In fact, that knowledge has been learned by the training process, and is encoded in the policy π_k. Spatial knowledge that was implicit in the anatomical structure of the retina becomes explicit in the policy.

The reinforcement learning problem described above has two unusual properties that constrain the choice of learning algorithm. First, the action space is continuous (as opposed to small and discrete). Second, the problem is episodic, and each episode spans only one choice of action, making it equivalent to a regression problem.

During learning, each receptive field maintains an estimate for π_k, the current best action, and R_k, the current maximum estimated reward after performing the current best action. Initially, each π_k is set to a random action, and the reward estimate is initialized to zero.

At the beginning of each iteration or training, the retina is randomly repositioned. For exploration, some noise ϵ is added to the current greedy policy. The retina agent executes $\pi^*(s) + \epsilon$, and measures the reward (R). Each individual receptive field's reward estimate and current policy are updated proportional to its state activation prior to the saccade (i.e. $\delta_k = \delta(I_k, s)$) since the optimal policy π^* is weighted according to those activations. A moving average learning rule updates both the reward estimate and current policy. For each receptive field k, the reward is updated as follows

$$R_k^{new} = \begin{cases} R_k^{old} + \delta_k \cdot \alpha \cdot (R - R_k^{old}) \text{ if } R > R_k^{old} \\ R_k^{old} \text{ otherwise.} \end{cases} \tag{4.4}$$

If the reward received, R, is greater than the current reward estimate, the current policy π_k for that receptive field moves closer to the global policy responsible for the increased reward

$$\pi_k^{new} = \pi_k^{old} + \delta_k \cdot \alpha \cdot (\pi^* - \pi_k^{old}). \tag{4.5}$$

The next section presents the results of applying sensorimotor embedding in a simulated domain and using a pan/tilt camera. The robust nature of the approach is evaluated using a lesion experiment and a vision inversion experiment.

4.4 Experiments

Experiments are presented in three sections. The first tests sensorimotor embedding in a simulated environment with a single light source. In the next section lesion and inversion experiments will show that sensorimotor embedding is robust to change. The third section applies the same algorithm to natural scene images and a physical pna/tilt camera, showing that the approach continues to work using real world data.

4.4.1 Simulation Experiment

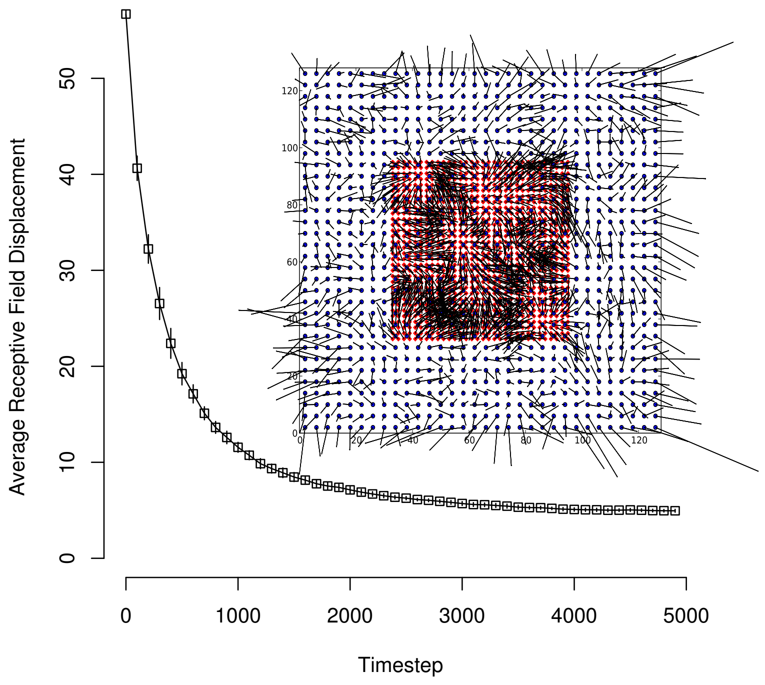


Figure 4.2: **Learned Sensor Geometry in Simulation**. This figure plots the mean geometric error as a function of training time. The mean and standard errors are shown for 10 independent training runs using a single dot image. The subfigure shows the result of interpreting learned receptive field policies as positions. Each line represents the error between the true position and learned position — the head (dot or diamond depending on the layer) is the true location of the field. The tail is the learned position. For clarity, only two layers are shown. The plot shows that error decreases as the training time of the saccade policy increases.

The simulated foveated retina had four layers of receptive fields and was trained on an image with a single white spot on a black background, meant to simulate the result of a saliency map. Each retina layer contained 32×32 receptive fields. The extent of each receptive field varied by layer, with the largest layer having receptive fields of size 4×4 (for a total retinal pixel area of 128×128). Actions corresponded to horizontal and vertical translations of the retina across the image. The policy for each receptive field was randomly initialized. The training rate was $\alpha = 0.5$. ϵ was normally distributed with a mean of 0 and a standard deviation of 10 pixels.

The first criteria for success computes the mean of the Euclidean distances between the learned position (interpreted as the additive inverse of the policy) and the true position $\text{pos}(I_k)$ of all receptive fields (Equation 4.6). Since translations of the roving eye retina are specified in pixels, this analysis compares pixel positions to action space positions. In experiments using a pan/tilt camera, ground truth actions are not available. The results of training are shown in Figure 4.2.

$$E_{\text{geometry}} = \frac{1}{N} \sum_{k=1}^{N} || - \pi_k - \text{pos}(I_k)||_2 \tag{4.6}$$

For the second criterion, comparisons are made with the accuracy of the learned saccade against the optimal saccade, which would center the retina on the area of high activation. Two-saccade accuracy, where the retina makes a second saccade after the first during testing but not training, was also examined.

During the training process, every 100 training steps, training was stopped and single saccade and two-saccade accuracy for 30 random repositions were performed. The average and standard errors of the accuracy over 10 training trials are shown in Figure 4.3, which also includes comparisons with a randomly initialized policy and an optimal policy (where each policy is initialized to the inverse of that receptive field's position).

The learning algorithm achieves near-optimal saccade accuracy after 5000 training steps. Comparing Figures 4.2 and 4.3, the geometric error decreases as accuracy increases, though the final sensor map only approximates the true positions of the receptive fields. The algorithm's final saccade error of five pixels is less than that of Pagel et al. [47] and requires only a quarter of the number of training steps.

4.4.2 Lesion and Vision Reversal Experiments

In natural scenes, or in cases where the number of receptive fields in the fovea changes as with macular degeneration, the maximum achievable reward changes. In these cases, the maximum achievable reward may decrease to a level below the current reward estimate for each receptive field, $R < R_k^{\text{old}}$ and so no updates will take place. To account for this kind of variation over time, the learning rule can be changed to maintain a recency-weighted average estimated reward, instead of maintaining an estimate of maximum reward.

This learning rule requires that the reward estimate be updated each timestep

$$R_k^{new} = R_k^{\text{old}} + \delta_k \cdot \alpha \cdot (R - R_k^{\text{old}}), \tag{4.7}$$

instead of only updating during timesteps where $R > R_k^{\text{old}}$.

In order to test adaptation, a small off-center part of the foveal region was lesion of the retina after 2000 steps of normal training. The mean post-saccade activation increases after lesioning when the agent uses the the robust learning rule (Figure 4.4), which may require more samples. The basic learning rule, however, does not adapt to the lesioning event.

Even though the reward estimates for each receptive field would adjust downward after a large change in the semantics of the motor commands, exploration still depends on adding noise to the previous policy estimate for each receptive field. In cases where the motor model changes radically, this exploratory bias may handicap any attempt to adjust.

Humans have shown some capacity for adapting to drastic changes in sensorimotor experience. For example, in a self study using prismatic inverting eye-wear, Dolezal reported both initial difficulty in simple reaching tasks followed later by comfortable mastery [15].

In Dolezal's inverted perceptual world, pointing up results in the visual perception of pointing down. By reversing the result of a motor command along one axis, it is possible to simulate a similar (but less complex) change in the relationship between the motor actions and perceptual response. Though this experiment does not capture the full range of altered sensorimotor contingencies, the experiment illustrates the need for a different kind of adaption in the face of significant changes in sensorimotor contingencies.

In this modification, each receptive field maintains an estimate of the optimal reward and policy as before. The retina also maintains an estimate of the maximum observed reward, a moving average of all the observed rewards, along with the reward estimates associated with each receptive field. The exploration/exploitation trade-off is driven by a parameter, γ, that measures the extent to which the learned policy for currently active receptive fields will be able to achieve the maximum observable reward as estimated by the retina as a whole.

For a given pre-saccade retina state s, compute both the current action estimate a and the reward estimate r_a. The parameter γ is then the ratio of r_a to r_{max}, the maximum observed reward for the entire retina. Intuitively, if r_a is close to r_{max} then the action a is likely close to optimal, and so little exploration is necessary. Similarly, if r_a is less that r_{max}, the action a is likely suboptimal, and so more exploration is required. The actual action taken is then

$$\gamma a + (1 - \gamma) a_{\text{exp}},$$

where a_{exp} is a random saccade.

A large negative change in the moving average of all the rewards served as an indicator of a major change to the retina motor or sensor map. When detecting this kind of change, the retina resets the reward estimates of all the receptive fields to their original values. This significantly decreases γ, triggering an increase in exploration and decreasing the contribution of the previously learned policy.

4.4.3 Natural Scene and Pan/Tilt Experiments

To recapture the features of the single spot case in natural scenes required constructing a *proto-saliency* map from natural scenes by first blurring the image under the retina using a Gaussian blur with a 5x5 filter size. Blurring is incompatible with the assumption that geometric information is not available. However, this blurring step is meant to simulate the optical characteristics of infants during early development [61]. The image is thresholded and pixels fall into the top one percent brightness level in the region under the retina are included. If the number of active pixels is less than 500 pixels, the agent proceeds to train on that portion of the image, otherwise the agent performs a new random saccade without training. This process avoids training in situations of homogeneous brightness that wash out any existing progress on learning the optimal policy.

Note that humans tend to avoid saccades to areas of high luminance at low spatial scales (e.g. sky, solid colors) [74]. By avoiding training when the number of active pixels after thresholding is too high, the agent avoids training on precisely these kinds of high-luminance inputs.

Due to the variation in learning performance across images, the model was trained over subsets of images randomly chosen from the Berkeley segmentation dataset [39]. For each run, a set of images (N=1, 5 or 10) to train over was selected. Training proceeded by cycling through the images, training 19 times over each image before moving to the next image in the cycle to continue training. To evaluated the learning performance geometric errors were measured every 100 steps of training.

The results are shown in Figure 4.6.

Even though the final error rates are higher than when trained with the synthetic scene (Section 4.4.1), note that the fixed point behavior of the policy (allowing repeated corrective saccades) does result in accuracy comparable to what training achieves on an ideal version of a saliency map after a similar number of training steps. Table 4.1 shows the accuracy after one and two saccades, as well as after the number needed to reach a fixed point (or in rare cases, a cycle – in which case the closest cycle point is counted).

Table 4.1: **Saccade Accuracy**. This table shows accuracy in pixels of a single saccade, two saccades in succession, and after repeated saccades reach a fixed point. Saccades from low resolution regions of the retina have decreased accuracy. Multiple successive saccades can compensate for this decrease in accuracy.

1 Saccade	2 Saccades	Fixed Point
20.4	12.5	7.6

For the physical pan/tilt experimental setup, a Logitech QuickCam Orbit AF was placed 15 feet from a single light source. To reduce training time, the exploration policy did not search randomly for a bright light. The agent performed a random saccade away from the light source. During training the agent than performed the opposite saccade back towards the light source, and used the resulting retinal activations to learn a function from field activation to optimal saccades using the algorithm described with the proto-saliency method. Unlike a learned policy, this open-loop training policy cannot account for relocation of the salient light source.

Figure 4.7 shows the decrease in saccade error and the increase in post-saccade reward (or activation) after intervals of 100 training steps. Each data point is the mean of 10 test trials. Each trial randomly saccades away from the light source, then computes the return saccade as the activation-weighted average of the learned receptive-field policies. For a trained retina, the post-saccade reward is independent of the initial random saccade, since the state of highest reward is reachable from any random starting position.

In the simulation experiments, the learned policies correspond to ground-truth pixel geometry, since actions for the simulated roving-eye camera are pixel unit translations over an image. The action space of the pan/tilt camera, however, is not represented in pixel-unit shifts. The motor commands represent control signals sent directly to the piezoelectric motors in the camera apparatus. Camera geometry, along with irregularities in camera control, make the correspondence between motor signals and pixel shifts in the field of view necessarily inexact. The geometry of these action space coordinates approximates (up to a scale factor) the ground truth geometry of the receptive fields in pixels. These experimental results confirm that, under simple assumptions, an agent can simultaneously discover motor and sensor maps for a foveated retina. Previous approaches to sensor map construction use dimensionality reduction techniques and do not exploit additional available domain structure, namely access to motor commands.

4.5 Discussion

The experiments presented in this chapter show how sensorimotor embedding can be used to learn the structure of a foveated retina. The the pattern of activation over a foveated retina provides a natural perceptual goal for agents learning sensor structure. The ballistic nature of the saccade actions means that the actions can be directly associated with sensor element coordinates. This property of *ballistic properties* was discussed in Chapter 3. The sensorimotor embedding approach to learning sensor geometry was able to adjust to lesion and reversal events, and learn from real-world data.

4.6 Conclusion

An agent can use sensorimotor embedding to learn sensor structure. In the next chapter, an agent will apply sensorimotor embedding to the problem of learning robot pose in both a **Gridworld** and **Roving Eye** domain.

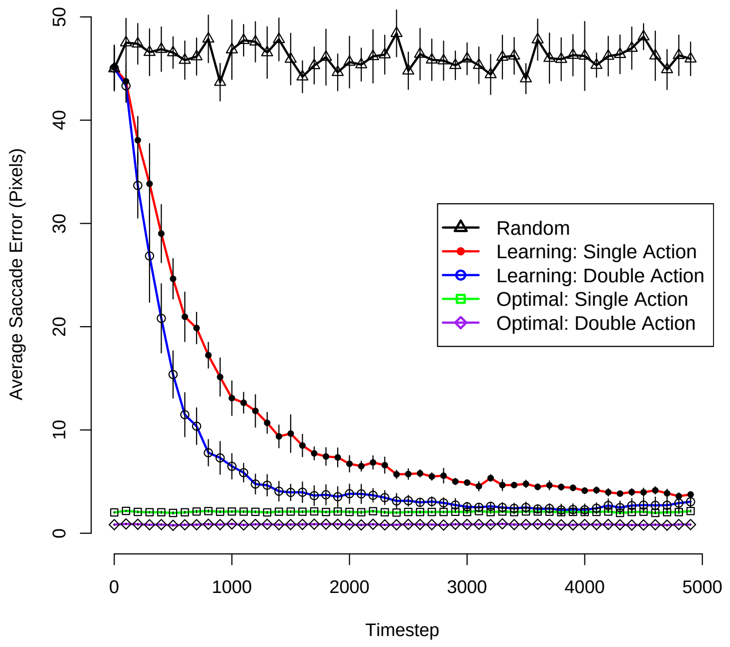


Figure 4.3: **Saccade Policy Training**. The saccade error as a function of the number of training iterations using the learning algorithm. The saccade error is computed over thirty random repositions every 100 timesteps for 10 independent trials. Note that even with an optimal policy, saccades are not entirely accurate because of low resolution in the periphery of the retina. The error decreases with training time.

