

**CONTROL OF SPATIAL AND TEMPORAL FIDELITY WITH  
ADAPTIVE SAMPLING**

by

Abraham J. Stephens

A dissertation submitted to the faculty of  
The University of Utah  
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

May 2011

Copyright © Abraham J. Stephens 2011

All Rights Reserved



# The University of Utah Graduate School

## STATEMENT OF DISSERTATION APPROVAL

The dissertation of Abraham J. Stephens

has been approved by the following supervisory committee members:

Steven G. Parker, Chair 3/15/2011  
Date Approved

Peter Shirley, Member 4/12/2011  
Date Approved

Claudio Silva, Member 4/12/2011  
Date Approved

Benjamin Watson, Member 4/5/2011  
Date Approved

David Luebke, Member 4/8/2011  
Date Approved

and by Alan L. Davis, Chair of  
the Department of Computer Science

and by Charles A. Wight, Dean of The Graduate School.

## **ABSTRACT**

Balancing the trade off between the spatial and temporal quality of interactive computer graphics imagery is one of the fundamental design challenges in the construction of rendering systems. Inexpensive interactive rendering hardware may deliver a high level of temporal performance if the level of spatial image quality is sufficiently constrained. In these cases, the spatial fidelity level is an independent parameter of the system and temporal performance is a dependent variable. The spatial quality parameter is selected for the system by the designer based on the anticipated graphics workload. Interactive ray tracing is one example; the algorithm is often selected due to its ability to deliver a high level of spatial fidelity, and the relatively lower level of temporal performance is readily accepted.

This dissertation proposes an algorithm to perform fine-grained adjustments to the trade off between the spatial quality of images produced by an interactive renderer, and the temporal performance or quality of the rendered image sequence. The approach first determines the minimum amount of sampling work necessary to achieve a certain fidelity level, and then allows the surplus capacity to be directed towards spatial or temporal fidelity improvement. The algorithm consists of an efficient parallel spatial and temporal adaptive rendering mechanism and a control optimization problem which adjusts the sampling rate based on a characterization of the rendered imagery and constraints on the capacity of the rendering system.

To my family

# CONTENTS

<b>ABSTRACT</b> .....	<b>iii</b>
<b>ACKNOWLEDGMENTS</b> .....	<b>ix</b>
<b>CHAPTERS</b>	
<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1 Interactive Rendering .....	2
1.2 Sampling .....	3
1.3 Spatial and Temporal Fidelity .....	4
1.3.1 Definition of Fidelity .....	4
1.3.2 Fidelity Management Related to Image Compression .....	5
1.3.3 Fidelity and Vision .....	7
1.3.4 Suitability of Computer Graphics Imagery .....	8
1.4 Thesis Statement .....	11
1.5 Contributions .....	11
1.5.1 Real-time Adaptive Frameless Rendering .....	11
1.5.2 Separable Response .....	12
1.5.3 Spatial and Temporal Fidelity Model .....	12
1.5.4 Scalable Parallel Architecture .....	12
1.6 Applications .....	12
1.6.1 Ray Tracing and Hybrid Rendering .....	12
1.6.2 High Resolution Displays .....	13
<b>2. RELATED WORK</b> .....	<b>14</b>
2.1 Sampling .....	14
2.2 Reconstruction .....	15
2.3 Adaptive Methods in Graphics .....	17
2.3.1 Rasterization .....	18
2.3.2 Adaptive Textures .....	18
2.3.3 Progressive Anti-aliasing .....	20
2.3.4 Adaptive Rendering .....	21
2.3.5 Video Encoding .....	21
2.4 Taxonomy of Fidelity Trade Offs .....	23
2.4.1 Independent .....	28
2.4.1.1 Free Temporal .....	28
2.4.1.2 Free Spatial .....	29
2.4.2 Spatially Dependent .....	29
2.4.2.1 Temporal for Spatial .....	29
2.4.2.2 Spatial for Spatial .....	31
2.4.3 Temporally Dependent .....	32
2.4.3.1 Spatial for Temporal .....	32

2.4.3.2	Temporal for Temporal	33
2.4.4	Combined	35
2.5	Adaptive Frameless Rendering	35
<b>3.</b>	<b>TEMPORALLY AND SPATIALLY ADAPTIVE RENDERING</b>	<b>37</b>
3.1	Adaptive Sampling	39
3.1.1	Sample Density Field	39
3.1.2	Framelet Tiling	41
3.1.2.1	Tiling Alternatives	43
3.2	Sample Rate Error Estimation	44
3.2.1	Characterization of Redundancy	44
3.3	Adaptive Response Decision	45
3.3.1	Constraints on Sample Density	46
3.3.1.1	Positivity and bounds	46
3.3.1.2	Conservation	47
3.4	Software Architecture	48
3.4.1	Parallel Hardware	48
3.4.2	Manta Interface	49
3.4.2.1	Timing	50
3.4.2.2	Transactions and Callbacks	50
3.4.2.3	Image Display	51
3.4.2.4	Ray Tracing	52
3.4.3	MantaGL Interface	52
3.4.3.1	Animation	53
3.4.3.2	Interoperation	54
3.4.3.3	Image Transfer	55
3.4.4	TSAR Interface	55
3.4.4.1	Tile Selection	55
3.4.4.2	Rendering Stack	57
3.4.4.3	Reconstruction	57
3.4.4.4	Framelet Cache Update	58
3.4.4.5	Error Estimation	59
3.4.4.6	Density Solver	59
3.4.4.7	Tile Update	59
<b>4.</b>	<b>ADAPTIVE SAMPLE PLACEMENT</b>	<b>60</b>
4.1	Spatial Structures	60
4.1.1	Spatial Tiling	61
4.1.2	Morton Curve	61
4.1.3	Morton Order Tree	61
4.1.3.1	Tree Traversal	61
4.1.3.2	Quad-tree	63
4.1.3.3	Interval Operations	63
4.1.3.4	Resampling Between Uniform Resolutions	64
4.1.4	Depth First Tree	64
4.1.4.1	Resampling Between Tilings	66
4.1.5	Summed Area Table	66
4.1.6	Tiling Construction	67
4.1.6.1	Serial Construction	67
4.1.6.2	Data Parallel Construction	68

4.2	Framelet Reconstruction	69
4.2.1	Reconstruction Artifacts	69
4.2.2	Algorithm	70
4.2.3	Temporal Filtering	71
4.2.4	Spatial Filtering	74
<b>5.</b>	<b>ERROR ESTIMATION</b>	<b>76</b>
5.1	Sample Rate Error	76
5.1.1	Schematic Interpretation	77
5.1.2	Ideal Sample Density	77
5.1.2.1	Frequency Domain Interpretation	79
5.1.2.2	Error Estimation	82
5.2	Spatiotemporal Domain Approximations	83
5.2.1	Variance	83
5.3	Frequency Domain Approximations	84
5.3.1	Localization	85
5.3.1.1	Wavelet Transform	86
5.3.1.2	Spatiotemporal DWT	89
5.3.1.3	Leading Edge of Time	89
5.3.2	Localized Levelwise Norm	90
5.3.3	DWT Error Estimation	91
5.3.3.1	Signed LLN Difference	92
5.3.3.2	Threshold Function	93
5.3.3.3	Computing the LLN	95
5.3.4	Streaming Discrete Wavelet Transform	96
<b>6.</b>	<b>ADAPTIVE RESPONSE CONTROL DECISION</b>	<b>100</b>
6.1	Decision Problem	100
6.1.1	Schematic Interpretation	102
6.1.2	Proportional Response	103
6.2	Local Constraints	104
6.2.1	Response Speed	106
6.2.2	Maximum Change in Density	107
6.3	Density Transport	110
6.3.1	Global Constraints	111
6.3.1.1	Conservation	111
6.3.1.2	Relational Linear Constraints	113
6.3.1.3	Example Solutions	116
6.3.2	Two-Dimensional Functional Solutions	117
6.3.2.1	Schematic Interpretation	120
6.3.2.2	Certainty	122
6.3.2.3	Functional Solutions	125
6.3.2.4	Limitations	126
6.3.3	Relational Heuristics	127
6.4	Density Mixture	128

<b>7. EXPERIMENTAL RESULTS</b>	<b>130</b>
7.1 Suitability	131
7.1.1 Graphics Scene Input and Algorithm Parameters	131
7.1.1.1 Test Scenes	132
7.1.1.2 Parameters	134
7.1.2 Cost Model	135
7.1.2.1 Model Parameters	136
7.1.2.2 Simplified Model	138
7.1.3 Cost Constraint	139
7.2 Quantitative Evaluation	141
7.2.1 Measuring Quantitative Quality	141
7.2.2 Reconstruction Artifacts	144
7.2.3 Comparison Pipeline	144
7.2.4 Quality Index Behavior	146
7.2.5 Suitability Results	147
7.3 Qualitative Evaluation	152
7.3.1 Individual Scene Analysis	153
7.3.2 Response Sensitivity	162
<b>8. CONCLUSION</b>	<b>167</b>
8.1 Interpretation	168
8.2 Characteristics of Appropriate Workloads	170
8.3 Algorithms	171
8.3.1 Spatiotemporal Sampling with Framelets	172
8.3.2 Reconstruction	173
8.3.3 Adaptive Response Model	174
8.4 Future Work	175
<b>REFERENCES</b>	<b>177</b>

## ACKNOWLEDGMENTS

I was very fortunate during my time in Utah to be assisted by several mentors who guided me through graduate school. The work presented in this dissertation is principally the product of a collaboration with my advisor, Steven Parker, and committee members, David Luebke and Ben Watson. We discussed this work over the telephone on a nearly weekly basis for several years. It is difficult to express my gratitude and admiration for my advisor Steven Parker. As long as I have known him, Steve has had an impossibly ambitious schedule, but he has always found time for me, even after starting a company and leaving the University. The research for this dissertation was conducted at the Scientific Computing and Imaging Institute. The research environment Chris Johnson and the faculty of SCI have created was an essential ingredient in my education. My collaborators at SCI included James Bigler, Ingo Wald, Christiaan Gribble, Lee Butler, Solomon Boulos, and many others. Peter Shirley was an excellent teacher who forged my understanding of computer graphics and provided me with straightforward and practical advice throughout my time in Utah. Pete was instrumental to my finishing this work after I left campus. Claudio Silva provided me with guidance and perspective both on campus and during several conference trips. My work at SCI was funded by the Department of Energy Center for the Simulation of Accidental Fires and Explosions. My time in graduate school was heavily influenced by internships where I met several valuable mentors, especially, Rocky Rhodes, Bill Mark, and Philipp Slusallek. Rocky introduced me to a group of people I would work with throughout graduate school, and several years later, helped me find my first job. Lastly, I recognize the most important discovery of my life so far, that of my fiancée, Nicole Fugere, whose patience and understanding have made the completion of this dissertation possible.



# CHAPTER 1

## INTRODUCTION

The fundamental characteristic distinguishing interactive real-time computer graphics rendering algorithms from off-line or batch rendering algorithms is the compromise achieved between spatial fidelity and temporal fidelity. Although the temporal cost of rendering a frame in a non-real-time system is an important consideration, there is no direct relationship between temporal cost of each frame and the possible temporal resolution of the imagery. The overall accuracy or combined spatial and temporal fidelity of rendered imagery is the degree to which the synthetic sequence of images mimics an actual or imagined visual experience. Fidelity differences may be observed in both the quality of spatial features in individual images and the temporal resolution, or accuracy of the image sequence, compared to an actual interaction over time.

In most high performance interactive rendering systems, spatial fidelity is the independent parameter, while temporal refresh rate, and therefore maximum temporal resolution, is dependent on the cost of rendering each frame. The overall spatial fidelity for a graphics system is determined by the set of rendering algorithms, geometric representations, realism of special effects, and display resolution which comprise the design of the system. Many choices effecting spatial fidelity such as the visibility algorithm, the realism of the material model, and the available geometric detail of scenes, are static choices made when the system is designed and cannot adapt to changes in the rendering workload. In contrast, temporal resolution and temporal fidelity are much more fluid. The refresh rate of the graphics system is not a fixed quantity. In many graphics applications, the spatial fidelity of the imagery is engineered to achieve an acceptable level of average temporal fidelity, e.g. sixty frames per second for real-time applications. At runtime, the actual temporal refresh fluctuates with the computational cost of achieving the spatial fidelity configuration. If the graphics workload changes in an unexpected way, or the capability of the graphics rendering engine is less than expected, it is not possible, in most systems, to reverse the dependent relationship between spatial and temporal fidelity, i.e. to make temporal fidelity the independent parameter, and dynamically adjust the spatial fidelity of the image sequence as needed.

Unlike temporal fidelity, in interactive graphics rendering, spatial quality is not simply dictated by a sample rate, it is usually formulated as the cumulation of a significant rendering engine and

scene design process. However, if spatial fidelity could be formulated in a more mutable or fluid manner, the spatial versus temporal fidelity trade off could be leveraged in a variety of circumstances to increase the overall quality and flexibility of interactive computer graphics. This dissertation formulates a rendering model for fine grained control of spatial and temporal fidelity based on the manipulation of spatial and temporal sample rates on the image plane. First, an algorithm is established to determine efficient spatial and temporal sample rates which vary both spatially across the image, and over time; then surplus sampling work is identified and leveraged to meet a specific fidelity strategy for the graphics system. This ability to leverage a fidelity trade off in rendered imagery is analogous to the trade off made by compression and encoding schemes in other forms of multimedia.

## 1.1 Interactive Rendering

The ideal continuous visual signal modeled by the imagery produced by a computer graphics system, usually a combination of rendering software and hardware, may be expressed as a function along the two-dimensional image plane and the temporal dimension. The two-dimensional image plane corresponds to the surface of the display device and the temporal dimension to the actual duration of the observation. The function  $\mathcal{I}(x, y, t)$  provides a pixel or image sample color value at the coordinates  $x$  and  $y$  on the image plane, at time  $t$ . The image function is equal to the composition of functions for the rendering model, geometry, and animation:

$$\mathcal{I}(x, y, t) = \mathcal{R}(x, y, \mathcal{G}(\mathcal{A}(t))) \quad (1.1)$$

The rendering function  $\mathcal{R}$  implements image formulation. Algorithms such as ray tracing and rasterization, combined with surface shading and other effects, are discrete algorithms for evaluating this continuous function. The third parameter of the rendering function is a time-dependent term composed of functions  $\mathcal{G}$  and  $\mathcal{A}$ , a geometric model and animation of the scene. In terms of a model like the Kajiya rendering equation [30], the temporally dependent parameter  $\mathcal{G}(\mathcal{A}(t))$  is the geometry, or visibility term, and  $\mathcal{R}$  contains all of the other terms related to illumination. Each function may consist of a number of additive terms, or image components; e.g. the rendering term  $\mathcal{R}$  may consist of image component terms for direct, indirect, and emissive illumination, each of which may be a function of  $x$ ,  $y$ , and  $\mathcal{G}$ .

In this ideal continuous model for image formation, all of the components of  $\mathcal{I}$  provide maximum fidelity compared with a natural scene, or an artist's interpretation. During the design of a

computer graphics system and during image synthesis, each component of  $\mathcal{I}$  must be discretized by a sampling function and eventually reconstructed on the output device at a finite resolution. Discretization takes place both implicitly through the selection of an implementation of each function, i.e. the selection of a ray tracer or rasterizer for  $\mathcal{R}$ , or a specific geometry level of detail for  $\mathcal{G}$ ; and explicitly through point sampling and reconstruction functions in  $x$ ,  $y$ , and  $t$ .

## 1.2 Sampling

The rendering process on a computer graphics system requires the selection of specific implementations for time-dependent animation  $\mathcal{A}$ , geometry representation  $\mathcal{G}$ , and rendering  $\mathcal{R}$ , as well as a discretization of each over  $x$ ,  $y$ , and  $t$ . One conventional set of choices is rasterization using a z-buffer for visibility, indexed triangle geometry, and animation limited to rigid body transforms. The set of choices includes how each object in the scene is modeled, the detail level of the animation, and the specific set of rendering effects used. Each choice alters the level of spatial and temporal fidelity of the rendered imagery in relation to an actual visual experience.

The discretization of certain components of  $\mathcal{I}$  such as the animation time stamp and the pixel discretization of the output image is accomplished by an explicit sampling operation. The sampling function operates on parameters in a domain outside of space and time, e.g. a sampling function on the geometry representation  $\mathcal{S}_d \cdot \mathcal{G}(x, y, t)$  might vary the geometry over space and time based on a discrete level of detail parameter  $d$ . Similar discrete fidelity parameters are common for shadows, texture detail, and lighting complexity where three or four discrete quality settings might be available for each component.

In addition to high level design decisions that place an upper bound on fidelity, the  $\mathcal{I}$  must be discretized in both space and time for display on a computer monitor with a fixed number of pixels and refresh rate. Discretization for rendering is accomplished by selecting spatial and temporal sampling functions.

$$\mathcal{I}'(x, y, t) = (\mathcal{S}_{xy} \cdot \mathcal{S}_t)(\mathcal{R} \circ \mathcal{G} \circ \mathcal{A}(x, y, t)) \quad (1.2)$$

The sampling function  $(\mathcal{S}_{xy} \cdot \mathcal{S}_t)$  may be applied to either side of the rendering expression. Applied to individual terms on the right side, the sampling function controls the discretization of individual image components. If the sampling function is applied to the left side of the equation, it is applied to the aggregation of all components, the final image produced by the renderer. Certain rendering algorithms such as ray tracing permit this type of sampling, i.e. since the color of each

primary ray in the image may be determined independently. Not every image component implementation or rendering algorithm scales in a way that allows postfiltering by a sampling function in the spatiotemporal coordinates of the image frame, e.g. the discrete resolution of a shadow buffer, a term on the right side, is not necessarily related to the resolution of the output image.

### 1.3 Spatial and Temporal Fidelity

In a nonadaptive rendering system, a fixed sampling function, either in terms of spatial sample positions on the image plane, or in terms of the choice of an algorithm from a design space, is selected for  $S_{xy}$ ; without additional adaptivity or control,  $S_t$  becomes an implicit function of the cost of evaluating  $\mathcal{I}$  at each moment in time. In this type of system, as the complexity of the animation, scene, and rendering effects increase, temporal samples move further apart and the temporal fidelity perceived by the user decreases. An adaptive rendering system may explicitly vary  $S_{xy}$ , and  $S_t$  in such a way that the computer system is able to balance the trade off between spatial and temporal fidelity.

#### 1.3.1 Definition of Fidelity

Consider a computer graphics animation of a car moving down a street, a ubiquitous situation in everyday life. The ideal measure of accuracy, or combined fidelity, is the difference between the real life visual experience and the synthetic experience of a sequence of images displayed on a computer screen. Differences may be predominantly spatial or temporal, and the fidelity of individual image components along a single axis may be different. For example, if the color of a certain location on the screen changes every 16ms due to the monitor refresh rate, but due to the movement of the car, that location in the continuous image changes every 8ms, e.g. due to the periodic motion of a rotating wheel, then the idealized temporal fidelity of the reproduction at that location is half that of the actual scene. Similarly, if the car is reproduced with less geometric complexity or a simplified illumination model, an image of the scene at an instance in time will have spatial differences with the actual scene.

It is impossible to measure this idealized fidelity difference because the viewer's perception cannot be recorded without some loss of fidelity due to discretization. As a thought experiment, however, the situation provides several useful results; both spatial and temporal change is present in the actual scene with magnitude varying between zero and infinity. The silhouette of the car against the background is a high energy spatial edge and its movement along the roadway creates high energy temporal edges. Regions with low magnitude temporal change occur along the body of the car which is a consistent color with variations due to illumination. Many spatial and temporal features are related, movement of a spatial feature causes temporal change, but both can occur

alone: a stationary spatial feature, or change in brightness across an area with uniform color. Since the maximum amount of sampling work possible over a certain region of both space and time is constant, fixing a sample resolution in space or time to achieve a certain level of fidelity will likely cause a decrease in the fidelity of the other dimension. Further, since both spatial and temporal features vary in scale and are not bandlimited, the fixed resolution will be inadequate in some regions and excessive in others.

The fidelity trade off is determined a priori by nonadaptive rendering systems and does not change as the content or constraints of the system fluctuate. In a renderer which uses super sampled antialiasing, the spatial fidelity gained due to an increase in the amount of sampling work performed per frame may lead to a decrease in the temporal resolution of the renderer, since the frame rate is proportional to the amount of work performed per frame. Likewise a progressive renderer fixes the temporal resolution and renders as many samples as possible in a certain amount of time.

The relative importance of each type of fidelity depends on the computer graphics application. In visual simulation, a situation similar to the moving car example, temporal fidelity is often more important than spatial fidelity. In entertainment or scientific visualization applications, spatial fidelity is often more important. Often the fidelity trade off may vary within a single application; for example, in a flight simulator, temporal fidelity is extremely important during takeoff and landing when a small visual delay might have large consequences, and may be less relevant at other times during the flight.

Temporal change may be induced by user interaction, even in situations where the graphics scene is not time varying in nature; e.g. medical visualization data sets often contain measurements from a single moment in time, or in CAD visualization where the data visualized is completely independent of time. In these cases, scene animation and temporal change is caused by input from the user such as manipulation of the scene geometry, shading parameters, or camera position. The idealized fidelity difference is still considered between the rendered image sequence and a non-discrete version of the imagery, i.e. manipulating an actual object, or an actual model of an object, versus the rendered imagery of the manipulation.

