University of Alberta

Real-Time Time-Warped Multiscale Signal Processing for Scientific Visualization

by

Matthew Hamilton

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

"Equally important is the necessity to animate, personalize scientific data. All that we know about the laws of nature, about atomic and sub-atomic events is mediated by the nervous system. All science is neuro-ecology. All our observations of the universe are neurological events. The brain is the recording instrument. Instead of forcing nature to fit the three dimensional model of our larengyal-manual symbolic mind, we must allow our nervous systems to be imprinted by the raw data– learn to think-experience like DNA, like electrons, like sub-atomic particles"

–TImothy Leary

Abstract

This thesis considers the problem of visualizing simulations of phenomenon which span large ranges of spatial scales. These datasets tend to be extremely large presenting challenges both to human comprehension and high-performance computing. The main problems considered are how to effectively represent scale and how to efficiently compute and visualize multiscale representations for large, real-time datasets. Time-warped signal processing techniques are shown to be useful for formulating a localized notion of scale. In this case, we use time-warping in order to adapt the standard Fourier basis to local properties of the signal, giving the advantage of being localized in the frequency spectrum as compared with the standard linear notions of scale. Time-warping is also shown to have theoretical advantages in terms of signal reconstruction quality and random noise removal. In practice, these advantages are shown to only hold under certain conditions. It is then shown in the thesis how convolution-based reconstruction techniques can be mapped onto graphics processing units (GPUs) for high-performance implementation of a multiscale molecular visualization framework. We show how the same technique can likely be used for time-warped multiscale reconstruction.

Acknowledgements

I would like to thank sincerely my supervisor Pierre Boulanger for his guidance and support during the research and preparation of this thesis. Early on, I spent time working with the Institute for Biomolecular Design (IBD) and received great guidance and support from both Mike Ellison and Doug Ridgway and they also are due great thanks. I would also like to thank Phil Bording for much support during the thesis as well as many lessons in high-performance computing. My undergraduate thesis advisor Todd Wareham also deserves a great deal of thanks for encouraging my pursuit of graduate school and always being supportive.

I would like to thank all my fellow members of the AMMI lab for making grad school more enjoyable. I must particularly thank Victor Ochoa Mayorga for his help in understanding many of the concepts that were required for this thesis and many pleasant related conversations.

I give thanks to the staff of Computing Science department for plenty of assistance of various sorts over the years. I also owe a big thank you to NSERC, iCORE and Alberta Ingenuity for financial support over the years that made this work possible.

Finally, I give thanks to my family and friends for all the support over the years.

Table of Contents

List of Tables

List of Figures

... ...

Chapter 1

Introduction

1.1 General Motivation

Computer simulation has long been recognized as a powerful tool in the physical sciences. Well-validated simulation models allow for affordable quantitative study of phenomena normally inaccessible through conventional experimental techniques. One need only look to mature modelling codes in atomic-level molecular dynamics to appreciate the value of this approach to science [65, 62].

It is well-known that interesting physical phenomenon span vast ranges of spatiotemporal scales. Likewise, simulations of such physical phenomenon must represent this vast ranges of scales if they are to serve as accurate predictive models. As computing capability continues to advance, simulations of more complex phenomena over larger spatiotemporal scales are becoming feasible.

However, many challenges still remain. Most important simulation codes are still many orders of magnitude short of the speed necessary to becoming high-throughput predicative tools. Even when large scale simulation runs are feasible, it is a massive challenge to interpret such experiments in a qualitative way. In order to understand phenomenon across span large ranges of spatiotemporal scale, an effective means to represent the simulation data at multiple scales is required.

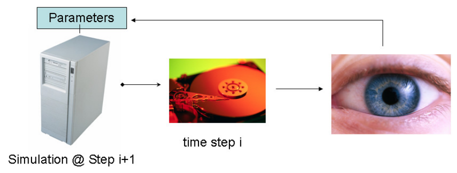
Finding insight in the data produced by simulation is a significant problem on several levels. Modelling is a delicate art involving the interaction of many components, from mathematical model formulation, experimental calibration and realization in code all the way to result interpretation. Human feedback and judgement are required at each of these stages of model engineering. A closed loop pipeline model emerges as a more suitable workflow model (Figure 1.1), commonly known as the computational steering model.

True insight into physical phenomena cannot be achieved until the model itself is validated. The capability to interactively steer parameters of a model with immediate feedback fuses the user and simulation, enabling one to explore how model components interact. This serves many purposes, enabling interactive debugging at all levels (*e.g.* theoretical, numerical, and software) of implementation while allowing accurate, validated computational simulations to serve as powerful tools in understanding physical process control.

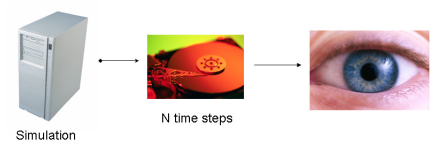
For this workflow model to succeed, effective human-simulation interfaces are needed. The field of *scientific visualization* has proposed to solve the problem of large dataset interpretation by visually-encoding data. Though research in computer graphics has produced techniques to generate many impressive visual effects, these do not necessarily help design visual mappings.

As a further challenge, the need for effective visual analysis shifts the supercomputing burden towards data interpretation. Rendering algorithms must be able to deliver real-time performance that scales with large amounts of simulaton data. Recently, price-competitive parallel computer architectures have emerged, providing orders of magnitude improvements over standard desktop CPUs. Among the most promising of these technologies are accelerated graphics hardware architectures. Effectively mapping useful visualization rendering algorithms poses challenges as programming paradigms shuffle to this new hardware.

These challenges are especially relevant in terms of spatiotemporal scale and effective steering of simulations. Vast amounts of data across many spatiotemporal scales threatens to overwhelm the user. This makes it difficult for intuitive understanding and subsequent steering of a simulation. An effective visualization strategy which allows the user to filter based on the desired scale would aid in achieving insight into the data. However, it is a complex matter to decide how to represent and filter based on scale. Moreover, we must balance any representational considerations with the need to deliver scalable and interactive performance in line with the goal of providing a tool for computational steering.



(a) Steering/Tracking, ”human-in-the-loop”



(b) Batch

Figure 1.1: Two computation-visualization workflow models

1.2 Thesis Overview

In this thesis, we explore the use of time-warped signal processing in terms of Gaussian scale space theory. We prove that time-warping in combination with scale space filtering preserves the causality property for the resulting scale space. We further show specifically how time-warping allows the underlying Fourier basis functions to adapt to the inherent non-stationarity of geometric signals. This allows for better signal localization and thus a notion of scale that adapts to the local properties of a signal.

Our hypothesis is: *A basis that adapts to a signal ought to be better correlated with the signal*. Because of these adaptive properties, time-warping holds promise to allow for improved reconstruction and noise removal for random noise. We are able to show sufficient conditions for a warping function that provides guaranteed theoretical improvements of reconstruction error and noise removal relative to the unwarped counterparts. We explore how this theory holds up in practice and discover some cases where it does not.

We next consider practical applications of multiscale representations. Physical processes across different scales are often described by separate physical laws of distinct character. Interesting physical processes often involve interactions across several spatio-temporal scales (*e.g.* turbulence).

We consider, for example, the relevance of spatiotemporal scale for a molecular dynamics simulations of protein folding. The protein backbone is essentially a one-dimensional structure embedded in three dimensions whose configuration changes over the length of the folding process. At the smallest spatiotemporal scales exists only noise, which can be visualized as minor wiggling due to essentially random processes. At longer time scales, this backbone starts to form into a folded structure. The folding process can be viewed as series of gradual folding steps. The protein backbone itself folds itself into medium scale structures (known as secondary structure) along contiguous subsegments of the backbone. At the largest scales, these medium scale structures arrange relative to each other, often called the tertiary structure [43]. In the case of a protein, these structures at various scales influence how other molecules may physically interact or bind with the protein.

In other physical problems, there are similar separations across scales where various sub-ranges become important. In terms of subsurface seismic wave data, domes and basins tend to be represented within the broadest of scales while faults, fractures, and sampling noise tend to occupy a distinct, finer range of scale [2]. In these examples and others, it is useful for a user to be able to filter out ranges of scales that are of less interest in order to clarify and focus a visualization to the relevant aspects of the data. Hence, a way to represent data at multiple scales that can be computed efficiently and interactively even for large datasets is required.

To this end, we present the implementation of a high-performance, GPU-based multiscale visualization framework for a time-varying molecular dynamics datasets. The resulting rendering framework allows the interactive selection of scale in terms of a time-varying molecular dynamics molecular surface visualization. This approach aims to minimize memory-bandwidth performance issues by avoiding excessive re-sampling steps and carefully utilizing software-managed cache present on modern GPUs. This allows us to provide real-time, computational steering capability to computational scientists looking to explore phenomenon across multiple scales.

To summarize, we list the contributions to be presented in the following thesis:

- proof of casuality of time-warped scale space
- local spectral interpretation of time-warped scale space filter
- establish theoretical proof of sufficient conditions for improved error bounds for time-warped convolution reconstruction and
- establish experimental validation of these improved error metrics and also conditions where the theory fails
- establish theoretical proof of sufficient conditions for method noise superiority of time-warped Gaussian noise removal
- high-performance, real-time GPU implementation of a scale-adjustable molecular surface for time-varying protein folding data

Chapter 2

Related Work

2.1 Computational Steering

2.1.1 Computational Steering Model

Simulation Modeling

Consider a physical process model which produces spatial data in time, where a particular time step at time t parameterized by a specification S can be written:

$$M(S,t). \tag{2.1}$$

S can encompass any aspects of the simulation-compute module of the system. This includes aspects at the software level and on up to theoretical model formulations. Specifically, we might consider S as being made up of the theoretical mathematical model of the process being modeled, numerical discretization/approximation scheme, and an implemented code running on a physical machine. Each of these components may be controlled or tweaked through an assortment of parameters which can have various effects on the efficiency, validity, accuracy, *etc.* of the modeling process.

Visualization

Frequently the numerical information produced by a simulation model $M(S,t)$ is evaluated and analyzed visually. We describe the underlying process more formally borrowing and extending notation from van Wijk's visualization model [67]. We may view the central process of visualization as:

$$I(t) = V(M(S_m,t),S_v,t) \tag{2.2}$$

where data from model $M(S_m,t)$ is mapped to a time varying image $I(t)$ according to parameters specified by S_v. When perceived by a user, such an image I contains information that conveys knowledge, whose increase can be modeled by:

$$\frac{dK}{dt} = P(I,K). \tag{2.3}$$

where the function P captures the knowledge K gained. Knowledge gained depends then on the perceptual-cognitive ability of a user as well as their current knowledge. For example, domain experts may be better able to extract meaning from a data image. Formally, we may express the current knowledge as

$$K(t) = K_0 + \int_0^t P(I,K,t)dt. \tag{2.4}$$

Computational Steering

We may now view steering as an interactive exploration process (denoted by E), where the user modifies both the simulation (S_m) and visualization (S_v) specifications based on their current knowledge. Supposing that $S = (S_m, S_v)$, this may be viewed as:

$$\frac{dS}{dt} = E(K) \tag{2.5}$$

where the current specification at a time is

$$S(t) = S_0 + \int_0^t E(K)dt. \tag{2.6}$$

We may extend this framework further such that the image mapping V can also depend on user knowledge. A good example would to use machine learning [38] to modify V by learning about the user's knowledge during an extended exploration interval. One can formally account for this by having V also as a function of K or some K‘ that approximates or is subsumed by K.

One may also consider the idea of solving inverse simulation problems [24]. Instead of specifying a physical state and viewing the simulated consequences, an inverse problem has a desired destination state specified by the user. Such a problem can frequently be expressed as a theoretically difficult (under-constrained) optimization problem, however in some cases it might become tractable under the guidance of interactive human-level domain knowledge and subsequent steering . At any rate, such a process would still require efficient computation of the corresponding forward problem.

2.2 High-Performance Parallel Architecture

Providing a computational steering capability for simulations of important real-world problems requires immense supercomputing capability. Though parallelization has always been an important aspect of supercomputing, for many years, computing capabilities improved steadily by increasing single thread execution speed. This was achieved through longer pipelines, non-deterministic execution models, more transitors per square area and steady increases in clock speed. Improvements in these factors provided performance increases for most applications, including inherently serial computations. However, physical limits like heat dissipation and quantum effects were compounded with the fact that main memory access time has remained relatively constant (known as the memory wall) [75], casting extreme doubt that significant improvements can continue along these lines.

In light of this reality, recently processors have increased their compute power by utilizing greater explicit parallelism [21, 48]. The philosophy adopted has been to avoid wasting die space to squeeze out extra single thread performance and instead place many simpler deterministic processors on a single die together. Under this regime, processors may attain many orders of magnitude greater theoretical arithmetic performance than conventional scalar processors and with better power efficiency. Furthermore, effective memory bandwidth can be increased as parallel threads are able to access separate memory banks simultaneously. In 2012, the newest graphics processor architectures serve as a good example, being able to achieve 3090.4 GFlops on a single chip with peak memory bandwidths of 192.256 Gb/s versus 10 or 20GFlops alongside peak memory bandwidth of 20Gb/s on the latest AMD or Intel chip offering [1].

Though increasing parallelism does afford performance gains, it is not a silver bullet. Inherently serial, single-threaded processes will not benefit and in fact run slower as light weight, deterministic threads. Moreover, speed improvements tend to only approach the theoretical peak for inherently parallel problems with large compute to memory-transfer ratios. Despite these issues, there is reason for optimism. Traditional high-performance computing sub-fields such as scientific simulation, computer graphics, *etc.* all contain significant inherently parallel problems that can achieve high-thoughput on these emerging parallel architectures. There is reason for excitement as some of these important computing sub-fields can look forward to a regime of speed growth that will exceed the traditional Moore's law curve.

As mentioned previously, the main issue for memory intensive problems in optimally targeting an application to a given architecture is deciding how to maximize effective memory bandwidth, given the simple fact that DRAM memory access and clock cycle times differ by a factor of at least 100 in modern chips. Relatively early on, Fatahalian *et. al* [17] noted that even though early GPUs could outperform top CPUs at theoretical arithmetic throughput, in even basic practical cases they were memory bandwidth limited when performing non-graphic computations. The next generation of GPUs showed some improvement in this regard. Govindaraju *et. al*

[19] explored how the two-dimensional coherence behavior of GPU caches could benefit a range of high-performance applications, achieving several fold speed improvement over well-optimized CPU implementations. High-performance tuning efforts on other architectures invariably spend a great deal of time focusing on the same problem: how to layout data in memory in order to maximize cache hit rates and thus improving effective memory throughput for a particular algorithm [76].

Despite disappointing performance of main memory for less arithmetic-intensive applications, there are several strategies for maximizing memory throughput. For many applications, memory access locations are often fixed or easily determined and not strictly dependent on the results of computation. In these cases, prefetching can hide latency, allowing one to overlap computation with memory access.

In order for prefetching to work effectively, one needs sufficient cache to store prefetched data. Recently, several architectures have provided software-managed caches on die. There appear to be two main advantages to a software-managed cache: ability to implement an arbitrary (1) prefetching scheme and (2) replacement strategy based on application-specific criteria.

Mark and Fussel [39] explain the difference between hardware-managed cache and software-managed scratchpads:

“The difference between these two approaches is fundamental. For a cache, the decision as to which elements of data should be stored on chip is automatically made by the hardware at run-time, with the decision typically made at a fine granularity (e.g. blocks of 32 bytes). With a software-managed scratchpad memory, the decision as to which data should be stored on chip is made either at compile time or made explicitly by software at runtime, usually at a coarser granularity.”

Though many alternate parallel architectures have appeared recently, two in particular provide a user-managed scratch pad memory on-die. The Cell processor [21] is a heterogeneous parallel architecture composed of a single more conventional processing core (called a *power processing element* or PPE) having hardware-managed cache along with 8 identical streaming SIMD cores (called *synergistic processing elements* or SPEs), each containing 256kB of fast, local store memory. The NVIDIA G80 GPU architecture is composed of many homogeneous scalar processors, with smaller groups of processors each sharing a fast 16kB local memory. In both cases, reading from local store memory is hundreds of times faster than reading from off-chip memory.

As an alternative to the bandwidth-limited memory architectures, application-specific circuits are often considered for specific HPC applications. Purpose-built computers for elastic wave propagation for seismic modeling have been proposed [4, 3]. Anton, a purpose-built computer for solving molecular dynamics problems [57] has also recently appeared. In both cases, the solutions avoid the use of high-latency, low-bandwidth DRAM, instead focusing on closely-coupled parallelism combined with large buses to static memory. Though these efforts appear promising, for the average researcher purpose-built computers would be financially infeasible, whereas high-end graphic cards only cost thousands of dollars and come equipped with reasonable software development tools.

2.3 Scientific Visualization for Sample-based Data

A computational steering system, in addition to supercomputing capability, requires some means to analyze the resulting data. Visualization is often employed as a useful tool for this purpose. Assuming that simulation data can be generated very rapidly and at an interactive rate, one requires the visualization method to be able to generate images just as rapidly. Moreover, the method must allow the user to spot interesting patterns or anomolies in the data. Let us now consider some visualization methods for point or sample-based data.

2.3.1 Glyph Rendering Samples

Sample-based data can be rendered very efficiently using standard graphic library routines coupled with the latest GPU-accelerated graphics pipelines. This can be an effective way of viewing sampled data in order to understand its meaning.

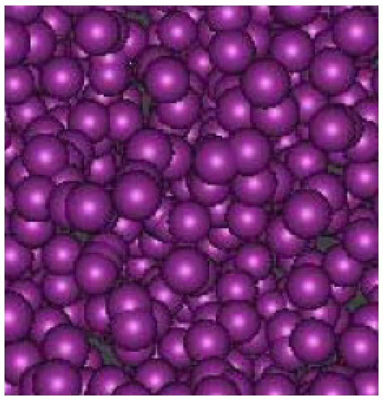
In an effort to improve on this result, it has also been proposed to use sphere-based primitives to render points. Here the advantage is that a sphere radius can be used to encode an additional scalar field associated with the underlying data point. More complex primitives such as ellipsoid can further encode additional dimensions of information. However, efficiently rendering large numbers of these primitives presents a challenge. Though a single point and scalar pair can encode sphere of specified radius, a tessellated sphere would require many triangle rendering primitives for high quality, hence presenting a data size explosion problem.

This problem can be overcome using programmable pixel shadrer hardware to render non-triangle primitives–effectively bypassing the *de facto* polygon rasterizers of modern graphics cards [59, 22, 61, 63, 52]. For example, general quadratic surfaces such as spheres, ellipsoids and cylinders have been sucessfully implemented without the need for polygonal representations [59, 61].

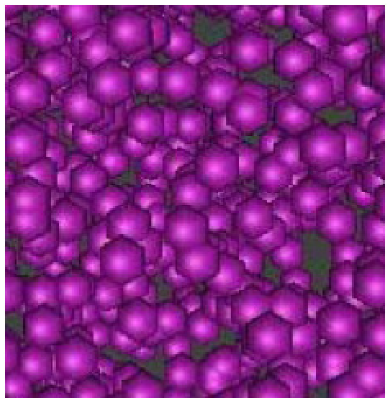
The main advantages of this sort of primitive are that they require less geometry per primitive, thus alleviating CPU-to-bus and dynamic memory bandwidth bottlenecks, thus allowing more complex scenes to be rendered at a high speed. Furthermore, GPU-primitives automatically adjust to the zoom factor, unlike traditional tesselated surface approaches (Notice the ”piecewise-linear” artifacts in Figure 2.1(b)).

2.3.2 Surface Reconstruction and Rendering

Rather than rendering and visualizing discrete samples as separate discrete objects, they can be interpreted as samples of an underlying continuous function. Recon-



(a) Smooth, closed-form rendering technique



(b) Piece-wise linear reconstruction artifacts

Figure 2.1: Effect of various GPU-based rendering strategies

structing a surface from a given set of samples can improve perception of large scale shape [37].

Volume rendering is one approach to reconstructing and rendering surfaces. It is in fact more general, allowing rendering of higher-dimensional scalar fields. To produce the rendered image, one can compute the volume rendering integral over a scalar field [73]. Given a scalar function $f(\mathbf{x})$ at position $\mathbf{x}$ with a ray direction vector $\mathbf{s}$ and an opacity function α, one can compute this integral as:

$$C(\mathbf{x},\mathbf{s}) = \int_{t_0}^{t_1} f(\mathbf{x}+t\mathbf{s}) e^{-\int_{t_o}^{t} \alpha(\mathbf{x}+t\mu)d\mu} dt. \tag{2.7}$$

In practical terms, one can compute this integral for each ray associated with the resulting image plane I to assign a final color, intensity value. This ray-casting model is often used in concert with a simple lighting model [41]. Hence, typically the associated gradient field is also required to be reconstructed from the given samples.

Since practical volume rendering involves fairly large datasets to compute the volume rendering integral, achieving good performance is critical. This is especially important for the case of visualization of dynamic, real-time data, as the costs of one-time preprocessing to achieve rendering speed-up no longer amortize as they do for static data.

From the perspective of a ray-casting based numerical solution to the volume rendering integal, the volume rendering problem can be viewed as essentially having two parts: *(i)* re-sampling and *(ii)* compositing. Re-sampling or reconstruction involves some form of interpolation in order to reconstruct the function $f(\mathbf{x})$ from a discrete representation of the function. The field is re-sampled along the cast rays, then composited (in order) along this ray according to a discretization of the volume rendering integral (See Figure 2.2 and Equation 2.7).

Existing fast, accurate volume rendering methods (for both scattered and regular data) can be classified as either (*i*) slice-based, implicitly ray-casting or (*ii*) explictly ray-casting. These two categories differ in the order in which they perform all of the required re-sampling and subsequent ray-compositing steps.

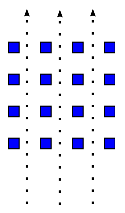


Figure 2.2: $f(\mathbf{x})$, given as a uniform is re-sampled onto a ray-determined grid (dashed lines) in order to discretely compute the volume rendering integral (Equation 2.7).

Performance Aspects

Slice-based approaches re-sample the volume data as a set of view-aligned, planar slices and exploit fast texture and trilinear interpolation capabilities of GPUs [42] to achieve high-performance. Memory access order follows the front-to-back, slice-by-slice form of this method. Hence, temporal locality and the relative predictability of data access along ray-paths is not exploited. Compositing is performed in the same manner, with intermediate computations accumulated in the slow, dynamic framebuffer memory.

In contrast, an explicit ray-casting approach allows finer control of memory access [60], allowing accumulation operations to be performed exclusively in registers, instead of requiring successive reads and writes into slow dynamic memory of the framebuffer.

Recently, GPU-based ray-casting volume renderers for uniform or regular grid data have become fast. Ray-casting volume renderers [60] can overcome inherent inflexibility and inefficiency in slice-based approaches and appear to largely overcome SIMD branch-divergence performance issues [60, 35] on the newest fine-threaded graphics architectures. Moderate-sized volumes can be rendered an order of magnitude faster than what is required for interactive viewing.

An alternate approach that can also be considered as explicitly ray-casting considers ray computation in the frequency domain [36]. This technique requires a Fourier transform pre-processing step in order to compute the frequency domain representation of the given input samples. If the given input samples contain only low-frequency information then fewer samples are required to represent **s** (Equation 2.7) in the frequency domain. The ray-compositing computation has an elegant interpretation in the frequency domain, making this part of computation efficient. However, it cannot accomodate for an arbitrary α (See Equation 2.7), though extensions to more flexible transfer functions have been explored [10].