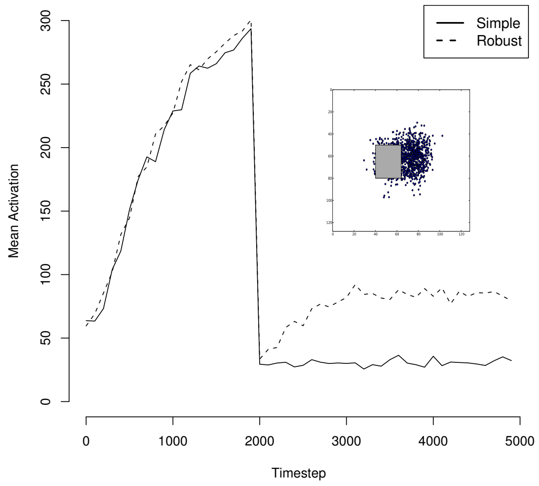


Figure 4.4: **Lesion Results**. Lesion at T=2000. As a result of lesioning, a retina, with a robust learning rule as described in this section, adapts its policy to favor saccades to regions just outside the damaged region (as shown in the subfigure), providing higher post-saccadic activation in the case of lesioning than the previous optimal saccades directly to the fovea. Note that this mechanism increases the position error relative to the ground truth, but provides a coordinate system consistent with the sensorimotor properties of the damaged retina. The basic learning rule fails to adapt following a lesioning event. With the modified learning rule, sensorimotor embedding can detect and adapt to a lesion event.

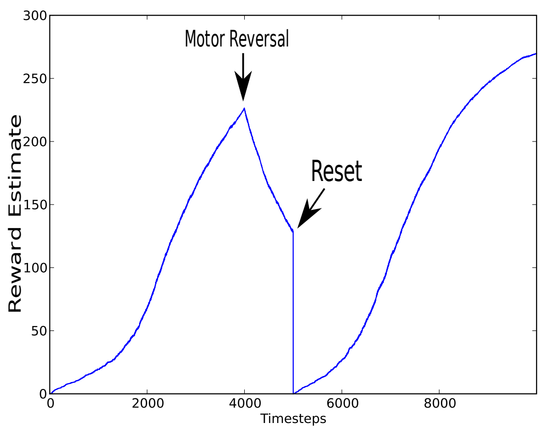


Figure 4.5: **Vision Reversal**. Vision reversal is simulated by switching the up and down motor commands. The moving average estimate of rewards experienced during training. A reversal that occurs after 4000 timesteps results in a decrease in the moving average reward estimate. After decreasing over 1000 timesteps, the retina resets the rewards estimate and the estimates for each receptive field and begins adapting to the new situation. This shows how sensorimotor learning can relearn important geometric features.

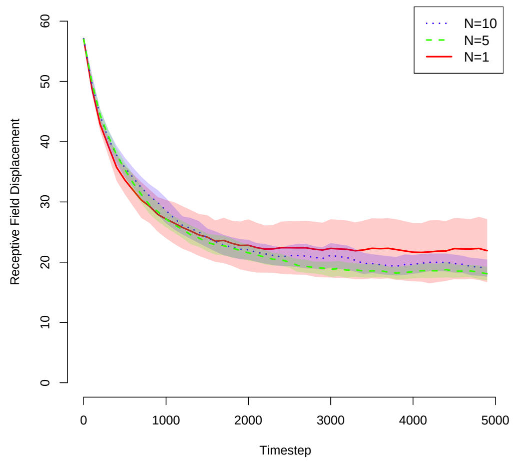


Figure 4.6: **Natural Scene Image Results**. Subsets of natural scene images were chosen randomly. This graph shows the mean and variance of ten runs for each subset size and is best viewed in color. Training across sets of images results in more consistent learning curves than training over single images, since the variance is smaller for training that takes place across subsets. Even in the single image case (where each run drew training examples from a single image) the mean learning curve was qualitatively similar to the others, but the high variance suggests that some images are "bad" sources of training examples. The agent did learn a saccade policy and retina geometry with an average error of 20 pixels. This demonstrates that sensorimotor embedding can learn sensor geometry on real world data.

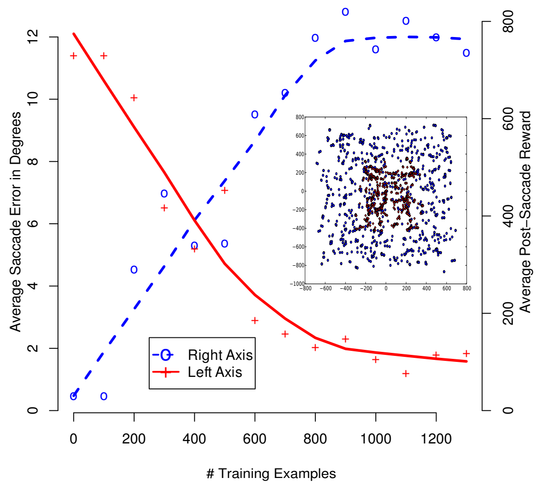


Figure 4.7: **Pan/Tilt Results.** Every 100 training timesteps, 10 test trials were performed with the pan/tilt camera, randomly saccading away from the light source, then using the learned saccade policy to attempt to recenter on the light source (as opposed to using the inverse of the random saccade as in training). As training progresses, each receptive field learns a policy that centers local activation at the fovea resulting in greater post-saccade reward (dashed line) and lower saccade error (solid line). The subfigure shows the corresponding action space coordinates of each receptive field for two different layers of receptive fields after training. The pan/tilt results show that sensorimotor embedding can recover sensor geometry on a physical robot.

Chapter 5

Learning Robot Position

In this chapter, *sensorimotor embedding* is evaluated by learning about the geometry of two simulated robot domains. The results show that sensorimotor embedding provides a better mechanism for extracting geometric information from sensorimotor experience than standard dimensionality reduction methods. The first, **Gridworld**, is a simple discrete type of Markov decision process meant to establish whether geometric information concerning the location of states in the domain can be extracted from policy trajectories using sensorimotor embedding. The second, **RovingEye**, provides an environment analogous to the visual ego-sphere of a developing robot.

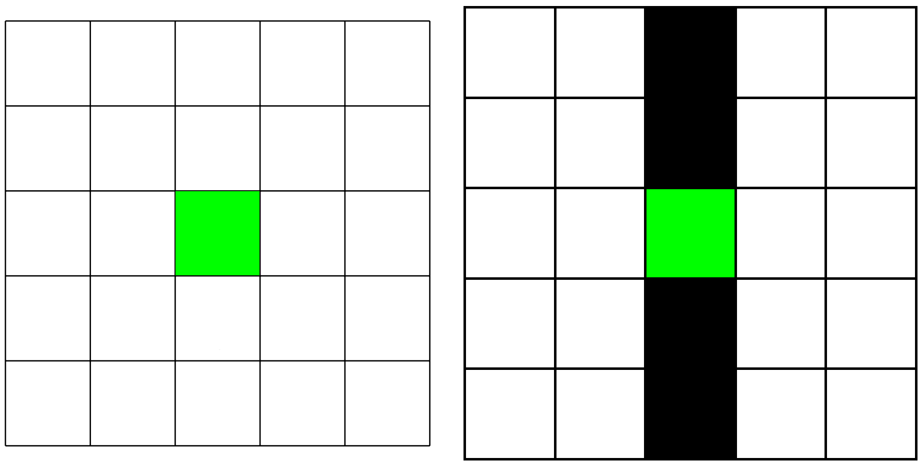
The experiments use both a value function method in the **Gridworld** domain and a policy search method in the **ImageBot** domain. One key question for sensorimotor embedding is whether the policy improvements also improve an agent's metric understanding of position and a key empirical result established in this chapter is that any method of improving an agent's policy will also improve the accuracy of the representation generated using sensorimotor embedding.

5.1 Gridworld Experiments

Gridworlds provide a simple discrete environment for analyzing the ability of different sensorimotor methods to recover the spatial layout of the world from the sensorimotor experience of the agent. There are many algorithms for learning optimal policies, and gridworlds provide a simple abstract model for testing these approaches. An example gridworld used in the experiments in this chapter is shown in Figure 5.1. Though gridworld domains may seem simple, the idea of a gridworld is applicable in a wide variety of modeling situations. For the experiments considered in this chapter, the simplicity of the gridworld domain allows for a clear examination of the properties of sensorimotor embedding.

For example, in the plain gridworld domain shown in Figure 5.1a, it is possible to explore the impact of policy improvement on representation in a coarse way. Consider the result of applying a random policy, and using the resulting action traces in sensorimotor embedding. Since the policy is random, it does not contain any information about the geometric structure of the domain. The result should also be random. Figure 5.2 provides one such random result. For the same gridworld environment, an optimal policy results in a faithful representation of the original state geometry in Figure 5.1a. In fact, the Procrustes error decreases as the policy improves (Figure 5.4).

Trajectories generated using a random policy did not lead to a reasonable representation of the corresponding states. However, after learning an optimal policy with Least-Squares Policy Iteration the same analysis resulted in a far more accurate reconstruction of the underlying state geometry [35]. This result shows that performing sensorimotor embedding using trajectories from an optimal policy leads to low-dimensional coordinates of the states that follow the ground-truth arrangement. This result demonstrates that optimal policies contain implicit information about environment geometry that sensorimotor embedding makes explicit.



(a) Plain Gridworld (b) Bowtie

Figure 5.1: **Gridworld Domains**. These domains are simple discrete Markov decision processes meant to illustrate how sensorimotor embedding infers geometric information about the relative locations of states. The left domain is a standard gridworld. The right domain replaces some states with barriers. The barriers cause problems for embodied Isomap, but sensorimotor embedding is able to discover the world geometry despite the barriers. The center state is the goal state. The agent receives a reward when it enters the goal state.

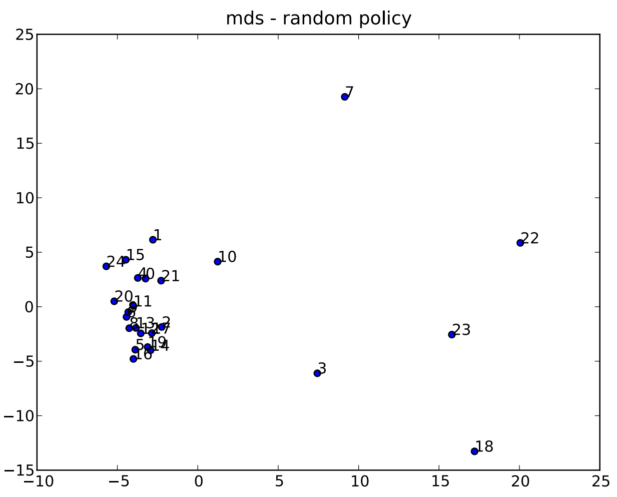


Figure 5.2: **Random Walk Policy**. This shows the result of inferring distances from a random walk policy via sensorimotor embedding in the plain gridworld domain (Figure 5.1a). A random policy is not optimal and does not implicitly encode any information about geometry. The inferred locations of the gridworld states end up in random positions after applying sensorimotor embedding to random action traces.

The gridworld environment is a convenient domain for exploring variations on the sensorimotor embedding algorithm described in Chapter 3. For example, the alternate method of comparing action traces, where zero actions are prepended instead of appended to traces, can be easily applied in this domain. This approach to trace length normalization produces results that appear equivalent to the standard approach described in Chapter 3 (Figure 5.5a). The performance of sensorimotor embedding can also be evaluated for different choices of goal state. For example, setting the goal state to the upper right hand corner results in a sensible, but somewhat skewed set of coordinates (Figure 5.5b).

In a gridworld environment as shown in Figure 5.1b, a barrier partially divides the domain. This division creates problems for accurate reconstruction of the domain using alternatives to sensorimotor embedding (Figure 5.6). In contrast, sensorimotor

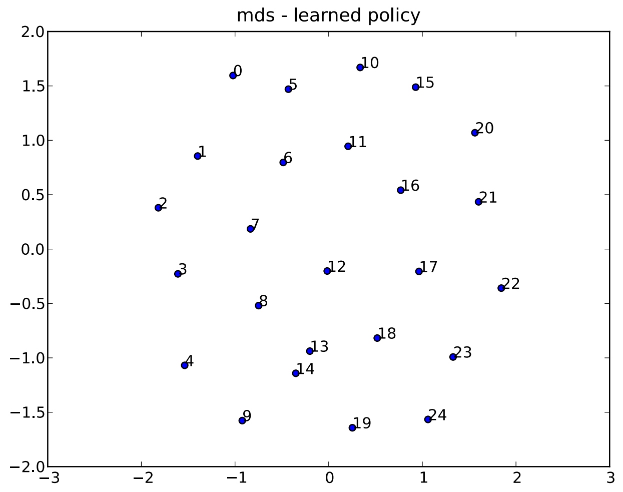


Figure 5.3: **Optimal Policy**. Multidimensional scaling applied to the distance matrix inferred from policies learned using LSPI [14]. Sensorimotor embedding is able to recover the state space geometry using the learned optimal policy. The result of sensorimotor embedding is a two-dimensional grid of states that follows the actual geometric relationships among the states shown in Figure 5.1a.

embedding is able to faithfully reconstruct the domain geometry.

Figure 5.7 highlights the advantage of using sensorimotor embedding over approaches that only use local distances. The sensorimotor embedding approach is able to determine the relative locations of states that are adjacent in the original environment, but separated by a barrier that prevents any direct movement between them. Approaches that only use local distance cues, like Isomap and action respecting embedding, as in Figures 5.6, fail to capture the global geometric structure of the domain using only two dimensions. By analyzing action traces, sensorimotor embedding can provide a two-dimensional representation of the state geometry that is close to the ground truth.

Gridworld experiments highlight the properties and advantages of sensorimotor embedding. However, one key way gridworlds fail to truly test sensorimotor

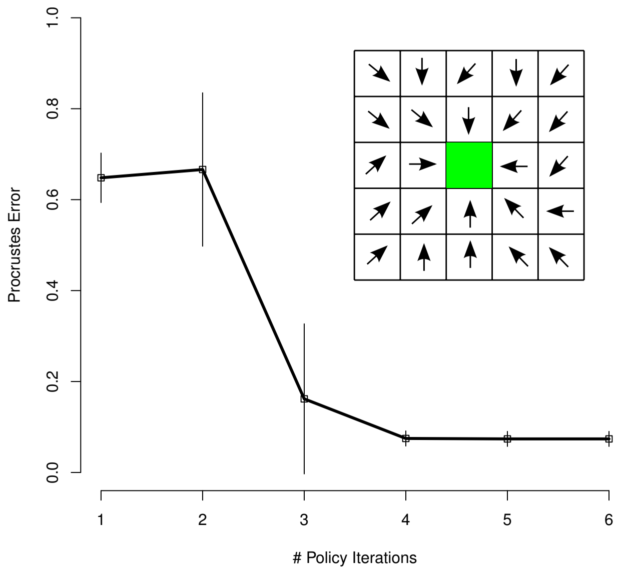
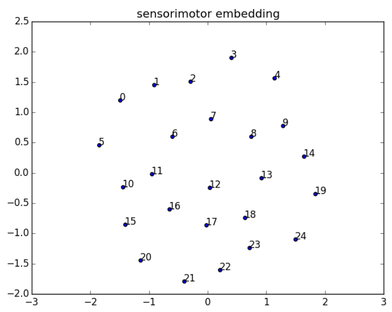


Figure 5.4: **Geometric Error and Policy Improvement**. The geometric error decreases as the policy improves with each iteration of least-squares policy iteration (LSPI) [35]; the subplot is a visualization of an optimal policy in the **Gridworld** domain used for this experiment. The closer a policy is to optimal, the less the geometric error after applying sensorimotor embedding.

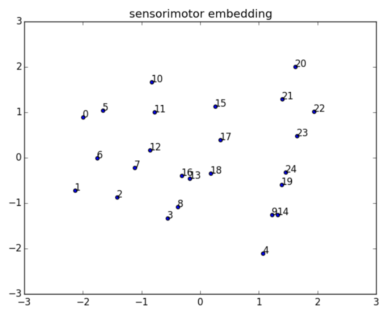
embedding, is that the sensory state is simple. For sensorimotor embedding to be applicable more broadly, it should work for the realistic sensory inputs of a developing agent. The **Roving Eye** domain is one way to introduce sensory complexity.

