SIMULATION AND VISUALIZATION OF ENVIRONMENTS WITH MULTIDIMENSIONAL TIME

by

Luther A. Tychonievich

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science
Brigham Young University
April 2008

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Luther A. Tychonievich

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Robert P. Burton, Chair
Date	Michael A. Goodrich
Date	Dennis Ng

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Luther A. Tychonievich in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Robert P. Burton Chair, Graduate Committee
Accepted for the Department	Parris K. Egbert Graduate Coordinator
Accepted for the College	Thomas W. Sederberg Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

SIMULATION AND VISUALIZATION OF ENVIRONMENTS
WITH MULTIDIMENSIONAL TIME

Luther A. Tychonievich
Department of Computer Science
Master of Science

This work introduces the notion of computational hypertime, or the simulation and visualization of hypothetical environments possessing multidimensional time. An overview of hypertime is provided, including an intuitive visualization paradigm and a discussion of the failure of common simulation techniques when extended to include multidimensional time. A condition for differential equations describing hypertime motion to be amenable to standard time-iterative simulation techniques is provided, but is not satisfied by any known model of physics. An alternate simulation algorithm involving iterative refinement of entire equations of motion is presented, with an example implementation to solve elastic collisions in hypertime. An artificial intelligence algorithm for navigating crowds in any arbitrary nD/mT environment is discussed, and an implementation is provided using collision cones and stochastic global optimization techniques. Possible models of hypertime energy and other open questions are discussed. Both algorithms are described and show favorable results, meeting all design criteria and running at interactive speeds on common desktop computer systems.

Contents

Front Matter **i**

Title Page i

Copyright ii

Signatures iii

Abstract v

Table of Contents vi

1 Introduction **1**

1.1 Background 1

1.2 Prior Art 2

1.2.1 2T String Theory 3

1.2.2 Failure of 2T Fields 4

1.3 Overview of This Paper 5

1.4 Notational Conventions 5

2 The Nature of Hypertime **6**

2.1 Intuitive Picture of Hypertime 6

2.1.1 Time Arrows 7

2.1.2 Dimensionality of "Now" 10

2.2 Consistent Hypertime Simulations 11

2.2.1 Consistency 12

2.2.2 Temporality 13

2.2.3 Additional Considerations 14

3 Hypertime Physics **16**

3.1 Ultra-Hyperbolic Fields 16

3.1.1 Point-Iterative Hypertime 17

3.2 Elastic Collisions 19

3.2.1 Distance-based Formulation 19

3.2.2 Equation Iteration 23

3.2.3 Collision Location 28

3.2.4 Testing 32

4 Hypertime Navigation **33**

4.1 Curving Time 33

4.2 *n*D Navigation 35

4.2.1 Maneuvering Boards (and related work) 35

4.2.2 Computational Tractability 37

4.2.3 Other AI Tasks 39

4.3 Cylindrical *m*T Navigation 39

4.3.1 Chain-mesh, Fatter When Longer 40

4.3.2 Chains and Radii on Maneuvering Boards 42

4.3.3 Implementation and Testing 43

5 Exploring Hypertime **46**

5.1 The Interactive Interface 47

5.2 Observing 2T Physics 48

5.3 Observing 2T AI 50

6 Conclusion **52**

7 Future Work **54**

References **58**

A Comments on Hypertime Energy **59**

B Definition of Terms **61**

Chapter 1

Introduction

1.1 Background

The Hyperdimensional Research Group at BYU and other related efforts have, for many years, designed techniques and produced algorithms for understanding and visualizing higher-dimensional spaces[1]. The need to visualize more than three simultaneous dimensions of data arises in many situations. One particular application class of hyperspace research treats time as a fourth spatial dimension in a 3D animation. Recent work with this time-as-the-fourth-dimension paradigm in order to bound moving objects in animation [3] demonstrates that time has certain characteristics fundamentally different from space. While exploring the nature of these distinctions, hypertime research was born.

Conceptually, the goal of hypertime research is to simulate and visualize environments where time is a multidimensional vector space. This restricts the laws of such a universe to permit only vector operations (e.g., no multiplication, division or strict ordering) and helps to unravel how time behaves differently from space. Initially, however, it was far from obvious that this conceptual problem was even

[1] Though too numerous to provide a complete list here, a small sample of these techniques can be found in [20, 1, 16, 8, 15, 11, 7, 12, 22].

approachable.

In the only known relevant publication approaching the subject of hypertime [29], astronomer Max Tegmark demonstrates that a simple extension of basic physical laws to include vector times yields partial differential equations with no solution. While this is not a conclusive proof that hypertime physics is impossible, it is certainly not encouraging.

My first task in opening this branch of research, then, is to define and implement models of hypertime that satisfy our intuitive understanding of "time," are internally consistent, and form an extensible basis for further hypertime research. This task leads to a discussion of hypertime mechanics, physical laws and mathematics, as well as efficient simulation techniques for these hypertime laws.

My other primary goal is to answer questions which deal with interactions in hypertime: what does hypertime look like, how can decisions be made within a hypertime setting, etc. Work on this goal led to the development of artificial intelligence and computer visualization techniques that work within the hypertime setting.

1.2 Prior Art

I consider computational hypertime a new field, with little history and virtually no directly applicable prior research. However, as no field exists in a void, I acknowledge several related publications in the field of theoretic physics over the past ten years. In particular, Itzakh Bars has published a sequence of papers on 2T string theory and Max Tegmark has published a small review which includes a discussion of why 2T Newtonian and relativistic physics are ill-posed problems.

1.2.1 2T String Theory

Any research dealing with many dimensions bears at least a superficial resemblance to string theory. String theory is a model of subatomic physics that relies on high-dimension mathematics, using more than twice as many dimensions as the typical 3D/1T model of the universe. This resemblance is generally only superficial because these additional dimensions wrap on themselves on the Planck scale; that is, anything traveling a straight line along any of these extra dimensions returns to where it started before it can go the diameter of a single proton. This microscopic looping of space greatly reduces the relation between string theory and traditional hyperspace research, which assumes the additional dimensions to be at least locally flat.

Most models of string theory still have exactly one time dimension, but Itzakh Bars and collaborators have produced a number of documents (including [5, 4, 6]) discussing a model of string theory that has two dimensions of time. Although the additional time dimension is not extremely short as are the additional string theory space dimensions, it is also not meaningfully hypertemporal. Bar makes it clear that his theory contains a special "gauge symmetry" which removes all hypertemporal phenomena. The additional time dimension appears to serve as a mathematical device to store certain information in a compact and elegant way, not dissimilar to homogenous coordinates in computer graphics [26]. Although this may simplify mathematics for expressing string-theoretic single-time phenomena, it does not contribute to an understanding of more general hypertime environments.

1.2.2 Failure of 2T Fields

Max Tegmark published the closest thing to a hypertime paper I have been able to discover [29]. In it he addresses questions about the probability of a universe with any dimensionality other than 3D/1T containing sentient observers. He concludes that such universes cannot exist if their physical laws are simplistic extensions of those we experience. In particular, he shows that given field-based physics (where the induced acceleration is a linear function of the separation between two particles) the basic equations of motion yield ultra-hyperbolic partial differential equations (PDEs). This particular class of PDEs does not permit well-posed predictive questions. In particular, no bounded set of observations $(\vec{x}, \vec{t})$ are sufficient to make valid assertions regarding any unobserved $(\vec{x}, \vec{t})$.

Tegmark is careful to state that his work should not be considered to disprove hypertime. However, his results appear to apply to a category of phenomena much broader than the field-based physics he discusses. After reading the paper I spent over a year working on a wide variety of increasingly bizarre alternatives—including field-based velocity, path-curvature, and jerk-forces; geodesics over various curved spaces (including general relativity and other curvatures); and various complicated static field effects—all without success[2]. Eventually, however, I was able to create a consistent hypertime extension, as discussed in Section 3 on pp. 16–32, by using a simulation paradigm fundamentally different from the standard methods for numerical PDE evaluation.

The most valuable insight provided by Tegmark to the field of hypertime is an understanding of what cannot work. It appears from his work that some of the questions we naturally ask about the universe depend upon the scalar nature

[2] Although all of these laws were designed theoretically, they were tested in simulation, rather than by proof. The simulated result of the ill-posedness of ultra-hyperbolic PDEs are characterized by phenomena where, given some $\vec{x}(\vec{t_1})$ the simulation can derive two states $\vec{x}(\vec{t_2})$ and $\vec{x}(\vec{t_3})$, but when the simulation tries to derive $\vec{x}(\vec{t_3})$ from $\vec{x}(\vec{t_2})$ the two derived $\vec{x}(\vec{t_3})$s are not the same. It was this simulated inconsistency that was taken to indicate the lack of success of each model.

of time. Fortunately, there are questions we can pose meaningfully in hypertime; several of these questions are discussed in the remainder of this paper.

1.3 Overview of This Paper

Due to the novelty of the subject matter and the scope of my research, I begin this paper in the following chapter with a philosophical overview of hypertime itself. Along with discussing this philosophy in general terms, I reduce certain core philosophical assertions to mathematical criterion. Following that high-level discussion are two chapters detailing two concrete hypertime simulations satisfying these criterion. The remaining chapters provide some observations from running these simulations and suggest directions for future work.

1.4 Notational Conventions

Throughout this paper, I use the following notational conventions:

- Scalars and indices are represented in lower-case Greek or Roman characters: a, α, i, etc.
- Vectors and vector-like objects in curved spaces are represented with $\vec{}$hats.
- Matrices are represented with bold-face capitals: $\mathbf{A}$, $\mathbf{V}$, etc.
- Sets are represented with block letter capitals: $\mathbb{O}$, $\mathbb{V}$, etc.
- Other mathematical entities are represented in calligraphy: $\mathcal{P}$, $\mathcal{S}$, etc.
- Structures members are accessed by typed subscripts; thus, $p_{i\,\mathbf{A}}$ is the $\mathbf{A}$ matrix in the structure p_i. Structures themselves are lowercase, like scalars.

Some additional notation, operator definitions and mathematical tools used in this paper are provided in Appendix A.

Chapter 2

The Nature of Hypertime

Before entering into the specifics of hypertime simulation and discussing the computational methods developed in this research, it is useful to discuss in more detail what is meant by hypertime itself. This chapter presents an intuitive picture of what hypertime entails, followed by an outline of some of the characteristics any simulation must have in order to be considered a consistent hypertime simulation.

2.1 Intuitive Picture of Hypertime

Intuition is a powerful tool for shaping our understanding of any phenomenon, though it does not always do so correctly. This section discusses several aspects of 1T intuition which can be extended to hypertime with minimal error and points out a few areas where 1T intuition fails in hypertime environments.

Hypertime is the study of a single multidimensional time, not of the multiple independent single-dimensional times or "parallel universes" found in fictional writing and movies. Through whatever etymological accident, the idea of universes that mirror our own in many ways but also contain subtle differences are know in popular media as "alternate dimensions," "parallel dimensions," or "alternate times." This is not what this paper discusses. Instead I use "dimension" in

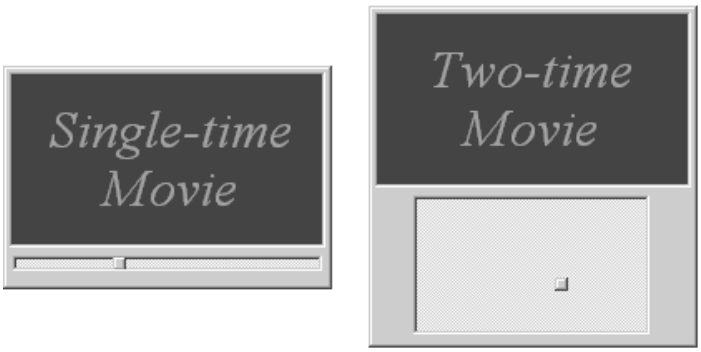


Figure 2.1: Sample of what a 2T hypertime movie player might look like, with a standard movie player for contrast. Note that the hypertime movie player has an entire time plane instead of merely a time line.

a purely mathematical sense to mean the number of distinct real values needed to identify a unique element of a vector space. Although methods employing parallel axes can be used to view multidimensional spaces [14], multidimensional time (like other m-dimensional vector spaces) can always be formulated with m linearly-independent, perpendicular time axes.

Perhaps the best way to explain what this means is to discuss the idea of a 2T hypertime movie player, such as is shown in Figure 2.1. Much like the various 1T movie players used in computers, a hypertime movie player has two basic elements: a display pane, where a representation of space is presented, and a time slider for selecting what time the display pane presents. However, instead of having a single linear slider to represent a section of the time line, a hypertime movie player has an area slider representing a section of the time area.

Given this introductory model, several questions about hypertime may be addressed; the most important deal with the ideas of past, future, and now.

2.1.1 Time Arrows

Given a single frame shown on a hypertime movie player, which other frames represent "future" states and which states "in the past"? In 1T, with a scalar time line,

this is easy to answer; at time t_0 all $t < t_0$ is past, all $t > t_0$ is future. Unfortunately, in mT $\vec{t}$ is a vector and vector spaces typically do not permit a strict ordering function (though some do, albeit in a fundamentally 1T way; see [9] for a discussion). Since a simple ordering of time is not available, some other definition of "past" is needed for hypertime.

To introduce other notions of "past", consider the following poem [30]:

> *Time has two arrows: the historic one,*
> *Which claims that a scar came from something once done;*
> *The other time arrow's by entropy told*
> *And claims that the future's chaotic and cold.*

History embodies the idea that it is easier to read the past than it is to predict the future. For example, though nothing about a person tells us if they are going to wash their hair tomorrow, we have little difficulty in telling if they washed their hair recently. Unfortunately, it is also easier to tell if a ball is about to fall on the ground (because we see it in the air) than it is to tell if it recently fell on the ground—an example where the future is easier to predict than the past. The idea of history, then, is ill suited for any kind of extension, not even being well understood in 1T.

Entropy is better defined; though several different mathematical definitions exist for different environments, it is always a statistical measure of the overall disorder of a system. The second law of thermodynamics states that for any system, entropy continuously increases (or is maximal) for all future times. This permits us to pose the following theorem.

Theorem 1:

In any hypertime simulation with a continuous entropy function $\mathcal{S} : \mathbb{R}^m \to \mathbb{R}$, at any point in time $\vec{t}^*$ not possessing maximal entropy, there is a unique time vector

$\vec{t_p}$ such that the following three conditions hold.