### **1.3.2 Fidelity Management Related to Image Compression**

Image or video encoding and compression is analogous in many ways to active fidelity management of an interactive graphics rendering system. Figure 1.1 shows a simplified compression pipeline for an audio or video encoding application. The first stage of the pipeline discretizes a continuous image signal with a sampling function, such as the sensor of a digital camera. Then analysis is performed on the discrete image to determine how to quantize the representation so that

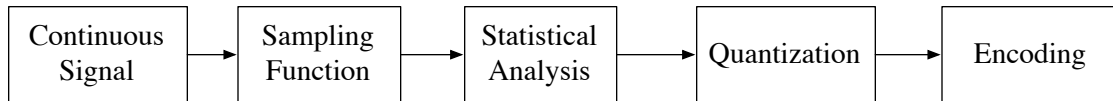
it may be encoded and transmitted to a viewer. Fidelity degradation may occur in the sampling, quantization, or encoding stages of the compression pipeline.

Each stage in the compression pipeline has an analog in the adaptive response pipeline shown in Figure 1.2; however, due to a feed back loop, the implementation and the interrelationships between the pipeline stages are different. Instead of sampling incoming light from a camera lens with a digital sensor, the graphics system renders image pixels of a synthetic scene on a computer display. The quantization and encoding stages of the compression pipeline are replaced by adaptive response and reconstruction stages. These adjust how rendering work is distributed across the image, and in turn use available image samples to increase the overall quality or efficiency of the rendering process.

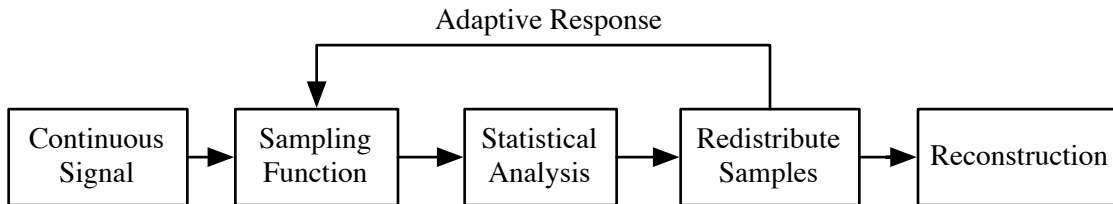
Temporally and spatially adaptive rendering differs from compression because it is a real-time problem, where the decision of how to manage computing resources to produce each image is an immediate problem that cannot be deferred until the future. Unlike a compression pipeline where an image frame at or before the immediate leading edge of time may be used to determine the quantization or encoding of an older image, the temporal constraint on real-time adaptive graphics requires control decisions to be made for immediate or future rendering work.

Image and video compression algorithms perform a filtering operation, starting with uncompressed full detail continuous imagery and producing a smaller stream of imagery, likely with less detail and possibly with the addition of certain artifacts. Although the class of artifacts introduced by adaptive sampling is similar, the fundamental filtering operation is different. The amount of detail synthesized by an adaptive rendering system depends on earlier adaptive response decisions due to the feed back loop. In the compression case, the decision to quantize or filter certain frequencies or details does not remove those details from subsequent input imagery; the full resolution input is always available.

Unlike conventional computer graphics imagery, fidelity trade offs are mutable in other types of multimedia, and the parameters available to manipulate them are less intrusive. In digital music and movies, the production and distribution of media are disjoint processes, the quality of musical instruments or the complexity of costumes or a set is independent of the eventual distribution technique or the viewing and listening devices employed by the consumer. During distribution, audio and video media is compressed and filtered to an appropriate fidelity level. In contrast, a computer graphics system must make a single choice about the relative importance of fidelity, e.g. level of geometric detail or inclusion of rendering effects, and each choice must be specifically engineered into the graphics application.



**Figure 1.1.** Multimedia compression pipeline.



**Figure 1.2.** Multimedia compression compared to the adaptive response model.

### 1.3.3 Fidelity and Vision

The degree to which a media compression or an adaptive rendering system degrades spatial or temporal fidelity is dictated by characteristics of human vision. Detail that is difficult or impossible to perceive may be eliminated without effecting the subjective quality of the media. In most cases, graphics systems are designed to avoid synthesizing unperceivable detail, and the adaptive rendering filtering process leads to a certain loss of fidelity. The capability of the human visual system guides the decision of which frequencies or detail in imagery to degrade.

The human visual system consists of eyes connected through optical nerves and other structures to the visual cortex located in the rear of the brain. The effective resolution of the system and the ability to perceive may be affected by properties of either structure. [52, 31]

Visual acuity, or the ability to see and distinguish detail, may be used to gauge effective eye resolution. Pattern recognition and localization are two tasks used to measure visual acuity in clinical settings. Localization tasks judge the relative difference in position between two stimuli, e.g. which set of points is closest, while recognition tasks, which are perhaps more relevant to computer graphics, distinguish two or more stimuli.

The lens and humorous regions of the eye cause diffraction of incoming light and effect how photons activate separated groups of photoreceptors. Efforts by Campbell and Green in the 1960s to measure visual acuity used lasers to reduce diffraction [10]. The researchers asked subjects to distinguish sinusoidal patterns produced by lasers focused on their retinas. The measurement resulted in an effective resolution of approximately 60 cycles, or distinguishable line shaped light impulses, per degree. This technique was later refined to measure a resolution of 150 cycles per degree [31].

Sinusoidal laser pattern experiments are designed to mitigate as much as possible the diffraction

caused by other structures in the eye and therefore indicate an upper bound on the fundamental limit of the human visual system. More recent work in the context of font display on computer screens has attempted to match resolution with the limit of effective visual acuity. Baxter and Corriveau [3] determined that 186 pixels per inch matches the limit of visual acuity from an 18 inch viewing distance. This corresponds to a sinusoidal pattern of 30 cycles per degree. For comparison, a 1440 pixel wide laptop screen viewed from 18 inches requires a visual acuity of 18 cycles per degree. This provides an approximate bound on the size of detail or features which must be reconstructed by an adaptive renderer.

Contrast, or relative difference in luminance between features, also plays a role in visual acuity performance. The relationship between contrast and feature size can be measured by decreasing the luminance of sinusoidal patterns. Most clinical visual acuity measurements use high contrast images. Maximum contrast sensitivity, the ability to distinguish between two low luminance levels, is limited by a spatial frequency of approximately 10 cycles per degree [31]. Just as in the case of spatial visual acuity, where high frequencies result in indistinguishable objects, beyond 10 cycles per degree, light from two colored regions may be diffracted in the eye making the two luminance levels indistinguishable.

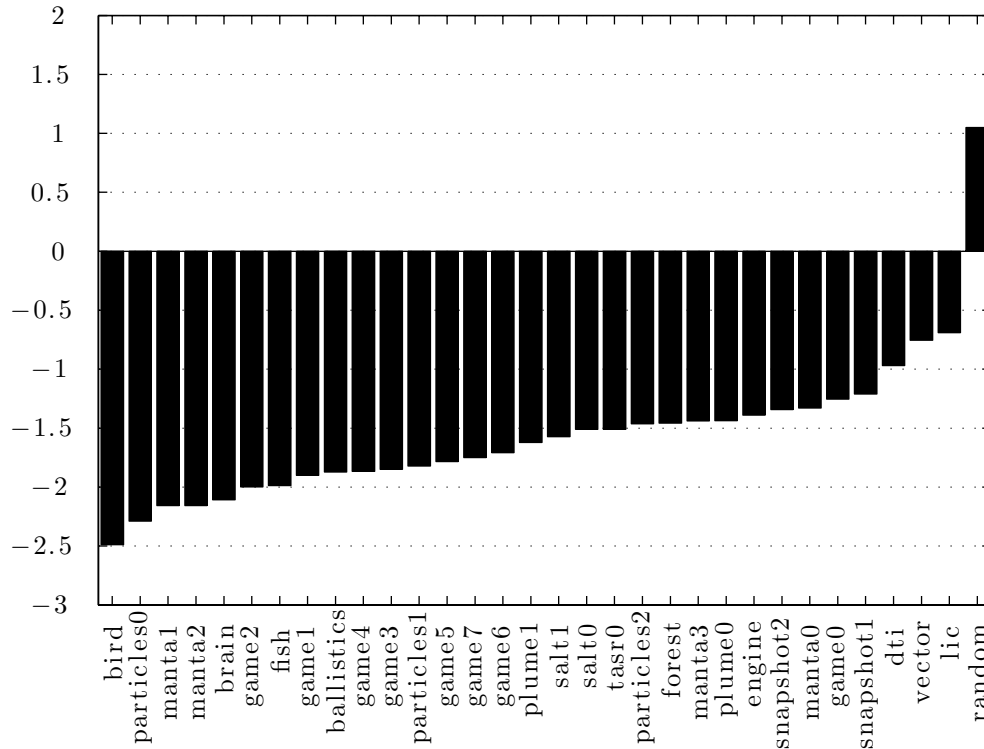
### 1.3.4 Suitability of Computer Graphics Imagery

Natural imagery statistics are examined in vision literature in terms of understanding fundamental human visual filtering mechanisms such as receptor sensitivity and the visual coding process [20]. Both second order statistics, such as the power spectrum, and higher order statistics used to correlate structures at different scales, are measures of redundancy in imagery. It is likely that anatomical constraints cause the visual system to exploit statistical redundancy and perform compression of viewed images during transmission from optical nerves to the cortex [21]. The objective of an adaptive rendering method is to exploit the same type of redundancy to decrease the cost of rendering an image.

Natural imagery has spectral characteristics suitable for adaptive sampling during rendering. The slope of the power spectrum of natural image ensembles is approximately  $-2$  on a log-log scale [20]. This means that the power spectrum, i.e. a histogram of frequency content, decreases as frequency increases with  $1/f^2$ . Sets of image ensembles from different natural habitats exhibit slightly different distributions [2], but the curve fit nearly always has negative slope. Figure 1.3 shows the power spectrum distribution for a set of thirty two images taken from both nature photography and computer graphics.

Representative examples of natural photography and rendered imagery are shown in Figure 1.4, along with the power spectrum of the examples plotted in log-log scale. Frequency varies from



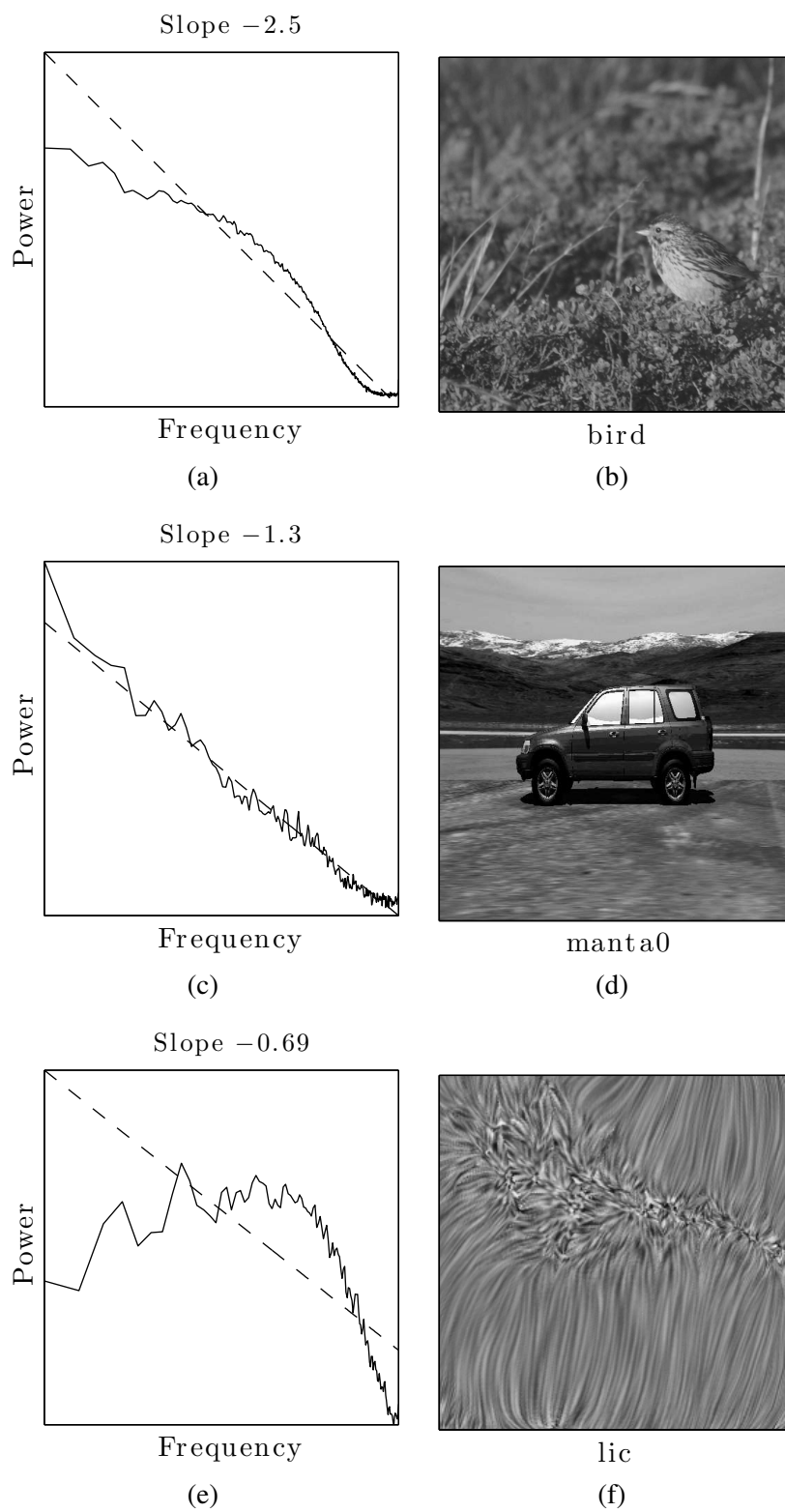


**Figure 1.3.** Power spectrum distribution for graphics workload.

low to high along the horizontal axis of each plot. The vertical axis indicates the amount of power in each frequency in the image, assuming the horizontal and vertical dimensions of the image are sampled at a uniform rate. The dashed line is a linear fit of the distribution, the slope of which is indicated at the top of the figure.

Uniform sampling renderers must choose a frequency, i.e. in Figure 1.4, along the horizontal axis, at which to sample. Any image regions containing frequencies above that rate will be aliased. Regions containing only frequencies well below, on the left side of the horizontal axis in the figure, will introduce sampling overhead. Adaptive anti-aliasing methods must also choose an initial sampling rate. These methods will decrease the amount of aliasing since a wider band of frequencies can be appropriately sampled, but will still introduce unnecessary overhead since much of the image will still only have lower frequency content than the initial sampling rate. Since the exact spectral behavior of rendered imagery cannot be known until sampling is performed, a sample density refinement algorithm should not be constrained by lower bound of an initial sample rate. The adaptive approach should be able to enlarge the band of appropriately sampled frequencies in either direction.

Power spectrum characteristics have been applied to computer graphics in both analytic and per-



**Figure 1.4.** Example power spectrum distributions.

ceptually guided techniques. Reinhard et al. [53] describe the implications of second order statistics to a variety of rendering applications including procedural modeling and reconstruction. They find that geometric modeling is more influential to the statistical properties of generated images than lighting and shading or other transformations like lossy compression or gamma correction. Results are obtained from a small set of images and only spatial statistics are considered. Bolin et al. propose an adaptive rendering technique based on a model of the human visual system. The reconstruction algorithm uses the characteristics of natural images to control variance when selecting unknown values [7].

The distribution of the power spectrum of natural and synthetic rendered imagery, the former assumed to be a higher fidelity representation of the latter, encourages an adaptive approach to image synthesis due to the dominance of low frequency content in the imagery. The selection of a sample rate capable of capturing energy in higher frequencies introduces redundancy into the synthesis process, something that may be avoided in an adaptive approach. Characterization of the amount of redundancy present in a graphics workload, as indicated by the shape of the power spectrum of representative imagery, is given in Figure 1.3. Although the average slope of the distributions shown is less steep than the  $-2$  of natural imagery, lower frequencies dominate all examples.

## 1.4 Thesis Statement

The principle thesis of this dissertation is:

Entertainment and scientific visualization workloads produce sufficient quantifiable redundant detail that can be exploited by temporally and spatially adaptive rendering leading to increased performance of high resolution, shading limited, or bandwidth limited applications.

## 1.5 Contributions

The central contribution of this dissertation is the algorithm and parallel design of an adaptive rendering system combining separable control over spatial and temporal fidelity and adaptive response based on statistical characteristics of rendered imagery. The approach is independent of the visibility or shading algorithm used by the graphics system, and employs scalable components suitable for implementation on massively parallel graphics processors.

### 1.5.1 Real-time Adaptive Frameless Rendering

The proposed algorithm introduces mechanisms to the Adaptive Frameless Rendering (AFR) approach of Dayal et al. [14] to increase the rendering coherence and use high performance mechanisms of conventional ray tracers. At the same time, the proposed approach avoids inefficient parallel structures, such as a complete hierarchical tiling, which limit scalability. These structures

are replaced with more efficient parallel mechanisms better suited to massively parallel computer systems. The result is a framework capable of interactive instrumented simulation or real-time performance on a individual workstation.

### **1.5.2 Separable Response**

Separable adaptive response is the ability of the control algorithm to effect the spatial and temporal sample rate of the rendering system independently. Many approaches described in Chapter 2, especially those which employ a probabilistic sampling mechanism, do not provide separable response. Despite the spectral relationship between temporal change and spatial details, the behavior is useful in situations scenes with significant background motion blur or depth of field. In these cases, the scale of spatial features is very large compared to the rate of temporal change.

### **1.5.3 Spatial and Temporal Fidelity Model**

The level of spatial and temporal fidelity, in terms of an isotropic spatial and temporal adaptive sample rate, may be modeled as an optimization problem in a two-dimensional space. The control decision which manipulates the level of spatial and temporal fidelity may be expressed as a relational system of equations with a small number of nonlinear constraints. The system allows enforcement of constraints on the graphics system, such as conservation of total rendering work.

### **1.5.4 Scalable Parallel Architecture**

The parallel architecture described in this dissertation is an extension to the Manta ray tracer pipeline architecture [5]. The architecture is extended first to accomplish interoperation with the rasterization graphics hardware pipeline, and second to interoperate between Manta, the graphics rasterization device, and the separate general purpose computation interface to the graphics device. This pipeline architecture allows the overhead of spatial and temporal adaptive sample rate control to be hidden, enabling the system to scale with the cost of tracing each ray. The number of image samples, and therefore rays needed, in turn depends on the spectral complexity of the scene.

## **1.6 Applications**

### **1.6.1 Ray Tracing and Hybrid Rendering**

With the advent of nonlocal shading techniques in both rasterized graphics and interactive ray tracing, degrees of freedom that effect rendering cost will increase. Instead of assuming a rendering cost linearly related to the number of polygons, texture operations, or shaders instructions, the cost will be largely determined, in a nonlinear manner, by the proliferation of nonlocal effects including shadows, indirect lighting, color bleeding, reflections, and refractions. Due to their

expense, especially in combination, these effects are uncommon in today's mainstream workloads but are ubiquitous in everyday life and will become a key element of future high fidelity workload.

### **1.6.2 High Resolution Displays**

The distribution of necessary rendering work will vary widely in nonlocal rendering workloads due to differences in the complexity of the effects across the image. One present day example of this type of work distribution is high resolution rendering. With a rendering workload having usual spectral and feature scale characteristics, high frequency regions of the image may require considerable super sampling for adequate reconstruction, while low frequency regions of the image will require fewer samples to reconstruct. If the output resolution is sufficiently high, the overhead of super sampling low frequency regions will dominate the cost of rendering. Efficiently identifying these regions and decreasing the amount of work performed in them will increase performance.

While the complexity of the rendering workload will increase, the complexity is bounded by, and its characteristics are governed by, the geometric structure and appearance of scenery, as well as the capabilities of display devices and the human visual system. While the ability to produce higher fidelity graphics imagery will increase, the viewer's ability to use it is largely constrained by the human visual system.

In this context, the objective of an adaptive rendering approach is to identify regions in the output which may be produced with less work. During rendering, this means identifying opportunities for undersampling the underlying image function and reducing the number of image samples used for reconstruction of those regions. This model moves beyond previous systems by exploiting both spacial and temporal undersampling at several stages in the rendering pipeline. By including a rendering pipeline from acquisition through viewing in the human visual system, downstream stages are able to provide adaptive response to upstream stages.

## CHAPTER 2

### RELATED WORK

This chapter describes the relationship between the adaptive rendering algorithm described in Chapter 3 and previous approaches through the classification of each approach within a taxonomy of fidelity trade offs.

#### 2.1 Sampling

Much of the early literature in computer graphics formulated nonuniform or adaptive sampling strategies starting with the uniform sampling theorem. The uniform sampling theorem states that the lower bound on a sampling frequency necessary to reconstruct a band limited signal is twice the highest frequency component of the signal [24]. Aliasing occurs when too few samples are used and a high frequency component of the underlying signal is misrepresented as a low frequency component. This leads to structured artifacts that the human visual system is acutely able to perceive.