Slightly different implementation strategies are needed for irregularly-gridded data [29]. Nonetheless, tetrahedral mesh volume rendering with ray-casting [71] implementations achieve real-time performance with good image quality. Volume rendering for ungridded, scattered point datasets has also been explored and is commonly known as splatting [72]. The chief problem with this ungridded data is that spatial neighborhood relationships are not explicitly represented as they are for gridded data. Reconstruction and ray-intersection computations both require efficient access to neighboring samples, hence a spatial data structure is required [47].

Since gridded data volume rendering implementations are mature, one simple way around this problem is to simply re-sample data onto a regular grid [49]. Though this approach makes sense for a single static dataset when used as a one-time preprocessing step, for dynamic data it is slow since re-sampling alone is expensive in terms of memory bandwidth. Moreover, even for the static case, re-sampling in practice can introduce various distortions or undesirable artifacts. Re-sampling can be avoided by meshing the scattered points and falling back on standard regular grid based rendering techniques [71]. However, constructing good meshes is computationally challenging in terms of handling dynamic data sets and the implied linear reconstruction of the grid is limited in terms of rendering quality.

Rendering approaches that reconstruct surfaces using scattered sample data directly also exist. Several efforts propose to solve this rendering problem by focusing on display of isosurfaces of the underlying density field. Pointwise subsampling of the desired isosurface [53, 12, 55, 66] has been used, with the resulting points rendered using 2D splats [5]. However this approach requires far too many isosurface splats if the initial data is large. Mueller *et. al* [45] present two approaches. One requires a preprocessing step to store neighbor information for basis function centres which limits its ability to handle time-varying data and moreover lead to expensive texture fetches not well-suited to achieving high cache hit rate. They achieve better performance by using a depth-peeling approach requiring many rendering passes and thus many slow main memory accesses.

Corrigan *et. al* [13] compute point-based volume rendering using a Fourier domain approach. This differs from uniform data Fourier methods in that an analytic frequency domain representation is computed directly from the given point samples. In terms of performance, this technique enjoys similar advantages to Fourier methods for gridded data: better performance for low frequency sampling rates on sufficiently bandlimited data or extremely zoomed-in views. However as with gridded data, arbitrary transfer functions cannot yet be supported with this technique.

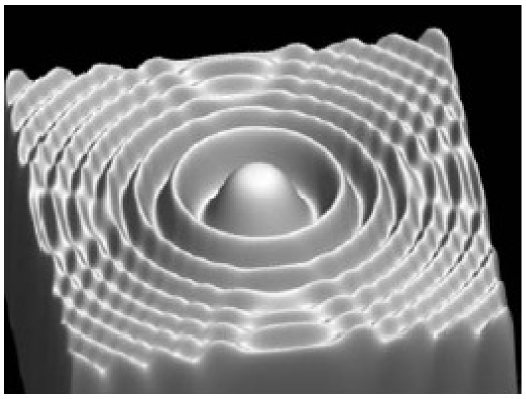
Spatially-based direct volume rendering offers greater flexibility than Fourier methods or isosurface-only rendering techniques for scattered data. Early direct approaches to splatting [26, 25] used “gather”-based techniques, retrieving each basis centre from memory at each ray sample point, leading to incoherent memory access patterns and poor cache reuse. This produced real-time performance on only tiny datasets. Later approaches used a “scatter” operation in order to exploit data re-use from cache, achieving faster overall performance [47, 46].

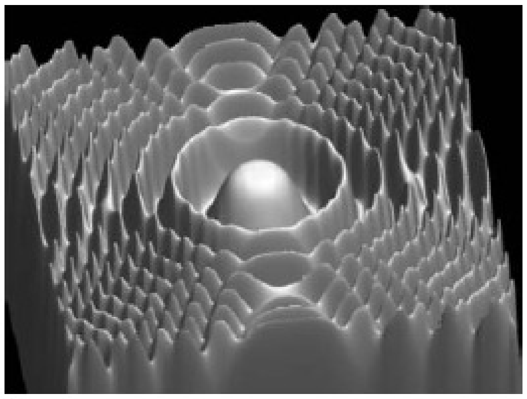
Though successful at achieving real-time performance with high visual quality, these implementations cannot handle time-varying data while still maintaining a real-time rate. The main obstacle is the data structure for spatial partitioning used to sort particles into image-aligned sheet buffers, maintained on the CPU [47, 46]. Initially, implementations of GPU-based spatial structures were burdened by inflexible, shader-based implementation without an explicit scatter capability and required elaborate workarounds [51]. With the advent of a flexible scatter capability and atomic memory operations in newer GPU cards [1], this part of the computation can be accelerated on the GPU fairly simply.

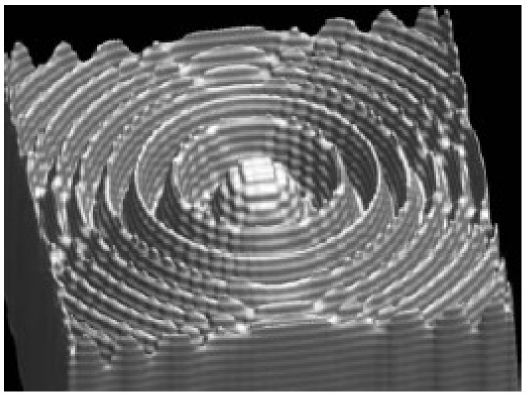
Reconstruction for Volume Rendering

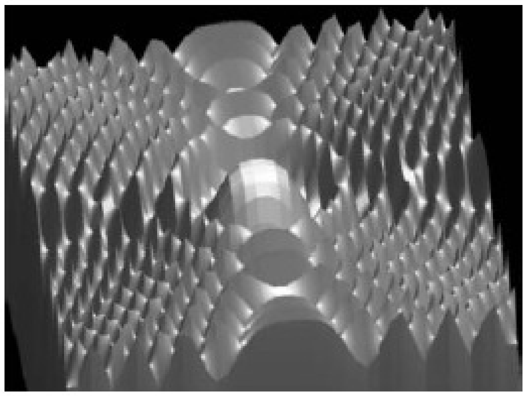
In this section, we will review reconstruction techniques from discrete samples required for volume rendering. Various techniques present trade-offs between quality and performance, for example specifically the ability to handle dynamic, real-time data.

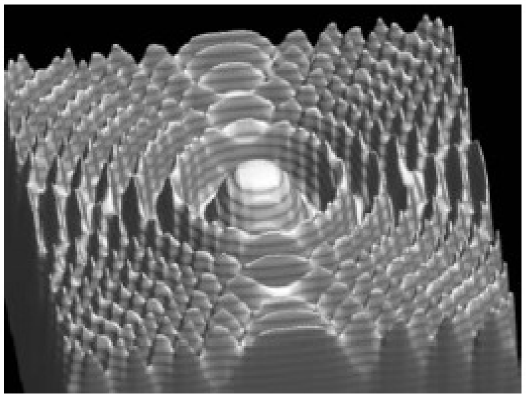
Given a set of function-value samples, the problem to reconstruct the original function from these samples is inherently under-contrained or ill-posed. Theories of function reconstruction must therefore be based on some underlying prior assumptions about the function in order to make it well-posed. In addition to reconstruction


(a) B-spline


(b) Catmull-Rom


(c) Cubic ($B = 0.5, C = 0.85$)


(d) Trilinear


(e) Cubic ($B = 0.26, C = 0.1$)

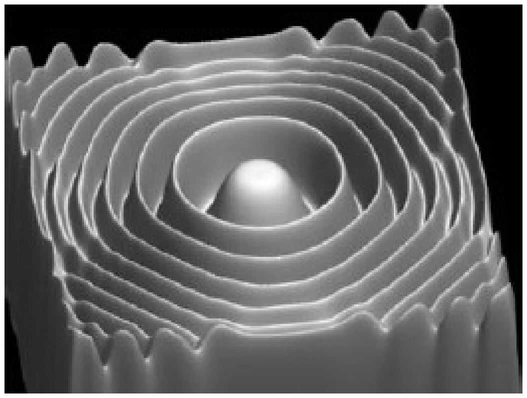

(f) Windowed sinc ($r = 4.8$)

Figure 2.3: Marscher and Lobb's results using various reconstruction filters on their challenging test function [40].

of a function from samples, implementation of lighting models for volume rendering also requires reconstruction of the underlying gradient field.

Convolution reconstruction is most commonly used in standard, real-time volume rendering implementations [47, 60]. Theoretically, this reconstruction is based on an orthogonal projection of a finite-energy function onto the space of bandlimited functions [64]. Sample values are the coefficients on the corresponding *sinc* basis functions. A bandlimited function can be reconstructed from these samples and the corresponding *sinc* basis. More concretely:

Theorem 1 ***Shannon's sampling theorem [56]:*** *Consider a finite-energy, Ω-bandlimited function f (i.e. there is $\Omega > 0$ such that $\hat{f}(w) = 0$ for all $|w| > \Omega$). Then f is determined by a set of regular samples $\{f(kT)|k \in \mathbb{Z}\}$ for $T = 1/2\Omega$, i.e. $f(x) = \Sigma_{k\in\mathbb{Z}} f(kT) sinc(x/T - k)$.*

This theorem essentially states that a bandlimited function can be reconstructed from a set of regularly-spaced samples whose frequency is twice the rate of the frequency upper bound Ω of the function.

In practice, this assumption fails to varying degrees. Many real-world signals are, in fact, not bandlimited. Therefore, one can reconstruct at best a bandlimited approximation to the signal. Even in the case of a truly bandlimited signal, the sinc function (an ideal low-pass filter) cannot be computed in practice due to its infinite extent and relatively slow decay in the spatial domain. Imperfect filters are instead used in practice. These inevitably introduce various sorts of distortion into the reconstructed signal, depending on exactly how they differ from the ideal filter. Thus, designing or choosing appropriate practical, imperfect filters for reconstruction is a significant area of research.

Marschner and Lobb [40] evaluated various practical filters in the volume rendering context. They proposed a challenging function for reconstruction whereby the distorting effect of various filters becomes very apparent through visual inspection (Figure 2.3). From a more precise theoretical standpoint, they proposed classifications of the distortion effects of various reconstruction filters and corresponding metrics to quantify its characteristics more precisely. The metrics capture the idea of deviation from ideal, however they fail to consider how a particular filter may be appropriate given the specific properties of the function being reconstructed, *e.g.* the metrics consider the deviation from ideal equally at all frequencies within the given bandwidth limit Ω. For example, distortion at low energy ranges of the spectrum clearly is not as important as distortion in the spectrum where there is a larger energy concentration. This signal dependent weighting could be incorporated into a reasonable distortion metric for reconstruction filters.

For gridded data, linear interpolation tends to be the fastest type of convolution-based reconstruction since the required sample neighborhood size is small, while quality is still quite acceptable [60, 40]. Moreover, specialized trilinear interpolation hardware is now standard on GPUs. However, it is true that linear filtering can introduce significant amounts of aliasing in certain cases that can become fairly perceptible [40] (See Figure 2.3). These artifacts probably have more to do with the discontinuous derivatives of the linear filter. Higher-order reconstruction kernels were made practical for real-time by Sigg and Hadwiger [58], using a trick that makes use of trilinear interpolation hardware, providing better quality results at acceptable speeds.

For ungridded, scattered data convolution reconstruction can also be used, but with less choices for reconstruction kernels in the multidimensional context. There is lots of work on triangulation of scattered data points, with extensive theories relating to multiscale representations [16, 33]. However, triangulations are expensive to create for time-varying data in terms of maintaining a real-time rate. Moreover, from a signal processing standpoint they produce only C^1 continuous surfaces and thus suffer gradient discontinuities and aliasing for rapidly varying signals such as the Marschner-Lobb test function (Figure 2.3).

Despite the difficulties of this reconstruction framework, in practice the numerical computation of the convolution integral parallelizes nicely. The data flow required for its computation allows reading from slower main memory only once when cache hits are perfect. It requires no fitting–the given samples *are* the coefficients. Thus, computing these reconstructions maps well onto parallel high-performance computing hardware.

Consider the opposite situation where one does not use sample values directly for reconstruction. That is, a reconstruction formula of the form:

$$f(x) = \sum_{k \in \mathbb{Z}} c_k \phi(x-k) \tag{2.8}$$

In the bandlimited reconstruction framework, $c_k = f(Tk)$ and $\phi(x-k) = sinc(x/T - k)$. Given this generalized form, it is possible to explore other functions ϕ with the added need to determine the appropriate coefficients c_k.

The bandlimited function assumption, though useful can fall short in its ability to represent real-world signals commonly encountered. Projection onto the space of bandlimited functions means that the space is spanned by integer translate sinc functions. This basis function has infinite spatial extent with slow decay and is thus numerically complicated. Splines replace the sinc basis with functions having compact support in the spatial domain. Moreover, the basis function does not need to be interpolating, *i.e.* such that $\phi(k) = f_k$ because the samples are pre-filtered with the basis function, a step analogous to the case of the classical reconstruction paradigm where a given function is pre-filtered by being made bandlimited. The advantage is that the spectrum of the reconstructed function may have energy at infinite frequencies, beyond any finite band-limit.

Splines like this do yield good practical reconstruction quality. On the other hand, filtering the signal in order to study the signal at various scales is still performed in terms of the standard Fourier spectrum. The prefiltering step requires a single pass over the data prior to a subsequent pass for reconstruction. Moreover, this framework cannot directly handle scattered datasets. Vuccini *et. al* convert scattered point-based samples into a uniform spline-based representation [68], but this requires preprocessing steps which compromise its ability to scale to large datasets while maintaining a real-time rate.

Other reconstruction techniques exist which can handle regular or scattered data, but require a preprocess involving a memory pass over data and then some kind of optimization problem solution in order to determine coefficients for samples in a recontruction equation. Moving Least Squares (MLS) for volume rendering [34] achieve good quality results, but not at a real-time rate. Radial basis function (RBF) fitting [9] has similar properties with the same disadvantages.

All the reviewed reconstruction techniques have in common essentially that the basis functions used are signal independent (*e.g.* convolution methods use the standard Fourier basis with the bandlimited assumption). In this thesis, *we hypothesize that signal dependent methods can have advantages over signal independent methods in terms of reconstruction quality*. While nonlinear reconstruction methods based on the solution of partial differential equations for interpolation have been proposed [18], they would typically be considered much too slow for real-time rendering purposes. What is lacking in terms of the state of the art are techniques for reconstruction that are signal dependent and can still be computed at real-time rates for large datasets.

2.4 Summary

Scientific simulations produce large amounts of data at ever-increasing rates as computing capacity continues to increase. The large amounts of data that can now be produced pose significant problems in terms of human interpretation of scientific phenomenon being studied computationally. Effective visualization tools have great potential to bridge the gap between raw quantitative data and qualitative human understanding.

The recent trend towards greater parallelism in consumer-grade computing, particularly graphics accelerators promises to allow intelligent processing of data for the purposes of visualization. Effective visualization algorithms now must be carefully targeted to such parallel architectures in order to make full use of the raw arithmetic capacity of parallel architectures.

One visualization technique that has benefitted greatly from such parallel acceleration technology is volume rendering. Volume rendering has become a common technique used for rendering and visualizing scientific data. Effective volume rendering requires appropriate signal processing of the underlying data under the constraint of real-time performance and large-data scalability; still presenting significant challenges in this area.

Particularly challenging for volume rendering is the case of non-uniform data. There are many existing techniques for volume rendering of uniformly-gridded data, however, increasingly common in scientific simulations are scattered point-based data sets. These pose additional problems for high-performance rendering. A careful strategy for implementation onto modern high-performance architectures such as GPUs must be considered after a careful analysis of the problem and its inherent parallelization properties.

Such an implementation would allow for the real-time reconstruction of surfaces from point-based simulation data such as molecular dynamics simulations. How precisely to reconstruct this surface then becomes an interesting questions. Tradeoffs between quality and maintaining interactivity must be considered. Existing interactive reconstruction techniques tend to be based on signal independent basis functions, since signal dependency inevitably adds additional computation complexity. Intuitively, however, one would guess that signal dependent methods could have advantages in terms of signal quality and discriminancy in terms of signal processing

In the following chapters, we attempt to take steps towards determing if signal dependent methods (i) can be advantageous for computational steering and visualization purposes and (ii) whether they can be made amenable to real time-implementaion using GPU-based high-performance acceleration. Specifically, we will examine how methods based on signal independent basis functions can be made signal dependent at the expense of some extra computational complexity through the notion of time-warping. Then we will examine how reconstruction techniques for point-based datasets can be implemented.

Chapter 3

Time-Warped Multiscale Signal Processing

In this chapter, we explore a generalization of the well-known linear scale space theory. Scale space theory is most often studied in terms of its application to images. However, it is also quite natural to represent scale in terms of geometric structures of various dimensions.

3.1 Linear Scale Space

We review the well-known linear scale space theory based on linear diffusion or equivalently convolution filtering using a Gaussian function. The idea is simple– the signal is filtered by a low pass filter to generate each scale, with coarser scales corresponding naturally to progressively attenuated higher frequencies [74] . More formally, the scale-space representation of a signal f can be written as:

$$f(t,s) = \int f(t)G(t,s)dt \tag{3.1}$$

where $G(t,s)$ is a normalized Gaussian kernel.

Though this representation seems to intuitively capture a useful notion of scale, it is not without its shortfalls. Linear scale space has been presented as a model of the inital, uncommitted stages of biological vision, with physiological evidence given to suggest that the model matches reality to some extent [54]. The result is that images are coarsened uniformly across their spatial extent. This, while useful, can result in some serious limitations.

3.1.1 Problem: Delocalization

An intelligent vision system (biological or otherwise) requires the ability to discriminate scale in a localized fashion. If a particular vision-guided task requires tracking two objects of different sizes or even the same object at different distances, then localization of two seperate ranges of scales within the field of view is required.

As noted, existing linear scale space theory represents scale uniformly across the given signal. One of the simplest consequences of this fact is that geometric edges tend not to be well-preserved in linear scale space, and quickly become blurred (Figure 3.1). Such edges can be seen to represent a local spike in signal bandwidth (Figure 3.2). In images, edges can represent the boundary between two objects or a region of high curvature on a surface. It can be useful to preserve these localized bandwidth spikes in certain situations. Failing to preserve the spikes, it can be seen that the linear scale space filtering attenuates the energy in each window uniformly without adapting to the local bandwidth therein (Figure 3.2).

3.2 Time-Warped Scale Space

In this section, we propose a generalization of Gaussian scale space in order to address some of the issues with linear scale space. The basic problem with linear scale space is that it is based on filtering in a signal independent basis with no means of local discrimination.

Scale spaces that can adapt to local features of a signal have, in fact, been proposed and studied. Many of these are based on nonlinear differential equations [50, 70]. Though they can produce good results, they have some serious disadvantages. Numerical schemes to solve these differential equations can be complicated

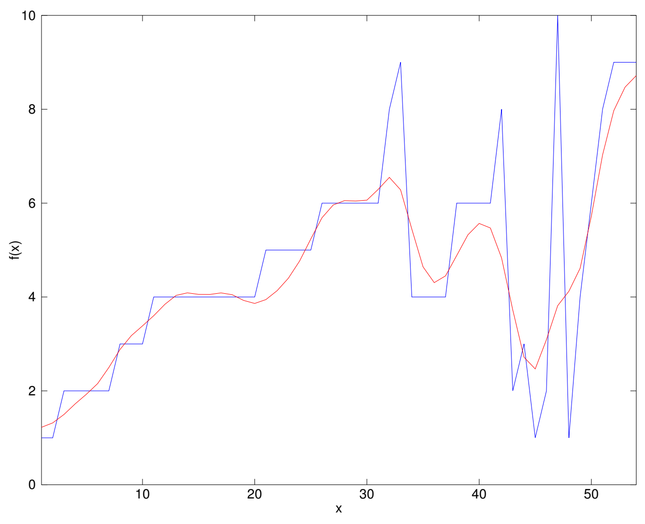
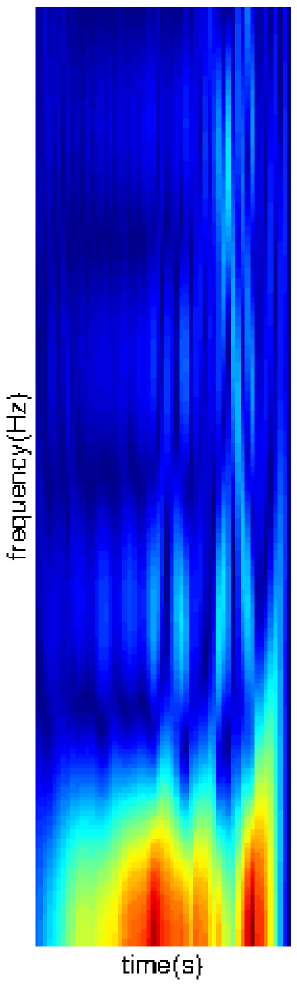


Figure 3.1: Effect of linear scale space filtering on sharp edges in a 1D signal. Original signal (blue) as compared to linear scale space filtered signal (red) at scale $s = 2.0$.

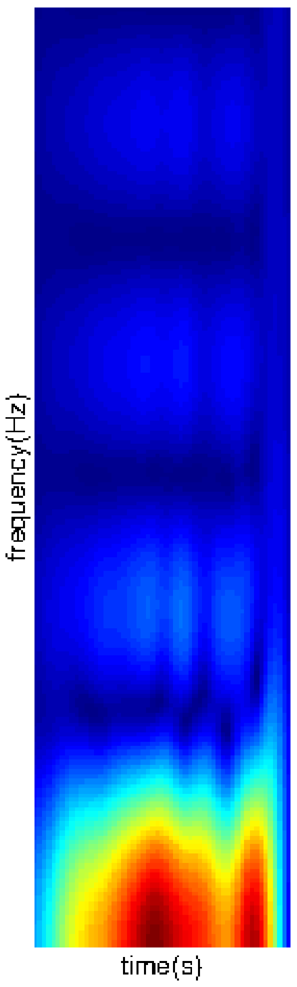
and additional noise and artifacts can be introduced through the fact that higher order derivatives must be computed. Moreover, solutions to solve the differential equations typically require multiple time-step iterations[50, 70], involving loads and stores of the entire dataset from slow dynamic memory in terms of implementation. Convolution-based reconstructions, as noted in the previous chapter, have more desirable high-performance characteristics in terms of potential memoy bandwidth bottlenecks. As we will see, the formulation of time-warped scale spaces adds this nonlinear capabilty to linear scale space while preserving much of the desirable high-performance characteristics of the linear case.

3.2.1 Time-Warped Signal Processing

We first explore the theory of time-warped signal processing. Clark *et. al* [11] introduced the idea of time-warped signals. The intuitive idea of a warping function is the re-parameterization of a signal. The set of sample values considered does not change, but the time-spacing between these samples is changed. More formally,



(a) Time-frequency analysis of given signal from Figure 3.1



(b) Time-frequency analysis of scale space filtered signal from Figure 3.1

Figure 3.2: Time-frequency analysis to demonstrate the effect of a sharp edge. Linear filtering does not adapt to this local signal bandwidth spike

given a signal $f(t)$ and a continuous invertible warping function $\alpha(t)$, one can consider a warped function $g(t) = f(\alpha(t))$. The original signal f may be recovered from $g(t)$ by the inverse warping function $\gamma(t)$, such that $f(t) = g(\gamma(t))$.