1. $\lim_{\delta \to 0^+} \frac{\mathcal{S}(\vec{t^*}+\delta\vec{t_p})-\mathcal{S}(\vec{t^*})}{\delta} < 0$; that is, entropy decreases along $\vec{t_p}$.
2. $\lim_{\delta \to 0^+} \frac{\mathcal{S}(\vec{t^*}-\delta\vec{t_p})-\mathcal{S}(\vec{t^*})}{\delta} > 0$; that is, entropy increases along $-\vec{t_p}$.
3. For all $\vec{t}$ such that $\vec{t} \cdot \vec{t_p} = 0$, $\lim_{\delta \to 0} \frac{\mathcal{S}(\vec{t^*}+\delta\vec{t})-\mathcal{S}(\vec{t^*})}{\delta} = 0$; that is, entropy remains constant along any time direction perpendicular to $\vec{t_p}$.

$\vec{t_p}$ is called the **past time direction** or **past time arrow** at $\vec{t^*}$.

Proof of Theorem 1:

All continuous multivariate functions $\mathcal{F} : \mathbb{R}^k \to \mathbb{R}$ have a gradient $\nabla\mathcal{F}$ defined. Let $\vec{t_p} = -\nabla\mathcal{S}(\vec{t^*})$. Then conditions 1, 2, and 3 fall right out of the properties of the gradient, completing the proof. □

The simulations presented in this paper are not equipped with entropy functions. Entropy, though almost universally manifest on the large scale, is a statistical property and hence difficult to measure in systems of only a few particles. I thus assume that such an entropy function would exist in a large-scale simulation and use the $\vec{t_p}$ time arrow, though without being able to appeal to Theorem 1 to prove it makes sense. Some non-conclusive evidence that this assumption is reasonable is provided by the results of my simulations, as discussed in Section 5.2 (page 48).

There are several other time arrows as well. The radiative arrow (stating that waves expand, not contract) is a statistical property and can be treated like the entropic arrow in most respects. The cosmological (in the future the universe expands, in the past contracts), weak and quantum (weak force radioactive decay and wave function collapse being irreversible) arrows all depend on large, complicated models of the universe which I have not extended into hypertime. The causal, psychological and perceptual arrows are awkwardly defined in 1T and are

generally thought to be some sort of manifestation of entropy or history. None of these provide useful insights for hypertime research.

2.1.2 Dimensionality of "Now"

A question related to the past and future, but somewhat more involved, deals with the definition of "now" or a "moment of time." We are accustomed to "now" being a point in time (e.g., 0T) and to it being the boundary between the past and the future. Given the definition of the past from the previous section, the boundary between past and future is $(m-1)$T, meaning both notions of "now" cannot be satisfied unless $m = 1$. Since $m \neq 1$ in hypertime, it is worth considering whether a "moment" should have $(m-1)$ or 0 dimensions.

On the side of a 0-dimensional "now" is the observation that a single experience is the result of a single state of the mind and sensation. Assuming that a 2T being could somehow incorporate an entire 1-dimensional continuum of states into a single perception, and act accordingly, seems to violate the claim that we have two dimensions of time. Additionally, though less compellingly, a movie-player *can't* show us more than a 0T moment, because we, being 1T observers, can't absorb more than a 0T moment at any point in time. Hence, at least in the display we are restricted to a 0T moment.

On the side of a $(m-1)$-dimensional "now", I already noted that the entropic time arrow, given by the gradient of entropy, is 1-dimensional; assuming the historic arrow is also 1-dimensional and aligned with the entropic arrow (which it always is in our 1T experience due to the constraints of single-dimensional spaces), then $m-1$ dimensions of time would be neither before nor after any given point. If so, an entire $(m-1)$-dimensional manifold of time would be equally "current" and could quite easily be considered a single "moment".

These two models of "now" appear to illustrate different but compatible ideas,

rather than two models for the same idea. However, they are verbally confounded by the fact that they are identical in 1T. To separate the ideas I have adopted the vocabulary of calling a 0T moment a **time point** and an $(m-1)$T moment a set of **current times** with its dual idea of a **time vector**.

Both ideas are important in the algorithms presented in this paper. For example, particle collisions occur at a specific time point but are resolved to conserve energy along a single past-future time vector. Hypertime AI is even more involved, treating the $m-1$ current time dimensions of an agent as a single entity but using a 1T planning algorithm for each 0T point along that $(m-1)$T manifold.

2.2 Consistent Hypertime Simulations

In addition to the discussion of some of the intuitive and philosophical basis of hypertime presented in the previous chapter, further insight may be gained by asking several more concrete questions about hypertime simulations.

This paper discusses only the interaction of point-like[1] particles moving over time. Thus, each nD/mT particle p is defined by a single mapping

$$\mathcal{P} : \mathbb{R}^m \rightarrow \mathbb{R}^n. \tag{2.1}$$

Although there are many other types of simulation that might be extended to hypertime (such as fluid dynamics, rigid- and soft-body simulations, etc.), I use the term **hypertime simulation** to refer to a piece of software that simultaneously evaluates several particles' relations $\mathcal{P}_i$ in a consistent, temporal manner as is discussed in the following sections.

[1] These particles to have mass and non-zero radii, but these qualities are considered static and can be treated as features of the interactions of the particles rather than dynamic state of the particles themselves.

2.2.1 Consistency

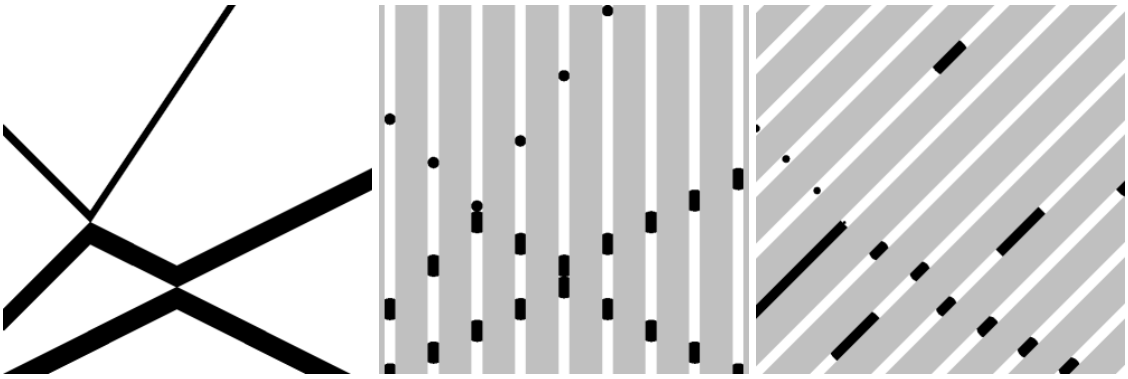
Any simulation, to be useful, needs to be both internally consistent and sufficiently precise. As these terms are often used to mean slightly different things in different sources, I provide the following definitions:

Consistency is a quality of an algorithm $\hat{\mathcal{P}}$ for approximating a mathematical function $\mathcal{P}$. Any consistent algorithm, given unbounded resources of time, space, and internal number representations, can reduce the error $\left\|\hat{\mathcal{P}}(\vec{t}) - \mathcal{P}(\vec{t})\right\|$ for any finite $\vec{t}$ in the domain of $\mathcal{P}$ to arbitrarily small levels.

Precision is a quality of a realization of an algorithm $\hat{\mathcal{P}}$ for approximating a mathematical function $\mathcal{P}$. Given the same bounds on resources available to the algorithm, a more precise algorithm has smaller error $\left\|\hat{\mathcal{P}}(\vec{t}) - \mathcal{P}(\vec{t})\right\|$ than does a less precise simulation of $\mathcal{P}$. An algorithm implementation is considered **sufficiently precise** if the outcome of simulated phenomena is close enough to the outcome of real phenomena that the results of the simulation can be used in place of empirical testing.

For single-time simulations, consistency is almost a given; attention is typically focussed on various details of precision and efficiency. However, the standard iterative methods for simulating 2T fields are not consistent; not matter how generous the resources, the expected error remains large. This fact, combined with the lack of conventional knowledge about what a hypertime particle's motion ought to look like (and hence any notion of sufficient precision), consistency becomes the dominant consideration in evaluating the quality of a hypertime simulation.

Both of the hypertime simulations presented in this paper are designed to be consistent. However, as mentioned in section 1.2.2, my earlier implementations of field-based velocity, acceleration, jerk-forces, and path curvature, hypertime geodesic computations, and many other time-iterative pseudo-physics simulations,



(a) The continuity of space-time.

(b) The (only) correct time-framed view.

(c) One incorrect time framing.

Figure 2.2: Two views of the same 1D/1T space-time demonstrate that, with no prior information, we can determine which direction must be time using only basic intuition.

though all consistent in 1T, were universally inconsistent in mT.

2.2.2 Temporality

Although the most important consideration in hypertime is ensuring the simulations are consistent, hypertime simulations should also be obviously temporal; that is, the time and space dimensions should exhibit characteristics in accordance with our expectations from 1T experience. Since part of the motivation in this work is to better understand the difference between time and space, models of hypertime equations should use principles that apply to standard time.[2]

Generally accepted models of the laws of nature do not treat time and space the same, nor even in a particularly similar way. An example of this may be seen in Figure 2.2. Figure 2.2(a) shows the space-time of a simple one-dimensional world where three objects bounce off of each other. Figure 2.2(b) shows several time

[2] There is a well-developed mathematical theory of time-like spaces (see [9] for details). This work has no direct bearing on hypertime as one of the core concepts in time-like spaces is a strict ordering function, where the ordering defines an implicit single time dimension. Since hypertime, by definition, has multiple time dimensions, this single-dimensional time-like work does not appear to be applicable.

slices of the scene, making the motion of the particles more explicit; Figure 2.2(c) shows an alternate slice along a non-time dimension, causing objects to appear to stretch and shrink, merge and disappear. It is immediately evident which direction indicates time and which space because we have an intuitive grasp of how the two differ.

By emulating 1T laws of nature in my own simulations I hope to preserve the spatial nature of simulated space dimensions and the temporal nature of simulated time dimensions in hypertime. However, I do not wish to restrict myself to any particular set of natural laws, motivating the following definition:

> **Definition 1:** Given a model $\mathcal{M}$ of a set of 1T laws, an nD/mT simulation technique is **temporal with respect to** $\mathcal{M}$ or simply **temporal** if both
>
> 1. setting $m = 1$ recovers a nD/1T simulation which satisfies $\mathcal{M}$, and
> 2. any any point in time $\vec{t}^*$, the simulation has no discontinuities of any order not present in $\mathcal{M}$.

Loosely, the second condition means the hypertime simulation doesn't contain "glitches," manifest by sudden changes in state, compared to the 1T laws. If 1T particles move smoothly, so do the mT particles.

I require that each hypertime simulation of a given phenomena be temporal with respect to accepted 1T models of that phenomena.

2.2.3 Additional Considerations

Things not Specified

In order to leave this work as general and simple as possible, I have decided not to impose constraints not directly related to the idea of hypertime. The geometry

of time and space is not restricted beyond requiring that both be locally flat so that around any given point they operate like vector spaces. No particular laws of nature needs to be satisfied in any given simulation, though all simulations do need to mimic some aspect of 1T laws to preserve temporality. Any simulation technique is permitted, provided is produces consistent, temporal simulations.

Necessarily Hypertemporal

All hypertime simulations must be necessarily hypertemporal. If, for example, everything were static along $m-1$ dimensions of time than mT would be equivalent to 1T; similarly, if $n=1$ we would be able to pose the time-space dual as 1T problems and learn little, if anything, from the plurality of time.

Extensibility vs. Visualization

The work presented here is intended to be an introduction and basis for future research. As such, all of the theoretic background of the simulations is posed in an arbitrary nD/mT setting so that it may form a workable framework for future work. However, the only dimensionality of hypertime that is amenable to visualization is 2D/2T,[3] where a time plane can be accessed by a slider and the entire space at each time presented in the view plane. Believing that interactive visualization is key to developing an intuitive understanding of an environment, all of my code is designed and optimized for this 2D/2T situation to provide a faster, more responsive interactive experience.

[3] 3D or 3T would be fairly simple extensions, using 3D visualization techniques to present the time and/or space volumes, but most 3D visualization depend on animation or internal structure of the scene to convey some part of the depth cue. These cues are not immediately available to us in hypertime particle simulations, so I have chosen to stick with 2D visualizations.

Chapter 3

Hypertime Physics

The overview of hypertime presented in the previous chapter leads to the first simulation task: creating a consistent simulation of a physical hypertime system. For the first year of my research it was far from obvious that this was even possible. As such, I first present some things that do not work, as well as why I believe they fail, after which I present a working simulation and a discussion of why it works were the others fail. I conclude with a brief description of tests performed using my code to verify consistency and temporality.

3.1 Ultra-Hyperbolic Fields

Section 1.2.2 discusses Tegmark's paper on the failure of hypertime fields. His claim is based on two findings from partial differential equations; one about how to categorize different PDEs, the other about the behavior of one of those categories, known as ultra-hyperbolic PDEs. The nature of these findings is outside the scope of this work; interested readers are referred to [13, 2]. This section presents instead a more general but less formal discussion of the results of many hypertime simulations.

3.1.1 Point-Iterative Hypertime

While Tegmark discusses only field-based physics, I have seen a much broader class of hypertime simulations fail for the same reasons. I here define this class, which I call point-iterative hypertime because it revolves around iteratively evaluating the system state at a sequence of time points, each based on the previously evaluated time point.

Any point-iterative hypertime simulation in nD/mT can be expressed in a state-space representation[1]. This representation is useful because it allows us to iterate the state of the simulation from an initial time point to any other time. This is a standard algorithmic setup of most 1T simulations, and is characterized by the following properties:

1. A non-empty set of particles $\vec{p}_i$, each with some persistent state including, but not limited to, a position vector $\vec{x}_i$. Let the number of real numbers stored in each $\vec{p}$ be b (e.g., for a 2D/3T particle with position and velocity, two numbers are needed for the position and six for the velocity, so $b = 8$). Since the state stored in $\vec{p}_i$ varies over time, we can write it as a function of time $\vec{p}_i(\vec{t})$. Note that $\vec{p}_i(\vec{t})$ is not known prior to simulating; nor should it be confused with the function $\mathcal{P}$ given in (2.1): the latter provides only the position $\vec{x}_i$ of the particle, while the former augments it with all the internal state of the particle necessary to uniquely identify its motion.

2. A (generally non-static) matrix-valued field $\mathbf{F} : \mathbb{R}^b \to \mathbb{R}^{b\times m}$. In standard physics, the definition of $\mathbf{F}_i$ depends on the states of all other particles in the universe at a given point in time[2]. The change in $\vec{p}$ due to an infinitesimal

[1] ...as can any finite-order differential equation. Tegmark's mathematical claim about PDEs is a direct parallel to the simulation claim in this section.