5.2 Roving Eye

In the **RovingEye** domain, a simulated eye moves around a static image. The goal of the agent is to learn to localize. This domain was used in related work learning sensor geometry [54, 66] and learning embeddings using action labels [10]. Unlike the simpler **Gridworld** domain, the **RovingEye** domain involves continuous action spaces and high-dimensional perceptual inputs in the form of sub-images of a natural



(a) Prepend Zero Actions



(b) Different Goal State

Figure 5.5: **Additional Gridworld Results**. The learned geometry on the left is generated using an alternate method of comparing action traces where zero actions are prepended, instead of appended to action traces when normalizing trace lengths. The results are similar to those generated using the standard method. The learned geometry on the right is the result of applying sensorimotor embedding to a gridworld with the goal state shifted to the upper left corner. This change results in a more skewed representation of the state coordinates.

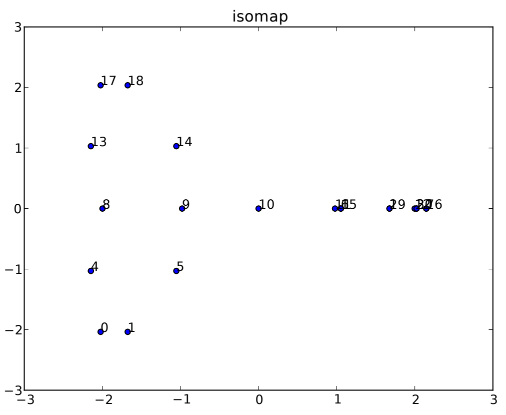


Figure 5.6: **Embodied Isomap Gridworld**. The two-dimensional embedding generated using Isomap with distances drawn from the magnitude of the local actions that move between states is shown. Unlike sensorimotor embedding, embodied Isomap does not properly reconstruct the two-dimensional structure of the gridwold shown in Figure 5.1b.

scene. The **Roving Eye** domain provides a good test for the ability of this scheme to reduce the dimension of the input data.

In the experiments in this chapter, the eye was a 128×128 array of pixels (Figure 5.9). The perceptual goal states for this domain were generated by specifying a gradient policy using a set of directional filters applied to the intensity image. By following the gradient policy from each starting point in the image, the agent can identify a much smaller set of local maxima, that when clustered, form a reasonable set of perceptual targets for learning. Following the gradient from each point in the image results in a trajectory that terminates at one of these perceptual goals. The clusters represent the regions of attraction for these local maxima.

The principal goal of this experiment is to establish a link between the quality of the embedding and the efficacy of the policies that bring the agent from points in the environment to perceptual goals. To this end, several different approaches to

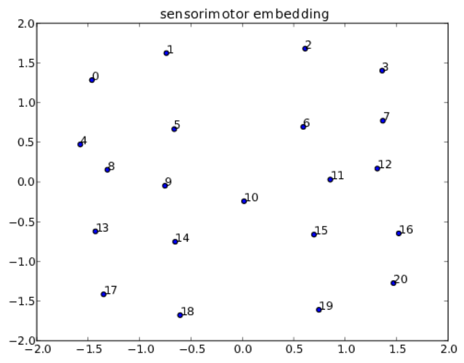


Figure 5.7: **Sensorimotor Embedding Gridworld**. The result of applying sensorimotor embedding to full trajectories. By using the full trajectories to the shared goal state to determine interstate distances, sensorimotor embedding is able to generate an accurate representation of the relative locations of the states of the gridworld shown in Figure 5.1b using only two dimensions.

generating trajectories of varying quality were used.

For each of the multi-step policies, the action space is limited to a set of 16 discrete actions representing movements of length 5px in 16 different directions. The first type of trajectories used for sensorimotor embedding resulted from just following following the gradient. For the second approach, ϵ-gradient, the agent followed the gradient but choose random actions with a probability of 15%. The agent used the highest-scoring sample trajectories as the input for sensorimotor embedding. The third approach used a near-optimal hand-coded policy. Fourth, the agent learned a ballistic policy using the same stochastic estimation method as in Chapter 4 [66].

Example trajectories are shown in Figure 5.10. These trajectories all attempt to acquire the same perceptual goal. When terminating, the agent receives a reward based on the distance to the goal. The score for a trajectory is discounted by the number of actions taken to reach the final state. Discounting has the effect of assigning higher scores to shorter, more efficient policies. The hand-coded policy generated the highest scoring trajectories.

Procrustes analysis (Section 3.2.1 and Dryden et al. [16]) is used to evaluate the quality of the representation that results from applying sensorimotor embedding using each set of generated trajectories. This analysis corrects for rotation and scale differences between sets of points before computing the residual geometric error. A lower error implies that points are a better statistical fit to the ground truth data, which consists of the true pose of the roving eye corresponding to each sensor signal. The scores along with corresponding errors are shown in Table 5.1.

Note that as the average score of the trajectories (measured over a sampling of points in the region of a single perceptual goal) increases, the error after Procrustes analysis decreases. For comparison, classic multi-dimensional scaling was applied to the raw intensity images, using pixel differences as a measure of dissimilarity. That approach (the classic linear dimensionality reduction approach) resulted in the highest error. Trajectories that score higher are more efficient and result in lower error after performing sensorimotor embedding. Figure 5.8 shows the importance of each component in the new representation. The better performing methods, such as sensorimotor embedding applied to ballistic trajectories, have the most weight concentrated on a small number of components in the new representation.

Figure 5.11 shows the result of sensorimotor embedding on randomly selected points used in the analysis in Table 5.1. For clarity, only the ground truth poses and the result of embedding gradient and ballistic trajectories are shown.

The ballistic trajectories result in a more accurate embedding than the gradient trajectories, as indicated by the Procrustes analysis in Table 5.1. The difference in quality between using optimal multi-step trajectories and learned ballistic trajectories indicates that discretizing the action space reduces the representational power of this approach. Similar but less substantial improvements are observable with other methods of generating trajectories.

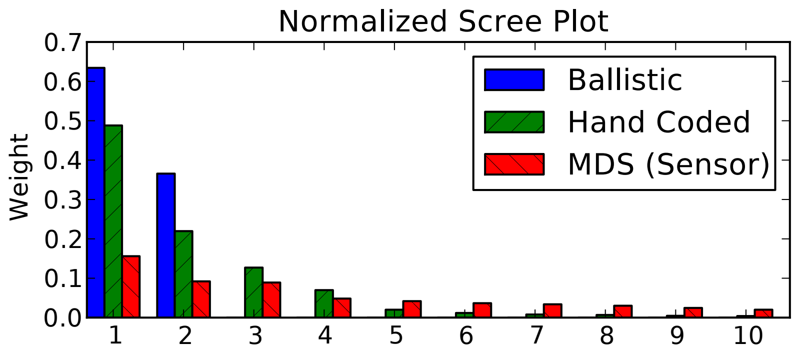


Figure 5.8: **Scree Diagram for Roving Eye Domain**. A scree diagram of the normalized weight of the first ten components in the new representation. A compact representation, such as that generated using ballistic trajectories, should have a small number of high weight components. A non-compact representation, such at that produced by MDS applied directly to sensor distances, will have a less concentrated weight distribution.

5.3 Discussion

The experiments in this chapter demonstrate that sensorimotor embedding provides a mechanism for representing geometry using sensorimotor experience, and that improvements in policies result in more accurate representations of geometry. Sensorimotor embedding allows agents to learn local geometry in an incremental and scalable way. In addition, since spatial representations are derived from actions using sensorimotor embedding, the resulting geometric representations are naturally calibrated to the agent's own body. Sensorimotor embedding performs better than other manifold learning methods, even methods that take into account agent actions like embodied Isomap. By applying manifold learning methods such as multidimensional scaling to action traces instead of raw sensor signals, sensorimotor embedding is able to benefit from geometric knowledge implicit in policies.

Table 5.1: **Roving Eye Experimental Results**. As the average trajectory score increases, the residual error after Procrustes analysis decreases. The ballistic trajectories result in the smallest error, in part because the ballistic trajectories are capable of expressing the precise distance relationships between points and goal states. Multi-step trajectories using discrete actions (even with the near-optimal hand-coded policy) are only capable of approximating the ground truth interpoint distances.

	MDS (Sensor)	Gradient	ϵ-Gradient	HC	Ballistic
Score	NA	0.35	0.51	0.62	0.67
Error	0.80	0.20	0.11	0.05	0.01

5.4 Conclusion

These experiments demonstrate that agents can use sensorimotor embedding along with interactive experience to recover the geometry of the environment in both the **Gridworld** and **RovingEye** domains. In addition, as policies improve so do the accuracy of the results of sensorimotor embedding, demonstrating that agents can acquire geometric knowledge incrementally and robustly through policy improvements. The next chapter will continue to demonstrate the general nature of sensorimotor embedding by applying sensorimotor embedding to the problem of learning the pose of a three dimensional object.

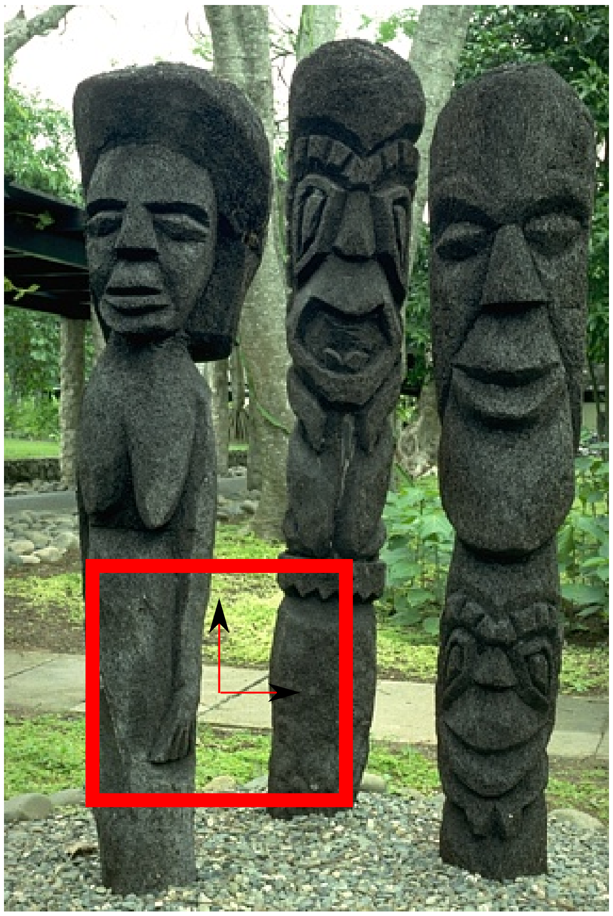


Figure 5.9: **Roving Eye Domain**. In the **RovingEye** domain, a simulated eye moves around a background image. This domain is useful for evaluating sensorimotor embedding on real world data.

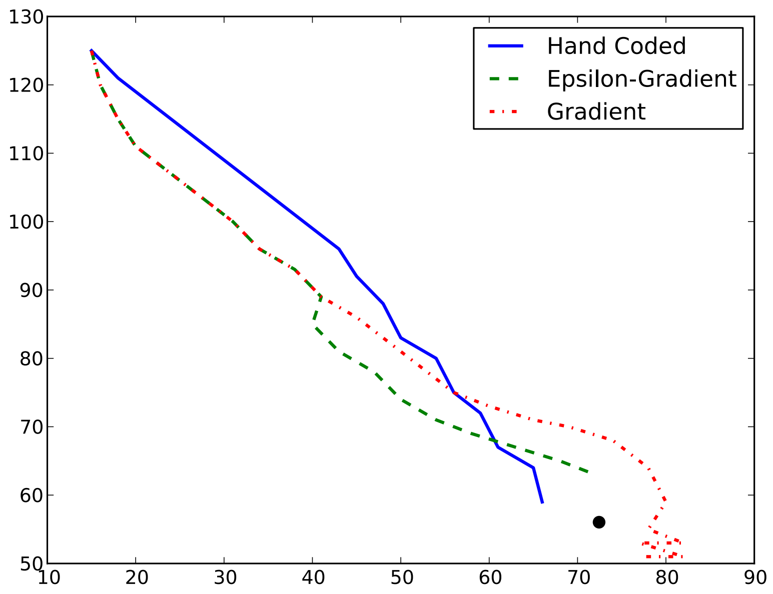


Figure 5.10: **Example Robot Trajectories**. This shows three example trajectories (gradient, ϵ-gradient, and hand-coded). The action sequences are used to determine interpoint distances in the corresponding embedding. The more efficient policies result in more accurate embeddings.

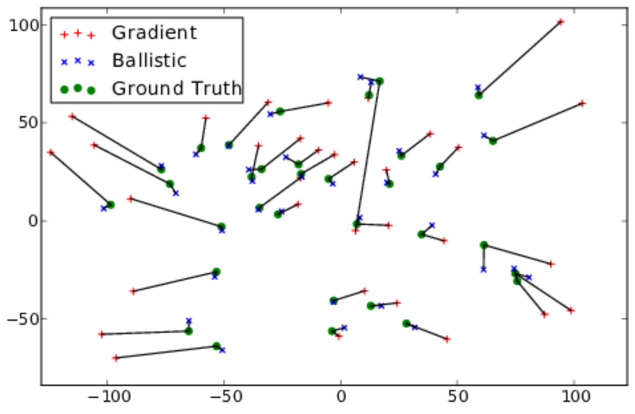


Figure 5.11: **Example Embeddings of Robot Positions**. The ground truth, gradient and ballistic sensorimotor embeddings for a set of randomly chosen points within the region of the largest goal state cluster are shown. Both ballistic and gradient embeddings are connected to the ground truth with line segments. The ballistic embedding provides the best approximation of the ground truth arrangement of the points.

Chapter 6

Learning Object Pose

In previous chapters, sensorimotor embedding was applied to the problem of learning sensor structure and robot position. In this chapter, sensorimotor embedding is applied to the problem of learning object pose, an important category of geometric knowledge for robots and other embodied agents. As demonstrated in this chapter, sensorimotor embedding performs better than standard unsupervised methods of pose estimation, and allows the agent to solve important problems like object alignment without complex calibration.

6.1 Motivation

Properly estimating object pose can simplify control tasks involving grasping [26] and object recognition [56]. Many approaches to pose estimation involve either sensor calibration, large labeled datasets, or unsupervised learning. Sensorimotor embedding provides an alternative to existing methods that performs better than unsupervised approaches, but does not require labeled training sets or ground truth data usually required for calibration approaches. Instead, the agent seeks a policy to achieve a perceptual goal. As a result of applying the policy, the agent learns to associate object orientation with the visual features used as policy inputs.

For a lander on a distant comet, determining orientation might mean the difference between aligning solar panels properly, or running down the batteries before the mission completes. For a developing robot, knowing object pose is an important prerequisite for other tasks, such as grasping. Sensorimotor embedding provides a theory for how object pose can be learned from developmental experience.

6.2 Setup

For object pose experiments, a three-dimensional object is simulated using Gazebo [32]. The object is painted using realistic textures to preserve the visual complexity of the task. The object is lit from the front. The agent controls the object in simulation by issuing pitch, yaw, and roll commands to manipulate the object pose. The simulated agent has the ability to adjust the pitch or yaw of the robot by $\pm\frac{\pi}{16}$ radians with each action. This experimental setup, though controlled by a computer algorithm, is similar to the setup used in psychology experiments on pose bias in humans (e.g. [31]). As discussed in Section 3.3, research shows that humans prefer certain orientations [23]. Adults prefer:

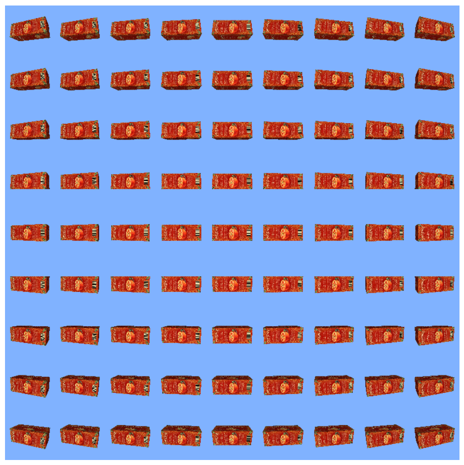
- planar views over 3/4 views;
- flat surfaces normal to lines of sight;
- upright orientations with respect to gravity.

