The Virtual Hippocampus: Spatial Common Sense for Synthetic Creatures

by

Damián Alberto Isla

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
August 2001

© Damián Alberto Isla, MMI. All rights reserved.
The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis and to
grant others the right to do so.

Author .................................................................

Department of Electrical Engineering and Computer Science
August 8, 2001

Certified by ............................................................

Bruce M. Blumberg
Associate Professor
Thesis Supervisor

Accepted by ............................................................

Arthur C. Smith
Chairman, Department Committee on Graduate Students
The Virtual Hippocampus: Spatial Common Sense for
Synthetic Creatures

by

Damián Alberto Isla

Submitted to the Department of Electrical Engineering and Computer Science
on August 8, 2001, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis investigates the problem of spatial perception and reasoning in synthetic
creatures. Taking the Hippocampus - a brain structure that is strongly believed to
constitute a "cognitive mapping" system in animals - as an inspiration, a system is
demonstrated that visually learns cognitive maps. This cognitive map can be used for
problems such as self-localization and landmark-learning (and in fact its formation is
strongly tied to these processes). The resulting map can also be used as a basis for
a model of object-persistence, whereby the creatures can maintain realistic ideas of
the location of mobile objects in the world. When all of these systems are integrated
into a single brain, the result is a creature that perceives and reasons about physical
space in a way that strongly resembles how animals themselves do.

Thesis Supervisor: Bruce M. Blumberg
Title: Associate Professor
Acknowledgments

Many, many thanks to my advisor Bruce Blumberg, whose advice and encouragement has been pivotal in defining, refining and executing this thesis. I leave the Media Lab with a philosophy that I owe largely to him: that it is the ethologists, psychologists and neuroscientists that know about intelligence. Over the years he has also plagued me with such infuriatingly down-to-earth questions as "Yeah, but how does that change its behavior?" For that, in retrospect, I thank him.

Thanks also to that fantastically intelligent, enthusiastic, too-cool-for-their-own-good bunch, the Synthetic Characters. Rob Burke and I collaborated a good deal on the design and implementation of C4, and the many discussions we have had over the past two years have been profoundly influential to me. The brilliant Marc Downie has also been invaluable source of knowledge and advice, and some day, I hope, I will understand his Music Creatures. And to the rest of the characters crew, past and present, many thanks: Matt Berlin, Scott Eaton, Jesse Grey, Mike Hlavac, Yuri Ivanov, Mike Johnson, Chris Kline, Spencer Lynn, Ben Resner, Ken Russel, Bill Tomlinson and Alex the Parrot. Hope to see you all again soon.

Thanks to the many MIT friends that have made life bearable over the years: Ari Benbasat, Ellie Kim, Yee Lam, Aaron Maldonado, Josh Lifton, Kevin O’Connor, Rob Pinder, Aditya Prabakar, Phil Sarin, Push Singh, Jean-Paul Strachman, Ted Weatherly, Stephanie Yang and of course, of course, of course, Maggie Oh. Since this work is not just the culmination of 6 years of MIT, but a culmination of (consider this a melodrama warning) a lifetime of exploration, I have to thank the great friends who have had such a large part in making me who I am. I’m talking of course, of the MOAR gang, Amar Benchikha, Phil Griffiths, Nick Guillou and Matthieu Verstraat.

Finally, I’d like to dedicate this thesis to my father, my mother and my brother, Gino, Becky and Nico. It’s been a long road since I left home and became a freshman at MIT, but you have never been out of my thoughts – whether in history, travel, photography or playwriting, you have always been the ultimate inspiration for my various explorations of the universe. And when I look in the mirror, I still see you.
Contents

1 Introduction: A Spatial Sense for my Virtual Dog 13
   1.1 Map of this Thesis ........................................... 15

I Designing Synthetic Creatures 17

2 Design Philosophy 21
   2.1 What is a Synthetic Creature? .......................... 21
   2.2 Ethology ....................................................... 22
   2.3 Embodiment .................................................. 24
   2.4 Expressivity .................................................. 25
   2.5 Dogs ........................................................... 26

3 The Framework 29
   3.1 Architecture Overview ................................. 29
   3.2 Sensation and Perception .............................. 31
      3.2.1 The Percept Tree ..................................... 34
   3.3 Working Memory ............................................ 36
      3.3.1 PerceptAttributes and Belief Matching ............ 37
      3.3.2 Prediction and Surprise ............................ 39
   3.4 Autonomic Variables ..................................... 39
   3.5 Attention and Action Selection ...................... 40
   3.6 Navigation and Motor Control ...................... 44
   3.7 Installations ............................................... 47
II Environment Learning

4 The Hippocampus

  4.0.1 The Properties of Place Cells ............................ 56
  4.0.2 The Function of Place Cells ............................ 57
  4.1 Simulating the Hippocampus ............................ 57
  4.2 The Virtual Hippocampus in C4 ............................ 59

5 A SOM-Based Virtual Hippocampus ......................... 63

  5.1 Environment Learning as Classification .................... 63
  5.2 Self-Organizing Maps .................................... 64
  5.3 An SOM-based Hippocampus Model .......................... 68
    5.3.1 Useful Visualizations ............................... 71
  5.4 Examples of Learned Environments ........................ 71
  5.5 Behavioral Implications .................................. 77
  5.6 Review of the SOM Model ................................ 82

6 PhysicalSOMs ............................................... 85

  6.1 The Mass-Spring Model .................................... 86
  6.2 PhysicalSOM Structure .................................... 87
  6.3 Environment Learning ..................................... 88
  6.4 Growing the PhysicalSOM ................................ 90
  6.5 Examples of Learned Environments ........................ 91
  6.6 Review of PhysicalSOMs .................................. 92

7 Self-Localization ........................................... 95

  7.1 A Self-Localization Algorithm ............................ 99
    7.1.1 Orientation Estimation ............................... 99
  7.2 Position Estimation with the LocalViewSpatialSystem .... 103
  7.3 Position Estimation with the CartesianSpatialSystem .... 107
  7.4 Object Localization ..................................... 107
  7.5 Example Results ......................................... 109
8 Reliable Landmarks 113
  8.1 Defining "Landmark" ................................. 114
  8.2 Stability .................................................. 115
    8.2.1 Position-Mobility ................................. 116
    8.2.2 Orientation Mobility ............................... 117
  8.3 Choosing Landmarks ................................. 119
  8.4 Example Results ...................................... 120

9 Summary of Environment Learning 123
  9.1 Summary of the observation / object-matching / self-localization / object-localization loop. ......................... 125

III Object Persistence 129

10 Object Persistence 133
  10.1 Models of Expectation ................................. 134
  10.2 Negative Knowledge ................................... 136
  10.3 Negative Knowledge with Spatial Extent .................. 136

11 Probability Diffusion 139
  11.1 Isotropic Diffusion ................................... 140
  11.2 Verification ........................................... 141
    11.2.1 Visibility Testing ................................ 142
  11.3 Object Matching Using Location Distributions ............. 145
  11.4 Summary of the Observation / Verification Loop .......... 147

12 Spatial Behavior 151
  12.1 Emergent Behavior ..................................... 151
  12.2 Emotional Behavior ................................... 157
    12.2.1 Surprise ........................................... 158
    12.2.2 Confusion ......................................... 159
12.2.3 Frustration ................................................. 160
12.2.4 Emotional Attention and Action Selection ............... 160
12.2.5 Emotional Animation ....................................... 161

13 Conclusions, Contributions and Future and Related Work 163

A Hyperplane/RBF Interpolation 171
# List of Figures

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1</td>
<td>C4 System Overview</td>
<td>30</td>
</tr>
<tr>
<td>3-2</td>
<td>The Canonical C4 Brain</td>
<td>31</td>
</tr>
<tr>
<td>3-3</td>
<td>Duncan takes a look around</td>
<td>33</td>
</tr>
<tr>
<td>3-4</td>
<td>A typical Percept Tree</td>
<td>34</td>
</tr>
<tr>
<td>3-5</td>
<td>Belief creation and matching</td>
<td>36</td>
</tr>
<tr>
<td>3-6</td>
<td>Duncan’s Verb Graph</td>
<td>46</td>
</tr>
<tr>
<td>4-1</td>
<td>The Spatial Brain</td>
<td>61</td>
</tr>
<tr>
<td>5-1</td>
<td>Example Panorama Vector</td>
<td>64</td>
</tr>
<tr>
<td>5-2</td>
<td>Topology- and input-space</td>
<td>66</td>
</tr>
<tr>
<td>5-3</td>
<td>Synthetic Local Views</td>
<td>70</td>
</tr>
<tr>
<td>5-4</td>
<td>SOM Visualizations</td>
<td>72</td>
</tr>
<tr>
<td>5-5</td>
<td>Neighborhood function</td>
<td>73</td>
</tr>
<tr>
<td>5-6</td>
<td>Examples of environments learned by a creature without depth perception.</td>
<td>75</td>
</tr>
<tr>
<td>5-7</td>
<td>Depth perception</td>
<td>77</td>
</tr>
<tr>
<td>5-8</td>
<td>Examples of environments learned by a creature with limited depth perception.</td>
<td>78</td>
</tr>
<tr>
<td>5-9</td>
<td>Navigation using desired local views</td>
<td>79</td>
</tr>
<tr>
<td>5-10</td>
<td>Foraging behavior in a dynamic environment</td>
<td>81</td>
</tr>
<tr>
<td>5-11</td>
<td>Place-aversion behavior</td>
<td>83</td>
</tr>
<tr>
<td>6-1</td>
<td>Environments learnt by the PhysicalSOM</td>
<td>93</td>
</tr>
</tbody>
</table>
7-1  The problem of Self-Localization ................................. 98
7-2  Deriving global bearing from orientation landmarks ............... 102
7-3  Bearing-independent comparison of local views ..................... 102
7-4  Landmark memory .................................................... 106
7-5  Calculating the Error vector in the CartesianSpatialSystem ........ 108
7-6  Position recognition ................................................. 110
7-7  Position tracking ..................................................... 111

8-1  Orientation stability ................................................. 118
8-2  Landmark reliability learning ...................................... 122

9-1  Summary of the Environment Learning Loop ........................ 127

10-1 The Sensory grounding function .................................... 136
10-2 Location-Expectation Violation .................................... 138
10-3 Expectation Violation response .................................... 138

11-1 Diffusion on a hexagonal map ..................................... 140
11-2 Distribution culling .................................................. 142
11-3 The framebuffer as source of both positive and negative knowledge . 144
11-4 Two walls problem .................................................... 145
11-5 Probability vectors .................................................... 147
11-6 Summary of the Observation / Verification Loop ................. 149

12-1 Salient object matching .............................................. 152
12-2 Emergent look-around ................................................. 154
12-3 Emergent search ....................................................... 156
12-4 Duncan emoting ....................................................... 162
Chapter 1

Introduction: A Spatial Sense for my Virtual Dog

Duncan is a virtual dog. Like a real dog, he has a steady, trusting gaze, and a little tail that wags when he’s happy. Like a real dog, Duncan thinks not with his brain, but with his stomach and will do anything for food. More relevantly, Duncan learns from experience how best to get food. Again, like real dogs.

Unlike real dogs, however, Duncan lives not in the real world but in a simulated graphical world. But even in this, he shares some of the problems that real dogs face. Since he has a spatial body (i.e. a body that occupies a metric space, virtual thought that space may be) Duncan has to be able to interact spatially with his world.

A tremendous amount has been written in the animal learning and behavior literature on the spatial abilities of animals, from rats’ abilities to remember complex routes through mazes to the ability of honey bees to return to sources of nectar. However, synthetic creatures remain largely spatially incapable.

There has of course been much work on path-planning systems that, given a preset network of locations and connections between them, are able to find optimal paths from a starting point to a desired goal state. These path-planning techniques culminate in the venerable A* algorithm.

There has also been much work in reactive navigation. These methods use simple local rules for making navigation or object avoidance decisions. Flocking (as in [49])
is an example of a technique in which there is no goal state and there is no world-map. Instead each member of a Boids flock makes independent navigation decisions based on the input provided by its simple synthetic perception system at each instant of time. In this case it is the desire to maintain an optimal proximity and a similar alignment with nearby flock-members that control the boid animations and that, emergently, produce flocking. There are also simple techniques for reactive obstacle avoidance, such as potential-field collision maps ( [31]). Such techniques, however, are limited, since they don't have the ability to look ahead at impending obstacles and therefore are of use only in open environments with convex obstacles.

Fundamentally, little work has been done on the psychology of space that has so preoccupied the animal behaviorists. This incorporates not only questions of learning - how does an animal map its physical environs - but also questions of how such creatures see their space, and how they localize themselves and the objects in it. How do they reason about their space, and what assumptions do they make about how things move?

In the Spring of 2000, the Synthetic Characters Group of the MIT Media Lab made Duncan, the virtual terrier. Duncan was the focus of the group's research on reinforcement learning. He was also the protagonist of Trial By Eire, an installation piece crafted for the opening of the Media Lab Europe in Dublin that had him herding sheep.

In Trial By Eire Duncan had to navigate a large arena that was, conveniently enough, an open environment with convex obstacles. And while, guided by the vocal commands issued by the shepherd/user, herding the sheep into their pen was a possible and occasionally enjoyable task, Duncan's spatial abilities were sadly limited. He had no ability to plan paths through his environment. He had no ability to anticipate the movements of others. He had no concept of "over there" or "behind the fence" or "between the sheep".

This thesis, like much of A-Life research, has two points to it. First, to elucidate some of the mechanics of spatial common sense in natural biological systems. The hope is that by attempting to simulate (or at the very least emulate) the spatial
behavior of animals, we might gain some insight into how animals themselves model their physical environment.

The second point is to draw from the study of animals appropriate lessons on how to design artificial systems with spatial common sense. I do not, for example, intend to argue that the hippocampus model developed in Part 2 of this thesis is an accurate recreation of how the actual hippocampus functions in animals. The hope, rather, is that animal spatial behavior - including the special insight that modern neuronal activity-recording techniques afford - can help us design better synthetic creatures.

And the hope ultimately, is to give Duncan a better sense of his world.

### 1.1 Map of this Thesis

Part 1 discusses synthetic creature design in general terms, starting with the philosophy which guides the Synthetic Characters Group (chapter 2) and ending with the cognitive framework that the group has developed to implement that philosophy (chapter 3).

Part 2 discusses the problem of Environment Learning. Chapter 4 discusses the neuroscience of environment learning, reviewing some of what is known of hippocampal place-learning and provides an argument for why synthetic creatures should also have hippocampus-like abilities. Chapter 5 introduces a simple Hippocampus model based on the Self-Organizing Map (SOM) technique for unsupervised learning. Chapter 6 introduces another physics-based SOM variant for hippocampal modeling. Chapter 7 discusses the problem of self-localization, and how the spatial models built over the previous chapters can be used to solve it. A related problem - that of choosing reliable landmarks - is covered in chapter 8. A summary of environment learning is presented in chapter 9.

Part 3 introduces the problem of object persistence, and shows how a simple probabilistic interpretation of location-confidence combined with the spatial maps learned in Part 2 can give a creature an improved sense of object location and apparent "spatial reasoning". Chapter 10 discusses object persistence in general terms as
location-expectation. Chapter 11 presents a model of object persistence as probability diffusion over the environment maps and 12 presents some of the behavioral results of this model.

Concluding remarks are provided in chapter 13.
Part I

Designing Synthetic Creatures
How does one go about designing an artificial brain?

The endeavor is a unique combination of engineering, cognitive science and educated guessing.

The overall approach taken in most of our work is a practical one: intelligent machines already exist and live among us. They are called animals. And they have a lot to teach us about how brains are structured and how they work.

They also remind us of one important fact: that evolution is the ultimate engineer. In no work of man do we see such a perfect coupling of action and environment, of problem and minimal solution. Animals are perfect survival machines.

Plus, they’re easier.
Chapter 2

Design Philosophy

2.1 What is a Synthetic Creature?

The research group with which I have done this work was conceived as the "Synthetic Characters" group. The title of this thesis, however, makes reference not to Synthetic Characters, whatever they may be, but to Synthetic Creatures. Why the distinction?

Let’s start with what they have in common.

Both inhabit graphical worlds. Both are controlled not through the applied forces of muscles and tendons on bones, but through the application of mathematical rules, be those rules an attempt to emulate physics or simply the replay of chunks of hand-crafted motion. Both in some sense have brains, and reflect an agent-based approach to behavior simulation. By this I mean that both characters and creatures must make their own decisions about what to do and when and how to do it. To the overall set and sequence of actions they perform we apply the distinguished term of "behavior".

Note that these similarities seem to focus quite a bit on implementation. Perhaps creatures and characters are made, more or less, out of the same basic building blocks (polygons and "if then else" expressions), and look, more or less, the same.

But what do we plan to do with them?

This is where the distinctions start to come in, for in professing to build creatures rather than characters, we profess to build something that might, some day, be useful outside of that graphical body in that graphical world. Perhaps we are developing
practical methods for perception, memory and, yes, spatial reasoning, that might someday be put into a physical robot, that might then be programmed to do *generally useful things* for the human race. Perhaps on the other hand, we are attempting to *elucidate the methods and mechanisms* of our own biological brains. Perhaps, as much of the work in the schizophrenic field of Artificial Life does, we are attempting to do both.

The work in this thesis deals specifically with the problem(s) of spatial learning and common sense. The work cannot possibly have any meaning, however, without a discussion of our philosophical starting-points (this chapter) and the route we have taken to implementing this philosophy (chapter 3).

Here are four of the major inspirations for our work.

### 2.2 Ethology

For some time, the field of experimental biology and its numerous subfields have been a major source of information and inspiration on the psychology of perception and learning. This might be attributable to two things: (a) scientists can do things to animals that they’re not allowed to do to humans and (b) animals are simpler versions of ourselves. They are raw behavior, untainted by social prejudice or self-consciousness and uncluttered by the logical or otherwise ”higher-level” cognitive or linguistic abilities that merely add noise to studies of basic intelligence in humans. The rat pulls the lever because it wants food, not because it thinks that is what is expected of it. Granted, it is more difficult to get information out of a rat, since we cannot ask it ”how did that feel?” or ”what did you think would happen?” Nevertheless experimental biologists today use hundreds of ingenious techniques for finding out what is going on in animals’ brains, from statistical examinations of their behavior over time to large scale neuron activation recording techniques (this latter being a major source of the place-cell data that informs much of this thesis).

The Synthetic Characters group, however, is particularly inspired by the subfield of Ethology, the study of animals’ behavior in their natural environment. Unlike many
of the other areas of experimental biology, Ethology emphasizes the tight coupling between an animal’s behavior and its environment. To an ethologist, to witness a rat running a maze is not to witness animal behavior at all - for that it must be observed in the natural environment for which it was designed.

These ethological biases have not only heavily influenced our approach to action-selection ([6]) but also led us to our approach of whole-world modeling. In order to have behavior, ethology says, you must have an environment to which that behavior is tuned. This means that we must have some reasonable level of complexity in our world, and it must contain other creatures to interact with.

Ethology is also useful for the degree to which it has traditionally been ignored by Artificial Intelligence. AI researchers have typically been more prone to mathematical or formal-logical formulations of intelligence and decision-making. Although Artificial Life pays ethology some lip service (more so to ecology) MIT Professor Bruce Blumberg’s *Silas* ([6] and [7]) was really the first instance of ethological models of behavior and perception being implemented in a virtual creature (and he later went on to found the Synthetic Characters group). Given its relative distance from AI (this is changing), there still remains a great deal of “low-hanging fruit” in the form of theoretical biological models of behavior that are ripe for the simulated picking.

Ethology is important for a final critical reason: that animals teach us how to cheat. While the mathematical and formal-logical methods used by AI researchers often do result in provably optimal policies and behavior, they don’t always necessarily do so in real-time, nor are they necessarily robust to noise. Animals, however, do run in real-time and are robust to noise. This is because they do not bother with optimal policies or principled solutions, but are built rather from hack upon hack, each hack exploiting some regularity of the world. The quest for the intrepid creature designer then becomes to find the right hacks, the hacks that exploit the right world-feature in the right way. Animals and their occasionally superstitious or quirky behavior, can help us find these hacks.
2.3 Embodiment

Embodiment is a term from the robotics literature and coined by Brooks (e.g. [10]). Embodiment is the act of having a physical body, and is interesting to us for what it entails about the brains that are connected to those bodies. Brains do not function in isolation, but rather are a component of a larger physical system and the division between brain and body is not as simple as is popularly thought. Is the spinal cord part of what makes a being intelligent (despite the fact that its functions are so simple)? Is the peripheral nervous system? Is a primary sensory neuron that is activated through physical compression of skin part of the brain or part of the body?

There are two notable points to embodiment. The first is that a brain does not and cannot function outside of a body, for it is the body and the physicality that provides the grounding for behavior (otherwise are we not simply pushing symbols around?) The second is that embodiment benefits from another feature of the physical world and that is complexity. It has been noted that a simple rule applied to a complex environment results in complex behavior. Likewise, a robot that is made to deal with the complexities of the real world not only faces and proves it can overcome that complexity but also benefits from it. In a sense the challenge of embodiment is to see how much can be achieved with simple rules that, again, exploit the right features of the physical world. It is notable that the principle of embodiment leads to much the same emphasis on the agent’s environment as ethology. Indeed, embodiment might be considered a sort of robotic ethology.

Now, our synthetic creatures do not have physical bodies, and the graphical environments they inhabit are by no stretch of the imagination as complex as any physical one. However, by giving them synthetic bodies, we impose on ourselves some of the same limitations faced by physical robots. In this case it is the graphical bodies and the graphical worlds that provide the grounding. It could be argued that these are still just symbols being shuffled around as always. They are, but at least they are being shuffled around in a constrained way. A way that, for example, prevents creatures from teleporting from one end of the world to another, or from peering into
each others’ brains in order to decide what to do. If we are attempting to emulate animal intelligence, we have to make sure that we solve the same kinds of problems.

One theme that will come back again and again is the theme of Sensory Honesty. In the physical world there are a number of unavoidable abstraction barriers, such as that a creature can only impact the world through physical motion, and information can only be gathered through appropriate sensing structures. They cannot see into each others’ brains. Perhaps Synthetic Characters can do this in good conscience, but as designers of Synthetic Creatures we cannot allow ourselves to be so lax. Sensory honesty is precisely that self-discipline that enforces realistic constraints (or as close to real as possible) on perception. The synthetic creatures we build will not, where possible, be able to perceive something that a real animal would not be able to perceive. Sometimes this means simply phrasing a question about the world in a different way. We cannot tell what another creature's object of attention is. But maybe the direction of its gaze tells us enough.

2.4 Expressivity

Dennet speaks of an intentional stance ([14]) as the collection of cues, verbal or nonverbal, that convey a creature’s intentions. He argues that it is these cues that allow us first, to predict a creature’s actions and second, through successful prediction, to attribute intentionality to those cues and actions. Ultimately, it is the conference of intentionality that allows us to see something else as "alive".

Another group of people that are very good at expressing intentionality, this time through motion, are animators. In The Illusion of Life ([57]) two Disney animators describe a slew of techniques for breathing physical life and personality into animated characters. Many of these techniques, such as "anticipation" motions that signal an imminent more extreme motion, are perfect implementations of the intentional stance.