[2] Frequently this is denoted by writing $\mathbf{F}(\vec{p}, \vec{t})$, since $\vec{t}$ is the only one of the variables upon which $\mathbf{F}$ depends that $\vec{p}$ also depends upon. I have chosen not to use this notation as it implies that $\mathbf{F}$ depends only on $\vec{p}$ and $\vec{t}$. It may be more instrumental to think of $\mathbf{F}$ depending on the entire simulated universe; something along the lines of $(\mathbf{F}(\mathbb{U}))(\vec{p})$.

change in time $d\vec{t}$ is defined as $d\vec{p} = \mathbf{F}(\vec{p})d\vec{t}$.

3. A k-step simulation of a particle with state $\vec{p_0}$ at time $\vec{t_0}$ to state $\vec{p_k}$ at time $\vec{t_k}$ may be computed as $\vec{p_{j+1}} = \vec{p_j} + \mathbf{F}(\vec{p_j})(\vec{t_{j+1}} - \vec{t_j})$.

For $m = 1$ this is a standard, if simple, iterative simulation technique which converges, as k goes to infinity (and equivalently, $(t_{j+1} - t_j)$ goes to zero), to the solution to the differential equation realized by $\mathbf{F}$. Other iterative methods, such as the ubiquitous Runge-Kutta methods, differ from it only in efficiency and numerical precision, not in principle.

Since iterative simulations move between pairs of $\vec{t}$s, the **time-path** the simulation follows is the sequence of $\vec{t}$s visited. This forms a 1-dimensional path in the m-dimensional vector space of time. In order for the simulation to be consistent, any closed path (one that begins and ends at the same point) would need to exert no net influence on the particle; that is, for any closed time path $\mathcal{C}$,

$$\int_{\mathcal{C}} \mathbf{F}(\vec{p}(\vec{t}))d\vec{t} = \vec{0}. \tag{3.1}$$

This bears a superficial resemblance to the definition of a conservative vector field and the fundamental theorem of line integrals, but is complicated by the fact that $\mathbf{F}$ is a matrix and $\vec{p}$ is a function of $\vec{t}$.

In practice, I was unable to find any point-iterative hypertime which was even vaguely temporal, fundamentally hypertemporal, and satisfied (3.1). Simple matrix extensions of conservative fields certainly did not satisfy (3.1); neither did any of a wide range of geodesic computations on dynamically curved space, more complicated vector field extensions, or any other tested environment. Like Tegmark, my conclusion is not that there is no consistent point-iterative hypertime, but rather that it is far from obvious that any consistent point-iterative hypertime exists.

Interpolating Polynomial($\mathbb{D}$)
1. let $f(x) = 0$
2. let $g(x) = 1$
3. for each $(x_i, y_i) \in \mathbb{D}$:
4. set $f(x) = f(x) + g(x)\frac{y_i x - f(x_i)}{g(x_i)}$
5. set $g(x) = g(x)(x - x_i)$
6. return $f(x)$

Figure 3.1: Example of equation iteration to find an interpolating polynomial.

3.2 Elastic Collisions

Although point-iterative hypertime does not appear to work, there are other simulation techniques that do. One such method adjusts the entire motion equation $\mathcal{P}$ (as given in (2.1) on page 11) rather than adjusting the state of the particle at a given time point. Starting with a nominal motion equation $\mathcal{P}_0$, this method iteratively refines the equation until it arrives at a motion equation that satisfies the simulated mathematical laws.

A simple example of equation refinement can be seen by using the method for creating an interpolating polynomial for a set of points $\mathbb{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ shown in Figure 3.1. At each iteration of the loop beginning on line 3, $f(x)$ interpolates one additional point until, by the end of the loop, it satisfies the requirement that it interpolates all of them. A similar method can be used to refine a set of motion equations for particles.

3.2.1 Distance-based Formulation

There are many ways to formulate elastic collisions. Newtonian physics teaches that elastic collisions are defined by the particles preserving momentum and kinetic energy without passing through one another; that definition is not well suited for hypertime due to the difficulty of defining kinetic energy over a multidimen-

sional time. Instead, I have developed a form more conducive for equation iteration based on distance:

Theorem 2:

Given

- two particles with position functions $\vec{x_1}(\vec{t}) = \mathbf{V}_1\vec{t}$ and $\vec{x_2}(\vec{t}) = \mathbf{V}_2\vec{t}$ and masses m_1 and m_2 (where $\mathbf{V}_i$ is the Jacobian matrix $\mathbf{V}_i = \frac{d}{d\vec{t}}\vec{x_i}(\vec{t})$),
- a collision at time $\vec{t} = \vec{0}$, and
- a past time vector $\vec{t_p}$, where where the "past" is defined by the set of all $\alpha\vec{t_p}$ for all positive scalars α,

the collision is resolved in a consistent, temporal way by replacing each x_i with x_i' where

$$\vec{x_i'}(\vec{t}) = \frac{m_i\mathbf{V}_i + m_j\mathbf{V}_j}{m_i + m_j}\vec{t} + m_j\frac{(\mathbf{V}_i - \mathbf{V}_j)\vec{t_p}}{\|(\mathbf{V}_i - \mathbf{V}_j)\vec{t_p}\|}\frac{\|(\mathbf{V}_i - \mathbf{V}_j)\vec{t}\|}{m_i + m_j}. \tag{3.2}$$

Note that in 1T, $\vec{t_p}$ would typically be -1 so that the past would be all negative t. In mT $\vec{t_p}$ would be some vector pointing into the past, such as the negative gradient of the desired entropy field per Theorem 1 (page 8).

Proof of Theorem 2:

Equation (3.2) is a simple function defined for all $\vec{t}$, and hence consistent. The proof can be completed by showing that (3.2) is temporal with respect to the preservation of past state, momentum, and kinetic energy but prevents the particles from passing through one another. The preservation of the past, prevention of collision, and preservation momentum applies for all values of $\vec{t}$, while energy is conserved along

the past/future time direction; these are all stronger claims than those required by the Definition 1 (page 14).

To show that the past is not modified,

$$\begin{aligned}
\vec{x_1'}(\alpha\vec{t_p}) &= \frac{m_1\mathbf{V}_1 + m_2\mathbf{V}_2}{m_1+m_2}\alpha\vec{t_p} + m_2\frac{(\mathbf{V}_1-\mathbf{V}_2)\vec{t_p}}{\|(\mathbf{V}_1-\mathbf{V}_2)\vec{t_p}\|}\frac{\|(\mathbf{V}_1-\mathbf{V}_2)\alpha\vec{t_p}\|}{m_1+m_2} \\
&= \frac{m_1\mathbf{V}_1 + m_2\mathbf{V}_2 + m_2\mathbf{V}_1 - m_2\mathbf{V}_2}{m_1+m_2}\alpha\vec{t_p} \\
&= \mathbf{V}_1\alpha\vec{t_p}
\end{aligned}$$

and similarly for $\vec{x_2'}(\vec{t})$.

To show the particles do not pass through each other, consider the offset between the two particles given by $\vec{x_i'} - \vec{x_j'}$:

$$\begin{aligned}
\vec{x_i'} - \vec{x_j'} &= m_j\frac{(\mathbf{V}_i-\mathbf{V}_j)\vec{t_p}}{\|(\mathbf{V}_i-\mathbf{V}_j)\vec{t_p}\|}\frac{\|(\mathbf{V}_i-\mathbf{V}_j)\vec{t}\|}{m_i+m_j} - m_i\frac{(\mathbf{V}_j-\mathbf{V}_i)\vec{t_p}}{\|(\mathbf{V}_j-\mathbf{V}_i)\vec{t_p}\|}\frac{\|(\mathbf{V}_j-\mathbf{V}_i)\vec{t}\|}{m_j+m_i} \\
&= \left(m_j(\mathbf{V}_i-\mathbf{V}_j) - m_i(\mathbf{V}_j-\mathbf{V}_i)\right)\vec{t_p}\frac{\|(\mathbf{V}_i-\mathbf{V}_j)\vec{t}\|}{\|(\mathbf{V}_i-\mathbf{V}_j)\vec{t_p}\|(m_i+m_j)},
\end{aligned}$$

which is a constant vector times a non-negative scalar which is zero only at the time point of the collision[3]. Hence, the particles stay on the same side of one another and touch only at the point of collision.

To show that momentum is conserved, first observe that the velocity is given by

$$\frac{d}{d\vec{t}}\vec{x_1'}(\vec{t}) = \frac{m_1\mathbf{V}_1 + m_2\mathbf{V}_2}{m_1+m_2} + m_2\frac{(\mathbf{V}_1-\mathbf{V}_2)\vec{t_p}\vec{t}^T(\mathbf{V}_1-\mathbf{V}_2)^T(\mathbf{V}_1-\mathbf{V}_2)}{\|(\mathbf{V}_1-\mathbf{V}_2)\vec{t_p}\|\|(\mathbf{V}_1-\mathbf{V}_2)\vec{t}\|(m_1+m_2)}$$

[3] unless $\mathbf{V}_i - \mathbf{V}_j$ is singular, for which case it is zero for all $\vec{t}$ in the null space of $\mathbf{V}_i - \mathbf{V}_j$. This corresponds to the particles not moving relative to one another along one or more dimension of time, which is equivalent to a less-hypertemporal $\left(\text{rank}(\mathbf{V}_i - \mathbf{V}_j)\right)$T collision.

and

$$\frac{d}{d\vec{t}}\vec{x_2'}(\vec{t}) = \frac{m_1\mathbf{V}_1 + m_2\mathbf{V}_2}{m_1+m_2} - m_1\frac{(\mathbf{V}_1-\mathbf{V}_2)\vec{t_p}\vec{t}^T(\mathbf{V}_1-\mathbf{V}_2)^T(\mathbf{V}_1-\mathbf{V}_2)}{\|(\mathbf{V}_1-\mathbf{V}_2)\vec{t_p}\|\|(\mathbf{V}_1-\mathbf{V}_2)\vec{t}\|(m_1+m_2)}.$$

The total momentum, as a function of time, is given by

$$\rho'(\vec{t}) = m_1\frac{d}{d\vec{t}}\vec{x_1'}(\vec{t}) + m_2\frac{d}{d\vec{t}}\vec{x_2'}(\vec{t}).$$

The second half of each $m\frac{d}{d\vec{t}}x(\vec{t})$ term $\left(\pm m_1 m_2\frac{(\mathbf{V}_1-\mathbf{V}_2)\vec{t_p}\vec{t}^T(\mathbf{V}_1-\mathbf{V}_2)^T(\mathbf{V}_1-\mathbf{V}_2)}{\|(\mathbf{V}_1-\mathbf{V}_2)\vec{t_p}\|\|(\mathbf{V}_1-\mathbf{V}_2)\vec{t}\|(m_1+m_2)}\right)$ cancels out, leaving us with

$$m_1\frac{d}{d\vec{t}}\vec{x_1'}(\vec{t}) + m_2\frac{d}{d\vec{t}}\vec{x_2'}(\vec{t}) = m_1\frac{m_1\mathbf{V}_1+m_2\mathbf{V}_2}{m_1+m_2} + m_2\frac{m_1\mathbf{V}_1+m_2\mathbf{V}_2}{m_1+m_2}$$

which simplifies to

$$= \frac{m_1^2\mathbf{V}_1 + m_1m_2\mathbf{V}_1 + m_1m_2\mathbf{V}_2 + m_2^2\mathbf{V}_2}{m_1+m_2} = m_1\mathbf{V}_1 + m_2\mathbf{V}_2,$$

a constant expression that is not dependent on time and is equal to the momentum before resolving the collision.

To show that kinetic energy is conserved, first note that without a working definition of kinetic energy in hypertime we can only use the 1T version of kinetic energy[4]. In 1T, t and t_p are scalars, the Jacobian matrices $\mathbf{V}_i$ are just vectors $\vec{v}_i$, and, for the future, $t_p t < 0$. This simplifies the future-time velocity to

$$\frac{d}{dt}\vec{x_1'}(t) \quad = \quad \frac{m_1\vec{v_1}+m_2\vec{v_2}}{m_1+m_2} + m_2\frac{(\vec{v_1}-\vec{v_2})\|\vec{v_1}-\vec{v_2}\|^2 t_p t}{\|(\vec{v_1}-\vec{v_2})\|^2(m_1+m_2)|t_p||t|}$$

[4] For several candidate definitions of mT version of kinetic energy, as well as why I have chosen not to use them, see the discussion in Section A on page 59.

$$= \frac{(m_1 - m_2)\vec{v_1} + 2m_2\vec{v_2}}{m_1 + m_2}.$$

Given the past kinetic energy

$$\mathcal{K}(t) = \frac{m_1}{2}\left\|\frac{d}{dt}\vec{x_1}(t)\right\|^2 + \frac{m_2}{2}\left\|\frac{d}{dt}\vec{x_2}(t)\right\|^2 = \frac{m_1\|\vec{v_1}\|^2 + m_2\|\vec{v_2}\|^2}{2},$$

the simplified velocity equation shows that it is the same as the future kinetic energy

$$\begin{aligned}
\mathcal{K}'(t) &= \frac{m_1}{2}\left\|\frac{d}{dt}\vec{x_1'}(t)\right\|^2 + \frac{m_2}{2}\left\|\frac{d}{dt}\vec{x_2'}(t)\right\|^2 \\
&= \frac{m_1}{2}\left\|\frac{(m_1 - m_2)\vec{v_1} + 2m_2\vec{v_2}}{m_1 + m_2}\right\|^2 + \frac{m_2}{2}\left\|\frac{2m_1\vec{v_1} + (m_2 - m_1)\vec{v_2}}{m_1 + m_2}\right\|^2 \\
&= \frac{m_1\left\|(m_1 - m_2)\vec{v_1} + 2m_2\vec{v_2}\right\|^2 + m_2\left\|2m_1\vec{v_1} + (m_2 - m_1)\vec{v_2}\right\|^2}{2m_1^2 + 4m_1m_2 + 2m_2^2} \\
&= \frac{m_1(m_1^2 + 2m_1m_2 + m_2^2)\|\vec{v_1}\|^2 + m_2(m_1^2 + 2m_1m_2 + m_2^2)\|\vec{v_2}\|^2}{2m_1^2 + 4m_1m_2 + 2m_2^2} \\
&= \frac{m_1\|\vec{v_1}\|^2 + m_2\|\vec{v_2}\|^2}{2}
\end{aligned}$$

Since I have proven stronger claims than those made in Definition 1, (3.2) is temporal with respect to elastic collisions. □

Theorem 2 is designed not only to work in any dimensionality, but also to adjust the motion equations of a pair of particles to resolve their collision in a manner ideal for simulation via equation iteration.