The implication of nonuniform sampling in the frequency domain is described in Cook's 1986 paper [11]. In the examples which use a band limited input signal, the Nyquist limit of the signal is used to select the parameters of a nonuniform sampling pattern. Since multiplication by a sampling function in the time domain is equivalent to convolution in the frequency domain, the sampling function spectrum must be sufficiently wide in relation to the Nyquist frequency, such that no copies of the spectra produced during convolution overlap with the baseband producing aliasing. Since most useful signals in computer graphics are not band limited, the convolution does introduce some overlap, the nature of which depends on the shape of the sampling function spectrum.

Reconstruction is implemented using a low pass filter to crop the baseband of the sampled signal in the frequency domain [41]. Several types of artifacts including prealiasing, postaliasing, and base band attenuation, may result due to reconstruction [43]. Prealiasing, the artifact referred to above as aliasing, occurs due to undersampling of high frequencies in the underlying signal. Postaliasing occurs when a reconstruction filter is applied that is wider than the Nyquist frequency and subsequent copies of the spectra are not cropped. Base band attenuation occurs if the width of the reconstruction operator is not sufficient to capture all of the frequencies in the signal. In this

case, high frequency components are lost and the reconstructed imagery contains fewer details than were sampled. If a nonuniform sampling function is used, noise appears throughout the spectrum and a low pass filter will be unable to isolate the baseband.

There are two categories of nonuniform sampling schemes relevant to adaptive sampling which result in different reconstruction approaches, schemes with uniform density, but not necessarily uniform sample spacing across a domain, and schemes where the density varies dramatically across a domain. Patterned and random schemes comprise the uniform density category, i.e. sample sets which are aperiodic or stratified. Aperiodic samples are distributed using some random process, while stratified samples are placed within fixed subregions of a domain. Nonuniform sampling may be accomplished by adaptively applying different uniform schemes across an image. Variable density sample sets usually result from an adaptive technique, or in boundary regions between uniform schemes, or due to a nonsynthetic sampling mechanism, or the result of poor stratification.

## 2.2 Reconstruction

Certain sets of nonuniform samples can be perfectly reconstructed through an invertible warping process. This process is suggested by the nonuniform sampling theorem which states that if a one-to-one continuous mapping from a set of nonuniform samples to a set of uniform samples exists, and the result of its application produces a band-limited signal, therefore uniform sampling theory may be applied to exactly reconstruct the signal. However, it in general is not possible to find such a mapping for signals encountered in computer graphics.

Reconstruction is often performed to resample a set of dense samples into a less dense or stratified set, e.g. to downsample ray traced samples, or other subpixel color samples, to the resolution of a screen for display. Local filters, a common reconstruction technique, are a category of reconstruction techniques where a filter kernel with compact support is placed at each resample output location and the filter function is applied to each input sample within its extent [11]. Local filters are effective if sampling density does not vary across the domain. In other situations, multistep techniques must be used to account for missing information between different sample densities. Cook suggests two approaches to implementation, either computing weights at each sample location using filters placed at resampling points or using the filter to precompute a discrete convolution kernel and performing a weighted gather at each resample point .

Dippe and Wold [17] specify two objectives for reconstruction filters, cropping most frequencies greater than half the Nyquist rate to reduce aliasing, and smoothing lower frequency noise introduced by the sampling scheme. The authors argue that in an adaptive nonlocal situation, the filter should be normalized to reduce noise caused by local changes in the sampling rate. This is

accomplished with a weighted average, dividing the sum of the product between samples and filter weights by the sum of the weights. Other early local filters include the difference of Gaussians proposed by Cook and the family of cubic filters proposed by Mitchell and Netravali [43].

In situations where the density of input samples varies considerably across a domain multistep, reconstruction methods may be used to approximate the signal in areas with sparse samples. The weighted average filter step used by Dippe and Wold was designed to reduce artifacts caused by changes in sampling density across the domain. Varying sample density might occur with jittered sampling as the offset amounts may be completely uncorrelated [51]. In this case, two neighboring samples may be pushed close together, or in stratified patterns, if samples happen to be placed near boundaries close to one another. In the case of adaptive sampling, dense regions may dominate the weighted average, even if they are much smaller in size than sparse regions. Mitchell in a 1987 paper [42] proposes a multistage filter which repeatedly applies a weighted average filter over different sized subregions from fine to coarse. The average is computed hierarchically, dense subregions are normalized before being combined with neighbors, so their contribution to the overall average does not depend on the number of samples they contain.

In their 1996 paper on the Lumigraph, Gortler et al. [25] encounter a nonuniform sample reconstruction problem when they attempt to reconstruct a continuous weighting function using samples collected from a hand-held camera. The function is given by the four-dimensional continuous integral of the Lumigraph function multiplied by a basis function. The authors describe a three-stage push-pull algorithm to compute the integral for each weight on a uniform grid by reconstructing the Lumigraph function using samples from the hand-held camera. There are two principle challenges: holes in the Lumigraph function data caused by the scattering must be filled, and the effect of varying sample density must be resolved. The first problem is addressed following a technique introduced by Burt [9] by creating an image pyramid where each coarser level is blurred by a local kernel from the finer level. Information from coarser levels is then used to fill in regions missing samples in the finer levels. The second problem is addressed using Mitchell's hierarchical weighted average.

The original push-pull algorithm had three stages, although the first stage including obtaining the initial samples is application dependent. The first-stage splatting approximates the integral in cells of a fine uniform grid by taking the average of the Lumigraph function samples in each cell weighted by the value of the basis function at those samples. Since the basis function has compact support, cells in regions without input Lumigraph samples have weights that are close to zero. In the next stage of the algorithm, termed pull, the integral is approximated again using a coarser grid resolution and the weights computed at the finer resolution to weight the average. This process is



repeated hierarchically using coarser and coarser cells which correspond to a filter with increasing support. To prevent regions of the grid with dense input Lumigraph samples from dominating the average, an upper bound is placed on the maximum weight which may be used in the sum. In the final stage of the algorithm, termed push, the coarser approximation is used to fill in holes in the finer evaluations. This is accomplished by checking to see if the computed weight at a resolution is below a threshold and if so, blending the value of the function at the coarser level into the finer level. The effect of the push-pull algorithm is that the coarser levels of the hierarchy perform reconstruction of smooth low frequency signals, which intuitively require fewer samples, for holes in the data while the finer levels still retain high frequency information where data were available.

The technique has been used without considerable modification in other applications where holes in sample data must be filled in, such as rendering point sample data sets and BRDF reconstruction. In Grossman et al. [26], the push-pull method is used to fill holes in point set data for rendering. In this context, the holes are pixels on the screen to which points from the input data set were not projected. Unlike the Gortler paper which obtained weights from basis functions evaluated at every input sample position, weights are computed using a depth buffer pixel coverage metric. Matusik et al. [40] use push-pull as an alternative to a more complex reconstruction method based on wavelet transforms. Here the technique is used to reconstruct a continuous BRDF function from a sparse set of measured samples.

In their 2003 paper on image completion, Drori et al. [18] employ a push-pull technique iteratively adding samples to the approximation between each iteration. The authors solve an image completion problem consisting of filling in a region of an image removed by a matte operation with a plausible substitution. The matte operation produces a large unknown region of an otherwise uniformly sampled image. At each iteration of their completion algorithm, push-pull is used to obtain a low frequency approximation of the missing region. To add new samples (and higher frequency information) to the missing region of the image, each level in the push-pull pyramid is searched to match a known pixel location in the approximated region. The search is performed over all pairs of known and unknown pixels over five levels of the pyramid and eight orientations.

### 2.3 Adaptive Methods in Graphics

Adaptive techniques in computer graphics attempt to efficiently control error or increase fidelity in regard to system constraints on processing, storage, or bandwidth. These techniques fall into the categories of video encoding, adaptive textures and shadow maps, adaptive sampling, and of course, adaptive frameless rendering.

### 2.3.1 Rasterization

Most conventional commercially realized graphics hardware performs uniform density sampling operations, but the extension of this hardware to nonuniform sampling is explored in the literature.

Rasterization graphics hardware supports several anti-aliasing methods which increase sampling density uniformly across the image. Super sampling increases the number of complete image samples, i.e. shaded samples with depth and other properties, per pixel and filters the result. Multisampling decouples shading operations from the number of individual visibility samples so only one shader is invoked per polygon per pixel while depth and other properties are computed for each sample intersected with the geometry. Similar to interleaved sampling in interactive ray tracing, these methods lower the total cost of rendering all of the samples in a pixel by amortizing the more expensive operations like shading between multiple image sample or output pixels. Although these methods can reduce some aliasing by increasing the sampling rate, they are all uniform approaches which rely on tiles of fixed sampling patterns and therefore are unable to leverage the aliasing-for-noise trade off provided by nonuniform sampling methods.

One example of nonuniform rasterization is the irregular z-buffer proposed by Johnson et al. [29]. Instead of rasterizing samples in each input primitive using a uniform pattern, the algorithm allows samples to be placed individually about the image. The conventional z-buffer algorithm is used to determine the nearest fragment to the viewer. The irregular z-buffer algorithm stores specified samples in a two-dimensional search structure, e.g. grids, BSP trees, and kd-trees, and then performs a triangle shaped region query over the search structure to determine which samples to rasterize in each primitive. The algorithm operates in two phases, first rebuilding the search structure and then querying it while rasterizing each primitive.

### 2.3.2 Adaptive Textures

Adaptive textures vary the distribution of texture samples across a texture image and are used in situations where uniformly sampled textures of sufficient resolution would consume too much storage or in situations where high texture sample density is only necessary in a small region of the image.

Adaptive texture-based radiosity is a rendering method which computes power per unit area measurements on diffuse surfaces in a scene [28]. These measurements are stored in a regular quad-tree-based texture structure. Each node in the quad-tree represents a single texture sample, the radiosity sum over the node area. The algorithm performs two passes over the image. The quad-tree is first initialized by performing an eye pass which computes primary ray footprints on each diffuse surface. These foot prints correspond to the resampling resolution of screen samples, and are used to limit the maximum depth of the quad-tree, which governs the maximum sample

resolution. During the light pass, the radiosity sum and photon count in each node is updated and nodes above a certain threshold are split. Additionally, the splitting algorithm divides nodes close to discontinuities detected using a binary edge filter. This is a refinement approach as nodes in the quad-tree are never merged after they are split.

Generalized adaptive texture maps in two, three, and four dimensions have been used for compression, and to provide locally appropriate resolution[34]. This technique is designed to leverage sampling on early programmable graphics hardware by introducing a level of indirection to each texture look up. It preserves random access look ups and avoids the search-based query of earlier approaches which was not possible on early hardware. The adaptive texture coordinate space is divided into a regular grid, where each cell in the grid contains actual samples at a different resolution. An offset into another buffer and a scaling factor are associated with each grid cell which enable a two-step look up into texture memory. Unlike the quad-tree search structure used in software, the regular grid with indirection provides a constant time look up; however, the high level regular grid with a fixed resolution may constraint adaptivity or increase the amount of storage necessary.

Shadow maps are a depth buffer method to approximate shadow coverage without explicit ray tracing. The shadow map consists of a depth buffer rasterized from the position of a light source facing the scene. When the scene is rendered from the viewer's position, the location of each fragment in world space is projected on to the shadow map. If the shadow map records a depth less than the distance from the fragment to the light source, the fragment is judged to be in shadow. Ideally, the area of the scene's projection to both the shadow map and the output image will be similar so that each fragment projects to a sample on the map. Aliasing occurs when the density of the shadow map falls below that of the screen pixels which sample it. For example, this occurs if the light source is further away from the scene than the viewer since more than one screen pixel might project to the same shadow map pixel.

Adaptive shadow map methods attempt to reduce aliasing artifacts by varying the resolution of depth samples across the map. Shadow sample density is increased in undersampled or high frequency regions. The algorithm must optimize distribution given a fixed amount of storage for the shadow map. Fernando et al. [19] accomplish this by placing a quad-tree over the shadow map. Each node in the quad-tree has a fixed resolution and samples are stored at each level in the tree to facilitate mip-mapping. Nodes are added to the tree in undersampled regions based on a cost heuristic. The cost heuristic accounts for the overhead of reading back shadow samples because the quad-tree must be created and maintained on the CPU. The method is progressive in that if the shadow image remains unchanged, the system can refine the quad-tree until the image

changes. Lefohn et al.[37] describe an approach which uses a GPU page table system and more efficiently distributes shadow sample storage. This approach uses a convolution-based edge detector to determine where shadow sample density should increase, based on frequency, instead of a strictly undersample detection-based method.

Strictly uniform or tiled uniform patterns cause aliasing artifacts in situations where rasterized samples are accessed or resampled at a varying resolution. This problem occurs in shadow mapping where queries are often made between uniform shadow samples. One solution proposed by Aila et al. [1] is to project 3D view image samples to the shadow map image and then rasterizing only those points. Unfortunately nonuniform rasterization is not supported by conventional hardware, presumably due to the performance advantage of regular sampling patterns over variable patterns which would require more coordination during parallel rendering.

### 2.3.3 Progressive Anti-aliasing

Progressive refinement is an adaptive sampling process in which samples are iteratively added to a pixel until some quality or expense threshold is reached. Whitted's early work in ray tracing included an adaptive refinement step for anti-aliasing as well as a mechanism to prevent geometry from slipping between rays [71]. The anti-aliasing algorithm examined each neighborhood for four regular samples and subdivided the neighborhood if the samples had sufficiently different values. Since new samples were always placed on a finer regular grid, alias reduction was accomplished through super sampling rather than a stochastic technique. Subdivision would continue until the values of subdivided neighborhoods sufficiently converged or the machine ran out of precision. Whitted's renderer traced rays along a regular grid. Geometric anti-aliasing was accomplished by computing a minimum bounding sphere around each object in the scene based on the distance to the camera. If a ray intersected the bounding sphere but did not intersect the enclosed object, the square neighborhood of rays on the grid would be subdivided. Convergence was determined using several measurements including sample luminance variance, separate channel contrast weighted by a luminance factor [41, 42]. In large sample neighborhoods, edge detection filters may be applied to the variance measure.

Painter et al. [51] propose a nonuniform adaptive sampling mechanism based on a kd-tree subdivision of the image. Each kd-tree node contains an estimate of the image variance of the region contained in the node's subtree. Refinement is performed to detect features, such as edges, and increase sample coverage by adding more samples to large image regions. Nodes are subdivided using an error function given by the product of the node area and the average variance of the local neighborhood in the kd-tree. The area term attempts to ensure that sample coverage is sufficient to not miss small features. Refinement depth is controlled using confidence interval computed from

sample variance.

These adaptive anti-aliasing methods are employed in noninteractive systems although straightforward implementation in interactive systems is possible given their use of simple spatial statistics-based control. These techniques only increase the sampling density across the image; it is not possible to decrease the initial sample density. Without frame-to-frame information or any additional prior knowledge of image behavior, initial sample density must be sufficiently high to detect high frequency behavior and as a result, there are fewer opportunities for spatial undersampling. Since the principle objective in noninteractive systems is to reduce aliasing, undersampling is not as important in off line applications as it is in interactive rendering where it provides an opportunity to decrease the amount of work required to render each image and increase performance.

### 2.3.4 Adaptive Rendering

Early progressive interactive approaches employed a very coarse level of adaptivity. Bergman et al. [4] propose a progressive rasterization-based system which initially displays only transformed vertices, then edges, polygons, and finally increasing degrees of shadows, shading, and anti-aliasing. Progressive improvements are applied between user input when the scene and camera are stable. The approach uses both model data and image statistics to adaptively decide which parts of the image to render at higher quality. For example, mesh normals may be used to determine which polygons are likely to have specular highlights and then Phong shading is applied only to those polygons. Intensity variance is used to determine which image pixels should be anti-aliased. This type of adaptivity was quite different than the progressive sampling techniques developed at the same time which focused on anti-aliasing to reduce spatial error. Since the approach targeted an interactive renderer, it introduced a very coarse notion of temporal change to adaptive control. The progressive rendering process was restarted upon user input which would introduce temporal error.

### 2.3.5 Video Encoding

Video encoding is a problem closely related to adaptive image display in that attempts to conserve bandwidth by adapting how information is encoded and transmitted based on the structure of the video stream. Bandwidth or storage, the number of bits used to encode the stream, is the constrained resource. There are two principle encoding categories, constant bit rate (CBR) and variable bit rate (VBR). CBR encoders use a fixed number of bits to encode each image so quality varies depending on image content. VBR encoders use a variety of techniques, including multiple passes, to estimate a distribution of bits per image such that quality is more consistent over time.

Encoded MPEG video streams are divided into groups of pictures (GOP) which may be divided along the lines of *scenes* in the video content. Each GOP contains three types of image

records. I-type images are essentially key frames and contain the most amount of information, P-type (predicted) pictures use motion estimation based on I-type pictures to predict (approximate) their content. B-type (bidirectional predicted) pictures contain the smallest amount of information and are computed based on motion estimation from previous and future pictures. Each picture is divided into macroblocks and statistics including motion estimates are stored per block.

Digital video encoding is used in both real-time and offline applications. In offline applications such as DVD encoding, the entire input stream may be examined several times to optimize compression. Real-time applications like video camera recording require encoding methods which operate on smaller stream windows. Bit rate constrains the encoding operation in two ways: first a fixed number of bits is available to encode a given segment of the stream (either communication bandwidth or available storage); second since the decoding device has a fixed buffer size, the bit rate must be sufficient to deliver whole images in time for display without overflowing the buffer.

The objective of offline VBR encoding is to optimize the perceptual picture quality throughout the entire video stream given a fixed number of bits, e.g. the storage capacity of a DVD[70], and decoding buffer constraints. This may be accomplished using a two-pass algorithm, the first pass computes statistics across the entire stream and the second pass uses these statistics to control encoding bit rate. The statistics computed in the first pass include a quantization scale, spacial activity measure, and a temporal activity measure. The quantization scale counts the number of bits necessary to encode each macroblock region in a picture. Spacial activity is measured by computing luminance variance. Temporal activity is computed by finding a lowest error motion vector (essentially projected translation) per macroblock in the image. After the first pass, a processing step uses the computed characteristics to determine a bit budget for each picture in the stream. The second pass encodes the stream by reducing the bit budget for easy pictures and increasing the budget for difficult pictures. The pass monitors encoding performance to attempt to detect exceptionally difficult situations like abrupt scene changes or fade to black transitions.

Real-time encoding algorithms attempt to perform the same operation but have less global information about the input stream. Instead of computing statistics in a first pass and then solving a global optimization problem to determine individual bit budgets and guide rate control, statistics are first initialized to known reasonable values and then updated throughout the encoding process. The updated statistics are used to classify the behavior of the input stream and perform rate control[44]. Content transitions such as an abrupt scene change, or fade-to-black followed by a new scene, may cause this approximation to become unreliable. These transitions may be detected by comparing measured statistics to a predictive model and using conservative encoding parameters when significant differences are observed. In addition to quasi-empirical predicative models, human visual

system factors may be used to guide VBR rate control [36].

There are some important similarities between VBR encoding and the proposed model for adaptive rendering. Both models compute statistics from the input stream to determine how to distribute constrained resources with the object of limiting perceptual artifacts. In both cases, this distribution may be based on human visual system factors as well as measured characteristics of the input stream. At the same time, there are several important differences between the two models. VBR encoding may adapt image quality only by changing the number of bits used to transmit or store an image; it operates blindly on the input stream. The model is unable to adapt upstream stages, such as the mechanism used to capture or discretize the image, to better match the input stream to the possible output representation.

Video compression and encoding represents just one stage in the graphics pipeline which adaptive rendering attempts to optimize. In an adaptive rendering model, the encoding stage would attempt to optimize both the encoded output stream and the upstream component which produces the input, i.e. the rendering engine, to minimize the amount of overhead introduced by downsampling, analogous to lossy compression.

## 2.4 Taxonomy of Fidelity Trade Offs

Like media compression, most computer graphics algorithms leverage some type of fidelity trade off, either explicitly between spatial and temporal fidelity, or between the individual components within the same dimension. Common approaches may decrease the computation cost of rendering operations, exploit coherence to amortize expensive procedures, or decrease the complexity of the graphics scene to adjust the rendering workload within the capacity of the graphics system.

Three characteristics distinguish these approaches, the fundamental trade off exploited by the approach, the adaptive mechanism, if any, employed, and the spatiotemporal response or characteristic behavior of the approach. Fidelity schemes may be divided into seven general categories, described in the subsequent subsections, based on the trade off between spatial and temporal fidelity exploited by the approach.

The definition of spatial fidelity, temporal fidelity, and accuracy given in Section 1.3.1 relies on a top-down analysis of the rendering system, where the output imagery is compared with an actual visual experience. The approximation to the visual experience is discretized along two orthogonal dimensions, the spatial and temporal axes, by a set of design choices. The cumulative effect along each axis of these design choices produces a spatial or temporal fidelity, or quality level, of the renderer. Very few design choices effect only spatial or temporal fidelity; the majority cause some change in both.

The relationship between spatial and temporal fidelity may be expressed as an abstract system or function with a single independent variable in most cases, and one or more dependent variables. This function represents the relationship between the overall level of spatial and temporal fidelity delivered by the system. The independent parameter may be either a fixed spatial or temporal fidelity level, selected by a designer based on the anticipated content of the scene, or derived from characteristics of the graphics scene or rendering system at runtime.