As designers of synthetic creatures we are very interested in creating creatures that appear alive. It is therefore very important for a creature to be communicative of its internal state. That communication could take the form of a terminal-screen
output, or a log file, of course, but we want something more intuitive and accessible. We use output modalities such as parameterized large-scale motion (a character can walk happily or sadly) and expressive facial animation to convey internal state.

The desire to convey internal state of course assumes that there is some internal state to be meaningfully conveyed, and this has lead us into the realm of emotion modeling and simulation. Emotions are excellent as indicators of general system state (perhaps a reason why they were evolved in the first place) and are easily understand and accessible precisely for their generality. Perhaps it is not meaningful to know that in 6 out of the 7 recent instances in which a character made a prediction, that prediction turned out to be wrong. It is meaningful, however, to know that the creature is confused. Part of the challenge and fun of emotion modeling is the mapping of strictly mechanistic phenomena to emotions.

Expressivity, to bring it back to Dennet, is also the doorway to empathy. If a human viewer attributes an emotion like sadness to a creature, then he attributes a host of his own personal associations with the emotion, and ultimately, perhaps, gives the creature credit for more than is actually there.

This might even be a useful survival strategy for some animals.

2.5 Dogs

For the past two years our group has focused on building virtual dogs, for the simple reason that they seem to combine most of the previous points. Dogs are extremely intelligent, and we can benefit greatly from the huge amount of work exploring how they learn. Dogs are also extremely adept at conveying intentions and conveying the appearance of empathy and understanding. Humans empathize with things that appear to empathize with them, and clearly attribute to dogs a host of human emotions. The dogs, of course, do everything they can to make this anthropomorphization easier.

From the point of view of a creature designer, dogs are also an attractive option for the very practical reason that not too much is expected of them. Part of the problem
of simulating humans is that we have definite expectations of humans (social, mental, linguistic, etc.) and are disappointed, annoyed, disgusted or amused (but not in the right way) when they fail to meet those expectations. On the other hand, when it is only a *dog* that is defecating in the middle of the living room, or chewing up the furniture, what can we do? It’s only a dog!
Chapter 3

The Framework

C4, standing for "Characters 4", is the fourth major incarnation of the Synthetic Characters Group's creature simulation system. It is a software kit which handles every aspect of our research and installation work, including graphics, networking, animation and behavior. All of the work described in this thesis was implemented under the C4 system.

C4 was designed primarily as a research platform. As such, flexibility and extensibility were prime considerations. Aspects of this work, including synthetic vision and the spatial system, were added on top of the base architecture. Since the initial design of C4, modules for social learning, classical and operant conditioning and motor planning have been added.

This chapter describes the layout of a typical C4 brain, and describes some of the installations that have been implemented under it.

3.1 Architecture Overview

This "virtual brain" architecture has been described in [26] and [12], and figure 3-1 presents a high-level overview. The central component is the World, which acts as a central switching point for communication between input devices, creatures and other processes. It also manages graphics rendering. Input devices include standard interfaces such as keyboards, mice and joysticks, as well as more exotic home-brewed
interfaces (such as the "surfboard" of the Sheep—Dog installation).

Creatures communicate with each other through messages passed to and distributed by the world. These messages, called data records are used extensively throughout C4 and represent sensory information that the creatures may perceive and act upon. For example, data records may contain visual information, which each creature posts every timestep, or it might contain user-input information that the creatures then "perceive" (in Sheep—Dog users speak into a microphone and the shepherd, the avatar, "hears" them). Since sensory honesty is strictly enforced (section 2.3, it is important that this event-passing mechanism is the only way in which creatures may communicate. Even physical actions (including location and body pose) must be bundled into data records and posted if they are to be perceived by other creatures in the world.

The other way in which creatures communicate with the user is, of course, through motion. Consistent with the emphasis on physically expressive characters (section 2.4) physical motion is an important output modality, both as an event-signaling medium (so that we can observe events as they occur) and as a means for creatures to express to the user aspects of their internal state (a walk cycle can be performed both "happily" and "sadly", depending on whether the creature is feeling cheerful or not). The world module manages the graphics and rendering state and therefore holds the "ground truth" on physical world-state (including creatures’ body configurations).

Figure 3-2 shows the structure of a typical C4 brain. It is divided into a se-
ries of layered systems that are executed in roughly top-to-bottom order in a single timestep. These systems are themselves divided into four major sections: Sensation and Perception, Working Memory, Attention and Action Selection and Navigation and Motor Control. All these systems communicate with each other through an internal blackboard.

### 3.2 Sensation and Perception

Each timestep, the data records that the world collected on the previous timestep are made available to each creature registered with the world. These data records are first processed by a creature’s Sensory System. This system plays an important role in the overall architecture as the prime enforcer of sensory honesty. One of its primary duties is to filter out sensory information that should not be perceivable by the creature, for example visual events that occur behind the creature. Local frame transformation (visual information is always attributed a location in the eye-space of the sensing creature rather than in world space) or sound intensity attenuation with
distance from the sound source are other means of enforcing sensory honesty that are performed by the Sensory System.

**Synthetic Vision**

Because much of this thesis deals with object permanence and the problem of dealing with occlusion, it was considered important to give creatures an accurate visual sense. Synthetic vision has been used extensively in A-Life research, including by the Synthetic Characters group in previous projects (e.g. [8]). In C4 we employed a simple form of synthetic vision through color-coding (see figure 3-3). This type of synthetic vision has been used before, for example in [60] and [40]. Each timestep, the visual creature takes as input a graphics rendering from the position and orientation of its eye. This rendering typically is a flat-shaded color-coded view of the world, in which each object and creature is assigned a unique color that acts as an identifying tag. Works like [60] go a step further than this work, by performing shape recognition on the silhouettes extracted from the rendering. Rather then performing full shape recognition, the synthetic vision system of C4 acts as a filter to the symbolic visual information that is received from the world in the form of VisualDataRecords. If the unique color-code corresponding to a given object is observed on a certain timestep, then the VisualDataRecord that was generated by that object (which includes symbolic shape and pose information that is not gathered visually) is passed through the Sensory System.

The other important function of the VisualSensorySystem is to perform location and bounding-box extraction. VisualDataRecords are always ascribed a single vector location, and when VisualDataRecords are posted by creatures to the world, they commonly post as their representative location the world-position of their root node. However, from the point of view of another creature, the root node is an arbitrary part of the observed creature’s body, and is certainly not honestly perceivable. Instead a more appropriate location can be extracted visually by examining the screen-space coordinates of the centroid of the object in the point-of-view rendering. This 2-vector combined with depth information from the rendering’s depth buffer yield the
NDC-coordinates of the object (see [18] for a discussion of NDC-space and camera projections). These coordinates can be converted into the local space of the camera (and of the observing creature’s eye) through the inverse-NDC transformation. Assuming that the x and y NDC-coordinates range from -1 to 1, and the z NDC-coordinate ranges from 0 (at the eye-position) to 1 (infinitely far), and assuming that the camera projection properties are given by a frustum defined by $f_{\text{near}}$, $f_{\text{far}}$, $f_{\text{left}}$, $f_{\text{right}}$, $f_{\text{top}}$ and $f_{\text{bottom}}$, the NDC-to-Local transformation is given by

$$z_{\text{local}} = \frac{(-f_{\text{near}}f_{\text{far}})}{(f_{\text{far}} - f_{\text{near}})z_{\text{ndc}} + f_{\text{far}}}$$

$$x_{\text{local}} = \frac{z_{\text{local}}}{2f_{\text{near}}} (f_{\text{right}}x_{\text{ndc}} - f_{\text{left}}x_{\text{ndc}} - f_{\text{right}} - f_{\text{left}})$$

$$y_{\text{local}} = \frac{z_{\text{local}}}{2f_{\text{near}}} (f_{\text{top}}y_{\text{ndc}} - f_{\text{bottom}}y_{\text{ndc}} - f_{\text{top}} - f_{\text{bottom}})$$

In addition to this location information, it will be important occasionally to collect bounding box information as well. Bounding boxes are stored as screen-space left, right, top and bottom offsets from the centroid. Average left, right, top and bottom depths are also stored (the average depth of the pixels at the extremity in question). All this information (including the symbolic shape and pose information passed in by the world) is bundled into a VisualDataRecord subclass called a BoundingBoxVisualDataRecord.
3.2.1 The Percept Tree

Once data records have been passed through the Sensory System, it is the task of the Perception System to interpret them. The distinction between sensation and perception is intended to mirror their neuroscientific definitions. Sensation is a physical phenomenon, such as the physical stimulation of a primary sensory neuron. Perception is the psychological reaction to that sensation - whether any attention is paid to it, whether it is considered painful, etc. Also like the neurosciences, perceptions can be internally generated, such as in the case of a creature’s observation of its own internal state.

The Perception System takes the form of a Percept Tree, a form of the hierarchy-of-sensors approach taken in much A-Life research (e.g. [28] and [9]). Elements of the tree are percepts, atomic perception units whose task it is to recognize and extract data from appropriate sensory stimuli. See figure 3-4.

Given a sensory stimulus in the form of a data record, a percept returns two things. First, it returns a single floating-point value representing the percept’s match confidence. This value mirrors the degree to which the data record seems to model the characteristic that the percept represents. For example, our creature may have a VisualPercept that is triggered by any visual event or information. In this case, this percept would return a confidence of 1 for a VisualDataRecord, but a confidence of 0 for an AuditoryDataRecord. Lower in the tree, ShapeRecognizerPercepts interpret
symbolic shape information, such that a SheepShapeRecognizerPercept will return
1 on a VisualDataRecord originating from a sheep creature, and 0 from a Visual-
DataRecord originating from a wolf creature. More interesting than these symbolic
percepts are probabilistic ones. Percepts can encapsulate any kind of recognition
mechanism, including standard trained machine-learning techniques, such as neural
nets or pattern-matchers. Such techniques often render not a definitive yes-or-no
match answer, but give rather the probability of a match. The uncertainty inher-
ent in these mechanisms is captured by the extracted confidence which can take any
double-precision confidence between 0 and 1.

The second thing that percept will return is an arbitrary piece of data that cor-
responds to the information that the percept is interested in. A BodyLocationPercept,
for example, will return a 3-vector representing an object’s location in the body-space
of the observing creature (as extracted directly from a VisualDataRecord). Likewise,
a ShapePercept extracts as its data a string representing the symbolic shape being
observed. Note that this is independent of any specific ShapeRecognizerPercepts
that might also be sensitive to that shape. Later, any processes that need to access
or act upon this information will query it by percept (for example, "what was the
BodyLocation of the red ball at time t?").

The hierarchical structure of the Percept Tree is a convenient and efficient orga-
nization (both conceptually and computationally) for percepts. It is not, for example,
useful to attempt to classify the shape of a data-record if it has already been de-
termined that the data-record is not visual. It therefore makes sense to place the
ShapePercept under the VisualPercept as a child. Two major assumptions are made:

1. A data-record is not relevant to a percept if it is not relevant to the percept’s
   parent (i.e. the parent has returned a confidence of 0 for the data-record).

2. A child percept only processes the extracted data of its parent.

Thus a ShapeRecognizerPercept can be a child of a ShapePercept, since it is not
relevant to any data-records that its parent is not relevant to and since it characterizes
data that its parent extracts.
Assumption 2 also allows the creation of generic classification percepts. Perhaps both object height and object brightness (represented by the HeightPercept and the BrightnessPercept) are represented by single floating-point values. Since the both pieces of extracted data have the same format, a series of 1-DClusteringPercepts (representing different portions of a 1-dimensional axis) could be added as children to both.

Once a datarecord has been passed through the percept tree as far as it will go, the combined confidences and data of all the relevant percepts constitute all that can be known about the event represented by the datarecord. All this perceptual information is cached in a single Belief object (see figure 3-5 for an example of belief creation). A belief is simply a mapping from percept to percept information (confidence and data). It is also a convenient way of bundling perceptual information that all comes from the same event. In this sense it is an engineering solution to the infamous binding problem in psychology (see for example [59]).

Beliefs are also the basic unit of Working Memory.

### 3.3 Working Memory

Working Memory is a black-board like repository for persistent Beliefs. Like the psychological formulation of working memory, the C4 Working Memory module holds a short-term perceptual history of the world. The beliefs that it holds map percepts
to confidence- and data-*histories*. It is on the basis of these histories that action-, attention- and navigation-decisions will be made. Put another way, the contents of Working Memory constitute the creature’s ”view” of the world.

### 3.3.1 PerceptAttributes and Belief Matching

Perceptual histories are stored in *PerceptAttribute* objects. In their simplest form, PerceptAttributes are simply circular buffers which hold the last $n$ seconds of perception. PerceptAttributes typically contain one buffer of confidence values and another buffer of data. They must therefore be able to answer queries like ”what was the confidence corresponding to percept $A$ at time $t$?” or ”what data was extracted corresponding to percept $A$ at time $t$?” As a convenience to some of the learning algorithms that have been implemented under C4, PerceptAttributes are also able to answer queries such as ”what was the highest confidence value within this range?” and ”what was the data at the highest confidence value within this range?”

The beliefs that are produced by the perception system have no history, because they capture only perception from the last timestep. However, in most cases, the object or event it represents has been perceived before. In the case of an object the creature is tracking, for example, we have in fact a steady stream of beliefs coming from the object, as each timestep it perceives it again. It is likely, therefore, that when a belief is produced by the Perception System, a belief corresponding to that object already exists within working memory. We can thus try to *match* the incoming belief.

To match a belief, a distance metric needs to be defined so that we can compare beliefs. One way to do this is to treat beliefs as a vector of values, with each percept-entry corresponding to a dimension. Distances between individual percept-entries can therefore be found and the magnitude of the overall difference vector computed. Furthermore we can use the product of the confidences as a weighting to each dimension, such that if we are less confident in a particular percept datum, then it influences less the overall belief matching process.

How the individual percept-entries are to be compared actually depends on the
PerceptAttributes, since it is the PerceptAttributes object that ultimately knows the format of data it contains. For some formats, there is an obvious distance metric. For vectors, for example, a euclidean distance can be computed. In the case of symbolic data (such as shape), a 0 is returned for perfect matches and an infinity for non-matches.

The distances between the incoming belief and all existing beliefs are computed, and the lowest is examined. If it is below a "matching threshold" then the incoming belief is considered to be a new instance of the old belief. If it is above that threshold, the incoming belief is considered to represent the appearance of a new stimulus, and so is added to Working Memory (and its history buffers are instantiated). In practice, shape and location tend to be the critical factors in determining a match.

Note that only percepts that are in both percepts are compared when finding the distance between two beliefs. This allows for multiple beliefs representing the same event or object to be combined in a single timestep. For example, the creature might receive both a VisualDataRecord from another creature and an AuditoryDataRecord if that other creature made noise. It is not possible, a priori, to link the visual and the auditory experience. However, through the matching process, it will be noted that the percept the two beliefs hold in common - the LocationPercept - have data values very close to one another. Therefore the two beliefs will be matched to the same existing belief in working memory, and the auditory experience will be considered as part of the visual one. This in turn allows Working Memory to answer to more complex queries, such as "what color was the sheep that baad at time t?" This another way in which Working Memory addresses the binding problem.

Not all beliefs will be matched every timestep. Some object will not observed on some timesteps. In this case the objects are left untouched. Where appropriate, the confidence in the data the belief contains is decayed. The rate of that decay should be roughly proportional to the expected variation in the data's value. Shape, for example, would not be expected to change from one timestep to the next, and so the confidence in an object's shape should not decay at all. However, physical location, should decay proportionally to the amount that the object has been observed to move.
3.3.2 Prediction and Surprise

The view of the world provided by Working Memory can be informed by much more than just direct perception. Just as our own perception is made up of a combination of perception and predictions, Working Memory incorporates predictions, assumptions and guesses made by a variety of systems about the state of the world (including the Spatial System that will be introduced later on). There are very practical reasons for giving creatures a basic ability to predict. Often the stream of sensory data coming from an object will be interrupted - because the object is out of the creature's field of view, or because it is occluded by another object. In order to maintain an accurate picture of the state of the object - and to continue to answer queries about the object when it is not observable - some prediction is necessary.

Even assuming that the state of the unobserved object has not changed is, in a sense, a prediction. When the piece of state in question is the object's shape or color, it is a fairly safe assumption. When the value in question is a dynamic one, other techniques are more accurate. For example, if the data is in the form of a scalar or vector, then standard function approximation techniques can be used to extrapolate its value. In other cases proprioception can be used to inform predictions, for example, predictions of egocentric location of objects based on self-motion.

Predictions - their occasional deviation from the actual state of the world and the magnitude of that deviation - also provide a basis for surprise. Surprise is an excellent method for focusing perception (a Belief which has just provided a surprising stimulus is an excellent candidate for the creature's object of attention).

Part III of this thesis discusses in depth a system for predicting location based on the structure of the environment.

3.4 Autonomic Variables

It is often useful to ascribe to each creature a set of emotional or motivational variables, indicating the relative influences of various competing drives in the decision-making process. Autonomic variables can represent general abstract system state -
"happiness" or "anger" - as well as motivational qualities such as hunger or thirst, which have definite "desired states" and definite strategies for attaining them (e.g. "eating" reduces "hunger").

These variables are called "autonomic" because they have automatic behavior. The general update rule for the autonomic variables we use is

\[ v^t = av^{t-1} + b + \gamma \]  

(3.2)

where \( v^t \) is the value of the variable at time \( t \). \( a \) is a multiplicative growth term and \( b \) is additive. \( \gamma \) is a feedback term from the rest of the system, and is the way in which behavioral or perceptual feedback can modify the values of the variables. Usually also associated with the variable is a "desired state", such that deviation from the state triggers relevant desired-state-restoring behaviors. "Hunger" for example, would have a desired state of 0 and a non-zero additive term, such that hunger, when left unattended, tends to rise. However, high hunger levels trigger eating behaviors which in turn reduce hunger (a high negative \( \gamma \) value) and thus restore balance.

While the Autonomic Variable System is controlled through extremely simple dynamics, it is a convenient and very high-level way to control overall behavior. For example, the creature can be made to search out and eat food by increasing its hunger drive, without needing to specify exactly which behavior will run in order to achieve this goal. It is also a convenient way to control overall "personality". This might include physical personality, since emotion variables can be used as direct arguments to the motor system for controlling emotionally-parameterized animation (see section 3.6).

### 3.5 Attention and Action Selection

Once the perceived/estimated state of the world is stored in Working Memory, the creature can make decisions about what to do. What exact decision-making mechanism is to be used has not yet been specified, and indeed a host of decision-making
techniques are possible here, including neural-nets, rule-based systems, chaining systems, etc.

The decision structure developed by the Synthetic Characters Group is the ActionTuple. The ActionTuple is 4-Tuple consisting of a set of triggers (the Trigger Context), a set of target-descriptors (the Object Context), a set of actions and a set of ending conditions (the DoUntil Context). The Trigger, Object and DoUntil contexts are all references to percept-values. For example, to describe an action like "pick up the red ball when you hear the whistle", an action tuple could be constructed as

- **Trigger**: the whistle-sound percept
- **Object**: the ball-shape percept
- **Action**: Pickup
- **DoUntil**: Pickup motion is finished

In other words, when a belief in working memory has a high confidence for the whistle-sound percept, this ActionTuple is activated. The creature performs the "Pickup" action on any belief with a high confidence for the "ball-shape" percept, and it performs it until the "Pickup" motion is complete (as gauged through feedback from the motor system).

ActionTuples are placed within ActionGroups, whose task it is to arbitrate between competing ActionTuples, especially when more than one can run at a particular time (i.e. their Trigger Contexts are satisfied). The ActionGroup must be able to prevent dithering (the quick switching back and forth between two or more actions) and enforce coherence of action (actions, once decided upon tend to run to completion) but must also allow for flexibility (if the state of the world changes significantly, or if something important happens, the creature must be able to change its mind about what action it should perform).

While essentially a context-triggered rule system, ActionTuples are an especially useful formalization of a rule-system because they present modular, learnable rules.
They are learnable in two ways. First of all, the relative value of each trigger-object-action-doUntil combination is learned through an on-line value back-propogation mechanism similar to reinforcement learning (e.g. [54]). Each ActionTuple is given a value indicating how successful it is in satisfying the creature’s drives (see section 3.4). By making the crucial causality assumption (that temporal contiguity implies causality), value is propagated backwards from consummatory behaviors known a priori to have high values (such as eating food, an inherent reward stimulus that always reduces hunger) to the behaviors whose executions seem to lead reliably to those consummatory behaviors being able to run. Value, however, is often propagated back not to the ActionTuple that actually ran but to the ActionTuple that, in retrospect, best fit the context at the time.

ActionTuples are also able to learn through innovation, a process by which the rules’ triggers can become progressively refined. Starting from the assumption that ”it is a good idea to sit down when a sound is heard”, through experience with the world (and noting that this rule seems to get rewarded sometimes), it might be decided that a more specific trigger than the ”Sound Percept” is needed. The ActionTuple can spawn copies of itself whose triggers correspond to children of the ”Sound Percept”. It might subsequently be discovered that ”it is a good idea to sit when a human utterance is heard” is a better obtainer of reward. As the trigger becomes even more refined, the creature may eventually realize that the true rule to learn is that ”it is a good idea to sit when a human utterance sounding like ’sit’ is heard”. This is equivalent to an ActionTuple whose trigger is the ”SitUtterance” percept, a child, we might presume, of the ”Human Utterance” percept.

When an ActionTuple is selected for activation, the actions in its action-list are performed. These actions, rather than directly performing animations or physical output, more often produce control signals that influence other systems, notably the Navigation and Motor Systems. These signals usually take the form of tokens that are placed in pre-established slots of the creature’s internal blackboard. For example, both the Navigation and Motor Systems take control input from the Motor-Desired slot of the blackboard, while the Navigation System also takes input
from the Navigation-Approach and Navigation-Orient slots. The action of a rule that instructs the creature to "sit when you hear a human utterance that sounds like 'sit'" will write a SIT string token into the Motor-Desired slot of the blackboard.

The other result of an ActionTuple's activation is the automatic writing of the ActionTuple's target into the Object of Attention slot of the blackboard. When an ActionTuple is activated, the first thing it does is perform a Belief Selection selecting the Belief from Working Memory that best fits its Object Context (i.e. the best target of the intended action). By posting the chosen belief as Object of Attention, the ActionTuple makes it available to other systems that might need to operate on it. The Navigation System will perform approach and flee behaviors on the Object of Attention, and the Motor System will perform the requested physical action on the Object of Attention if that action demands a target. The Object of Attention also directs the creature's gaze, which is both a way of giving the creature more information about the object and a way of providing feedback to an observer about what the creature is attending to.

Clearly, not all actions require a target - sitting, for example, can be performed without reference to any intended target. For this reason the Object Context is an optional component of the ActionTuple. However, since we wish to maintain the illusion of life (and since staying alert to the state of things even while performing an untargeted action is helpful) we found it useful to provide a list of characteristics that are generally interesting as objects of attention. This list takes the form of a separate ActionGroup, the AttentionGroup that runs prior to the Primary ActionGroup. The AttentionGroup also contains ActionTuples. However, these ActionTuples contain no actions. They do contain triggers (which specifically generally-interesting traits), Objects Contexts (which refer back to the generally interesting things) and DoUntil Contexts (typically short timer-based contexts which have the ActionTuples execute for 1-2 seconds before shutting down). "Generally interesting things", include objects that are moving quickly, objects that are large, and objects that have not been observed in some time. All these concepts are expressible as percepts.