The basic idea of warped signal processing is to warp the signal using the α function before performing any signal modification then invert the warping back to the original signal space using the γ inverse warping function. Given a filter kernel $G(t)$ with signal $f(t)$ and warping function $\alpha(t)$ and $g = f(\alpha(t))$, one can define time-warped convolution similarly. We consider the convolution of g with the kernel K:

$$g * G(\tau) = \int g(t')G(\tau - t')dt' \tag{3.2}$$

We may consider a re-parameterization of this convolution via the γ , *i.e.* consider $\tau = \gamma(t)$. This yields the concept of time-warped convolution:

$$(f(\alpha(t))) * G(t)|_{\gamma(t)} = \int g(\gamma(t'))G(\gamma(t) - \gamma(t'))dt' = \int f(t')G(\gamma(t) - \gamma(t'))dt' \tag{3.3}$$

3.2.2 Generalized Time-Warped Scale Space

Given that linear scale space can be defined in terms of convolution, we give a natural generalization of scale space based on combining it with time-warping:

Definition 1 *Consider a signal $f(t)$ and a continuous invertible function $\alpha(t)$ with its inverse $\gamma(t)$. Then one can define the time-warped scale space relative to α as $F(t, \alpha, s) = (f(\alpha(t)) * G(t, s)|_{(\gamma(t)}$ where G is a normalized Gaussian kernel and s is the continuous scale parameter, where $F(t, \alpha, 0) = f(t)$.*

An equivalent way of viewing this definition is that we are exploring the linear scale space of the forward warped function $g = f(\alpha(t))$. Hence, any properties of standard scale space will apply to the scale space of g. Also, we note that it is straight forward to extend this definition into two and higher dimensions in order to apply it to images–it simply requires the notion of a warping function in two-dimensions.

3.2.3 Scale Space Property

The standard linear scale space is known to have several desirable properties, the validity of which follows from the assumption of modelling an uncomitted (*i.e.* signal independent) visual front-end [14]. A warping function introduces the possibility of nonlinearity based on some committed prior knowledge embodied by the warping function.

There are two essential properties that must be satisfied: the signal should be causal (that is, no new structures or features should spuriously emerge) and it should be possible to establish relationships between scales, *i.e.* that an extremum at a coarse scale must be "caused" by extrema at a finer scale [28]. This is known as the *causality* condition. When one-dimensional scale space filtering was first introduced, it was immediately noted that no new non-degenerate extrema can be created as the scale parameter is increased [74]. Moreover, it was also shown that existing extrema are not enhanced as the scale parameter is increased, that is maxima decrease and minima increase [28].

An important question at this juncture is whether the nonlinearity introduced by the warping function preserves the essential features of a scale space. It turns out that this is indeed the case. The essence of the arguments that follow is actually quite simple. Warped scale spaces are essentially isomorphic to the linear case, upto the warping transformation. As we show, this transformation preserves these essential aspects of scale space.

Lemma 1 *Consider a signal $f(t)$ and forward/inverse warping function pair α, γ and let $g(t) = f(\alpha(t))$. Let $G(t,s)$ be the linear scale space of g and $F(t,\alpha,s)$ be the α-warped scale space of f. Then for a given s, t_e is an extrema of $G(t,s)$ if and* *only if $\alpha(t_e)$ is an extrema of $F(t,\alpha,s)$.*

Proof: By definition, $G(t,s) = F(\alpha(t),\alpha,s)$. Thus, $\frac{\partial}{\partial t}G(t,s) = \frac{\partial}{\partial t}\{F(\alpha(t),\alpha,s)\} = \frac{\partial}{\partial t}F(t,\alpha,s)|_{t=\alpha(t)}\alpha'(t)$. Since there is an extrema at t_e , $\frac{\partial}{\partial t}G(t,s)|_{t=t_e} = \frac{\partial}{\partial t}F(t,\alpha,s)|_{t=\alpha(t_e)}\alpha'(t) = 0$. $\alpha(t)$ is a warping function, hence $\alpha'(t) > 0$ for all t. Thus, $\frac{\partial}{\partial t}F(t,\alpha,s)|_{t=\alpha(t_e)}\alpha'(t) = 0$ if and only if $\frac{\partial}{\partial t}F(t,\alpha,s)|_{t=\alpha(t_e)} = 0$, if and only if $t'_e = \alpha(t_e)$ is an extrema point of $F(t,\alpha,s)$. ▮

Corollary 1 *Consider a signal $f(t)$ and a warping function pair $\alpha(t)$ and $\gamma(t)$. Then $F(t,\alpha,s)$ is such that no new extrema are created as s is increased.*

Proof: Suppose that it did in fact create a non-degenerate extrema of $F(t,\alpha,s)$ at some point t_e at $s = s_e$. Then by the previous lemma, this non-degenerate extrema would imply a corresponding created extrema of $G(t,s)$, which gives a contradiction. ▮

Corollary 2 *In terms of an α-warped scale space of a function $f(t)$, extrema are not enhanced. That is, $\frac{\partial}{\partial s}F(t,\alpha,s) \leq 0$ at spatial maxima and $\frac{\partial}{\partial s}F(t,\alpha,s) \geq 0$ at spatial minima.*

Proof: By the above results, we know there is a one-to-one and onto correspondance between spatial extrema of $G(t,s)$ and $F(t,\alpha,s)$. Suppose t_e is a spatial extrema of $G(t,s)$, therefore $\frac{\partial}{\partial s}G(t,s) \leq 0$ at a maxima and $\frac{\partial}{\partial s}G(t,s) \geq 0$ at a minima. Since $\frac{\partial}{\partial s}G(t,s) = \frac{\partial}{\partial s}F(t,\alpha,s)|_{t=\alpha(t')}\alpha'(t)$ where $\alpha'(t) > 0$ and since we know there is a one-to-one and onto correspondance between spatial extrema of $G(t,s)$ and $F(t,\alpha,s)$, it follows that $\frac{\partial}{\partial s}F(t,\alpha,s) \leq 0$ and $\frac{\partial}{\partial s}F(t,\alpha,s) \geq 0$ at maxima and minima of $F(t,\alpha,s)$ respectively. ▮

3.3 Arclength-Warped Scale Space

We have defined a time-warped scale space and justified that it holds at least the basic properties required of a scale representation scheme. The next most important question is regarding how to choose (and compute) a warping function. A warping function could represent a physical transformation of a signal, for example to model a doppler shift [11]. However, this is not immediately appropriate for a single, fixed geometric signal. More relevantly, it has been proposed that the derivative of a warping function represents an estimate of the local bandwidth of signal [11]. The derivative of a warping function intuitively represents the degree of stretch or dilation of the signal induced by the warping or re-parameterization, compensating for local bandwidth variations accordingly.

A strategy for creating a warping function follows: estimate a signal's local bandwidth, then create a warping function whose derivative function matches the bandwidth function. In this light, we consider the following inverse warping function for a signal f,

$$\gamma(\tau) = \int_0^\tau \sqrt{1+|f'(t)|^2}dt \tag{3.4}$$

where $\gamma(\tau)$ measures the arc length along $f(t)$ from 0 to τ. $\gamma(\tau)$ is monotone and continuous, thus invertible and so $\alpha(\tau) = \gamma^{-1}(\tau)$ exists. In this case $f(\alpha(t))$ is $f(t)$ re-parameterized in terms of its arclength. This function can be viewed to estimate local bandwidth of the signal f in terms of the first derivative of the signal itself.

3.3.1 Computing Warped Convolution

Warped convolution generalizes standard convolution. We may discretize the convolution integral for the warped case in an almost identical fashion to the way standard convolution is handled. The only difference is that the inverse warping function $\gamma(t)$ must be computed, which can add extra compute time to the calculation.

In the case of arclength, the warping function is clearly dependent on the given signal; hence, it cannot be precomputed or pre-specified. Since, in practice one only has the given samples of the signal $f(t)$, we may only approximately reconstruct the arclength warping function $\gamma(t)$. In practice, it is perhaps simplest to compute a linearly interpolated reconstruction of $f(t)$. Computing the arclength of this approximate reconstruction is then straight forward–one need only compute the length of each linear segment and sum these.

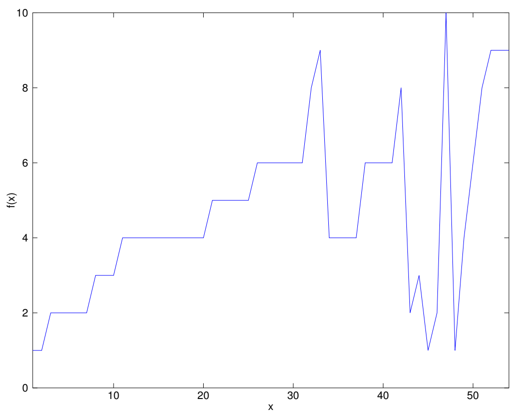
3.3.2 Experimental Local Interpretation

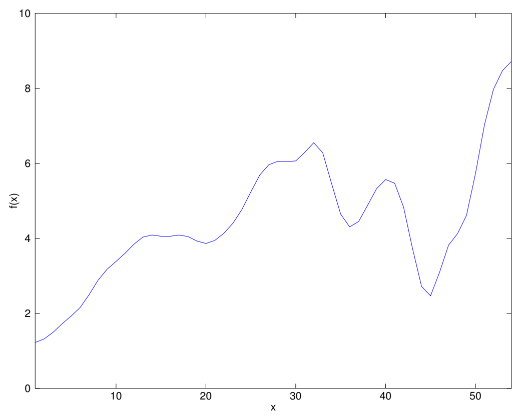
We explore the effect of the arclength warping function in terms of a time-warped scale space. As noted above, the arclength distance warping function estimates the local bandwidth of a given signal in terms of its derivatives. Consider the results of Figure 3.3. One can see that the warped filtering for the same level of scale ($s = 2.0$) produces a strikingly different result from the linear case. It can be clearly seen in the figure that large edge features are being preserved much longer in the warped case.

Previously, it was suggested that these sharp edges represent local bandwidth spikes in the signal (Figure 3.2). The linear filter's inability to preserve these interesting edge features could be viewed as the inability to adapt to local variations of the signal. In contrast, we see in Figure 3.4 that the arclength-warped filtering appears to adapt to local spikes in bandwidth. We plot the derivative of the warping function (Figure 3.5) and compare with the difference plot of the time-frequency plots of the linear and warped filtered signals. On the signal where the derivative of the warping function is small, in effect the warped filter is behaves very similarly to the linear case and in turn the time-frequency plots are nearly identical. In contrast, at parts of the signal where the derivative of the warping function has peaks, the time-frequency plots between linear and warped cases differ much more.

3.4 Summary

In this chapter, we have explored the limitations of linear scale space. In response to these limitations, we have proposed time-warped scale spaces to allow for nonlin-


(a) Given original signal


(b) Linear scale space at scale $s = 2.0$

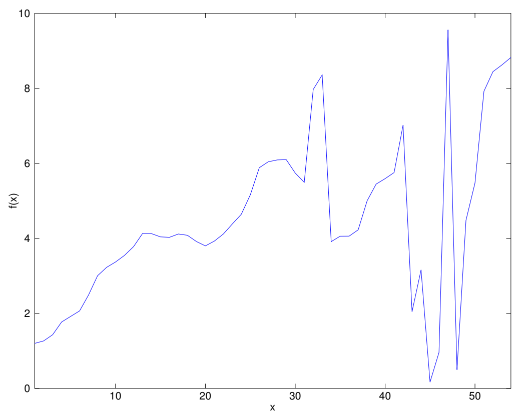
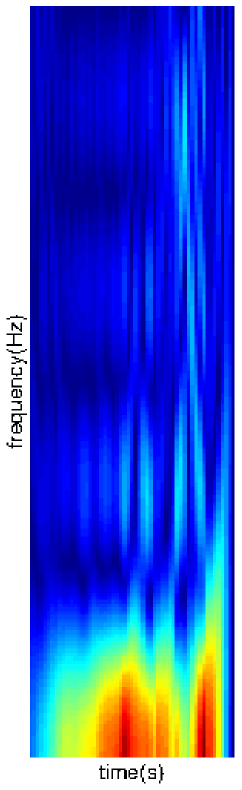
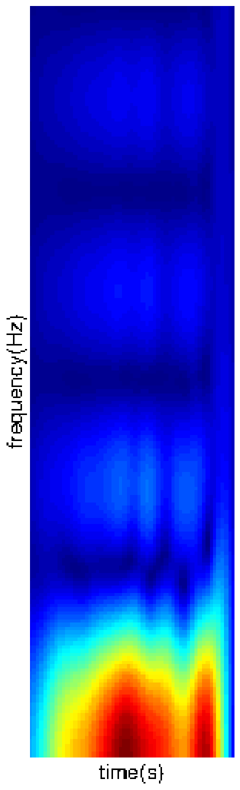

(c) Arclength-warped scale space at scale $s = 2.0$

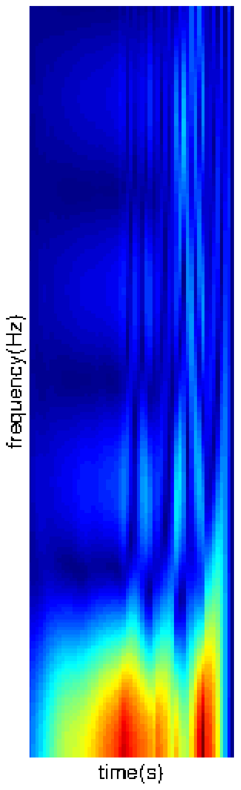
Figure 3.3: Linear scale space versus arclength-warped scale space



(a) original signal Figure 3.1

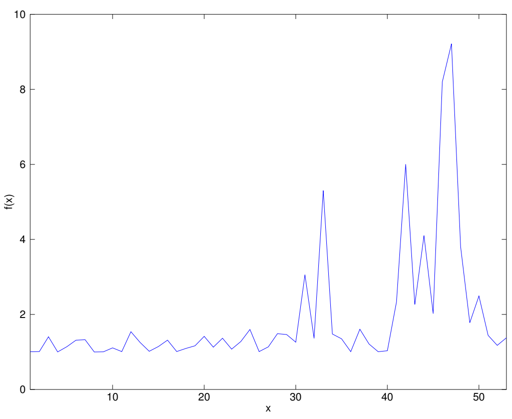


(b) linear filter signal

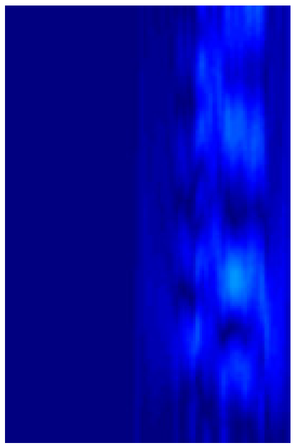


(c) arclength-warped filter signal

Figure 3.4: Time-frequency interpretation of time-warped filtering



(a) Plot of $\gamma'(t)$



(b) Difference of linear and warped time-frequency plots

Figure 3.5: Local bandwidth of warping function and its effect on local adaptation of warped scale space filter

earity in terms of a convolution formulation. We have shown that warping based on an arclength warping function enables one to overcome these limitations in linear scale space in exchange for a relatively minor increase in computational complexity.

Time-warping allows for the scale space filter to adapt itself locally to a signal. In the case of the arclength warping function, it provides an estimate of the local bandwidth of a signal and adapts filtering accordingly. This allows the filter to preserve edge-like features, effectively assigning them to a larger scale than in the linear case.

In terms of a computational steering system, an efficiently computable, edge-preserving, and multidimensional representation of scale would be extremely useful. This would allow for the ability to interactively explore ranges of scales in a scientific simulation while preserving interesting discontinuous features. Such a tool could be a useful part of a computational steering system. In Chapter 5, we consider how this could be implemented in a GPU-based parallel computing setting.

Given the positive results regarding warping, a natural question is whether it can be extended into the case of higher-dimensional signals. The work of Boulanger [6] suggests that the natural generalization of arclength, geodesic distance, does in fact provide the same advantages. However, it must be noted that in general, a filtering based geodesic distance cannot be cast into the warping framework in a straightforward fashion. Time-warping has an interpretation in terms of standard Fourier analysis (*i.e.* in terms of $g = f(\alpha(t))$) and a similar interpretation would be useful in higher dimensions. Future work here could involve establishing a connection to warping for the geodesic case.

While we have shown that arclength is a useful warping function since it allows us a simple way to estimate the local bandwidth of the signal, it is not clear if this is an optimal choice for a warping function. It would seem natural that other methods may be appropriate and perhaps preferable. In the next Chapter, we work further towards answering this question by attempting to formalize one aspect of a "good" warping function.

Chapter 4

Time-Warped Signal Reconstruction and Noise Removal

In the previous chapter, we introduced time-warped signal processing and explored using it in order to generalize scale space filtering. Warping was shown to be a way to adapt the standard Fourier basis to local aspects of a signal. In this chapter, we explore how time-warped signal processing can be used to improve signal reconstruction and noise removal.

4.1 Theoretical Aspects of Time-Warped Reconstruction

The idea of warping a signal with a given warping function implies a reconstruction formula analogous to the classical Shannon reconstuction formula (*i.e.* Theorem 1).

Theorem 2 ***Clark et al. [11]:*** *Consider a finite-energy, Ω-bandlimited function f (i.e. there is $\Omega > 0$ such that $\hat{f}(w) = 0$ for all $|w| > \Omega$). Then f is determined by a set of not necessarily regular samples $\{f(t_n) | n \in \mathbb{Z}\}$ if a one-to-one and continuous mapping $\gamma(t)$ exists such that $kT = \gamma(t_n)$ for $T = 1/2\Omega$ and if $g(t) = f(\alpha(t))$ is bandlimited to Ω, i.e. $f(t) = \Sigma_{n\in\mathbb{Z}} f(t_n) sinc(\frac{\gamma(t)-\gamma(t_n)}{T})$.*

The basic idea of warped signal processing is to: (1) warp the signal using the α function, (2) perform signal modification then (3) invert the warped and modified function back to the original signal space using γ, the inverse warping function.

To illustrate this point, it is easy to show that the warped reconstruction formula (Theorem 2) is equivalent to reconstructing $g(t) = f(\alpha(t))$ using the classical Shannon reconstruction formula, then computing $f(t) = g(\gamma(t))$. (See proof of Theorem 2 in Clark *et al.* [11])

We note that this implies that for most warping functions, the warped reconstruction formula is in fact a non-uniform sampling theorem. That is, this is a uniform sampling theorem on the function $g = f(\alpha(t))$ and these samples become non-uniform for most warping functions when we compute $f = g(\gamma(t))$. Hence, like in the classical case, a sufficient sampling rate is required in order to avoid aliasing when reconstructing the signal.

4.1.1 Convolution Reconstruction Error

Both the classical and time-warped bandlimited signal reconstruction theorems give a theoretical formula for the perfect reconstruction of the sampled function; in that sense they are equivalent. However, in practice, the *sinc* reconstruction kernel is substituted by a low-pass kernel with compact support and an approximation error is introduced. As mentioned in Chapter 2, there are two chief sources of error using the convolution reconstruction formula in practice: (1) the bandlimited signal assumption fails and (2) practical reconstruction filters deviate to some degree from the ideal.

More concretely, based on the Shannon reconstruction theorem (Theorem 1), one can reconstruct a signal from its samples by the following:

$$f(x) = \sum_{k\in\mathbb{Z}} f(kT) sinc(x/T - k). \tag{4.1}$$

As mentioned previously, this formula cannot be computed. In practice, we substitute the *sinc* kernel with a compactly-supported kernel h where $\int h(t)dt = 1$.

This yields the following reconstruction formula:

$$f_{rec}(x) = \sum_{k\in\mathbb{Z}} f(kT)h(x-kT). \tag{4.2}$$

Marschner and Lobb [40] proposed metrics for evaluating the distortion characteristics of practical reconstruction filters. Deviation of a filter from the ideal *sinc* filter can be evaluated in terms of the frequency response curve of the practical filter. Specifically, they propose a *smoothing* metric which essentially measures the difference in energy under the frequency curve within the specified bandlimit of a practical filter h from the ideal low-pass filter. The other metric they propose is a *postaliasing* metric, which essentially measures the area of the frequency curve of the practical filter h outside the specified bandlimit.

Assuming H is the Fourier transform of h, we consider the definition of smoothing as defined by Marschner and Lobb [40]. They define the smoothing metric of a filter h relative to a bandlimit Ω as:

$$s(h) = \frac{1}{\Omega}\int_{-\Omega}^{\Omega}(1-|H(w)|)dw. \tag{4.3}$$

Essentially, this denotes the area under the curve in the frequency domain between the ideal *sinc* filter and h. The postaliasing error metric represents the area under the curve of H outside the given specified bandlimit Ω. Thus, it can be defined as the difference between the total filter energy and the smoothing metric:

$$p(h) = \frac{1}{\Omega}\int_{-\Omega}^{\Omega}|H(w)|dw - s(h). \tag{4.4}$$

These metrics were shown to be relatively effective in predicting the reconstruction performance of a filter. However, they are agnostic to the particular signal being reconstructed. Instead, we consider a set of signal-weighted metrics relative to a finite energy, bandlimited signal f. A signal-weighted smoothing metric for a filter h relative to a signal f can be defined as:

$$s(h,f) = \frac{1}{\Omega}\int_{-\Omega}^{\Omega}|F(w)|(1-|H(w)|)dw. \tag{4.5}$$

Analogously, one can define a signal-weighted postaliasing metric. Let $B = \int |f(x) - f_{rec}(x)|dx$. We then define the signal-weighted postaliasing:

$$p(h, f) = B - s(h, f) \tag{4.6}$$

Hence,

$$B = p(h, f) + s(h, f), \tag{4.7}$$

where $s(h, f)$ can be interpreted as the amount of signal energy within the specified bandlimit lost to smoothing error. The remaining signal energy is distributed outside the bandlimit and results in post-aliasing, that is $p(h, f)$.

4.1.2 Time-Warped Convolution Reconstruction Error

It is known that a time-warped analogue of the classical reconstruction theorem exists. As in the classical case, the given formula cannot be computed in practice unless we substititute a practically computable kernel h. Hence, using the result of Theorem 2, we get:

$$f_{rec}(x) = \sum_{n \in \mathbb{Z}} f(t_n) h(\gamma(x) - \gamma(t_n)) \tag{4.8}$$

As mentioned previously, the time-warped reconstruction formulas can be viewed as the classic convolution reconstruction performed on $f(\alpha(t))$, where the resulting reconstructed signal is then un-warped using $\gamma(t)$. Hence, we may consider the error of reconstructing a warped $f(\alpha(t))$ versus an unwarped $f(t)$. Since the reconstruction error metrics given in the previous section were formulated in terms of the signal's Fourier representation, we first consider the effect of warping on the spectrum of a signal. It can be shown [11]:

$$F(w) = \int_{-\infty}^{\infty} G(k) P_\alpha(w, k) dk \tag{4.9}$$

where

$$P_\alpha(w, k) = \int_{-\infty}^{\infty} e^{i2\pi k \alpha(t)} e^{-i2\pi wt} dt. \tag{4.10}$$

That is, $P_\alpha(w,k)$ is the Fourier transform of the function $p(t) = e^{i2\pi k\alpha(t)}$. We refer to P_α as the reverse transfer function for α. It can also be seen that:

$$G(w) = \int_{-\infty}^{\infty} F(k)P_\gamma(w,k)dk \tag{4.11}$$

where

$$P_\gamma(w,k) = \int_{-\infty}^{\infty} e^{i2\pi k\gamma(t)}e^{-i2\pi wt}dt. \tag{4.12}$$

This defines $P_\gamma(w,k)$ which we refer to as the forward transfer function.