Let the variables  $Q_s$  and  $Q_t$  represent the spatial and temporal fidelity indices or quality levels, respectively, described in Section 1.3.1. These variables are placeholders for the overall subjective quality of the graphics system compared to an actual scene, which are not measurable qualities of the system or its input at a specific point in time. In terms of these fidelity indices, a certain graphics rendering approach  $\hat{\mathcal{I}}$ , analogous to the discrete image synthesis function  $\mathcal{I}'$  given by equation 1.2, may be expressed:

$$Q_t = \hat{\mathcal{I}}(Q_s) \quad (2.1)$$

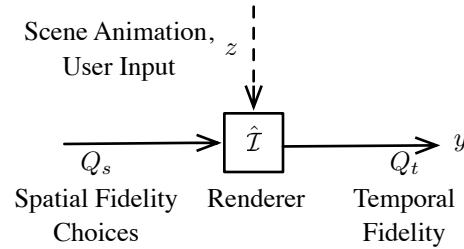
In this case, the level of temporal fidelity  $Q_t$  depends on the spatial fidelity level  $Q_s$  specified to the rendering system; the variable  $Q_t$  is independently determined. The equation describes a *spatial for temporal* trade off since the temporal fidelity of the rendered imagery depends on the spatial fidelity level specified. The relationship is illustrated in Figure 2.1; the actual temporal quality observed by the user may vary over time as the scene animation and user input changes the graphics workload. The effect of these disturbances to the system, indicated by the variable  $z$  in figure, may cause slight fluctuation, but does not alter the relationship between spatial and temporal fidelity in the system.

An adaptive temporally dependent system is shown in Figure 2.2 where the independent spatial fidelity parameter is given by the function  $\Psi$ :

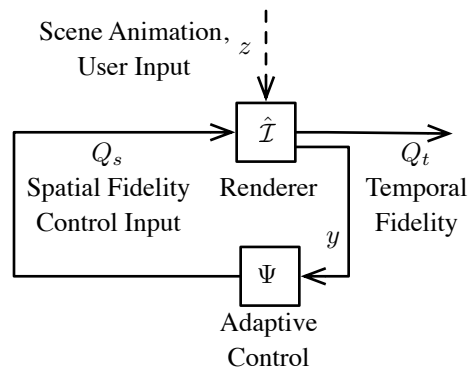
$$Q_t = \hat{\mathcal{I}}(Q_s), \quad Q_s = \Psi(\dots) \quad (2.2)$$

Many approaches are distinguished by the control function  $\Psi$ , which responds to varying characteristics of the system, indicated by  $y$  in the figure. In some cases, the control algorithm follows hardware counters or hardware measurements from the rendering engine; in other systems,  $\Psi$





**Figure 2.1.** Spatial for Temporal Trade Off of the rendering system  $\hat{I}$  with quality levels  $Q_s$  and  $Q_t$ . In this example, graphics scene  $z$  and the generated imagery  $y$  are independent of the overall fidelity level provided by  $\hat{I}$ .



**Figure 2.2.** Dependent temporal system schematic for equation 2.1. The control input, a spatial fidelity level  $Q_s$ , is combined with disturbances  $z$ , to produce a spatial fidelity level  $Q_t$  in the system  $\hat{I}$ .

consists of analysis of the graphics scene or statistical analysis of output imagery. In caching based systems that exploit temporal coherence,  $\Psi$  is often implemented as a cache lookup or replacement mechanism. Different choices of  $\Psi$  are described in Section 2.3.

Table 2.1 corresponding approaches. So-called *free* trade offs occur when one type of fidelity is increased independently of the other, e.g. improving a low level rendering operation such as the rasterization of a triangle or ray intersection may increase temporal performance because the computational cost of rendering each image decreases but should not change the spatial quality of rendered imagery. *Spatially dependent* trade offs increase spatial quality by decreasing either temporal fidelity, or decreasing a different spatial fidelity component. *Temporally dependent* trade offs are just the opposite; temporal quality is increased in exchange for spatial quality, or a reduction in a different component of temporal quality. The mechanism described in this dissertation falls into the final category; in systems exploiting a *mixed* trade off, both spatial and temporal fidelity are modulated across space and time to meet a certain criterion. These approaches specify a level of spatial and temporal quality, and then adjust both components in different regions of the image to

**Table 2.1.** Fidelity trade offs leveraged by various rendering approaches.

Fidelity Trade Off (e.g. trade <i>reduction</i> for <i>increase</i> .)	Examples	
Free Temporal	Early Depth Test	[33]
	Ray Tracing Acceleration Structures	[65]
Free Spatial	Display hardware post process.	
Temporal for Spatial	Adaptive Refinement	[4]
	The Render Cache	[67]
	Multi frame rate Rendering	[59]
	InfiniteReality	[45]
	Geometry Aware Framebuffer LOD	[75]
Spatial for Spatial	Progressive Anti-aliasing	[42]
Spatial for Temporal	Frameless Rendering	[6]
	Level of Detail Techniques	[38]
	Shader complexity	[8]
Temporal for Temporal	All Frequency Relighting	[50]
	Reverse Reprojection	[47]
	Temporal Radiance Caching	[23]
	Incremental Instant Radiosity	[35]
	Reprojection Shader Caching	[57]
Mixed Adaptive	Interruptable Rendering	[74]
	Adaptive Frameless Rendering	[14]
	TSAR (see Chapter 3)	

achieve it. Classifying rendering approaches based on a fidelity trade off in this manner yields four categories: in each case, the independent parameter is modulated, generally decreased, to inversely effect the dependent variable.

Table 2.2 classifies a variety of related approaches which exhibit a fidelity trade off with the classification scheme of Table 2.1. In addition to the independent and dependent parameter, the techniques are characterized by properties including the scope of the trade off, i.e. it effects per-pixel or per object behavior, a whole frame, or an individual image components (denoted *Img. Com.*); the basis of the control algorithm, e.g. occlusion (*Occ.*) or spatial fidelity (*Sp. Fid.*); and reconstruction algorithms, and the benefit and detriment to the resulting imagery. The table also indicates whether the technique relies on the progressive assumption (*Pr.*), described in Section 2.4.2, and if the technique is capable of separable response (*Se.*).

**Table 2.2.** Fidelity trade off characteristics of several alternative approaches.

Example	Pr.	Se.	Trade Off	Scope	Control	Recon.	Benefit	Detriment
Adaptive Refinement [4]	✓	-	Sp./Te.	Algorithm	Te. Cost	-	Interactivity	Dramatic spatial fidelity
Progressive Anti-aliasing [42]	✓	-	Sp./Sp.	Pixel	Sp. Fid.	Static Fil.	Anti-aliasing	Fixed frame rate
Adapt. Virt. Environments [22]	-	-	Sp./Te.	Object	Join. Fid.	-	Interactivity	Geometric detail
Frameless Rendering [6]	✓	×	Sp./Te.	Pixel	Random	Composite	Highest Te. Fidelity	Low fidelity dynamic sp. features
InfiniteReality [45]	×	×	Sp./Te.	Frame	Te. Cost.	Bilin.	Consistent frame rate.	Sp. resolution or scene complexity
The Render Cache [67]	✓	×	Te./Sp.	Pixel	Den. Age	Static Fil.	Ray or path traced fidelity	Upsampling artifacts
Level of Detail Techniques [38]	-	-	Sp./Te.	Geometry	Sp. Fid.	-	Geometry	Spatial fidelity
Interruptable Rendering [74]	×	✓	Both	Frame	Join. Fid.	-	Either Sp. or Te.	Complement
Disp. Hardware Post Process [33]	-	-	Free Sp.	Frame	-	-	Color fidelity.	None
Early Depth Test [33]	-	-	Free Te.	Raster	-	-	Frame rate	No fidelity effect
Adapt. Frameless Rendering [14]	✓	×	Both	Pixel	Join. Fid.	Sp. Te. Fil.	Sampling efficiency, anti-aliasing	Resampling artifacts
All Frequency Relighting [50]	✓	×	Te./Te.	Img. Com.	Te. Fid.	Wavelet.	Frame rate	Indirect illumination fidelity
Acceleration. Structures [65]	-	-	Free Te.	Geometry	-	-	Frame rate	No fidelity effect
Multi frame rate Rendering [59]	×	×	Sp./Te.	Frame	Static	Optical	Te. sampling efficiency	Sp. fidelity at lower rate.
Shader complexity [8]	-	-	Sp./Te.	Img. Com.	Static	-	Sp. fidelity	Frame rate
Geo. Aware F. B. LOD [75]	×	×	Sp./Te.	Img. Com.	Te. Cost.	Sp. Bilat.	Frame rate	Sp. resolution
Reverse Reprojection [47]	×	×	Te./Te.	Img. Com.	Occ.	-	Frame rate	Illumination fidelity
Temporal Radiance Caching [23]	-	-	Te./Te.	Img. Com.	Te. Fid.	Te. Fil.	Frame rate	Illumination fidelity
Inc. Instant Radiosity [35]	✓	-	Te./Te.	Img. Com.	Den. Occ.	-	Frame rate	Illumination fidelity
Reproj. Shader Caching [57]	×	×	Te./Te.	Img. Com.	Occ.	-	Frame rate	Img. components
TSAR (See chapter 3)	×	✓	Both	Framelet	Sep. Fid.	Sp. Te. Fil.	-	-

Within this taxonomy, the distinction between interactive or real-time rendering, and offline or batch rendering is important. In the latter case, although a trade off exists between the computational cost and quality of each frame, the overall fidelity of the eventual output is not constrained by either fidelity level  $Q_s$  or  $Q_t$ , since the rendering is not performed in real-time. Approaches that actively manage spatiotemporal characteristics to exploit coherence and amortize expensive operations, e.g. the spatiotemporal architecture for animated rendering proposed by Havran et al. [27], or the caching scheme proposed by Dietrich et al. [15], do increase rendering speed, but since they are not real-time approaches, there is no relationship between computational speed and temporal fidelity or temporal resolution of the output. Computational speed may limit the amount of output produced, i.e. the number of frames rendered in an hour, but not the resolution with which time may be sampled. The temporal fidelity  $Q_t$ , in practice the temporal sample rate, is not dependent on the cost of rendering each frame since the rendered movie may be replayed at an arbitrary speed or even in an arbitrary order.

### 2.4.1 Independent

Many acceleration techniques, such as decreasing spatial structure traversal cost or the amount of rasterization overdraw, may increase one fidelity type without effecting others, e.g. the temporal refresh rate is increased without altering the spatial quality of imagery. These approaches are free, considering only the relationship between spatial and temporal fidelity. In a so called free trade off, the dependent fidelity is still a function of a design parameter, e.g. ray packet size or amount of raster parallelism, but is independent of the complementary fidelity.

#### 2.4.1.1 Free Temporal

Free temporal fidelity improvements deliver fixed spatial quality while temporal quality is dependent on an unrelated function or parameter such as the cost of tracing an individual ray or rasterizing a sample. In the domain of interactive ray tracing, free temporal fidelity improvements concern the construction of acceleration structures, the selection of a traversal algorithm, or the intersection between rays and objects in the scene. The design space described by the survey of Wald et al. [65] consists of grid, bounding volume hierarchy, and kd-tree spatial structures and a set of algorithms to rebuild or update the structures as the scene animation changes. One example of a free temporal fidelity improvement in rasterization is the early depth test, or z-cull, described by Kilgariff et al. [33]. This optimization performs a depth test before material shading in the case that the shader is known to not alter the depth of a fragment allowing more expensive shading operations to be avoided in cases where the fragment will be discarded by the depth test. Another example is the acceleration of a rendering algorithm by implementation in new hardware; Schmittler et al. [55]

describe the architecture of an interactive ray tracing approach. This architecture would produce the same imagery as a conventional system, but at a higher temporal resolution.

### 2.4.1.2 Free Spatial

Free spatial fidelity approaches are more difficult to realize than independent temporal improvements since nearly any change to the graphics system effects the computational cost of rendering and the temporal refresh rate. One example is a postprocess implemented in display hardware for the purpose of color correction and display calibration. Another example is changing the viewing conditions of the display device, e.g. by dimming lights in the viewing room to improve contrast, or moving the user to obtain a better viewing angle on the display. In these cases, the spatial fidelity is improved without any effect on the temporal performance of the renderer. Free spatial fidelity improvements must be implemented in a manner that has no effect on the computational cost of rendering.

## 2.4.2 Spatially Dependent

The majority of techniques described in computer graphics literature of the past several decades fall into the temporally dependent category because they make a spatial fidelity trade off in exchange for a lower per-frame rendering cost. Early computer graphics systems, designed with very limited computational budgets, especially those used for visual simulation, employed the opposite relationship between spatial and temporal fidelity. These systems guaranteed a real-time response latency, e.g. 60 frames-per-second, and contained hardware mechanisms to fluctuate the spatial fidelity of individual frames.

Spatially dependent trade offs often rely on the *progressive assumption*, the notion that after a certain point in time, or within a large interval of time, the overall accuracy of rendered imagery is dominated by spatial fidelity because the graphics scene has stopped changing temporally. This situation occurs in reactive computer graphics applications, such as the visualization of static data, where the scene does not change without user interaction to move the camera, or in situations with band limited temporal change where an upper bound may be placed on the necessary temporal refresh rate. The algorithm described in this dissertation is designed for computer graphics applications which do not abide by the progressive assumption.

### 2.4.2.1 Temporal for Spatial

Although an interactive approach by modern standards, the adaptive refinement framework proposed by Bergman et al. [4] exploits the fundamental trade off between temporal cost and spatial quality. The algorithm adds spatial details to the frame buffer following a seven step procedure, first

drawing only vertices, then adding edges, polygons, shadows, Gouraud, and Phong shading, and finally anti-aliasing, in successive steps. The number of steps performed is determined adaptively based on the amount of time available for rendering, essentially the required temporal quality which is the independent parameter. The first several steps produced imagery in under a second, while in 1986, the advanced shading and anti-aliasing required twenty minutes.

The adaptive manipulation of geometric level of detail to maintain a consistent temporal refresh rate is described by Funkhouser et al. [22] as an optimization problem with relational constraints. The approach maximizes a benefit heuristic while keeping a cost heuristic within a temporal budget based on the render time of the previous frame. The benefit heuristic employs a model for spatial fidelity based on the effective raster sample rate of an object at a certain level of detail. The cost model is determined by fitting a linear curve to experimental data based on the graphics system and workload.

The InfiniteReality computer graphics system described by Montrym et al. [45] contains two adaptive spatially dependent mechanisms designed to allow visual simulation applications to maintain a consistent refresh rate. The graphics hardware detects both geometry limited and raster fragment fill limited conditions and either notifies the application so that geometric complexity may be reduced, or reduces the resolution of the output frame buffer to decrease the amount of rasterization required.

The Render Cache, described by Walter et al. [67, 66], is an early sample reprojection and caching approach where a buffer containing samples from a spatiotemporal volume at the leading edge of time is reconstructed to produce imagery. Image samples are produced by high quality ray tracing or path tracing rendering algorithms at a lower temporal rate based on the cache replacement algorithm. Sample replacement is directed by a priority image buffer, based on the density of cached samples across the image. The algorithm tracks object movement by associating an object identifier with individual samples in the cache.

Velázquez-Armendáriz et al. [64] describes an implementation of the Render Cache algorithm employing both the CPU and GPU; the approach uses an edge detection scheme to improve the performance of spatial reconstruction. Similar to the approach described in this dissertation, sample rendering is performed by the CPU while analysis and reconstruction are performed on the GPU. Like the original algorithm, sample density, coverage, and age guide adaptive sampling; scene geometry is processed by the system, but only to detect edges used to guide spatial reconstruction.

The majority of other reprojection and caching approaches are classified as temporal-for-temporal trade offs, described in Section 2.4.3.2, because certain image components are cached while other components, such as visibility, are rendered each frame. The cached image components tend to

be low frequency signals, such as indirect illumination, which are expensive to render, but do not dominate the spatial fidelity of the imagery. In contrast, the Render Cache algorithm reconstructs frames from final image samples stored in cache, without newer information. The amount of temporally accurate sampling at the leading edge of time performed to update the cache is the independent fidelity parameter. The temporal fidelity controlled by this parameter is traded for higher spatial fidelity delivered by ray tracing and path tracing rendering algorithms.

Multiframe rate rendering, described in Springer et al. [59, 60], renders different objects in the scene at various temporal refresh rates using separate rendering devices, and then composites the resulting images. Compositing is performed either optically by overlapping the projection of two independent display devices, or in a digital process. The approach divides the scene into groups of objects whose accuracy is dominated by spatial fidelity, termed visual quality, versus temporal fidelity, or interaction quality; the former is rendered by a *slow client* device, and the latter rendered on another device, termed the *fast client*. The classification of an object as interactive or static based on its temporal characteristics determines the rendering device and algorithm the object is assigned to, and therefore the level of spatial fidelity.

The frame buffer level of detail approach described by Yang et al. [75] adjusts the rasterization resolution of fine-grained image components to maintain a consistent temporal refresh rate. The approach uses a joint bilateral filter based on color, depth, and surface normal to upsample low resolution components. Adaptive control of spatial resolution is performed in a feedback loop using a heuristic that estimates the time spent to render an object based on the latency of the previous frame and frame buffer coverage of each object. The coverage estimate, expressed as a percent of the frame buffer, is provided as a scene parameter.

#### 2.4.2.2 Spatial for Spatial

Approaches that attempt to redistribute rendering work spatially across individual frames without altering the cost of rendering the entire frame may trade spatial fidelity between different regions. Progressive refinement, or anti-aliasing approaches, perform this trade off by first rendering a low resolution image to the frame buffer and then progressively adding samples along image features at increasing spatial resolution. These systems operate on the progressive assumption that the scene does not change within the short window of time during which samples are added. An early example in a noninteractive context is described by Mitchell [42]; the mechanism is common in conventional rendering engines, such as Manta, Bigler et al. [5], and OpenRT Dietrich et al. [16]. The amount of refinement or anti-aliasing performed may be determined either by a certain temporal interval, in which case the temporal refresh rate is fixed, or a fixed sample budget, in which case the temporal fidelity of the approach might vary, but would depend on other spatial fidelity trade offs

like the complexity of the scene or material properties.

### 2.4.3 Temporally Dependent

Since the introduction of high performance commodity graphics hardware, the principle fidelity trade off addressed in interactive computer graphics literature has shifted to temporal dependent approaches. This is likely the result of ubiquitous graphics hardware which reliably delivers an acceptable spatial fidelity level at real-time rates. Instead of substantially reducing spatial quality to maintain interactivity, within the last decade, the principle question has become how to increase spatial fidelity with a minimal effect on temporal fidelity. Unlike previous approaches, the starting point for improvement is already real-time performance. Temporally dependent approaches fall into two categories, those where spatial fidelity is the independent parameter, and others where the temporal fidelity of one image component is exchanged for another.

#### 2.4.3.1 Spatial for Temporal

Spatial for temporal trade offs are distinguished from the converse trade off, temporal for spatial, by the dependent parameter which is temporal in the former case and spatial in the latter.

The spatial for temporal fidelity trade off is a frequent consideration in the design of material shader programs and is illustrated by the relative performance of the three illumination models evaluated by Boulos et al. [8] in an interactive ray tracing system with animated scenes. While the Cook style imagery, of Cook et al. [12], arguably provides the greatest spatial fidelity of the three; individual frames are more expensive and are rendered at a much lower temporal rate than the Whitted imagery, of [71], or direct illumination imagery. The choice between the fidelity levels provided by these algorithms, or other illumination and shading models, is usually made statically by the graphics system designer based on the expected workload of the rendering system.

Scene complexity or geometric level of detail is another common design decision that effects spatial fidelity. The survey of Luebke et al. [38] describes several approaches to adapt the geometry complexity of the scene during interactive rendering. Reducing geometry level of detail does not necessarily reduce spatial fidelity; in instances where the relative sample rate of the image compared with detail in the geometry is low, a coarse geometric level of detail serves to prefilter the visibility image component and reduce aliasing. Geometric level of detail approaches are employed by many of the spatially dependent mechanisms described in Section 2.4.2,

The adaptive refinement approach introduced by Bergman et al. [4] included the notion of a *golden thread* rendering procedure, a simple operation that could be repeated indefinitely to increase image fidelity; frameless rendering described by Bishop et al. [6] with further analysis provided by Zagier [76], is one such procedure. Instead of updating an entire frame buffer of pixels at one



time, the frameless rendering approach renders individual pixels in a random order and immediately displays them. The system proposes an extreme compromise between spatial and temporal fidelity; there is little latency between animation update and display since individual samples are displayed immediately, so the temporal fidelity of the approach is very high. However, in regions with temporal change, the random update procedure may not adequately or consistently sample spatial features, leading to lower spatial fidelity. The progressive assumption must be made to obtain adequate spatial sampling in nonstatic regions, as the image feature must remain unchanged for a sufficient amount of time that the random sampling process revisits it.