3.2.2 Equation Iteration

Each step of the equation iteration should find one particle collision and update the motion functions of the involved particles to eliminate it. This requires slight modifications to the equations given in Theorem 2; in particular, rather than restricting

the original equations to the form $\vec{x}(\vec{t}) = \mathbf{V}\vec{t}$, the modified equation should linearize the motion of each particle around the point of impact and replace the linear component of motion with the form presented in Theorem 2. This idea is formalized in the following theorem.

Theorem 3:

Given a past time direction $\vec{t_p}$ and two particles with continuous position functions $\vec{x_1}(\vec{t})$ and $\vec{x_2}(\vec{t})$ colliding at time $\vec{t^*}$, let $\mathbf{V}_1 = \frac{d}{dt}\vec{x_1}(\vec{t^*})$ and $\mathbf{V}_2 = \frac{d}{dt}\vec{x_2}(\vec{t^*})$. The collision is resolved in a consistent, temporal way by replacing each $\vec{x_i}$ with $\vec{x_i''}$ where

$$\vec{x_i''}(\vec{t}) = \vec{x_i}(\vec{t}) - \mathbf{V}_i(\vec{t} - \vec{t^*}) + \vec{x_i'}(\vec{t} - \vec{t^*}) \tag{3.3}$$

and $\vec{x_i'}(\vec{t})$ is the same equation given in (3.2).

Proof of Theorem 3:

Since $\mathbf{V}_i(\vec{t} - \vec{t^*})$ is the first-order term of the Taylor expansion of $\vec{x_i}(\vec{t})$ about $\vec{t^*}$, $\vec{x_i}(\vec{t}) - \mathbf{V}_i(\vec{t} - \vec{t^*})$ contains only zero- and two-or-higher-order terms; that is, $\frac{d}{d\vec{t}}\left(\vec{x_i}(\vec{t}) - \mathbf{V}_i(\vec{t} - \vec{t^*})\right)(\vec{t^*}) = \mathbf{0}$. This fact, combined with the proof to Theorem 2, will complete this proof.

To show the new equations do not pass through each other near $\vec{t^*}$, the offset becomes the same constant vector times a positive scalar it was in the proof to Theorem 2, plus the offset at $\vec{t^*}$, plus higher-order terms. The higher-order terms may create a collision at another time, but cannot do so in the immediate neighborhood of $\vec{t^*}$. The offset at $\vec{t^*}$, if non-zero, should be the same direction as the offset of immediately before $\vec{t^*}$, and so will line up with the constant vector term in $\vec{x_i'} - \vec{x_j'}$. Thus, the particles do not collide near the resolved collision time.

From Theorem 2, the resolution of $\mathbf{V}_i(\vec{t}-\vec{t^*})$ is $\vec{x_i'}(\vec{t}-\vec{t^*})$; since the resolution preserves the past, $\vec{x_i'}(\alpha\vec{t_p}-\vec{t^*}) = \mathbf{V}_i(\alpha\vec{t_p}-\vec{t^*})$, so

$$\vec{x_i''}(\alpha\vec{t_p}) = \vec{x_i}(\alpha\vec{t_p}) - \mathbf{V}_i(\alpha\vec{t_p}-\vec{t^*}) + \vec{x_i'}(\alpha\vec{t_p}-\vec{t^*}) = \vec{x_i}(\alpha\vec{t_p}),$$

which gives us past conservation.

To show momentum conservation, Theorem 2 gives us

$$m_1\frac{d}{d\vec{t}}\vec{x_1'}(\vec{t}-\vec{t^*}) + m_2\frac{d}{d\vec{t}}\vec{x_2'}(\vec{t}-\vec{t^*}) = m_1\mathbf{V}_1 + m_2\mathbf{V}_2;$$

inserting this into $m_1\frac{d}{d\vec{t}}\vec{x_1''} + m_2\frac{d}{d\vec{t}}\vec{x_2''}$ yields

$$\begin{aligned}
&m_1\frac{d}{d\vec{t}}\left(\vec{x_1}(\vec{t}) - \mathbf{V}_1(\vec{t}-\vec{t^*}) + \vec{x_1'}(\vec{t}-\vec{t^*})\right) + m_2\frac{d}{d\vec{t}}\left(\vec{x_2}(\vec{t}) - \mathbf{V}_2(\vec{t}-\vec{t^*}) + \vec{x_2'}(\vec{t}-\vec{t^*})\right)\\
&= m_1\frac{d}{d\vec{t}}\vec{x_1}(\vec{t}) + m_2\frac{d}{d\vec{t}}\vec{x_2}(\vec{t}) - m_1\mathbf{V}_1 - m_2\mathbf{V}_2 + m_1\frac{d}{d\vec{t}}\vec{x_1'}(\vec{t}-\vec{t^*}) + m_2\frac{d}{d\vec{t}}\vec{x_2'}(\vec{t}-\vec{t^*})\\
&= m_1\frac{d}{d\vec{t}}\vec{x_1}(\vec{t}) + m_2\frac{d}{d\vec{t}}\vec{x_2}(\vec{t}) - m_1\mathbf{V}_1 - m_2\mathbf{V}_2 + m_1\mathbf{V}_1 + m_2\mathbf{V}_2\\
&= m_1\frac{d}{d\vec{t}}\vec{x_1}(\vec{t}) + m_2\frac{d}{d\vec{t}}\vec{x_2}(\vec{t}),
\end{aligned}$$

which is the definition of the initial momentum.

Kinetic energy is little harder; acknowledging that t, t^* and t_p are scalars with $t_p(t-t^*) < 0$ for the future, $\vec{v_1}$ and $\vec{v_2}$ are vectors instead of matrices, and write the velocity

$$\frac{d}{dt}\vec{x_i''}(t) = \frac{d}{dt}\vec{x_i}(t) - \vec{v_i} + \frac{(m_i-m_j)\vec{v_i} + 2m_j\vec{v_j}}{m_i+m_j} = \frac{d}{dt}\vec{x_i}(t) + \frac{2m_j(\vec{v_j}-\vec{v_i})}{m_i+m_j}.$$

The kinetic energy is thus

$$\begin{aligned}
\mathcal{K}''(t) &= \frac{m_1}{2}\left\|\frac{d}{dt}\vec{x_1''}(t)\right\|^2 + \frac{m_2}{2}\left\|\frac{d}{dt}\vec{x_2''}(t)\right\|^2\\
&= \frac{m_1}{2}\left\|\frac{d}{dt}\vec{x_1}(t) + \frac{2m_2(\vec{v_2}-\vec{v_1})}{m_1+m_2}\right\|^2 + \frac{m_2}{2}\left\|\frac{d}{dt}\vec{x_2}(t) + \frac{2m_1(\vec{v_1}-\vec{v_2})}{m_1+m_2}\right\|^2.
\end{aligned}$$

Bearing in mind that $\|\vec{z}\|^2 = \vec{z}^T\vec{z}$ and that $\vec{w}^T\vec{z} = \vec{z}^T\vec{w}$,

$$\begin{aligned}
\mathcal{K}''(t) &= \frac{m_1}{2}\left(\left\|\frac{d}{dt}\vec{x_1}(t)\right\|^2 + \frac{4m_2}{m_1+m_2}\left(\frac{m_2(\vec{v_2}-\vec{v_1})^T}{m_1+m_2} + \frac{d}{dt}\vec{x_1}(t)^T\right)(\vec{v_2}-\vec{v_1})\right) \\
&\quad + \frac{m_2}{2}\left(\left\|\frac{d}{dt}\vec{x_2}(t)\right\|^2 + \frac{4m_1}{m_1+m_2}\left(\frac{m_1(\vec{v_1}-\vec{v_2})^T}{m_1+m_2} + \frac{d}{dt}\vec{x_2}(t)^T\right)(\vec{v_1}-\vec{v_2})\right) \\
&= \frac{m_1}{2}\left\|\frac{d}{dt}\vec{x_1}(t)\right\|^2 + \frac{m_2}{2}\left\|\frac{d}{dt}\vec{x_2}(t)\right\|^2 \\
&\quad + \frac{2m_1m_2}{m_1+m_2}\left(\|v_1-v_2\|^2 + \frac{d}{dt}(\vec{x_1}(t)-\vec{x_2}(t))^T(\vec{v_2}-\vec{v_1})\right).
\end{aligned}$$

Because the position functions are locally smooth, within a neighborhood of t^* it holds that $\frac{d}{dt}(\vec{x_1}(t) - \vec{x_2}(t)) = \vec{v_1} - \vec{v_2}$, causing the last term to cancel out, leaving

$$\mathcal{K}''(t) = \frac{m_1}{2}\left\|\frac{d}{dt}\vec{x_1}(t)\right\|^2 + \frac{m_2}{2}\left\|\frac{d}{dt}\vec{x_2}(t)\right\|^2,$$

the kinetic energy before the collision. For $m = 1$, this neighborhood extends to the nearest collisions in each time direction, as we expect from standard 1T physics. □

The proof presented above only demonstrates that kinetic energy is conserved in the neighborhood of the collision. While this is enough to prove Theorem 3 and satisfy the definition of a consistent temporal simulation, it is easy to see that kinetic energy is preserved along the entire time path defined by the set of $\vec{t}$ that satisfy the partial differential equation $\left(\frac{d}{d\vec{t}}(\vec{x_1}(\vec{t}) - \vec{x_2}(\vec{t}))\vec{t}\right)^T(\vec{v_2} - \vec{v_1}) = \|\vec{v_2} - \vec{v_1}\|^2$. In effect, this means that the nonlinearities imposed by the collision curve not only the position at other non-past non-future times, but also the very definition of what "future" is. I chose not to model this curving of the future time paths for the simulations in this paper, approximating it with straight futures instead.

Resolve Collision(p_1, p_2, $\vec{t^*}$)

1. let $\mathbf{V}_1 = p_{1\,\mathbf{V}}(\vec{t^*})$
2. let $\mathbf{V}_2 = p_{2\,\mathbf{V}}(\vec{t^*})$
3. let $\mathbf{V}_{=}\mathbf{V}_1 - \mathbf{V}_2$
4. let $\mathbf{A} = \frac{p_{1\,m}\mathbf{V}_1 + p_{2\,m}\mathbf{V}_2}{p_{1\,m}+p_{2\,m}}$
5. let $\vec{x_p} = \frac{\mathbf{V}\vec{t_p}}{\|\mathbf{V}\vec{t_p}\|(p_{1\,m}+p_{2\,m})}$
6. let $\mathbf{V}_v = \mathbf{V}^T\mathbf{V}$
7. let $p_{1\vec{x}}(\vec{t}) = p_{1\vec{x}}(\vec{t}) + (\mathbf{A} - \mathbf{V}_1)(t - t^*) + p_{2\,m}\|v(t - t^*)\|x_p$
8. let $p_{2\vec{x}}(\vec{t}) = p_{2\vec{x}}(\vec{t}) + (\mathbf{A} - \mathbf{V}_2)(t - t^*) - p_{1\,m}\|v(t - t^*)\|x_p$
9. let $p_{1\mathbf{V}}(\vec{t}) = p_{1\mathbf{V}}(\vec{t}) + \mathbf{A} - \mathbf{V}_1 + p_{2\,m}\vec{x_p}\frac{(\vec{t}-\vec{t^*})^T}{\|\mathbf{V}(\vec{t}-\vec{t^*})\|}\mathbf{V}_v$
10. let $p_{2\mathbf{V}}(\vec{t}) = p_{2\mathbf{V}}(\vec{t}) + \mathbf{A} - \mathbf{V}_2 - p_{1\,m}\vec{x_p}\frac{(\vec{t}-\vec{t^*})^T}{\|\mathbf{V}(\vec{t}-\vec{t^*})\|}\mathbf{V}_v$

Figure 3.2: Resolving an elastic collision using the distance-based formulation.

Pseudocode realizing the collision resolution technique presented in Theorem 3 is presented in Figure 3.2. Note that the parameter $\vec{t_p}$, indicating the past time direction, is not specified in Figure 3.2. Setting $\vec{t_p} = -\alpha\vec{t^*}$, where α is any positive scalar, will leave the state at initial time $\vec{t} = \vec{0}$ unchanged; this is the setting I have used in my simulations. Other choices are still consistent with elastic collisions, but remove the ability to specify an initial condition since collisions will change the state at $\vec{0}$.

The pseudocode given here is expressed in terms of first-class functions; if a particle is involved in six collisions, its position function is built out of seven different routines, each stored and evaluated for each position or velocity query. In early implementations this overhead caused the entire simulation to run very slowly. However, by noting the similar structure of the functions and storing a list of the vectors and matrices needed at each step rather than calling code on the heap, I achieved a speedup of roughly one order of magnitude. Even this speedup left the code too slow for practical use (taking ten minutes to resolve collisions for seven particles); additional optimizations are noted in the next section.

3.2.3 Collision Location

Locating the points in time where particles collide is generally the most difficult part of the actual simulation. The intersections of particles p_i and p_j are the zeros of $\vec{x}_i(\vec{t}) - \vec{x}_j(\vec{t})$, and initially I fed this function into standard numerical root finders. However, standard numerical methods (including Newton's method, Broyden's method, the method of steepest descent, and Runge-Kutta methods) are complicated by the presence of zeros that represent already-resolved collisions, by over-determined or under-determined systems when $n \neq m$, and by discontinuities in the derivative. The best off-the-shelf root finder I tried was still far too slow for practical use, motivating the following customized method.