Since the motor and navigation actions being performed are not targeted, the
only impact the AttentionGroup has typically is to direct the creature’s gaze in the
direction of the chosen object (though this results indirectly in a correspondingly
greater amount of information about this object and that region of visible space).
And since the AttentionGroup runs before the Primary ActionGroup, the Primary
ActionGroup often overrides any decision about the object of attention that it makes.

3.6 Navigation and Motor Control

Actual animation output of the creature is mediated by the Navigation and Motor
Systems. Both systems take as input tokens posted to appropriate blackboard slots
by the Action Selection System.

The task of the Navigation System is to control gross movement around the
world. It responds both directly to explicit navigation requests (such as requests
for approaches toward the object of interest) and indirectly to implied requests. For
example, the Action System may make a request like ”Beg within 10 units of the
shepherd”. In this case BEG is posted to the Motor-Desired slot, and 10 will be
posted to the Navigation-Approach slot. The Navigation System will note the ap-
proach request, realize that the BEG cannot yet be performed and so override the BEG
with a more immediately appropriate motor instruction such as WALK. Once it has
navigator the creature to the appropriate distance from the target (the shepherd),
the Navigation System stops overriding the BEG request. The request is then allowed
to pass through to the Motor System, where the physical action of begging will be
performed. The Navigation System will thus override any motor requests as long as
its own navigation-conditions are not satisfied.

The Navigation System is also responsible obstacle avoidance. The mechanism
by which this takes place varies. In the Sheep—Dog installation a global collision
map based on potential field methods was used (see for example [31]) to generate
suggestion vectors which Duncan the sheepdog would use to modify his bearing in
order to avoid obstacles. For the work described in this thesis, a variant of the
Navigation System, the VisualNavigation System was used to avoid obstacles visually.

44
By the time the Navigation System has finished running on a particular timestep it has either overridden the Action System’s motor request with a motor request of its own, or it has allowed the Action System’s request to pass through. It is then the turn of the Motor System to attempt to comply with the motor request made.

The Motor System

The Motor System is an animation control system. When one of the higher-level systems request a physical action to be performed, it is the job of the motor system first to get into a position where the action can be performed (for example, if one is sitting on the ground, one cannot run without first standing up) and second actually performing the action - i.e. modifying the joint-angles of the geometry hierarchy making up the creature’s body. In other words is the job of the Motor System to perform requested physical actions without violating the basic physical constraints of a simulated-physical body (constraints such as position and velocity continuity, etc).

The Motor System makes use of a *Verb Graph*, a convenient animation-control structure. The Verb Graph is essentially a finite state machine that links static poses (nodes) with animations (arrows). Figure 3-6 shows a typical motor system. When a motor request arrives, it is either for a static pose or an animation. In either case, the Motor System must path-plan through the graph in order to reach the desired pose or animation (of course playing all the animation on the way).

Recent Motor-System work has extended this basic idea of the Verb Graph by breaking animation source into its basic elements and then constructing a *Posegraph*. Posegraphs allow compatible animation-segments, though they may belong to different animations, to be combined. It also allows animation-transitions to be created automatically. See [16].

Verbs and Adverbs

It is said *The Illusion of Life* ([57]) that ”there is no such thing as a walk cycle”. Instead there are many walk cycles, for when the characters is sad, happy, proud, ashamed, etc. In other words, a character never moves without betraying something
about its internal state. In keeping with this ideal of expressive characters, we would like our motor system to be able to provide parameterized actions, not just verbs, but *adverbs* as well. *Walk*, but walk *happily*.

The C4 Motor System achieves this functionality through blending of animation examples. These examples are hand-crafted by an animator and then hand-annotated with a set of coordinates in emotional-parameter space. Then at run-time a new appropriate emotional animation can be synthesized by taking as input the current emotional state of the character and interpolating the joint-angle values. This is exactly the scheme used in [51].

The additional adverb input to the Motor System is provided through a *Motor-Adverb* slot in the internal blackboard. The adverb provided is either set by the Action System (ActionTuples can write both verb and adverb requests) or is provided automatically by emotional or motivational variables contained in the Autonomic Variable System.

The adverb does not always necessarily refer to an emotional variable. In the case of locomotion, the single-dimensional adverb is used as a bearing-controller (a value of -1 causes a tight left turn, a value of 1, a tight right turn, a value of 0, a straight-ahead walk). This bearing-adverb is usually set by the Navigation System.
Animation Layering

A character never does just once thing at a time. Animation Layering exploits the fact that not all motion is mutually exclusive. Indeed, a human character can walk at the same time as it shakes its head at the same time as it waves its hand. These would be considered different layers.

The Improv system ([44]) divides motion into different layers, whereby motions are only mutually exclusive within a single layer. Examples of layers might include "postural" (gross body action, such as walking or sitting), "gestural" (extremity motion, such as hand-waving or head-nodding) and "emotional" (eyes, eye-brows, tail where applicable, etc). Thus a character could walk at the same time as he waves, but couldn’t nod at the same time as he shakes his head.

Much like the Improv system, C4 provides support for multiple animation layers. Unlike Improv, rather than providing explicit locking mechanisms whereby high-priority actions override low-priority claims to degree-of-freedom resources, C4 again uses a form of weighted blending. High-priority animations have much higher weight than low-priority ones, and thus almost completely cancel them out.

Each layer can have an entire verb-graph system associated with it. In other cases, rather than going the example-based route, the animation is generated procedurally. Such is the case with the "Look-At layer" which controls head and eye movement (and keeps head and eyes locked on the object of interest).

3.7 Installations

Four major projects have been implemented to date with C4. Most of them have used the character of Duncan the Highland Sheep—Dog, the star of C4. The first three projects have shown Duncan becoming progressively more sophisticated, while the fourth has shown C4 applied to a pack of Duncan’s canine cousins.
Isle of Man’s Best Friend

*Isle of Man’s Best Friend* was Duncan’s first public appearance. Shown at the Game Developer’s Conference 2000, *Isle* was a short technology demo that allowed a user to take control of a highland shepherd by giving vocal commands to Duncan. At the time, Duncan’s behavioral repertoire was very limited, having only a few physical action he could perform, such as beg, sit, paw and solicit-to-play. However these actions were parameterized and the demo featured the ability to perform simple shaping on Duncan’s actions by rewarding progressively better examples of the action (if a high-beg was desired, then initially all begs were rewarded and then as begs become more frequent, only high begs).

Sheep—Dog: Trial By Eire

For *Sheep—Dog: Trial By Eire*, Duncan moved from the highlands of Scotland to Ireland, where he was employed in a demo-game to herd a flock of sheep from one end of an obstacle course to another. Modeled on the shepherding time-trials, the user again took on the role of a (newly remodeled) shepherd. The user communicated with Duncan via a voice interface using 6 standard shepherding commands (e.g. ”bye” to circle clockwise around the flock and ”away” to circle counter-clockwise around the flock). The one-shot voice-learning interface was one highlight of the new demo, with a robust example-based acoustic pattern matcher that was able to classify commands with 80- to 90-percent accuracy with just a few training examples.

For *Sheep—Dog* we moved away slightly from the learning aspect of the research, though with Duncan’s newly-introduced Navigation System, it became a demonstration of Duncan’s basic spatial competencies as well as the emergent interactions between multiple autonomous and semi-autonomous creatures. It also became something of a stress-test for the system - the demo featured one Duncan, one Shepherd, six sheep (all running a C4-implemented version of Boïds), three obstacles and one camera, all running one version or another of C4.

*Sheep—Dog* was exhibited at the opening of the Media Lab Europe in Dublin in

**Clicker By Eire**

For *Clicker By Eire* we scaled back to just the Shepherd and Duncan once again. *Clicker* was the first full implementation of D-Learning, the integrated learning system which incorporates several forms of learning at different levels of the system, from automatic percept-clustering to ActionTuple value back-propogation to Action-Tuple innovation. In it, Duncan could be "clicker-trained", a training technique used by dog-trainers. Through clicker-training, Duncan could learn to associate arbitrary utterances with arbitrary actions purely through reward feedback from the user. Through clicker-training, Duncan could learn up to 6 commands within 15-30 minutes.

**AlphaWolf**

At the time of this writing, AlphaWolf, an installation piece for Siggraph 2001 is being built with C4. In AlphaWolf, C4 controls the behavior for members of a small wolf pack. Through interactions forced by users, the wolves fight, growl, bark and play their way to the same sort of dominance-submission relations that can be seen in the real thing. The AlphaWolf brains are similar to Duncan-brains, though enhanced with more sophisticated social competency modules such as the emotional memories.
Part II

Environment Learning
How do we divide up physical space in our minds?

How do we go about locating ourselves and other objects in that space?

The answer: effortlessly.

Whatever the mechanisms, these processes occur for us completely unconsciously. We don’t have to think about it. We just do it. And we make it look easy.

Synthetic creatures have a long way to go in this regard. Their spatial perception is based more often on geometric convenience than psychological utility. If we want our synthetic creatures to act like real creatures, shouldn’t they perceive their physical environment in a similar way?

Primary inspiration to much of the research described in this section is a brain structure strongly suspected to constitute an environment-learning system in animals - the Hippocampus.
Chapter 4

The Hippocampus

In keeping with the ideal of biologically-inspired synthetic creature design, one thing we can do in our quest to design spatially-savvy synthetic creatures is to ask "what do real animals do?" There is, of course, an enormous amount written on the spatial/homing abilities of animals, from rats running mazes to bee-dances giving directions to new nectar-sources.

Though most spatial-research has had to do with large-scale navigation and spatial memory, a fascinating new source of information on the psychology of space was opened up with the advent of large-scale neuronal recording techniques and their application to the hippocampus.

The hippocampal formation is a relatively small set of neural structures that lies under the cerebral hemispheres. Part of the limbic system, the formation gets its colorful name (Latin for "sea-horse") from the two interlocking 'c'-shaped structures that constitute it: the dentate gyrus and the hippocampus proper. The hippocampus is immediately distinctive anatomically, due to the density and regularity of the neural connections. This is perhaps why the dentate gyrus was the first area where LTP (long-term potentiation, a form of long-term learning) was found experimentally ( [52]). Neuroscientists later found another interesting behavior while experimenting on animals: that certain cells within the CA1 and CA3 regions of the hippocampus are sensitive to the physical location of the animal. For example, rats running mazes were found to have cells that fired rapidly only in specific locations within the maze.
These "place cells" were soon found to be complemented by "head-direction cells", which are sensitive to global orientation. Orientation-sensitive cells are now known to reside in an array of brain structures, including the posterior parietal cortex, the retrosplenial cortex, the dorsal presubiculum and the anterior thalamus ([36]).

4.0.1 The Properties of Place Cells

[48] provides an excellent overview of place cell and place-field properties. Below I recapitulate some of the most salient experimental findings.

**Place fields are formed soon after initial entry into a new environment.** Studies into how long place fields take to form have found various different ranges of periods, most commonly 10-30 minutes, up to 4 hours.

**Place field are directional along practiced routes but not in open environments.** It is found that in environments such as hallways, linear tracks or along practiced routes in open environments, certain cells will fire only in response to travel in a specific direction. When wandering open environments, by contrast, cells are omnidirectional, responding to location regardless of orientation.

**Distal Landmarks are crucial to orientation.** It appears that distal landmarks are more influential in establishing the animal’s perception of its world-orientation than local landmarks. Experiments show that place fields follow reorientations of distal cues, but not reorientations of local ones.

**Place fields persist when landmarks are removed.** This shows that the animal’s self-localization ability does not depend entirely on visible landmark cues. In one experiment ([39]) it was found that removing a landmark from a cylindrical environment impaired the animal’s ability to self-orient, but did not effect the shape of the place fields. In other cases, such as [47] it was found that extinguishing all light once the animal had spent some time exploring an environment also had no effect on the place field configuration.
**Place field shift with experience.** Place fields along an often-traversed path tend to shift backwards with more experience.

**Place cells are sensitive to more than location.** Evidence abounds for the fact that place cells are sensitive to non-spatial cues, such as odor, texture underfoot and current task. They can also come to represent multiple locations within a single environment.

### 4.0.2 The Function of Place Cells

Though the CA1 and CA3 pyramidal cells of the hippocampus show clear correlations between firing rates and physical location, there is still some controversy as to what exactly these representations are used for. The original hypothesis forwarded in the seminal *The Hippocampus as a Cognitive Map* ([41]), that the hippocampal spatial learning system is used in environment navigation, remains a popular view. However, other functions have been suggested, for example, that the hippocampus acts as an emotional center ([32]), as a basis for Pavlovian inhibition ([15]) or as a basis for Working Memory ([42]). The hippocampus has also been implicated in the processes of context retrieval and episodic memory formation [48]. With such a wide range of hypothesized roles, the function of the hippocampus is clearly still a focus of much research.

### 4.1 Simulating the Hippocampus

From our point of view, the interesting thing about the Hippocampal place cell phenomenon is that it provides new low-level insight into the psychology of space. It tells us, for example, what elements are crucial for the definition and recognition of location (local visual landmark cues) and orientation (distal cues). It tells us what inputs seem to effect the state of the animal’s self-location representation (visual combined with vestibular combined with proprioceptive).

Perhaps most importantly, the place cell phenomenon suggests a *strategy* for spa-
tial perception. The hippocampus shows that the brain deals not with continuous space but with a discretization of space, with the discrete elements represented by place cells. The mapping from continuous-valued input and experience to discrete location and how that mapping is formed is 100% of the problem. Nevertheless discretization is a strategy with which computer scientists (like myself) are very comfortable. Clearly a good idea, whether implemented in neurons or bytes, is still a good idea.

Unfortunately there are certain aspects of place field formation we would like to know that current recording techniques cannot tell us. It would be interesting to know a little about the economy of place-cell allocation, whether there is tension between the desire to represent space at high resolution and the desire to keep the representation sparse and efficient. This is a major concern for this work (since we have to run everything in real-time on a single PC) and is difficult to address in hippocampus research because neuronal-assembly recording techniques are not (yet) comprehensive. For every neuron that is being recorded there are thousands that are not.

The hippocampus and the place-cell phenomenon has been extensively modeled. Redish ([48]) distinguishes between three different classes of models.

1. Local View Models: In these models the input to the hippocampus system is a continuous vector based on egocentric local view. This view is somehow mapped to a discrete neural state representing the place-cell response.

2. Path Integration Models: In addition to local view, path integration information is used to determine system response (indeed, path integration sometimes supersedes local view as location-determining stimuli).

3. Associative Memory Models: State transitions are stored, which map state (location) and action to a new expected state.

One model that bears discussion is that proposed by McNaughton et al. in [36]. This work presents the Hippocampus as a path integration system, and shows how
high-level interactions between a head-direction system and a position system, both represented as attractor networks, can theoretically account for most place cell dynamics. However, no mechanism for learning is provided and exact encoding of the local view is left vague. Nevertheless, it is notable that the system described in this thesis fits into the general structure that McNaughton describes.

4.2 The Virtual Hippocampus in C4

Why Do We Want a Hippocampus?

Chapter 3 describes a fairly sophisticated behavior-generation and learning mechanism. However, in the abstract, it is unclear how useful these mechanisms are. The primary goals of this part of the thesis are to formalize the psychological concepts of space and location and to expose them to the rest of the system in a form that it can use. A creature cannot learn to avoid or frequent a place - or to have any emotional associations with one - if a place is not represented as a first-class object within the brain framework. Dividing space into place fields (in a division that must be learnt) seems like a good way to do this. Furthermore, the cognitive map implemented in the virtual hippocampus allows for planned navigation and object persistence, and it provides the creature with a division of space that is more economical and more conceptually useful than the standard cartesian map that is used for most A-Life research. In addition to giving our creature more spatial abilities, we also get more truly animal-like behavior out of them. The creatures must learn their environment on their own, and can occasionally make the same kinds of mistakes that a real animal might (for example, becoming disoriented at conflicting landmark cues due to false assumptions about stable objects). All of these phenomena become possible when space becomes an object about which the creature can reason.

Integration into C4

Given that we would like our virtual creatures to have a hippocampus-like structure, how would we go about incorporating it into the framework described in section 3?
This section will provide a high-level description of the systems involved.

The Virtual Hippocampus itself will be implemented by a component called the Spatial System. This system will maintain the list of individual place cells, perform the necessary learning, etc. Individual Beliefs (as defined in section 3.3) which have locality will be able to interface with the Spatial System through Spatial Models, the memory structures that will hold their short-term spatial histories (a special case of the PerceptAttributes objects of section 3.3.1). The place-cells in the Spatial System can also play a role as Percepts in the Percept Tree. This use not only underlines locality as a property which object in the world can have, but it also exposes location to the high-level behavioral learning mechanisms of the Action System. Thus associations between place and action ("when in place A, do action B") and place and emotional state ("good things happen when I am in location A") are allowed to form through the exact same mechanisms that modulate the rest of the creature’s behavior. In this case, location is allowed to become just another variable. See figure 4-1 for an overview of the C4 Spatial Brain.

In addition to these explicit structures, it is also assumed that the creature has some basic spatial abilities such as the ability to transform points from eye-local space ("eye-space") to body-local space, usually aligned with the creature’s hips ("body-space"). This would presumably be done through a form of proprioception. It is also assumed that the creature will assume for itself a position and orientation in a hypothetical world-space, and that it can transform points from body-space to that world-space (although the mapping will of course change both with the creature’s motion and the creature’s learning the environment). This "internal world-space" has nothing, of course, to do with the world’s actual coordinate system, but rather is simply a useful abstraction for the creature.

In the next chapter we will begin with the specifics: how is the Virtual Hippocampus structured, what does it learn and what does it learn from?
Figure 4-1: The Spatial Brain

The major new system is the Spatial System. Working Memory stores location-data histories in the form of SpatialModels which also interface with the Spatial System. The place-cells it contains can also act as percepts in the Perception System. Finally, the cognitive maps built in the Spatial System can be used for navigation.
Chapter 5

A SOM-Based Virtual Hippocampus

5.1 Environment Learning as Classification

One approach to the problems of environment learning and self-localization is to treat it as a classification problem, in the traditional machine learning sense. This formulation takes in a vector representing current perception information and outputs a discrete location. Any number of algorithms and structures can be used here. Particularly appropriate are the unsupervised learning techniques, techniques in which categories are not specified beforehand but instead need to be discovered (or decided upon) by the algorithm. K-means and other clustering algorithms are good examples (see [5]). Each of these clusters then defines a location, and can be considered analogous to a place cell in the Hippocampus, or a percept in the Percept Tree.

The critical question that remains unanswered is how exactly to boil down ”current perception” to a single vector to be processed by a standard machine-learning algorithm.

Another name for ”current perception” is ”local view”, the term used in the ethology and experimental biology literature. This term itself can take many definitions. It might, for example, represent a bitmap-like snapshot of the current visual scene, or as a combination of sensor-readings and current behavior (as in [34]). Here we
A simple 2-landmark local view is concatenated into a 4-dimensional panorama vector. This vector will then be fed into the SOM learning algorithm.

will define local view as a set of *landmark vectors*, a set of reference-direction-aligned offsets to visible landmarks. At any point in space, this set of vectors is absolutely unique. They furthermore have the convenient property that nearby locations have similar landmark vector sets. This is the definition of "local view" that will be used in the remainder of this thesis.

How do we convert a set of 2-Vectors into a single input vector? We can simply concatenate them, such that the resulting vector takes the form shown in figure 5-1. This will be called the *panorama vector*.

Having established a useful format for our input data, it still remains to be decided which is the best learning algorithm to use to classify these panoramas. I argue that it is the Self-Organizing Map.

### 5.2 Self-Organizing Maps

Self-organizing maps (SOMs) of the type described by Teuvo Kohonen of the Helsinki University of Technology (see [29]) are clustering structures in which a discrete lattice of nodes is overlaid on the space of input vectors. Through training, the distribution of nodes comes to reflect the distribution of the inputs.
SOM Structure

A SOM takes the form of a regular lattice of nodes, usually two-dimensional and usually hexagonal (as is the case in this series of experiments). Each node is given two sets of coordinates. First, it is given fixed coordinates in topology space, the space defined by the topology of the SOM itself. Second, each node is given (will learn) a position in input space (i.e. the space from which the inputs will be drawn).

The primary task of the SOM will be to provide a mapping from one space to the other. The input-to-topology-space mapping is a form of categorization. The topology-to-input-space mapping is a form of sampling. See figure 5-2. The former process is the more common one, in which we will give the SOM an input vector and receive back a node which represents that input’s ”category”.

Note that both spaces are continuous and that both spaces are of arbitrary dimensionality. Because it is easier to visualize, the topology space is typically 1- or 2-dimensional, reflecting the inherent dimensionality of the lattice topology. Since it is precisely the dimensionality-reducing aspect of SOMs that make them useful for both categorization and visualization problems, SOM lattices of more than 2 or 3 dimensions are rarely used.

SOM Learning

Each node is given an initial random value in the input space. Inputs are presented every timestep. For each input, a best-matching unit (BMU) is selected from among the nodes and its input-space value and the values of the nodes within a certain neighborhood are moved closer to the input vector.

The update rule is defined as

\[ v_{i}^{t+1} = v_{i}^{t} + \alpha \cdot h(n_{i}, n_{BMU}, t) \cdot (v_{Input}^{t} - v_{i}^{t}) \]  \hspace{1cm} (5.1)

Where \( v_{i}^{t} \) is the vector value of the ith node at time \( t \)
\( n_{i} \) is the ith node
\( n_{BMU} \) is the best-matching unit for the current timestep
Figure 5-2: Topology- and input-space

The left-side figure shows a node labeled as $Node_{2,1}$ ($(2, 1)$ being its indices in the lattice) in its fixed position in topology space. The right-side figure shows the same node in a three-dimensional input-space.

- $\alpha$ is the learning rate
- $h(n_i, n_j, t)$ is a neighborhood function
- $v_{input}^t$ is the input vector at time $t$

Though intuitive, SOMs are clearly made up of many parts:

**Learning Rate** $\alpha$ The learning rate controls how far towards the input vector the BMU is moved. A value of 1 would make the two equal. In order to guarantee convergence, this learning rate is typically decayed over time to 0.

**Neighborhood Function** $h(n_i, n_j, t)$ This function defines a neighborhood around the BMU and determines how nodes within that neighborhood will be affected by a learning step. It can actually be considered a function of the topological distance between two nodes (i.e. the number of steps to get from one to the other in the mesh). Some applications use a step function (i.e. move all nodes within $n$ steps of the BMU an equal amount). Others move far neighbors less than close ones. Many decay the size of the neighborhood itself with time (hence the time parameter).

And implicitly:
**Distance Metric** Used in determining the BMU, the node whose value returns the lowest distance to the input vector is selected. Typically straight Euclidean distance is used, though other types of metrics can also be useful (an angular distance, for example, in which the node-value and the input vector are normalized and then an angle extracted).

Despite these many parameters, the SOM learning algorithm provides robust results for a wide range of parameter-values when sampling randomly from a static input distribution. For a full discussion of SOMs and their many variants, see [29].

**SOM Applications**

Fundamentally SOMs are a clustering technique closely related to vector quantization (see [24]). As such SOMs and their variants have been used in a large variety of machine learning tasks, including image analysis and compression, speech analysis and handwriting pattern-recognition.