Intuitively, these transfer functions describe how spectral energy is transferred between $f(t)$ and $g(t)$ in relation to the given warping function $\alpha(t)$ and its inverse $\gamma(t)$. The re-parameterization due to warping does not alter the total signal energy, hence the transfer functions must be normalized. That is:

$$\int_{-\infty}^{\infty} |P_\gamma(w,k)|dk = 1 \tag{4.13}$$

We are now in a position to evaluate how warping alters the spectrum of a function and how this relates to the smoothing error involved in terms of reconstructing the function.

Definition 2 *Consider a warping function $\alpha(t)$. We say that $\alpha(t)$ is a compressive warping function if reverse transfer function $P_\gamma(w,k)$ is such that $P_\gamma(w,k)$ = 0 for $k > w$.*

The basic idea of a compressive warping function is that energy is transferred strictly to a lower (and not higher) frequency during warping. We choose, for example, a simple warping function, say $\alpha(t) = 0.5t$. Then $\gamma(t) = 2t$. By the well-known Fourier scaling theorem, we know that scaling the time axis results in a corresponding scaling of the frequency axis. Hence, one can see this results in a compressive warping function. However, for a bandlimited signal $f(t)$, this scaling will thus also scale the bandwidth. The resulting warped reconstruction formula would be equivalent to the unwarped case for such a γ. Hence, this could be seen as a trivial compressive warping function in that it is effectively equivalent to not warping at all in terms of reconstruction.

Thus, we require a further method of discrimination for a compressive warping function. A warping function applied to a bandlimited function may clearly alter the bandwidth. It is argued that what is crucial is how the spectral energy is distributed relative to the bandwidth. A simple linear scaling warping function does not alter the distribution of signal energy relative to the bandlimit; it merely scales the bandwidth.

We argue that a ”good” warping function, redistributes energy into lower frequencies relative to the bandlimit. If a given warping function preserves the bandlimited property but does change the bandwidth of a function, it is possible to normalize the warping function such that the bandlimit does not change, by introducing a compensative scaling term and applying the Fourier scaling theorem.

Definition 3 *We say that $\alpha(t)$ is a bandlimited-preserving warping function if for an arbitrary bandlimited signal $f(t)$, $f(\alpha(t))$ is also bandlimited.*

Lemma 2 *Given a bandwidth-preserving warping function $\alpha(t)$, there exists a warping function $\alpha_d(t) = c\alpha(t)$ for a real c such that for a bandlimited signal $f(t)$, $f(\alpha_d(t)$ has the same same bandlimit as $f(t)$.*

Definition 4 *Given a bandwidth Ω, consider a bandwidth-preserving warping function $\alpha(t)$ that has been normalized to Ω. We say that $\alpha(t)$ is a relatively compressive warping function if $P_\gamma(w,k)$ is such that $P_\gamma(w,k)$ = 0 for $k > w$.*

This definition appears to omit trivial warping functions and capture the essence of a “good” warping function. The basic idea is that after warping, a significant amount of signal energy has been redistributed to lower frequencies. We now can show that this produces less smoothing error during the convolution reconstruction process for a certain class of reconstruction filters:

Theorem 3 *Let $\alpha(t)$ be a relatively compressive warping function (with inverse $\gamma(t)$) and a normalized reconstruction filter h such that $H(w)$ is monotonically decreasing. Then $s(h, f(\alpha(t)) \leq s(h, f)$.*

Proof: For the following, we require the Riemann sum definition of an integral. The integrals that will follow are defined on $[-\Omega, \Omega]$. We consider a partition of this interval with ΔX_i denote the ith subinterval. Let c_i represent any point in the ith subinterval. We evaluate:

$$\begin{aligned}
s(g,h) &= \frac{1}{\Omega}\int_{-\Omega}^{\Omega} |G(l)|(1-|H(l)|)dl && (4.14)\\
&= \frac{1}{\Omega}\int_{-\Omega}^{\Omega} |\int_{-\Omega}^{\Omega} F(k)P_\gamma(l,k)dk|(1-|H(l)|)dl && (4.15)\\
&\leq \frac{1}{\Omega}\int_{-\Omega}^{\Omega}\int_{-\Omega}^{\Omega} |F(k)P_\gamma(l,k)|dk(1-|H(l)|)dl && (4.16)\\
&= \frac{1}{\Omega}\lim_{m\to\infty}\sum_{j=1}^{m}\lim_{n\to\infty}\sum_{i=1}^{n}|F(c_i)|P_\gamma(c_j,c_i)|\Delta X_i(1-|H(c_j)|)\Delta X_j && (4.17)\\
&= \frac{1}{\Omega}\lim_{m\to\infty}\lim_{n\to\infty}\sum_{i=1}^{n}|F(c_i)|\sum_{j=1}^{m}|P_\gamma(c_j,c_i)|(1-|H(c_j)|)\Delta X_j\Delta X_i && (4.18)\\
&= \frac{1}{\Omega}\int_{-\Omega}^{\Omega}|F(k)|\int_{-\Omega}^{\Omega}|P_\gamma(l,k)|(1-|H(l)|)dldk && (4.19)\\
&\leq \frac{1}{\Omega}\int_{-\Omega}^{\Omega}|F(k)|(1-|H(k)|\int_{-\Omega}^{\Omega}|P_\gamma(l,k)|dldk && (4.20)\\
&= \frac{1}{\Omega}\int_{-\Omega}^{\Omega}|F(l)|(1-|H(l)|)dl && (4.21)\\
&= s(f,h) && (4.22)
\end{aligned}$$

We explain this result on a line-by-line basis. Equation 4.15 comes from substituting Equation 4.11. Equation 4.16 is a well-known property of integrals. Equation 4.17 comes by Riemann sum definition of integral. Equation 4.18 comes by re-arrangement of the order of sums in the finite summation formulas. Equation

4.19 then re-applies the defintition of Riemann integral on the re-arranged sums, giving a new set of integrals. In Equations 4.20, it is possible to pull $|(1-H(k))|$ outside the integral under inequality because $\alpha(t)$ is relatively compressive, thus $P_{\gamma}(l,k) = 0$ for $k > l$ and because $|(1-H(k))| \geq |(1-H(l))|$ for $l \leq k$ by monotonicity of H and the fact that h is normalized. We then get Equation 4.21 since by Equation 4.13, $\int_{-\infty}^{\infty} |P_{\gamma}(w,k)| dk = 1$.

▮

The essence of this argument is that for a monotonically decreasing $|H(w)|$, the deviation of $H(w)$ from the ideal low-pass box filter is less for lower frequencies. Hence, the warped signal is better suited for the unideal filter than the unwarped signal. In other words, for a relatively compressive warping function, signal energy is transferred to lower frequencies which have less weight in the smoothing metric integral.

However, this theorem, on the surface, is inconclusive. Namely, it does not promise a substantial decrease in the smoothing error due to time-warping; the inequality is not strict. Looking at the derivation, one can see the critical aspect of $s(g,h)$ that determines how much less the smoothing error will be for a particular frequency k. That is, the value of $\int_{-\Omega}^{\Omega} |P_{\gamma}(l,k)|(1-|H(l)|) dl$ versus that of $(1-H(k))$.

Thus, P_{γ} determines how much the smoothing error is reduced. A pertinent question is: how do we compute P_{γ} in terms of the warping function and how do particular aspects of the warping function influence P_{γ}? Let us consider:

$$\begin{aligned} P_{\gamma}(w,k) &= \int_{-\infty}^{\infty} e^{i2\pi k\gamma(t)} e^{-i2\pi wt} dt && (4.23) \\ &= \int_{-\infty}^{\infty} e^{i2\pi k\gamma(t) - i2\pi wt} dt && (4.24) \\ &= \int_{-\infty}^{\infty} e^{-2\pi i\phi(t)} dt && (4.25) \end{aligned}$$

Lemma 3 *For γ, the arc-length warping function, $P_{\gamma}(w,k) \simeq 0$ for $k >= w$*

Proof: We consider an integral of the form:

$$\int_{-\infty}^{\infty} e^{-i\phi(t)} dt \tag{4.26}$$

We may use the method of stationary phase approximation to compute this integral. The function being integrated is oscillatory. The basic idea of this technique is that sinusoids will approximately cancel where the phase of the function varies rapidly. Significant non-zero contributions of the integral occur where $\phi' = 0$, known as points of stationary phase. It is assumed for the sake of approximation that all other contributions cancel completely.

In the case of the integral for $P_\gamma(k,w)$, we have, $\phi(t) = -k\gamma(t) + wt$. Hence, $\phi'(t) = -k\gamma'(t) + w$. Thus, $\phi'(t) = 0 \iff -k\gamma'(t) + w = 0 \iff \gamma'(t) = w/k$. In other words, points of stationary phase for the integral occur where $\gamma'(t) = w/k$. For $\gamma(t) = \int_0^t \sqrt{1 + |f'(\tau)|^2} d\tau$, it is easy to see that $\gamma'(t) \geq 1$. Hence, $\gamma'(t) = w/k \iff k \geq w$ and we can thus say that under the stationary phase approximation, $P_\gamma(w,k) \simeq 0$ for $k \geq w$. ▮

This result basically answers the question posed previously in regards to the arclength warping function. That is, the derivatives of the arclength function correspond with nonzero contributions of the reverse transfer function integral. This, in combination with previous results, implies that warped reconstruction with an arclength-based warping function will reduce reconstruction error:

Corollary 3 *Given $\gamma(t) = \int_0^t \sqrt{1 + |f'(\tau)|^2} d\tau$. Let $\alpha(t) = \gamma^{-1}(t)$ Consider a bandlimited signal f with bandlimit Ω. Let $g = f(\alpha(t))$. Let $f_{wrec}(x) = \Sigma_{k \in \mathbb{Z}} f(t_n) h(\gamma(x) - \gamma(t_n))$ and let $f_{rec}(x) = \Sigma_{k \in \mathbb{Z}} f(kT) h(x - kT)$. Then for reconstruction error measures $E_f = \int |f(t) - f_{rec}(t)| dt = s(f,h) + p(f,h)$ and $E_g = \int |f(t) - f_{wrec}(t)| = \int |g(t) - g_{rec}(t)| dt = s(g,h) + p(g,h)$, $E_g \leq E_f$.*

Proof: We observe that arclength is bandwidth preserving, unless signal is strictly monotone (and then could be re-normalized to be bandwidth-preserving). Hence, the bandlimit of g and f are both Ω. This implies that $p(f,g) = p(g,h)$. By Theorem 3, $s(g,h) \leq s(f,h)$, thus implying that $E_g \leq E_f$. ▮

4.2 Practical Time-Warped Reconstruction

We have proven that at least in theory, time-warping can decrease reconstruction error in terms of practical convolution-based reconstruction. In this section, we explore the practical aspects of the reconstruction and how well it adheres to the proposed theory.

In practice, various approximations and sources of error are inevitable. For a typical nontrivial warping function, the time-warped reconstruction theorem is, in fact, a non-uniform sampling theorem, as opposed to the classical uniform case. However, in practice, we are often dealing with a set of uniform samples as input. This inevitably introduces some tension between the theory and practice.

Moreover, in order to use the warped reconstruction formula, we must reconstruct the inverse warping function $\gamma(t)$ from the given samples, another inevitable source of approximation error.

4.2.1 Computing Warping Function

In the previous chapter, we were required to reconstruct the arclength warping function from a given set of samples in order to compute warped convolution. In terms of practical reconstruction, we are typically re-sampling the signal onto a grid with a higher sampling rate. In this case, it can be seen that one thus requires to have $\gamma(t)$ computed on the higher-resolution re-sampling grid. For the purposes of the following experiments, simple linear interpolation appears to suffice.

4.2.2 Reconstruction Results

We seek to experimentally verify that arclength warping can improve reconstruction error versus classical convolution reconstruction. For these purposes, we use the normalized Gaussian filter, $h(x,\sigma)$. This filter meets the monotonicity requirements of Theorem 3. We consider reconstruction results over a range of σ values for h.

Synthetic Data

We consider a one-dimensional version of the popular Marschner-Lobb test function [40]. That is:

$$f(x) = 1 + cos(12\pi cos((\pi x)/2.0)), x \in [-1,1] \tag{4.27}$$

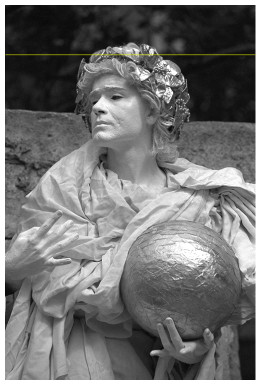
We sample this signal on the interval $[-1,1]$ with a sampling interval of 0.05. It can be shown that this signal has 99.8% of its energy below a frequency of 10Hz [40]. Hence, this sampling rate is sufficient to capture the signal's energy. However, a large amount of the signal's energy is in the higher frequency region of its bandlimit, hence it presents a challenging reconstruction test case. We reconstruct the signal onto a lattice on $[-1,1]$ with a sampling interval of 0.01.

Real Data

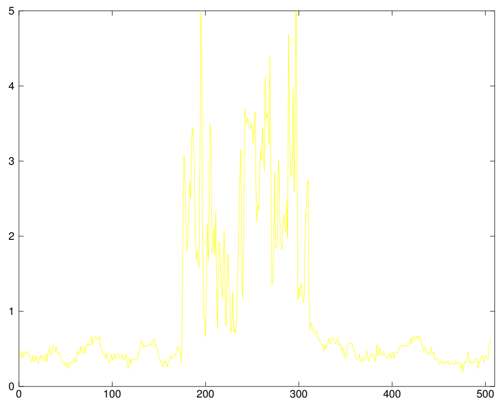
We perform a similar upsampling reconstruction experiment on a realistic dataset. We subsample every 5th sample from a scanline of a real-world image (Figure 4.1) and then reconstruct the signal from these samples. In this case, the original image serves as a ground truth.

Results and Analysis

In Tables 4.1 and 4.2, we present the computed reconstruction error, the sum of absolute value of the differences between the reconstructed function and the ground truth for the synthetic and real data cases. Looking at these tables, one can see



(a) Example real-world image



(b) 1-D image line ground truth for reconstruction

Figure 4.1: Real-world dataset used for reconstruction and noise removal experiements

Table 4.1: ML-test function reconstruction error comparison

σ	classical error	warped error
0.066667	25.129	30.419
0.133333	20.405	27.512
0.200000	21.519	24.879
0.266667	25.553	23.287
0.333333	29.856	22.534
0.400000	33.697	22.201
0.466667	37.070	22.178
0.533333	40.044	22.761
0.600000	42.679	23.832
0.666667	45.022	25.195
0.733333	47.114	26.615
0.800000	48.990	28.172
0.866667	50.685	29.747
0.933333	52.232	31.396
1.000000	53.645	33.042
1.066667	54.934	34.637
1.133333	56.114	36.196
1.200000	57.197	37.716
1.266667	58.196	39.174
1.333333	59.119	40.570

Table 4.2: Real data reconstruction error comparison

σ	classical error	warped error
0.066667	42.712	43.251
0.133333	41.616	42.346
0.200000	41.600	41.609
0.266667	42.364	41.400
0.333333	43.630	41.416
0.400000	44.977	41.629
0.466667	46.273	41.992
0.533333	47.435	42.398
0.600000	48.486	42.912
0.666667	49.495	43.493
0.733333	50.371	44.080
0.800000	51.154	44.649
0.866667	51.840	45.200
0.933333	52.452	45.735
1.000000	53.019	46.249
1.066667	53.552	46.743
1.133333	54.033	47.221
1.200000	54.464	47.680
1.266667	54.856	48.116
1.333333	55.215	48.541

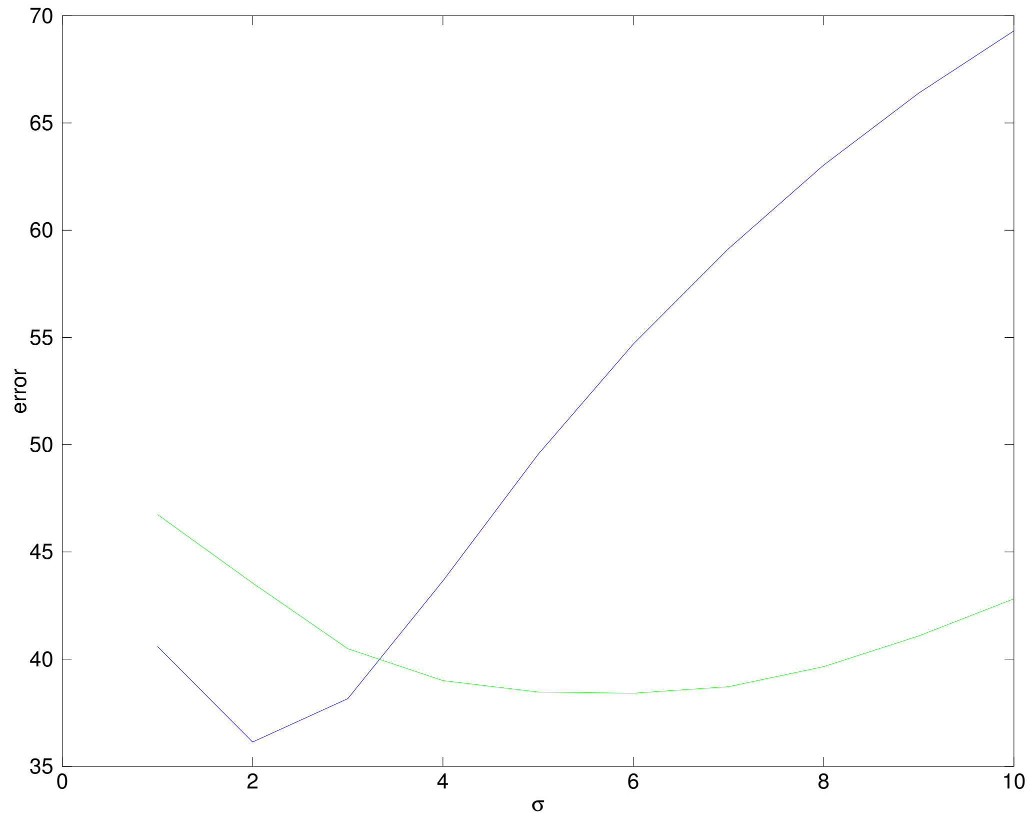


Figure 4.2: Plot of σ versus error for warped (green) and classical (blue) reconstruction for the synthetic ML-test data

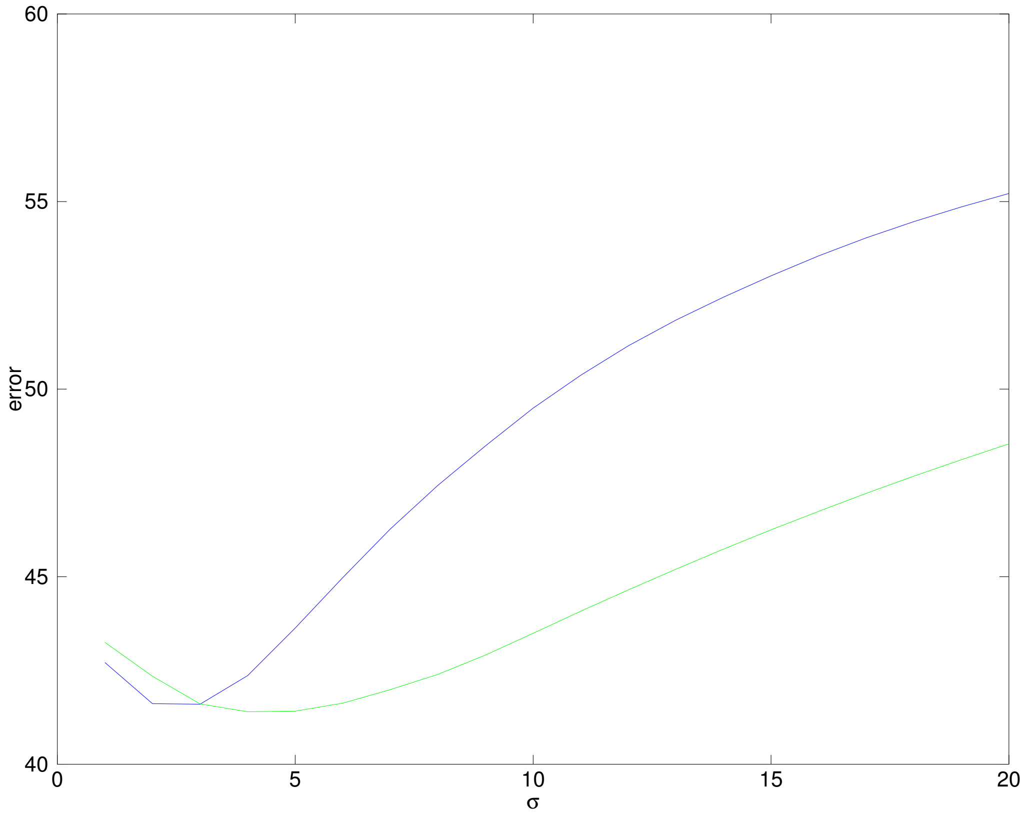
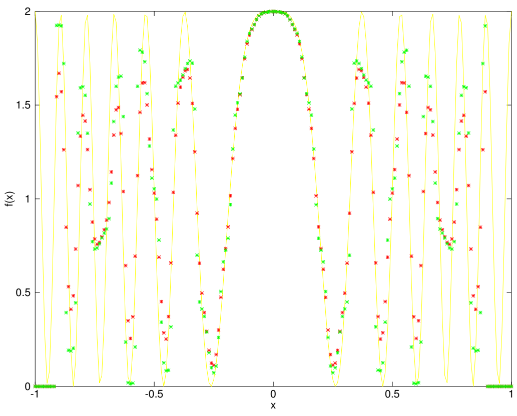
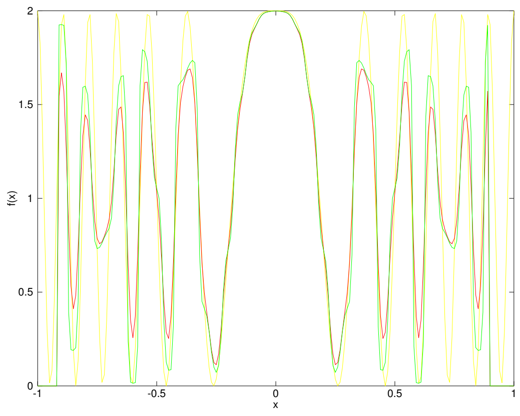


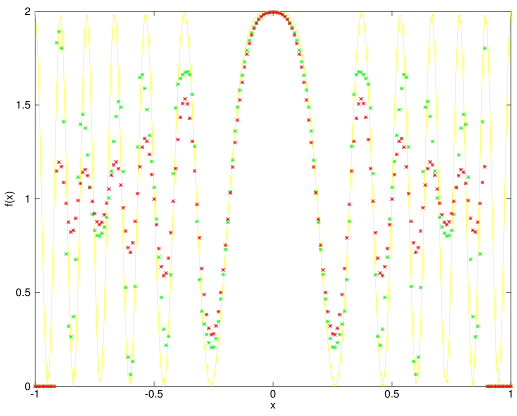
Figure 4.3: Plot of σ versus error for warped (green) and classical (blue) reconstruction for the real data