Custom Collision Detector

This custom collision detector is based Newton's method, but tuned for the peculiarities of hypertime collisions. To find the time point where two particles, p_i and p_j, intersect, given a time guess $\vec{t_{ij}}$, this algorithm begins to apply the standard Newton method: that is, it analytically finds the intersection of the first-order Taylor polynomials of the position functions. These linear Taylor approximations, given the information available via the pseudocode in Figure 3.2, are

$$\hat{\vec{x}}_i(\vec{t}) = p_{i\,\vec{x}}(\vec{t_{ij}}) + p_{i\,\mathbf{V}}(\vec{t_{ij}})(\vec{t} - \vec{t_{ij}}).$$

For ease of notation, define $\hat{\mathbf{V}}_{0\,i} = p_{i\,\mathbf{V}}(\vec{t_{ij}})$ and $\hat{x_{0}}_i = p_{i\,\vec{x}}(\vec{t_{ij}}) - p_{i\,\mathbf{V}}(\vec{t_{ij}})\vec{t_{ij}}$ which reduces the linearized motion equations to

$$\hat{\vec{x}}_i(\vec{t}) = \hat{x_{0}}_i + \hat{\mathbf{V}}_{0\,i}\vec{t} \quad \text{and} \quad \hat{\vec{x}}_j(\vec{t}) = \hat{x_{0}}_j + \hat{\mathbf{V}}_{0\,j}\vec{t}$$

The analytic solution to this approximation is the smallest-magnitude $\vec{t}$ which minimizes $\|\hat{\vec{x}}_i(\vec{t}) - \hat{\vec{x}}_j(\vec{t})\|$. If $n = m$ and $\hat{\mathbf{V}}_{0\,i} - \hat{\mathbf{V}}_{0\,j}$ were of full rank, the solution would be

$$\hat{\vec{t}} = (\hat{\mathbf{V}}_{0\,i} - \hat{\mathbf{V}}_{0\,j})^{-1}(\hat{\vec{x_0}}_j - \hat{\vec{x_0}}_i);$$

however, given, in general, $n \neq m$ and $\hat{\mathbf{V}}_{0\,i} - \hat{\mathbf{V}}_{0\,j}$ is not necessarily full rank, using the Moore-Penrose pseudo-inverse [24, 25] instead to finds the smallest value of $\vec{t}$ which minimizes $(\hat{\mathbf{V}}_{0\,i} - \hat{\mathbf{V}}_{0\,j})\vec{t}$,

$$\hat{\vec{t}} = (\hat{\mathbf{V}}_{0\,i} - \hat{\mathbf{V}}_{0\,j})^{+}(\hat{\vec{x_0}}_j - \hat{\vec{x_0}}_i).$$

This value of $\hat{\vec{t}}$ can then be used as the new $\vec{t_{ij}}$ in the next Newton-like iteration, repeating until either $\vec{t_{ij}}$ stops changing, meaning the algorithm has found a time of closest approach, or $\vec{t_{ij}}$ fails to converge after a specified number of steps, meaning it has failed to find a collision with the given initial guess.

Given a time of closest approach $\vec{t_{ij}}$, the time of collision is $a\vec{t_{ij}}$, where $0 < a \leq 1$, such that $\|p_{i\,\vec{x}}(a\vec{t_{ij}}) - p_{j\,\vec{x}}(a\vec{t_{ij}})\| = p_{ir} + p_{jr}$, where p_{kr} is the radius of particle p_k. If no such a exists, no collision exists at the given point of closest approach. Any standard numerical method may be used here to find a; I used an implementation of Newton's method, using the formula $\frac{d}{da}\|\vec{x}(a\vec{t})\|^2 = 2\vec{x}(a\vec{t})^T \left(\frac{d}{d\vec{t}}\vec{x}(a\vec{t})\right)\vec{t}$ to derive the slope. Pseudocode implementing this two-stage collision finder is shown in Figure 3.3.

Finding all Collisions

One challenge in collision detection is resolving the collisions each only once with collisions closer to the initial time being resolved before collisions in the more distant future. One solution is to use a collision table. Given k total particles, this is

Find Collision(p_i, p_j, TOL, STEPS, REPEAT)
1. let $\vec{t}^* = \vec{\infty}$
2. repeat REPEAT times:
3. select some $\vec{t}$
4. repeat STEPS times (or until break):
5. let $\hat{\vec{t}} = \left(p_i\mathbf{v}(t) - p_j\mathbf{v}(\vec{t})\right)^+ \left(p_{i\,\vec{x}}(\vec{t}) - p_{j\,\vec{x}}(\vec{t}) + \left(p_j\mathbf{v}(\vec{t}) - p_i\mathbf{v}(\vec{t})\right)\vec{t}\right)$
6. if $\|\vec{t} - \hat{\vec{t}}\| <$ TOL:
7. let $f(a) = \|p_{i\,\vec{x}}(a\hat{\vec{t}}) - p_{j\,\vec{x}}(a\hat{\vec{t}})\| - p_{ir} - p_{jr}$
8. let a = findzero(f, 0, 1, TOL)
9. if a and $\|a\hat{\vec{t}}\| < \|\vec{t}^*\|$ then let $\vec{t}^* = a\hat{\vec{t}}$
10. break
11. let $\vec{t} = \hat{\vec{t}}$
12. if $\vec{t}^* \neq \vec{\infty}$ then return $\vec{t}^*$

Figure 3.3: Numerical method for discovering collisions of particles with arbitrary n, m, and particle radii.

a k-by-k table where the (i, j)th cell stores either the time at which particles p_i and p_j collide or a flag indicating that they do not collide. By updating the rows and columns associated with each particle after each collision resolution, this process not only guarantees the collisions are resolved near the initial time first with no duplications, but also reduces the computational complexity of the collision resolution process by a linear factor. The updating procedure and driver code for resolving all collisions is shown in Figure 3.4.

The collision resolver is designed to guarantee that resolving any collision does not impact the state of any particles at the initial time. Though there is no impact at the initial time due to resolving a collision, there is generally a small non-zero impact on the trajectories of the involved particles near the initial time. This can mean that resolving a collision of particles p_a and p_b at some large time $\vec{t}_1$ causes p_a to collide with p_c at some small $\vec{t}_2$, when, previous to resolving the collision at $\vec{t}_1$, p_a and p_c did not collide. It is also possible that resolving the collision at time $\vec{t}_2$ will cause p_a and p_b to come back into collision. I do not have a proof that this

```
Resolve All Collisions(ℙ = {p_1, p_2, ..., p_k}, TOL, STEPS, REPEAT)
 1. let T be a k-by-k table
 2. for each (p_i, p_j) ∈ ℙ × ℙ:
 3.    if i < j then T_ij = Find Collision(p_i, p_j, TOL, STEPS, REPEAT)
 4. while T is not empty:
 5.    let i, j be the location of the smallest entry in T
 6.    Resolve Collision(p_i, p_j, T_ij)
 7.    for each a ∈ {1, 2, ..., i − 1}:
 8.       T_ai = Find Collision(p_i, p_a, TOL, STEPS, REPEAT)
 9.       T_aj = Find Collision(p_j, p_a, TOL, STEPS, REPEAT)
10.    for each a ∈ {i + 1, i + 2, ..., j − 1}:
11.       T_ia = Find Collision(p_i, p_a, TOL, STEPS, REPEAT)
12.       T_aj = Find Collision(p_j, p_a, TOL, STEPS, REPEAT)
13.    for each a ∈ {j + 1, j + 2, ..., k}:
14.       T_ia = Find Collision(p_i, p_a, TOL, STEPS, REPEAT)
15.       T_ja = Find Collision(p_j, p_a, TOL, STEPS, REPEAT)
16.    T_ij = ∅
```

Figure 3.4: Using equation iteration to resolve all collisions of a set of particles.

process will necessarily terminate, but all test runs did terminate.

Efficiency and Optimization (in 2D/2T)

Note that with k particles, for every collision resolved there are $2k - 4$ calls to Find Collision. If n or m are larger than 2, each of these calls necessitates several numerical pseudo-inverse calls, greatly slowing down the simulation. However, for $n = m = 2$ the pseudo-inverse can be performed explicitly using only a dozen floating-point operations. This optimized collision detection method, combined with the implementation details noted at the end of the last section, reduced run-time significantly; where the average test case of seven particles using first-class functions and a generic zero finder took a little over an hour to resolve, the same cases take only a few seconds with the optimized code and algorithms.

3.2.4 Testing

I performed empirical testing as a check to ensure the hypertime collision method described above works in practice. An instance of this method with $m = 1$ did indeed produce standard elastic collisions. I ran many tests with 2D/2T, as well as some with 3D/2T and 3D/3T, with k between 2 and 30. For all values of n, m, and k and initial particle states tested, the collision resolution process did eventually terminate and the initial condition remained unmodified by resolving the collisions. In the 2D/2T tests, where I had the visualization tools to use in testing it, no particle overlaps were observed in the final simulations. While empirical testing is not conclusive, this result does suggest that the code correctly simulates consistent, temporal elastic hypertime particle collisions.

Chapter 4

Hypertime Navigation

When asking questions such as "What is hypertime like?", what is generally meant is "What would it be like to live in a hypertemporal world? How would our brains react to such an environment? How do hypertime beings make decisions?" Although no one can answer these questions from direct experience (since we cannot ourselves live in hypertime), approximate answers can be gained by designing algorithms that perform "intelligent" tasks in hypertime.

The mathematically simplest such task of which I am aware, and the one I have implemented an algorithm to perform in hypertime, is the task of navigating through a crowd without colliding with anything.

4.1 Curving Time

At its most basic level, intelligence manifests itself in the making of informed choices. These choices are assumed to influence some set of unobserved parameters (the future) while unable to impact some set of observed parameters (the past). This ability to sort events into two groups based on causality gives us the historic time arrow, which in 1T always lines up with the entropic and other time arrows. Whether the various time arrows could disagree at all is a question for

philosophers or theoretical physicists; as a computer scientist, I am content merely to observe they do line up in 1T and set up the mT simulations accordingly.

Theorem 1 (page 8) states entropy extends radially in time from some initial time point, implying that the "current" times (those neither in the future nor the past compared to each other) are smooth nested manifolds of dimensionality $m-1$ surrounding that point. Given the assumption that agents operate some considerable time after the initial time, the set of "current" times of each agent is vast, possibly even infinite. There are many possible solutions to this unbounded present information. For example, it may be possible to bound the neighboring time actually needed to make decisions at a given time point. Such a head-on approach is more complicated than is necessary, though; instead, I have chosen to curve the geometry of time.

Section 2.1 on pp. 6–7 introduces the image of a time plane. The proposed curving of time modifies that image, replacing the infinite flat expanse of time with an infinitely long tube of time. Thus, no matter where the initial time may be, as long as it is several times the circumference of the time tube away from the present time the present time frontier will be approximately a circumference of the tube. This reduces the problem of separating history and the future to a finite problem without any additional problem-specific optimizations.

Mathematically, this means that, given $\vec{t} = (t_1, t_2, \ldots t_m)^T$, I define $\vec{t} = (t, \vec{s})$, where $t = t_1$ and $\vec{s} = (t_2, \ldots t_m)^T$, and let $\vec{s}$ reside in any wrapping geometry, such as an elliptical or toroidal geometry. For the purposes of this work, the exact model of the finite-sized geometry is not important; that it is finite and locally flat is all that is required.

Note that time was not curved in the previous chapter. All of the mathematics involved in elastic collisions assumed a flat time.

4.2 nD Navigation

Navigating crowds of moving obstacles without collisions is a nontrivial task, even in 1T, and extending these algorithms to 2T adds an additional level of complexity. Many good algorithms for the 1T world do exist, but the problem remains a matter of active research in artificial intelligence, where it is used for robotics, and in computer graphics, where various crowd algorithms are used for dynamically-generated content. Since the algorithm should work in 1T as well as mT environments, I begin the analysis of mT crowd navigation by selecting an appropriate 1T algorithm to extend.

In selecting an algorithm to extend to hypertime, it is important the 1T version be both extremely fast (necessary to accommodate the computational overhead needed to add extra dimensions of time and still retain real-time computation speeds) and based only in time-extendable mathematics. Neither of these criteria are satisfied by common methods based on speculative simulation and exploration of decision trees, which generally exhibit runtimes exponential in the number of dimensions; however, both are satisfied by what is perhaps the oldest known crowd navigation algorithm.

This section discusses the algorithm in a nD/1T setting only. Section 4.3 discusses extending it to mT.

4.2.1 Maneuvering Boards (and related work)

Algorithms for crowd avoidance generally are regarded as being restricted to the computer age; however, maritime navigators have long possessed a sophisticated algorithm for planning safe courses through crowded harbors using charts known as maneuvering boards. Though I have not been able to determine how long maneuvering boards have been in use, I have found books detailing nuances of their

use printed as far back as 1903 [23] and as recently as 2001 [19]; the 1903 reference does not imply that the tool is at all new, nor does the 2001 publication suggest it is waning in popularity.

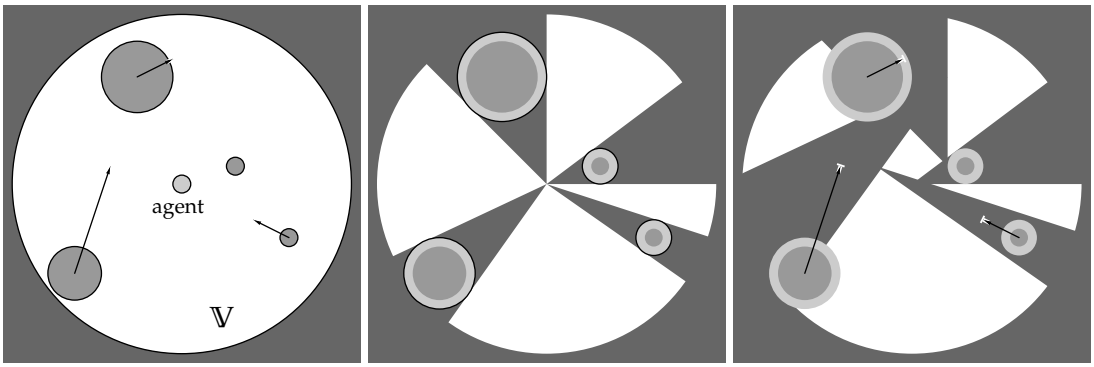
However, it is not the longevity of maneuvering board-based navigation techniques that is attractive, but rather their efficiency and geometric base, which make them prime candidates for hypertime extensions. The underlying algorithm is known by many names in the AI community, including "a maneuvering-board approach" [31], "geometric constraint-based reasoning" [28], "the relative velocity paradigm" [17], "velocity obstacles" (in both linear [18] and nonlinear [27] forms), and "collision cones" [10]. None of these instances differ on a fundamental level from one another; all are computerized versions of 2D maneuvering boards[1] with particular implementation details aimed at different operating environments.