### 2.4.3.2 Temporal for Temporal

Precomputed radiance transfer, described by Sloan et al. [58] using spherical harmonics, and extended to higher frequency effects using wavelets by Ng et al. [48], caches lighting transfer functions from a preprocess such that the illumination of an object from multiple lights may be manipulated and combined at run time. While the class of techniques provides high spatial fidelity due to the sophisticated lighting effects such as indirect illumination, general animation of the scene is constrained by the precomputation. These constraints reduce the amount or type of temporal change permitted in the aggregate imagery, resulting in a temporal for spatial trade off.

Considering only the precomputed illumination image component, the combination of the progressive assumption applied to complex illumination effects, and temporal redundancy, is employed by Overbeck et al. [50] to increase the performance of an incremental precomputed radiance transfer approach. While the application is substantially different, Overbeck et al. employ a similar statistical analysis to the approach described in this dissertation to characterize the relationship between spatial and temporal image signal behavior. Both approaches employ wavelet transforms to characterize their respective signal, either illumination or final frame image samples, and both rely on the observation that low frequency temporal energy dominates the power spectrum. The principle difference between the use of the wavelet transform in these two approaches is that substantial computation and incremental update is performed in the wavelet scaleogram space by Overbeck et al., while the transform is only used for characterization in the process of making an adaptive control decision.

Another incremental approach exploiting temporal coherence of indirect illumination in an instant radiosity algorithm is described by Laine et al. [35]. Instant radiosity, described by Keller [32], is a spatial fidelity compromise where the diffuse radiance function across the geometric surfaces in the scene is sparsely and irregularly sampled by tracing rays from light sources to a set of virtual point lights at the first or second bound. Indirect illumination at visible points in the frame is reconstructed by combining the direct illumination from each virtual point light. The incremental

approach updates only a subset of the virtual point lights each frame based on the sample density on the hemispheres of the light source and a validity heuristic given by occlusion of existing virtual point lights due to changes in geometry or the light sources. In this approach, the temporal fidelity of the indirect illumination component is traded for increased fidelity of the visibility, texture color, and other components.

Gautron et al. [23] describe a radiance caching approach which exploits temporal coherence, analogous to spatial undersampling. The approach attempts to low pass filter a noisy discrete radiance signal to obtain a low power indirect illumination component. Filter shaping is performed by computing temporal gradients in the hemispherical space of incident light to attempt to smooth high frequency temporal artifacts such as tearing or popping. The temporal gradients are also used to compute an estimated lifespan of cached information based on a model of temporal validity. The approach uses a reprojection and a spatial upsampling mechanism similar to the Render Cache. Although the scope of the technique is different, again temporal reconstruction of a single image component versus final image samples, the use of a temporal gradient to guide the temporal sample rate of the illumination signal is similar to the approach described in Chapter 3 and distinguishes temporal radiance caching from other approaches which do not respond directly to characteristics of the cached signal.

Reverse reprojection, proposed by Nehab et al. [47], is a temporal cache lookup technique where a cache query is performed by projecting the image coordinates of a sample at the leading edge of time into the cache backwards through time, in contrast to a forward reprojection approach where the cache contents must be scattered or gathered based on the coordinates of the new frame. Reverse reprojection allows the cache to be organized as a uniform frame buffer with an implicit coordinate system, instead of a collection of cache entry structures relying on nonuniform coordinates. Sitthi-Amorn et al. [57, 56] employ a reverse reprojection mechanism to automatically improve the temporal performance of material shaders by selectively caching image components. In the final image, caching of selective image components may preserve fidelity by allowing higher frequencies of important features to be updated at a fast rate while lower frequencies of the same features are cached, e.g. the visibility image component contains very high frequency edges at object silhouettes. These features might not be cached by the system, while indirect illumination, a low frequency component, might be cached. When visibility and indirect illumination are combined by the rendering algorithm, the accurate silhouettes contribute more to the overall fidelity of the imagery than indirect illumination.

#### 2.4.4 Combined

Combined approaches manipulate both the spatial and temporal fidelity of the rendered imagery to increase the efficiency of the overall graphics system or implement a bias towards one type of fidelity or another. These approaches require very fine-grained control over the rendering since they must control both the varying spatial quality and temporal update of the rendered imagery.

The interruptible rendering approach, described by Wolley et al. [74], attempts to balance the level of spatial and temporal quality of an interactive dynamic scene through explicit control over frame refresh and rasterization resolution. The approach performs progressive refinement on a dynamic scene, by repeatedly rendering the scene at a single animation time with geometric level of detail. The change in spatial and temporal fidelity is estimated by the algorithm using a heuristic based on geometric level of detail and the displacement of the geometry by animation over time. While the spatial fidelity of the frame increases over time due to the refinement process, the temporal fidelity decreases due to the animation. When temporal error exceeds spatial error, the progressive refinement process is interrupted, and the frame is displayed. The temporal adaptivity employed by interruptible rendering does not require the progressive assumption since the algorithm expects and responds to temporal change in the imagery. The approach performs separable response since the action of refining the current frame or displaying the frame and then updating the animation exclusively improves either spatial or temporal fidelity.

### 2.5 Adaptive Frameless Rendering

The algorithm described in this dissertation is a direct descendent of the adaptive frameless rendering algorithm (AFR) described by Dayal et al. [14]; although the new approach introduces a number of capabilities not found in frameless rendering, its original motivation was to increase the efficiency of AFR by exploiting coherence in the rendering workload and increasing the parallelism and scalability of data structures and algorithms used for adaptive control.

Unlike the majority of approaches described in Section 2.4.3.2, these algorithms perform adaptive control using statistics computed from final image samples which are the reconstructed combination of all image components, evaluated at a specific spatial and temporal location.

Frameless rendering proposed by Bishop et al. [6] is an asynchronous screen refreshing mechanism where pixels are displayed immediately after they are rendered instead of synchronously after an entire frame is rendered. In a frameless renderer, pixels are selected for rendering and refresh randomly across the image over time so as not to bias any one region. This results in an image containing samples taken at many different moments in time, and introduces temporal sampling artifacts in dynamic scenes. Adaptive frameless rendering (AFR) modifies the distribution

of samples across the image in both space and time to more efficiently sample the changing image [14].

In AFR, response to spectral changes in the underlying signal is performed through modifications to a kd-tree which is used to guide individual sample placement. During rendering, leaves of the kd-tree are selected for sampling with equal probability, and the leaves are sized in screen space based on an error estimate. The sample tile with maximum error is split into two tiles and the two sibling tiles with minimum total error are merged into a single tile. Through this mechanism, the probability of sampling the region with maximum error increases at the expense of the region with minimum error. Over time, this leads to an increased relative sampling density in certain regions and a decreased relative density in others.

The error estimation function is based on variance across the tile weighted by sample age. Older individual samples within a tile contribute less than current ones. This variance error metric is based on ray sample convergence and is conservative. If a tile has low variance, then the samples within it must have converged, and therefore, the tile region is likely not undersampling the underlying image function. The mechanism is efficient to implement, and its effect seems reasonable for a single ray-ray tracer-based system. The sampler increases the likelihood that a certain region will be sampled by increasing the number of tiles in the region until the individual samples converge.

The TSAR model described in Chapter 3 is an extension of the adaptive frameless rendering (AFR) algorithm described by Dayal et al. [14, 13]. AFR software designs using the Manta interactive ray tracer allowed exploration of the design space of parallel adaptive sampling implementations. This experience led to the generalized model and prototype design in this dissertation. The new model has several differences: single sample rendering is not required, spatial and temporal response is explicitly decoupled, and rendering and communication between pipeline stages is not explicitly frameless. There are differences in the design of the prototype too: samples are gathered from scalable hierarchical search structures instead of scattered into fixed size deep buffers, and the TSAR model is generalized such that a kd-tree tiling might not be used in the sampler.

Dayal's treatment of statistical entropy in AFR experiments provides evidence of the redundancy which may be exploited by the TSAR algorithm. The principle result is that the amount of entropy, or quantifiable information content, in AFR images is approximately twice that of uniformly oversampled images. The AFR process was able to double the amount of information using 40 percent fewer samples than uniform rendering. Entropy is given by the minimum possible size representation, so it is still likely that both the AFR and uniform processes produced redundant samples; still, the smaller number of total samples used by AFR is evidence that it was able to exploit considerable redundancy.

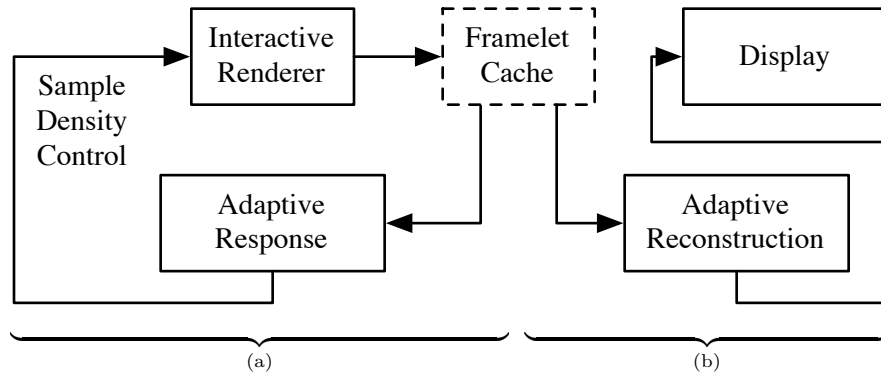
# CHAPTER 3

## TEMPORALLY AND SPATIALLY ADAPTIVE RENDERING

This chapter describes the temporally and spatially adaptive rendering (TSAR) algorithm and a parallel software framework used to implement it; subsequent chapters describe the mathematical models employed by each stage of the algorithm. TSAR is a rendering control system with two fundamental goals: first, to determine an efficient spatial and temporal sampling rate which eliminates redundant rendering work, and second, to redistribute surplus rendering capacity to meet a fidelity strategy selected for the system. Fidelity management strategies, described in Section 2.4, range from simple compromises between spatial and temporal fidelity, e.g. eliminating as much spatial redundancy as possible without introducing significant error, or spatial fidelity loss; to more sophisticated adaptive strategies that bias spatial or temporal fidelity at different locations across the image and over time.

The TSAR algorithm, shown in Figure 3.1, adds two substantial stages to the end of the canonical graphics pipeline of an interactive renderer. The first stage, labeled 3.1a, adaptively responds to changes in the statistical characteristics of the rendered imagery and adjusts the sample rate configuration of the renderer. The second stage, labeled 3.1b, reconstructs complete image frames for viewing using batches of samples rendered at different spatial and temporal sample rates. Unlike the single sample approaches of other techniques [67, 14], the two components of the TSAR algorithm operate on temporally coherence batches of samples called *framelets*. Framelet sample containers are produced by the interactive renderer and stored in a common buffer, or *framelet cache*, which is accessed by both stages.

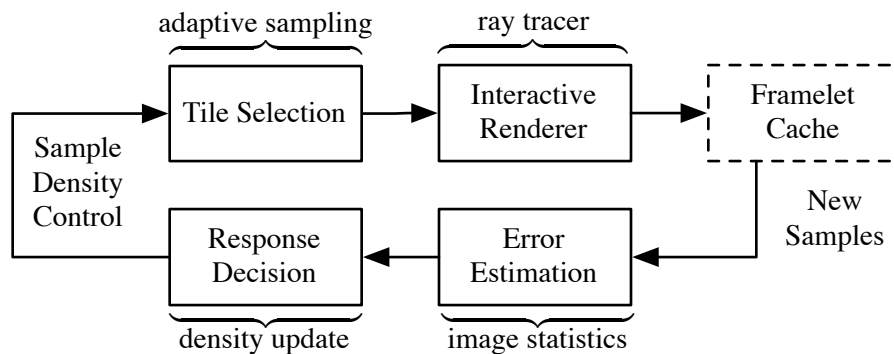
The response and reconstruction components of the TSAR algorithm may be attached to any interactive rendering system which provides the capability to render small image tiles at different spatial sample rates. Temporal adaptivity is achieved by varying the refresh rate of image tiles. Fine-grained control of the sample rate in this manner is possible in both rasterization and ray tracing rendering systems. Control of interactive ray tracing systems is the focus of this dissertation because the per-sample cost of evaluating image pixels is greater. The central hypothesis tested by the TSAR prototype is that adaptive response eliminates a sufficient quantity of redundant rendering



**Figure 3.1.** Response and reconstruction. The interactive rendering and adaptive response components (a) communicate with reconstruction and image display (b) through the framelet cache buffer.

work, with minimal computational overhead, such that the overall spatiotemporal fidelity of the rendering system improves.

This dissertation addresses the efficiency and performance of the adaptive response stage by introducing an efficient model to measure spatial and temporal sample rate error and then respond by modulating individual sample densities across the image. The adaptive response stage is divided into three components: tile selection, or sample placement; error estimation; and adaptive response decision, shown in Figure 3.2. These three stages are the topics of Chapter 4, Chapter 5, and Chapter 6, respectively. Efficiency and performance are obtained through parallel implementation and amortization of expensive operations, e.g. the sample placement stage which controls the interactive renderer, places samples in large batches instead of individually, the estimation and response stages are formulated for parallel efficiency. The efficiency and visual performance of the adaptive reconstruction stage in the TSAR model is described in Section 4.2, but is not the focus of this dissertation.



**Figure 3.2.** Stages of adaptive response.

### 3.1 Adaptive Sampling

The adaptive sampling mechanism, labeled *tile selection* in Figure 3.2, controls the rendering work performed by the ray tracer by selecting regions of the image for rendering at each animation time step. The image is divided into discrete tiles, or framelets, containing the same number of samples but varying in spatial extent. Regions of the image with smaller framelet tiles have higher spatial sample rates. Temporal adaptivity is achieved by varying the refresh rate of individual tiles across the image. Framelet tiles that are selected for rendering more frequently have higher temporal sample rates. Tile selection is guided deterministically by a pair of spatial and temporal sample density fields defined across the image.

The TSAR algorithm uses two different representations of the spatial and temporal sample rate across the image. The tile selection, rendering, and reconstruction stages operate on framelet tiles, while the error estimation and response decision stages operate on scalar fields. These two representations are suitable for different tasks in the pipeline. The framelet tiling is a hierarchical multiresolution structure with exceptional data locality within individual tiles, but over which spatial search or neighbor finding is inefficient. The scalar field representation is a flat data structure, in either a Cartesian or space filling data layout. Within the scalar field, neighbor finding or spatial search is very efficient; however, the resolution of the field is uniform.

In addition to data layout and spatial query suitability, density representation also depends on the type of parallelism exhibited by respective stages. The framelet tile representation of sample density is principally used when two criteria are met: performance is obtained from amortization over large data batches, and when the appropriate unit of parallelism is the framelet tile, with little data communication between tiles, e.g. sample rendering in the ray tracer. Scalar fields, stored at a much lower resolution than the smallest framelet tile, are used when parallel global communication is necessary; and when spatial search, or neighbor finding is necessary, e.g. solution to the response decision optimization problem using a spatial partial differential equation.

#### 3.1.1 Sample Density Field

The adaptive rendering system maintains a spatial and a temporal sampling rate which are defined by separate sampling functions  $\mathcal{S}_s$ , and  $\mathcal{S}_t$ . This formulation of the TSAR model assumes that the spatial dimensions are sampled isotropically, with the same sample density in the horizontal and vertical dimensions, i.e.  $\mathcal{S}_x = \mathcal{S}_y = \mathcal{S}_s$ . The assumption is not required by the model, but simplifies the implementation and enables a three-dimensional spatiotemporal volume to be represented in two dimensions. For uniform sampling, both  $\mathcal{S}_s$  and  $\mathcal{S}_t$  have the form:

$$\mathcal{S}_t(t) = \sum_{k=1}^N \delta\left(t - k\frac{1}{N}\right) \quad (3.1)$$

In a nonuniform, or adaptive case, the distance between Dirac delta functions would be modulated based on a function of  $t$ .

The sampling functions may be expressed as spatial and temporal sample densities by convolving the Dirac comb functions with a density reconstruction kernel, to obtain a continuous scalar field, termed the sample density function. The kernel must only be broad enough to resample  $\mathcal{S}$  to the resolution of  $\theta$ , i.e. a box filter of width  $\frac{1}{N}$ :

$$\theta_s = \mathcal{S}_s(s) * K(s) \quad (3.2)$$

$$\theta_t = \mathcal{S}_t(t) * K(t) \quad (3.3)$$

The exact sampling functions used by the interactive renderer in Figure 3.2 need not be known or controlled by the adaptive response system. Instead, the scalar density fields  $\theta_s$  and  $\theta_t$  are communicated opaquely through the tile selection stage, during which framelet tiles containing a specific number of samples, or amount of sample density  $\theta$ , are communicated to the renderer for sampling. The interactive renderer samples the spatial extent of the tile using its native sample placement mechanism, and then resamples the results to the uniform grid of framelet sample positions. Later, the adaptive response error estimation stage uses image samples placed within framelets based on  $\mathcal{S}_s$ , and  $\mathcal{S}_t$  to perform analysis of the imagery and adaptive response.

In the spatiotemporal volume, over the course of an interactive graphics session, the sample density functions indirectly represent sample rates on a spatial plane perpendicular to the temporal axis at the leading edge of time. Therefore, although sample density control is performed over two spatial dimensions and one temporal dimension, the solution, i.e. the fields  $\theta_s$  and  $\theta_t$ , vary only in the horizontal and vertical dimensions and are represented by a pair of two-dimensional fields, or a two-dimensional vector field.

Control of the temporal sample rate is similarly opaque; temporal adaptivity in the tile selection stage is performed by either immediately refreshing, or differing until the future, each framelet in the tiling. In this respect, the  $\theta$  field is two and a half-dimensional, in that the temporal dimension adjusts sampling behavior either at the leading edge of time, or just beyond it, i.e. without selecting a specific temporal coordinate in the future.



The combined spatial and temporal sample density may be expressed as field of spatiotemporal volume elements where each element has uniform size, i.e. width and height  $\frac{1}{N}$ , and depth in the temporal direction  $\Delta t$ . The spatial resolution  $N$  is a free parameter which determines the degree to which operations on elements are amortized across the image domain,  $\Delta t$  is the temporal refresh unit. As shown in Figure 3.3, the elements have uniform size in space and time, but the number of samples or density within each element varies.

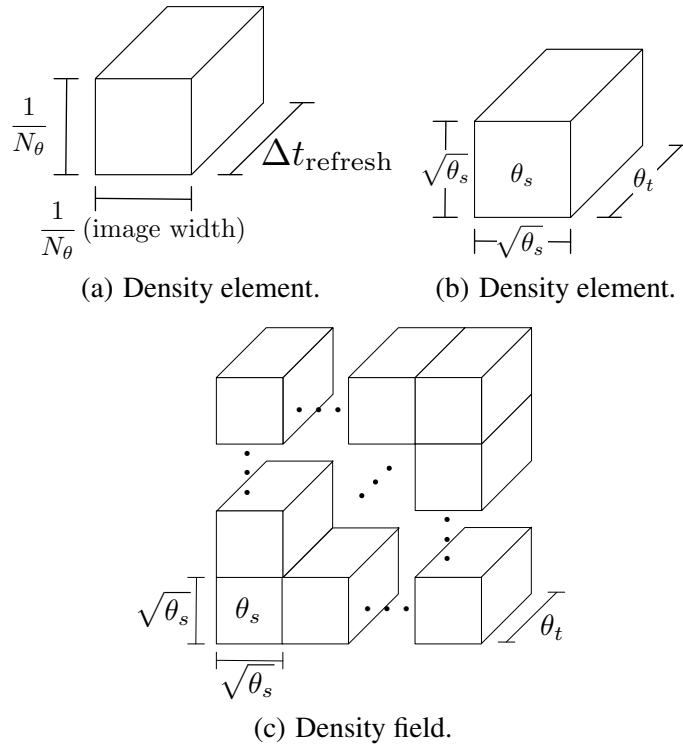
Interpretation of  $\theta_s$  and  $\theta_t$  as a field of volumetric elements enables spatial constraints and systems of equations to be written, both in terms of sample density along each dimension, and the total amount of combined density, or volume density, of the element. These constraints are enforced in the adaptive response control decision stage, which modulates, or transports, density across the image while maintaining conservation and positivity.

### 3.1.2 Framelet Tiling

An ideal adaptive rendering system would allow arbitrary placement of samples in space and time, and obey continuously varying spatial and temporal sample density functions, both with minimal overhead compared to a nonadaptive system. The AFR system accomplished arbitrary placement by selecting sample positions based on a probability distribution; however, probabilistic sample placement conflicted with the ray packet amortization mechanism of the underlying rendering system and dramatically increased the overhead of adaptive sampling.

Conventional parallel rendering systems including graphics processors and software ray tracing systems obtain high performance by amortizing the cost of operations over a large set of operands. In the Manta system, primary ray image samples are traced in packets of rays covering tiles across the image. The cost of acceleration structure traversal and geometry intersection for the whole packet may be reduced by testing with the packet frustum, instead of individual rays, and tracking special cases or optimization characteristics, such as packets with a common ray origin. Spatial coherence between rays in the packet increases the coherence of memory access to spatial structures and increases the efficiency of large memory caches. Ray packets also organize operand data into coherent regions of memory, suitable for operation with vectorized instruction sets. Ray packet tiles in Manta nominally contain a maximum of 64 rays; packets are used both for tracing primary rays which produce samples on the image plane, and for tracing secondary effects such as shadows and reflections.