The adjacency relations defined by the lattice and the highly non-affine mapping from input space to topology space have made SOMs useful data visualization tools, particularly for high-dimensional input data. Kohonen cites visualization as ”the main application of the SOM”. SOMs have been used in industry for visualizing system and engine conditions (e.g., [13]). SOMs can also be used to visualize similarities of highly symbolic information, including language and sentence analysis, in the form of contextual or semantic maps. Examples if these applications include the formation and recognition of sentences ([37]), semantic category formation ([50]) and full-text analysis ([25]).

SOMs can also be used as function approximators, by taking the topology-space coordinates and deriving through interpolation a likely input-space value. Kohonen himself cites the example of control of a robot-arm ([29]). In this work we will use the function approximation capabilities of SOMs for the task of self-localization (see chapter 7).
5.3 An SOM-based Hippocampus Model

Our approach will be to train a Kohonen-style Self-Organizing Map on the panorama vectors described above. Since the resulting nodes take on values in an input-space that define potential local views, the scheme is integrated into C4 as a LocalViewSpatialSystem, a variant of the Spatial System structure described in section 4.2.

SOMs are particularly appropriate for environment learning for a number of reasons:

- They give generally good results when the inherent dimensionality of the inputs is the same as the dimensionality of the map itself. In the formulation described here, input vectors are twice as long as the number of landmarks in the world (typically between 6 and 10) yet fundamentally describe a two-dimensional space. A topologically two-dimensional SOM therefore does well.

- They have adjacency relationships built in. This will be useful for route-planning.

- They are experience-sensitive, automatically allocating more nodes to regions of the input space that are frequently visited.

One of the original design motivations behind the Kohonen SOM was to capture the ability of some actual neural assemblies to organize themselves into topographic maps. Kohonen points to examples such as somatotopic and tonotopic cortical maps, where large neural assemblies are known to self-organize into coherent maps of sensory input. The somatotopic map of the S1 region of the cortex, for example, constitutes a model of the distribution of somatic sensors over the surface of the skin, such that nearby regions of skin are represented by correspondingly nearby regions of cortex, and regions of skin that show high density of sensors are represented by correspondingly large regions of cortex. These correspondences notwithstanding, the somatotopic map looks nothing like a miniature human-body outline in the brain. This is because the map is relational, not metric - much like the maps that SOMs create.
The mechanics of map formation in the brain are fairly well understood. Growing neural tracts are guided, through a variety of mechanisms, to the cortex, where they synapse upon cortical neurons that are initially highly unordered (much like a randomly-initialized SOM). It is through the presentation of and adaptation to sensory input signals that the cortical neurons gradually become coherently sensitized, through what is assumed to be an essentially Hebbian process. In a final parallel with the decaying learning rate of the SOM, many of these maps seem to have well-defined critical periods in which adaptation and map-formation can take place and beyond which no amount of sensory stimulation has any effect (the formation of ocular dominance columns in the V1/V2 region of the Visual Cortex is an example of this. On the other hand, somatopic maps seem to be adaptive for life).

While the SOM learning algorithm is quite a bit higher-level than Hebbian adaptation, there are elements of it that have similar effects. The neighborhood function, for example, is one way of approximating the local-excitation/global-inhibition effect observable in actual neurons (close neighbors are brought into a position of higher excitation, far neighbors are either repulsed or unaffected).

It should again be noted that the "cognitive map" of the Hippocampus does not quite belong in the same category as the cortical maps, because hippocampal place cells are not topographically arranged. Furthermore hippocampal maps are formed much more quickly (minutes) than cortical maps (in the experimental literature, days or weeks and many thousands of trials). Nevertheless, SOMs do seem to do a good job of capturing some of the essential dynamics of place-field formation observed in real brains.

Synthetic Local Views

One of the primary difficulties of Kohonen-style SOMs is their tendency to fold, shear and become otherwise mangled. As already discussed, a series of inputs drawn randomly from a static distribution will distribute the SOM nodes in a smooth and regular way. However, the highly temporally- and spatially-correlated inputs coming from a creature moving about a physical space will only rarely render a coherent
Figure 5-3: Synthetic Local Views

From a single observed local view (bottom, center) six synthetic local views are "imagined" and fed into the SOM learning algorithm. Note that the synthetic views tend to cluster around the landmarks, thus ensuring higher spatial resolution in those areas.

One way to overcome this problem is to generate a number of synthetic local views based on a single visual input. Knowing the geometry of how the landmark vector sets are formed allows us to generate the expected views at other locations in the world, essentially answering the question "what would these landmarks look like from over there?" This means that from a single local view, many valid SOM input vectors can be generated. While these input vectors contain no new information in the strict sense, they do smooth out the SOM. This might be considered analogous to running the same training set through a neural net several times. No new information is being introduced but it helps the dynamics of the learning algorithm.

This technique also allows us some basic control over the resolution of the map. As we might imagine that higher resolution in areas of interest, such as around landmarks, would be useful, we could concentrate our synthetic local views on those areas. This would cause the SOMs to allocate more nodes to that region of the input space, thereby giving us better spatial discrimination there.

The approach taken, therefore, is to generate a number (on the order of 10 to 20) of synthetic local views as experienced at a random point in the body-space of the creature. These random points are concentrated around landmarks. See figure 5-3.
5.3.1 Useful Visualizations

Visualizing the SOM can be tricky at times, given that they are structures which exist in high-dimensional spaces. Viewing the SOM in topology space is certainly possible but not necessarily very useful, since it is difficult to make its relation to the input space apparent. Here I will discuss two visualization methods that I have found useful for understanding and evaluating the results of the SOM learning process. See figure 5-4 for illustrations of both of these techniques.

In this section I use mainly a color-coding visualization method whereby each node in the SOM is given a unique color identifier according to its fixed position in topology-space. When the SOM has been sufficiently trained, a top-down view of the environment can be rendered in which each point is given the color of the best-matching unit at that point. This indicates the location the creature would consider itself to be at if it were placed in that physical location in the environment. Since the color-labels of the nodes in topology space vary smoothly from node to node (i.e. such that topologically close neighbors have similar colors) we would hope that this color-coded rendering of the physical environment would show a similarly smooth change in the color. A sure sign that the environment learning has been unsuccessful is if there are discontinuities in the coloring of the color-coded rendering.

Another simple visualization is to render the SOM in topology space with their input space values broken back down into a set of landmark vectors. This visualization underlines the SOM’s input-space value as a representative local view. While not as informative as the color-coding method, this visualization can again be used to verify continuity of the input-space values.

5.4 Examples of Learned Environments

It is fairly straightforward to implement a simple Hippocampus model using the canonical SOM algorithm. Inputs to the map, as already mentioned, are the panorama vectors described in section 5.1. For now we will assume that each creature has an absolute knowledge of a reference (north) direction, that the creature has a 360 degree
Figure 5-4: SOM Visualizations

(a) a sample 2-landmark environment configuration.  (b) the SOM topology space with node input-space values visualized as landmark vector sets.  (c) a color-coded rendering of the sample environment.  (d) the color key for (c), with each node colored with its color label.
field of view, that no occlusion is possible and that objects observed are all stationary and therefore make acceptable landmarks. All of these limitation will be addressed in later chapters.

**Creatures without depth perception**

Although it is a more strict limitation than actually occurs in nature, it is useful and interesting to explore the environment-learning capabilities of SOMs for creatures that have no depth perception, i.e. that can detect only direction to landmarks rather than direction and distance. We can achieve this by normalizing the landmarks vectors before concatenating them into the panorama vector on which the SOM will be trained. To avoid discontinuities in the input space we can choose to leave the vector unnormalized when its length is less than 1 unit, though comparable results are achieved whether this is done or not.

A specified number of landmarks were placed in the environment by drawing from a uniform 2-dimensional distribution, with $x$ and $y$ ranging from world-coordinates -100 to 100 units (distance and length measures will remain in generic "units" since the SOM algorithm is not scale-dependent). A single synthetic view was generated every timestep, by randomly choosing one of the landmarks and then sampling from a gaussian distribution centered on that landmark with a variance of 80 units. Timesteps were 0.01 time-units in length.

The SOM was a 15-by-15 node hexagonal map. The following learning parameters were used:
**Learning rate:** $1.5(0.9)^t$

**Neighborhood Function:** The piece-wise linear function shown in figure 5-5.

**Distance metric:** Euclidean distance $|p_{node} - p_{observed}|$

Figure 5-6 shows a progression of learned environments with 1, 2, 3 and 4 landmarks.

With one landmark (figure 5-6a), a radial pattern is formed - given only a knowledge of reference direction, the creature cannot in fact distinguish between locations along a thin slice of space radiating out from the landmark, because all those locations give exactly the same input (the same normalized direction vector toward the landmark). However, we see that when two landmarks are used the creature begins to triangulate its position, and so is able to distinguish regions of space between the two landmarks. This effect is increased with the addition of more landmarks.

From these initial results we can already draw a few simple but important conclusions.

**Landmark-dependent space**

As we might have expected, the shapes of the place fields vary greatly with the number of landmarks in the environment. A one-landmark environment can clearly only take the form shown in figure 5-6a, since with the exception of locations very close the landmark, all locations along a single stripe radiating out from the landmark look exactly the same. With the introduction of a second landmark, the creature begins to perceive a "depth" to its environment, i.e. to define finite regions of space. By the time we come to a four-landmark environment, a roughly square grid of place fields has formed, reminiscent of a Cartesian plane. The conclusion then, is that the creature's ability to triangulate its position and the resolution of that triangulation process increases with the number of landmarks. This is not, of course, a surprising result, but the results do show that this process falls naturally out of the SOM algorithm.
Figure 5-6: Examples of environments learned by a creature without depth perception. Four learned environments with 1 to 4 landmarks. Images in the left column show the simple simulated environment. Images in the right column show the same environment in which each point has been labeled with the color of the BMU at that point. The color is only a label here. The SOMs were trained for 5000 timesteps.
Subjective Space

The place fields that result are clearly not all of the same size or shape. However, they do represent regions in which the input vectors were all to some degree similar. This results, for example, in the sliver-shaped place field that forms between two landmarks in a two-landmark environment (5-6b). This sliver of space is clearly not a regular grid element. However, it does represent some sort of coherent experience for the creature - the experience, namely, of being between the two landmarks. This is a division of space that has no obvious mathematical or geometric definition, and yet that is subjectively very meaningful for the creature. Likewise, though some outlying regions of the environment might be very large, they might be represented by a single node in the final trained SOM because all the locations it contains represent more or less the same subjective experience. The important conclusion is that subjective space is not the same size or shape as physical space. In other words, it is not a metric space.

Creatures with limited depth perception

As already noted, removing all depth perception beyond a certain threshold is a far more restrictive than necessary. Though our sense of distance does indeed get worse the farther something is away, we can nevertheless tell when one distant object is further than another. In an attempt to capture this low-resolution at large distances, we can pre-process the landmark vectors such that the rate of change of perceived distance decreases the farther away something is. In other words, the mapping from actual distance to perceived distance should take the general form shown in figure 5-7.

One function which behaves in this way is

\[ v^* = \frac{\log(1 + |v|)}{|v|} v \]  

(5.2)

where \( v \) is the actual full-sized vector to the object and \( v^* \) is the processed vector that will be used in the panorama vector and will eventually be fed into the SOM.
Figure 5-7: Depth perception

Actual distance from a landmark should be mapped in the indicated way to perceptual distance. This ensures that differences in length and size matter less at great distances.

learning mechanism. In the equation, $v$ is normalized and then multiplied by the factor $\log(1 + |v|)$, a monotonically increasing function with a derivative that decreases asymptotically to zero. This reflects the fact that the difference between 2000 and 2500 units (where the units for a human scale might be meters) is perceptually less different than the difference between 5 and 10 meters (though perhaps not conceptually so).

When these pre-processed panoramas are used to train the SOM, the resulting maps are comparable but generally smoother and more uniform (see figure 5-8). The square shape of the SOM is clearly visible even when only 1 landmark is used in the world. There is also clearly better location perception between landmarks.

Other encodings for the landmarks vectors have been used, including the family of arctan functions

$$v^* = \frac{a}{|v|}\tan^{-1}\left(\frac{|v|}{b}\right)v$$  \hspace{1cm} (5.3)

This distance-correcting preprocessing of equation 5.2 is performed in the LocalViewSpatialSystem, and will be implied in the chapters that follow.

## 5.5 Behavioral Implications

The question, once we have this new system for dividing up space, is ”how does it impact the behavior of our creature?” The goal so far has been to come up with a psy-
Figure 5-8: Examples of environments learned by a creature with limited depth perception.

A logically useful definition for ”location”. Defining location in terms of ”expected local view” allows us to do exactly that. In this section, I will discuss three ways in which the creature can begin to benefit from these new spatial categories.

Navigation

With a lattice of nodes exhaustively covering the entire environment, and with adjacency relations built into the lattice, path-planning can clearly be done using any of the standard techniques (Branch-and-bound, A*, etc.). Given a desired end-state (end-location) a shortest path through the topology of the lattice can be found. This path can in turn be used to direct the creature’s navigation. This demands the ability to find local (body-space) desired-directions based on two landmark vector sets. Fortunately, this is quite simple:

\[ v_{\text{offset}} = \frac{1}{N} \sum_{i=0}^{N} (v_{i,\text{observed}} - v_{i,\text{desired}}) \]  \hspace{1cm} (5.4)

This calculates the average difference between corresponding observed and desired landmark vectors. See figure 5-9.
Figure 5-9: Navigation using desired local views
To navigate to a goal state (left) the differences between corresponding vectors in the current and desired local views are summed, giving a vector towards a location that will render the desired local view (right).

Foraging

One of the advantages of the SOM model is that it is fairly robust to environmental change, meaning that the place-field distributions that it renders remain meaningful even after minor changes in landmark distribution. We can think of a hypothetical foraging problem: a creature is shown a source of food in a certain location once it has learned the environment. It is then reintroduced to the environment some time later, only now the landmarks are slightly differently arranged (perhaps the distance between two of the is larger, for example). Where will the creature go to find the food source again? It will not, of course, go to the exact same metric world-location as last time (since it has no conception of metric world-location). Rather, it will return to the place field that previously held the food source’s location.

Figure 5-10 shows just such an experiment. A target was given to the creature after it had learned the environment and then the environment was reconfigured. The creature was then asked to find the target again. Renderings were also produced of the place field distributions after the rearrangement of the landmarks. Both the new distributions and the new target-location show that the placefields seem to retain their ”semantic” meanings after environmental change. In other words, a place field that represented the region of space ”between landmark a and landmark b” will generally still represent that same space after the change. Note from figure 5-10d, however, that after radical topological change in the geometric relationships of the landmarks,
the coherence of the map begins to break down.

Nevertheless, these results show that the SOM model will be robust to minor landmark-position variations due to perceptual noise, and that the creature will still be able to function after some amount of environmental change, even after the critical learning period is over. This might explain how the representations of environments frequented over very long timescales can remain coherent despite changes.

Of course this environmental robustness must be balanced with the process of finding and tracking only reliable landmarks - an object that moves around the environment should be considered mobile, and therefore should not be part of the localization process. See chapter 8. Biegler and Morris conducted a series of experiments exploring the relationship between landmark stability and foraging location in [3], and their results are consistent with the behavior displayed here.

**Place-Aversion**

Another way of illustrating the utility of the space-divisions that are created by the SOM algorithm is by demonstrating place-aversion behavior. We might assume that a creature living in a virtual world can, through experience, form emotional associations with specific locations. In the case of one set of simple experiments, a creature was "punished" for entering certain areas, meaning that an immediate negative association was formed. In subsequent navigation, the area in question was, if at all possible, avoided.

Place-aversive navigation is simple to implement. The creature was given a goal location to which it would navigate on command. It would do this by taking its current location (the current BMU) and finding the BMU-neighbor that was closest to the goal location. If this closest neighbor was marked with a negative emotional association, place-aversion kicked in, and the next nearest-neighbor was tried and so on. Whichever neighbor was eventually decided upon, it became the next navigation-target by finding the offset vector between the current local view and the input-space value of the node, as described above. In this simple model, the offset vector was used as a force which pushed the creature around the world.
Figure 5-10: Foraging behavior in a dynamic environment

A creature was introduced to and allowed to learn a simple three-landmark environment (a). After learning had completed, the landmark configuration was modified in a number of ways, as seen in (b), (c) and (d). A single target (represented by the black X) was tracked through all of these changes. The X therefore represents the same location in each diagram.
Figure 5-11 shows the results of some of these experiments. The important point behind these results is that though punishment occurred at single points, representative regions of space were avoided. These regions are psychologically meaningful. Figure 5-11d, for example, shows how the creature, after a single punishment, avoids passing between the two lower landmarks. Likewise figure 5-11e shows how after a single punishment the large area at the center of the three landmarks is avoided.

The emotional-association mechanism in this case is clearly trivial. However, this example shows how careful division of the environment can render locations that are useful and efficient in guiding navigation and learning. If the division of the space were a plain, unadapted hexagonal mesh, the concept of ”between the two landmarks” would be difficult to learn (perhaps requiring many learning trials) and inefficient to represent. With this system, however, such a concept falls naturally out of the SOM learning algorithm, and can be represented with only a few nodes.

5.6 Review of the SOM Model

The SOM has given us a good starting point, however they also suffer from a series of weaknesses that will lead us to consider other alternatives.

- Scalability. The learning algorithm has a run-time linear in both number of nodes and number of landmarks. Especially with the extra burden placed on the system by the synthetic viewpoints generated in order to guarantee map coherence, this run-time becomes unacceptable for real-time when many landmarks (i.e. 20) are used in the environment. While not many landmarks are strictly needed for self-localization (technically only 3 are needed) using a larger number of landmarks allows for better (and quicker) discrimination of moving objects (see chapter 8).

- Difficult Configuration. SOMs have many, many parameters, and all these parameters must work together in order to guarantee good results. Unfortunately there are no principled ways to choose many of these parameters, with manual
Figure 5-11: Place-aversion behavior

(a) A three-landmark environment. (b) The place-field distribution of the environment after learning has occurred. (c) A sample navigation to a goal with no averse locations. (d)-(h) Navigation in the presence of 1, 2 or 3 averse locations. In all the diagrams, the goal location is represented by the black X, and the motion is from top to bottom.
parameter-tweaking being the only alternative. To make matters worse, many of the SOM parameters have effects which are highly non-intuitive, and so modifying them in direction that would seem to improve it could, in retrospect, only make it worse.

- No provision for growth. The space-representation for these creatures must be capable of growth. As the creature discovers new areas to explore, for example, it must be able to allocate new nodes to represent it in order to leave undisturbed the representation of known areas. SOMs are topologically fixed (although some variants have experimented with topology-modification).

- Critical Period. The learning rate decays to zero (or near zero) after some time in the canonical SOM learning algorithm. While this could be biologically justifiable, a creature must clearly have some mechanism for representation-growth after the critical period has ended. Either a persistent synthetic creature needs its spatial representations to maintain enough plasticity after the critical period to accommodate environmental change, or some mechanism to spawn entirely new maps to cover new areas needs to be devised.

The last two points stem from a single fact: that Kohonen-style SOMs were not intended for on-line learning. In the next chapter I will present a variant of the SOM that addresses these problems head-on.
Chapter 6

PhysicalSOMs

One of the main difficulties of working with SOMs is that most of the desirable properties of the resulting map have to be modeled indirectly - smoothness is guaranteed by training the map with multiple synthetic views (whose number and distribution has to be chosen appropriately), the neighborhood function ensures continuity and the learning function sets how long the map should adapt for (and the correct choice for this length of time is anyone’s guess). Getting a map that’s useful for space-discrimination out of this jumble of parameters can be difficult.

The difficulty is also conceptual: the nodes in the SOM have values in an $n$-dimensional non-metric space. This space is difficult to work with because many spatial relationships that are true of the physical world (such as distance relations and collinearity relations, etc.) do not hold. These issues might well give us the sense SOMs make the problem more difficult than it needs to be. If it is a two-dimensional space we are modeling, why not model it in two dimensions? Or in other words, why not assume a two-dimensional structure to the model?

This is exactly what the CartesianSpatialSystem does. All the nodes in this implementation live in the same kind of space as the creature does - a two-dimensional Cartesian plane. Thus the space is metric and important spatial relations are kept and easily visualized.

The problem of node allocation still applies, of course: how do we compress and stretch the map in a way that provides high resolution where it is needed and low
resolution where it is not, and still guarantee a smooth distribution of nodes? One solution is provided by the computer graphics literature, specifically, from the area of physics simulation. The structure we wish to lay down on the input space should act like a stiff rubber sheet. Like a rubber sheet it should show some pliability (since we want to stretch it in order to represent the space well) but at the same time it should show some resistance to folding and twisting. One of the simplest physical models for a rubber sheet such as this is as a mesh of point-masses connected by springs. This technique is used extensively for such graphics tasks as cloth and skin simulation (e.g. [1]).

One of the most important features of the new structure is that it will not be bound by a pre-determined critical period, i.e. no decaying learning-rate will slow its adaptation over time (though it will, as before, converge upon a stable solution). Because of this, it is also going to be robust to sudden environmental changes, and to extensions of the environment. We are going to be able to grow the rubber sheet when necessary.

The inherent schizophrenia of this thesis rears its head again: we are not necessarily improving upon the performance of the SOM implementation of the Virtual Hippocampus. On the contrary, some of the desirable traits that it won us will not carry over. On the other hand, the system developed in this chapter is conceptually easier to work with and shares most of the desirable traits of the SOM implementation.

### 6.1 The Mass-Spring Model

Perhaps the simplest physics simulation technique is the mass-spring model. In this model, all objects are represented by a series of point masses connected by springs. Each of these elements obeys the basic rules of Newtonian physics. Each timestep, each spring, defined by a rest-length, a spring constant and its end-points (masses), are asked to apply its forces to each of its endpoints.
\[ f_1 = k \cdot (|p_1 - p_2| - l_{rest}) \cdot (p_1 - p_2) \]  
\[ f_2 = -k \cdot (|p_1 - p_2| - l_{rest}) \cdot (p_1 - p_2) \]  

Where

- \( f_i \) is the force applied to the first and second mass
- \( p_i \) is the position of the first and second mass
- \( k \) is the spring constant
- \( l_{rest} \) is the rest-length of the spring

Each mass is then allowed to update its state (both position and velocity) according to the following formulas:

\[ v^{t+1} = v^t + \Delta t \cdot \frac{f}{m} \]  
\[ p^{t+1} = p^t + \Delta t \cdot v^{t+1} \]  

Where

- \( v^t \) is the velocity of the mass at time \( t \)
- \( p^t \) is the position of the mass at time \( t \)
- \( \Delta t \) is the timestep
- \( f \) is the accumulated force (from equation 6.1)
- \( m \) is the mass

Formulas 6.1 and 6.2 in effect implement the forward-Euler integrator. For a more complete description physics-simulation techniques, including more sophisticated (and more stable) integration schemes, see [46].

### 6.2 PhysicalSOM Structure

The point-mass has a natural analog in the SOM node. For the physical SOM nodes become physical point-masses and node-adjacencies become springs.