At its core, the maneuvering board algorithm is a method of selecting a target velocity such that, for at least a finite time window, the predicted motion of each obstacle does not come in contact with the position of the agent due to its target velocity. Typically, this is done by assuming that all obstacles will maintain a constant velocity. Mathematically, given a set of achievable velocities $\mathbb{V}$, a set of obstacles defined by their position, velocity, and radius $(\vec{x}_i, \vec{v}_i, r_i) \in \mathbb{O}$, and a time horizon $t_{\max}$, the solution provided by the maneuvering board algorithm is the $\vec{v} \in \mathbb{V}$ closest to the desired velocity subject to the constraints

$$\|(\vec{x} - \vec{x}_i) + (\vec{v} - \vec{v}_i)t\| > r + r_i \quad \forall\, t \in (0, t_{\max}] \quad \forall\, (\vec{x}_i, \vec{v}_i, r_i) \in \mathbb{O}. \tag{4.1}$$

Graphically, the set of velocities permitted by (4.1) is shown in Figure 4.1. A simple set of geometric operations allow us to plot the velocities that would lead

[1] Apparently none were designed to be computerized maneuvering boards; the group at Martin Marietta [31, 28] discovered the link when presenting a prototype of their work to naval officers; the group at NASA and UCLA [17, 18, 27] make no reference to either that earlier work or to maneuvering boards; all other known sources can be traced back to one of those two groups' papers.



(a) The agent, four obstacles, and set of attainable velocities $\mathbb{V}$.

(b) Obstacles padded with radius of agent; illegal velocities for motionless obstacles.

(c) Illegal velocities after displacing by obstacle velocities.

Figure 4.1: Graphic depiction of the velocities allowed by (4.1), assuming $t_{\max} = \infty$. For other values of $t_{\max}$ the shaded triangles in (b) and (c) would be truncated. The best v is the point in the white region of (c) that is closest to the ideal velocity.

to a collision; our eyes can then pick out the permitted velocity which is closest to the one we desire. This is the core of the maneuvering board algorithm.

4.2.2 Computational Tractability

Visually finding the optimal $\vec{v}$ given a plot of $\mathbb{V}$ is easy. Computationally, however, the allowable subset of $\mathbb{V}$ is a concave, possibly discontinuous set and selecting the optimal element is, in nD, a computationally expensive problem. None of the papers cited in the previous section provide an efficient implementation for the nD case; the only implementation explicitly described is based on polyhedral approximations of the collision zones and was implemented only in 2D.

Suppose we have k obstacles and approximate the boundaries of the illegal region caused by each with a concave polyhedron, expressed as a conjunction of q linear inequalities. Then the optimal $\vec{v}$ either lies in the middle of one of the bounding regions (of which there are qk) or at the intersection point created by n boundaries. There are $\binom{k}{n}q^n$ possible intersection points, each of which takes $O(n^3)$ operations to discover. In general, all $qk + \binom{k}{n}q^n$ possible points must be checked

against $q(k-1)$ other inequalities to prevent returning a $\vec{v}$ which is blocked by one or more of the obstacles.

With proper optimizations, this brute-force technique works quite well in 2D where $q = 3$ is ample and there are only $\frac{9}{2}k(k-1)$ easily-detectable intersections; it was the method used by [31] and others. However, 3+D adds additional complications. The edges of the collision cones can be modeled precisely with only two linear inequalities in 2D; in 3D, they are curved and can only be approximated. The polygonal faces cause the algorithm to favor the centers of the faces, which are closer to the center of the cone, and in my tests were seen to cause frequent collisions as two agents would both stick to a plane until they were so close they could no longer avoid colliding with one another. With discretizations of at least $q = 7$ the agents begin to avoid each other fairly well, with only one collision every few hundred frames. However, $q = 7$ in 3D gives $\frac{343}{6}k(k-1)(k-2)$ possible intersections each of which takes much longer to find and check than in 2D. In my test implementation the 3D case runtime was measured in minutes or even hours for the same ks that the 2D case solved several times a second.

Given this lack of scalability in the only algorithmic engine actually described in the literature, I have developed a stochastic optimization algorithm specifically designed to take advantage of the bounded region and the knowledge of which vectors would be better than the current best so far.

Given a current best velocity $\vec{v_0}$ and a target velocity $\vec{v_{\text{goal}}}$, generating only test vectors $\vec{v'} \in \mathbb{V}$ such that $\|\vec{v'} - \vec{v_{\text{goal}}}\| < \|\vec{v_0} - \vec{v_{\text{goal}}}\|$ and checking to see if $\vec{v'}$ is permitted by all obstacles is sufficient to show that $\vec{v'}$ is better than $\vec{v_0}$. Beyond generating points only within the possibly better region, any stochastic optimization technique works. I used a two-part algorithm, alternately generating vectors near the previous best point to climb up continuous allowable regions and generating random vectors in the test region to escape local maxima. As my implemen-

tation of this method provided real-time collision-free simulations on a standard desktop computer even when simultaneously simulating thirty or more 3D agents with high-collision goals such as having all the agents try to reach the opposite side of the crowd, I expended no additional effort to decrease runtimes.

4.2.3 Other AI Tasks

The structure of the maneuvering board problem allows many goals and constraints beyond merely avoiding obstacles. Any maneuvering-based task that can be expressed in terms of satisfying and non-satisfying velocities can be added to the constraints generated by the obstacles. For example, [28] discusses maneuvering-board obstacle generation for ensuring that various types of submersible sensor arrays maintain contact with targets, avoiding obstacles of arbitrary geometry, and coping with known nonlinear obstacle trajectories. One particular task that becomes useful in hypertime is staying within a certain distance of another agent, which is essentially the inverse of the obstacle avoidance problem.

4.3 Cylindrical mT Navigation

Section 4.1 noted the use of a cylindrical time geometry $\vec{t} = (t, \vec{s})$, with all values of $\vec{s}$ for any given t being equivalently "now." This means that the navigation algorithm must store the state of each agent for all values of $\vec{s}$ and plan how to advance this entire $(m-1)$-dimensional time manifold simultaneously. Thus, the cylindrical-time extension of (4.1) is to select the best $\vec{v}(\vec{s})$ such that

$$\| \left(\vec{x}(\vec{s}) - \vec{x}_i(\vec{s})\right) + \left(\vec{v}(\vec{s}) - \vec{v}_i(\vec{s})\right) t \| > r + r_i \quad \forall\, t \in (0, t_{\max}] \quad \forall\, \vec{s}. \tag{4.2}$$

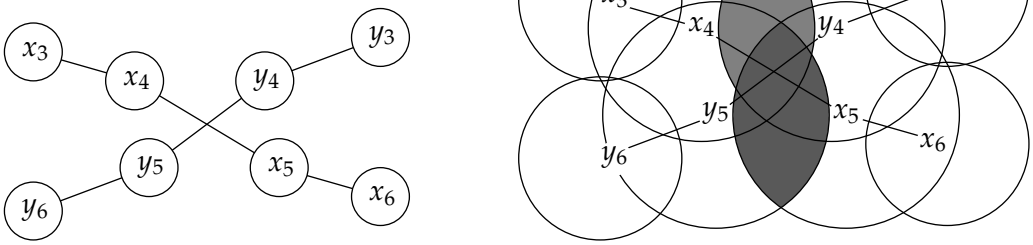
It is generally non-trivial to select an appropriate family of functions for $\vec{x}(\vec{s})$ and $\vec{v}(\vec{s})$, one that is computationally efficient and sufficiently expressive to allow the agents to maneuver freely. A form based on partial Fourier series over the $\vec{s}$ axes is in a sense optimal, in that as additional terms are added it provides a complete basis for all possible positions, while for a small set of terms it is the set of all positions with smooth motion along the $\vec{s}$ axes. However, solving (4.2) in this context is computationally prohibitive even in $\vec{s}$-geometries where such series are easily defined.

A simpler but still sufficiently expressive function family is given by interpolating between several sample points. Given a set of q points uniformly distributed over the $\vec{s}$-geometry, linear interpolation over all other values of $\vec{s}$ is computationally simple and, for large q, is just as expressive as any other method.

4.3.1 Chain-mesh, Fatter When Longer

On its face, a discrete approach to $\vec{x}(\vec{s})$ and $\vec{v}(\vec{s})$ makes solving (4.2) almost trivial. If, for each of the q time points in the discretization we create a separate instance of the 1T agent navigation algorithm, these subagents can satisfy (4.2) at the discretization points by avoiding other subagents of the same value of $\vec{s}$, ignoring all other subagents. However, this introduces two problems. First, the velocity of the agent along any time path with non-zero $\vec{s}$ is unbounded, introducing arbitrarily large discontinuities and violating the second condition of temporality given in Definition 1. Second, the discretization described allows agents to collide at times other than the discretely-sampled values of $\vec{s}$, as illustrated in Figure 4.2(a).

The solution to the first problem is fairly simple. The maneuvering-board algorithm implicitly assumes that each agent has a maximum speed; along the t-axis, this embodied in the definition of $\mathbb{V}$; in the $\vec{s}$-discretization it is enforced though a requirement that, for neighboring samples $\vec{x}_i(t) = \vec{x}(t, \vec{s}_i)$ and $\vec{x}_j(t) = \vec{x}(t, \vec{s}_j)$,



(a) 2T Chain meshes that collide at $s = 4.5$ (s-value for each subagent shown inside circles)

(b) Agents x and y sees their radii padded by their s-velocities, detecting collision.

Figure 4.2: An example of how collisions might be overlooked due to discretization, shown for $m = 2$. By making each sub-agent large enough to overlap with all neighboring sub-agents in the same mT agent we are able to make these collisions illegal, with the cost of being somewhat overly cautious.

$\|\vec{x}_i(t) - \vec{x}_j(t)\|$ must be less than some constant. In a 1T context, this is sort of like having the q different agents all chained together in a big mesh, except that, since $\vec{x}_i(t)$ and $\vec{x}_j(t)$ represent different $\vec{s}$-times, the agents in this chain loop are perfectly able to occupy the same point in space at the same t-time. This means that each 1T portion of the discretized agent, $\vec{x}_i$, has constraints added to keep it close to all of its neighbors but no constraint to avoid colliding with them.

Solving the collisions between subagents in the discretization is somewhat harder. An example of this kind of problem is found in Figure 4.2(a), where the two agents collide between $s = 4$ and $s = 5$, but none of the neighboring subagents detect the collision. One way around this is to make the working radius of each sample equal to its static radius plus half the distance to its farthest neighboring $\vec{s}$-discretization. This ensures that if there is a collision anywhere between two discretizations at least one of the discrete samples discovers agents x and y overlap. It does have the side effect that agents moving quickly in $\vec{s}$ will give a wider berth to all obstacles than will an agent moving slowly in $\vec{s}$, but this is not so large a concession as to make the agents act in a visibly incorrect way.

4.3.2 Chains and Radii on Maneuvering Boards

Implementing the extensions to the maneuvering board algorithm which are necessary for the hypertime navigation technique presented in the previous section is fairly straightforward. The radius increase needs no computational explanation; simply pad both the agent and the obstacle's radii with half their $\vec{s}$-velocity.

The chain constraints are essentially the inverse of the collision avoidance constraints, blocking velocities that lead to the agents getting too far apart rather than those that lead to their getting too close together. Given a set of neighboring subagents with position, t-velocity, and chain-length $(\vec{x}_i, \vec{v}_i, \ell_i) \in \mathbb{S}$, we find the $\vec{v} \in \mathbb{V}$ closest to the desired velocity subject to the constraints

$$\left.\begin{array}{ll} \|(\vec{x} - \vec{x}_i) + (\vec{v} - \vec{v}_i)t\| > r + r_i & \forall\, (\vec{x}_i, \vec{v}_i, r_i) \in \mathbb{O} \\ \|(\vec{x} - \vec{x}_i) + (\vec{v} - \vec{v}_i)t\| < \ell_i & \forall\, (\vec{x}_i, \vec{v}_i, \ell_i) \in \mathbb{S} \end{array}\right\} \quad \forall\, t \in (0, t_{\max}]. \tag{4.3}$$

Not all of the algorithmic extensions are as simple, however. One challenge arises when the "chain" between neighboring subagents is taut. If x_i's neighbors pull it in several different directions, it is quite possible that no safe maneuvers exist. In practice, this means that moving at near-maximum $\vec{s}$-velocity increases the risk of collision. This is reflected by adding the objective of keeping the $\vec{s}$-velocity as small as possible as a standing goal to be merged with any other maneuvering goal that might exist for an agent.

A far more pressing difficulty with this model, however, lies in defining what the maneuvering goals of each 1T agent ought to be. Consider, for example, a 2T agent where at time $(t, s) = (0.25, 2)$ the agent is to be at $(x, y) = (1, 1)$ and at time $(t, s) = (0.75, 5)$ the agent is to be at $(x, y) = (-1, 3)$. If we look at the subagent corresponding to $s = 0.5$, where should it be trying to go at $t = 0$? Is that the same place to which it should be going at $t = 1.8$? At $t = -20$? Although this subagent

does not need to reach either goal itself, it does need to minimize the s-velocity of the subagents that do, both for the safety reasons noted above and to increase the maneuverability of those subagents.

In general, this is a difficult problem. I do not know of an algorithm for determining subagent goals that will produce provably minimal error. I did test a number of approximate algorithms, including averaging goals based on their distance from a subagent in time, averaging based only on the magnitude of the $\vec{s}$-separation between the agent and the goal, and minimizing the $\vec{s}$-velocity unless a goal was only t-separated from the subagent. All of these caused the agents to either hesitate, minimizing $\vec{s}$-velocity at the cost of moving toward goals, or to move towards a goal at the cost of the maneuverability of the neighboring agents; in crowded simulations, many goals were missed. Ultimately I selected a weighted average between the nearest-in-time goal and the desire to minimize $\vec{s}$-velocity, which, though still not ideal, did enable the agents to reach their goals in each test run unless space-time separation of the goals (in the sense of the minimum velocity needed to get between them, $\frac{\|\vec{p}_i - \vec{p}_j\|}{\|\vec{t}_i - \vec{t}_j\|}$) was close to the maximum speed of the agent. The implementation of this idea is shown in Figure 4.3.

4.3.3 Implementation and Testing

To test this method I implemented a stochastic optimizer as discussed in Section 4.2.2 subject to the constraints given by (4.3). My implementation functions for all nD/1T and nD/2T environments, using a vector-of-polynomials structure to encode (4.3). The choice of restricting the work to 2T allowed for a number of code optimizations as it was not necessary to implement a complicated wrapping geometry for the $\vec{s}$ portion of the time $(t, \vec{s})$; since $\vec{s}$ is at most single-dimensioned, the use of modular floating point numbers is sufficient.