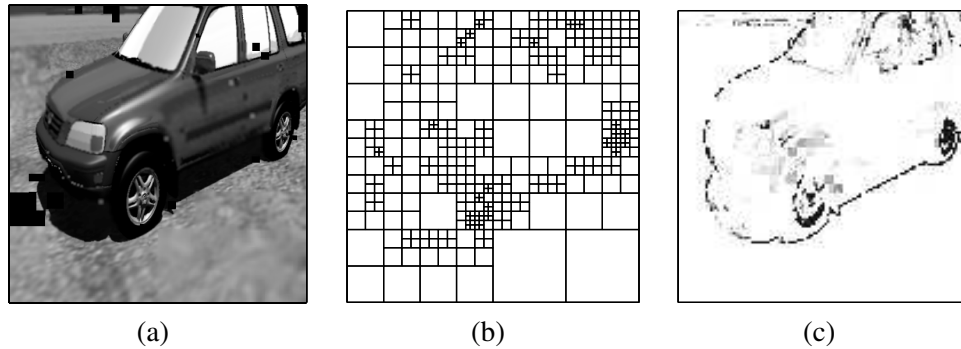
The framelet tiles in the TSAR algorithm are analogous to rasterization tiles or ray packets. The tiles themselves are containers of image samples on a uniform grid, and are the principal unit of parallel work for the rendering stage through error estimation. Framelets contain a greater number of samples than ray packets or rasterization tiles, nominally 256 or 1024, both to increase the amount



**Figure 3.3.** Spatiotemporal density volume element and field.

of amortization and to provide a sufficient signal window for discrete spectral transforms which are used for sample rate error estimation. Figure 3.4(a) shows an image reconstructed by averaging overlapping samples in the framelet cache, the framelet tiling used to produce samples is shown in Figure 3.4(b), and the sample density function used to produce the tiling is shown in Figure 3.4(c).

The tiling, or spatial arrangement of individual framelets, is a piecewise constant approximation of the sample density field. Each framelet consists of a fixed number of spatial samples but the spatial extent of the framelet varies, changing the resolution of the uniform grid of samples within the framelet. The interactive renderer need not rasterize samples or trace rays matching the framelet sample grid exactly; however, the discrete image samples produced by the renderer must eventually be resampled to the framelet grid on which the error estimation stages of the response model rely. The TSAR model allows for the interactive renderer to sample the graphics scene at a higher rate than framelet resolution, and there is no control over the degree to which secondary effects such as shadows or reflections are sampled, as long as the renderer output is resampled to the framelet resolution. In super sampling cases where the primary ray sample rate is higher than framelet resolution, the resampling operation to framelet resolution is likely a low pass filter which limits the high frequencies detected during the error estimation stage.



**Figure 3.4.** Example framelet tiling. Naively reconstructed frame (a), framelet tiling (b), and spatial sample density function (c).

### 3.1.2.1 Tiling Alternatives

There is a fundamental tension in the design between uniform spatial tilings that have variable sample rates in each tile, and those with hierarchical or nonuniform spatial tilings with uniform sample density. The former is used in image compression, where the input is divided into spatial tiles, and each tile is encoded with a varying number of bits. Certain operations, such as spatial search, memory lookup, or overlap detection appear to be easier in the uniform tiling case, since the number and position of tiles does not change. This is especially true in the case of image compression, where a tile division and encoding of the image is performed once and then decoded and accessed many times. While each operation may be performed in constant time with a uniform tiling, a binary search, and possibly a lookup structure such as an octree or a kd-tree, is necessary in the nonuniform case.

In adaptive rendering, spatial tilings are used in a more dynamic fashion where the tiling must be constructed, filled with data, and then queried multiple times per refresh cycle. Given these operations, specifically frequent update or construction, the disadvantages of both designs are largely equivalent. Parallel rendering engine coherence criterion requires that sampling work be submitted with a structured organization, often in fixed size, to obtain performance. In the uniform tiling case, organization of samples for submission to the rendering engine would require some type of subdivision, essentially a subtiling, within each high level spatial tile. This sublevel tiling would possess all of the problems of a nonuniform high level tiling. Essentially, the uniform tiling would need to be turned into a fixed minimum depth in a hierarchical tiling in order to be submitted to the rendering engine.

Uniform tilings, in the form of scalar fields, are used to communicate two scalar fields in the TSAR algorithm: sample rate error and the sample density field. The latter is a high precision representation of the framelet tiling, e.g. the framelet tiling shown in Figure 3.4(b) has only four

different tile sizes or resolutions; however, there are more than four scalar density values within the corresponding field shown in Figure 3.4(b). The uniform tiling or scalar field representation is used for communication between the error estimation stage and the tile selection stage of the adaptive response system. Unlike the framelet tiling containing rendered samples, in these stages, each tile in the scalar field has a fixed amount of data, i.e. one scalar, so the problems of search and memory lookup are not encountered.

The specific data structures and algorithms used to construct spatial tilings, query their contents, and resample them to uniform fields, is described in Chapter 4.

The framelet tiling and sample density volume element field are nearly opposites of each other; in the framelet tiling, each tile has an equal number of samples, but varying spatial extent. While in the sample density fields, or spatiotemporal volume element field, each element has equal extent, but a varying sample density amount.

## 3.2 Sample Rate Error Estimation

The first goal of the TSAR control algorithm is to identify redundant spatial and temporal sampling work performed by the rendering system. Elimination of this redundancy increases efficiency and produces surplus rendering capacity. The second goal of the algorithm is to redistribute surplus capacity to other regions of the image to reduce aliasing and increase fidelity, or to bias spatial and temporal fidelity in some other manner.

### 3.2.1 Characterization of Redundancy

Redundant rendering work, in terms of extra sampling of the graphics scene on the image plane, occurs when an increase of the sample rate produces diminishing improvements to the fidelity of the rendered imagery. While output fidelity is a function of both the adaptive response and adaptive reconstruction components, shown in Figure 3.1, redundancy detected during adaptive response must be defined in terms of the image samples available in the framelet cache, without considering reconstruction which is a downstream stage in the pipeline. At a single infinitesimal point in the image, the efficiency of the current sample rate is proportional to the amount of additional energy in the frequency domain captured if the sample rate is increased; i.e. the second derivative of an integral over a low pass band in the power spectrum:

$$\text{Efficiency}(2f) = \frac{d^2}{df^2} \left[ \int_1^f \mathcal{F}^s(s) ds \right] \quad (3.4)$$

Across a region of the image, such as the extent of a framelet tile, the efficiency of the sample rate may decrease more quickly in some areas than in others, e.g. the efficiency of increased sample rates in low frequency areas is less than along sharp features. If the sample rate is allowed to vary continuously, then at each point in the image, the sample rate  $2f$  is selected at such a point that the efficiency drops below a certain threshold. In the TSAR model, the sample density must be fixed for small regions of the image, i.e. within the framelet tiles. The amount of sample rate redundancy is the sum of the differences between the fixed single sample rate for the tile  $f_{\text{tile}}$ , and each respective maximally efficient sample rate, in set  $\theta^*$ , across the tile:

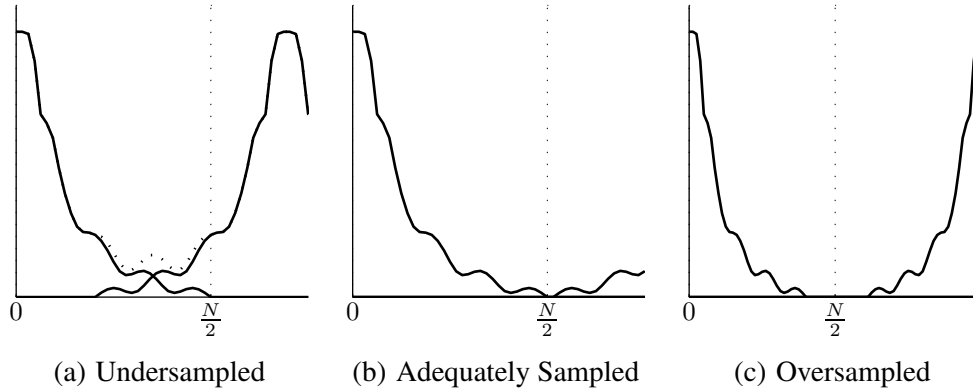
$$\begin{aligned} f_{\text{tile}} &= \max_{f \in \frac{1}{2}\theta_{\text{tile}}^*} (f) \\ \text{Redundancy}(\theta_{\text{tile}}^*) &= \sum_{f \in \frac{1}{2}\theta_{\text{tile}}^*} |f_{\text{tile}} - f| \end{aligned} \quad (3.5)$$

Based on the efficiency of the sample rate across the image and the amount of redundancy incurred by fixed rates within framelet tiles, the adaptive response algorithm classifies the sample rate in one of three ways: oversampled, adequately sampled, or undersampled. These classifications result in an adaptive response of decreasing, leaving unchanged, or increasing, the sample density within a region of the image. The power spectrum of these behaviors for a one-dimensional synthetic signal is illustrated in Figure 3.5; the frequency  $\frac{N}{2}$  indicates the highest frequency captured by a framelet with resolution  $N$ .

The shape and magnitude of the sampled signal's power spectrum provide a response direction and magnitude, and constitute a signed estimate of the the amount which the sample density should change within a region of the image. Characterization of the underlying image signal in terms of an unknown ideal sample rate, and the distance to it in the spatial and temporal dimensions, is considered in Section 5.1.2.1.

### 3.3 Adaptive Response Decision

The sample rate error estimation stage produces an estimate of the difference between the current spatial and temporal sample density and the ideal sample densities based on a local characterization of the rendered image. The estimate is made based on the signal alone, without regard to bounds on the current sample density, or the overall distribution of rendering work across the image. In the adaptive response decision stage, the locally greedy error estimates made independently across the image are modulated to adhere to local constraints on the spatial and temporal sample rates and



**Figure 3.5.** Characterizations of signal redundancy in one dimension, using the frequency domain power spectrum.

global constraints on the combined sample density field as a whole.

### 3.3.1 Constraints on Sample Density

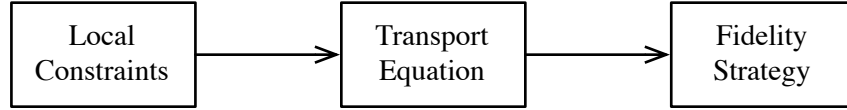
The application of constraints are solved in three steps, shown in Figure 3.6. First, local constraints are enforced on the error estimate, and a combined change in the joint spatial and temporal sample density  $\Delta\theta$  is computed. Next, the combined change at each element across the sample density field is modulated to adhere to global constraints. Lastly, the modulated amount of combined sample density change is redistributed between the spatial and temporal sample density at each element. Modulation in the last step may be used to implement a specific fidelity strategy, e.g. to bias temporal response over spatial response or vice versa.

Interpreted, in Figure 3.7, as a field of spatiotemporal volumetric elements, the first step computes the net flux  $\Delta\theta$  in volumetric sample density across the element. The total sample density flux is transported across the domain in the second step, and remixed into spatial and temporal density components in the last step.

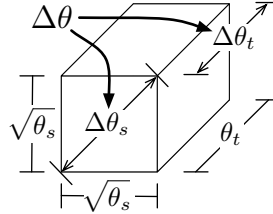
#### 3.3.1.1 Positivity and bounds

The sample density must be greater than zero across the domain, i.e. the density of each volume element must be greater than zero. The adaptive renderer is only capable of rendering samples at discrete sample rates between a minimum and maximum; these bounds define, in part, the  $\theta_{\text{ideal}}$  sampling density across the image:

$$\bigwedge_m : 0 < \theta_{\min} \leq \theta(m) \leq \theta_{\max} \quad (3.6)$$



**Figure 3.6.** Adaptive response control decision.



**Figure 3.7.** Manipulation of the spatial and temporal sample densities. The net flux across the element  $\Delta\theta$  is divided between a change in density in the spatial dimensions  $\Delta\theta_s$  and temporal dimensions  $\Delta\theta_t$ .  $\Delta\theta$  is determined by global constraints, while  $\Delta\theta_s$  and  $\Delta\theta_t$  are the result of a mixture process based on sample rate error.

Unique bounds may be specified for  $\theta_s$  and  $\theta_t$ . The parameter  $m$  specifies a position in the field.

### 3.3.1.2 Conservation

The total amount of work performed by the adaptive renderer over a suitable spatiotemporal extent must not change. If the total amount of sampling work increases, the cost of a framelet batch may cause the maximum possible temporal resolution to decrease. If the total amount of sampling work decreases substantially, the renderer may not be fully subscribed leading to wasted capacity. Therefore, the total change across time over the sample density field must be zero. For combined sample densities  $\theta^n = \theta_s^n \cdot \theta_t^n$ :

$$\sum \theta^{n+1}(m) - \sum \theta^n(m) + C = 0 \quad (3.7)$$

Sample density is strictly conserved if the surplus term  $C$  is omitted. If a running surplus of sample density is permitted, then the net flux across the domain may not exceed the total running surplus:

$$\sum_{i=0}^n C^i < C^{n+1} \quad (3.8)$$

Underutilization of the renderer will likely occur for  $C \gg 0$ .

### 3.4 Software Architecture

The TSAR prototype software is implemented within the Manta pipeline architecture [61, 5] and combines the task parallel operations on a multicore CPU with data parallel operations of the GPU. Ray tracing, scene animation, and tile scheduling are performed by threads on the CPU, while error estimation, the adaptive response control solver, tile update, and reconstruction, are performed on the GPU. Pipeline organization with a three frame depth minimizes the amount of time either processor must wait for data from the other.

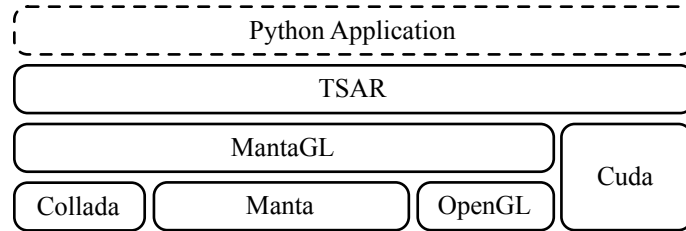
The prototype is designed to operate either in a full ray tracing mode, where an entire image frame is reconstructed and displayed by a rasterizer, or in a hybrid configuration, where a texture containing partial ray traced effects is reconstructed and used in a multiple pass rasterization algorithm. The hybrid configuration also allows rasterized information such as the depth buffer of the current frame to aid reconstruction.

The software architecture consists of three programming interface layers, shown in Figure 3.8. The lowest layer components are the Manta ray tracer, OpenGL rasterizer, and Manta Scene Graph (MSG) which is used to control animation and scene properties. The Manta renderer provides an opaque interface for evaluating image samples through ray tracing. Its architecture provides a scalable, structured, parallel pipeline to control CPU threads, which is used by higher layers in the TSAR design. The next highest layer, the MantaGL pipeline, controls coordination and communication between the MSG animation, the Manta renderer, and the OpenGL rasterization components. The third highest layer implements the TSAR control algorithm, by submitting rendering work to the MantaGL layer, and processing image samples produced by it. The majority of the TSAR layer is implemented with CUDA [49] programs running on the GPU. At the top level, a driving graphics application obtains reconstructed images from the TSAR layer and provides animation commands or user input to the Collada, Manta, and OpenGL components. Depending on the runtime configuration, the adaptive renderer will either produce whole frame images, or partial frame images to be used as textures by the graphics application.

#### 3.4.1 Parallel Hardware

The TSAR prototype is designed to be used on commodity desktop workstation computer systems, consisting of a multicore CPU and a general purpose computation capable GPU. The principal challenge of the software architecture is to saturate the parallel capacity of the whole system by keeping both processors fully subscribed as much of the time as possible.





**Figure 3.8.** TSAR prototype software stack.

Following discrete graphics hardware terminology, components of the rendering system residing on the GPU plugin board are collectively referred to as the *device*, while the remainder of the system, including main memory and the CPU, is referred to collectively as the *host*. The GPU device employed by the TSAR prototype operates in two separate modes; OpenGL rasterization rendering is performed in *graphics* mode, while CUDA programs are executed in *compute* mode. While data may be shared between programs running in graphics and compute mode on the device, the two modes are mutually exclusive and may not execute concurrently.

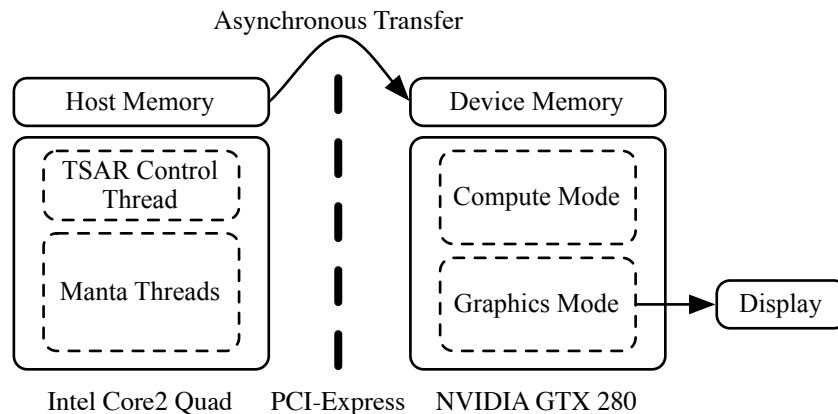
The memory system of the prototype workstation is divided between separate memories on the host and device. In the prototype, communication between memories is performed in large asynchronous batch transfers. The TSAR pipeline performs a major data transfer between each stage in the pipeline, once between ray tracing and adaptive response, and once between adaptive response and reconstruction.

The software design concerns scheduling tasks between four threads of execution shown in Figure 3.9. Ray tracing worker threads, treated as a single entity, and the graphics control thread execute concurrently on the host; graphics mode rasterization operations, and compute mode thread execute on the device.

### 3.4.2 Manta Interface

The Manta programming layer provides an opaque interface to a scalable parallel ray tracing engine. The pipelined architecture of the Manta engine, and the constraints that the model places on communication and synchronization, allow the renderer to scale efficiently. The pipelined rendering and display components eliminate the serial image display bottleneck and statically load balanced callback routines, are used to safely coordinate between ray tracing on the CPU and analysis, or additional rendering passes on the GPU.

The simplified Manta parallel pipeline shown in Figure 3.10 contains four components divided between two stages. Although the figure shows only two rendering threads plus an application thread, the pipeline may scale to hundreds of threads on a large multiprocessor super computer.



**Figure 3.9.** Prototype hardware architecture.

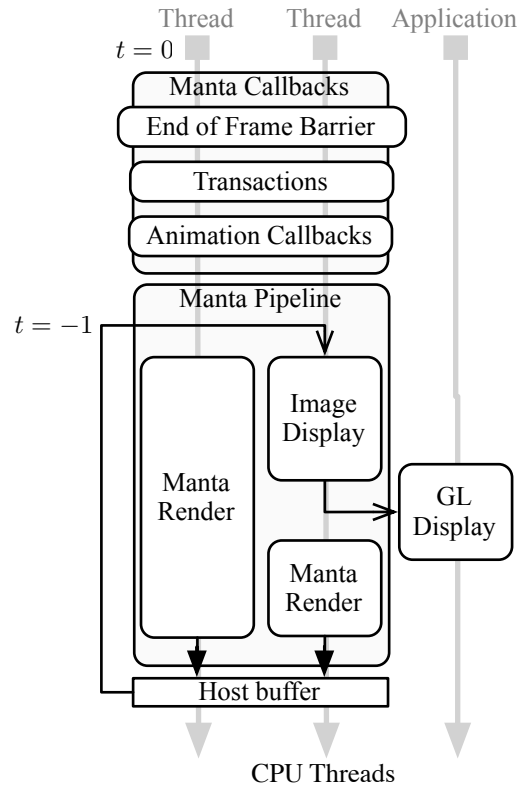
The pipeline is two stages deep and induces a one-frame latency. At time  $t$ , the scene animation is sampled and rays are traced while the image previously rendered at  $t - \Delta t$  is displayed. The one frame latency is indicated by the upstream connection between the host buffer and image display components in the figure. In the Manta pipeline architecture, configuration and animation tasks, in the callback section of the figure, are statically load balanced, while rendering is dynamically load balanced. By performing image display of the previous frame in a single thread concurrently with ray tracing of the current frame, the serial display task does not cause a bottleneck in the pipeline since all threads remain occupied and the display thread can obtain rendering work when finished. The objective of the MantaGL and TSAR parallel pipelines is to hide the serial overhead of similar operations by adding additional stages to the pipeline.

### 3.4.2.1 Timing

The prototype uses the timing facilities provided by Manta, which may be configured in a fixed rate or wall clock mode, as the clock for animation. The fixed rate time mode is used to record information from software counters, or to save comprehensive snapshots of the system's state to disk for later inspection. The data shown in most of the figures containing fine grained system behavior were created using a fixed time rate simulation. In wall clock time mode, the Manta control loop starts each new frame as soon as all rendering threads have completed the previous frame.

### 3.4.2.2 Transactions and Callbacks

The Manta control loop uses a transaction mechanism based on callbacks into application code to update configuration safely at the beginning of each cycle through the loop. Depending on type, one or more Manta threads may execute each callback concurrently while the other rendering



**Figure 3.10.** Manta pipeline.

threads wait at a barrier.