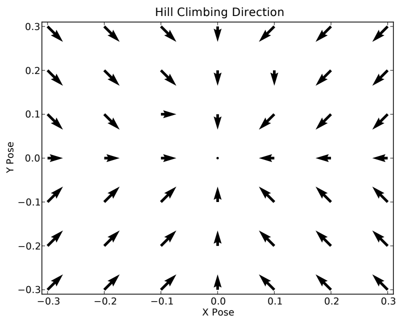
These biases follow a developmental trajectory. Newborns do not show as pronounced a gaze bias as adults, but the bias increases with age (see [49] as discussed in Chapter 3). The experimental results in psychology indicate that a useful perceptual goal when learning object pose is to seek a view on an object that is normal to a flat surface. A normal view to a flat surface would maximize total interpoint distance for a set of coplanar points. So if an agent seeks an object orientation that maximizes the coplanar interpoint distance, the agent would also show a preference for normal views on flat surfaces. Coplanar interpoint distance can be computed for pairs of images and forms a gradient.

The maximum defines the perceptual goal state. The gradient can be determined from two images of an object taken at slightly different orientations by collecting a set of robust features (e.g. SIFT or SURF features) from each pose. The feature sets for each pose are $f_1 = \{\langle x_i^1, a_i^1 \rangle\}$ and $f_2 = \{\langle x_i^2, a_i^2 \rangle\}$. Here the x_i terms are the locations of the features and the a_i terms are the corresponding feature appearances. For the appearance matched features, RANSAC [17] finds the fundamental matrix for matched points such that for matching points x_i^1 and x_j^2, $x_i^1 F x_j^2 = 0$. For inliers, RANSAC again finds a homography matrix H such that x_i^1 H - $\mathrm{x}_\mathrm{j}^2 = 0$. The remaining inliers form the coplanar feature correspondences. Figure 6.1 shows the result of this gradient analysis on a set of varying poses around a desired view of an object.

Least Square Policy Iteration (LSPI) is used to learn a policy for achieving a perceptual goal state as defined by the gradient. Once an agent has a policy for achieving a perceptual goal state, the action traces that result from application of the policy provide the information necessary to reconstruct object pose. These action traces consist of incremental changes of both pitch and yaw. The agent compares these action traces as described in Chapter 3. The resulting distance matrix is then transforming into a set of low-dimensional pose points using multidimensional scaling.

In addition to sensorimotor embedding, PCA and Isomap were applied to the problem of recovering object pose. PCA and Isomap were applied to image sets consisting of many pitch and yaw object configurations in an attempt to recover the original pitch and yaw parameters. The results of sensorimotor embedding, PCA,


(a) Gradient Images



(b) Computed Gradient

Figure 6.1: **Object Rotation and Gradient.** The images used for the gradient computation are shown on the left. The perceptual goal state depends on the gradient defined by changes in average interpoint distances for coplanar points. The right plot shows the computed gradient using changes in average interpoint distances for coplanar points. Using average interpoint distances for coplanar points results in a perceptual goal state that is normal to the object surface as in human trials.

and Isomap were compared with ground truth data using Procrustes analysis as described in Section 3.2.

6.3 Results

Figure 6.2 shows the ground truth pitch and yaw positions of the object used in this experiment, and the results of applying PCA, Isomap, and sensorimotor embedding. Procrustes analysis is used to evaluate the quality of each embedding.

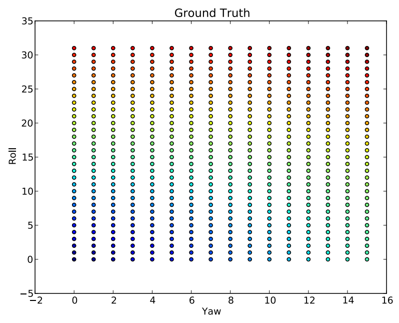
The Procrustes error for PCA and Isomap in this problem is much higher than the error for sensorimotor embedding. In addition, sensorimotor embedding has more weight on fewer components after matrix decomposition than either PCA or Isomap, indicating that it is modeling the variation in terms of many fewer components than either PCA or Isomap.

In Figure 6.3, the Procrustes error after sensorimotor embedding is shown as a function of the number of iterations of LSPI. As the policy for manipulating the object improves, so do the results of sensorimotor embedding. The scree diagram compares the distribution of eigenvalues between PCA and sensorimotor embedding. As discussed in Section 3.2, a smaller number of higher value eigenvalues indicates a better embedding.

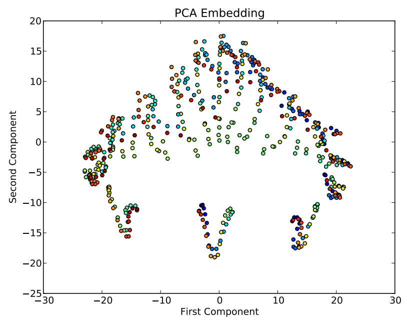
The agent can use learned pitch and yaw coordinates as features for an alignment task. The agent is given a target orientation in sensorimotor space and needs to learn a policy to align the observed object with the target. The agent is provided a reward when the object matches the target. In this case, the output of sensorimotor embedding provided the necessary information to make solving this task easy. As shown in Figure 6.4, the agent was able to learn a policy in only a few iterations of LSPI. Not only is recovering pitch and yaw from action sequences generated using a learned policy possible, but the resulting knowledge can be used to solve a task.

Figure 6.2: **Object Pose Results**. The ground truth yaw and pitch orientations for the pose reconstruction experiment is shown in (a). The results of PCA applied to the object images are shown in (b). PCA does a poor job reconstructing the ground truth object pose. The result of applying Isomap is shown in (c). The result of sensorimotor embedding using action traces is shown in (d). The pitch and yaw reconstruction from sensorimotor embedding is far more faithful than either PCA or Isomap.

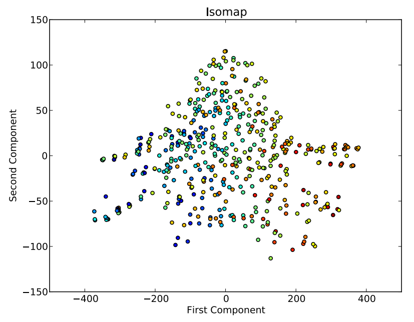
(a) Ground Truth



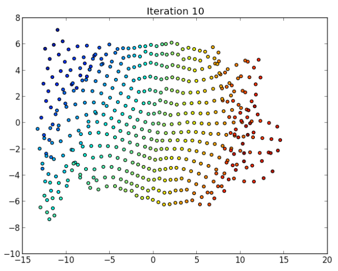
(b) PCA

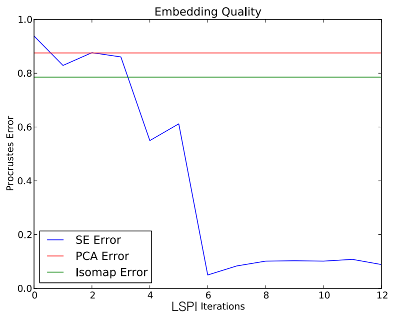


(c) Isomap

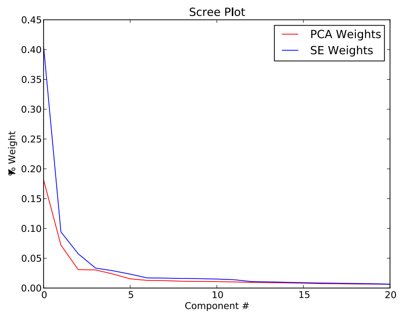


(d) Sensorimotor Embedding





(a) Procrustes Error



(b) Scree Diagram

Figure 6.3: **Procrustes Error and Eigenvalue Comparison**. The Procrustes error of sensorimotor embedding is compared with both Isomap and PCA over the iterations of LSPI. As the policy to achieve the desired pose improves, so does the agent's knowledge of the pitch and yaw of the object at the start of every sequence.

6.4 Discussion

The results demonstrate that sensorimotor embedding can be used to recover pose information from sets of images. Unlike unsupervised methods, sensorimotor embedding requires that an agent be able to interact with or move around the object. Though this narrows the potential applications of sensorimotor embedding, developmental agents that can interact with the world would be able to learn more accurate pose models using sensorimotor embedding. The distribution of the eigenvalues for sensorimotor embedding provides evidence for the assertion that the manipulation policy implicitly encodes the dimension of true variation, even if the agent actions are complex and multivariate. Sensorimotor embedding, in relying on the action traces and local action distances, recovers these exact dimensions of variation.

In this chapter, the agent learned a manipulation policy using LSPI. The perceptual goal state was determined using a gradient inspired by human studies. An agent could instead use hill-climbing directly as the policy, though in general a

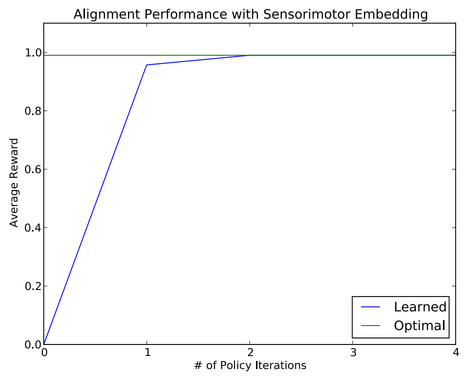


Figure 6.4: **Pose Alignment Task**. An agent can solve the alignment task using reinforcement learning over learned pitch and yaw coordinates. Learning the optimal policy requires only a few iterations of LSPI using the input from sensorimotor embedding.

learned policy will result in better results, particularly if the gradient is not smooth. The gradient described in this chapter does use some general notions of the geometric relationship between coplanar points. For developing agents, this knowledge would have to be encoded as part of the learning toolset. Another plausible theory, inspired by results from developmental psychology, is that these goal states are taught. That is that, when caregivers and children are observed under controlled conditions, the caregiver may present objects to the child in a biased way. A developing robot could similarly be shown certain object orientations then be tasked with manipulating the object to fit those orientations.

Taking a step back, the learning process turns thousand dimensional data into two dimensional data using, as an intermediate step, a policy that the agent uses to manipulate object pose and achieve a perceptual goal state. The process is shown in Figure 6.5, and by following the process, a developing agent can recover knowledge

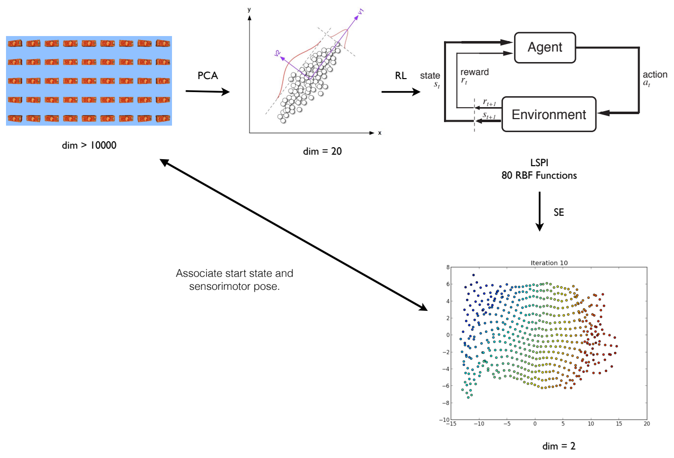


Figure 6.5: **Sensorimotor Pipeline**. The learning pipeline for geometric features corresponding to object pose. The end result of the learning process is an association between images of over 10,000 dimensions corresponding to start states, and two dimensional pose features. The initial application of PCA and LSPI are described in Chapter 2. Sensorimotor embedding uses the resulting policy to generate the two dimensional representation of the object's pose as described in Chapter 3. This substantial reduction in dimension makes sensorimotor embedding an ideal method for generating low-dimensional features.

of object pose without having to resort to unsupervised learning from images, or preprogrammed understanding of how to infer object pose. One important area of future work is to develop ways to associate learned object poses for different objects. If the methods of manipulation are substantially similar, then both objects should share the same sensorimotor space, but if actions required to manipulate the objects are different, another method of deducing similar poses across the different objects will be necessary.

6.5 Conclusion

An agent can apply sensorimotor embedding to learn object pose and use that pose information to learn an alignment task. This method performs better than principal component analysis and Isomap. In the following two chapters, sensorimotor embedding will be applied to the problem of learning and using depth features.

Chapter 7

Learning Depth

Depth is an important form of geometric knowledge for both natural and artificial systems. In this chapter, a developmental program learns about depth by applying sensorimotor embedding. The agents described in this chapter have stereo vision and can perform vergence actions. Vergence actions allow an agent to align its eyes on a single shared point of interest. The chapter will show how sensorimotor embedding can be used to learn depth features for agents with stereo vision and vergence capabilities.

7.1 Motivation

In this chapter, a developmental program learns sensorimotor features for depth. Robots can infer depth directly from certain kinds of sensors, such as laser rangefinders or calibrated stereo vision sensors. The developmental program presented in this chapter uses uncalibrated stereo vision sensors to learn depth. These stereo cameras are capable of performing convergence actions, where both cameras rotate to align on a shared point of interest. This is similar to humans, who use vergence cues to infer local depth. The learned depth information can serve as a feature for learning a task as discussed in Chapter 8, or as a basis for learning other, monocular distance cues, as in human visual development [20].

By learning depth cues autonomously, a robot would be able to operate away from any human assistance even in the case where physical changes to the robot would normally require manual recalibration.

7.2 Setup

7.2.1 Vergence

The following experiments use vergence actions, first simulated using stereo pairs of images, then using a robot with articulated cameras. There are several possible approaches to implementing vergence actions. For example, a dominant eye first tracks to a point of interest. This is followed by an action by the subordinate eye to bring the point of interest into binocular alignment. If the simulated eyes are connected as part of the same head mechanism, the initial motion affects both the dominant and subordinate eyes, and the vergence action that follows affects only the subordinate eye.

Another model of vergence involves tracking a point of interest in parallel. Imagine a line emanating from the midpoint between two stereo sensors out towards an object. Where an object of interest is on this mid-line determines the amount of vergence required to bring the object into binocular alignment, meaning that the object of interest is centered in the foveas of both sensors. Symmetric vergence actions bring both eyes into binocular alignment on the object. The degree of vergence depends on the location of the object of interest on the mid-line. For a physical or simulated robot, a third "wide"-angle camera at the midpoint of the foveated cameras on a shared head mount would allow the agent to track a point of interest in two dimensions, followed by a vergence action to bring the point of interest into binocular alignment. This multi-resolution camera approach is similar to the setup found in [21], where the authors used multiple cameras with different resolutions for tracking and object recognition.

For human vision, eye motion consists of *version* and *vergence* components. For the version component of motion, both eyes move in the same direction. For the vergence component of motion, both eyes move in opposite directions. The interaction between version and vergence components of motion is governed by **Hering's Law of Equal Innervation**, which states that the movement of one eye is accompanied by a movement of the other eye of equal amplitude and velocity, either in the same or in the opposite direction [27]. Hering's Law was originally interpreted to mean that the velocities of both eyes are always equal. However, subsequent research has shown that it is not the amplitude or velocity of the movements of the two eyes that are equal, but the amplitude and velocity of the vergence component in each eye and the version component in each eye. The version and vergence components can cancel when combined, leading to unequal movement between both eyes.

This following experiments use the mid-line approach to vergence, though future work would include a more nuanced approach to vergence based on models of version and vergence interactions in humans.

7.2.2 Stereo Pairs

The agent learning depth from real stereo pairs consists of a left and right cameras (similar to the **Roving Eye** robot as described in Chapter 5). These left and right cameras scan along their respective images in opposite directions along the same horizontal line. This approximates mid-line vergence actions. Both the left and right cameras are foveated, using the same model of foveation as described in Chapter 4. The agent controls both cameras and can perform a single action, convergence by one pixel. This experimental setup is shown in Figure 7.1.