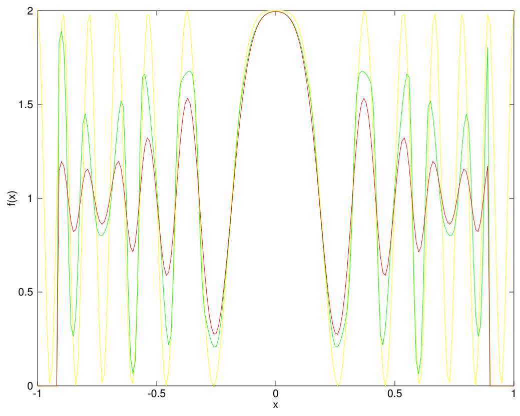
(a) Reconstruction at $\sigma = 0.20000$



(b) Linear interpolated reconstruction at $\sigma = 0.20000$

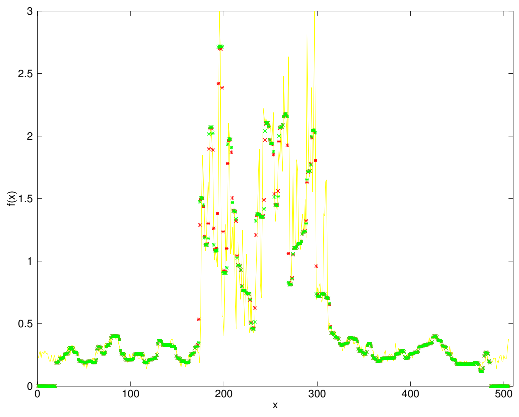


(c) Reconstruction at $\sigma = 0.46667$

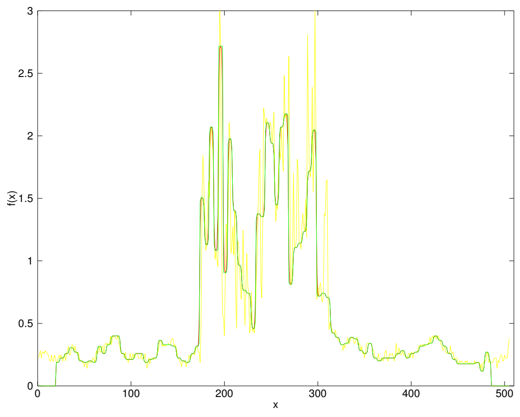


(d) Linearly interpolated reconstruction at $\sigma = 0.46667$

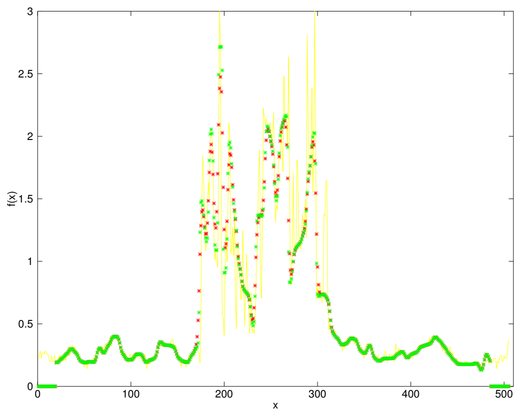
Figure 4.4: Synthetic data reconstruction of classical (red) and warped convolution (green) versus ground truth signal (yellow)



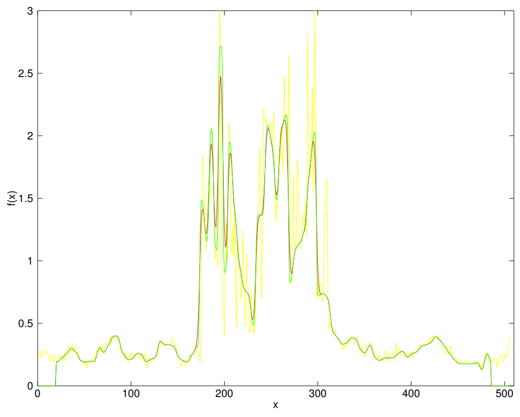
(a) Reconstruction at $\sigma = 0.13333$



(b) Linearly interpolated Reconstruction at $\sigma = 0.013333$



(c) Reconstruction at $\sigma = 0.40000$



(d) Linearly interpolated reconstruction at $\sigma = 0.40000$

Figure 4.5: Real data reconstruction of classical (red) and warped convolution (green) versus ground truth signal (yellow)

interesting results regarding the question of whether or not warped reconstruction can improve on classical reconstruction in practice.

We plot these reconstructions for various σ values both above and below this threshold in order to compare the warped and classical reconstruction (Figures 4.4 and 4.5). For the smaller σ value, it is difficult to see much difference between the warped and classical reconstructions in either case. For the larger σ value, one can see that the warped reconstruction has better preserved edges and appears to be a more accurate reconstruction.

It appears that for larger values of σ, the warped reconstruction error is less, in line with the predictions of the theory presented earlier in this section. On the other hand, for smaller values of σ, it is in fact greater. To be specific, there appears to be a threshold σ value below which the difference occurs, favoring the classical method. For example, in Table 4.1, this error measure threshold occurs around $\sigma = 0.2$ and in Table 4.2 around $\sigma = 0.13$. One can see this clearly in the error measure plots (Figures 4.2 and 4.3).

This peculiar result requires an explanation, since it appears to violate the predictions of the theory we have presented. We know that for large σ values, smoothing error will dominate and postaliasing will be minimized. As σ decreases, postaliasing error increases and smoothing error decreases. At some point this appears to reach a minimum where both smoothing and postaliasing error are minimized. Above the minimum, smoothing error dominates whereas below the minimum, postaliasing error dominates. By the theory presented earlier in the chapter, for a relatively-compressive warping function, smoothing error is always less for warped reconstruction while the postaliasing errors are the same, since it is not signal-dependent, but only dependent on the kernel itself. It appears that in the case of the warped reconstruction method, this minmum is reached sooner, for a larger σ value.

In order to explain this deviation from the theory, it must be noted that the warped reconstruction formula (Theorem 2) is not a uniform sampling theorem for a nontrivial warping function, such as the arclength function used for these experiments. It is only uniform in terms of the warped signal, *i.e.* $f(\alpha(t))$. The local rate of nonuniform sampling for the warped formula case will be determined by $\gamma'(t)$, the local rate of signal distortion introduced by the warping function. However, in the case of these experiments, we are using a uniform sampling of the signals. For a uniformly sampled signal, areas where $\gamma'(t)$ is significantly greater than 1 will be undersampled relative to areas where $\gamma'(t)$ is close to 1, when considered in terms of the warped reconstruction formula. This undersampling increases the aliasing error above that which is predicted by the error theory presented in the previous subsection.

Therefore, in practice, the aliasing error of warped reconstruction is greater than that of the classical reconstruction. Since the $f(\alpha(t))$ is under-sampled, its effective bandwidth must be lowered. For larger σ, this aliasing is relatively insignificant compared to smoothing error. However, as σ decreases, the smoothing error tends to become smaller (though this is not necessarily a monotonic relationship for a normalized Gaussian kernel) while the postaliasing error becomes larger and more significant than the smoothing error. In the case of the warped reconstruction, this postaliasing error is greater and it is not offset by the fact that the warped smoothing error is provably less than or equal to the classical smoothing error. As shown in Figures 4.2 and 4.3, the classical reconstruction error becomes less due to the fact that, in practice, the warped reconstruction has greater error due to postaliasing.

Our reconstruction theorem predicted less error for warped reconstruction versus classical for a given particular kernel. This breaks down in practice for kernels with very little smoothing error where the warped reconstruction does not get the expected non-uniform sampling input. This sampling induced aliasing then becomes more significant than the savings in smoothing error provided by the warped reconstruction formula.

One advantage of the warped approach is that one can choose a large, wide reconstruction kernel and not suffer as severely from oversmoothing or blurring reconstruction; it is more “foolproof” in some sense. This property also comes in handy in terms of reconstructing and filtering in the presence of noise, as we will show in the following section.

4.3 Time-Warped Noise Filtering

Thus far in this chapter, we have showed that time-warping can improve reconstruction. The reasoning behind this is deceptively simple: the warped reconstruction formula reconstructs the α-warped signal, then inverts the re-parameterization. The α-warped signal contains the same total energy as the original signal but with a greater fraction of energy represented in terms of lower frequencies when a relatively compressive warping function is used. Practical reconstruction filters resemble the theoretically perfect reconstruction filter more closely at lower frequencies and hence this warped reconstruction technique reduces reconstruction error.

We consider this fact in light of using warped filtering for noise removal. In Chapter 3, we showed that the arclength warped scale space was better able to adapt to local bandwidth spikes and thus is able to preserve large edge features. It stands to reason that this would also provide an effective edge-preserving noise reduction technique.

Along these lines, one of the results of Boulanger [6] was that geodesic distance-based Gaussian filtering was shown experimentally to have better edge-preservation properties than standard Gaussian filtering. As noted previously, in terms of one-dimension, geodesic distance becomes arc-length and can be viewed through the time-warping framework explored in this thesis. In order to augment these experimental results, we seek an exact theoretical error model.

4.3.1 Theoretical Analysis of Time-Warped Noise Reduction

Noise Removal by Gaussian Filtering

Consider a signal f that has been corrupted by an additive noise,

$$\hat{f}(t) = f(t) + n(t) \tag{4.28}$$

where $\hat{f}(t)$ is the observed signal, $f(t)$ the original "true" signal and n is the noise perturbation. Assume that n is Gaussian white noise with a zero mean and a variance v^2.

It is known that low pass, Gaussian filtering can significantly reduce Gaussian white noise. Of course, even if the noise is reduced, the act of filtering itself will also inevitably distort the underlying ground truth signal. This distortion is known as *method noise*. One way to model this is as follow:

Theorem 4 *(Gabor 1960) [8] The image method noise of the convolution with a Gaussian kernel g_h is $f(t) - f(t) \star g_h = -h^2 \Delta f(t) + o(h^2)$.*

This result shows that the method noise of the Gaussian smoothing at a particular signal sample is roughly proportional to the second derivative of the function. Method noise increases with the variance of the Gaussian kernel used, yet this must be balanced by setting the kernel wide enough such that noise is reduced by averaging. For a given width h, the method noise varies locally with the second derivative. Method noise would be close to zero where the second derivative is near zero, but would be much higher at interesting feature points where the second derivative is large.

In contrast, we may also consider an expression that evaluates the sum of the error over the whole signal:

$$\int |f(t) - f(t) \star g_h| dt = \int |F(w) - F(w) G_h(w)| dw = s(f, g_h) \tag{4.29}$$

Notice that this is the same as the smoothing metric used previously to measure smoothing error for reconstruction. In this case, it is the smoothing error introduced by filtering with the Gaussian filter g_h.

Noise Removal by Time-Warped Gaussian Filtering

Consider a noisy signal $\hat{f}(t) = f(t) + n(t)$. Let $\alpha(t)$ be a warping function. Then $\hat{f}(\alpha) = f(\alpha(t) + n(\alpha(t))$. However, $n(\alpha(t))$ is still a i.i.d zero mean noise function with variance v^2. In other words, after warping, the function is still corrupted by zero-mean Gaussian noise with the variance v^2.

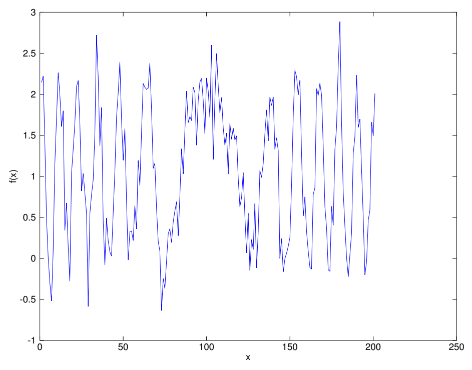
We may consider time-warped convolution with a Gaussian filter in order to surpress noise. One can evaluate the method noise of this technique similarly. Using the Gabor result above, one can see that:

$$\begin{aligned} f(t) - (f(\alpha(t)) * g_h(t))(\gamma(t)) &= f(\alpha(t)) - f(\alpha(t)) * g_h \\ &= -h^2 \Delta f(\alpha(t)) + o(h^2) \\ &= -h^2 (f''(\alpha(t)\alpha'(t) + \alpha''(t) f'(\alpha(t)) + o(h^2) \end{aligned}$$

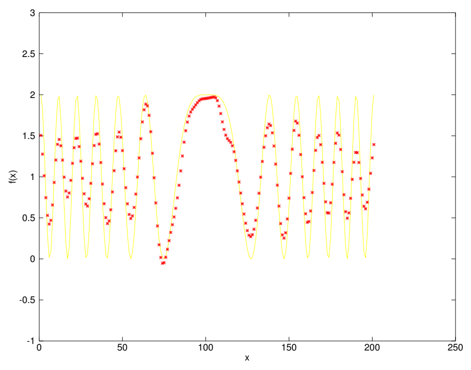
There does not appear to be an obvious way to use this result to show that warping would reduce method noise on a pointwise basis. Therefore, we consider the total method noise error associated with warping smoothing:

$$\int |f(t) - (f(\alpha(t)) * g_h(t)|_{\gamma(t)}| dt = \int |f(\alpha(t) - f(\alpha(t)) \star g_h| dt = s(f(\alpha(t), g_h) \tag{4.30}$$

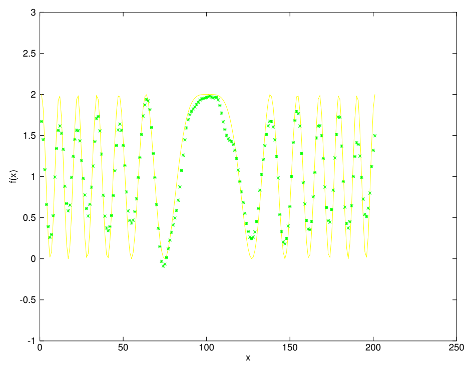
This is the same as the smoothing error metric used in terms of analyzing time-warped reconstruction previously. We know that smoothing error is less in terms of a relatively-compressive warping function and a low-pass filter (Theorem 3). The



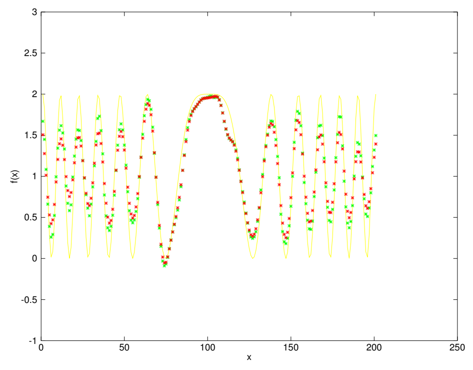
(a) Noisy signal



(b) Denoised with Gaussian filter $\sigma = 6.0$



(c) Denoised with time-warped Gaussian filter $\sigma = 6.0$



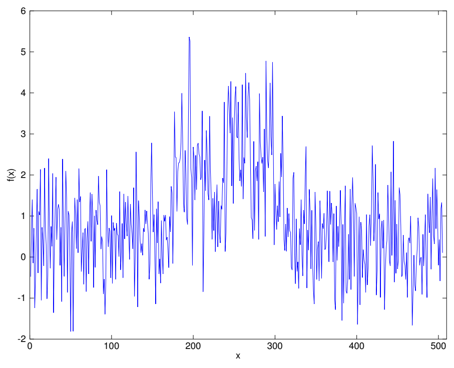
(d) Overlay for comparison

Figure 4.6: Synthetic data denoising using classical (red) and warped convolution (green) versus ground truth signal (yellow)

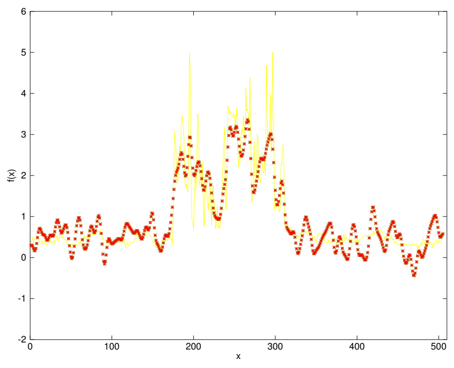
same result implies that the total method error is less when filtering is performed with a bandwidth-normalized relatively compressive warping function.

Theorem 5 *Let $\hat{f}(t) = f(t) + n(t)$. Let $\alpha(t)$ be a relatively compressive warping function relative to f. Then $\int |(\hat{f}(\alpha(t)) * g_h(t)|_{\gamma(t)} - f(t)|dt \leq \int |\hat{f}(t) * g_h(t) - f(t)|dt$*

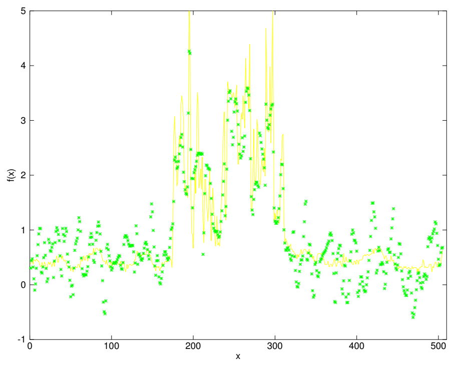
Proof: Follows from fact that $n(\alpha(t))$ is still a i.i.d zero mean noise function with variance v^2 and Theorem 3. ▮



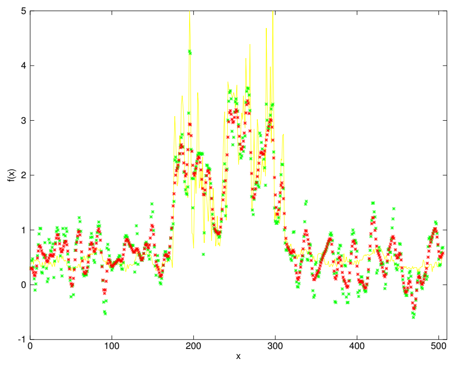
(a) Noisy signal



(b) Denoised with classical filter $\sigma = 6.0$

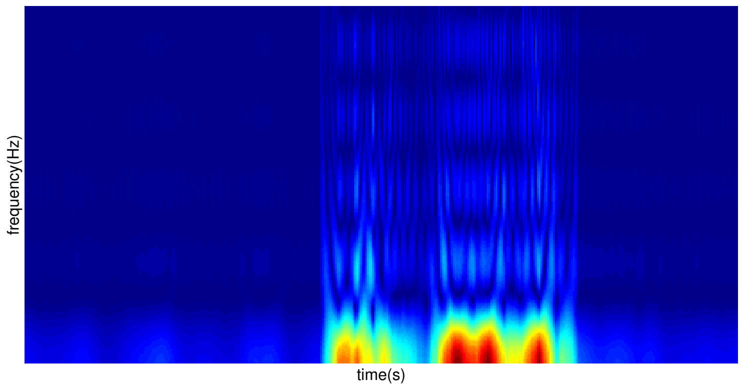


(c) Denoised with time-warped filter $\sigma = 6.0$

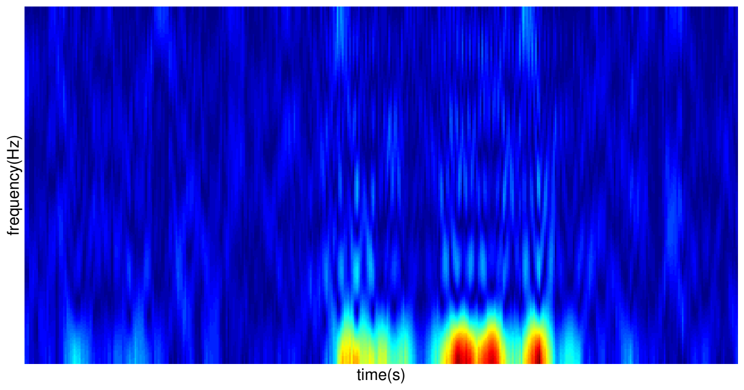


(d) Overlay for comparison

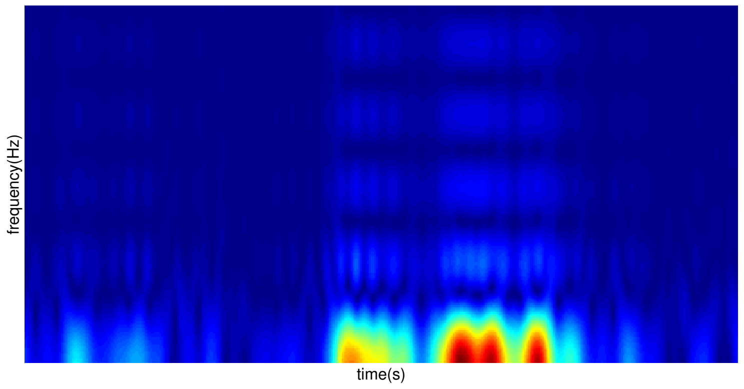
Figure 4.7: Real data denoising using classical (red) and warped convolution (green) versus ground truth signal (yellow)



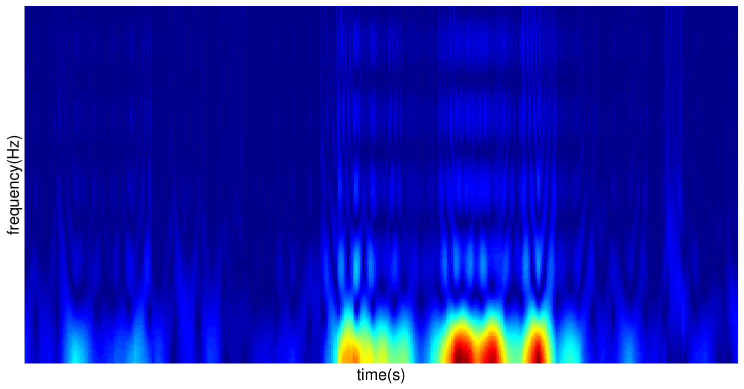
(a) Original signal



(b) Noisy signal



(c) Denoised with classical filter $\sigma = 6.0$



(d) Denoised with time-warped filter $\sigma = 6.0$

Figure 4.8: Spectrogram of real data denoising results

4.3.2 Experimental Results

We consider for completeness whether improved noise removal of relatively compressive warping functions also holds in practice. We add Gaussian zero-mean white noise with a variance 0.3 to our test signals. We then filtered with a Gaussian filter of size $\sigma = 6.0$. We show noise removal results for both synthetic and real data (Figures 4.6 and 4.7). We observe that there is a significant decrease in total method noise error relative to the original signal (Tables 4.3 and 4.4). This seems to confirm the prediction of Theorem 5 that the time warped filtering with a relatively compressive warping function decreases the total sum of error.

Table 4.3: Total method noise error of noise removal technique for synthetic dataset

	Total Error
Classical Filtering	51.597
Noisy Warped Filter	43.334
Ground truth Warped Filter	34.58

Table 4.4: Total method noise error of noise removal technique for real dataset

	Total Error
Classical Filtering	104.29
Noisy Warped Filter	91.241
Ground truth Warped Filter	83.04

One can see how these noise removal techniques operate in terms of the local aspects of the signal spectrum by looking at their spectograms (Figure 4.8). The noise is uniform across the time-frequency representation of the signal (compare Figure 4.8 a) and b)). One can see that warped filtering adapts to the local bandwidth spikes of the signal and preserves them (Figure 4.8 d)) whereas they can be seen to be flatter and attenuated in the classical filtering case (Figure 4.8 c)).