The MantaGL and TSAR layers use animation callbacks executed by the rendering threads at the beginning of the renderer control loop to coordinate ray tracing with operations with the graphics control thread. Since animation callbacks are invoked on every cycle through the Manta control loop, they permit the Manta clock and new frame events to be safely communicated to other threads through semaphores or barriers. The MantaGL pipeline uses an animation callback with two barriers to synchronize OpenGL interaction with the Manta renderer.

### 3.4.2.3 Image Display

Image display is the second stage in the Manta rendering pipeline and is executed by a single thread immediately after the animation callback phase of the control loop, concurrently with the ray tracing pipeline stage. Three forms of image display are supported within Manta; the display thread may establish a separate connection to an X11 server and transmit the image asynchronously to the driving graphics application, the display thread may synchronize with the application thread and allow it to execute graphics copy commands before both threads proceed, or the display thread may communicate a pointer to the image buffer to the application thread and then proceed immediately.

The last case requires the least synchronization between the ray tracing and GPU side of the system; however, the shared buffer must be protected by a semaphore since it is accessed asynchronously, at different times, by either thread. After the image buffer handoff is complete, the image display thread continues executing rendering work with the other Manta threads.

#### 3.4.2.4 Ray Tracing

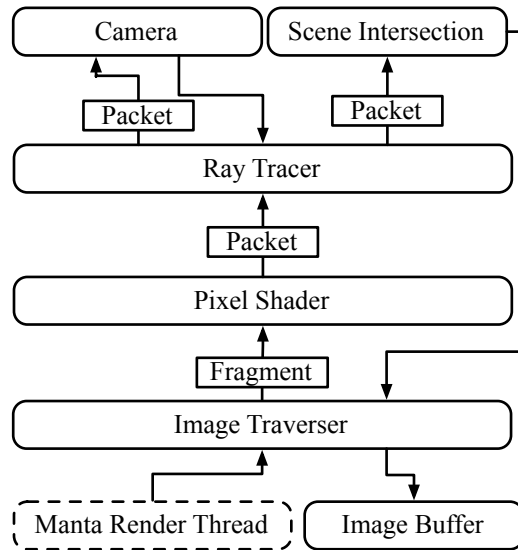
Rendering is the first stage for a new frame in the Manta pipeline; it is executed by all but one thread immediately after the transaction and callback phase of the Manta control loop. During the rendering stage, each thread executes a set of modular functions to partition and order rendering work for each frame. With the exception of mutual exclusion around a queue used for work distribution, there is no communication between threads executing the rendering stack. Each function in the stack, shown in Figure 3.11, divides rendering work into smaller and smaller pieces, between which there is little data dependence or communication. First the image is divided into *fragments* containing individual output pixels for ray tracing. Rendering work within fragment pixels is divided into *ray packets* which are populated with viewing rays by a camera and then intersected with the graphics scene.

The configuration of the Manta rendering stack is unsuitable for adaptive rendering for two reasons: the image traverser assigns rendering work composed of batches of individual pixels, and the output image produced by the rendering stack is resampled to pixel resolution. Work distribution between threads is based on output image pixels; the renderer filters subpixel samples into a single value which is written to the framebuffer, instead of preserving information at a higher resolution. In an adaptive renderer, the spatial sample density may vary from significant undersampling, e.g. less than one sample per pixel, to supersampling with 16-64 samples per pixel; the assumption that each pixel would require the same amount of work, or at least the same number of primary ray image samples, is not valid. High resolution information, such as subpixel image sample values, is necessary to adequately characterize the image signal; this information must be computed and stored by the ray tracer for use during sample rate error estimation.

### 3.4.3 MantaGL Interface

While the lower level Manta layer provides an opaque interface for tracing primary image sample rays, the MantaGL layer provides an interface that abstracts the data transfer required for communication between the Manta ray tracing threads on the CPU and compute or graphics tasks on the GPU. The layer also synchronizes the camera models used by the rasterizer and ray tracer such that corresponding pixel coordinates can be obtained in both renderers.

Implementation of the MantaGL and TSAR programming interfaces is accomplished by making



**Figure 3.11.** Manta rendering stack.

small modifications to several modular components in the Manta architecture to track framelet information and enable adaptive sample placement, and by adding a set of callbacks to coordinate between Manta rendering threads and the GPU control thread.

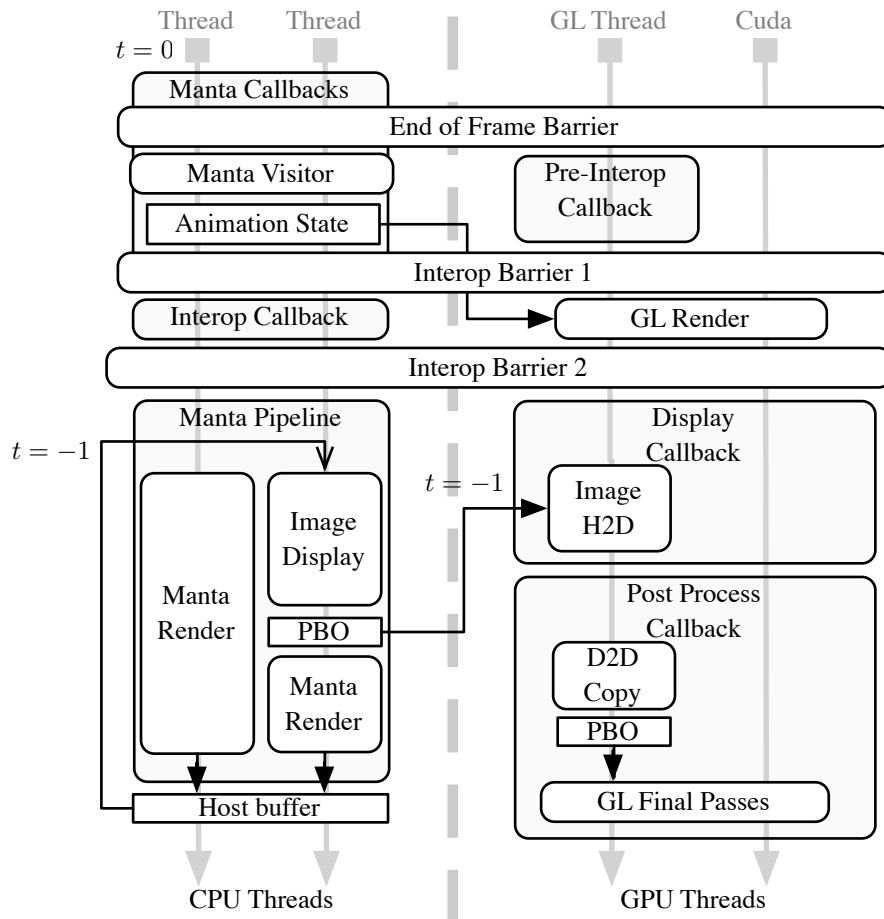
The MantaGL pipeline is shown in Figure 3.12; the pipeline spans three types of threads on two processors: Manta threads executing on the CPU, to the left of the dashed line, and graphics and compute threads executing on the GPU to the right of the dashed line. All threads execute asynchronously, although the graphics compute processes are controlled by a separate application thread running on the CPU. The MantaGL pipeline is divided into four callbacks within which the TSAR pipeline is implemented.

### 3.4.3.1 Animation

The TSAR prototype uses the MSG library to execute animations loaded from Collada format files. These animations enable dynamic scenes with rigid body and skinned movement and are parameterized on time. Finegrained control over scene animation and the amount of temporal change, as opposed to experimenting with user input, allow for repeatable experiments.

Animation update based on the current frame time is performed during an animation callback, described in Section 3.4.2.2, during which a Manta thread running on the CPU traverses the scene graph with a update visitor, transforms geometry, and updates or rebuilds acceleration structures.

The graphics control thread executes a *pre-interoperation* callback concurrently with animation update in Manta; although the default implementation is empty, this callback is used to copy to the new framelet tiling from the device to the host in the TSAR pipeline.



**Figure 3.12.** MantaGL pipeline.

### 3.4.3.2 Interoperation

The MantaGL pipeline contains two interoperation barriers encountered before the ray tracer begins processing the current frame and the adaptive response algorithm is executed on the GPU; these barriers synchronize initialization of the new frame, and a new animation time stamp, within both engines.

The first interoperation barrier separates animation state update on the CPU from rasterization of the new frame on the GPU. The second barrier separates the initial GPU scene rasterization pass from ray tracing for the new frame, and adaptive response on the GPU. In a hybrid rasterization and ray tracing configuration, the initial graphics pass would produce input to the ray tracer determining which regions of the image require sampling. In the nonhybrid case, the ray tracer simply samples the whole image.

Callbacks labeled *pre-interop* and *interop* are provided to execute work on the idle processor before the first and second interoperation barriers where the CPU and GPU are idle, respectively. In

the TSAR layer, these callbacks are used to copy the new framelet tiling to the host and then execute the tile selection algorithm.

### 3.4.3.3 Image Transfer

The MantaGL pipeline uses a specialized Manta image buffer implementation containing memory buffers on both the host and device. Both buffers store color images with a depth component that may be used to composite ray traced and rasterized imagery.

Ray tracing a batch of framelets is the greatest bottleneck in the system; the fast semaphore protected handoff allows the Manta image display thread to perform a very small amount of work, i.e. only copying a pointer, before joining the other rendering threads. The asynchronous handoff, and subsequent host to device copy by the GL control thread, occurs concurrently with frame reconstruction.

## 3.4.4 TSAR Interface

While the Manta pipeline has a two-frame latency, the TSAR pipeline is three stages deep, resulting in a three-frame latency between the time that the scene animation is sampled and the display of imagery containing the animation. This pipeline depth allows the GPU and CPU to execute computation and data transfer asynchronously and minimizes the amount of time that either processor is waiting for the other.

The TSAR software architecture consists of 33 components, including computational stages, data transfer, and data structures. Figure 3.13 contains a comprehensive diagram of the pipeline which is described in the remainder of this chapter. The organization is best understood either by following the path of a frame refresh through the pipeline stages from  $t = 0$ , or by considering the operations performed by each type of thread. Concurrent tasks may occur in matching horizontal regions between barriers.

### 3.4.4.1 Tile Selection

The tile selection stage which is performed by the host during the interoperation callback of the MantaGL pipeline is responsible for selecting framelet tiles for rendering such that the temporal sample density  $\theta_t$  of the system is achieved by the renderer. The algorithms used to select framelet tiles for rendering based on temporal sample density are described in Chapter 4.

Tile selection uses three data structures, the new framelet tiling containing the desired spatial and temporal sample rates across the image, a temporal history indicating the actual temporal refresh rate across the image, and a framelet batch table containing the framelets to be updated by the renderer at the current time. The new framelet tiling is copied from the device to the host during the

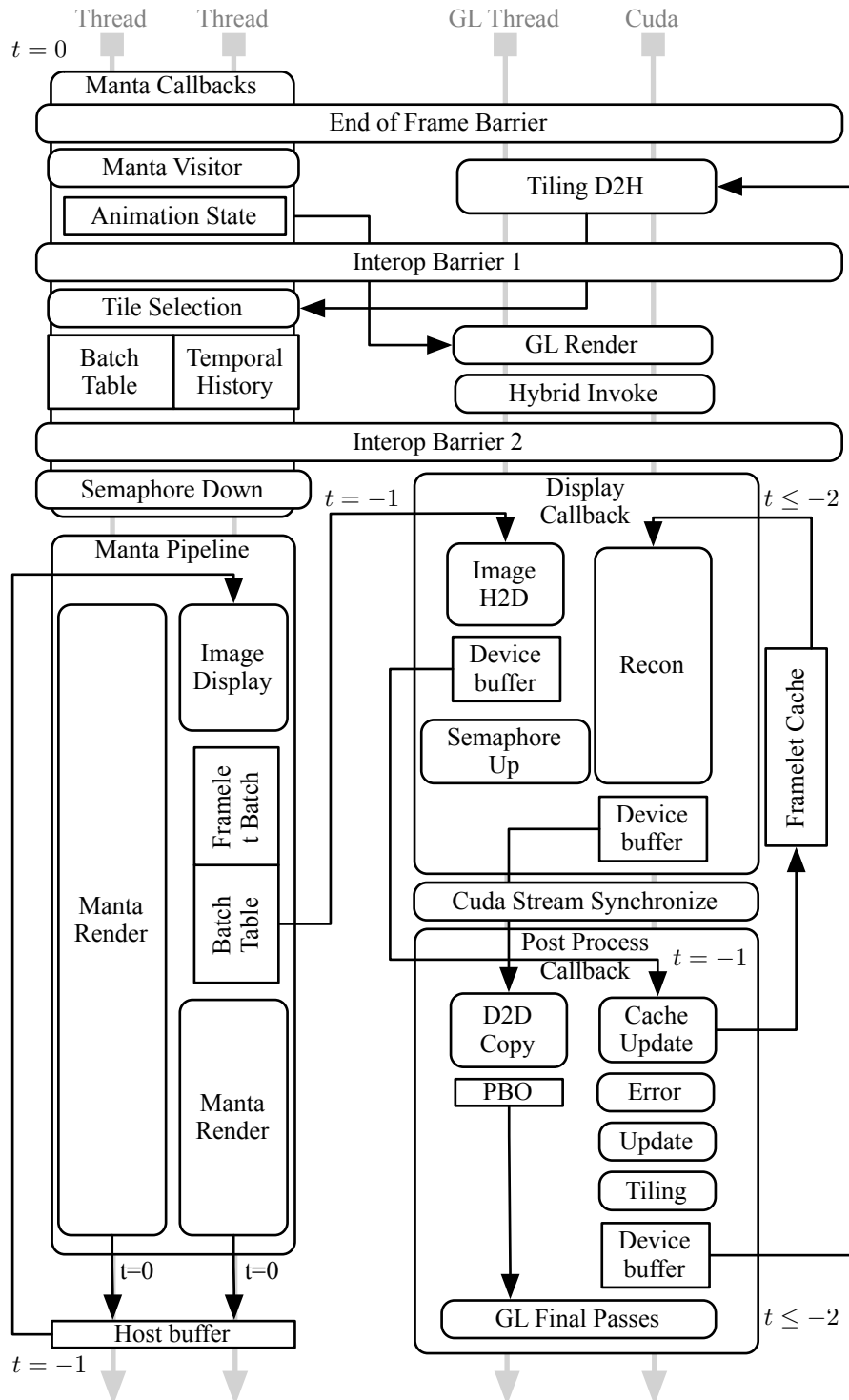


Figure 3.13. TSAR prototype parallel pipeline.



update which are stored in the batch table.

### 3.4.4.2 Rendering Stack

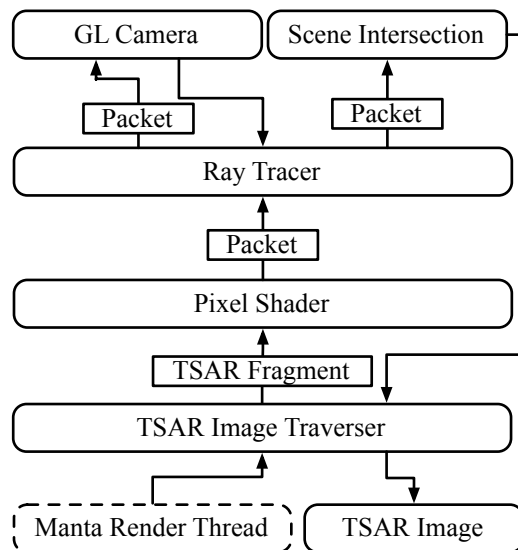
Framelet rendering is performed by Manta following the animation update and tile selection stages executed in the callback section of the pipeline. Several modular components of the Manta rendering stack, shown in Figure 3.14, are reimplemented to perform adaptive sampling and to organize the output of the renderer as a batch of framelets instead of a single contiguous image. Adaptive sample placement within the Manta pipeline is achieved by providing specialized implementations of the *image traverser* and *image buffer* modules, shown in Figure 3.14. Other components of the Manta rendering stack remain unchanged.

The image traverser is responsible for constructing Manta fragments containing pixel locations corresponding to framelet samples. Spatially adaptive sampling is performed by configuring the pixel sampler for single sampling with pixels placed on a high resolution integer tiling across a virtual image buffer. The resolution of the tiling matches the maximum precision necessary to place sample adaptively, i.e. instead of pixel centers  $\frac{1}{2N} + (0, \dots, N)$  in one dimension, a sub-pixel precision  $M$  is used such that pixel centers are located  $\frac{M}{2N} + (0, M, 2M, \dots, N)$ , and each pixel has  $M^2$  possible subpixel coordinates. Since the components of the Manta rendering stack downstream of the image traverser do not use pixel coordinates, the virtual coordinates may pass through the software stack to the image buffer where they are transformed into a sample position in a image buffer for the framelet batch. The use of a virtual high resolution coordinate system for pixel placement does not incur measurable overhead compared to the row major sequential pixel coordinates used in the standard pipeline.

The TSAR image buffer adds a framelet information table to the MantaGL image buffer. The table contains the ID of each framelet as well as the offset in the image buffer and the destination of the data in the framelet cache. Within the image buffer, each framelet is stored in a continuous memory with samples stored in Morton order. Both the image buffer and the framelet table are copied from the host to the device. The Manta fragment interface between the image traverser and pixel sampler conceals the framelet structure and pixel sample order from downstream components in the render stack.

### 3.4.4.3 Reconstruction

The reconstruction stage of the TSAR pipeline is responsible for producing an image at the spatial and temporal display resolutions of the rendering system using the framelets stored in the cache which contain samples at varying spatial and temporal resolutions. The algorithms used for



**Figure 3.14.** TSAR implementation of the Manta rendering stack. The TSAR Image Traverser creates Manta Fragments with pixel locations on a high resolution integer grid based on the framelets selected for rendering. The virtual coordinates assigned by the Image Traverser and converted back into pixel locations by the TSAR Image buffer. The remainder of the pipeline components from Figure 3.11 are unmodified.

reconstruction, which principally involve the elimination through smoothing of temporal artifacts while retaining sharp spatial features, are described in Chapter 4.2.

Reconstruction occurs in the last stage of the TSAR pipeline and is executed in two steps: first, a device compute mode procedure extracts framelet information from the framelet cache and sample density fields; this information is copied into a texture which is used by OpenGL to perform spatial and temporal filtering in a series of final passes. Before reconstruction begins in device compute mode, the framelet cache contains information that is at least two pipeline cycles old, i.e.  $t \leq 2$ ; concurrently with reconstruction, the newly rendered framelets from  $t = -1$  are copied from the host to the device.

#### 3.4.4.4 Framelet Cache Update

Framelet cache update is the first step in the adaptive response stage of the TSAR pipeline and is responsible for copying the incoming batch of framelets into the cache buffer. The cache replacement policy removes framelets from cache by comparing the age of each framelet to the spatial age threshold within the extent of the framelet computed during reconstruction.

#### 3.4.4.5 Error Estimation

Sample density error estimation determines the direction and magnitude of an adaptive response to improve the sampling efficiency and fidelity level of the renderer based on statistical characteristics of the new batch of framelets. Each error estimation routine, described in Chapter 5, uses a set of persistent data structures to characterize temporal behavior of the image. The input to the error estimation step is the new batch of framelet samples and the output is a pair of spatial and temporal sample rate error estimates.

The error estimation step only operates on regions of the error function within the extent of the incoming batch of framelets. As a result, at any single instance in time, much of the error estimate fields report zero error because no information about the adequacy of the sample rate in regions outside of the framelet batch is available.

#### 3.4.4.6 Density Solver

The density solver step in the adaptive response stage of the pipeline uses the error estimate computed in the previous step to modulate the spatial and temporal sample density fields. The input to the density solver, described in Chapter 6, is the current spatial and temporal sample density fields, and the error estimate fields computed in the previous step. The newest image signal information available to the density solver is estimated error from framelet samples rendered at time  $t = -1$ . The density solver outputs the updated spatial and temporal sample density fields which are used to update the framelet tiling.

#### 3.4.4.7 Tile Update

Framelet tiling update is the last step in the adaptive response stage. The framelet tiling is a piecewise constant approximation of the spatial sample density field, with a temporal rate assigned to each tile by quantization of the temporal sample density. Instead of updating the existing framelet tiling through a procedure of splitting and merging tiles, the approach taken by AFR[14], the tile update procedure constructs a completely new tiling using a parallel algorithm described in Section 4.1.6. The input to the tile update step is the new spatial and temporal sample density fields and the output is the new framelet tiling which is copied from the device to the host during the next pre-interoperation callback.

## CHAPTER 4

### ADAPTIVE SAMPLE PLACEMENT

Chapter 3, which described the TSAR algorithm, presented the model and underlying modular framework; the following chapters describe the design space of the system prototype and different implementation choices. Results of the prototype implementation are presented in Chapter 7. This chapter introduces the core algorithms and data structures necessary for temporally and spatially adaptive rendering which is the first stage in the TSAR algorithm. The consequences of different representations for the framelet tiling are presented as well as algorithms for constructing and resampling tilings.