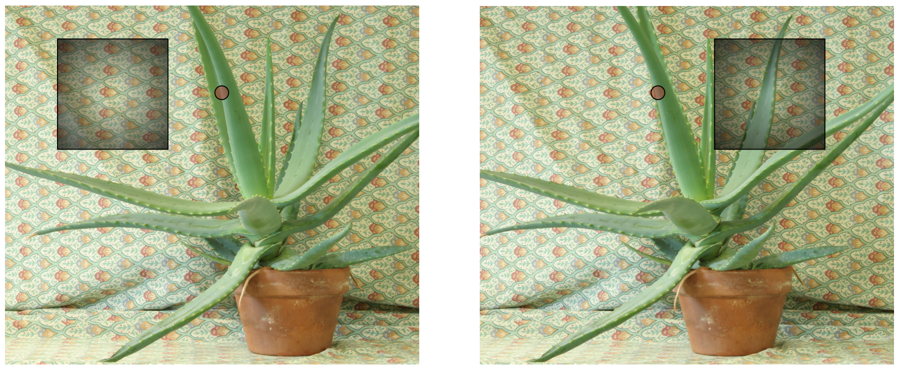


Figure 7.1: **Stereo Pair Vergence**. Two cameras move in opposite directions along the same horizontal line of a pair of stereo images. These cameras are foveated. The circled target is a point of high saliency in both images. The agent controlling the cameras has one available action, convergence by one pixel. This experiment is used to evaluate if an agent using sensorimotor embedding can learn depth. The stereo pair images were taken from the Middlebury Stereo Datasets [25].

A set of shared points of interest were determined in advance using points of high saliency. For each point, the agent's cameras were positioned so that the points of high salience in their respective images were of equal distance from each camera. This could be accomplished by using a third camera located at the midpoint between the two eyes, driven to points of high saliency using the methods described in Chapter 5. The left and right cameras start from a fixed width. In a three dimensional binocular robot, this initial movement would be equivalent to aligning the head to a point of interest in a scene prior to performing any vergence action. The agent followed a policy of converging until a reward was given. Reward was given when the differences between both cameras was minimized.

Since the left and right cameras are foveated, the underlying pixels are not compared when computing difference between images. Instead the components of the fovea, which may contain many underlying pixels, are compared. Each field's value is the average of the underlying pixels. The fovea model used here is discussed in more detail in Chapter 4.

Each field's value in the fovea is given by average activation of all the underlying pixels, so for a field f consisting of a set of pixel intensities $\{p\}$, the activation is given by

$$a = \frac{1}{|f|} \sum_{p \in f} p. \tag{7.1}$$

The difference between both images is then calculated using the field activations as

$$d(r, l) = \sum_i ||f_i^l - f_i^r||. \tag{7.2}$$

where i is the index of a field with the same coordinates in both the left and right retinas. Pixel activations that belong to smaller fields in the high resolution fovea have more of an impact on Equation 7.2 than pixels in the periphery of the camera image.

In Figure 7.2, the fovea filter is applied to a portion of the image. As with the model of foveation described in Chapter 4, salient points will have higher activation in the fovea. In Figure 7.3, the result of saliency applied to the stereo pair is shown. Saliency is highly correlated between both images, allowing the agent to identify shared salient features for vergence actions.

After applying the policy, the next step in sensorimotor embedding is comparing the action traces. In this experiment, the action traces are sequences of vergence actions. The length of each sequence changes with the disparity of the target points. Let $\{a_t\}_{t=1}^n$ and $\{a_t\}_{t=1}^m$ be two action traces consisting of a different number of convergence actions. Let $m < n$. Applying the sequence space metric as described in Chapter 3,

$$\delta(\{a_t\}_{t=1}^n, \{a_t\}_{t=1}^m) = \sum_{t=m+1}^{n} ||a_t||. \tag{7.3}$$

If the vergence actions are $a_t = \pm 1$ then $||a_t|| = 1$. The distance between two action traces in this domain is the difference in action trace lengths. After computing

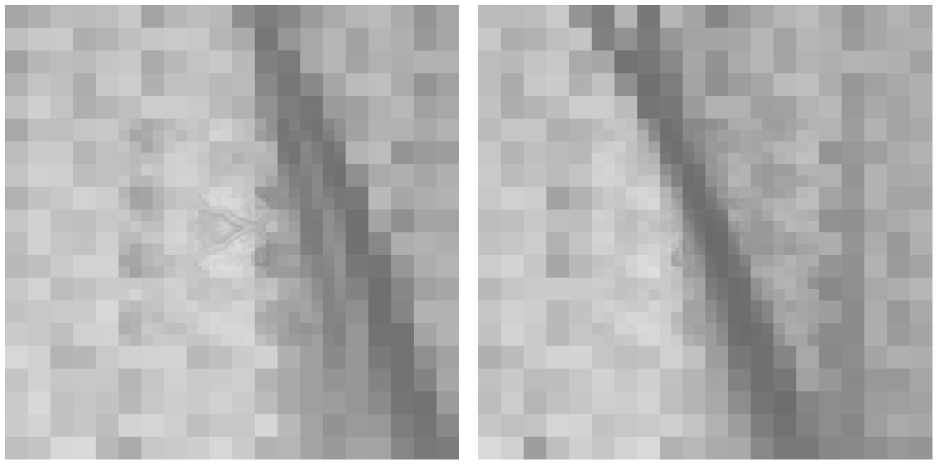
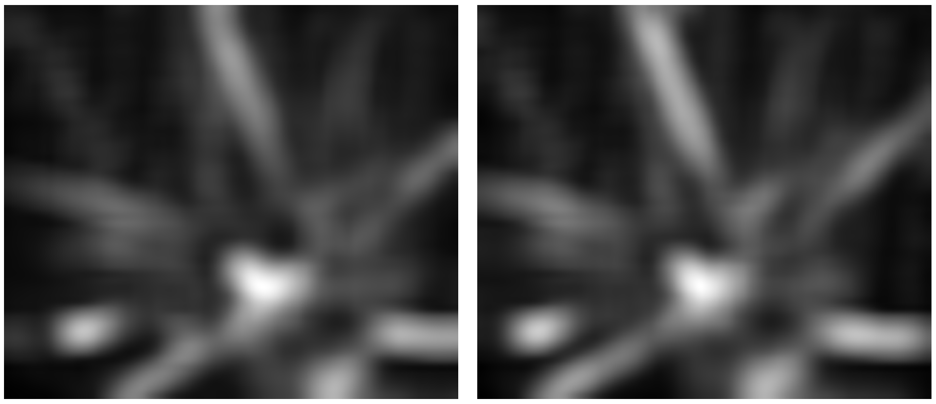

(a) Left Fovea (b) Right Fovea

Figure 7.2: **Foveated Filter Example**. A portion of the aloe image with the fovea filter applied. The fovea model is the same as in Chapter 4. To generate these images, the activation across overlapping layers is averaged together. As the results will show, foveation does not impact an agent's ability to identify correspondences between natural scene images.



(a) Left Saliency (b) Right Saliency

Figure 7.3: **Saliency Map Examples**. A saliency map of the images in the stereo pair used to generate points of interest for the agent. These maps were computed using the method of Itti and Koch. Salient points in one image are highly correlated with salient points in the other image. This allows the agent to identify shared salient features for convergence actions.

distances between action traces, the agent performs multidimensional scaling on the resulting distance matrix. The results of sensorimotor embedding are compared to the true disparities for the points of interest in the stereo pair. The disparity of two corresponding points in a stereo pair is the absolute value of the difference of the horizontal coordinates for the corresponding points. So for a pair of corresponding points with horizontal coordinates x_L and x_R, the disparity is $|x_L - x_R|$. The ground truth disparities are provided as part of the Middlebury Stereo Datasets [25].

7.2.3 Simulation

This experiment uses a simulated robot in Gazebo [32]. The robot moves along a track and has two parallel cameras. In addition to supporting linear motion along the track, the cameras support vergence. Both cameras rotate inward in a coordinated fashion, and converge on a point of joint focus. Three meters in front of the robot is a stop sign that serves as salient visual target for the robot. Saliency in this environment is determined using the method of Itti and Koch. Figure 7.4 shows the simulated robot. The simulation code is available online [68].

In the simulation, the vergence angle changes were restricted to increments of 0.01 radians. The left and right cameras were both foveated, and the difference between the left and right cameras was used to identify the proper vergence angles using Equation 7.2. Like in real stereo images, the minimal image difference occurred when both cameras focused on the salient object in the simulation.

In the simulated setup, the vergence angle required at any position along the track can be computed directly using the law of sines. Figure 7.4 shows the relationship between the vergence angle θ for the left camera and the distance to the target Z. Since this agent uses symmetric vergence actions, the right camera would also move by θ radians in the opposite direction. The baseline B is 0.5 meters. For

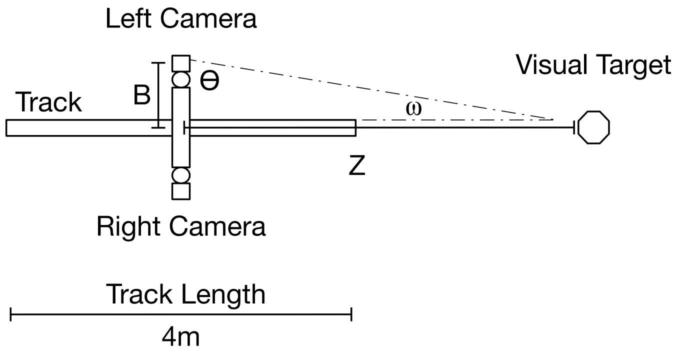


Figure 7.4: **Simulated Gazebo Robot**. The simulated robot in Gazebo. The cameras are mounted on the left and right sides of the robot. The cameras can rotate inward in a coordinated fashion to converge on a point of interest. A stop sign serves as a salient visual target for the robot. The baseline B is 0.5 meters. The distance to the object Z changes depending on the robot position along the track.

distance Z and baseline B, the law of sines gives us

$$\frac{\sin(\theta)}{B} = \frac{\sin(\omega)}{Z}. \tag{7.4}$$

Also, $\theta = \frac{pi}{2} - \omega$, leading to the following formula for the distance Z,

$$\frac{B\sin(\frac{\pi}{2} - \omega)}{\sin(\omega)} = Z. \tag{7.5}$$

An agent applying sensorimotor embedding first learns a vergence policy. In this experiment the policy was learned using Least Squares Policy Iteration (Section 2.2). The reward function provided reward when the difference between the left and right cameras was minimized. The features for the policy were the activations of each element in the foveated cameras. Action traces were collected at different distances from the target object. The resulting action traces were compared by comparing their relative lengths. The derivation of this comparison is the same as in Section 7.2.2. After applying multidimensional scaling to the distance matrix, the resulting one dimensional points were associated with the start states of each action trace.

7.3 Results

7.3.1 Stereo Pairs

Using the Aloe image pair, the difference between the left and right fovea varies as a function of position. In one example case the agent discovered a minimum fovea difference at pixels 320 in the right image and 446 in the left image for a disparity of 126 pixels. The ground truth disparity was 128 pixels. For other salient points in this image pair, the estimated disparities were never more than 2 pixels from the ground truth provided with the dataset.

Examining the minimum locations of the left and right foveas show that they correspond to the same point of interest in both the left and right stereo images. In this example, the agent had to deal with a local minimum along the vergence trajectory due to an aloe leaf that appears in the left image but is occluded in the right image. The fovea location for both these points of interest is shown in Figure 7.6. Despite this local minimum, the minimum fovea difference between the left and right cameras achieved through convergence still results in an accurate disparity measurement.

Since both cameras always began at the same distance from the target, the length of each action trace depended only on the disparity of the target point. A point with the larger disparity would be associated with the shorter action trace and a point with a smaller disparity would be associated with a longer action trace.

For two action traces of different lengths, the difference in lengths was equal to the difference in disparities. An action trace associated with a target point disparity of 120 and an action trace with a target point disparity of 140 would differ by 20 actions. The distance between these traces would be 20. Using Procrustes analysis to compare the learned depths after multidimensional scaling results in an error of .0003. This indicates that the correspondence between learned depth and ground truth is accurate after accounting for changes and scale and rotation.

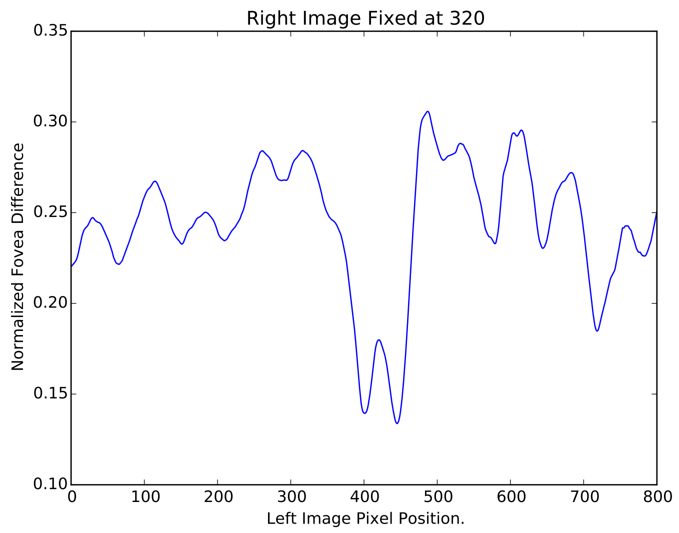
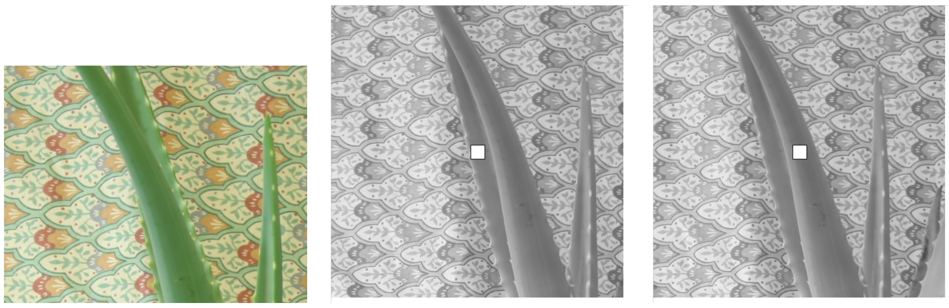


Figure 7.5: **Fovea Stereo Difference**. The difference between the left and right foveas changes as a function of convergence, with a minimum within 2% of the true disparity. Fixing the right fovea on the target and moving the left fovea results in the following plot of left and right foveal difference. Using this approach to compute stereo distance highlights a potential source of aliasing. Note that the false minimum corresponds to a point that is occluded in the left image but visible in right image. The minimum difference between foveated cameras provides a perceptual goal for the agent in this domain.

7.3.2 Simulation

Figure 7.9b shows the result of applying sensorimotor embedding to the vergence action traces generated in simulation. Each trace consisted of a number of convergence actions that rotate each camera by 0.01 radians. These were then compared to each other using the same approach as in the previous section. Multidimensional scaling was applied to the resulting distance matrix, yielding a set of values associated with the depth of the target.



(a) Right Image (b) False Minimum (c) True Target

Figure 7.6: The agent has to deal with occlusions in stereo image pairs. A false minimum and the true target are shown in the middle and right images respectively. The left image shows view of the other camera. The agent can identify the shared salient point of interest since it minimizes the difference between the left and right foveated cameras.

The image difference in the simulated domain occurs at a point in vergence space that matches the ground truth vergence required for binocular alignment (Figure 7.7). This indicates that image difference is an adequate source of information for constructing a reward signal for learning a vergence policy. In this domain, with only a single action, learning such a policy requires only standard methods. Figure 7.8 shows the result of applying least squares policy iteration in this domain. After each policy iteration, the current policy was tested at different distances to see if the agent would correctly converge on a point of interest. After nine iterations, the agent was able to learn a perfect vergence policy.