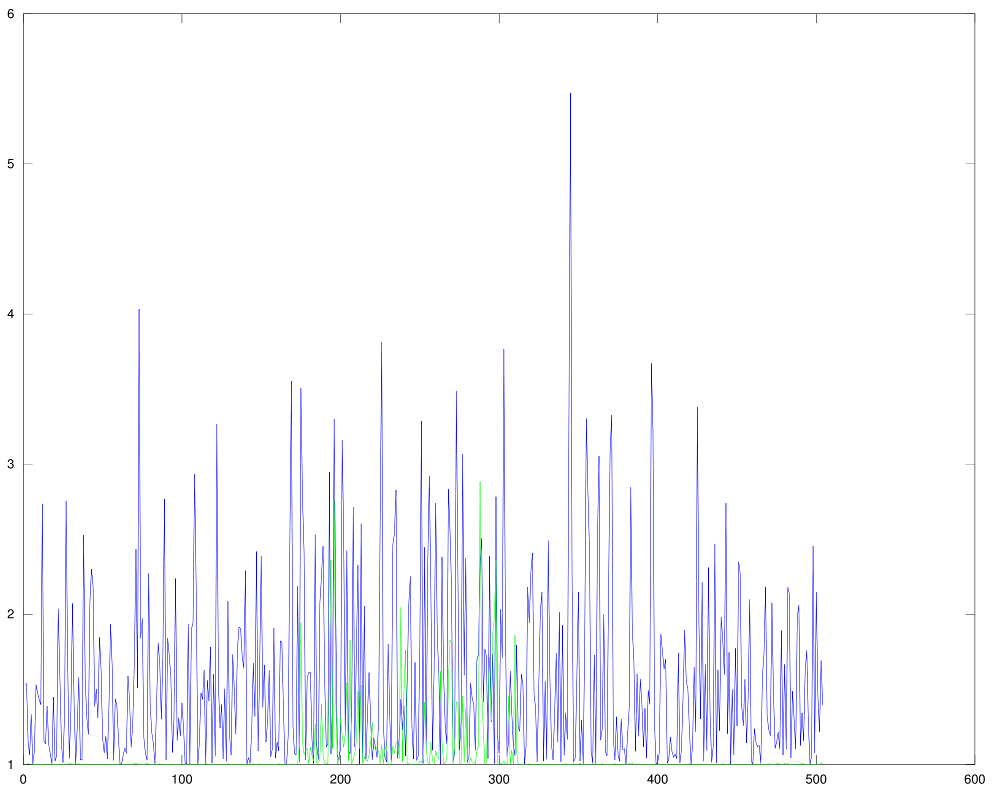


Figure 4.9: Plot of noisily-reconstructed $\gamma'(t)$ (blue) and the ground truth $\gamma'_{gt}(t)$ (green)

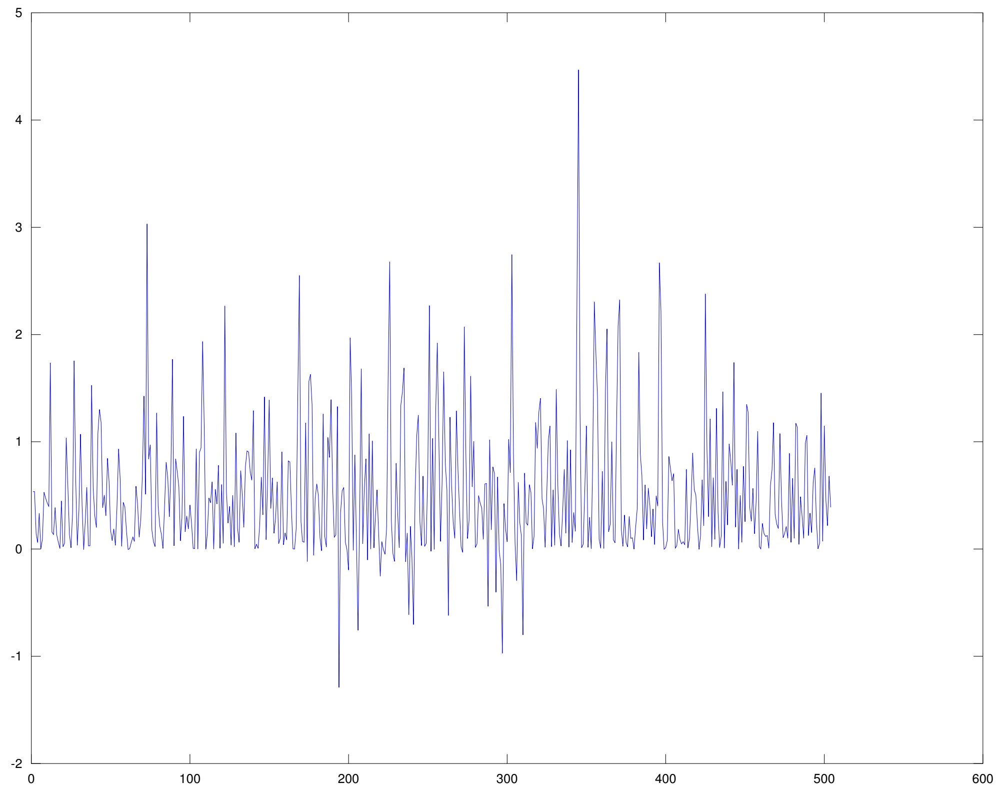


Figure 4.10: Plot of $p = \gamma'(t) - \gamma'_{gt}(t)$

Warping Function Computation from Noise-Corrupted Signal

This adaptation is not surprising given the results we seen in Chapter 3 in Figure 3.4, where the edge preservation of warped scale space was shown to have a windowed spectrum interpretation. The only difference between this result and that of Figure 3.4 is that here the warping function is estimated from the noisy signal instead of the ground truth. Theorem 5 assumes that we have the warping function of the ground truth signal. Of course, in practice, the warping function of the ground truth signal is not available. Given that the experimental results of Tables 4.3 and 4.4 seem to match the theory, it is tempting to conclude that warping function computation is not significantly affected by the presence of noise.

We first ask the question, how effective is this noise removal technique when the ground truth warping function is known? We can see in Tables 4.3 and 4.4 that filtering with the ground truth warping function gives a further decreases the smoothing error over the one based on the noisy signal. Clearly, accurately estimating the warping function from the noisy signal is important to achieving effective noise removal.

We examine more precisely how noise affects this warping function computation. As the noise variance is increased, the noisy warping function has more error and the total method noise error increases above that of the ground truth warping function. We see in Table 4.5 that as noise variance increases, the effectiveness of the noisy warped filter diverges from the ground truth warped filter. Moreover, once noise levels are sufficiently high, it appears that the noisy warped filter actually performs worse than the classical filter. This is not surprising. Obviously, the error of the warping function computation relative to the ground truth warping function will increase with noise variance, eventually distorting to the point where it no longer performs better than a classical, unwarped method.

We ask more precisely how does the noise effect the warping function. We

Table 4.5: Total method noise error of noise removal technique relative to various noise levels for the real dataset.

σ_n	0.3	0.5	0.7
Noisy Warped Filter	83.514	114.62	192.91
Ground truth Warped Filter	83.275	107.56	163.99
Classical Filtering	96.960	120.68	172.13

consider the derivative of a warping function vs. the ground truth warping function. It seems that in practice, the warping function computed from the noisy function tends to mostly overestimate the actual local bandwidth (Figures 4.9 and 4.10). We have no formal theoretical proof of this. However, it seems reasonable that adding random noise to a function will, on average, increase the magnitude of the derivative of a function, hence increasing the local bandwidth as estimated by the arclength warping function. We can see that in some cases, it in fact decreases this derivative and causes an under-estimate of the local bandwidth, but this appears to only occur for a very small number of samples in practice (See Figure 4.10). Since for higher noise levels this overestimation becomes more significant, the warped filter tends to under-smooth, preserving noise rather than removing it (Figure 4.11). Eventually, the warped method actually begins to perform worse since over-preserving noise incurs a worse error penalty than over smoothing significant edges.

4.4 Summary

In this chapter, we have made a compelling case that time-warping with an appropriate signal-dependent warping function can serve as a powerful tool to create a set of nonlinear basis functions for signal reconstruction and filtering. We showed that time-warping can improve convolution-based reconstruction. The improvement can be proven to be guaranteed in theory. This improvement comes at the expense of a relatively small amount of extra computation and without a significant change in the parallelization properties of the resulting algorithms. The chief reason for this

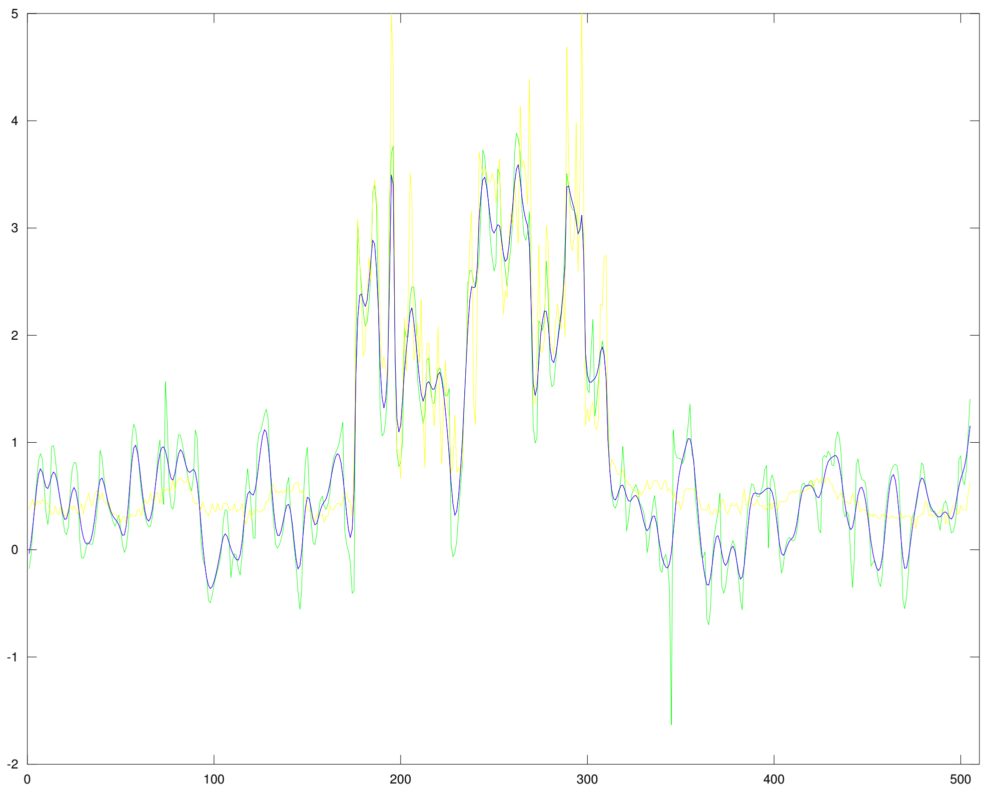


Figure 4.11: Plot of various filtered real data signals with noise added at variance 0.9 ; yellow is ground truth; blue is signal filtered with classical filter, green is signal filtered with warped filter

improvement in reconstruction is due to the fact that warping with an appropriate warping function transforms signal energy into lower frequencies relative to the bandlimit. This also implies that time-warped filtering for the removal of Gaussian noise results in less distortion of the underlying signal with the same noise removal potential.

In practice, we observe experimental limits to these improvements. For reconstruction, this is due to the fact that different signal sampling rates are expected in theory for the classical and warped cases. We find that in practice for a Gaussian kernel, the theory holds up for large σ values but does not for smaller σ values because the uniform sampling does not match the non-uniform sampling expected by the warped reconstruction formula, thus resulting in aliasing error that is not predicted by the theory.

For noise removal, our practical results in time-warped removal deviate from theory because we assume the ability to perfectly reconstruct the ground truth warping function. In practice, this is reasonable only when noise levels are relatively low. When noise levels are high, the reconstructed warping function deviates from the theoretical ground truth. We showed that experimentally, it seems that with increasing noise levels, the warping function tends to be reconstructed such that noise is over-preserved when warped filtering is applied, thus making the linear classical methods in fact more effective.

There are many more questions along this line of research. The most obvious is in exploring different warping functions; perhaps arclength is not optimal. The results of this chapter imply that restricting candidate warping functions to be relatively compressive is a useful starting constraint. The other limitation of these results is that we are restricted to one dimensional signals. Extending the theoretical and practical results to higher-dimensions would be extremely useful for many practical multdimensional datasets.

Chapter 5

Real-Time Multiscale Molecular Surface Reconstruction

Molecular dynamics datasests, particularly those corresponding to protein folding simulations, provide a massive dataset which spans vast spatiotemporal scales relating to a still relatively poorly understood physical phenomenom. Protein folding simulations are extremely expensive computationally; in fact, only relatively small proteins may be simulated completely into their folded state within reasonable times on modest supercomputing platforms. Moreover, the datasets produced contain many time steps. Thus, a computationally efficient tool is required in order to analyze the many time-steps in a real-time or interactive fashion.

In this chapter, we present an implementation of a GPU-based molecular visualization tool that allows for a highly interactive, user-adjustable representation of the data at multiple scales. This can be achieved by leveraging the massive parallel computational capabilities of GPUs in accordance with an efficient parallelization and implementation strategy.

Though there are existing solutions to this problem, our approach differs in several fundamental ways. Our approach avoids intermediate 3D representations during the rendering process, producing the final 2D image in a single pass over memory to retrieve the data. Hence, intermediate volumetric representations or polygonal representations are not required to be computed from the given input atomic location data of a molecular dynamics simulation. This allows for an effective combination of high-quality and flexible (in terms of transfer function) volume rendering of the resulting molecular electron density field with very good interactivity and high-performance properties. Moreover, this direct approach which avoids intermediaries turns out to likely be also appropriate for computing time-warped reconstructions for molecular dynamics datasets.

5.1 Multiscale Molecular Surfaces

5.1.1 Molecular Representations

Standard molecular data is composed at the core of 3D atom locations, atomic densities, and bonds between atoms, along with various other information. It is a scattered, particle-based dataset with bonds representing inter-particle relationships.

The most obvious mapping for particle data is to display particles as points in 3D space, also known as a 3D scatter plot. Though 2D scatterplots can sometimes be effective in showing larger features in data, this sort of mapping in 3D does not work well [69], given a lack of depth cues from perspective, occlusion or lighting/shading sources. Moreover, there is no obvious means to represent the scalar value attached to the point, other than by color.

Some improvement can be achieved by representing points as spheres. This representation provides perspective and occlusion cues that allow for some extra perceptual discrimination while still conveying detail at the finest level of resolution. Moreover, a scalar value can be represented by the size of the sphere. For molecular data, this form of representation corresponds to the classical ball and stick molecular models or van-der-Walls surface models, where the size of the sphere representing atoms is based on the van-der-Wals radius. These can be an effective representation for visualizing molecules and rendering can scale very well when rendered efficiently and at high quality using modern GPU techniques [59].

Molecular surfaces are chiefly useful in terms of visualizing interactions with the boundaries of molecules, *i.e.* the outward facing part of the molecular that is exposed to other molecules. The most commonly used representation here is the Solvent Excluded Surface (known as SES) [43]. This surface is defined by rolling a variable size probe sphere over the van-der-Walls surface. The size of the spherical probe atom decides in some sense the scale of the solvent interaction surface. However, since it is based on a hard-sphere assumption, the resulting surface can still contain discontinuities. Typically these surfaces are tesselated then rendered, however this is difficult to achieve in real-time for a temporal dataset from a molecular dynamics simulation. One approach uses GPU accelerated raycasting and a parallel computation of a simplified SES surface formulation [32, 30]. They achieve the ability to handle time-varying data in real-time and the ability to change the probe molecule size in real time. However, the acceleration introduces view-dependence and some visual artifacts.

The other main alternate strategy for defining a molecular surface is based on isocontours of a more physically realistic electron density model (soft sphere), in contrast to the more approximate hard sphere atom model. One approach along these lines [15] uses Gaussian basis functions to represent this electron density function. This representation allows one to explore the Gaussian scale space of the density field, where the choice of scale size is roughly analogous to the spherical probe size in the SES representation.

In terms of implementation, this corresponds to a convolution-based reconstruction in three-dimensional space, which has relatively nice parallelization properties. Krone et al.[31, 44] implemented this form of surface using Gaussian basis functions . This work re-samples the density field onto a regular grid as a pre-processing step, where it may then be rendered using standard volume rendering techniques. This, however, adds a step of re-sampling onto a regular grid which is expensive since it requires reads and writes from global memory. For example, Krone *et al.* [31] report that for 4,378 atoms this step required 0.2 seconds to generate a 256x256x256 volume on an NVIDIA Geforce GTX260, while Krone et. al [44] report that this step is the most costliest in their approach. Moreover, this re-sampling step can introduce artifacts depending on the chosen re-sampling resolution. In order for this step to be efficient enough to allow real-time rendering for a dynamic molecular dataset, the chosen re-sampling resolution must be relatively low in order to avoid a significant memory reading and writing penalty, thus also increasing the chances of introducing artifacts.

In terms of rendering, one approach [31] simply directly volume renders this re-sampled uniform volume using existing fast volume rendering techniques. Other approaches are based on meshes in that the isosurfaces of the density field must be extracted as a triangular mesh from the re-sampled grid [44]. While this approach was shown to achieve real-time performance for an interactively adjustable multi-scale surface, it can only be achieved when the sampling grid and isourface mesh are sufficiently sparse, potentially leaving artifacts and lack of smoothness. Moreover, the mesh rendering approach produces just a rendering of an isosurface and cannot handle arbitrary transfer functions.

5.2 GPU-Based Multiscale Molecular Surface Reconstruction

We describe the GPU implementation strategy for an interactive multiscale molecular surface reconstruction and rendering system. We use NVIDIA CUDA for the GPU parallelization and describe the implementation in terms of the CUDA parallel programming model. The soft sphere model of a molecular surface allows us to reconstruct a scalar density field, represented by scattered point-based samples. In order to render this, we require a high-performance, point-based volume rendering code capable of handling a time-varying dataset.

Unlike existing approaches, we propose that by using the shared memory capabilities of CUDA and NVIDIA GPUs we can combine rendering and reconstruction, thus requiring no intermediate representations. This allows for a higher-quality rendering than the intermediary approaches since there is no loss or artifacts due to re-sampling and/or interpolation. Also, we may volume render the electron density field using any transfer function and we are not limited to only iso-surface rendering. We carefully design the algorithm in order to maximize performance.

5.2.1 Dataflow Analysis and Mapping to the CUDA Model

As we discussed in the Chapter 2, the chief issue with leveraging the massive parallel computing potential of GPUs is that of achieving a suitable parallelization and mapping of the problem onto the GPU hardware. Specifically, an efficient mapping to the hardware will parallelize the problem such that memory bandwidth bottlenecks can be minimized. Initially, it is useful to analyze the problem in terms of its dataflow or communication pattern. The precise nature of the dataflow can then inform one how to map onto hardware.

In terms of the NVIDIA CUDA GPU parallel programming model, separate concurrent tasks are assigned to threads. Each thread executes on its own processor. Processors are grouped into units known as multi-processors. Processors within the same multi-processor share a user-manged cache (known hereto as shared memory) as well fixed functionality caches.

In terms of communication, threads may be required to load input data from the global memory (slow DRAM). This we will refer to as *intra-thread communication* hereforth. Furthermore, threads may then be required to intercommunicate amongst themselves. Threads can intercommunicate chiefly in two ways: (1) By reading and writing data from and to global memory in a synchronized fashion or (2) communicating through user-managed shared memory. We refer to this as *inter-thread communication*. Hence, minimizing communication time involves optimizing both inter-thread and intra-thread communication.

Under the CUDA framework, parallel computations are viewed in terms of separate threads. These threads are grouped into *work blocks* according to a decomposition scheme such that threads of the same work block are executed together on the same multiprocessor [1]. The implementation design challenge here is to choose a grouping of threads into work blocks that yields optimal performance.

Threads in the same work block utilize a common shared memory. In terms of optimizing intra-thread communication, shared memory can be used to cache memory locations that are either (i) repeatedly accessed by a single thread or (ii) repeatedly accessed by multiple threads, thus overall reducing slow accesses to global memory. In terms of intra-thread optimization, the key thing to determine about a parallel algorithm is memory access coherence within a thread and between threads; caching should be utilized in order to exploit this coherence.

In terms of optimizing inter-threads communication, shared memory can be used as a mutually-accessible scratch pad with which threads of a single work-block can communicate. The key thing to be determined here is the pattern of thread communication with regards to the given parallel algorithm.

Once an algorithm has been formulated as parallel, concurrent threads, these threads must then be grouped into work blocks. A good work block grouping must be choosen so that (i) threads which must intercommunicate are grouped in the same work block as often as possible and (i) threads which access memory coherently are grouped as often as possible.

Our implementation appoach is described in the following subsections. It is made up of two separate CUDA kernels, one for spatial grid reconstruction and the other to compute the actual volume rendering of the dataset. The structure of the

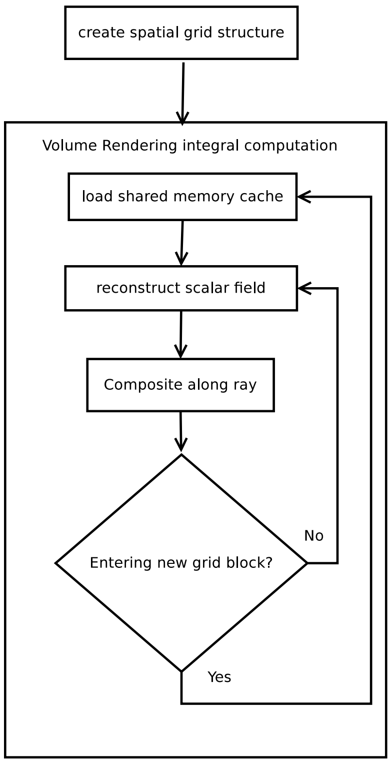


Figure 5.1: Block diagram of CUDA implementation. Spatial grid creation and volume rendering are comprised of two separate CUDA kernels.

implementation is summarized in the block diagram of Figure 5.1.

5.2.2 Spatial Data Structure

One key component for reconstruction and filtering of point-based datasets is a spatial data structure which greatly accelerates neighborhood computations. One of the major improvements of GPU architectures that allows for a more efficient point-based volume rendering implementation is scattered write capability. As mentioned earlier, the previous best attempts [47, 46] created and traversed spatial data structures necessary for efficient performance on the CPU, transferring point data over the PCI bus to the GPU during rendering. One can now overcome this bottleneck by creating the spatial structure on the GPU using the scatter capability. This allows us to work exclusively with data stored in GPU memory instead of having to involve

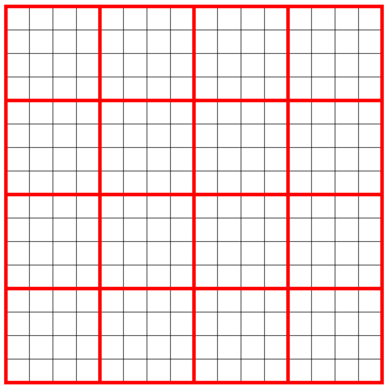


Figure 5.2: Workblock grouping of pixel-based threads

the CPU and its associated memory.

For our point-based volume renderer, we use a uniform grid with implementation ideas similar to those used by Green [20]. Though a uniform grid does not guarantee a bounded number of points in any 3x3x3 grid block neighborhood, it can be shown that under a hard sphere assumption where the range of sphere sizes is bounded, good performance can be achieved for neighbor searching [27]. Often, the point-based data does not correspond to spheres in a meaningful sense, but is sampled within a bounded range of rates. Hence, the uniform grid performs well for neighbor searching if the grid size is chosen correctly. Indeed, this has proven to be a sensible assumption in physical simulation such as particle-based fluid simulations [23].