The relationship between neighborhood function shape and map coherence is not an intuitive one, yet the neighborhood function must fall within a very definite range
of forms in order for a SOM to distribute itself correctly. If the neighborhood function
shrank the neighborhood over time, it had to be ensured that this shrink-rate was
carefully coordinated with the overall learning function (also a function in time).
These details and many other make SOMs difficult to work with.

On the other hand, the PhysicalSOM uses the natural dynamics of the spring-mesh
to guarantee map coherence. Again this is a matter of making explicit a property
that we had merely hoped for in the SOM. Counter-intuitive function-tweaking is
replaced by a direct modeling of desirable traits like evenness and smoothness.

Unfortunately simply laying out a spring per adjacency relationship is not enough
to guarantee robustness. A number support structure is required. This includes

- Springs between adjacent nodes
- Springs to second-degree neighbors
- Curvature springs (forces alignment of topologically collinear nodes).

It is often useful to speak of the scale-factor of the hexagonal lattice. This is the
distance between each node and its neighbors in topology space. It is also the initial
input-space distance between adjacent nodes before any adaptation has begun to take
place.

6.3 Environment Learning

By what method will our spring-mass lattice adapt to environmental configuration?
Since we have dispensed with the local-view as stored value of mass-nodes, it is instead
direct - though to the creature, hypothetical - world-location that is adapted to fit the
environment. Note that there is a valid trivial solution: merely lay down a regular,
unchanging hexagonal mesh of nodes and let them represent the space where they
fall. This is, however, a highly inefficient route, since on the one hand it is unclear
how large to make this mesh and where to put it, and on the other it models all space
with exactly the same resolution - including those regions that are uninteresting and
have never been visited.
Recalling our "Synthetic View" technique from the previous chapter, we again need to ensure that resolution is high in places of interest, such as around landmarks. The natural analog to SOM contraction is spring rest-length contraction. In order to control this contraction, we need to find a way to measure the "interest" of a location. Distance between the node and the observed world-location of the nearest landmark would be one such measure (with a lower value being considered more interesting), though linking this measure directly into spring length calculations leads to instabilities in the mesh - the meshes tend to either compress infinitely or inflate infinitely.

One the other hand, topological distance from the BMU of the landmark is a more stable representation, because it does not depend on world-location of the Physical-SOM nodes, and is simpler because it has no scale-dependencies (being measured simply in integer "steps").

How is this measure of "interest" or "desired resolution" mapped to connection rest length? In a parallel to the Neighborhood Function of the canonical SOM learning algorithm (section 5.2), we define a **Connection Length Function** which will do exactly that. It is most intuitively defined as a function that takes in an integer neighborhood and returns a factor that will multiply the topology-space length of a connection to render an input-space length - a useful formulation, since some of our connections connect 2nd-order neighbors (see 6.2). Note that this function returns a scale of interest for a node. The procedure will therefore be to compute this measure for each node in the map, and then have each connection combine the interests of its two endpoints. Taking the minimum of the two works particularly well. Thus the rest-length update rule is as follows:

$$C_{rest-length} = |Top(C_1) - Top(C_2)| \cdot \min (h(C_1), h(C_2)) \quad (6.3)$$

where $C_1$ and $C_2$ are the endpoint nodes of the connection, $Top(C_1)$ returns the topology-space coordinates of $C_1$ and $h(C_1)$ is the connection length function.
6.4 Growing the PhysicalSOM

When a node in a hexagonal lattice has less than 6 neighbors, it is considered incomplete. If the node represents interesting space, however, or if interesting unrepresented space lies beyond the node, then the node might be a good candidate for completion, whereby its missing neighbors are spawned and placed in a likely location in input space.

How do we decide when to complete an incomplete node? It would be convenient if the measure of interest from above could also be used for making decisions about how to grow the physicalSOM. Thus any incomplete node within a certain \( n \) steps of a landmark would be chosen for completion. This turns out to be a bad idea, however, since part of the power of maps like these is their ability to represent potentially huge areas with only a few nodes. If for example, the environment contained a far distal landmark, that landmark would indeed have a BMU on the edge of the map. This BMU would promptly be completed, and in the next step, the BMU would be the new closest edge node, and so on until every patch of space between the edge of the original map and the landmark was filled in. This is, of course, an enormous waste of resources.

A more effective method is to make a growth-determination in input-space, by simply saying that if the distance between the world-position of an incomplete node and a landmark is less than a threshold, then the node should be completed. A threshold that has worked well is 4 times the scale-factor of the lattice. This metric works well because it leaves appropriately unrepresented areas of the environment that are too distant to be interesting but fills in the space around nearby landmarks.

Though most growth takes place around areas of interest, it is also useful to have the SOM grow in areas that the creature explores. It can be ensured, for example, that the creature’s own position is always represented by a complete node.

Assuming we have decided to complete a node, how do we decide where to put its new neighbors? Given the structure of the hexagonal mesh, a newly spawned node has a clear topological position. However, an input-space position must also
be assigned. This position can be predicted using the topological neighborhood of the new node. In the implementation described here, a hyperplane-rbf predictor was constructed (see Appendix A). The example points were the topology-space positions of the neighboring nodes and the sample-values were their input-space positions. This predictor was then sampled at the topological position of the new node and the result became the position of the new node in input-space.

Node Culling

Depending on the order in which the environment is explored and the behavior of the objects in it, node culling can sometimes be necessary to ensure the coherence of the map. It may be, for example, that an object at the edge of the map is investigated in some detail (and so is represented with fairly high resolution in the map) before the object suddenly and unexpectedly moves. Now, we have an area of high resolution representing an essentially completely uninteresting space. Furthermore, because it is uninteresting, the connection lengths will grow, exploding this region and probably applying undesirable stress on the rest of the map. The correct thing to do in this situation is to cull incomplete nodes that are outside of a certain threshold distance from any object. This way the space-representation remains compact. Empirically, a good culling-threshold is twice the growth threshold. Whatever this length it must be longer than the longest possible spring length, in order to a spawning another because it is under the growth threshold, only to have the new node removed the next timestep because the new node itself is above the cull-threshold.

6.5 Examples of Learned Environments

Figure 6-1 shows the results of learning with a physicalSOM in a variety of environments. Since the physicalSOM nodes live in the same world-space as the creature and the landmarks, it is possible to visualize overlaid on the environment map. The color-coding visualization is also useful, particularly for comparing the resulting maps to those learnt by the canonical SOM.
The following parameters were used for all of the environments of figure 6-1:

**Spring constant:** 2.0

**Curvature spring constant:** 100

**Node mass:** 1.0

**Hex-mesh scale-factor:** 50

**Connection Length Function:** \( f(x) = \frac{x^2}{9} \)

**Completion threshold:** 200

**Simulation time:** 1000 timesteps

While the resulting maps look significantly different from those learnt by the canonical SOMs, it should be noted that they nevertheless share most of the desirable properties, most notably adaptive resolution in regions of high interest.

### 6.6 Review of PhysicalSOMs

PhysicalSOMs were conceived as a way to learn the same kind of psychologically useful environment maps as those learnt by the canonical SOM algorithm. PhysicalSOMs achieve this, and in a way that is far more intuitive than SOMs. They are easier to build and it is easier to ensure evenness and smoothness in the resulting map - this is achieved through direct modeling of those traits through parameters such as the connection length function. PhysicalSOMs have two very important additional properties: they can be grown (section 6.4) and they do not have critical periods beyond which adaptation cannot take place. This makes them far more useful for dynamic environments in which objects can switch landmark status, new areas can be discovered, etc.

The PhysicalSOM algorithm is also faster than the normal SOM. Because the input-space is always 2-dimensional (rather than the \( N \)-dimensional input-space of
Figure 6-1: Environments learnt by the PhysicalSOM
the SOM), the run-time is on the order of $\Theta(N + L)$ (where $N$ is the number of nodes and $L$ is the number of landmarks) whereas for the SOM it is $\Theta(NL)$. Of course, care has to be taken through careful selection of the PhysicalSOM parameters that an infinite number of nodes is not spawned.

How is it that PhysicalSOMs should do the same job, faster than canonical SOMs? As in any engineering task, a tradeoff has been made. That tradeoff has been run-time and conceptual ease for generality. The PhysicalSOM is no longer a generally useful learning technique, because so much of their functioning is tied very closely to the problem of environment-learning, with concepts such as landmark-distance intimately incorporated into the growth and learning mechanisms. This is, for this task, an acceptable price.

Unfortunately one thing which the PhysicalSOM has not improved upon is the number of parameters necessary. One of the difficulties of SOMs is the sheer number of parameters that are necessary for any kind of reasonable learning to go on - and some of these parameters are entire continuous-valued functions. The PhysicalSOM does not cut down on the number of parameters significantly, although I feel that the parameters that it does make use of are more intuitive and easily tweakable.
Chapter 7

Self-Localization

World-space is a fiction. After all, global position is not a quality that can be directly perceived (as local-space position can, for example). In fact, world-space is not a quality of the world at all – a global coordinate frame would imply a global origin, and though some global-origins are useful for some creatures (a wolf might think of its den as the global origin) they are clearly not a creature-independent inherent quality of the environment. In other words, world-space is a fiction, but a useful fiction.

It is a fiction, for example, that has considerable predictive power. A creature could perceive everything in local-space, and then perhaps update the expected local position of every object in the world according to self-motion perception. Or an easier approach might be to hypothesize a "world-space", localize each object in it and then localize itself in it. In this case, the true power of the world-space fiction is its ability to predict local views.

World-space is also useful for object-matching. Sometimes two objects appear perceptually identical, and only the world-space fiction can be used to distinguish between them. This allows for object-persistence (the object probably has the same qualities this time that I perceived it as it did last time) which can again be considered a form of prediction.

Before any of this can take place, however, self-localization – the process of localizing yourself in the world-space – needs to take place.

Self-localization is one of the most important functions of the SpatialSystem.
Though matching to a BMU is part of this problem (in a sense a discretization of the problem for the sake of simplicity for the rest of the behavior system), true self-localization is going to require a much higher degree of precision than simple node-matching will allow. In particular, we would like to get a continuous location and bearing out of the process. Why not simply accept the BMU as the highest resolution of space necessary and move on? One reason is that without correction for drift within a place field (i.e. all locations that are mapped to the same place field) we cannot accurately ascribe locations to objects in the world, nor can we accurately predict where they will be when we haven’t observed them for some time. The point is not, of course, to discover the virtual world’s ”ground truth” world coordinate system (this would in fact be a violation of the abstraction barrier described in section 3.2), but rather to establish one which is self-consistent. That the red box is at \( \hat{l} = 50, \hat{23} \) is not as important as it being 30 units from the blue box and 75 units from the white box.

Our problem of self-localization is made easier by the fact that we almost always have a fairly good initial guess at where we are: we could, for example, always guess that we are in the same place as we were last time. Or even better, we could update our position and bearing according to our estimation of how far we have moved since the last timestep. Animals actually make such estimates, of course, both based on motor proprioception and on acceleration tracking of the Vestibular System of the inner ear.

An inherent circularity should be pointed out here: that we are using landmarks to determine our location, and then using our location to determine the location of the landmarks. When each depends on the other, where does the ”grounding” come from?

The grounding in fact comes from the visual input. Given a particular local view, all the information that can initially obtained is local information, i.e. object offsets from the current location and bearing - whatever they may be. It is precisely this local view information that will form the input to the self-localization algorithm. We will combine this local input with the assumption that the world-positions of all static
objects in the world have not changed to derive an estimate of global position and bearing.

An example: an agent materializes in an environment with three objects (figure 7-1a). The world-positions of the objects are, of course, unknown, as is the position of the agent itself. Since it has never been in this environment before and knowing that any coordinate frame is as good as any other, the agent sets its own location as \( <0,0> \) facing straight north (bearing \( <0,1> \)). This gives the objects the locations \( <-1,2>, <1,2> \) and \( <1,3> \). If the objects are not observed to move over the period that the agent observes them, they will be considered stable enough to be considered landmarks.

Suddenly a malevolent creature-designer beams the agent to another location in the same environment. Again, the three landmarks are visible, but now from a different angle. Since the landmarks have been seen before (and since they are known to be stable) a world location is not immediately ascribed to them. Instead their local offsets will be observed \( <-0.5,2>, <-0.5,1> \) and \( <0.5,1> \), figure 7-1b) and then the existing world map mentally explored for a position and bearing which would be expected to render this local view. And one is found: assuming the landmarks did not move, the agent concludes it is at location \( <2,2.5> \) facing due east (figure 7-1c).

Of course in the pristine environs of the Cartesian plane, such clean answers are possible. Our task, however, is complicated by a number of factors, including noisy perception, unstable landmarks and redundant local views (given a small enough number of landmarks, multiple locations can give the same local view, especially if landmarks cannot be distinguished purely visually. This is a form of what McCallum calls *perceptual aliasing* ([35]). These factors combine to ensure that a pure trigonometric close-formed solution will never be possible. Instead, self-localization will be a process of error-minimization, and the world-position and orientation decided upon will be the one that seems to minimize inconsistencies with the previously-observed state of the world.

Section 7.1 describes a self-localization algorithm that accomplishes this error-
Figure 7-1: The problem of Self-Localization

(a) A creature is thrown into a new environment. Never having seen it, it establishes a new coordinate frame. (b) The creature is transported to a new location in the world. All that the creature can initially perceive is local (body-space) positions of landmarks. (c) The creature figures out how its current local view can fit into its existing world model.
minimization through a form of gradient-descent.

It should be noted that Self-Localization is critical in addressing some of the drawbacks of the LocalViewSpatialSystem. The experiments described in section 5.3 rely on absolute knowledge of a reference location. This shortcoming can only be overcome through some form of self-localization. This means that an orientation will be guessed at based on the current local view and past experience (memories of landmark configurations). Only once an orientation is established can environment-learning then take place, i.e. can world-space positions for objects (including landmarks) be assigned etc. Self-localization can therefore be properly thought of as completing the environment-learning process.

7.1 A Self-Localization Algorithm

Since places are defined in terms of local view, the process of self-localization is essentially a process of determining which location has the local view that best matches the one currently observed (as discussed in chapters 5 and 6, each node in the map has an input-space value that encodes a representative local view). The complication is that the local views encoded in the map assume an invariant reference direction, of which the creature does not have direct knowledge. The process of self-localization will therefore have two steps: orientation-estimation followed by position-estimation.

7.1.1 Orientation Estimation

There is a fair amount of evidence that animals explicitly represent global orientation. Some seem to have direct perceptual access to a reference direction either from sensitivity to the Earth’s magnetic field or from sensitivity to the direction toward the sun coupled with an accurate internal clock to account for daily changes in sun’s azimuth (e.g., [53]). Not all animals seem to have such mechanisms, however, and instead must form their own conceptions of reference direction. Often these seem to be tied to visual cues, such as prominent polarizing landmarks.

The case for explicit modeling and tracking of a global reference direction in
mammals is strengthened by the existence of head-direction cells, cells that seem to be sensitive to the offset of the animal from a globally-aligned reference direction. These cells are found in a number of brain structures, including the post-subiculum, the anterior thalamic nuclei and lateral dorsal nucleus of the thalamus (see [55] and [56]). In the Virtual Hippocampus orientation tracking is absolutely necessary for the very practical reason that panorama-vectors are not comparable until they are aligned to a global reference direction.

There are two ways of estimating orientation:

- We can use explicit "orientation landmarks", a set of objects that are observed to have constant or near-constant bearings-from-north. Far distant stationary objects are prime candidates. If a mountain in the distance has consistently been observed to be in a direction 30° from north, its observation on the next timestep can be used to determine a likely heading for the creature. Quite simply:

\[ \theta^t = \varphi_{average} - \gamma^t_{observed} \]  

(7.1)

\( \theta^t \) is the estimated world-bearing of the creature at time \( t \), \( \varphi_{average} \) is the observed average of the angle between the orientation-landmark direction-vector and the reference direction (north) and \( \gamma^t_{observed} \) is the body-space bearing of the landmark direction vector as observed at time \( t \). See figure 7-2.

If multiple "orientation landmarks" are visible then the average of their suggestions can be taken as the current bearing.

Chapter 8 describes how these orientation landmarks are chosen.

- If no orientation landmarks are visible, then ordinary "position landmarks" can be used to approximate an orientation. In a process similar to that used in the gradient descent, memories of recent landmark configurations can be recalled to generate an expected local-view at the current position, given with respect to the reference direction (i.e. expected local view if the creature were in the current location facing straight north). The current panorama should be similar to the
expected one, though at a different orientation. To normalize the orientation of
the observed panorama, we can calculate a local reference direction (LRD), a
weighted vector average of all the landmark vectors (see figure 7-3). The LRDs
of both vector sets can be computed and then the observed LRD rotated to
coincide with the expected one. Thus, very similarly to the above orientation-
landmark calculation, the current bearing is calculated as

$$\theta^t = \phi_{expected} - \phi_{observed}$$

(7.2)

Where $\phi_{expected}$ represents the angle-from-north of the expected panorama’s
LRD and $\phi_{observed}^t$ is the body-space bearing to the LRD as observed at time $t$.

The calculation of an LRD is a useful technique for comparing orientation-
-independent local panoramas. It is was taken from [2], which uses a similar
scheme to generate vectors to goals for navigation purposes.

Path integration, the updating of estimated position and orientation based on
vestibular and self-motion cues, is assumed to occur every timestep. The techniques
above are intended to reconcile path-integrated orientation estimates with visual cues.
It is hoped, in other words, that orientation-estimates provided by the above tech-
niques will represent small corrections to the path-integration estimate. If a single
decision-point for the weighting of the influence of visual cues versus path-integration
cues is desired, this is it. The two estimates could be combined simply as

$$\theta^t = \alpha \theta^t_{visual} + (1 - \alpha) \theta^t_{path-integration}$$

(7.3)

Where $\alpha$ is the weighting factor for visual orientation cues. This factor could conceiv-
ably be made to increase over time, such that on initial entry into an environment
the creature uses only path-integration cues in order to facilitate the choosing of
landmarks, while later relying primarily on visual cues.

Of course, if no visual estimate is possible (if no landmarks of either type were
visible, for example) then the path-integration estimate is the one accepted.
Figure 7-2: Deriving global bearing from orientation landmarks

(a) Given a local view, the creature may not know what the global orientation of the view is. (b) However, given that the orientation landmark (the circular landmark) has been observed most timesteps at global bearing $\varphi_{average}$, a global orientation can be derived.

Figure 7-3: Bearing-independent comparison of local views

(a) Two local views are difficult to compare (b) until their LRDs are computed and aligned.
7.2 Position Estimation with the LocalViewSpatialSystem

We can think of the SOM lattice as a function mapping 2-vectors in the topology space to an input-space that encodes local views in the form of panorama vectors. This mapping from topology-space to input-space is defined by the discrete samples provided by the lattice nodes. Mapping points not represented precisely by nodes is a matter of interpolating the input-space values using a multidimensional interpolation technique such as the hyperplane/rbf-fitting technique described in appendix A.

In self-localization, however, it is the inverse mapping we are interested in - we would like to find a topology-space point that maps to a panorama similar (preferably identical) to the panorama we are currently observing. That point will be our current “world” position.

While there is certainly no close-formed inverse to the hyperplane/rbf function, we can use an iterative solution technique such as gradient descent. The hyperplane/rbf function is, of course, highly nonlinear, and would be riddled with local minima that would prove a problem were it not for the fact that we almost always have a very good initial guess - our last known world position.

The hyperplane/rbf interpolator is useful for another reason: that only a subset of available points can be used in its construction. In the case where we are fairly sure that we are near our last position, the interpolator can be constructed using only nodes in a small neighborhood around the suspected solution. This neighborhood can be found by taking nodes within a given number of topological steps of the BMU. The interpolator can be queried at any point in topology-space of course, though its results will be more accurate in the neighborhood of the examples.

The calculation of the gradient is fairly straightforward since the Hyperplane/RBF interpolator function is differentiable. The error function we are trying to minimize is

\[ E(x, p_{observed}) = (f(x) - p_{observed})^2 \]  

(7.4)

Where \( f(x) \) is the output of the interpolator, \( x \) is the current position estimate (a
2-dimensional vector) and \( p_{\text{observed}} \) is the observed panorama (encoded as described in section 5.1). Differentiating we obtain

\[
\nabla E(x, p_{\text{observed}}) = 2J(x) \left( f(x) - p_{\text{observed}} \right) \tag{7.5}
\]

Here \( J(x) \) is the jacobian matrix of the Hyper-plane/RBF interpolator function, \( f(x) \) (see appendix A for details and derivation). It is a 2-by-2 \( \times l \) matrix (where \( l \) is the number of landmarks) and so is compatible for multiplication with the 2 \( \times l \)-dimensional \( p* \) to produce a 2-vector representing the direction of steepest ascent.

The overall gradient descent is implemented as follows:

1. Generate an initial position estimate, \( x \).

2. Calculate the current error

\[
e = E(x, p_{\text{observed}}) \tag{7.6}
\]

3. Generate an initial learning rate, \( \lambda \).

4. Calculate the gradient at the current position.

\[
v = \nabla E(x, p_{\text{observed}}) \tag{7.7}
\]

5. Generate a new position estimate using

\[
x^* = x + \frac{\lambda v}{||v||} \tag{7.8}
\]

6. Calculate the new error

\[
e^* = E(x^*, p_{\text{observed}}) \tag{7.9}
\]

7. If \( e^* > e \), then return to the previous estimate of space and halve lambda

\[
x \leftarrow x \quad \lambda \leftarrow \frac{\lambda}{2}
\]

If \( e^* < e \) then update the current position estimate and double lambda

\[
x \leftarrow x^* \quad \lambda \leftarrow 2\lambda
\]
8. Repeat steps 4 to 7 until the error is below a threshold or \( \lambda \) is below a threshold.

There are two ending-conditions for this algorithm because (a) we know that we are close enough when the error drops to near-zero and (b) we cannot be sure that a zero error is possible every timestep - it could be that the best we can do is a significantly high non-zero error. In this case, we simply wish to say that if our rate of change (measured by \( \lambda \)) is near-zero, then there’s probably no point in going on.

Redefining ”World Space”

It should be noted that an interesting thing has happened to our definition of ”world space”. We have essentially equated it to the topology-space of out SOM. In other words, when our synthetic creature localizes itself, it does so with respect to the 2-dimensional space on which our SOM is laid out. Likewise, when we localize objects within the environment, the positions we ascribe to them will be points in the SOM’s topology space - the local views, essentially, that would be expected from the objects’ positions.

This new space will cause us headaches. It is not a metric space, and relationships of distance and direction have no direct mapping onto physical space. On the other hand, adjacency relationships are retained, as is the concept of a reference direction. And the two principle reasons for having a concept of world-space at all - the ability to generate local view predictions and the ability to match object-locations from one timestep to the next in a self-position independent way - still apply.

Path Integration in the LocalViewSpatialSystem

Path integration could be used to generate initial guesses at world position, though this is more difficult than it might seem, since the path integration information in the virtual realm will likely consist of metric differences between positions on subsequent timesteps, and the SOM deals not with metric space but with the highly non-metric topology space. The mapping of this self-motion information between one and the
Figure 7-4: Landmark memory

Memories of recently-observed landmarks (left) can be used to complete the local view for a creature with a limited field of view. However the remembered landmark vectors must be updated through path integration.

other is therefore not obvious.