The angle of vergence is inversely proportional to the distance to the cameras. Since the convergence actions were limited to 0.01 radian increments, the true vergence could only be approximated. For distant focal points, aliasing occurred since the optimal vergence angle in the discrete set of achievable vergence angles was identical for slightly different positions. The ground truth vergence angles are shown

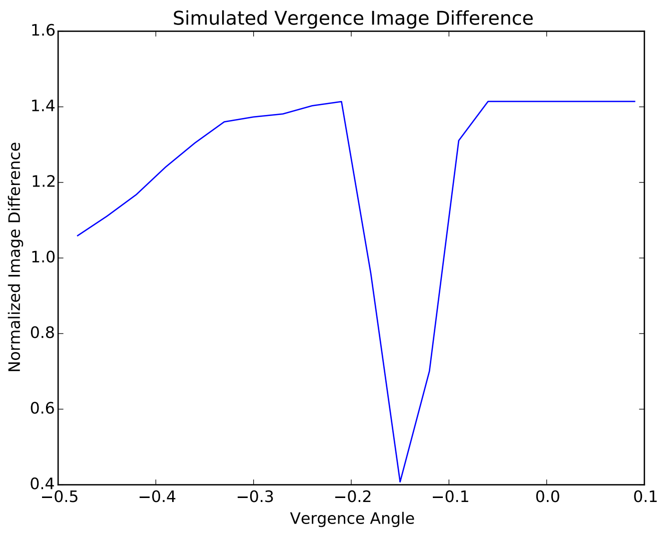


Figure 7.7: **Simulated Image Difference**. Image difference in the simulated domain as a function of vergence angle. The simulated domain does not have a false minimum as was the case in the real stereo pairs example. The minimum difference corresponds to the correct vergence angle for binocular alignment.

in Figure 7.9a. After sensorimotor embedding, the scale of the learned sensorimotor features is different from the actual vergence angles, but the functional relationship was similar (Figure 7.9). The Procrustes error between vergence angles and sensorimotor features was 0.0257, with most of the error attributable to sensorimotor aliasing. Sensorimotor embedding and vergence actions provide the agent a method of generating features related to depth.

7.4 Discussion

Saliency maps play an important role in this chapter. A developing agent needs identifiable perceptual goal states. In early development, the perceptual range of an agent is limited, and so salient goals must be constructed out of simple features.

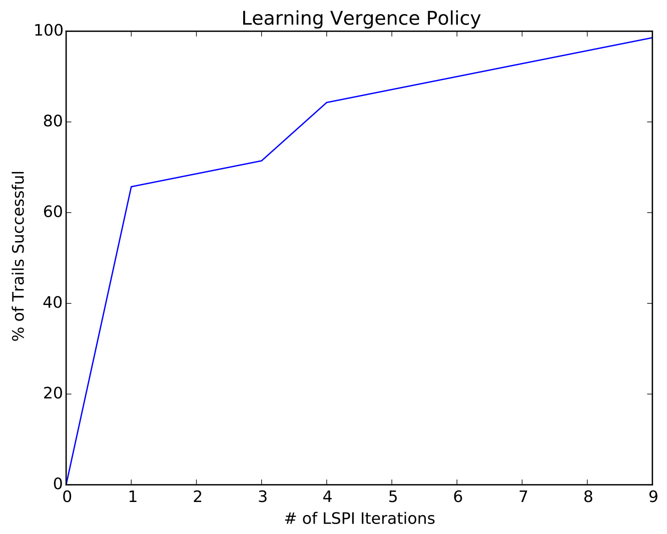
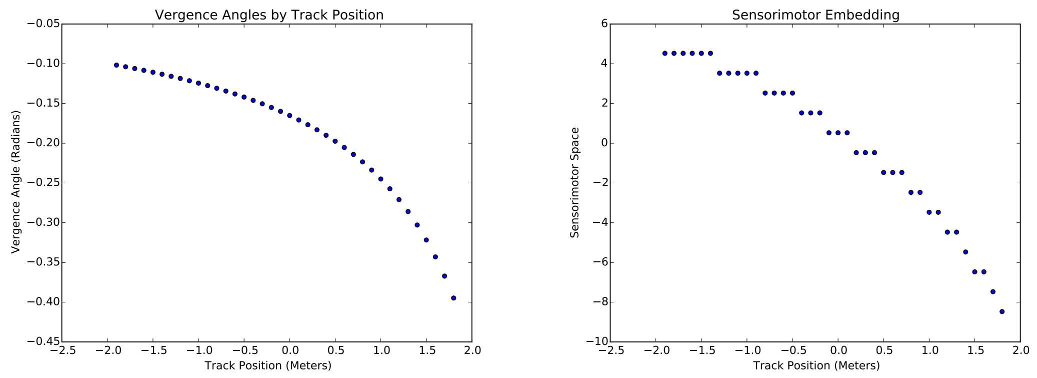


Figure 7.8: **Learning a Vergence Policy**. A vergence policy can be learned using least squares policy iteration. This agent learns the task in nine iterations. Learning a policy is the first step in sensorimotor embedding.

Saliency maps, which have been shown to predict gaze in adults, do so with surprising accuracy using only simple "bottom up" features [30].

Though not central to this thesis, it is worth considering whether sensorimotor embedding would serve as a computational theory of the acquisition of stereo cues in humans. The motor system that controls eye movements is more complicated than the simple action space presented in this chapter, as is the interaction between version and vergence motions.

Sensorimotor embedding analyzes the sequence of motor actions that result in proper vergence instead of using angle measurements directly. For a developmental system, angles would not be available to the agent. Biological systems, like the human eye, have more complicated action space. In humans, the eye is controlled by six extra-ocular muscles (Figure 7.10). The movement subspace corresponding



(a) Ground Truth (b) Sensorimotor Embedding

Figure 7.9: **Sensorimotor Embedding Features in Simulation**. The angle of vergence required is inversely proportional to the distance of the object. For these experiments, the vergence was limited to 0.01 radian increments. For distant focal points, this results in aliasing, since the optimal vergence angle in the discrete set is identical for slightly different distances. The ground truth vergence, calculated using Equation 7.5 angles for the same track positions are shown on the left. Except for a scale change, the functional relationship between position and the sensorimotor features is similar to the ground truth, indicating that sensorimotor embedding is learning an important feature for depth. The Procrustes error is 0.0257 when comparing points in vergence space to points in sensorimotor space with the same track position.

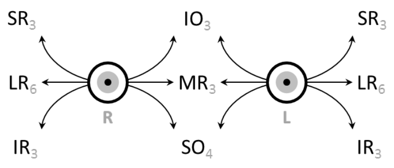


Figure 7.10: **Eye Musculature**. This graphic shows the mapping between six extraocular muscles and three cranial nerves that control eye movements. Unlike the simple action spaces used in the experiments presented in this chapter, the human ocular muscle system is more complex, with six muscle groups working in a coordinated fashion to perform a variety of visual actions.

to vergence involves the coordinated movement of a set of muscles. In this case the internally available action trace for sensorimotor embedding would be the specific nerve stimuli corresponding to the vergence actions. As long as these nerve impulses can be compared in a reasonable way, sensorimotor embedding can be applied.

Whether or not sensorimotor embedding is a viable computational theory of learning depth in humans, the experiments in this chapter show that sensorimotor embedding can be applied by robots to learn depth features.

7.5 Conclusion

An agent can learn about depth using sensorimotor embedding. Applying sensorimotor embedding requires learning a vergence policy, applying that policy to generate action traces, then applying multidimensional scaling to those action traces. Though the policies and action traces are simple compared to previous experiments, sensorimotor embedding can be applied unmodified to learn yet another useful geometric feature. The next chapter will demonstrate how an agent can learn control policies based on these depth features.

Chapter 8

Visual Mountain Car

The goal in the mountain car is to drive an under-powered car out of a valley in the shortest amount of time. The problem is easily stated, but complex enough that it has become a classic test for reinforcement learning algorithms. The state of the car (position, velocity) in a typical mountain car problem is provided to a reinforcement learning agent. In the visual mountain car problem, **the agent is not provided with position or velocity**, but must instead identify the relevant state through visual inspection of the surrounding environment. Like an isolated lander in peril, or an independently developing agent, the acquisition of positional knowledge must be autonomous.

8.1 Motivation

In this experiment, a developmental program learns sensorimotor features that stand in for the position and velocity of the mountain car. The agent uses that information to solve the problem using reinforcement learning. Unlike Procrustes analysis on learned geometry presented in previous experiments, learning useful sensorimotor features demonstrates that the geometric concept learned by the agent is useful in solving a complex control task. For visual mountain car, the learned features are a functional proxy for position and velocity. Note that the goal of the agent is not to learn **true** positions or velocities, since learning those would require reference to an external fixed scale. For many autonomous applications, the way a robot represents these features internally need only be sufficient to allow the robot to learn or relearn required skills.

The developmental program described here can run on multiple different morphologies without changing the parameters or content of the program. Only the robot's own perceptual experience changes from experiment to experiment. Integrating sensorimotor embedding with developmental programming to solve a reinforcement learning task on multiple different robot morphologies demonstrates the robust nature of this approach. It is the only truly calibration-free methods of extracting geometric features for solving control tasks.

8.2 Setup

The visual mountain car is implemented as a simulated robot in Gazebo [32]. The robot moves along a track and has two parallel cameras. In addition to supporting linear motion along the track, the cameras support vergence: Both cameras can rotate inward in a coordinated fashion, and converge on a point of joint fixation. Three meters in front of the robot is a stop sign that serves as salient visual target for the robot. Our approach to saliency in this environment is that of Itti and Koch [30]. A diagram of the simulation is included in Figure 7.4.

The learning process is divided into stages. Prior to learning depth cues, the robot can learn the structure of the cameras as in Chapter 4. During the next stage the agent learns to represent position by learning a vergence policy and then applying the policy at different positions along the track. The resulting action traces are compared using the methods described in Section 3.1. After applying multidimensional scaling to the action trace distance matrix, action traces (and corresponding start states) are associated with a one dimensional set of points. These points are the features that allow the agent to represent position. After this stage of learning the agent associates with each stereo image start state with a single one-dimensional number. This number is a proxy for position. Chapter 7 describes this stage in detail.

The agent models velocity by tracking the change in its own internal representation over time [70]. Position and velocity are used as the input to the reinforcement learning agent tasked with solving the mountain car task.

The mountain car dynamics are inspired by the problem of driving an under-powered car out of a valley. Because the car is under-powered, moving only forward will not succeed. Any policy that works needs to oscillate between moving forward and backward to build up enough momentum to reach the goal. The position and velocities are updated in mountain car according to

$$v_{t+1} = v_t + ap + A\cos(\omega x_t + \phi) \tag{8.1}$$

$$x_{t+1} = x_t + v_{t+1}. \tag{8.2}$$

In this equation, $a \in \{-1, 0, 1\}$ is the action corresponding to $\{forward, neutral, reverse\}$. The variables p, ω, ϕ, and A control the power of the car and force of gravity in the simulation. In these experiments these were set to

$$p = 0.001 \tag{8.3}$$

$$\omega = \frac{\pi}{4} \tag{8.4}$$

$$\phi = \frac{\pi}{2} \tag{8.5}$$

$$A = 0.0025. \tag{8.6}$$

The parameters ω, ϕ, and A are the angular frequency, phase, and amplitude for the cosine function. The position of the robot is limited to the track, so $x \in (-2, 2)$. The velocity is also limited to $(-0.07, 0.07)$. The robot completes the task if it can reach the goal state, defined in this task to be $x > 1.9$. This task is shown in Figure 8.1

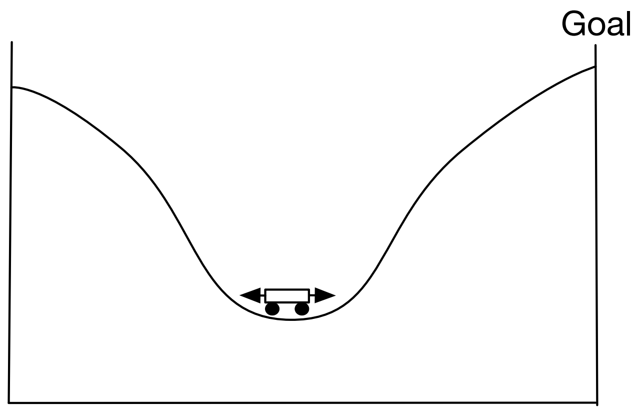


Figure 8.1: **Mountain Car Domain**. The mountain car domain is a classic problem in reinforcement learning. In this problem, an underpowered car has to escape from a valley. The car dynamics are shown in Equations 8.1 and 8.2. Sensorimotor embedding is used to find policies for a variant of the mountain car problem where the agent receives visual input and must deduce position and velocity.

After learning features that model position and velocity, the agent learns to solve the corresponding mountain car using as input these internal features. The policy is learned using Sarsa with CMAC function approximation [41]. The learning rate is set to 0.3 with gamma and lambda parameters set to .99 and .9 respectively. The CMAC has two levels with a resolution of 0.1. The CMAC implementation is available online [67].

The complete developmental program that learns position in the visual mountain car simulator has several steps. The first two steps are covered in previous chapters on learning sensor geometry and depth:

1. Learn the geometry of the cameras (Chapter 4).
2. Learn a vergence policy using the cameras and apply sensorimotor embedding to vergence action traces to estimate position (Chapter 7).
3. Solve the mountain car policy using learned sensorimotor features.

This entire developmental program is then run **without any changes** on two alternative robot morphologies, a robot with a longer baseline between cameras (1.4m instead of 1m) and a robot with vertically oriented cameras (Figure 8.3). The only difference between these experiments is the perceptual experience of the robot. The learning process and parameters are identical across all the morphologies. This approach performs well on these different platforms, and demonstrates that sensorimotor embedding provides a robust method of learning geometry even when the underlying robot geometry changes.

8.3 Results

Unlike previous experiments, the agent is not evaluated on how well it learns position, but instead on how well the learned sensorimotor features perform as a basis for learning the mountain car task. To evaluate the learning agent, the learning process is halted every 10,000 training steps and the current best policy for the agent is used to solve the mountain car problem at 100 randomly selected states. The average number of steps to reach the goal is used as the performance metric. This evaluation is repeated 10 times to produce Figures 8.2 and 8.4. A random agent is also evaluated in order to provide a benchmark for the increase in performance due to learning. An agent trained directly using position and velocity is also presented, and provides an example of the ideal learning performance in this task. The results of this evaluation are shown in Figure 8.2.

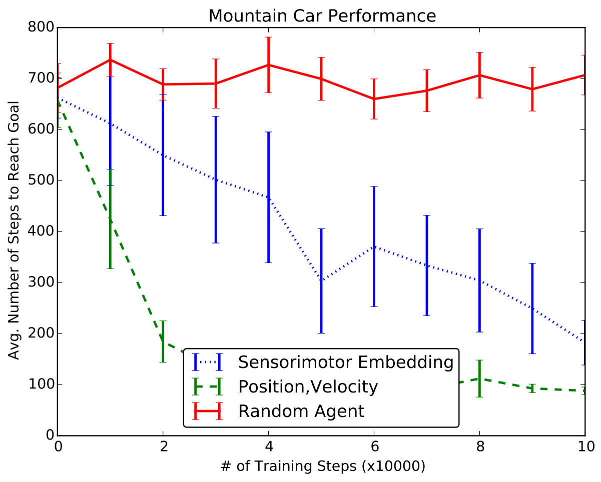


Figure 8.2: **Sensorimotor Features for Mountain Car**. The average number of steps required to reach the goal state is plotted as a function of the number of training steps for a random agent, an agent using sensorimotor features, and an agent using traditional position and velocity. The agent using sensorimotor features learns slower than an agent that has access to the position and velocity, but over longer training periods the performance is comparable.