5.2.3 Ray-casting Computation and Scalar Field Reconstruction

Once the given point-data has been placed into a point-based data structure, ray-casting volume rendering may be computed. The basic algorithm we use is as follows. We first must index the given points in terms of a uniform grid spatial data structure. We use a fixed grid in world coordinates which is aligned to the screen viewplane. The individual grid blocks of the spatial structure can be indexed by a three-dimensional positive integer vector (x, y, z)

Once the spatial data-structure has been created, ray-casting computation of the volume rendering integral can be performed. A single ray is cast for each pixel and we propose that a single CUDA thread is assigned to handle only one ray. Under an orthogonal projection, the spatial grid projects uniformly as a two-dimensional grid on the camera view plane. Since the spatial data-structure is aligned with the camera plane, the $(x, y, 0)$ plane projects directly onto the camera plane and subdivides it at a super-pixel level. This provides a convenient way to group spatially-neighboring rays (mapped to threads) into work blocks (Figure 5.2).

Though this mapping is convenient, one also can show it allows for an efficient implementation. Since ray computations are "trivially parallel" and thus completely independent, there is no inter-thread communication needed for this rendering approach. With regards to intra-thread communication, we know that there is memory access coherence that can be exploited. We have inter-ray coherence in terms of spatially adjacent rays in the same work-block as they share mutual neighbors. Moreover, along a single ray, samples are re-used along adjacent time-steps. We can see this coherence clearly in Figure 5.3.

A work block of threads handling spatially-adjacent rays begins to execute a kernel to compute the volume rendering integral. Each work block computes rays that lie within an interval of grid blocks (x_p, y_p, z) for fixed values of x_p and y_p and all z within the range of values for the given space grid. A single thread iterates over sampling positions along a ray in a front-to-back order, moving from grid block to grid block. At the first sampling step in each grid block, one thread of a work block (thread 0) initially fills the shared memory cache with the data that will be required for scalar field reconstruction then synchronizes execution with the other threads of the work block (blue positions in Figure 5.3). In the case of the classical molecular surface, this location and density values of the atoms within a 3x3x3 grid neighborhood must be cached.

We describe more precisely how this cache filling is performed. For all samples within a particular space grid block, we require all the neighboring samples within the immediate 3x3 grid block neighborhood to be in cache. Once the current ray sampling postiton moves into the next grid block (a blue position in Figure 5.3), it is required that the cache be updated. It must be noted that at these blue positions, a 3x3x2 subset of the previously cached neighborhood may remain in the cache, since adjacent space grid blocks have significant overlap in their 3x3x3 neighborhood. Thus, at each blue position, we are only required to load new neighbors into the cache from the front-most 3x3x1 space grid blocks now within the adjacent 3x3x3 neighborhood.

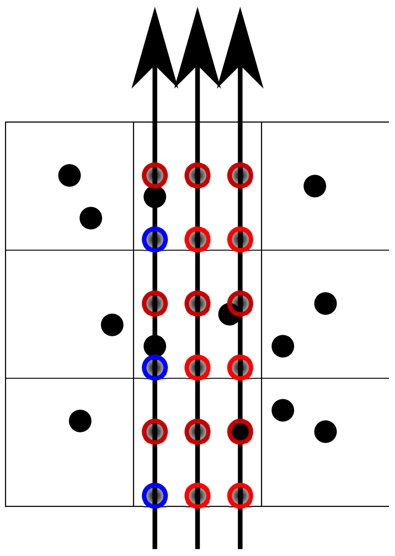


Figure 5.3: Diagram of ray-casting

After the neighborhood cache has been filled at a particular ray-cast sampling step, it is then time to reconstruct the scalar field at the sampling point. The CUDA kernel on a per-thread basis iterates through the cached set of neighbors and computes the reconstruction according to the classical convolution based formula,

$$f(x) = \sum_{k \in \mathbb{Z}} c_k \phi(x-k) \tag{5.1}$$

where we choose ϕ to be a normalized gaussian kernel. If we utilize isosurface rendering, surface intersection tests are performed and a gradient is computed for lighting and shading computation purposes. Otherwise, standard ray-compostion formulas used for volume rendering are employed. These are standard practices in any typical volume rendering implementation [60].

5.3 Time-Warped Molecular Surface Definition

Given the interesting properties of time-warped scale spaces and reconstruction described in previous chapters, we consider if it could also be incorporated into this rendering framework. In previous chapters, we explored time-warped filtering and reconstruction for one-dimensional signals. Since typical molecular surfaces can be described through classical convolution-based reconstruction and filtering, a natural question to ask then is whether the advantages of warping can be extended towards molecular surfaces.

Time-warped signal implementation involves first computing a warping function, then the convolution sum. In the case of reconstruction, the warping function has to be additionally interpolated at the resolution of the reconstruction grid; however, simple linear intepolation appears to suffice. This same strategy applies to generalizing warped signal processing to two and higher dimensions when the samples are given on a regular grid.

In the case of molecular dynamics data this is not so. Rather, molecular data is scattered and point-based, with the additional constraint that the atom locations are situated on a one-dimesional subspace defined by the bonding structure. Though the molecule is intrinsically one-dimensional, the atomic density field we aim to reconstruct is a scalar map on a three dimensional space.

It is not immediately clear how to compute a warping function in this case. One could ignore the bonding information and consider atoms as points, then compute a warping function on the three-dimensional domain. However, the point-based nature of the data complicates warping computation and interpolation versus that of a regular grid. Moreover, by ignoring the inherent one-dimensional aspect of the molecule, we would not be able to design a warping function that allows scale space filtering whereby edge features of the one-dimensional molecular skeleton are preserved.

5.3.1 3D Warping Function for Molecular Reconstruction

It is worth noting that a molecular bonding structure has fixed geodesic distances between atoms, regardless of the particular orientation of the atoms. This intuitively seems like it could form the basis for a warping function. Since these distances are fixed, a distance matrix could be precomputed. This matrix could be loaded into the shared memory cache on a row basis. Hence, instead of the classical convolution reconstruction formula being computed in the kernel, the time-warped analogue could then be computed. This warping function would have to be interpolated for each sampling point along a ray. This is not trivial given that the molecule bonding structure is one-dimensional, but the reconstruction space is multidimensional. Moreover, molecules may have non-trivial topologies.

We present a potential warping function for a molecule that can preserve edges or discontinuities in both *(*i) the one-dimensional molecular skeleton and *(ii)* the atomic density field. For these purposes, we assume we are given a set of 3D points that represent atoms $A = \{a_i = (x_i, y_i, z_i)\}$ and a set of bonds $B = \{(a_i, a_j) | a_i, a_j \in A\}$ and a density value $f(a_i)$ for each $a_i \in A$.

It is relatively straighforward to define a warping function on the one-dimensional molecular skeleton, formally the set of of atoms and bonds; we simply apply the same ideas as have been discussed in the previous chapters regarding one-dimensional signals. First, we observe that the pair $M = (A, B)$ make up a graph. We may define a distance metric between atoms based on the shortest paths through this graph; it could be said that these minimal paths are the geodesics of the molecular skeleton. We can further refine this metric by augmenting the graph with edge weights based on the density values, $f(a)$. In order to match the one-dimensional, arclength-based warping functions in previous chapters, we set the first order differences of the density values on a bond's endpoints as edge weights. This will create a compressive warping effect on the signal as in the previous chapters. Thus, for $a_i, a_j \in A$, we let $dist_M(a_i, a_j)$ be the shortest path distance on the edge-weighted graph induced by the molecular skeleton. This distance function can be interpolated in order to also cover points that lie on a bond: For a point I on a bond $b = (a_i, a_j)$,

$$dist_M(I,a) = min(||I - a_i|| + dist_M(a_i,a), ||I - a_j|| + dist_M(a_j,a)) \tag{5.2}$$

However, this one-dimensional warping function must be generalized for space points that lay adjacent to the molecular skeleton. To this end, we require the following definitions (Figure 5.4):

Definition 5 *Consider a bond $b = (a_i, a_j) \in B$ and a point P. Let $V = a_i - a_j$, $R = P - a_j$, $T = \frac{V\dot{R}}{||V||}$ and $I = A + T\frac{\dot{V}}{||V||}$. We say that P is in the span of bond b if I is on the line segment between a_i and a_j and we denote this by $P \in span(B)$, where $span(B)$ is the set of all points in the span of B.*

We may project a point P onto the line induced by a bond b and determine whether it is in the span. If P is not in the span of b, we define the projection of P onto b as the closest of the endpoints of b to P, *i.e.* whichever of a_i and a_j that minimizes $||P - a_i||$ and $||P - a_j||$, respectively. More formally:

$$proj_b(P) = \begin{cases} I & : P \in span(B) \\ min(||P - a_i||, ||P - a_j||) & : P \notin span(B) \end{cases} \tag{5.3}$$

For our purposes, it is also important to know the distance from P to the corresponding point of projection on the bond. In the case that $P \in span(b)$, $D(P,b) = ||IB - P||$. Otherwise, $D(P,b) = min(||IB - a_i|, ||IB - a_j||)$.

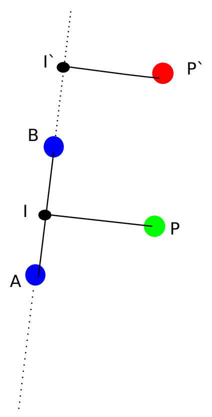


Figure 5.4: A and B form the endpoints of a molecular bond. P projects onto the line induced by A and B and is in the span of the bond. P' projects outside the span of the bond.

This provides the basis to define a potentially useful molecular warping function. It is important to note that in order to compute the warped convolution integral, we only require *differences* of the inverse warping function at samples and reconstruction points, as opposed to the value at a given point (see Equation 3.2). Hence, we need only define the distance between P, a reconstruction point, and $a \in A$, an atom. The distance is defined as:

$$dist(P,a) = min_{b \in B}\{D(P,b) + dist_M(I,a)\}. \tag{5.4}$$

To summarize, we get the following reconstruction formula. For a reconstruction point $P = (x,y,z)$, we can reconstruct the *geodesically warped atomic density field*, S by the equation:

$$S(P) = \sum_{i=1}^{i=n} d(a_i) g_\sigma(dist(P,a_i)). \tag{5.5}$$

5.3.2 Experimental Evaluation of Geodesic Warping Function

We have to note that this formula above is not strictly a warping-based filtering. The geodesic distance filtering of Boulanger [6], similarly is not strictly a warping function since the geodesic distance does not have a euclidean embedding, in general. For the molecular distance function given above (Equation 5.4), there does not exist a Euclidean embedding that generally either. We do know that approximate Euclidean embeddings can be found [7] and hence it is possible to analyze and understand the effect of these distance functions on signal reconstruction and filtering in terms of warping, at least approximately.

The notion that the local rate of change of the warping function chooses the local bandwidth of the signal can intuitively be seen to also apply in the case of a non-warping, distance function used for filtering and processing. In the case of our molecular distance function, the local rate of change of the distance function determines how and where edges and discontinuties shall be preserved with respect to multiscale filtering.

The proposed distance function (Equation 5.4) has two main aspects of edge preservation. Edges in the molecular bonding structure are preserved and edges in the atomic density field are preserved. Similar to the case of relatively compressive warping functions examined in the previous chapters, this distance automatically detects these edges and adapts the local rate of change of distance accordingly.

We examine a simple, synthetic two-dimensional, three atom molecule in order to demonstrate this more clearly (Figure 5.5). First, we suppose that these three atoms have a constant density. We can then reconstruct atomic density fields using the classical (Euclidean distance) and "warped" (molecular weighted geodesic distance) reconstruction methods (Figure 5.6). In Figure 5.6 b), we can see that the edges of the bonding structure are "carved" or preserved better by the non-Euclidean distance function. However, it appears to also introduce small discontinuities that are distracting.

This should not be surprising since the distance function is only piecewise smooth. We may also experiment with smoothed versions of the distance function. We try the following modified distance function where we square the projection

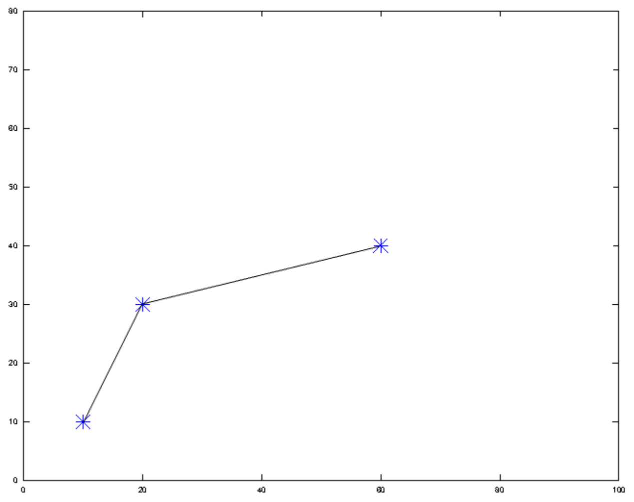


Figure 5.5: Simple two bond, three atom molecule

distance term:

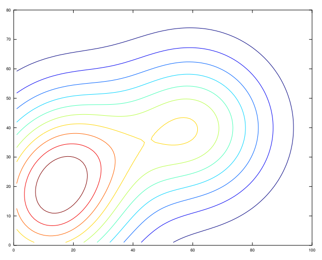
$$dist(P,a) = min_{b \in B}\{(D(P,b))^2 + dist_M(I,a)\} \tag{5.6}$$

We see in Figure 5.6 c) that this yields a much smoother, edge preserving surface.

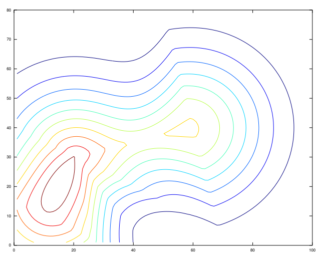
We now experiment with the same synthetic molecule where the atomic densities are not all of uniform value. We choose for the center atom adjacent to the two bonds to have a density 2.5 times the side atoms. We show the result in Figure 5.7. In contrast to our previous examples, the actual topologies of the iso-contours of the geodesic surface are altered as compared to the Euclidean case.

5.3.3 Problems with this Approach

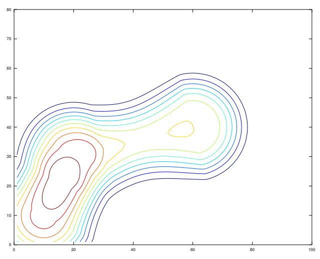
These results we have thus far are encouraging. However, these apply only to a simple molecule with a simple topolgy. The projection-based approach seems to intutively allow one to interpolate the 1-D molecular warping function. When applied to a ring-like molecular structure, this approach does not appear to work correctly. We see that obviously unacceptable artifacts are produced when applied to a



(a) Isocontours of atomic density, classical scale space

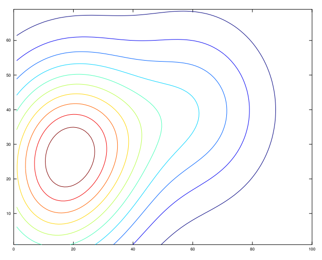


(b) Isocontours of atomic density, geodesic molecular scale space

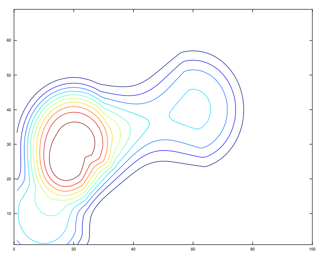


(c) Isocontours of atomic density, geodesic molecular scale space

Figure 5.6: Comparison of molecular surface reconstruction strategies



(a) Isocontours of atomic density, classical scale space



(b) Isocontours of atomic density, geodesic molecular scale space

Figure 5.7: Edge preservation

ring-like molecule (Figure 5.8).

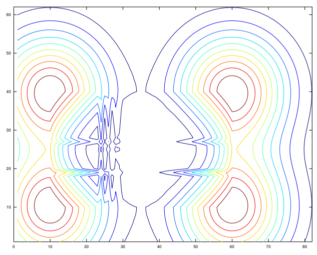


Figure 5.8: Non-simple topology molecule and resulting artifacts of the proposed warping-based approach

Given the apparent success of this idea on the simple topology molecule, it seems there ought to be some way to define a reasonable means of interpolation for this molecule. However, at this point it is not entirely clear how this may be done and it is left for future work.

5.4 Experimental Results

Table 5.1: Rendering framerates for various datasets and screen resolutions using isosurface transfer function

Dataset	250x250	500x500
villin (496 atoms)	170fps	100fps
dna (1049 atoms)	90fps	45fps
PDB:1VIS (2500 atoms)	45fps	15fps
PDB:1TII (5684 atoms)	22fps	7fps
PDB:1AF6 (10,000 atoms)	10fps	3fps

We ran the implemented code on an NVIDIA Geforce 580 GPU. Sizes of the spatial data structure (extent and number of grid subdivisions) were tuned individually for each dataset. There are likely reasonable strategies to automate this task, but since we achieve real-time rates it is quite practical to have these parameters as user

Table 5.2: Rendering framerates for various datasets and screen resolutions using basic cloudy transfer function

Dataset	250x250	500x500
villin (496 atoms)	180fps	170fps
dna (1049 atoms)	140fps	110fps
PDB:1VIS (2500 atoms)	120fps	90fps
PDB:1TII (5684 atoms)	60fps	70fps
PDB:1AF6 (10,000 atoms)	40fps	60fps

tunable and allow the user to adjust them interactively until optimal performance setting is achieved.

We are able to achieve interactive framerates up to moderate size molecules (See Table 5.1). Since the spatial datastructure is created each timestep, the method seemed to roughly perform the same for single and multiple time-step datasets. The result is that we are able to visualize protein folding datasets in real-time at arbitrary scale for moderate size molecules. We can see that the timing of our method is very much dependent on the screen resolution (see Table 5.1). It is also highly dependent on the chosen ray sampling rate. We choose a minimal sampling rate that yields no visible artifacts, which will vary by the size and shape of the molecules and transfer function. For example, for isosurface rendering it seems a ray sampling rate up to 5 times greater than for a basic cloudy transfer function is required in order to generate an image without noticable artifacts. This extra computation is then reflected in the rendering times.

We show the results for several moderate size molecules visualized across a range of scales, from small to medium and then large (See Figures 5.3, 5.4, 5.5). In addition to isosurface renderings, we may also volume render the electron cloud with a transfer function (Figure 5.4).

It is difficult to compare these results in terms of timing performance directly with the best achieved elsewhere in the literature by Krone *et. al* [44]. Though

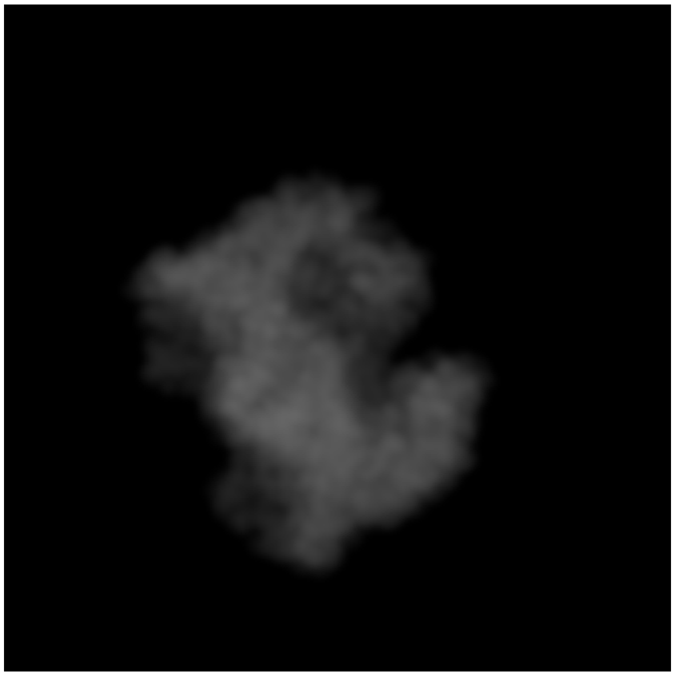
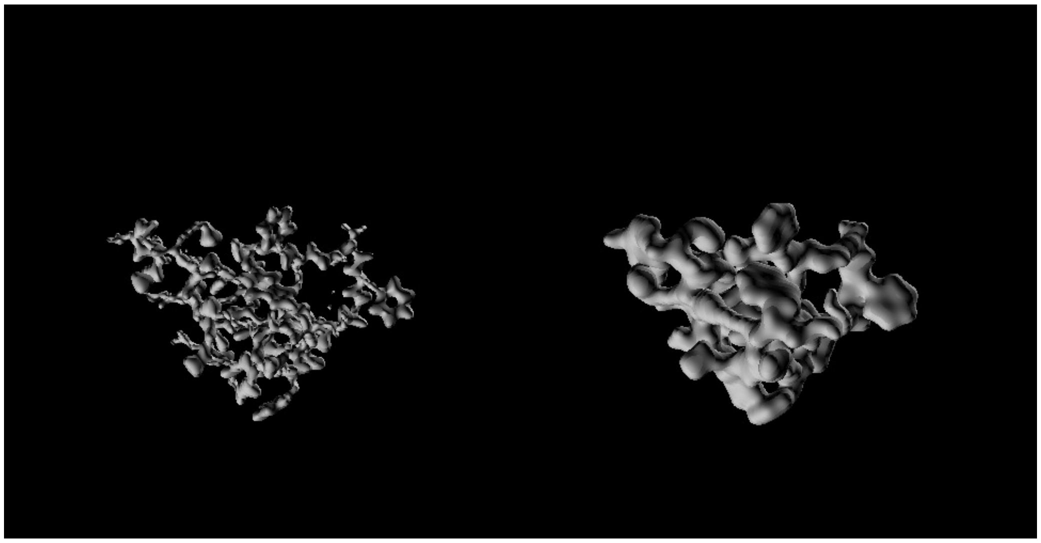


Figure 5.9: pdb:1AF6 rendered with a non-isosurface transfer function

Krone *et. al* appear to achieve greater scalability using the same NVIDIA GPU, cautious comparisons must be made. In order to have a fair comparison of these methods, we would have to consider relatively small screen resolution and ray-sampling rate. Then it could be seen that the re-sampling volume induced by these rays is equivalent in some sense to the one usedin the Krone *et. al* method; the difference then being that we render directly from this resampling step, whereas their method uses a rendering technique that takes the regular volume as input.

Re-sampling onto a relatively small grid will not result in significant loss or artifacts if the width of the Gaussian kernel used for reconstruction is significantly wide. However, in cases where small details must be preserved, this underlying sampling grid must be increased in size which in turn decreases the performance of this method.