However, a simpler way in which path integration can be incorporated is through memories of recent local views. This is especially important for creatures with limited fields of view, which cannot possibly capture the entire 360° scene in a single observation. In order to correctly “gel” the map in its early learning phase, that full view of the scene is necessary. It can be provided by remembering recent local views and updating them appropriately with self-motion perception. If a landmark is observed on a subsequent timestep then its prediction can of course be discarded. If it is not, however, this landmark memory can be used as an estimate of the landmark’s current local (reference-direction-aligned) position and be used in self-localization. See figure 7-4.

These estimated landmark vectors can decrease in confidence with the time since they’ve been seen. After the creature’s confidence in a landmark is below a threshold it can stop using it in the self-localization process.
7.3 Position Estimation with the CartesianSpatialSystem

Self-Localization with the CartesianSpatialSystem is far easier, since the PhysicalSOM nodes "live" in the same (metric) space as the creature. Each landmark has a previously-estimated world-space position, \( x_{\text{landmark}} \), and so, coupled with the creature’s current observation of it, suggests a world-space position for the creature:

\[
x_{t+1} = x_{t_{\text{landmark}}} - v_{t_{\text{landmark}}}^{t+1}
\]

(7.10)

where \( v_{t_{\text{landmark}}}^{t+1} \) is the world-aligned local offset to the landmark observed at time \( t+1 \). The final world estimate can then be an average of all these "suggestions"

\[
x_{t+1} = \frac{\sum_{i=0}^{N} x_{t_{\text{landmark}i}} - v_{t_{\text{landmark}i}}^{t+1}}{N}
\]

(7.11)

Note that his could again be thought of as reconciling the expected and observed local views. The expected local view of a single landmark is

\[
v_{t_{\text{expected}}} = x_{t_{\text{landmark}}} - x_{t}
\]

(7.12)

where \( v_{\text{expected}} \) is the expected world-aligned local landmark vector. The error vector would therefore be the average of the discrepancies between observed and expected local views:

\[
e = \frac{1}{N} \sum_{i=1}^{N} v_{i_{\text{expected}}} - v_{i_{\text{observed}}}
\]

(7.13)

See figure 7-5. It can be shown that position estimate of equation 7.11 minimizes the magnitude of the error vector produced by equation 7.13.

7.4 Object Localization

Once the creature has localized itself with respect to its own world model, it should rightly go back and recalculate the world-positions of the visible objects and land-
Figure 7-5: Calculating the Error vector in the CartesianSpatialSystem

marks in the world.

Remember that when bounding-boxes and retinal locations are first extracted from the framebuffer they are not yet matched into an object history. In order to match the incoming belief with a persistent one already in Working Memory, the retinal coordinates need to be transformed into World-space - so that the creature’s own motion does not affect the position-distance calculation. Self-Localization, however, has not yet occurred, and so the retinal-to-world transformation is based only on the creature’s estimates of its world position and orientation - again taken from perception of self-motion and its position in the previous timestep. Poor estimates of self-location result, of course, in poor estimates of object-location. Through self-localization, the retinal-to-world mapping is refined (this time by using visual landmark cues as described in this chapter). It therefore becomes possible to go back to the objects that were observed in the current timestep and regenerate their world-positions based on the original retinal-space position extracted from the framebuffer and the new retinal-to-world mapping computed by the Self-Localization algorithm.

This Object Localization process is especially important for the stability metrics that will be described in the next chapter. If a creature has exceptionally poor path-integration abilities then even stationary objects will appear to move when the creature moves. After Self-Localization has corrected for this deficit, however, the stationary objects will be found to be in the same place they were before.
7.5 Example Results

There are two classes of problems that the process of self-localization solves: position recognition and position-tracking.

Position recognition

When a creature is introduced to a new environment, it is allowed to make assumptions about where it is, since its position initially is arbitrary. However, when a creature is reintroduced to a known environment it must be able to recognize its position from memory. This can be simply re-stated as a self-localization problem with low (zero) confidence in initial position and orientation. This has two effects:

- When matching the creature’s BMU we have typically used a euclidean distance similarity function. We can substitute this for a bearing independent similarity function, which compares two panoramas by first normalizing their orientations (by aligning their LRDs) and then finding the euclidean distance.

- Behaviorally, at this point, it is a good idea to stop and look around. This allows the creature to complete its local view (if its field of view is limited).

Figure 7-6 shows the results of one series of experiments, in which a creature with a limited field of view was reintroduced into a previously learnt environment. This creature was using a LocalViewSpatialSystem. Although upon initial reintroduction the creature’s position estimations are very poor (due to its incomplete local view) upon looking around it is able to find its correct position in topology space.

Unfortunately, the above method does sometimes result in minor inaccuracies in bearing-determination, which in turn can lead to place field shifting. This is because there are sometimes significant differences between the LRD of the observed panorama and the LRD of the BMU’s input-space value. These discrepancies can, however, be corrected for through further exploration of the world and observation of more concrete orientation cues (such as orientation landmarks).
Figure 7-6: Position recognition

(a) shows the creature in its learned environment and its corresponding mental model of its position in topology-space. In (b) through (d), the creature is reintroduced to a random point in the known environment. By looking around and completing its local view, it progressively refines its model of where it is in the world.
Figure 7-7: Position tracking

Position Tracking

The more common (and easier) form of self-localization is position tracking, where the creature must update its position from timestep to timestep, possibly taking into account self-motion information. This task is easier because the creature’s previous estimate of position and orientation serve as good initial guesses on the subsequent timestep. Figure 7-7 shows a trace of the creature moving about a simple world with the corresponding topology-space motion shown (7-7b).
Chapter 8

Reliable Landmarks

In the previous chapter a major assumption was made: that the creature knew which objects in its world made good landmarks. In this chapter that assumption is going to be justified by showing that creatures can learn landmarks on the basis of its experience with the world. This landmark learning process is designed to occur simultaneously with environment learning and self-localization. Indeed, it cannot truly occur without them.

What do we want in a landmark? Since we are going to be using them for self-localizing, the most obvious answer is that we want them to be stationary. Determining whether an object is stationary or not, however is not as cut and dry as we might think. After all, even humans make mistakes in this task under certain circumstances. As we sit on a train, we get the distinct sense of forward motion as we watch another train on a neighboring track slide by ... until we realize that it was the other train that was moving and we had been standing still.

Perhaps our intuition says that it should be easy to tell when objects are stationary because the accelerometers in our brains are so good at telling us when we are moving. This is usually true, but the previous example shows that there is a very strong visual component to self-motion perception (many Disney World rides depend on the same phenomenon, presenting overwhelming visuals which, combined with the physical rocking back and forth of the room to evoke acceleration and deceleration effects, gives audiences an overwhelming sense of motion). In the end all that one can
truly tell, the bias that the vestibular system provides notwithstanding, is that the distance between one’s eye and another object is increasing or decreasing. Which one is moving is a guess.

Ultimately, however, all frames of reference are arbitrary and equally valid. The difference between them is not "correctness" but utility - predictive utility, to be more precise. Fundamentally, our landmark choice is supposed to help us self-localize better. And what is the point of self-localization beyond that it provides a creature with excellent object-location prediction abilities? Rather than keeping track of all objects in the world, we keep track of one object - our bodies - and assume that everything is probably more or less where it was before.

Path-integration certainly has a place in landmark selection (again, as a bias for one reference frame or another) and it has a definite place in the algorithm described here. This chapter will show, however, that it is possible to perform landmark selection - and more fundamentally reference frame selection - without any kind of path integration at all. This should be interesting not just for the sheer theoretical glory of it, but also for all those budding physical synthetic creatures - robots - whose path integration systems still leave much to be desired.

8.1 Defining "Landmark"

The problem of landmark selection has been approached in many different ways. [4] and [3] tackle it from a biological perspective, studying how perceived position-stability effects an animal's conception of space. On the opposite end of the spectrum (and my bibliography), [58] discusses landmark selection for mobile robots using a Bayesian framework.

For the purposes of the current work, "landmark" will be taken to mean a recognizably stable point in space. In general these will be positions of stationary objects. Thus a rock that has never been observed to move might be chosen as a good landmark. We can also choose as landmarks certain physical features of objects. For example, since the location of an object is considered to be the centroid of the
currently-visible portion of it, a wall makes a poor landmark (since its position jumps around depending on what part of the wall the creature is looking at). On the other hand, the *edges* of the wall are excellent landmarks, because they are unambiguous and are never seen to move (unless the wall itself moves, of course). Since object-edges are extracted by the visual system of C4, and since their position can be represented with a *Spatial Model*, object edges are indeed used as landmarks.

Note that this definition of landmark is nonetheless very limited. Distinctive physical features, variations of the landscape, areas of distinctive coloration, etc. should all be usable as landmarks. The exclusion as potential landmarks of physical features well-characterized as single points in space is simply a limitation of the perception system as currently implemented under C4. These features are, in principle, fully treatable by the algorithms described in this chapter. (It is arguable that features not reasonably-characterized by single points in space are not landmarks!)

### 8.2 Stability

One simplifying assumption we can make is that the landmark status of each object in the world is binary: either the object is considered stationary, or it isn’t. This is in contrast to a system which would confer upon objects a "degree of stability". The gradient descent algorithm of chapter 7 could indeed support a "degrees of stability" approach, perhaps through a selective weighting of the error vector according to how stable each corresponding landmark is considered to be. It is questionable, however, whether this would in fact be a good idea. After all, when self-localizing, once we realize that an object is moving, we would want its weight to drop completely to zero, rather than weighting it less. The self-reinforcing aspect of the also tends to have the negative side-effect of "playing favorites", whereby a single object ends up as the only position landmark (the self-reinforcing dynamics lead to a winner-take-all situation). It is also possible for this single landmark to be a moving one, resulting in completely incorrect behavior. To avoid this, landmark status is kept as a 1 or a 0: either it is included in the self-localization gradient descent and given equal weight with all
other landmarks or it is excluded altogether.

This is not to say that a "degree of stability" metric is not useful. On the contrary, how else are we to decide which are the landmarks in the first place?

Chapter 7 briefly discussed two forms of landmarks, position and orientation landmarks. This distinction reflects the two-fold nature of the self-localization process itself: the creature first must determine its orientation with respect to the world, and second must determine its position. In both of these tasks, the creature must leverage its experience with the world: objects that have been stable in bearing-to-reference should be used for determining orientation. Objects that have been stable in location should be used for determining position.

As it turns out, it is more mathematically convenient to define the inverse of stability, i.e. mobility.

### 8.2.1 Position-Mobility

A simple mathematical formalism which captures our concept of mobility is variance. In the case of a one-dimensional signal, variance is the expected squared distance of the signal value at a given time from its mean. Assuming that a buffer of $n$ signal values is remembered, the variance of the remembered portion of the signal can be computed as

$$\text{var}(x) = \frac{1}{n} \sum_{i=0}^{n} (x_i - \mu)^2$$  

(8.1)

where $\mu$ is the mean all $n$ samples of $x$. This function returns a single scalar result. In the case of a 2-dimensional signal (which is what we have when measuring 2-dimensional position variance) the variance would be expressed as a $2 \times 2$ covariance matrix. Since in this case variance direction does not interest us, we could simplify this by measuring, as the one-dimensional case defines, the average squared distance from the mean

$$\text{mobility}(x) = \frac{1}{n} \sum_{i=0}^{n} (x_i - \mu)^T (x_i - \mu)$$  

(8.2)
where both \( x \) and \( \mu \) are now vectors.

Note however, that this metric does not quite render the effect we would like. Consider an object in the creature’s world that has been perfectly stable for as long as the creature can remember, and that suddenly, one particular timestep, takes off zig-zagging at top speed through the world before returning to its original position a few timesteps later. Since this brief, though extreme, motion lasted only a few timesteps out of a memory potentially thousands of timesteps long, its impact on the average deviation from the mean is actually fairly minimal - it certainly doesn’t capture the fact that we now wish to discount this object entirely as a potential landmark.

What is true, however, is that at the extremity of its motion, the object’s instantaneous deviation from the mean was very large. It is this recent large deviation that we wish to remember.

The approach I have taken, therefore, is to run the incoming deviation-from-mean signal through an asymmetric low-pass filter. This filter has fast rising edges and very slow (and therefore long-remembered) falling edges. The update rule is

\[
\begin{align*}
\text{if}(|x_t - \mu| > m_{\text{position}}) & \quad m_{\text{position}} \leftarrow \alpha_{\text{rising}} m_{\text{position}} + (1 - \alpha_{\text{rising}}) |x_t - \mu| \quad (8.3) \\
\text{else} & \quad m_{\text{position}} \leftarrow \alpha_{\text{falling}} m_{\text{position}} + (1 - \alpha_{\text{falling}}) |x_t - \mu|
\end{align*}
\]

Values for \( \alpha_{\text{rising}} \) vary from 0 to 0.9. The much slower \( \alpha_{\text{falling}} \) is typically given a value like 0.99. This asymmetry reflects the fact that significant movement very quickly increases the mobility of an object, and that we are slow to re-confer landmark status on an object after we have seen it move.

### 8.2.2 Orientation Mobility

Just as we need to find a way to rate the position-stability of objects in the world, we also need to rate the constancy if their bearings-from-north. Note that this is not a matter of finding the stability of the object’s body-space bearing - this value changes with the creature’s orientation. Rather the creature’s orientation must be ”normalized
Two landmarks are observed at time $t_0$ and $t_1$. Note that the global bearing towards landmark $a$ ($\phi^{t}_{a,\text{global}}$) changes dramatically, whereas the bearing toward $b$ remains more or less the same.

out” of the equation, so that the bearing depends only on the creature’s position. Figure 8-1 illustrates how a nearby object shows poor global-bearing stability while distant objects show high global-bearing stability.

The calculation for Orientation Mobility is very similar to that for Position Mobility. At each timestep, the body-space landmark direction vector is rotated by the creature’s current global bearing-from-north, such that the landmarks global bearing-from-north is

$$\phi_{\text{global}} = \theta + \phi_{\text{body-space}}$$  \hspace{1cm} (8.4)

where $\phi_{\text{global}}$ and $\phi_{\text{body-space}}$ are the global and body-space bearings of the landmark in question and $\theta$ is the current world-bearing of the creature itself.

A 2-dimensional unit vector in the direction of $\phi_{\text{global}}$ is fed through an asymmetric low-pass filter, just as in the previous section.

$$ if \ (|v_t - \mu| > m_{\text{orientation}}) \quad m_{\text{orientation}} \leftarrow \alpha_{\text{rising}} m_{\text{orientation}} + (1 - \alpha_{\text{rising}}) |v_t - \mu| $$  \hspace{1cm} (8.5)
\[
\text{else } m_{\text{orientation}} \leftarrow \alpha_{\text{falling}} m_{\text{position}} + (1 - \alpha_{\text{falling}}) |v_t - \mu|
\]

where \(v_t\) is a unit vector in the direction of the global bearing computed in equation 8.4 at timestep \(t\).

Note that, since the vectors being compared are normalized, the orientation-mobility calculation does not render a metric that is directly comparable to the position-mobility (you could not, for example, render a value for total mobility by summing the two). However, since we are only going to be comparing orientation-mobilities to other orientation-mobilities, this does not cause problems.

## 8.3 Choosing Landmarks

So far we have merely defined two metrics for stability. We have not yet discussed how these metrics will be used to decide which objects should be considered landmarks.

One simple approach might simply be to set a mobility threshold, so that objects with mobilities below that threshold are considered stationary, and objects with mobilities above is are considered mobile. This approach would have problems, however, due to the creature’s initial uncertainty about the landmark-status of objects in its world. In this early stage, it is possible that everything would be seen to be moving and so even stationary objects would have high mobilities. In this case it would be dangerous to conclude that there are no stationary objects in the world, because the creature could then not self-localize at all.

Another danger to avoid is choosing a singe landmark as both orientation and position landmark. This is, unfortunately a completely stable error condition, and simply has the creature orienting with respect to and localizing with respect to a single landmark. The result is that the creature and the single landmark seem never to move, and the rest of the objects in the world spin around the landmark as, in actuality, the creature moves about the world. This danger, fortunately, is easily avoided by simply ensuring that an object chosen as a position landmark is never also chosen as an orientation landmark.

The landmark choice algorithm is as follows
1. At the beginning of time all objects are labeled as position landmarks (and not orientation landmarks) and are ascribed a 0 position mobility.

2. Each tick, the lowest position-mobility is found.

3. Any objects with mobilities of less than twice the lowest position mobility are chosen as position landmarks.

4. Of the remaining objects, the lowest orientation mobility is found.

5. Any of the remaining objects with orientation mobilities of less than twice the lowest orientation mobility are chosen as orientation landmarks.

This procedure relies on the fact that even if incorrect initial landmark-choices are made, the stationary objects will still appear more stable than non-stationary ones.

Assume a world with a number of stationary objects and a single moving one. A creature will include the moving object in its initial choice, and as a result the self-localization process will come to rest on a location and orientation that makes it appear that everything has moved. Nevertheless, the greater number of stationary objects (and their agreement with the creature’s own self-motion perception) keep the mobility of the stationary objects lower than that of the actual moving object. Eventually the mobility of the moving object rises above twice the mobility of the most stable object, and it will be discounted from the self-localization process.

Note that the above algorithm allows for there to be no orientation landmarks at all. In this case, orientation determination relies simply on local cues and self-motion perception.

8.4 Example Results

Selection of orientation landmarks

In a simple environment consisting of a few local landmarks and a single distant one (see figure 8-2a), the distant object is very quickly singled out as a good orientation landmark. This usually occurs a few frames after the creature has begun moving.
Selection of Stationary Landmarks

In another case, the creature was placed in a seven landmark environment (one distal landmark, and six local ones, see figure 8-2b). Of the local landmarks three were moving randomly. Again, within a few seconds of observation, the moving landmarks are found and their landmark status switched off.

Alternate Frame of Reference

In this case the creature was placed in a world where all but 1 of 7 objects (one distal) were moving in the same elliptical manner (see figure 8-2c). The creature very quickly picks out the distal object as an orientation landmarks, but takes a while longer to start ruling out position landmarks – all local objects are considered landmarks, and so all of them, and the creature itself, appear to be moving. Eventually however, the single stationary object is ruled out, and the identically-moving objects are taken as forming a stationary frame of reference. The creature and the stationary object appear to be moving in this frame of reference in the opposite elliptical path. This is a case in which the Landmark and the Self-Localization systems manage to come up with a "simplest solution" that fits all the observed information – in this case, that it is itself and a single other object that are moving, rather than everything else in the world.
Figure 8-2: Landmark reliability learning

Left column: the mental (Cartesian) map formed by the creature. Red squares indicate position landmarks, circles represent orientation landmarks and triangles non-landmark moving objects. Right column: The actual environment, with object motion where indicated.
Chapter 9

Summary of Environment Learning

This system has described an integrated system for environment learning. Since we started this section with a summary of biological place cell properties, it would seem appropriate to return to those properties and see how well we have done in reproducing them.

**Place fields are formed soon after initial entry into a new environment.** Place-field formation in the SpatialSystem is also very quick - too quick perhaps in the case of the CartesianSpatialSystem. The time-scale on which the LocalViewSpatialSystem learns an environment would seem more true to the observed biological time-scale.

**Distal Landmarks are crucial to orientation.** Indeed, through the definition of orientation-landmarks as objects of constant or near-constant bearing-to-north, we have shown a dependency on more distal landmarks for determining global orientation.

**Place fields persist when landmarks are removed.** Because path-integration provides an ”expected landmark vector set”, place fields will persist whether the landmarks are actually observed or not (if only one or two landmarks are observed, or if the ”lights” go out).

**Place field shift with experience.** Place fields can easily be made to shift with
experience, though not in the way described by Redish.

**Place field are directional along practiced routes but not in open environments.**

No place cells are direction-sensitive in C4.

**Place cells are sensitive to more than location.** Places cells are not sensitive to non-spatial cues.

We have gotten about 50% in this breakdown. This is not, however, too bad a result. The intention had not been to reproduce every aspect of hippocampal function, but rather to take from it what was useful. Some of the aspects that we have left out are interesting, but are of unclear use at this point. Directional place-cells may be useful for path-planning, but it is unclear whether they would necessarily do better than non-directional ones. And the biological sensitivity to non-spatial cues is fascinating, but it would require more research into the definition of ”context” and ”task” before it could be meaningfully implemented in C4.

**LocalView versus Cartesian SpatialSystem**

We have also developed two parallel models of Hippocampal environment learning. The question is, which is better?

The answer is, as it has to be, neither. The LocalViewSpatialSystem seems to produce more appropriate cognitive maps and also shows more similar dynamics to those shown in actual Hippocampal assemblies. On the other hand, its run-time demands and its current incapacity for growth make it inconvenient for incorporation into a creature that will persist for any length of time in an environment of any complexity. This is most likely simply a matter of more research. Though the critical period aspect of the SOM learning has already been cited several times as a weakness of the approach, there is some indication that real cognitive maps go through a similar process. McNaughton suggests that full spatial mapping involves the allocation and management of multiple frames of reference ([36]). If a scheme for allocating and switching between these frames were devised, then perhaps the SOM approach could
be more viable, since each reference-frame SOM would contain only a few landmarks and would not need to be that extensive.

The development of the CartesianSpatialSystem was an engineering contingency. PhysicalSOMs are easily controlled and easily learnt. Unfortunately, they are also correspondingly less robust. If the animal makes an incorrect position-determination for some reason on a certain timestep, then all the objects could be localized in a correspondingly incorrect location. The LocalViewSpatialSystem, by contrast, is more robust to single-timestep errors.

Fortunately, the processes of self-localization and landmark-selection function under both representations of space.

9.1 Summary of the observation / object-matching / self-localization / object-localization loop.

It is, finally, informative to go back and look at the high-level view of this environment-learning system. Figure 9-1 shows all the major components of the spatial system so far.

Observation begins every timestep, with object locations being received visually in the retinal-space of the creature. The first thing to happen is that these retinal positions are transformed into world-positions by the creature, where the retinal-to-world space mapping is based on the creature’s body configuration (where the mapping to body-space is known more or less accurately) and the creature’s estimate of its own location and orientation in the world. This estimate is a combination of last known/perceived location and proprioceptive/vestibular information. Once in world-space, the objects can be compared to objects already in the Working Memory persistent Belief repository. Usually it will be matched up with a persistent Belief, and its data added to that belief’s data-history. When object-matching is done, the objects registered as orientation-landmarks are used to estimate the creature’s global orientation (taking again the creature’s prior estimation of its own global orientation
as a starting point). With this global orientation determined (or approximated), a gradient-descent is performed to match the bearing-adjusted local-view to the world-model already contained in the SpatialSystem. From this process we get a new estimate for the position of the creature. With this new location, the world-positions of observed objects are updated (in combination with the local offsets that were actually observed). Finally, the stability metrics are recomputed for each objects, and where applicable, the landmark-status of objects is updated.

This part of the thesis has described a system in which a hippocampus-style environment-learning structure has been developed and used to solve problems of spatial cognition. It describes a single integrated system for environment learning, self-localization and landmark learning, all occurring simultaneously in a full autonomously-acting synthetic creature. In addition to showing some of the same place-field formation dynamics seen in real animals, the design has, overall, shown itself to be practical, to be capable of running in real-time and to contribute tangibly to a creature’s outward behavior and to its sense of life.
Figure 9-1: Summary of the Environment Learning Loop
Part III

Object Persistence
So we have our map. Now what do we do with it?