Active Goal($\mathbb{G} = \{(\vec{t_1}, \vec{p_1}), (\vec{t_2}, \vec{p_2}), \ldots, (\vec{t_k}, \vec{p_k})\}, \vec{t}$)

1. $a = 0$
2. $w = 0$
3. for each $(\vec{t_i}, \vec{p_i}) \in \mathbb{G}$ where ($\vec{t_i}$'s t component) $>$ ($\vec{t}$'s t component):
4. $\quad d_i = \frac{1}{\|\vec{t} - \vec{t_i}\|^2}$
5. $\quad a = a + d_i$
6. $\quad$ if $w < d_i$:
7. $\quad\quad w = d_i$
8. $\quad\quad j = i$
9. if $a \neq 0$:
10. $\quad$ return (reach p_j)$\frac{w}{a}$ + (minimize $\vec{s}$-velocity)$\frac{a-w}{a}$
11. else:
12. $\quad$ return (minimize s-velocity)

Figure 4.3: A heuristic method for deciding on a goal velocity to maximize likelihood of reaching a set of hypertemporal goals, each expressed as a time and the location that ought to be occupied at that time.

An instance of this method with $m = 1$ reduced fully to the 1T formulation presented in Section 4.2. I implemented a test where each agent was given a goal on the opposite side of the mass of other agents, with goals reset once the first set of goals was reached. In both 2D and 3D cases fewer than one collision every ten goal resets occurred. Padding the agent's avoidance radii by the distance necessary to fully reverse velocity given their maximum acceleration reduced this to no collisions after an entire hour of simulation. These results are comparable to those reported by [31, 28, 17, 10, 18, 21].

I also ran tests in nD/2T, for $n \in \{2, 3, 4, 5\}$. Each test consisted of k agents, where k varied from 2 to 5, each with 2–10 goals separated in space and time such that all goals could be reached at half maximum velocity and no two agent's goals overlapped. Agents never collided in any of the test cases, a stronger result than in 1T due in part to the implicit radius padding discussed in Section 4.3.1. For $n > 2$ no goals were ever missed by more than a quarter of the radius of the agents. For $n = 2$ almost half of all goals were missed, with the error in position averaging k times the radius of the agents. This is in keeping with the theoretic implications of

sidling, as discussed in greater detail in Section 5.3, and the resultant errors were reduced to arbitrarily small levels as the agents were given more time between goals.

The nature of this algorithm is intrinsically difficult to reverse; the selection of the next velocity is largely independent of the current velocity so knowledge of what velocity was chosen does not reveal what velocity an agent had before that decision. To permit fully interactive viewing, I chose to run the complete simulation once and store, during the simulation, the position of each agent at each point in time. The initial simulation runs at real-time speeds or nearly so, and after the computation the resulting behavior may be inspected easily, thus satisfying the desire for interactivity in the simulation.

Chapter 5

Exploring Hypertime

Chapters 4 and 5 presented methods for simulating hypertime physics and hypertime crowd navigation. Having implemented these methods and having run many tests with both, I offer several observations on hypertime environments.

I ran three kinds of tests; of these, the first two were designed to demonstrate that the simulations run correctly. Standard software testing demonstrated that the code does not crash and does what it is supposed to mathematically. Additionally, tests were run to verify the criteria of temporality and consistency that

- 1T agents don't collide,
- 1T physics conserves energy and momentum and fully models elastic collision laws,
- all evaluations of system state at a single time vector are the same,
- all elastic collisions are resolved, and
- agents always move toward goals unless such movement would lead to a collision[1].

All of these criteria were satisfied in all tests run.

[1]No rigorous test for this criterion was devised; instead, simulation output was inspected for any obvious violations.

The third kind of tests are both more interesting and harder to describe. Part of the goal of this research is to provide an interactive environment in which to explore hypertime phenomena. As such, I have spent many hours setting up simulations and observing their output, exploring the behavior of particles and agents in the easily-visualized 2D/2T setting. The results of these tests are not pass/fail evaluations or even statistical data; instead, they are observations of trends in simulated hypertime behavior, intended to spur further hypertime research and give a feel for some of the characteristics of hypertime.

5.1 The Interactive Interface

The interface for both simulations is designed to look like the 2T movie player presented in Section 2.1. The window is divided into two pieces, a space plane or viewport and a time plane with a time slider. Although some automated tests were run, the primary interface is based on a simple point-and-click interface; for example, to set up a navigation task the user clicks the initial position of an agent and then the times and locations of each of that agent's goals, then adds the next agent and its goals, etc. After setup is complete both tools simulate agent or particle motion and then allow the user to drag the time slider and observe the state at each time.

Most of the understanding of the simulations is gained through the interactive quality of the viewing experience. As such, static images of the simulations are not helpful, but some time-lapse images and schematics illustrating certain principles are presented later in this section.

Figure 5.1: A time-lapse view of a single straight time path with four particles experiencing four collisions. The time path passed close to two collisions which are clearly visible (between the right-most particle and its two closest neighbors); the other two collisions, being farther in time from the visualized time path, cause the less sudden curving in this image most obviously seen in the trajectory of the leftmost particle.

5.2 Observing 2T Physics

Hypertime physics can be somewhat confusing. I find that it is easiest to understand by selecting straight time paths to view repeatedly, building intuition for what happens along these paths. The general effect is that, unless the time path happens to pass directly through the time point of collision, particles follow curved, roughly hyperbolic paths that "bounce" off of one another at a distance. An example of this is shown in Figure 5.1. Although this effect creates the illusion of some sort of force field, all particles touch at the time the collisions they are involved in are resolved and no other forces act on the particles. However, collisions resolved later in the resolution process can move the particles away from their point of impact and even without this, time paths not passing through the time point of collision do not demonstrate that the colliding particles to touch.

After presenting Theorem 1, I observed this paper does not include a direct measure of entropy, which is based on energy and macroscopic behaviors not yet

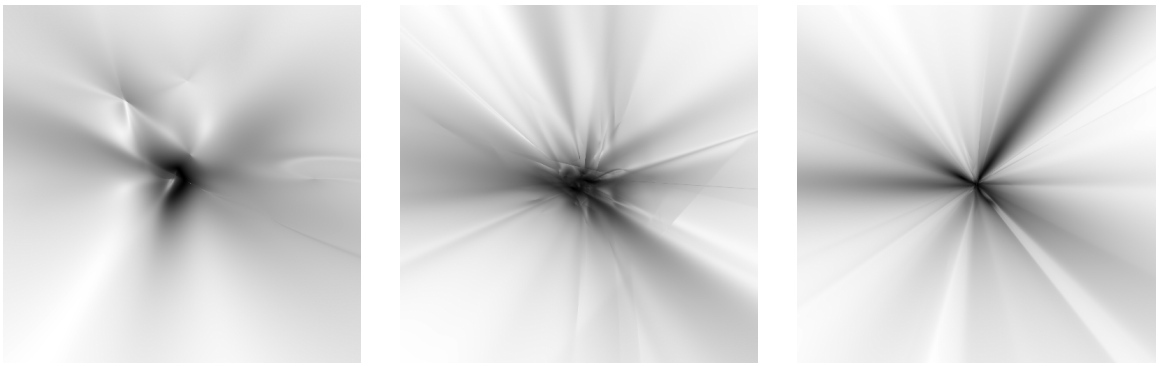


Figure 5.2: An entropy heuristic over the time plane, with the initial time at the center of the image and white meaning particle behavior more like free gas (and presumably at higher entropy) than black. The images were created by randomly seeding (left to right) 5, 10, and 15 particles respectively.

defined in hypertime (see page 9). However, I wanted to get some feel for entropic behavior, even without a firm statistical measure. When simulations begin in an orderly form, it does appear that the simulated situation becomes less orderly when the time separation from that orderly initial condition increases. Given that the highest-entropy state of free particles is free gas expansion (that is, particles moving away from a central location at a speed proportional to their distance from that location), I used the correlation between hypertime particle speed and position as a heuristic approximating entropy. Although this heuristic is inaccurate for low-entropy situations, plots of the heuristic value such as those presented in Figure 5.2 demonstrate that the particles do reach free gas behavior at times far enough from the initial time point.

This tends to support the hypothesis that I am justified in assuming the model of past given by Theorem 1 despite the lack of the continuous entropy metric the theorem itself requires.

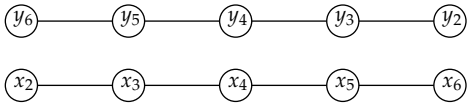
5.3 Observing 2T AI

Although interactive viewing of 2T AI is more intuitive than 2T physics, based largely on our intuitive understanding of the AI task in question, motion tends to be much more complicated than in the physics simulations and I was unable to capture a meaningful time-lapse view. Instead, I offer the following observations.

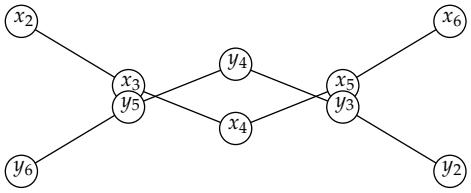
Most of the simulations run much as expected; the agents dodge past one another to reach their goals. However, one unexpected and very common maneuver I have dubbed "sidling", as illustrated in Figure 5.3, where agents that would have collided at one s-value maneuver to change the s-value of collision, but cannot pass one another. This phenomenon is caused by the fact that the simulation has $m \geq n$ (in this case, they are both 2), and so it is possible to have the sort of anti-aligned positions that cause the problem. Simulations in 3D/2T never demonstrated this problem as the agents had an additional space dimension in which to maneuver.

Other than this one quirk (which is not manifest in all simulations) the agents act in a very satisfyingly intelligent way, easily dodging one another and achieving their goals with little difficulty. It does appear, though, that hypertime navigation is fundamentally more complicated than 1T navigation, as the subagents often are unable to take the most obvious course toward their goals due to the mT consistency constraints.

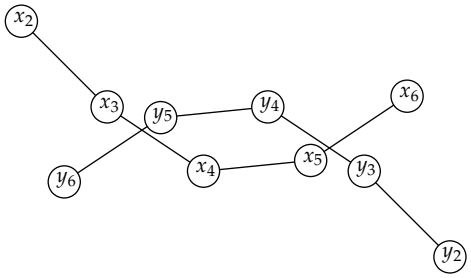
On the whole, my observation from running and observing hypertime AI simulations is that though hypertime differs qualitatively from 1T, it does so in an approachable manner. It is not unreasonable to state that hypertime is generally "like" our standard 1T experience, though complicated by the need to consider, in addition to the future, the impact of any maneuver on a continuum of future "present" states.



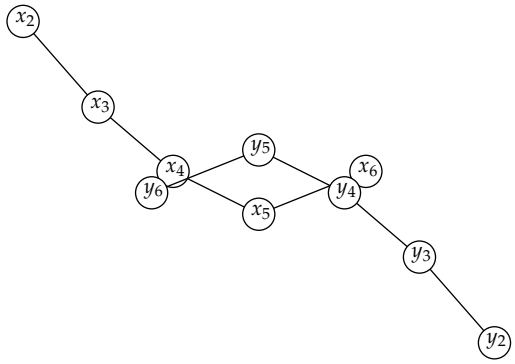
At $t = 1$, agents x and y are depicted over $s \in [2, 6]$. Agent x's goal is to move upward, y's is to move downward.



At $t = 2$, both agents have moved as far in the direction of their respective goals as possible. Only at $s \approx 4$ is there a potential collision; however, the maximum s-velocity constrains the agents from moving further at other s-times as well.



At $t = 3$ agent x has decided to move to the left, agent y to the right, to avoid colliding at $s = 4$. This, however, creates an $s \approx 4.5$ potential collision, requiring the agents to move away from their goals for $s > 4.5$.



At $t = 4$ the leftward motion of x and the rightward motion of y continue, restoring the same situation as at $t = 2$ except now the potential collision is at $s \approx 5$.

Figure 5.3: An example of how hypertime agents "sidle" around one another to avoid collisions. Note that the maneuver only moves the time of potential contact, rather than bypassing it. Sidling only occurs where $m \geq n$ (in this case $m = n = 2$); otherwise, the additional space dimensions are enough to dodge other agents.

Chapter 6

Conclusion

I began by stating several objectives for establishing a framework for hypertime research.

- My original intent was to explore the difference between time and space. Mathematical theories have focused on an ordering principle; by providing an alternate mathematical definition of time as being consistent and temporal, the 2T physics simulation has shown that a visually-consistent temporal environment does not need an obvious ordering in its time. Even so, an ordering appears in the simulation technique itself; whether completely unordered techniques exist remains an open question.
- I intended to demonstrate that, contrary to prevailing opinion, hypertime is possible and basic questions about what hypertime is "like" can be answered. Although such subjective questions into the nature of any environment can never be answered fully[1], the simulations presented, with interactive interfaces and hypertime implementation of laws common to our 1T experience, have made steps toward achieving this goal.
- I intended to define and implement models of hypertime that satisfy an in-

[1] To see this, try giving a definitive answer to the question "what is 3D/1T like?"

tuitive understanding of time, are internally consistent, and form an extensible basis for future research. I have posed these requirements in a formal way and the two simulations presented give a foothold for further hypertime physics and AI research. Further, the interactive visualization paradigm provides an intuitive method for observing and understanding the results of future hypertime work. As an aside, observing and interacting with the results of hypertime simulations has strengthened and clarified my own understanding of hypertime itself and provided much of the intuitive feel for the domain that directed the discussion in this paper.

- I hoped to explore alternate simulation paradigms in place of the standard iterative techniques. Although not known to be original to myself, the method used for the physics simulation does this and, in 1T, would be linear in runtime, possibly allowing it to compare favorably with other techniques. The particular algorithm aside, the fact that the accepted simulation workhorse failed in a domain where an alternate algorithm works demonstrates the value of separating the algorithm from the problem.

Another contribution of this research, beyond the immediate contributions to higher-dimensional computational research, is a demonstration of the usefulness of interactive visualization techniques to understand environments which are both mathematically difficult to define and completely outside the scope of human experience.

Chapter 7

Future Work

Many aspects of my work can be extended. For example, although I have provided the core of algorithms for solving elastic hypertime collisions and obstacle avoidance for any arbitrary values of n and m, I only completed implementations for 2D/2T hypertime. Other potential extensions and improvements include

- deriving a full model of hypertime kinetic energy (see Appendix A for a discussion of some of the issues here) and using it to derive a more accurate model of hypertime elastic collisions;
- attempting some sort of iterative migration of the collision times in the physics simulation in hopes of converging on a solution were all the colliding particles actually touch;
- running large-scale physics simulations and measuring entropy directly;
- implementing additional kinds of simulations; for example
  - hypertime games where the goal is to win in the majority of future times,
  - field laws generated by hypertime carrier particles (gluons, gravitons, etc), or
  - multi-agent cooperative AI or other AI tasks;

- or converting either the physics to cylindrical time or the AI to flat time so both may be included in the same simulated environment.