The agent that uses sensorimotor embedding is worse than an agent with access to the true position in velocity in two ways. First, the agent performance at each interval is more variable. Second, these agents, on average, require more training to achieve a comparable level of performance with the ideal agent. Both of these issues result from the aliasing introduced by the sensorimotor embedding process described in Chapter 7. Since vergence actions are discrete movements, the actual degree of vergence captured by sensorimotor embedding is a discrete value of limited resolution. This aliasing of nearby positions becomes more pronounced further from the target. The accuracy of vergence as a feature is also known to decline with distance in humans, and is widely seen as a reason that depth perception in humans involves many additional monocular image features.

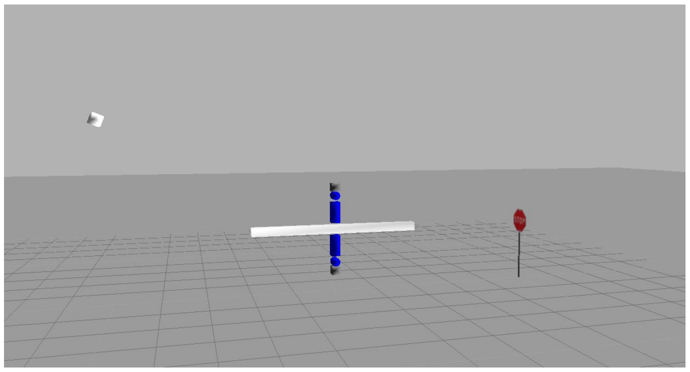


Figure 8.3: **Alternate Robot Morphologies**. The same learning process can be run on a robot with a different morphology. Here the cameras are mounted vertically on the track instead of horizontally. The developmental program runs on this agent without any changes, demonstrating that sensorimotor embedding generalizes to different robot shapes.

Despite the aliasing, the learning agent is able to learn a policy for solving the mountain car task using only the sensorimotor embedding features. Since the sensorimotor embedding approach runs a developmental program to achieve this result, the program can easily be rerun on robots with different morphologies to achieve the same result. In Figure 8.4, the performance of this developmental program is evaluated on a robot with a wider baseline between cameras and on a robot where the cameras are vertically aligned instead of horizontally aligned (Figure 8.3). Performance is the same across all morphologies, demonstrating that sensorimotor embedding generalizes to different robot shapes.

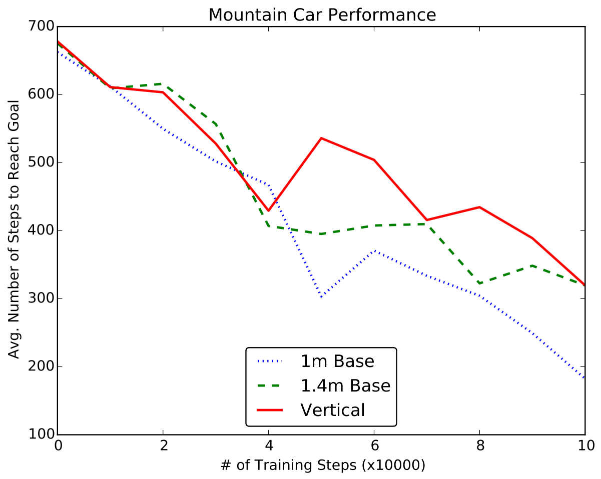


Figure 8.4: **Alternate Robot Performance**. The average number of steps required to reach the goal state is shown as a function of the number of training steps for three different robot morphologies, the standard robot, a robot with a wider baseline, and a robot with vertical cameras. The performance differences are not statistically significant, and the results indicate that sensorimotor embedding generalizes to different robot shapes.

8.4 Discussion

The increase in baseline is meant to evoke the idea of a robot that can physically grow in size. The change in orientation of the cameras themselves is a more drastic change which shows the versatility of this approach on a wide variety of possible robot configurations. As mentioned in Chapter 1, a key motivation for this work is in applications where autonomous robots are far from any lab environment, and may need to learn or re-learn basic skills. In the case of a change in morphology, how does an agent know to relearn these basic features and skills? For an autonomous lander, the detection of an anomalous event can be made by remote operators, and rerunning the developmental program can be triggered remotely.

In the case of a robot without that kind of human guidance, relearning could be triggered using the same reinforcement signal used in sensorimotor embedding. As the morphology changes, the performance of existing learned policies will change, leading naturally to retraining of the underlying policies. The agent has only to detect this change in performance, and rerun the batch component of sensorimotor embedding. A version of this approach is presented in Chapter 4, where the agent must relearn geometry after a lesioning event. In this case, the task is better described as punctuated, instead of non-stationary, as the change being modeled is sudden and the optimal policies on both sides of the change do not change. It remains an open question how best to integrate the batch components of sensorimotor embedding for non-stationary tasks that change gradually over time, such as would be the case for a robot whose morphology is slowly changing with age and wear.

8.5 Conclusion

As the results in this chapter show, useful geometric features can be learned using sensorimotor embedding. Agents that learn these features can use these features to solve control problems. The same developmental program that learns geometric features and uses those features to learn a control policy can be run unchanged on a variety of robot morphologies.

Chapter 9

Discussion and Future Work

"Either this is madness or it is Hell." "It is neither," calmly replied the voice of the Sphere, "it is Knowledge; it is Three Dimensions: open your eye once again and try to look steadily." –Edwin Abbott Abbott, *Flatland: A Romance of Many Dimensions*

This chapter discusses the results of the dissertation in aggregate. New avenues for future research are presented along with potential improvements to the existing algorithm and applications. They include extending sensorimotor embedding to handle probabilistic transition functions and policies, discovering depth and other visual features using human eye models, applying sensorimotor embedding to the problem of option discovery, and learning how to communicate sensorimotor features.

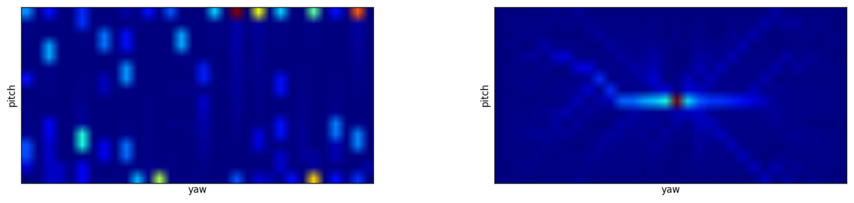
9.1 Perceptual Goals and Development

The results presented in this thesis depend on having achievable perceptual goals in each domain that are not too difficult for an agent to achieve during early sensorimotor development. In human development, reflex actions reduce the complexity of the policy search space. The analogue to reflex actions in developmental robotics are simple routines that augment and organize raw early motor experience. In artificial developmental systems, both the perceptual goals and the simple actions have to be specified in advance. The relationship between perceptual goals, reflex actions, and high level skills is murky, especially in the case of human development. A general guideline for building developmental programs is to separate the hard problem of development into stages of simpler problems.

With development it is possible to adapt autonomously to changes as was demonstrated in this dissertation. In Chapter 1, the plight of the Rosetta space probe was discussed. The probe's lander bounced several times before coming to rest at an unknown location on the asteroid's surface, cutting that portion of the mission short due to a related power failure. Consider what an autonomous probe could achieve if it had the ability to adapt in that distant environment. It would adapt to damage, relearn basic skills, reacquire relevant geometric features, and reorient itself so as to power its solar cells and continue its mission. Sensorimotor embedding provides one component of a developmental program that would be able to adapt autonomously millions of miles away.

9.2 Cognitive Models of Geometry

Sensorimotor embedding is a general method of acquiring geometric knowledge. An important question for future work is whether sensorimotor embedding would also serve as a model of human development. There is evidence from developmental psychology to support the idea that perceptual goal states exists in a variety of situations that a developing infant experiences. For example, the view bias figures presented in Section 3.3 show that view bias increases with age (Figure 3.3). When applying sensorimotor embedding to the object pose problem in Chapter 6, the agent demonstrates a similar evolution in view bias while learning a policy to learn the



(a) Early Training View Bias (b) Late Training View Bias

Figure 9.1: **View Bias Changes During Training**. An agent running sensorimotor embedding acquires a pronounced view bias as the policy for acquiring the perceptual goal state improves. The left image shows the initial distribution over object views. The right image shows the distribution over object views at the end of training the policy. This change in view bias is also observed in developing infants (see Section 3.3).

perceptual goal in that domain (Figure 9.1).

For sensorimotor embedding to be a viable model for cognitive science, it would have to provide a testable hypothesis. There is research that shows that early perceptual experience impacts the development of perceptual skills [6] and research that shows that early motor experience impacts later motor skill acquisition [53]. Does early motor experience impact perceptual development? Bushnell and Boudreau proposed a connection between motor development and changes in perceptual skills, particularly the impact of motor development on the timing of developmental changes. Sensorimotor embedding, as a concrete algorithmic theory for learning geometry, depends on learning motor skills in order to acquire perceptual knowledge, and so is a concrete realization of this idea. Finding a way to test whether sensorimotor embedding works as a theory of cognition is another potential area of future work.

9.3 Human Models of Eye Motion

Chapter 7 presented results for an agent learning depth features from vergence actions using sensorimotor embedding. The model of vergence presented in that chapter does not capture the complexity of human eye movements. For humans, a combined movement of the two eyes in the same direction is known as a version movement and the eyes can move together laterally, vertically, or in an oblique direction. A movement of both eyes in opposite directions is a vergence movement. In horizontal vergence, the visual axes move within a plane containing the interocular axis. The motion that brings images of objects at a particular distance into clear focus is known as accommodation.

One key feature of the human visual system is that multiple different actions tend to occur simultaneously in a coordinated fashion. For example, horizontal vergence and accommodation normally occur together. These two responses are accompanied by an appropriate change in pupil diameter. The three concomitant changes are known as the near-triad response [27]. These complexities, combined with properties that govern version and vergence interaction such as **Hering's Law of Equal Innervation**, present a unique challenge for sensorimotor embedding.

Separating the actions that are relevant for specific geometric features from a set of simultaneous actions is a challenge for sensorimotor embedding with human visual motion models. For example, version actions may be relevant for ego-centric coordinates, while vergence actions provide depth cues. Accommodation and pupil dilation, however important for clarity of vision, do not contribute to either of these geometric features. Stepping back, it seems clear that human vision plays an important role in human knowledge of geometry. The sensorimotor hypothesis is that the actions the visual system performs to bring objects into binocular focus are a key source of geometric information. Developing sensorimotor embedding to a point where this hypothesis can be rigorously explored is an important avenue of future work.

9.4 Probabilistic Sensorimotor Embedding

In sensorimotor embedding discussed in Chapter 3, both the transition function and policy were deterministic functions. However, in general Markov Decision Problems discussed in Chapter 2, the transition functions and policies are probabilistic functions. A transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ defines a distribution over possible transitions from state-action pairs to successor states. Agents should also be able to adopt a random policy, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, that defines a distribution over actions, of which one is sampled when running the policy. Stochastic policies have been shown to perform better than deterministic policies in certain cases [40].

Sensorimotor embedding compares action traces generated by following policies. Random transitions or random policies can generate multiple distinct action traces from the same starting position. To account for the multiplicity of traces, sensorimotor embedding needs to be extended to handle comparing action traces drawn from a distribution. There are two possible approaches:

1. An agent samples multiple action traces with the same start state then uses the mode of the sample as the canonical action trace for comparison.

2. For each pair of start states, the agent samples action traces for both start states, and tracks the average distance computed over multiple samples.

In situations where transitions are stochastic but the policies are ballistic, such as the pan/tilt camera experiment discussed in Chapter 4, a sampling approach is not needed, since the action trace produced by the policy has length one, and the policy itself is deterministic. For the tasks presented in this dissertation, the policy for achieving perceptual goal states, if optimal, should tightly control any randomness inherent in the transition function for the domain, and should reduce the need to sample many action traces in order to generate accurate action trace distance estimates.

Probabilistic sensorimotor embedding will allow agents to acquire geometric features in uncertain domains or when using stochastic policies, and will expand the applications of sensorimotor embedding beyond the domains presented in this dissertation.

9.5 Variations of Sensorimotor Embedding

Sensorimotor embedding depends on learning a policy, comparing action traces, and finding a low-dimensional representation with the same distance relationships as the action traces. Variations on sensorimotor embedding could explore alternatives for each of these components. For example, multidimensional scaling is used to find low-dimensional coordinates. This algorithm could be replaced by a non-linear manifold learning method. Action traces are compared using the metric described in Chapter 3, but other methods of comparing action traces are also possible. Dynamic time warping might be a useful method of comparing action traces with continuous action parameters and varying sample times. Since dynamic time warping is not a metric, the formal properties of sensorimotor embedding would need to be relaxed [43].

Another variation worth exploring is incorporating state information in the later stages of sensorimotor embedding. State information is used by the policy to generate action traces, and the current approach extracts geometric features only from the action traces. Though action traces contain a significant amount of implicit information about geometry, being able to incorporate state traces as well may result in improved accuracy and help to disambiguate aliased action traces, e.g. the aliased vergence traces in Chapter 7.

9.6 Visualization and Option Discovery

Manifold learning methods are commonly used to visualize complex data [34]. Sensorimotor embedding, as a manifold learning method, can be used to visualize policies. As an example, a simple agent was trained on the mountain car task (Figure 8.1). After training, the agent started at random positions and velocities and the resulting action traces were recorded. Applying the sensorimotor embedding to these action traces results in three distinct clusters (Figure 9.2). Inspecting the clusters reveals that sensorimotor embedding divides the action traces into three categories; traces that result from failure to complete the task, traces that result from opportunistic starting positions, and traces that rock the car back and forth to escape the valley.

In addition to visualization, sensorimotor embedding can be used to discover useful options. Options are complex actions or subroutines agents can run that simplify the search for good policies. Given a set of action traces generated using a trained policy, a set of options can be discovered by first dividing the action traces into small fixed length action sequences. These sequences can be compared using sensorimotor embedding. Clustering the resulting low-dimensional representation of the trace segments and taking the pre-image of each cluster mode generates a set of candidate options for inclusion in the agent action set. These sequences of actions are options that simplify learning a policy.

This approach was applied in the mountain car task (Figure 8.1). Traces were divided into action sequences of length five. After clustering, sequences corresponding to the mode of each cluster were included as options for training a new agent on the same task. Figure 9.3 shows the learning performance over time for an agent using random options, options discovered through sensorimotor embedding, and no options. The agent with options discovered using sensorimotor embedding learned the task faster than the agent with no options. Providing a set of randomly generated options disrupted the learning process of the agent in this domain.

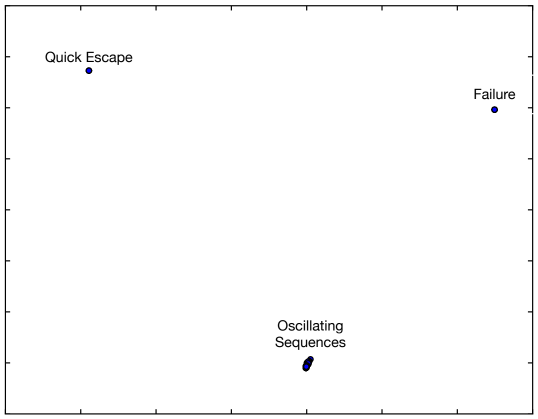


Figure 9.2: **Policy Visualization Example**. Sensorimotor embedding can be applied to action traces for the purpose of visualization. Here an agent was trained to solve the mountain car problem using Sarsa(λ) as described in Chapter 2. After training, traces were recorded for 1000 starting random starting points in the task. The low dimensional representation of the traces was generated using sensorimotor embedding. The clusters represent failed traces, opportunistic starting locations, and traces that involve rocking back and forth to escape the valley. This example shows how sensorimotor embedding may be useful for visually inspecting how policies behave.

As these examples demonstrate, sensorimotor embedding for visualization and option discovery is a promising direction for future work.