We have already been using it to keep track of ourselves in the world. What about everything else?

We return to the idea of place-field as percept - as a quality that can be attributed to objects in the world.
Chapter 10

Object Persistence

The developmental psychologist Jean Piaget famously noted that there seemed to be an age at which children suddenly became more spatially savvy. In one of his experiments, infants were shown a toy which clearly held their attention. When the toy was hidden under a cloth, a three-month-old child would simply seem to forget about it. A five-month-old, on the other hand, would continue to observe the cloth, waiting perhaps for the toy to re-emerge. A nine-month-old would lift the cloth.

These findings seemed to suggest that the toy had an existence beyond the immediate perceptual image for the five-month-old but not for the three-month-old. Piaget called this phenomenon Object Persistence, and was one of the notable developments observed in his third and fourth stages of development. (See [17] for a summary of Piagetian development).

Why is an entire third of a thesis ostensibly about "Spatial Common Sense" being spent on the psychological phenomenon of object persistence? Perhaps it was not framed this way by Piaget, but object persistence can be thought of as the correct formation of location-expectations. The five-month-old continues to look at the cloth because it has a mental image of the toy as something beyond its immediate perceptual image, and more importantly with a specific location that is not currently within view. If the location of the toy were not part of its mental model, how would it know where to look? What would be the behavior of an infant that did have object persistence but did not maintain an accurate model of location? The answer
is, probably something similar to the three-month-old, who appears to lose interest once the toy is out of sight.

The point is, whether an infant - or a synthetic creature - has a persistent image of the desired object or not is a rather moot question without a model of location to distinguish the two. The Working Memory structure described in section 3.3 provides half the solution - the Beliefs that it contains persist whether new perceptual information is added to them on the latest timestep or not. The location-persistence it provides is rudimentary - unless an object is observed in a new location, the previous one is assumed.

Animals clearly have a more sophisticated object persistence system than this, one that not only fills in missing perceptual data but also predicts it. For dogs, the impulse to predict likely locations is so strong that it will ignore perceptual data ... going after that thrown ball with gusto even though it never left your hand.

But the dog has another ability - the ability to look around, not find the ball and eventually come back to you. The "looking around" is an extremely informative task, since it is confirming for the dog locations where the ball is not. The dog will look around a small area around where it expected the ball to land, and eventually give up. In other words, it looks in places where the ball most likely is given its last observed position and velocity, and it intelligently updates its conception of "most likely place" based on a steady stream of negative information. Which places are likely depend strongly on the structure of the physical environment: the ball couldn’t have flown through a wall, on the other hand it might have rolled under the couch. One place will be checked, the other will not.

This is what spatial common sense is: the ability to guess at an object’s location and to update that guess intelligently based on the physical structure of the world.

10.1 Models of Expectation

There are a number of different types of expectation. Statistical expectation, for example, describes the likely value taken on for a variable that is observed discretely,
and that comes from a static distribution. They typically describe discrete events, such as the role of a die, or coin toss. The expected value of a variable with a gaussian distribution is the value on which the gaussian is centered.

For our purposes, expectations are going to take a more complicated form. Most of them will depend, for example on recently-observed values. The prime example is location: if the object was last observed in a certain location, then it is mostly still near there (or, to be more specific, in a nearby place that's not currently in view). The expectation becomes even more precise when we include velocity and acceleration information.

Kline (see [27]) provides an excellent discussion of expectation theory. One of his most important conclusions is that a failure to observe is just as significant an event as an actual observation. After all, it would look nonsensical for the dog to run to the place where it predicted the ball would land and simply stop, staring at that place as if the ball actually were there - without, in other words, incorporating the negative knowledge or negative observation that the ball is not in that location.

Kline describes a 2-by-2 grid of possible observation states. (See figure 10-1). Along one axis is whether or not an observation was made. Along the other axis is whether an observation should be able to be made. The two "acceptable" possibilities are therefore if an observation was expected and made and if no observation was made and none was expected. On the other hand the two error states are if something was observed when it was not expected to be observed (a form of surprise) and if an object was not observed when it was expected to be.

These latter two states are known as Expectation Violations, and a great deal of expectation-intelligence is given to dealing with them. Kline describes behavioral responses to expectation violations (showing surprise, confusion etc.). However, he does not address the fact that a large part of dealing with expectation violations should be revamping the mental model to come up with a new expectation that is consistent with all available knowledge.
From [27]. This work classifies object-observation status into one of four categories every timestep. Two indicate consistency with expectations (accept-1 and accept-2) and two indicate error states (reject-1 and reject-2).

10.2 Negative Knowledge

The problem of location-prediction was tackled in [27] as well. In that implementation, a Kalman filter (see [23] for a reference) was used to track observed position and then generate linear predictions for timesteps in which the tracked object was not observed. This implementation had the problem, however, that meaningful "backup" predictions could not be made when the main prediction was invalidated (see figure 10-2). The difficulty with a Kalman-filter or a similar filtered approach, as this example illustrates, is that the Kalman filter does not readily incorporate negative knowledge (i.e. a prediction has been observed to be wrong) into its ongoing state predictions.

10.3 Negative Knowledge with Spatial Extent

Though there may not be an obvious way to do it, it might be imagined that a kind of "prediction invalidation" mechanism could be built into a Kalman filter that would more or less do the correct thing when a prediction is observed to be wrong. However, in the case of location-perception, even this would not be enough. Consider the example (which will surely be revisited) of a ball rolling behind a wall. At some
point, the Kalman filter predicts, the ball should roll out again. When it fails to, however, simply providing a single "prediction is wrong" signal is not enough, because what we have in fact is a range of locations at which the ball is known not to be. The Kalman filter would have to be able to incorporate, therefore, not just a discrete "invalid prediction" signal, but an entire continuous range of "verifiably false" values.

What would be the correct response to a continuous range of verifiably false values? Figure 10-3 illustrates the answer. The probability of the ball being in a visible location is set to 0. However, for the visible regions, the location-distribution should retain the same shape (albeit scaled up perhaps) as before. Note that the response should not be to refit a gaussian to the resulting distribution, since if this were done, we would never expect the ball to come out from behind the wall because the distribution would constantly shift to make somewhere near the center of the wall the most likely position. Our representation must be able to take the form shown in figure 10-3b, where the most likely position is just inside the right edge of the wall. This distribution is a sort of truncated gaussian, a form which cannot be meaningfully incorporated into the Kalman-filter framework, and can be represented in a continuous fashion at all only with great difficulty (and perhaps not at all, since the shape of the truncation region in two-dimensional space is completely arbitrary).

The Kalman-filter or a similar approach could appropriately be used in any case where a wrong expected value was always accompanied by an observation of the true value. In this case the creature could react appropriately and not have to worry about updating the mental model of that object. It might also be used for truly discrete signals, such as modeling intervals between occurrences of discrete events (where a false prediction simply results in wrong timing, to be corrected as soon as the event finally does occur). However, in the case of location-prediction and modeling, some other technique will be needed in order to be able to incorporate negative knowledge of the type described here.
Figure 10-2: Location-Expectation Violation

The Kalman filter is able to make reasonable predictions of future state given a few observations (here, at times $t_0$ and $t_1$). However, when one of those predictions is invalidated (here at time $t_6$) the Kalman filter cannot generate reasonable "back-up" predictions.

Figure 10-3: Expectation Violation response

(a) Proper response to an invalid location distribution is to (b) zero the distribution in the visible region and scale the resulting distribution up to maintain integration to 1. (c) After a long time has passed, the distribution should be perfectly uniform within the occluded area.
Chapter 11

Probability Diffusion

In order to address the issues raised in the previous chapter, let us first go back a bit. In section 3.3.1 it was noted that confidence in object data should decay over time. Another way of interpreting "confidence" is as a probability. Thus when an object is observed, the probability that its location is within a certain place field (place-percept) is 100% (our confidence is 1.0). However the longer we do not observe that object, the less confident we are in that data, reflecting the decreasing likelihood that the object is still in the same location.

Given the contiguity of place-fields, however, we can do better than simply decaying the likelihood of the last observed place-percept. An object, from one place, cannot go just anywhere, but instead needs to go to an adjacent location.

In a continuous space, we might imagine the location probability of an object as kind of gas, which as long as the object is observed, is held in a tight, concentrated point. When the object leaves the field of view, however, that gas is released, and spreads (at rate we have yet to determine). The relatively low concentration of the gas at the outskirts of the spreading cloud reflect the unlikelihood that the object should have reached that location in such a short time. At the same time, the bulk of the gas remains at the last observed location, reflecting the likelihood that the object has not moved. As time passes, however, the gas becomes thinner and more uniform, eventually spreading to fill the available environment. At the limit, every location is equally likely.
The distribution of Brownian gases behaves in exactly this way. The concentration of the gas can in fact be expressed as a gaussian function whose mean is the location of the original source, and whose variance is proportional to both time since release and diffusion rate (\cite{61}).

Though we are not here dealing with a continuous space, the maps that we described in chapters 5 and 6 provide a good discretization of space. Assuming on a given timestep that an object is 100\% likely to be within a region of space represented by place-cell A (i.e. we observed it to be there), then on a subsequent timestep in which the object is not observed, it can be assumed that the probability diffused to adjacent place-fields, which, given the coherence of the maps formed by SOMs (chapter 5) and physicalSOMs (chapter 6), are represented by A’s topological neighbors.

The two conclusions, therefore, are:

- Location-probability can be treated as a diffusion process.
- That continuous diffusion can be approximated by diffusion through the discrete topology of the environment map.

11.1 Isotropic Diffusion

Discrete isotropic diffusion takes place on a regular lattice of nodes (see figure 11-1). Assuming each node holds an amount of probability \(p_{i,j}^t\), the update equation for a
hexagonal lattice is

\[ p_{i,j}^{t+1} = (1 - 6\alpha) \cdot p_{i,j}^{t} + \alpha \left( p_{i,j+1}^{t} + p_{i+1,j+1}^{t} + p_{i+1,j}^{t} + p_{i+1,j-1}^{t} + p_{i,j-1}^{t} + p_{i-1,j}^{t} \right) \]  (11.1)

assuming the topology shown in figure 11-1. \( \alpha \) is a diffusion rate that takes a value between 0 and \( \frac{1}{6} \). The intuition behind this equation is simple: every timestep, each node gives a fraction of its total probability (\( \alpha \)) to each of its neighbors, keeping the rest for itself. Assuming an initial total amount 1 of probability over all nodes, this diffusion equation conserves this total, a property that is useful in maintaining a valid distribution (though to avoid problems resulting from precision errors, the total probability can be periodically normalized).

### 11.2 Verification

The difficulty with the Kalman filter approach to location-prediction that I discussed in section 10.2 is its inability to incorporate negative information. In this case, the negative information is ”the space that can be seen to be empty”. Unlike the invalidation of a single location-prediction, the process should provide a whole continuous space of points where the object is not.

Put another way, we may consider the location distribution defined above as a continuous or discrete space of guesses. The incorporation of negative knowledge then becomes a process of guess-verification.

Fortunately, it is much easier to incorporate verification information using the discrete probability distribution described here than using the Kalman filter approach. The fundamental difference is that the location distribution is not defined by a single elegant mathematical expression that cannot incorporate inelegantly-defined negative information (”Visible space” can have no elegant definition, given that it is constrained not only by the camera frustum but also by occluding geometry). Instead, our distribution is made up of a collection of discrete buckets of probability. We can likewise discretize our negative knowledge in terms of our place-fields. The question
Figure 11-2: Distribution culling

(a) The creature might be searching for an object with the distribution shown in red. (b) Ideally, this distribution could be precisely culled by the space that is currently visible to the creature (shown in blue). (c) As a simplification the problem can be discretized in terms of place cells.

then becomes "is the place field visible?" If that question can be answered, then the probability-contents of those nodes that are visible can be zeroed, essentially carving visible regions of space out of the object’s location probability distribution. This naturally leaves only areas hidden by occluders and those outside of the creature’s field of view. See figure 11-2.

Of course, in order to maintain a proper distribution, the probabilities contained by all the other nodes are scaled up by a normalization factor of $\frac{1}{1-p_{culled}}$, where $p_{culled}$ is the summed probability contents of all the visible nodes (contents that were then zeroed).

That question can be answered going back to the creature’s basic perceptual abilities, most notably, its visual system.

11.2.1 Visibility Testing

The greatest source of negative information is the visual system. A Visual Verification System allows us to use the raw frame-buffer data as a source of both positive (an
object was observed at a certain location at a certain time) and negative knowledge (a region of space was observed to be empty) knowledge. See figure 11-3.

Note that a true determinant of place-field visibility would require a great amount of pixel sampling, which, due to run-time considerations, would not be feasible. It is found however, that rough estimates usually perform adequately.

The Test-Point Method

One such estimate is to detect whether a point at the world position of the node being tested and floating 5 or 10 units above the ground plane is visible. In order to test this, the world coordinates of the node are transformed into local (eye coordinates) and then (via the NDC transformation described in section 3.2) into retinal (NDC) coordinates. Once in retinal coordinates the integer pixel-coordinates can be derived from the screen resolution. The observed depth at that point can be extracted from the depth buffer, and compared with the depth component of the transformed node position. If the observed depth is shallower than the transformed one, the test-point was not visible, and the entire place-field should be considered occluded. If it is deeper, then the test-point is visible, and the place-field, correspondingly, should be considered visible.

Note that whatever method is used, it behooves us to choose a heavily occlusion-favoring method. In other words, it is better to mistake a visible location for occluded than vice versa. The price for a false occlusion determination is relatively small - in the worst case, the creature continues to believe an interesting object might be in a location it is not. The reaction will probably be to approach that location, and on approaching, gain a better view of it and eventually realize the mistake. On the other hand, a false visibility determination could result in the creature dismissing the object’s actual location as a possibility, and thus lose track of it entirely, possibly even spawning a new persistent belief in Working Memory when the object does again become visible.
Figure 11-3: The framebuffer as source of both positive and negative knowledge. From the framebuffer, object bounding-boxes and locations are extracted. These perceptions are pushed through the percept tree and eventually form Beliefs that maintain SpatialModels. The framebuffer also provides the Verification System with visibility information which that system uses to provide negative knowledge about where objects aren’t. Both types of information are combined in the Spatial System, where the location distributions are maintained.
11.3 Object Matching Using Location Distributions

The object location distributions can also be used in the object matching process in which incoming beliefs are matched against persistent ones. Recall from section 3.3 that incoming sense-perceptions must be compared against persistent beliefs in order to determine whether they represent the same object, or whether in fact they are new. In this case persistent percept attributes must return a distance between the histories or models that they contain and the incoming data. Ordinarily, because the incoming distance data is in the form of a vector, straight Euclidean distance would have been used. This will not always yield the correct match in cases where the environment configuration would prevent an object from being in a certain location.

Consider the problem illustrated in figure 11-4. In the environment there are two walls with a small gap between them. A red ball is observed to move behind the left wall. A few seconds later, a red ball is observed to move out from behind the right wall. A straight euclidean distance metric would happily match the two balls into the same persistent belief, since their locations are relatively close together. The correct answer, however, is of course that the two observations represent two entirely different balls: if the two were the same, the ball would have been observed through the gap to pass between the two walls. If using the probability diffusion system,
verification would have prevented probability from diffusing from behind one wall to
behind the other. The new observation of the red ball coming from behind the right
wall would have been placed in a place field which showed 0 chance of holding the
red ball known to be behind the left wall. Therefore this second observation would
have been considered to represent the appearance of an entirely new object.

    The new distance metric is given as

\[
d = \frac{1 - p_{b_{mu}}}{p_{b_{mu}}} \tag{11.2}
\]

where \( p_{b_{mu}} \) is the amount of probability contained (for the current object in ques-
tion) in the BMU of the observation, i.e. the place-field that contains the observation’s
location. This expression does indeed have the property that distance between an ob-
servation location and an object’s distribution is low if the observed location is very
likely to hold the object and high if it is unlikely to hold the object. However, it also
has the unfavorable property that it uses absolute probability for its calculation, such
that a location with low probability - even if it is the most likely location - will render
a large distance. This defect can be overcome using a normalization term, \( p_{b_{mu}} \), the
probability of the highest matching unit, the node with the highest probability for the
object. The metric becomes

\[
d = \frac{p_{b_{mu}} - p_{b_{mu}}}{p_{b_{mu}}} \tag{11.3}
\]

Thus a location renders a large distance only if it is low relative to the most likely
location. If an object turns out to be in the most likely location, the distance is zero.
On the other hand, if a distribution is extremely diffuse, with every node holding
a tiny bit of probability, the distance between any location and the distribution is
small, because any location could well actually hold the object.
Figure 11-5: Probability vectors

Each node is annotated with a vector of probabilities corresponding to each spatial model being tracked in Working Memory.

11.4 Summary of the Observation / Verification Loop

As noted in section 4.2, the $SpatialModel$ is used as the memory structure for world location information in Beliefs. Each node in the SOM or PhysicalSOM is annotated with a vector of probabilities, where the $i$th element of the vector corresponds to the probability that the space it represents contains the $i$th SpatialModel (see figure 11-5).

Each timestep:

1. Objects are matched

2. Visible place-fields are marked by the Verification System

3. Visible place-field probabilities are zeroed.

4. Each observed objects’ distribution is zeroed over the entire map, except for at its BMU where it is set to 1.

5. All distributions are diffused

Figure 11-6 summarizes the Observation / Verification Loop and its relation to the Environment Learning Loop of section 9.1. Note that in the diagram the process of Self-Localization depends on the probability culling step. This dependency has not been implemented, but it is believed that the absence of a landmark can be as
useful in self-localizing as its presence. If we turn a corner and expect to find a tree-
landmark, and the landmark is not there, then perhaps we are not at the corner we
thought we were.

Finally, it is important to note we that we have a new method for answering
location queries from the rest of the system. When Working Memory is queried for
the world location of an object on a timestep in which the object was not observed,
rather than returning the last known valid location (what we would have done before)
we will return the center of the place-field most likely to hold the object, i.e. the one
with the highest probability.
Figure 11-6: Summary of the Observation / Verification Loop
Chapter 12

Spatial Behavior

How does the object persistence system of the previous chapter impact the behavior of the creature? Even though the creature is still essentially a reactive decision-maker (it is doing no form of planning) it is capable of a whole new class of spatial behaviors, all of which have to do with location-expectation. The creature can now react to where it expects objects to be. Furthermore, unlike previous systems, the creature is able to react "intelligently" when an expectation is verified to be false by creating a new expectation based on visible space and the structure of the physical environment.

12.1 Emergent Behavior

Some interesting new behavior occurs without any explicit changes at all.

Salient Moving Objects

An object might be considered salient when it provides an unexpected stimulus. A creature’s attention system (as described in section 3.5) might correspondingly be instructed to direct the creature’s gaze toward salient objects. One possible unexpected stimulus might be an objects appearance in an unexpected location. This salience would have a clear metric under the system as described: it would simply be the "distance" computed in section 11.3. Thus an object which showed a great distance for the location matching but still matched would be salient. An example of
Figure 12-1: Salient object matching

The salience of a location is proportional to the ratio of the maximum value in the location-distribution to the value of the actually observed location. In (a), the object was seen in the last timestep, and so the distribution is fairly tightly centered around the last observed value. The ratio of $p_{\text{max}}$ to $p_{\text{observed}}$ is high and therefore so is the salience of the observation. In (b), the object has presumably not been observed in some time, and so the distribution is rather diffuse. The ratio of $p_{\text{max}}$ to $p_{\text{observed}}$ is correspondingly lower.

This would be an object that has just moved from one place-field to another (see figure 12-1). Assuming we observed the object at one timestep $t$ in location A, our natural prediction in the next timestep would be that the object has not moved (assume that the object has been stationary). Thus the location distribution of the object in the next time step is a fairly sharp gaussian centered on the last known location. When object is observed again, it is close enough to be matched with it previous observation - but it is matched in a place-field that it was not expected to be. Indeed, the faster the object moves (the farther in a single timestep) the less likely it will have been for the object to be there based on the previous timestep’s observation, and the more salient it is. Thus, by simply adding the expedient that the creature should look at salient things (a natural and general strategy) the creature begins to pay more attention to moving object.
Emergent Look-Around

The act of looking around - sweeping the eyes from one side to another of a scene - is hard to justify without a visual system, given that the act brings no change in information, though it can certainly be hand-coded in as an animation. Even with a visual system, the look-around behavior is of limited usefulness if the system cannot incorporate the negative knowledge that is gained from the empty space that are observed.

Interestingly, however, the look-around emerges by itself from the probability diffusion system described in the previous chapter. The explanation is fairly straightforward. Assume an open plane with no occluding obstacles. If the creature decides or is instructed to look at an object, the creature directs its eyes toward the most likely location for that object. If it is in the expected location then the creature has succeeded. If the object is not there, then the distributions in the visible place fields are zeroed, and renormalized in the rest of the field. This causes locations previously thought to be unlikely to hold the target object to be scaled up in likelihood. Assuming an open field, the next most likely location will probably be the one nearest to the previous estimate outside of the field of view. This location therefore becomes the next target for the creature’s gaze. If this location also does not hold the object, the next one is tried and so on. The result is that the creature’s eyes sweep through the scene until the object is found. Figure 12-2 shows traces of a look-around performed emergently by Duncan as he was looking for a sheep. Note that it is not always a clean uni-directional sweep, but instead jumps back and forth between extremes as first one side and then the other becomes the most likely place to hold the object.

Emergent Search

When the creature decides or is instructed to approach an object, the emergent look-around described above becomes an emergent search. This is an especially interesting task when there are occluders in the world, because the creature must then physically move to look behind them. The same rule then applies as it did above: a location is
Figure 12-2: Emergent look-around

(a) At first Duncan can see both shepherd and sheep. (b) When the sheep leaves his field of view, Duncan assumes the sheep to be somewhere near where it was last seen. (c) and (d) When Duncan looks for the sheep again, he looks at the current most likely location. (e) He finally finds the sheep. Areas of red in the right column indicate the probability that the sheep is located in each place field. Blue nodes are those considered visible.
investigated, and if it does not hold the intended target, a new estimate is generated.

Figure 12-3 shows the trace of one interaction with Duncan the sheepdog. In a two-wall environment, Duncan is shown a sheep and then called away by the shepherd. While Duncan is looking away, the sheep is moved behind the nearest wall. When told to approach, Duncan looks around a bit, and then concludes that the sheep must have moved behind the wall. He goes to look for it there and succeeds in finding it. He is called away again. This time the sheep is moved behind the other wall. When Duncan is again instructed to seek out the sheep he returns to the original hiding place only to find the sheep gone. He continues to search. Behind the other wall is actually one of the first places he looks, since he already got a good look at the open central area when he was making his way over to the first wall. Thus by simply being instructed to approach a target, Duncan autonomously searches his target out, intelligently updating his “best guess” of the object’s location as he goes based on negative visual feedback.