(a) villin (b) villin

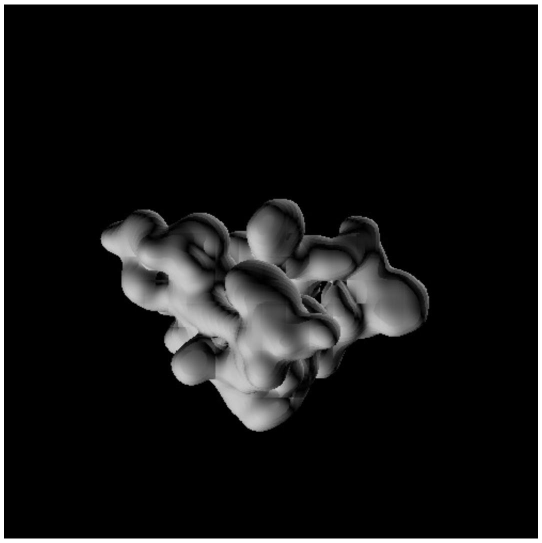
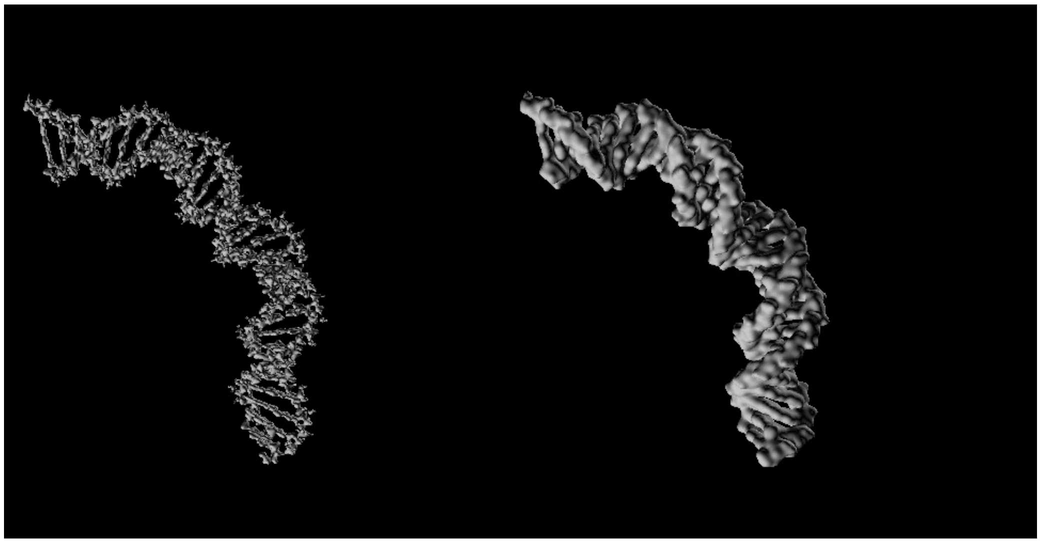
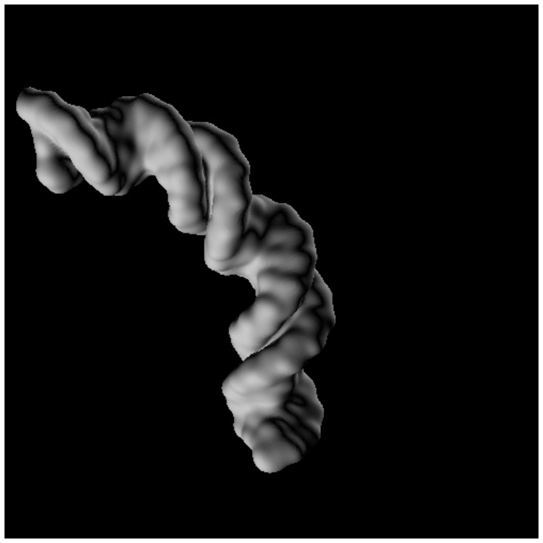

(c) villin

Figure 5.10: villin headpiece at various scales

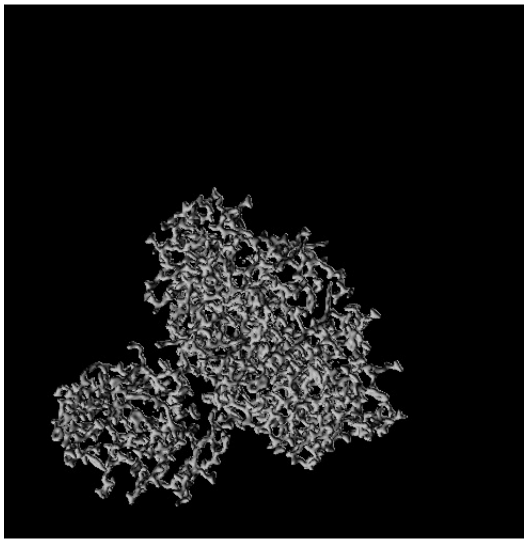


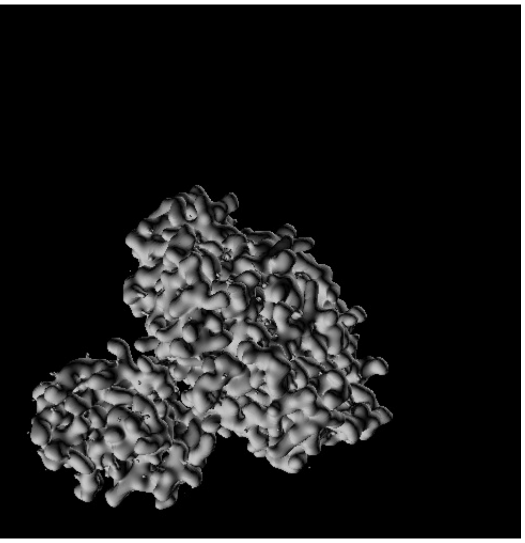
(a) dna (b) dna



(c) dna

Figure 5.11: dna at various scales


(a) 1VII


(b) 1VII

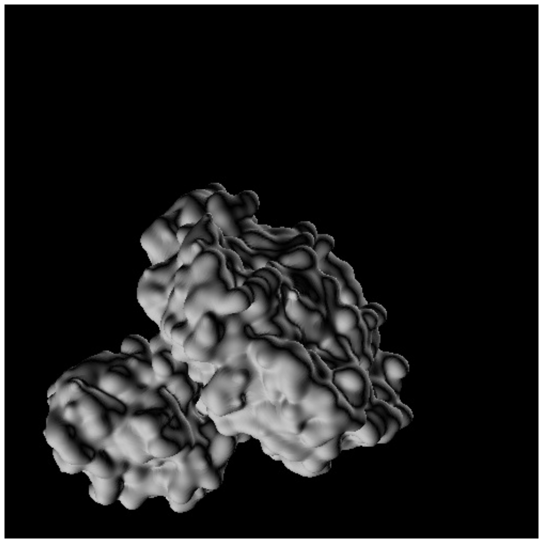

(c) 1VII

Figure 5.12: 1VII at various scales

5.5 Summary and Future Work

In this chapter, we have explored interactive multiscale visualization for real-time molecular dynamics simulation data. We have presented the implementation of a GPU-based multiscale reconstruction and rendering system. This allows the efficient visualization of time-varying molecular dynamics simulations at an interactively adjustable scale, based on a "soft sphere", Gaussian density molecular surface model. Unlike existing real-time methods, no intermediate representation is required, as the electron density field is reconstructed directly from the simulation data during the actual rendering process. The key to achieveing efficiency of this implementation is the use of a GPU-created spatial data structure in order to limit neighbor comptations and effective use of user-managed cache provided by NVIDIA GPUs and exposed through the CUDA programming API.

Compared to other approaches, this allows for high-quality rendering without artifacts and greater flexibility in rendering transfer function while still maintaining a good ability to scale to large datasets at an interactive rate. Perhaps most importantly in terms of this thesis, the implementation approach described in this chapter appears it would be appropriate for implementing a time-warped version of a convolution-based molecular surface, in that the caching scheme could allow the caching of a precomputed molecular warping function and its efficient interactive interpolation.

In terms of future work, these results could be scaled to use multiple GPUs for greater performance and scalability. Also extending this implementation strategy to support the time-warped scale space and reconstruction framework examined in this thesis would provide a very flexible multiscale molecular visualizaton tool.

Chapter 6

Conclusion

In this thesis, we have explored various aspects of multiscale signal processing and visualization. One downfall of standard Fourier analysis is that it is based on uniform oscillators and stationary signals. The time-warped signal framework appears to provide a means to augment existing Fourier tools in order to address non-uniformities in signals. Time-warping essentially allows the Fourier basis to adapt to the signal. Time-warping can be used to augment existing Gaussian scale space theory. We show that warping preserves the causality condition of a scale space and allows local adaptation of the scale filtering. This allows one to produce an edge-preserving scale space filtering that has similar computational complexity to the standard linear scale space formulation.

We further show that warping can provide significant benefits in terms of basic signal processing operations such as signal reconstruction and random noise removal. We show that a class of warping functions known as *relatively compressive* warping functions can be guaranteed in theory to improve reconstruction error and removal of random noise from signals. The result also is shown to hold in practice with the Gaussian filter for both synthetic and real-world signals over a range of different filter variances. It is also shown to fail in practice for a certain sub-range of Gaussian filters.

Finally, we have explored high-performance, interactive multiscale visualization. It is common to utilize a multiscale representation when visualizing molecular dynamics datasets, in particular those corresponding to protein folding simulations. These simulations produce dynamic data at significant throughput. Visualization processes must be able to match this throughput to allow immediate, responsive simulation interaction. We have given a GPU-based implementation of a point-based volume renderer designed to handle time-varying datasets from a molecular dynamics simulation. The implementation presented avoids producing an intermediate representation, allowing volume rendering of the electron density cloud at user-selected scale using an arbitrary transfer function. This enables the visual exploration of data and at user-selectable spatio-temporal scale in a highly interactive fashion and could be coupled directly to a live simulation in order to facilitate a computational steering system. Perhaps most importantly for this thesis, the presented implementation seems that it would be suitable to provide an ability to interactively visualize a time-warped multiscale molecular surface with the features of a time-warped scale space as presented in Chapter 3, once the problem of how to compute a suitable warping function in real-time could be solved. It seems promising that the molecular bonding structure based on bonds of fixed length could be the basis of interesting and efficiently computable warping function. This is left for future work.

Many other interesting future directions are suggested by the results of this thesis. Time-warped signal processing itself provides many avenues for future work. Extending the time-warped scale space into higher dimensions are obviously one interesting avenue. For example, we do not know how to extend these results into two-dimensions relative to the geodesic distance function used by Boulanger [6]. Though signal processing based on this distance metric appears to behave very much like warping and would likely admit many of the various theoretical results presented in this thesis, this still remains uncertain. If such a result could be proven then it follows that the interesting reconstruction and noise removal results would have analogous results in two and higher dimensions.

Beyond this problem, the choice of warping function itself presents many possible future directions. Choosing an effective warping function that is also efficiently computable is crucial in order to apply the ideas in this thesis into higher dimensional spatio-temporal datasets in domains where multiple scales become important, for example in fluid dynamics data and seismic data where phenomenon such as self-similarity and scale-separation are known to exist. We hypothesis in Chapter 5 that there are domain specific physical constraints in the molecular dynamics domain which may possibily be exploited in order to partially pre-compute a warping function for a molecular electron density cloud. For all of these cases, appropriate GPU acceleration techniques must be utilized in order to achieve good performance.

Bibliography

[1] Nvidia cuda compute unified device architecture programming guide, 2008. Version 2.0.

[2] Mukerji T. Bergbauer, S. and P. Hennings. Improving curvature analyses of deformed horizons using scale-dependent filtering techniques. *American Association of Petroleum Geologists Bulletin*, v. 87:pp. 1255–1272., 2003.

[3] R. P. Bording and A. Atle. Purpose Built Computers for the Wave Equation. *AGU Fall Meeting Abstracts*, pages C718+, December 2007.

[4] Ralph Phillip Bording. *Wave equation difference engine*. PhD thesis, University of Tulsa, 1995.

[5] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-Quality Surface Splatting on Today's GPUs. In *Proceedings of Symposium on Point-Based Graphics*, pages 17–24, 2005.

[6] Pierre Boulanger. Multiscale edge detection based on a new geometrically intrinsic filter. In *Proceedings SPIE*, volume 2350, pages 264–278, 1994.

[7] Alexander Bronstein, Michael Bronstein, and Ron Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[8] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Self-similarity-based image denoising. *Commun. ACM*, 54(5):109–117, May 2011.

[9] Martin D. Buhmann. *Radial Basis Functions*. Cambridge University Press, New York, NY, USA, 2003.

[10] Chang-Chieh Cheng and Yu-Tai Ching. Transfer function design for fourier volume rendering and implementation using gpu. volume 6918, page 691806. SPIE, 2008.

[11] J. Clark, M. Palmer, and P. Lawrence. A transformation method for the reconstruction of functions from nonuniformly spaced samples. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 33(5):1151 – 1165, oct 1985.

[12] Christopher S. Co and Kenneth I. Joy. Isosurface generation for large-scale scattered data visualization. In Guenther Greiner, Joachim Hornegger, Heinrich Niemann, and Marc Stamminger, editors, *Proceedings of VMV 2005*, pages 233–240, 2005.

[13] Andrew Corrigan and John Wallin. Visualization of meshless simulations using Fourier volume rendering. In *Proc. of the ECCOMAS Thematic Conference on Meshless Methods*, pages 65–70, July 2007.

[14] Remco Duits, Luc Florack, Jan de Graaf, and Bart M. ter Haar Romeny. On the axioms of scale space theory. *Journal of Mathematical Imaging and Vision*, 20(3):267–298, 2004.

[15] B. S. Duncan and A. J. Olson. Shape analysis of molecular surfaces. *Biopolymers*, 33(2):231–238, February 1993.

[16] Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, 1994.

[17] Kayvon Fatahalian, Jeremy Sugerman, and Pat Hanrahan. Understanding the efficiency of gpu algorithms for matrix-matrix multiplication. In *Graphics Hardware*, pages 133–137, 2004.

[18] G. Gilboa, N. Sochen, and Y.Y. Zeevi. Forward-and-backward diffusion processes for adaptive image enhancement and denoising. *Image Processing, IEEE Transactions on*, 11(7):689 – 703, jul 2002.

[19] Naga K. Govindaraju, Scott Larsen, Jim Gray, and Dinesh Manocha. A memory model for scientific algorithms on graphics processors. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 89, New York, NY, USA, 2006. ACM Press.

[20] Simon Green. Cuda particles. NVIDIA CUDA SDK, June 2008.

[21] Michael Gschwind, H. Peter Hofstee, Brian K. Flachs, Martin Hopkins, Yukio Watanabe, and Takeshi Yamazaki. Synergistic processing in cell's multicore architecture. *IEEE Micro*, 26(2):10–24, 2006.

[22] Stefan Gumhold. Splatting illuminated ellipsoids with depth correction. In *VMV*, pages 245–252, 2003.

[23] Takahiro Harada, Seiichi Koshizuka, and Yoichiro Kawaguchi. Smoothed particle hydrodynamics on gpus. In *Proceedings of Computer Graphics International*, 2007.

[24] J. Edward Swan II, Marco Lanzagorta, Doug Maxwell, Eddy Kuo, Jeff Uhlmann, Wendell Anderson, Haw-Jye Shyu, and William Smith. A computational steering system for studying microwave interactions with missile bodies. In *IEEE Visualization*, pages 441–444, 2000.

[25] Yun Jang, Ralf P. Botchen, Andreas Lauser, David S. Ebert, Kelly P. Gaither, and Thomas Ertl. Enhancing the Interactive Visualization of Procedurally Encoded Multifield Data with Ellipsoidal Basis Functions. *Computer Graphics Forum*, 25(3):587–596, 2006.

[26] Yun Jang, Manfred Weiler, Matthias Hopf, Jingshu Huang, David S. Ebert, Kelly P. Gaither, and Thomas Ertl. Interactively visualizing procedurally encoded scalar fields. In *VisSym*, pages 35–44, 339, 2004.

[27] Dong Jin Kim, Leonidas J. Guibas, and Sung Yong Shin. Fast collision detection among multiple moving spheres. *IEEE Trans. Vis. Comput. Graph.*, 4(3):230–242, 1998.

[28] J.J. Koenderink. The structure of images. *Biological Cybernetics*, Vol. 50:pp. 363–370, 1984.

[29] Shankar Krishnan, Cludio T. Silva, and Bin Wei. A hardware-assisted visibility-ordering algorithm with applications to volume rendering. In *In Data Visualization (2001*, pages 233–242, 2000.

[30] M. Krone, K. Bidmon, and T. Ertl. Interactive Visualization of Molecular Surface Dynamics. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2009)*, 15(6):1391–1398, 2009.

[31] M. Krone, M. Falk, S. Rehm, J. Pleiss, and T. Ertl. Interactive exploration of protein cavities. *Computer Graphics Forum*, 30(3):673–682, 2011.

[32] Michael Krone, Carsten Dachsbacher, and Thomas Ertl. Parallel computation and interactive visualization of time-varying solvent excluded surfaces. In *Proceedings of the First ACM International Conference on Bioinformatics*

and Computational Biology, BCB '10, pages 402–405, New York, NY, USA, 2010. ACM.

[33] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):1053–1060, Sept.-Oct. 2006.

[34] Christian Ledergerber, Gael Guennebaud, Miriah Meyer, Moritz Bacher, and Hanspeter Pfister. Volume mls ray casting. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1546, 2008.

[35] Warren Leung, Neophytos Neophytou, and Klaus Mueller. Simd - aware raycasting. In *Proceedings of Volume Graphics 2006*, 2006.

[36] Marc Levoy. Volume rendering using the fourier projection-slice theorem. In *Proceedings of the conference on Graphics interface '92*, pages 61–69, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[37] Yanchao LI. Oriented particles for scientific visualization. Master's thesis, University of New Brunswick, 1997.

[38] Kwan-Liu Ma. Machine learning to boost the next generation of visualization technology. *IEEE Computer Graphics and Applications*, 27(5):6–9, 2007.

[39] William R. Mark and Donald Fussell. Real-time rendering systems in 2010. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 19, New York, NY, USA, 2005. ACM.

[40] Stephen R. Marschner and Richard J. Lobb. An evaluation of reconstruction filters for volume rendering. In *VIS '94: Proceedings of the conference*

on Visualization '94, pages 100–107, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[41] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995.

[42] Michael Meiner, Ulrich Hoffmann, and Wolfgang Straer. Enabling classification and shading for 3d texture mapping based volume rendering using opengl and extensions. In *In Proc. of IEEE Visualization*, pages 207–214. IEEE, 1999.

[43] Paul Mezey. *Shape in Chemistry*. VCH Publishers, 1993.

[44] Thomas Ertl Michael Krone, John E. Stone and Klaus Schulten. Fast visualization of gaussian density surfaces for molecular dynamics and particle system. In *In Proceedings of EuroVis 2012*, 2012.

[45] C. Mller, S. Grottel, and T. Ertl. Image-Space GPU Metaballs for Time-Dependent Particle Data Sets. In *Proceedings of VMV '07*, pages 31–40, 2007.

[46] N. Neophytou, K. Mueller, K. T. McDonnell, W. Hong, X. Guan, H. Qin, and A. Kaufman. Gpu-accelerated volume splatting with elliptical rbfs. In *Joint Eurographics - IEEE TCVG Symposium on Visualization*, 2006.

[47] Neophytos Neophytou and Klaus Mueller. GPU Accelerated Image Aligned Splatting. In *Volume Graphics*, pages 197–205, 2005.

[48] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krger, Aaron E. Lefohn, and Timothy J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[49] Sung Park, Lars Linsen, Oliver Kreylos, John D. Owens, and Bernd Hamann. A framework for real-time volume visualization of streaming scattered data. In Marc Stamminger and Joachim Hornegger, editors, *Proceedings of Tenth International Fall Workshop on Vision, Modeling, and Visualization 2005*, pages 225–232. DFG Collaborative Research Center, November 2005.

[50] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, Jul 1990.

[51] Timothy J. Purcell, Craig Donner, Mike Cammarano, Henrik Wann Jensen, and Pat Hanrahan. Photon mapping on programmable graphics hardware. In *Graphics Hardware*, pages 41–50, 2003.

[52] G. Reina and T. Ertl. Hardware-Accelerated Glyphs for Mono- and Dipoles in Molecular Dynamics Visualization. In K. W. Brodlie and D. J. Duke and K. I. Joy, editor, *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization 2005*, pages 177–182, 2005.

[53] Patrick Reuter, Ireneusz Tobor, Christophe Schlick, and Sébastien Dedieu. Point-based modelling and rendering using radial basis functions. In *GRAPHITE*, pages 111–118, 2003.

[54] B.M. Romeny. *Front-End Vision and Multi-Scale Image Analysis: Multi-scale Computer Vision Theory and Applications, written in Mathematica*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[55] Paul Rosenthal and Lars Linsen. Direct isosurface extraction from scattered volume data. In *EuroVis*, pages 99–106, 2006.

[56] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[57] David E. Shaw, Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J. P. Grossman, C. Richard Ho, Douglas J. Ierardi, István Kolossváry, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C. Wang. Anton, a special-purpose machine for molecular dynamics simulation. *Commun. ACM*, 51(7):91–97, 2008.

[58] C. Sigg and M. Hadwiger. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, chapter Fast third-order texture filtering. Addison Wesley.

[59] Christian Sigg, Tim Weyrich, Mario Botsch, and Markus Gross. GPU-Based Ray-Casting of Quadratic Surfaces. In *In Proceedings of Eurographics Symposium on Point-Based Graphics*, pages 59–65, 2006.

[60] Simon Stegmaier, Magnus Strengert, Thomas Klein, and Thomas Ertl. A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *Volume Graphics*, pages 187–195, 2005.

[61] Carsten Stoll, Stefan Gumhold, and Hans-Peter Seidel. Incremental raycasting of piecewise quadratic surfaces on the gpu. In *Proceedings of IEEE Symposium on Interactive Raytracing*, 2006.

[62] John Stone, Justin Gullingsrud, Paul Grayson, and Klaus Schulten. A system for interactive molecular dynamics simulation. In John F. Hughes and Carlo H. Séquin, editors, *2001 ACM Symposium on Interactive 3D Graphics*, pages 191–194, New York, 2001. ACM SIGGRAPH.

[63] Rodrigo Toledo and Bruno Lvy. Extending the graphic pipeline with new gpu-accelerated primitives. *Techreport*, 2004.

[64] Michael Unser. Sampling50 years after shannon. pages 569–587, 2000.

[65] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. Berendsen. Gromacs: fast, flexible, and free. *J Comput Chem*, 26(16):1701–1718, December 2005.

[66] Kees van Kooten, Gino van den Bergen, and Alex Telea. *Point-Based Visualization of Metaballs on a GPU*, chapter Chapter 7, pages 123–148. NVIDIA, 2007.

[67] Jarke J. van Wijk. The Value of Visualization. In *IEEE Visualization*, page 11, 2005.

[68] Erald Vuçini, Torsten Möller, and Meister Eduard Gröller. On visualization and reconstruction from non-uniform point sets using b-splines. *Computer Graphics Forum*, 28(3):1007–1014, June 2009. 2nd Best Paper Award.

[69] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, second edition, 2004.

[70] J. Weickert, B.M.T.H. Romeny, and M.A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *Image Processing, IEEE Transactions on*, 7(3):398–410, Mar 1998.

[71] Manfred Weiler, Martin Kraus, Markus Merz, and Thomas Ertl. Hardware-based ray casting for tetrahedral meshes. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 44, Washington, DC, USA, 2003. IEEE Computer Society.

[72] Lee Westover. Footprint evaluation for volume rendering. In *SIGGRAPH*, pages 367–376, 1990.

[73] Peter L. Williams and Nelson Max. A volume density optical model. In *VVS '92: Proceedings of the 1992 workshop on Volume visualization*, pages 61–68, New York, NY, USA, 1992. ACM.

[74] Andrew P. Witkin. Scale-space filtering. In *IJCAI*, pages 1019–1022, 1983.

[75] Wm. A. Wulf and Sally A. McKee. Hitting the memory wall: implications of the obvious. *SIGARCH Comput. Archit. News*, 23(1):20–24, 1995.

[76] Kamen Yotov, Thomas Roeder, Keshav Pingali, John A. Gunnels, and Fred G. Gustavson. An experimental comparison of cache-oblivious and cache-conscious programs. In *SPAA*, pages 93–104, 2007.