9.7 Communicating Geometric Knowledge

This dissertation demonstrates how an agent can use geometric knowledge to solve a task. Autonomous agents working together need to share geometric knowledge about the environment. Being able to communicate about geometric features of the local environment would also be useful for cases like Rosetta's lost lander, where a description of the local landscape with ground control would have helped diagnose the probe's situation. One essential element of communication is discovering shared reference distances. Though individual agents can learn grounded geometric features of the environment using sensorimotor embedding, this knowledge is tied to each agent's individual policies and actions. Finding a method for multi-agent systems to discover how to communicate about learned geometric features is another promising direction for future work.

9.8 Conclusion

Several directions for future work were discussed, including using realistic models of eye motion, extending sensorimotor embedding to stochastic domains, and using sensorimotor embedding for visualization and option discovery. The next chapter summarizes the contributions in this dissertation, and concludes.

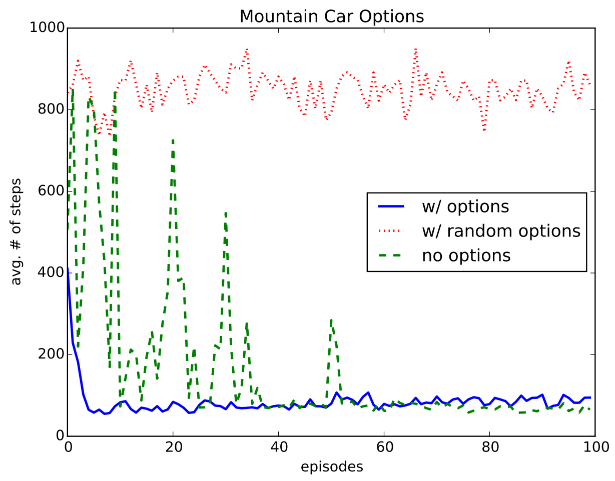


Figure 9.3: **Option Discovery with Sensorimotor Embedding**. Sensorimotor embedding can be used to learn useful options. In this experiment an agent was trained to solve the mountain car task (Figure 8.1). After training, action traces were recorded at random starting positions. Subsequences of length five were extracted from these traces. Sensorimotor embedding was applied to these trace segments and the resulting low-dimensional points were clustered. The pre-image of the cluster modes were used as options for training a new agent. To compare, agents were trained with no options and with the same number of randomly generated options. The agent with options generated using sensorimotor embedding learned the task faster than the other agents. This shows that sensorimotor embedding can be used for option discovery.

Chapter 10

Conclusions

I'm beginning to feel at home with these prisms. Reaching is accurate for at least the most familiar and practical tasks, say reaching for particular door handles. –Hubert Dolezal *Living in a World Transformed*

This thesis focuses on the discovery of geometric knowledge. By developing knowledge of geometry over time using sensorimotor embedding, agents do not have to depend on knowledge encoded in the agent's programming, and can adapt to unexpected changes. This method eschews complex calibration schemes and adopts an approach that leverages the embodied experience of a developing agent. In key scenarios encountered by a developing agent, this approach performs better than principal component analysis and other methods of dimensionality reduction, and is general enough to run on robots with different morphologies with no configuration. This final chapter reviews the contributions of this thesis.

10.1 Contributions

The main technical contribution, sensorimotor embedding, was described in detail in Chapter 3. Sensorimotor embedding is a developmental approach to learning about geometry. The approach involves first learning a policy for achieving a perceptual goal state, comparing the resulting action traces, and generating a low-dimensional representation of the associated start states by applying multidimensional scaling to the action trace distances. Sequences of actions encode information about geometric features of the environment, and sensorimotor embedding provides a principled method of extracting that information and making it available to the developing agent. Chapter 3 also discusses how to best evaluate sensorimotor embedding and other manifold-learning methods and suggests that Procrustes analysis is a useful tool.

The main experimental chapters demonstrate how sensorimotor embedding can be used to learn geometric features in different domains. Four important domains were studied. In each case the agent was able to learn geometric features from experience in a robust and general way without the need for manual calibration. Chapter 4 shows how sensorimotor embedding can be applied to learn the geometry of a foveated sensor. The learning process was shown to adapt to sensor changes including sensor lesions and image inversion, both changes that would cripple most robots. Chapter 5 applies sensorimotor embedding to learn positions of states in gridworlds and the position of a roving-eye robot. Chapter 6 uses sensorimotor embedding to learn the pose of an object. Chapter 7 learns the features for depth using vergence policies. Each of these domains contains useful geometric knowledge that sensorimotor embedding is able to discover. Sensorimotor embedding works in dramatically different domains, demonstrating that sensorimotor embedding can be applied successfully in a wide variety of different situations.

Chapter 8 shows how these geometric features can be brought together and used as inputs for a higher level learning process. In this chapter an agent solves the Visual Mountain Car task, a variation of a popular control problem where the agent is not provided with state information, but must instead infer that information from visual experience. To solve this task the agent first learns to estimate its position on the track using sensorimotor embedding. Using this inferred position, the agent is able to solve the task using reinforcement learning. The robust nature of sensorimotor embedding is also tested by running the same developmental program on robots with different morphologies. These experiments demonstrate that sensorimotor embedding can serve as a foundation for adapting to a changing world, and for high-level learning.

10.2 Conclusion

This dissertation demonstrates many applications of sensorimotor embedding. In each case a developing agent was able to learn geometric features of the environment. That knowledge was tested against ground truth using Procrustes analysis, used as a feature for solving a control problem, tested through lesion and inversion experiments, and tested on robots with different morphologies. Agents can learn about the geometry of sensors, about the geometry of the local environment, and about the pose and depth of objects. The algorithmic ideas presented in this dissertation provide an answer for how agents can come to know, understand, and use geometric knowledge in a robust and general way.

Bibliography

[1] Kirsti Andersen. *The Geometry of an Art: The History of the Mathematical Theory of Perspective from Alberti to Monge*. Sources and Studies in the History of Mathematics and Physical Sciences. Springer, London, 2007.

[2] Emily Baldwin. Did philae graze a crater rim during its first bound? http://blogs.esa.int/rosetta/2014/11/28/did-philae-graze-a-crater-rim-during-its-first-bounce/, 2014.

[3] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is “nearest neighbor” meaningful? In *International Conference on Database Theory*, pages 217–235. Springer, 1999.

[4] Chistopher Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[5] Pratik Biswas, Tzu-Chen Liang, Kim-Chuan Toh, Yinyu Ye, and Ta-Chung Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE Transactions on Automation Science and Engineering*, 3(4):360–371, 2006.

[6] Colin Blakemore and Grahame Cooper. Development of the brain depends on the visual environment. 1970.

[7] Byron Boots, Arunkumar Byravan, and Dieter Fox. Learning predictive models of a depth camera & manipulator from raw execution traces. In *IEEE International Conference on Robotics and Automation*, pages 4021–4028. IEEE, 2014.

[8] B. Borchers. CSDP: A C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.

[9] M. Bowling, D. Wilkinson, A. Ghodsi, and A. Milstein. Subjective localization with action respecting embedding. *Robotics Research*, 28:190–202, 2007.

[10] Michael Bowling, Ali Ghodsi, and Dana Wilkinson. Action respecting embedding. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 65–72, New York, 2005. ACM.

[11] Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A course in metric geometry*, volume 33. American Mathematical Society Providence, 2001.

[12] Andrea Censi. *Bootstrapping vehicles: a formal approach to unsupervised sensorimotor learning based on invariance.* PhD thesis, California Institute of Technology, 2013.

[13] Y Choe and N H Smith. Motion-Based Autonomous Grounding: Inferring External World Properties from Encoded Internal Sensory States Alone. *Proceedings of the National Conference on Artificial Intelligence (AAAI-2006)*, 2006.

[14] Trevor F. Cox and Michael A. A. Cox. *Multidimensional Scaling.* CRC Press, 2001.

[15] Hubert Dolezal. *Living in a World Transformed: Perceptual and Perfomatory Adaptation to Visual Distortion.* Academic Press, 1982.

[16] Ian L Dryden and Kanti V. Mardia. *Statistical Shape Analysis.* Wiley, New York, 1998.

[17] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[18] Bernd Fritzke. A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, pages 625–632, 1995.

[19] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. *Advances in Neural Information Processing Systems*, 17:513–520, 2005.

[20] E Goldstein. *Sensation and perception.* Cengage Learning, 2013.

[21] Stephen Gould, Joakim Arfvidsson, Adrian Kaehler, Benjamin Sapp, Marius Messner, Gary R Bradski, Paul Baumstarck, Sukwon Chung, and Andrew Y Ng. Peripheral-foveal vision for real-time object recognition and tracking in video. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 7, pages 2115–2121, 2007.

[22] John C Gower and Garmt B Dijksterhuis. *Procrustes problems*, volume 3. Oxford University Press Oxford, 2004.

[23] K.L. Harman, G. Keith Humphrey, and M.A. Goodale. Active manual control of object views facilitates visual recognition. *Current Biology*, 9(22):1315–1318, 1999.

[24] Justin Wildrick Hart and Brian Scassellati. Mirror perspective-taking with a humanoid robot. In *AAAI*, 2012.

[25] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[26] Radu Horaud, Fadi Dornaika, and Bernard Espiau. Visually guided object grasping. *IEEE Transactions on Robotics and Automation*, 14(4):525–532, 1998.

[27] Ian P Howard. *Binocular vision and stereopsis.* Oxford University Press, 1995.

[28] Yoshito Ikemata, Akihito Sano, Kiyoshi Yasuhara, and Hideo Fujimoto. Dynamic effects of arc feet on the leg motion of passive walker. In *IEEE International Conference on Robotics and Automation*, pages 2755–2760, 2009.

[29] L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, 2001.

[30] L. Itti, C. Koch, and E. Niebur. A model of saliency-based fast visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, November 1998.

[31] K.H. James, G.K. Humphrey, and M.A. Goodale. Manipulating and recognizing virtual objects: Where the action is. *Canadian Journal of Experimental Psychology*, 55:111–120, 2001.

[32] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[33] T. Kohonen. *Self-Organizing Maps.* Springer, 2004.

[34] W.J. Krzanowski. *Principles of Multivariate Analysis: A User's Perspective.* Oxford University Press, 2000.

[35] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[36] Honglak Lee, Chaitanya Ekanadham, and Andrew Ng. Sparse deep belief net model for visual area V2. *Advances in neural information processing systems*, 20:873–880, 2008.

[37] Fan Lu, Sündüz Keleş, Stephen J Wright, and Grace Wahba. Framework for kernel regularization with application to protein clustering. *Proceedings of the National Academy of Sciences of the United States of America*, 102(35):12332–12337, 2005.

[38] S. Mahadevan and M. Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in markov decision processes. *The Journal of Machine Learning Research*, 8:2169–2231, 2007.

[39] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[40] T Michael and I Jordan. Reinforcement learning algorithm for partially observable markov decision problems. *Proceedings of the Advances in Neural Information Processing Systems*, pages 345–352, 1995.

[41] W. Thomas Miller and Filson H. Glanz. The university of new hampshire implementation of the cerebellar model arithmetic computer. 1996.

[42] J. Modayil. Discovering sensor space: Constructing spatial embeddings that explain sensor correlations. In *International Conference on Development and Learning*, 2010.

[43] M. Müller. *Information Retrieval for Music and Motion.* Springer, 2007.

[44] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[45] L. Olsson, C. Nehaniv, and D. Polani. From unknown sensors and actuators to actions grounded in sensorimotor perceptions. *Connection Science*, 18(2):121–144, 2006.

[46] J K O'Regan and A Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, 24(5):939–972, 2001.

[47] M. Pagel, E. Maël, and C. von der Malsburg. Self Calibration of the Fixation Movement of a Stereo Camera Head. *Autonomous Robots*, 5(3):355–367, 1998.

[48] S.E. Palmer. *Vision Science: Photons to Phenomenology.* MIT Press, Cambridge, MA., 1999.

[49] A. F. Pereira, K. H. James, S. S. Jones, and L. B. Smith. The origins of human visual object recognition: Early biases and developmental changes in self-generated object views. *Journal of Vision*, 2010.

[50] D. Philipona, J. K. O'Regan, and J. P. Nadal. Is There Something Out There? Inferring Space from Sensorimotor Dependencies. *Neural Computation*, 15(9):2029–2049, 2003.

[51] D. Philipona and J.K. O'Regan. The sensorimotor approach in CoSy: The example of dimensionality reduction. *Cognitive Systems*, pages 95–130, 2010.

[52] J. Piaget, B. Inhelder, and B. Inhelder. *The psychology of the child.* Basic Books, 2000.

[53] Jan P Piek, Lisa Dawson, Leigh M Smith, and Natalie Gasson. The role of early fine and gross motor development on later motor and cognitive ability. *Human movement science*, 27(5):668–681, 2008.

[54] D. Pierce and B. Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92(1-2):169–227, 1997.

[55] R. P. N. Rao and D. H. Ballard. Learning saccadic eye movements using multi-scale spatial filters. *Advances in Neural Information Processing Systems*, 7:893–900, 1995.

[56] Silvio Savarese and Fei-Fei Li. 3d generic object categorization, localization and pose estimation. In *ICCV*, pages 1–8, 2007.

[57] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[58] JM Schott and MN Rossor. The grasp and other primitive reflexes. *Journal of Neurology, Neurosurgery & Psychiatry*, 74(5):558, 2003.

[59] K B Shimoga and A A Goldenberg. Soft materials for robotic fingers. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992.

[60] Satinder Singh, Michael R James, and Matthew R Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 512–519. AUAI Press, 2004.

[61] A. Slater. *Perceptual Development: Visual, Auditory and Speech Perception in Infancy.* Psychology Press, 1999.

[62] William D Smart. Explicit manifold representations for value-function approximation in reinforcement learning. In *AMAI*, 2004.

[63] Dana Spiegel. The development of geometric knowledge. http://alumni.media.mit.edu/ spiegel/papers/, 1998.

[64] Nathan Sprague. Basis iteration for reward based dimensionality reduction. In *IEEE 6th International Conference on Development and Learning*, pages 187–192, 2007.

[65] Gilbert W Stewart. On the early history of the singular value decomposition. *SIAM review*, 35(4):551–566, 1993.

[66] J. Stober, L. Fishgold, and B. Kuipers. Learning the sensorimotor structure of the foveated retina. In *Proceedings of the Ninth International Conference on Epigenetic Robotics*, 2009.

[67] Jeremy Stober. CMACs in Python. https://bitbucket.org/stober/cmac, 2008.

[68] Jeremy Stober. Visual mountain car domain. https://bitbucket.org/stober/vizmc/src, 2014.

[69] Jeremy Stober, Lewis Fishgold, and Benjamin Kuipers. Sensor map discovery for developing robots. Technical report, AAAI, 2009.

[70] Jeremy Stober and Benjamin Kuipers. From pixels to policies: A bootstrapping agent. In *7th IEEE International Conference on Development and Learning*, pages 103–108, 2008.

[71] Jeremy Stober, Risto Miikkulainen, and Benjamin Kuipers. Learning geometry from sensorimotor experience. In *2011 IEEE International Conference on Development and Learning*, volume 2, pages 1–6. IEEE, 2011.

[72] Daniel Stronger and Peter Stone. Towards autonomous sensor and actuator model induction on a mobile robot. *Connection Science*, 18(2):97–119, 2006. Special Issue on Developmental Robotics.

[73] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1):625–653, 1999.

[74] B.W. Tatler, R.J. Baddeley, and I.D. Gilchrist. Visual correlates of fixation selection: effects of scale and time. *Vision Research*, 45(5):643–659, 2005.

[75] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[76] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[77] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[78] Heinz Warner and Seymour Wapner. The Innsbruck Studies on Distored Visual Fields in Relation to an Organismic Theory of Perception. *Psychological Review*, 1955.

[79] K. Q. Weinberger and L. K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *The 21st National Conference on Artificial intelligence*, pages 1683–1686. AAAI Press, 2006.

[80] Kilian Q Weinberger and Lawrence K Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, volume 6, pages 1683–1686, 2006.

[81] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Artificial intelligence: Autonomous mental development by robots and animals. *Science*, 291(5504):599, 2001.