Emergent search is a good example of how an intelligent low-level mechanism can make the job of the high-level decision-making processes easier. As noted in section 3.6, there are any number of reasons why Duncan should approach a sheep - it could be in order to perform any number of actions towards it (”beg” toward it, ”bite” it, ”bark” at it, etc.). These actions could be strategies for satisfying any number of drives - in other words the actions are all completely unrelated. In a traditional hierarchical action selection mechanism, ”approach” and ”seek” would need to be included explicitly as sub-strategies for performing the action in question. In C4 however, that functionality is spread to the Spatial and Navigation Systems, so that navigation and search capabilities are implied in all targeted behaviors. This also makes any learning much easier at the Action-Selection level, since the learning mechanism does not need to figure out why 50% of the time performing a search before performing the action helps and 50% of the time it doesn’t.
Figure 12-3: Emergent search
Correct Object (Mis-)Matching

The example of the two walls with a gap between them has already been mentioned in section 11.3, but it bears re-iterating here. Again the example states that if a ball is observed to roll behind the left wall, and a few seconds later an identical ball rolls out from behind the right wall, though they are close together and visually identical, the conclusion must be that the second sighting is of an entirely new object, because if the ball had moved to behind the right wall, it would have been observed to pass through the gap (see figure 11-4). Using the probability diffusion system described in the previous chapter, the correct determination is indeed made, since constant viewing of the areas of space between the two walls effectively keeps zeroed any probability content in the corresponding place fields. Since the original ball’s probability field cannot diffuse through the gap, the probability of the ball appearing from behind the right side of the right wall is 0, infinitely unlikely. The new sighting will therefore not be matched into the old belief and a new belief will instead be created.

This is a significant result for a couple of reasons, the first of which is that it shows a very different kind of ”reasoning” going on compared to the previous examples. It shows, in fact, a kind of negative reasoning, reflecting the negative conclusion that the second sighting is not the same ball as before. Note also that the ”decision” that the second sighting represents a second ball is a highly distributed one made by the confluent effect of a number of brain systems. The Spatial System diffuses probabilities, the Verification System keeps visible place fields zeroed. The belief-matching metric is provided by the SpatialModel representing the first ball and the decision to spawn a new node is made by Working Memory itself. This is an exciting instance of many individually unintelligent processes producing a very intelligent and complex decision.

12.2 Emotional Behavior

Section 2.4 discussed how expressivity is an important component in designing synthetic creatures. Interestingly, a number of the processes described in the previous
chapter have emotional interpretations. The names given to them here may not be
perfect mappings, but they do get the ideas across (I attribute the imperfection of
the mapping to an ambiguity of the language rather than a technical shortcoming).

12.2.1 Surprise

The emotion of surprise was already hinted at in section 12.1, only there it was called
"salience". A general metric of salience is, given a distribution of expected values,
how unlikely the actually-observed value was. In other words

\[
salience = \frac{(1 - p_{\text{observed}})}{p_{\text{observed}}} \tag{12.1}
\]

where \( p_{\text{observed}} \) is the probability of the observed location. This can optionally be
normalized by the probability of the most likely value, \( p_{\text{max}} \)

\[
salience = \frac{(p_{\text{max}} - p_{\text{observed}})}{p_{\text{observed}}} \tag{12.2}
\]

This is, of course, the distance metric of section 11.3. In the case of location-
expectation, this can also be interpreted as surprise at seeing an object in a location
it was not expected. An example might be Duncan turning around and finding the
sheep directly behind him when we expected it to be across the field behind the wall,
or your mother-in-law appearing on your doorstep for a surprise weekend visit when
you thought she was in Cincinnati.

The last example underlines how surprise - here defined as a deviation from the
expected - can also be modulated according to the attitudes held towards the surprising
object. If Duncan was looking forward to a little lamb for dinner, then finding
it closer than he expected is a pleasant surprise. To find it farther would be a dis-
appointment. To find your step-mother closer than you thought she would be might be a fearful surprise whereas to find her farther would be relief. This modulation
might automatically be inferred based on whether the navigation-instruction implied
an approach or a flee.
Surprise Dynamics

Surprise could be a quality attributed to varying degrees to individual objects, or individual surprises could simply modify the global emotional state. In its current implementation in C4, only surprises by the current object of attention register and impact the global state. The dynamics of surprise are

\[ s^{t+1} = 0.9s^t + 10\frac{p^{t+1}_{\text{max}} - p^{t+1}_{\text{observed}}}{p^{t+1}_{\text{observed}}} \]  

(12.3)

where \( p^t_{\text{max}} \) and \( p^t_{\text{observed}} \) are defined as they are above at time \( t \).

12.2.2 Confusion

A similar but inverted expectation violation occurs when an object is observed not to be in the location it was expected. I call this confusion. In this case, the amount of confusion is directly proportional to the amount of probability that was culled in the verification process. Again confusion can be modulated by attitude towards the object. You would certainly be confused not to find the sheep where you expected it, but you might be either worried or relieved not to find your step-mother where you expected her to be.

Confusion Dynamics

Confusion is updated as follows

\[ c^{t+1} = 0.9c^t + 10p_{\text{culled}} \]  

(12.4)

where \( p_{\text{culled}} \) is the amount of probability corresponding to the object of attention culled out by the verification process. Confusion is only non-zero when the object of attention was not observed. When it is observed the confusion is zeroed.
12.2.3 Frustration

Meanwhile, long-term failure to see the object of attention might result in frustration. In this case, the term seems to apply no matter what the attitude held toward the object in question, since it is probably never good to not be able to see your object of attention.

Frustration Dynamics

Frustration grows linearly over time for as long as the object of attention is out of sight, and resets to 0 when it is seen again.

\[ f^{t+1} = f^t + 0.1 \]  

(12.5)

All three of these emotions can be brought to bear on the emergent search task. When Duncan turns around and finds that he can’t find the sheep, he is initially confused. This confusion, as he continues to fail to see it, gradually gives way to frustration. If Duncan should happen to catch a view of the sheep in an unexpected location, he would be surprised. Note however, that if the routine is continued to the sheep’s discovery behind the second wall, Duncan will not be surprised when he finally finds it. This is because by the time he nears the second wall, behind that wall is the most likely location for the sheep (because all other locations have been exhausted). Since he finds the sheep exactly where (at that point) he expected it, Duncan is not surprised.

12.2.4 Emotional Attention and Action Selection

With practical, grounded definitions for surprise, confusion and frustration, it is interesting to consider how these emotional level might be channeled back into the basic Action-Selection mechanisms. These ideas have not as yet been implemented in C4.

- Surprise, it has already been suggested, can be used to direct attention, such that an object that provides a surprising stimulus is a good candidate for the
object of attention. If it is even more surprising, it might interrupt the current Primary Action Group action. This could be achieved with a simple ”Stop for a minute and observe the object of attention” ActionTuple placed in the Primary Action Group.

- Frustration reflects the general difficulty with which the current action is proceeding. This might naturally be fed back into the action system as a ”stop the current action, whatever it is” signal.

- Surprise and Confusion together constitute a general measure of how much things are going according to expectation. This might be used to tune the exploration vs. exploitation decisions that the Action-Selection mechanism is making. The optimal state, it might be presumed, is ”a little bit confused” and ”a little bit surprised”. If surprise and confusion levels are too low, then the creature can probably afford to do a little exploration. If they are too high, then the creature should exploit some more. An exploration behavior might prompt Duncan to search for a hidden but potentially larger food source, whereas exploitation might tell Duncan to go back to his reliable small one.

12.2.5 Emotional Animation

As noted in section 3.6, emotional state can also affect animation quality. If Duncan had emotional walk cycles (he does not) he could made to walk in a confused, surprised or frustrated way. Or, as was actually implemented, Duncan’s facial animation layer could be modified to reflect his emotional state. Duncan was given an \texttt{EyebrowMotorAction} motor skill that read the three emotional states out of working memory and blended the eyebrow positions between three position depending on the relative weights of each of the emotions. See figure 12-4 for Duncan, the Dog of a Thousand Faces.

Lastly, extreme emotional states can of course trigger full-body undirected actions. The creature could be made to jump in the air when surprised, or throw up its paws/hands in disgust when too frustrated.
Figure 12-4: Duncan emoting

Duncan (a) confused (b) frustrated (c) surprised and (d) neutral.
Chapter 13

Conclusions, Contributions and Future and Related Work

The Virtual Hippocampus project began with the desire to thoroughly explore the psychology of space. It seemed clear to me that the way space was being perceived by our synthetic creatures was neither biologically plausible nor behaviorally useful. A 3-vector representing an object’s location in a cartesian world space does not lend itself to behavioral complexity. For example it does not capture the overall structure of the space in question. A point $<3.2,4.5,1.1>$ is still the same point whether the space is full of objects and structure or it is a yawning, gaping void. If everything in the world shifts suddenly to the left by three meters, do the ”locations” remain behind? No, because location in our minds is not defined by a Cartesian plane but by its relationship to the physical structure of the world. The main problem with the 3-vector approach is that it does not define location as an experiential phenomenon. Is being at $<3.2,4.5,1.1>$ substantially different from being at $<3.2,4.5,1.2>$? Euclid might say that they are 0.1 similar, but I argue that they represent the same place psychologically, because the experience is, to the best of the creature’s perceptual ability, identical.

The system I have built treats space in this way. In other words, it is designed for ”psychological utility” rather than ”geometric convenience”.

”Common sense” is a term that is generally hard to substantiate, and I hope I
have not misused it in the title of this thesis. The numerous emergent behaviors described in chapter 12 lead me to think that I have not. There are certainly aspects too numerous to mention of spatial common sense that have yet to be tackled. I hope, however, that the approach taken in this thesis will be seen as a good first step.

Contributions

If a bulleted list of contributions that this thesis offers were demanded of me, this is the list I would produce:

- Explored a psychological definition of location based on local view.

- Showed an integrated system of environment-mapping, self-localization and landmark-learning.

- Showed an application of the space-division in the form of the Object-Persistence system.

- Designed an object persistence system that formalizes the concept as location-expectation, and showed how both positive and negative knowledge can be meaningfully incorporated into a model of object-location.

- Demonstrated a full creature operating with all the above systems displaying more sophisticated spatial behavior than has previously been seen in synthetic creatures.

Limitations and Future Work

It is a platitude that "the more you know, the more you know you don’t know.” But sometimes it is particularly true.

While the local view distribution learning of chapters 5 and 6 was reasonably useful, further experimentation with that process is in order. In particular, both SOMs and PhysicalSOMs impose a pre-set topology on the modeled space, in this case hexagonal lattice. It is possible, however, that the topology of the space itself
could be successfully learned. Part of the power of the pre-set topology is that it models successfully regions of space that the creature has never visited (including, for example, space *inside* other objects. This arguably contributes to the system’s predictive abilities, but it is possible that a topology-varying method, such as Neural Gas (for example, [19]) or other SOM-variants, would better avoid modeling space that is not useful.

The current work also takes a rather simplistic definition of ”landmark”. As currently implemented, a landmark is simply the location of a stationary (or orientation-stationary object). It is clear, however, that landmarks can take many forms, including specific object-features (a *part* of a wall), landscape shape, terrain type, coloration features, etc. In particular, though most landmarks are visual, the system needs to be able to incorporate non-visual spatial cues (”I hear running water, so I must be near the stream”). Right now self-localization occurs strictly on the basis of the best matching unit. It is possible, however that a more probabilistic approach to self-localization – representing probability-of-location in much the same way that object-localization does in the probability-diffusion system – would lend itself more readily to making use of non-spatial cues.

Representing spatial extent is difficult, despite the fact that the SOM representation appears well-suited for it. The crux of the difficulty is the object-matching process, which is simple when we are matching a single point (which is rotation-invariant) to two-dimensional distribution but fiendishly complex when we are matching an arbitrary two-dimensional shape to a three-dimensional distribution (assuming orientation as one of those dimensions). It would also be desirable to be able to match objects on the basis of *partial* observations. The limitations of the perceptual system also make this difficult (see below).

The probability-diffusion system was by-and-large successful, but a number of obvious extensions are in the works. In particular a system for physics-based diffusion is implemented but not integrated. This system gives momentum to the probability-bubbles as they diffuse about the map. If an object has been observed to be traveling in a certain direction, then when the creature turns away, the probability-bubble will
continue to diffuse preferentially in that direction at a rate that is consistent with the object’s last observed speed. Of course the bubble will become more diffuse, and will eventually slow to a stop, becoming, again, a uniform distribution. Designing an algorithm that behaves this way involves the application of a pared-down form of the gas equations and the mathematics become complicated enough that system stability becomes an issue.

One fun and interesting extension to the mapping/object persistence system would be to incorporate the third dimension. Thus a 3-dimensional map of an environment could be built (presumably based on 3-dimensional local view) and used to keep track of object positions. Perhaps this could lead to objects being sought not just "behind the wall" but also "under the bed".

Scalability remains a major concern. SOMs’ inability to scale has already been noted several times, and though PhysicalSOMs allow for growth to take place, they still require, for example, all nodes in the map to be treated every timestep. A way needs to be found to break space into separate areas of representation (akin to McNaughton’s "reference frames" from [36]) so that only one such frame need be updated at a time.

Perhaps the largest deficit in the current work is the lack of support for hierarchical representations of space. Meter-squared resolution in the space representation of your house may be useful when you are in your house, but when you are at work, you think of your house in very different terms. Spatial information at this level is most likely symbolic. However it can still be used for spatial tasks, such as planning routes. It is not obvious how such a symbolic spatial system would interface with the low-level space-modeling of the Virtual Hippocampus.

Finally, though a detail, the vision model used for this work was extremely simple — essentially bounding-box and location-extraction — and made some spatial perception hard, especially object-edge detection. The root of this trouble was that the vision was not taken particularly seriously. A more robust design that was better-integrated with the rest of the system (a system that somehow leveraged expectations about visual scenes, for example) would make the entire process considerably less painful.
Particularly egregious is the fact that all objects are modeled essentially as flat surfaces, which does well for walls, acceptably for other characters (since they are not generally too large, and never occupy more than one location at a time) and terribly for large irregularly-shaped object, of which, I am told, there are a considerable number in the real world. Two possible general approaches to this problem might be either to build a 3-dimensional model of the object surface or to build an image-based model of object appearance from multiple views (a biologically more plausible approach).

Related Works

A work like this one draws from many different fields. Among the most important sources are the following:

Neuroscience

None of this thesis would have happened, of course, without the neuroscientists who discovered the place-cell phenomenon, and who, as a field, have been studying it for decades. We are all especially indebted to O’Keefe and Nadel, whose *The Hippocampus as a Cognitive Map* ([41]) threw down the gauntlet for the Hippocampus as an environment learning and mapping system. I found Redish’s *Beyond the Cognitive Map* extremely useful both for its excellent overview of the place-cell phenomenon and as an introduction to the extensive Hippocampus literature. Also notable among the throng is the work of McNaughton, especially *Deciphering the Hippocampus Polyglot* ([36]) whose few pages contain enough good ideas to fuel half a thesis. Gallistel’s work ([21]) is also very important for his argument that animals form *metric* maps of their environment (as opposed to some sort of associative memory mechanism).

The SAB Proceedings (Animals to Animats) over the years have contained some great works that lie on the blurry edge between neuroscience and artificial life. Prescott et al. ([45]) demonstrate a system for long-range navigation that ties together clusters of landmarks and then plans paths on these higher-level representations. Such a system is direly needed for the current work. Also appreciated is
whose local reference direction trick allowed my self-localization algorithm to work.

**Artificial Intelligence and Artificial Life**

The C4 architecture is very much an evolution on many great systems that have come before in the field of Artificial Life. Most relevant, of course, is the work of Blumberg ([6], [7] and [8]) who, besides being my advisor, pushed the adoption of ethological models of motivation, perception and action-selection in synthetic creatures.

Terzopulos has also done much impressive work over the years, in particular the *Biomechanical Fishes* with Tu ([60]) and more recently the *Cognitive Modeling* with Funge ([20]).

Besides being one of the earliest examples of behavior-based animations, Reynolds’ *BOIDS* algorithm ([49]) was implemented in C4 for the *Sheep—Dog: Trial By Eire* project, and remains to this day an important lesson in minimalist representation and action-specification.

There has been some work done on the theory of expectations. Kline’s work ([27]) has already been cited extensively. [43] is an early work which deals with the subject (and that makes an interesting distinction between ”surprise” and ”expectation violation”, a distinction that is not respected in this work). Work on expectation generation and violation continues also within the Synthetic Characters group, not only with this work, but also with Burke’s masters thesis ([11]) on time and rate modeling (an implementation and extension of some of the theories of Gallistel from [22]).

Of course all of us everywhere owe everything to grandfather Minsky and his *Society of Mind*, of which all of our works are, ultimately, merely implementations.

**Animation**

*The Intentional Stance* ([14]) and *The Illusion of Life* ([57]) are often cited in our work as twin inspirations for our emphasis on expressive autonomous creatures - Thomas and Johnston tell us how to animate, Dennet tells us *why* to animate. The work of Perlin (e.g. [44]) has dominated the field of behavioral animation in recent years,
as the paradigms he introduced with the Improv system have fast become standard tools in the toolbox of self-respecting motor systems everywhere. Another recent and important influence has been Rose’s work on Verbs and Adverbs ([51]), which allows parameterized motion blending for the purposes of expressive animation (soon, no doubt, to become a standard tool in the toolbox of every self-respecting motor system). Further work has come from within our group itself, specifically with Marc Downie’s expansive masters thesis on Posegraphs ([16]) an extension of the Verb-graphs of [51] that allows automatic animation splicing and transition-generation as well as motor learning.

Robotics

A tremendous amount of inspiration has come from the roboticists, who, unlike the rest of us, actually have a problem to solve. Maja Mataric has done much excellent work in the area of mapping and mobile robot navigation (e.g., [33]), including one paper that makes explicit reference to the hippocampus ([34]). Also impressive is the vast corpus of Thrun whose probabilistic approach to just about everything was influential in the development of the object persistence system of chapter 11 (and, having read [58], would also be sure to have an influence in any revisions of the self-localization algorithm of chapter 7). The probability diffusion system bears a strong resemblance to the occupancy grids (alternatively evidence grids or certainty grids) found in the work of Moravec (e.g. [38]). The intention has also been to implement a system similar to the Semantic Hierarchy of Kuipers ([30]), which provides a way to build an abstract symbolic or topological map on top of a low-level space-modeling representation.

And finally, I owe much philosophically to the work of Brooks (e.g. [10]). C4 has a clear subsumption-like aspect to it and our emphasis on the importance of whole-creature building (as opposed to the toy-worlds of much traditional AI research) can be traced directly to his work. I don’t know whether he would necessarily approve of the whole ”virtual embodiment” trick we try to pull, but I have my fingers crossed.
Appendix A

Hyperplane/RBF Interpolation

Hyperplane/RBF Interpolation is a convenient method for building function-approximators based on a number of examples. These examples can map arbitrary-dimensioned vector inputs to arbitrary-dimensioned vector-outputs. The resulting interpolator function can be used for both interpolation and extrapolation (i.e. arguments need not be bounded by the examples supplied) and return better results in areas near the examples.

As the name suggests, there are two steps to constructing and sampling the interpolator. First, a hyperplane is fitted to the examples, and then a group of radial basis functions are fitted to account for deviations from the hyperplane for the example supplied.

All of the following explanation assumes a single output dimension. In order to achieve multiple output dimensions the same methods are repeated for each dimension.

Hyperplane fitting

The hyperplane fitting is a standard Least Squared Error technique. A hyperplane is defined by a vector $x$ such that
\[
\begin{pmatrix}
    v_1^T & 1 \\
v_2^T & 1 \\
    \vdots & \vdots \\
v_n^T & 1
\end{pmatrix}
x = b
\tag{A.1}
\]

\[
Ax = b
\tag{A.2}
\]

The vectors \(v_1...v_n\) are the \(n\) example points and \(b\) is the vector of output values (provided along with the examples). The above equation has a solution \(x\) if all the examples lie along a hyperplane. If they do not, a least-squares fit renders

\[
\begin{align*}
    E &= (Ax - b)^2 \\
    \frac{\partial E}{\partial x} &= 2A^T(Ax - b) = 0 \\
    \Rightarrow A^TAX - A^Tb &= 0 \\
    \Rightarrow x &= (A^TA)^{-1}A^Tb
\end{align*}
\tag{A.3}
\]

The residuals, the deviation of each example from the hyperplane, are then easily calculated as

\[
r_i = \begin{pmatrix} v_i^T & 1 \end{pmatrix} x - b_i
\tag{A.7}
\]

**RBF-Fitting**

In order to account for the residuals calculated above, a radial basis function is placed in the location of each example. Again, a least-squares approach is taken:

\[
\begin{pmatrix}
    g_1(|v_1 - v_1|) & g_2(|v_2 - v_1|) & \cdots & g_n(|v_n - v_1|) \\
    \vdots & \vdots & & \vdots \\
    g_1(|v_1 - v_n|) & g_2(|v_2 - v_n|) & \cdots & g_n(|v_n - v_n|)
\end{pmatrix} w = \begin{pmatrix} r_1 \\
    \vdots \\
    r_n \end{pmatrix}
\tag{A.8}
\]

\[
Bw = r
\tag{A.9}
\]
where \( w \) is a vector of rbf weights which we want to find, and \( g(radius) \) is the radial basis function. Solving

\[
E = (Bw - r)^2 \\
\frac{\partial E}{\partial w} = 2B^T(Bw - r) = 0 \\
\Rightarrow w = (B^T B)^{-1} B^T r
\]

(A.10)  (A.11)  (A.12)

**Sampling**

To sample the resulting function at a new point \( v_{new} \) we use

\[
f(v_{new}) = \left( \begin{array}{c} v_{new}^T \\ 1 \end{array} \right) x + \sum_{i=0}^{N} w_i g_i(|v_i - v_{new}|)
\]

(A.13)

The RBF used can take many forms, although the Gaussian basis is most often used. It has been suggested that an appropriate width for the Gaussian (variance) is \( \frac{2}{3} \) the average distance between examples.

**The Jacobian**

We will again treat each output dimension separately, in that each output dimension will get a complete gradient in the input space. The result will therefore be a Jacobian matrix in which each row holds the input-space gradient for a separate output-space dimension (i.e. in which direction in the input space should we move to achieve the quickest increase in the value of a particular output dimension?). The following discussion will assume (without loss of generality) a single output-space dimension and therefore a single gradient to be found.

Since the interpolator function is a linear sum of \( N + 1 \) components, (where \( N \) is the number of input examples), we can take the same approach to finding the gradient as we did to sampling, i.e. find the gradient for each component separately and then sum them up.

The gradient of the hyperplane is simply
\[ \nabla \left( \begin{pmatrix} v^T & 1 \end{pmatrix} x \right) = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \] (A.14)

where \( n \) is the input-space dimensionality. Note that this result is not equal to \( x \), since \( x \) is of dimensionality \( n + 1 \).

The gradient of the RBF is slightly trickier, because the RBF is a one-dimensional function of radius. However, assuming that this function is monotonic and decreasing, we can note that the magnitude of the gradient will be equal to the one-dimensional derivative at the radius in question, and the direction will be toward the rbf center. Thus

\[ \nabla (w_i g_i(|v_i - v|)) = w_i \hat{g}_i(|v_i - v|) \frac{(v_i - v)}{|v_i - v|} \] (A.15)

Thus the gradient for a single output dimension is given as

\[ \nabla f(v) = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \sum_{i=0}^{N} w_i \hat{g}_i(|v_i - v|) \frac{(v_i - v)}{|v_i - v|} \] (A.16)
Bibliography