In addition to these direct hypertime extensions, there are 1T extensions of some of the ideas I used in hypertime. For example, the stochastic method I presented to solve the maneuvering board problem appears to be unusual in its ability to cope with 3+D situations and may allow extensions with more precise goals and constraints.

I am hopeful that, as I have demonstrated that hypertime can be considered, simulated, and visualized, other researchers will discover additional related topics and areas of research that I have never even contemplated.

Bibliography

[1] ARMSTRONG, W. P., AND BURTON, R. P. Perception cues for n dimensions. *Computer Graphics World VIII*, 3 (March 1985), 11–28.

[2] ASGEIRSSON, L. Über eine Mittelwerteigenschaft von Lösungen homogener linearer partieller Differentialgleichungen zweiter Ordnung mit konstanten Koeffizienten. *Mathematische Annalen 113* (1936), 321–346.

[3] BAINES, D. Accelerated ray traced animations exploiting temporal coherence. Master's thesis, Brigham Young University, 2005.

[4] BARS, I. Hidden symmetries, $AdS_D \times S^n$, and the lifting of one-time physics to two-time physics. *Physics Review D 59*, 4 (February 1999), 045019.

[5] BARS, I., AND KOUNNAS, C. Theories with two times. *Physics Review B 402* (1997), 25–32.

[6] BARS, I., AND KUO, Y.-C. Interacting two-time physics field theory with a brst gauge invariant action. *ArXiv High Energy Physics — Theory e-prints* (May 2006).

[7] BRINGHURST, G. L., AND BURTON, R. P. Raytracing in four-dimensional space. *Journal of Imaging Technology 17*, 4 (August/September 1991), 165–167.

[8] BURTON, R. P. Raster algorithms for cartesian hyperspace graphics. *Journal of Imaging Technology XV*, 2 (April 1989), 89–95.

[9] BUSEMANN, H. *Timelike spaces*. Dissertationes mathematicae. Warszawa: Panstwowe Wydawn, Naukowe, 1967.

[10] CHAKRAVARTHY, A., AND GHOSE, D. Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Transactions on Systems, Man and Cybernetics—Part A: Systems and Humans 28*, 5 (September 1998), 562–574.

[11] CLUFF, E., BURTON, R. P., AND BARRETT, W. A. Characterization and categorization of higher dimensional presentation techniques. In *SPIE/SPSE Symposium on Electronic Imaging Science and Technology* (1990), pp. 83–96.

[12] CLUFF, E., BURTON, R. P., AND BARRETT, W. A. A survey and characterization of multidimensional presentation techniques. *Journal of Imaging Technology 17*, 4 (August/September 1991), 142–153.

[13] COURANT, R., AND HILBERT, D. *Methods of Mathematical Physics*. Interscience, New York, 1962.

[14] CURTIS, D. B. The design of a parallel axes graphics software system. Master's thesis, Brigham Young University, April 1986.

[15] CURTIS, D. B., AND BURTON, R. P. Parallel axes graphics using lincoln's log method as an alternative to binocular parallax graphics. In *SPIE/SPSE Symposium on Electronic Imaging Science and Technology* (1990), pp. 172–181.

[16] CURTIS, D. B., BURTON, R. P., AND CAMPBELL, D. M. An alternative to cartesian graphics. *Computer Graphics World X*, 6 (June 1987), 95–98.

[17] FIORINI, P., AND SHILLER, Z. Motion planning in dynamic environments using the relative velocity paradigm. In *IEEE Conference on Robotics and Automation* (1993), pp. 560–565.

[18] FIORINI, P., AND SHILLER, Z. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research 17*, 7 (1998), 760–772.

[19] GOVERNMENT, U. S. *Radar Navigation and Maeuvering Board Manual*, 7th ed. U.S, Government, 2001.

[20] ISAACSON, P. L., BURTON, R. P., AND CAMPBELL, D. M. Presentation of hypothesized 4-d phenomena. *Computer Graphics World VII*, 8 (August 1984), 48–63.

[21] LARGE, F., VASQUEZ, D., FRAICHARD, T., AND LAUGIER, C. Avoiding cars and pedestrians using velocity obstacles and motion prediction. In *IEEE Intelegent Vehicles Symposium* (June 2004), pp. 375–379.

[22] MELVILLE, J. D., AND BURTON, R. P. Piles for hyperdimensional graphics. *Computers and Graphics 21*, 1 (1997), 51–60.

[23] MILLER, F. S., AND EVERETT, A. F. *Instructions for the Use of Martin's Mooring Board and Battenberg's Course Indicator*. Authority of the Lords Commissioners of the Admiralty, 1903.

[24] MOORE, E. H. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society 26* (1920), 394–395.

[25] PENROSE, R. A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society 51* (1955), 406–413.

[26] ROBERTS, L. G. Homogenous matrix representation and manipulation of n-dimensional constructs. *MS-1505* MIT Lincoln Laboratory, 1965.

[27] SHILLER, Z., LARGE, F., AND SKHAVAT, S. Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *IEEE International Conference on Robotics and Automation* (2001), vol. 4, pp. 3716–3721.

[28] SMITH, T. C., EVANS, R., TYCHONIEVICH, L. P., AND MANTEGNA, J. AUV control using geometric constraint-based reasoning. *IEEE Symposium on Autonomous Underwater Vehichle Technology* (1990), 150–155.

[29] TEGMARK, M. On the dimensionality of spacetime. *Classical and Quantum Gravity 14* (1997), L69–L75.

[30] TYCHONIEVICH, L. A. Arrows of time. Unpublished rhyme, December 2003.

[31] TYCHONIEVICH, L. P., ZARET, D., MANTEGNA, J., EVANS, R., MUEHLE, E., AND MARTIN, S. A maneuvering-board approach to path planning with moving obstacles. *Proc. 11th International Joint Conference on Artificial Intelligence (IJCAI)* (1989), 1017–1021.

Appendix A

Comments on Hypertime Energy

A somewhat unexpected result of running the physics simulations presented in this paper is that, while each collision does conserve the 1T energy, the total energy of a 2T system seems to increase with the number of collisions resolved. If more than several dozen collisions are resolved, the end result often has the particles moving so fast (except at the initial time point $\vec{t_0}$, where the specified initial velocities hold) that they are scarcely visible. This implies that perhaps there is a definition of kinetic energy that applies in mT, not just 1T.

The most obvious approach to kinetic energy would be a straightforward extension from 1T:

$$\mathbf{K} = \frac{m}{2}\mathbf{V}^T\mathbf{V}.$$

However, this suffers from a serious drawback in that it implicitly states that the basis of the time vector-space matters. To see this, change the time basis with the unitary m-by-m matrix $\mathbf{U}$; then

$$\vec{t^*} = \mathbf{U}\vec{t} \quad \text{and} \quad \mathbf{V}^* = \mathbf{V}\mathbf{U}^T,$$

which preserves

$$\mathbf{V}\vec{t} = \mathbf{V}^*\vec{t^*} \quad \text{and} \quad \|\vec{t}\| = \|\vec{t^*}\|,$$

as we expect from a change of basis. The new energy, however, is

$$\mathbf{K}^* = \mathbf{V}^{*T}\mathbf{V}^* = \mathbf{U}\mathbf{V}^T\mathbf{V}\mathbf{U}^T = \mathbf{U}\mathbf{K}\mathbf{U}^T,$$

which is not the same as the old.

Another hypertime extension of kinetic energy is

$$\mathbf{K} = \frac{m}{2}\|\mathbf{V}\|_2^2.$$

Like the previous method, this works in 1T; unlike the previous method, it is coordinate-system independent; the 2-norm of a matrix does not change when multiplied by a unitary change-of-basis matrix. However, it suffers from a different drawback in that it would say the following two velocities

$$\mathbf{V}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad \text{and} \qquad \mathbf{V}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

have the same kinetic energy, when intuitively $\mathbf{V}_2$ represents more motion, and hence more kinetic energy, than $\mathbf{V}_1$.

A third alternative is to have energy expressed in terms of the singular values $\vec{\sigma}$ of the velocities. The singular value of velocity $\vec{v}$ in 1T is the single scalar $\sigma = \|\vec{v}\|$, which suggests energy can be expressed as either

$$K = \frac{m}{2}\vec{\sigma}^T\vec{\sigma} \quad \text{or} \quad \vec{K} = \frac{m}{2}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_{\min\{m,n\}}^2)^T$$

Both of these formulations are independent of the chosen time basis and both support the intuitive notion that $\mathbf{V}_2$ has more energy than $\mathbf{V}_1$. The scalar form has all the advantages of scalars, permitting comparison operators and division; the vector formulation, however, preserves additional information about the velocities which may prove useful for some purpose.

Selection of one of these models or derivation of a different model, together with the accompanying extension of energy-based physical laws to hypertime, is beyond the scope of this paper and is left to future work.

Appendix B

Definition of Terms

Consistency is a quality of an algorithm $\hat{P}$ for approximating a mathematical function P. Any consistent algorithm, given unbounded resources of time, space, and internal number representations, can reduce the error $\left\| \hat{P}(\vec{t}) - P(\vec{t}) \right\|$ for any finite $\vec{t}$ in the domain of P to arbitrarily small levels.

Current Times are the $(m-1)$-dimensional set of times for which the entropy is expected to be equal to that at a given reference time.

Hyperspace is any environment defined by a set of objects in n "space" dimensions which change over 1 "time" dimension, where n is greater than 3. Hyperspace is also denoted nD and should not be confused with the idea of faster-than-light travel.

Hypertime is any environment defined by a set of objects in n "space" dimensions which change over m "time" dimensions in a temporal way (see Definition 1 on page 14), where both n and m are greater than one. Hypertime must also be intrinsically hypertemporal; there should not be an equivalent formulation with only one time dimension (or only one space dimension). Hypertime is also denoted nD/mT and should not be confused with the idea of a "parallel dimension."

Jacobian Matrix is the result of the full derivative of a vector valued function. Given $\vec{x} = (x_1, \ldots, x_n)^T$ and $\vec{t} = (t_1, \ldots, t_m)^T$,

$$\frac{d}{d\vec{t}}\vec{x}(\vec{t}) = \begin{bmatrix} \frac{\partial x_1}{\partial t_1} & \cdots & \frac{\partial x_1}{\partial t_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_n}{\partial t_1} & \cdots & \frac{\partial x_n}{\partial t_m} \end{bmatrix}$$

Matrix Induced Norm given a vector norm $\|\cdot\|_p$, is defined as

$$\|\mathbf{A}\|_p \equiv \max_{\|\vec{x}\|_p=1} \|\mathbf{A}\vec{x}\|_p.$$

For the ℓ_2 vector norm, it can be shown that the value of the induced norm of $\mathbf{A}$ is the largest singular value of $\mathbf{A}$; that is, $\|\mathbf{A}\|_2 = \sigma_1$.

Moore-Penrose Pseudo-inverse of any matrix $\mathbf{A}$, denoted $\mathbf{A}^+$, is defined by[1]

$$\mathbf{A}^+ = \lim_{\delta\to 0}(\mathbf{A}'\mathbf{A} + \delta\mathbf{I})^{-1}\mathbf{A}' = \lim_{\delta\to 0}\mathbf{A}'(\mathbf{A}\mathbf{A}' + \delta\mathbf{I})^{-1}.$$

When a matrix is invertible, it's inverse is equal to it's pseudo-inverse. The pseudo-inverse solves the least-squares problem, meaning that $\mathbf{A}^+\vec{b}$ gives the shortest $\vec{x}$ such that $\mathbf{A}\vec{x} - \vec{b}$ is as small as the null spaces of $\mathbf{A}$ will allow.

Moment is an ambiguous term referring either to a time point or a set of current times. The unrelated definition from kinetics relating to angular inertia is not used in this paper.

Past is the set of times for which the entropy is expected to be less than at a given reference time.

Past Time Arrow denoted $\vec{t_p}$, a vector pointing in the direction of maximal expected entropic decrease; it is perpendicular to the current times.

PDE is short for **p**artial **d**ifferential **e**quation. A PDE is an equation where some derivative of a function of several variables is expressed in terms of it's lesser derivatives. Not all PDEs have solutions. Interested readers are referred to [13] or any college PDE textbook for more information.

Precision is a quality of a realization of an algorithm $\hat{P}$ for approximating a mathematical function P. Given the same bounds on resources available to the algorithm, a more precise algorithm has smaller error $\left\|\hat{P}(\vec{t}) - P(\vec{t})\right\|$ than does a less precise simulation of P. An algorithm implementation is considered **sufficiently precise** (for some purpose) if the outcome of simulated phenomena is close enough to the outcome of real phenomena that the results of the simulation can be used in place of empirical testing (for that purpose).

Singular Value Decomposition of any p-by-q matrix $\mathbf{A}$ is a factorization $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}'$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary p-by$p$ and q-by-q matrices, respectively,

[1] Here $\mathbf{A}'$ denotes to the conjugate transpose of $\mathbf{A}$.

and $\mathbf{\Sigma}$ is a p-by$q$ matrix where $\mathbf{\Sigma}_{ii} = \sigma_i$, with $\sigma_i \geq \sigma_{i+1} \geq 0$, and $\mathbf{\Sigma}_{ij} = 0$ for all $i \neq j$. The matrix $\mathbf{\Sigma}$, and thus the **singular values** $\vec{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_{\min\{p,q\}})^T$, is uniquely determined by the singular value decomposition, though the matrices $\mathbf{U}$ and $\mathbf{V}$ are not.

Time-path is a continuous sequence of points in time.

Time Point is a particular vector in $\mathbb{R}^m$ specifying a zero-dimensional point in time.

Ultra-hyperbolic PDEs belong to a class of PDEs for which solution techniques are not generally known. Interested readers are referred to [2] for a more technical discussion of ultra-hyperbolic PDEs.

Vector Norms are denoted throughout this paper as $\|\cdot\|$ (e.g., the norm of a vector $\vec{x}$ would be $\|\vec{x}\|$) and are functions that take a vector and return a non-negative real value such that

- $\|\vec{x}\| = 0$ if and only if $\vec{x}$ is the zero vector,
- $\|k\vec{x}\| = |k|\|\vec{x}\|$ for any scalar k, and
- $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$.

This work assumes the ℓ_2 norm, also called the Euclidean norm, defined by

$$\|\vec{x}\|_2 = \|\vec{x}\| = \sqrt{\vec{x}'\vec{x}}$$

where $\vec{x}'$ denoted the conjugate transpose of $\vec{x}$. For the most part this assumption is not necessary, but certain algorithms (e.g., the use of the pseudo-inverse) implicitly assume the ℓ_2 norm and would have to be replaced by semi-definite programs or other minimizing techniques if other norms were desired.