#### 4.1 Spatial Structures

Row-major image pixel layout is ubiquitous in computer graphics, but it is an ill-suited representation for operations on neighborhoods of pixels and lacks any intrinsic support for hierarchical manipulation; these operations which are an intrinsic component of the TSAR algorithm require a different spatial structure for image representation. The adaptive rendering approach described in this dissertation produces imagery by rendering samples at varying spatial and temporal resolutions based on characteristics of the underlying image signal. This requires a suitable representation for multiresolution data and a set of operations for efficient parallel analysis, manipulation, and output.

Space filling curves are functions that provide a mapping from a two-dimensional or greater domain, to an index or offset in a one-dimensional domain. The curve function must be invertible, one-to-one, and onto. The row major pixel layout is an example of a very common space filling curve that biases mapped locality of the first dimension, i.e. the row, over the locality of data along each column. The Morton curve [46] and related tree structures described in this section avoid biasing either of the Cartesian dimensions and intrinsically provide a hierarchy supporting multiresolution operations. The efficiency of these operations, in terms of computational and memory complexity, contribute significantly to the realization of an interactive real-time implementation of the temporally and spatially adaptive rendering algorithm.

### 4.1.1 Spatial Tiling

The terms *tile*, *pixel*, and *sample*, are used interchangeably in this section to indicate a small region of a two-dimensional domain. An image produced by the rendering system is then a collection of tiles at a certain resolution, or a set of resolutions in a multiresolution case. Operations may be performed directly on this tiling, or the tiling may be resampled into a uniform grid which is suitable for display on a conventional monitor.

### 4.1.2 Morton Curve

The space filling curve generally attributed to G.M. Morton [46], also known under the names Z-curve and N-curve, is obtained by uninterleaving the bits of a one-dimensional index into coordinates in two or more dimensions. The curve provides a balanced locality trade off between row and column major ordering [63].

Cartesian coordinates are converted to one-dimensional offsets along the space filling curve by interleaving the bits of each coordinate. As illustrated in Figure 4.1, which shows the first 16 tiles along a Morton curve in two dimensions, the tile at Cartesian position (3, 2) is obtained by interleaving two-bit sequences  $3 = 11$  and  $2 = 10$  to obtain a four-bit sequence  $1101 = 13$ . The Cartesian position is the thirteenth coordinate along the curve. This procedure may be accelerated by precomputing look tables following the approach of Wise et al. [73]. The more common operation is to obtain Cartesian coordinates of a tile along a Morton curve by reversing the interleaving of the tile index, e.g. to convert uniform tiles along a Morton curve to row-major order for display.

### 4.1.3 Morton Order Tree

The integer order of indices in the Morton curve may be easily adapted for representation of binary trees by adding an offset based on the depth of the tiles at a certain level in a complete binary tree. For example, the tiling in Figure 4.1 contains 16 tiles corresponding to depth 4 in a complete binary tree; therefore, the curve indices are offset by  $2^4 = 16$  to obtain tiles 16 . . . 31 of a Morton order kd-tree. Figure 4.2 contains tilings in the first six levels of a Morton order kd-tree; Figure 4.2(e) corresponds to the tiles shown in Figure 4.2.

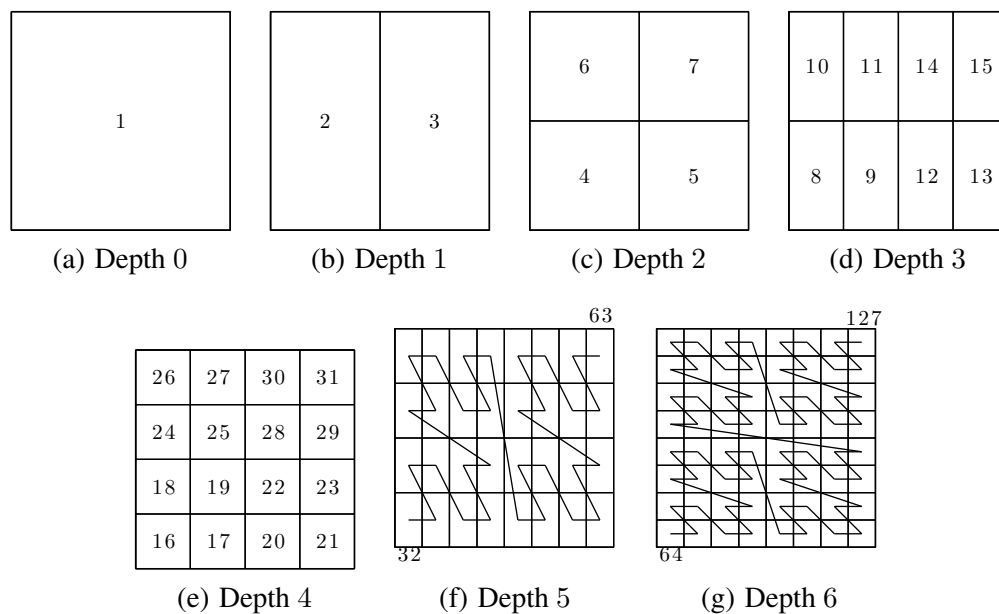
#### 4.1.3.1 Tree Traversal

Traversal through a Morton order kd-tree is accomplished by shifting a tile index right or left.

$$\mathcal{M}_k = k^{\text{th}} \text{tile in Morton order} \quad (4.1)$$

11	10	11	14	15
10	8	9	12	13
01	2	3	6	7
00	0	1	4	5
	00	01	10	11

**Figure 4.1.** Coordinates of the first 16 tiles in a two-dimensional Morton curve. Note that in a Morton order quad-tree the first tile in the lower left corner would be labeled 16.



**Figure 4.2.** Tiles in a Morton order kd-tree.

$$\text{Parent}_2(\mathcal{M}_k) = \mathcal{M}_{k/2} \quad (4.2)$$

$$\text{Children}_2(\mathcal{M}_k) = \{\mathcal{M}_{2k}, \mathcal{M}_{2k+1}\} \quad (4.3)$$

### 4.1.3.2 Quad-tree

Quad-tree may be represented by a Morton order tree by using only square tiles and skipping each tiling at odd depth. The traversal algorithm must be modified to skip odd depths by shifting indices two bits between generations instead of one bit.

$$\text{Parent}_4(\mathcal{M}_k) = \mathcal{M}_{k/4} \quad (4.4)$$

$$\text{Children}_4(\mathcal{M}_k) = \{\mathcal{M}_{4k}, \dots, \mathcal{M}_{4k+3}\} \quad (4.5)$$

The advantage of the quad-tree is that the shape of each tile is the same, regardless of depth in the tree. Applied to framelet-based adaptive sampling, the square tiles provide an isotropic sample density at the cost of one unused bit in the tile indices representing the tree.

### 4.1.3.3 Interval Operations

$\text{MinChild}_d(\mathcal{M}_k)$  returns the smallest child of  $\mathcal{M}_k$  and depth  $d$ , i.e. the first child encountered in a breadth first traversal of the complete tree. If  $\mathcal{M}_k$  is more shallow than  $d$ , then the ancestor is returned.

$$\text{MinChild}_d(\mathcal{M}_k) = \begin{cases} \mathcal{M}_{k \cdot 2^{(d - \log_2 k)}} & \text{if } \log_2 k < d \\ \mathcal{M}_{k \cdot 2^{-(\log_2 k - d)}} & \text{otherwise} \end{cases} \quad (4.6)$$

For example,  $\text{MinChild}_4(\mathcal{M}_5)$  is the first tile at depth 4 with ancestor  $\mathcal{M}_5$ . Since  $\log_2 5 = 2 < 4$ , the first case is used:  $\mathcal{M}_{5 \cdot 2^{(4 - \log_2 5)}} = \mathcal{M}_{20}$ .

The breadth of the subtree of  $\mathcal{M}_k$  at depth  $d$  is given by  $\text{Overlap}_d(\mathcal{M}_k)$ . This function provides the extent of a coarse tile in terms of fine tiles at a given depth; combined with  $\text{MinChild}$ , it is used to resample Morton order tilings at different uniform resolutions.

$$\text{Overlap}_d(\mathcal{M}_k) = \text{MinChild}_d(\mathcal{M}_{k+1}) - \text{MinChild}_d(\mathcal{M}_k) \quad (4.7)$$

Parallelization of operations upon uniform Morton order tilings may be accomplished by partitioning the tiling into power of two sized square intervals. The square partitions of a fine tiling form a coarse tiling.

#### 4.1.3.4 Resampling Between Uniform Resolutions

Consider resampling the uniform tiling in the Morton quad-tree at depth 6 in Figure 4.2(e) containing 16 tiles, to the tiling in Figure 4.2(c) containing 4 tiles. The finer resolution tiling contains tile indices 16 . . . 31, and the lower resolution output contains tiles 4 . . . 7. Each tile index is shifted two bits to the right to obtain the destination tile in the lower resolution tiling. The intervals of the Morton order curve in the finer resolution tiling corresponding to the same tile in the lower resolution tiling are continuous, e.g.  $\mathcal{M}_4^*$  in the output is produced from tiles 16, . . . , 19 in the input.

$$\mathcal{M}_k, k \in \left\{ \underbrace{16, 17, 18, 19, \dots}_{\mathcal{M}_4^*}, \underbrace{28, 29, 30, 31}_{\mathcal{M}_7^*} \right\} \quad (4.8)$$

If the resample operation is implemented using a parallel processor, it is very efficient to divide the fine resolution input tiles between processing elements based on the coarser resolution output tile, since it overlaps a continuous region along the linear curve.

#### 4.1.4 Depth First Tree

The breadth first Morton order quad-tree is sufficient for operations on uniform resolution fields represented by uniform tilings; operations such as resampling of multiresolution tilings are much less efficient. Consider resampling the simple tiling in Figure 4.3(a) to the uniform tiling  $\mathcal{M}_k^*, k \in \{4, 5, 6, 7\}$  in Figure 4.2(c).

$$\mathcal{M}_k, k \in \left\{ 4, 7, \underbrace{20, 21, 23, 24, 25, 26, 27}_{\mathcal{M}_5^*}, \underbrace{88, 89, 90, 91}_{\mathcal{M}_5^*} \right\} \quad (4.9)$$

The procedure is more complicated than in the uniform resampling case because coarse tiles in  $\mathcal{M}^*$  overlap disjoint intervals within tiling  $\mathcal{M}_k$ , e.g. tiles 20, 21, 23 near the beginning of the input contribute to  $\mathcal{M}_5^*$ , but so do tiles 88, 89, 90 and 91 which occur at the end. The resampling



procedure must locate these intervals and update values of the coarse tiles safely and efficiently. Parallel resampling might result in a data hazard if two of the overlapping intervals are operated upon by different processing elements at the same time. The potential hazard may be avoided by coalescing all of the resampling work for a coarse tile overlapping disjoint intervals, e.g.  $\mathcal{M}_5^*$ , into a continuous interval within a different space filling curve. Coalescing may be accomplished by reordering the tiles such that the complete subtree of  $\mathcal{M}_5^*$  is contiguous in the new space filling curve. Parallel resampling may be safely performed by dividing the continuous intervals of the new curve between processing elements. This new curve is a depth first ordering of the quad-tree.

The maximum depth of the complete tree must be specified to determine tile order since all possible tiles in the first subtree, based on its maximum depth, must be ordered before the first tile in the second subtree. Let

$$\mathcal{D}_k = k^{\text{th}} \text{tile in depth first order} \quad (4.10)$$

$$\mathcal{D}(\mathcal{M}_k) = \text{tile in depth order corresponding to } \mathcal{M}_k \quad (4.11)$$

The reordering operation finds  $\mathcal{D}(\mathcal{M}_k)$  for each tile in the Morton tiling, then sorts the tiles by  $\mathcal{D}_k$  such that each subtree of  $\mathcal{M}^*$  is a continuous interval along the curve. Consider  $\mathcal{D}(\mathcal{M}_k)$  for a complete binary tree; the number of tiles visited in depth first order before  $\mathcal{M}_k$  is equal to all tiles visited up to and including the parent  $\mathcal{M}_{k/2}$ , plus the size of a sibling subtree  $\mathcal{M}_{k-1}$ , plus one if  $k$  is odd. The size of the complete binary subtree  $\mathcal{M}_k$  is

$$\|\mathcal{M}_k\| = \sum_{i=0 \dots (m-\log_2 k)} 2^i = 2^{(m-\log_2 k)+1} - 1 \quad (4.12)$$

where  $m$  is the maximum depth of the tiling.

$\mathcal{D}(\mathcal{M}_k)$  may be written as a recurrence:

$$\begin{aligned} \mathcal{D}(\mathcal{M}_k) &= 1 + \mathcal{D}(\mathcal{M}_{k/2}) + \begin{cases} \|\mathcal{M}_{k-1}\| & k \text{ is odd} \\ 0 & \text{otherwise} \end{cases} \\ \mathcal{D}(\mathcal{M}_1) &= 1 \end{aligned} \quad (4.13)$$

Like tile indices of the Morton order trees, indices in the depth first order trees are assigned based on a complete binary kd-tree, of which half of the indices are unused in a complete quad-tree. Since an entire subtree must be counted before proceeding to neighboring tiles, siblings at shallow depths fall very far apart in the curve.

#### 4.1.4.1 Resampling Between Tilings

Reordering the tile indices in Figure 4.3(a) obtains the indices shown in Figure 4.3(b). The overlapping tiles for resampling are now coalesced into continuous intervals along the curve.

$$\mathcal{D}_k, k \in \left\{ \underbrace{3, 36, 43, 53, 54, 56, 57, 58}_{\mathcal{M}_5^*}, \underbrace{68, 75, 83, 90, 97}_{\mathcal{M}_6^*} \right\} \quad (4.14)$$

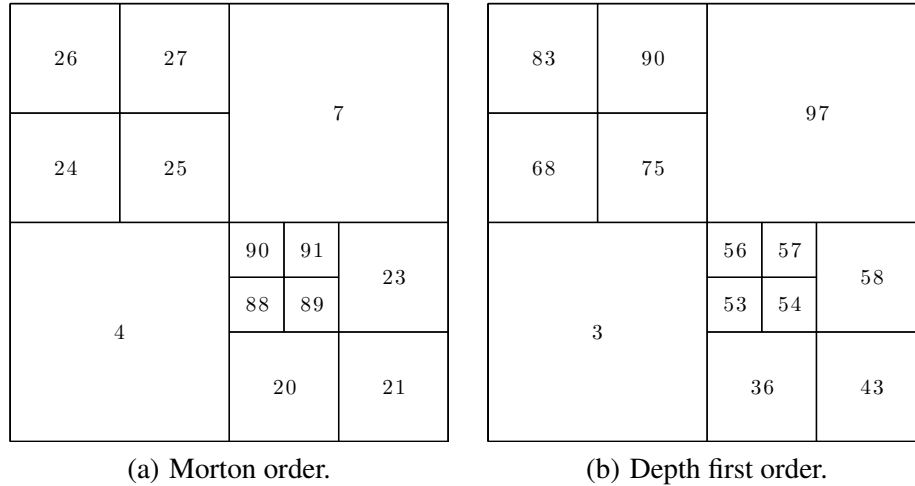
The multiresolution tiling does not necessarily consist of complete subtrees; in most cases the tiling is formed by a nonoverlapping cut through the tree. As a result, the functions MinChild and Overlap may not be written in closed form; they require a search over the tiling in curve order. Fortunately in the case of resampling, the search may be performed for continuous intervals between specific values.

$$\text{Overlap}(\mathcal{M}_k) = \{\mathcal{D}(\mathcal{M}_k), \dots, \mathcal{D}(\mathcal{M}_{k+1})\} \quad (4.15)$$

For example,  $\text{Overlap}(\mathcal{M}_5^*) = \{\mathcal{D}(\mathcal{M}_5^*), \dots, \mathcal{D}(\mathcal{M}_6^*)\} = \{\mathcal{D}_{34}, \dots, \mathcal{D}_{66}\}$ ; although these tiles do not appear in the depth first order tiling in 4.3(b), their indices bound the possible overlapping interval of  $\mathcal{M}_5^*$ .

#### 4.1.5 Summed Area Table

The integral over a coarse tile can be computed by finding the interval of fine tiles at the scalar field resolution contained within it. For example, in a complete Morton order quad-tree, coarse



**Figure 4.3.** Comparison between Morton tiling and depth first tiling.

tile 4 contains fine tiles in the 1D interval  $[64, 79]$ , inclusive, at depth four. The interval can be found quickly by bit shifting the tile id from its depth to the depth of the scalar field. Since the quad-tree curve advances by four times per level, moving from depth two to depth four requires shifting four bits left:  $4 \ll 4 = 64$  and  $(4 + 1) \ll 4 = 80$ , so the interval for tile 4 and depth 4 is  $[64, 80) = [64, 79]$ .

The integral over 1D intervals in the Morton curve may be evaluated quickly by computing a prefix sum along the curve over the scalar field. The integral over tiles  $[64, 79]$  is evaluated by translating the interval into scalar field coordinates. Index 64 is the first tile at depth 3, so the translated interval is  $[0, 15]$ . The integral is computed by subtracting the prefix sum to the left of the translated interval from the prefix sum at its right extent.

#### 4.1.6 Tiling Construction

The construction of multiresolution tilings is one of the two most common operations performed on Morton order trees in this dissertation. The predominant use of a multiresolution tiling is to approximate a discretized scalar field with a nonoverlapping set of piecewise constant tiles. The tiles vary in size such that the integral of the scalar field over each tile is approximately the same.

##### 4.1.6.1 Serial Construction

Construction of the Morton order quad-tree may be performed serially using a tile stack and an output queue. First the tile stack is initialized with the root tile  $\mathcal{M}_1$ , then while the stack is not empty, tile  $\mathcal{M}_k$  is removed from the stack. If the integral over the discretized scalar field within the tile is less than the tile threshold, or if  $\log_2 k$  is greater than the maximum depth,  $\mathcal{M}_k$  is added to an

output queue, otherwise  $\text{Children}_4(\mathcal{M}_k)$  is added to the stack.

#### 4.1.6.2 Data Parallel Construction

Simple parallelization of the serial algorithm by adding children nodes to a work queue tended by multiple threads suffers from exponential growth of the stack, since during refinement of the tiling, each thread adds four tiles to the stack for each one removed. This quickly causes the stack to fill which prevents the algorithm from making forward progress.

Instead of storing tile ids on the stack, the algorithm may be reformulated so that each thread operates within an interval on a Morton curve over the quad-tree. The total number of intervals does not need to grow exponentially and may be bounded by the amount of memory or number of parallel processors available. If additional processing elements or stack space become available, an interval may be split into pieces. In the most common case when the system is fully subscribed, each processor advances through a single interval on the curve.

Each interval is represented as a one-sided inclusive range in the Morton order quad-tree, e.g. tile 4 is represented as  $[4\ 5)$ . During each step, the processor may push or pop an interval on the stack, or split an interval. Pushing constitutes subdividing the tile  $[4\ 5) \rightarrow [16\ 20)$ . Popping is exactly the opposite,  $[18\ 20) \rightarrow [4\ 5)$ . Splitting an interval is performed if additional threads or storage are available for processing. The thread creates three new tiles for the first three quadrants of current tile, then advances the current token to the third child. For example splitting  $[16\ 20)$  yields  $[64\ 65)$ ,  $[65\ 66)$ ,  $[66\ 67)$ ,  $[67\ 80)$ . The interval cannot necessarily be divided into equal pieces because the integral of sibling tiles 17, 18, and 19, has yet to be evaluated.

The algorithm proceeds as follows: first, a token array with a set of coarse tile intervals covering the domain is initialized on the stack. Each processor loads an interval and computes the integral over it using a prefix sum. If the integral is less than some threshold or if the tile index is greater than another threshold, the index is added to the output queue. Otherwise, if the tile is a third child, i.e. the last sibling of a common parent, a new tile is popped from the stack. In the case that the tile is not the third child, it is split if there is enough room on the stack, or simply put back on the stack if additional storage is not available. If the output queue is filled, it may be flushed before the algorithm continues with the current contents of the tile stack.

Although this algorithm solves the exponential stack growth problem, it does not immediately address the exponential growth of parallel work, and does not attempt to load balance between threads. Consider a simple initialization case where intervals for tiles 4, 5, 6, 7 are written to the input array. The first four processors will pick up 4, 5, 6, and 7, and quickly fill the array with subsequent children; however, the remaining processors will exit immediately. If the output queue is sufficiently large, these four processors would produce the entire tiling in a very inefficient manner.

This type of load imbalance can be avoided by either initializing the algorithm with a greater number of intervals, or interrupting the kernel after a sufficient number of intervals are split, and restarting it. After the restart, a greater number of processors will pick up intervals from the input array. In this model, load imbalance caused by early processor termination when an interval is exhausted may be addressed by stream compacting the input interval array after each iteration of the algorithm.

The algorithm is invoked exactly the number of times necessary to output a tiling containing the maximum number of tiles (which should be almost constant in TSAR). Later invocations of the kernel simply do not perform any work: the threads start, read null tokens from the input array, then immediately exit.

## 4.2 Framelet Reconstruction

Image reconstruction is the final stage in the TSAR pipeline in which the rendered samples contained in the framelet cache are combined to create a uniform resolution image for display to the user. Framelets in the cache are rendered at varying resolution and produced at different times. The operation is performed in two steps; the first consists of copying information from the framelet table into memory accessible by the graphics device, and the second step performs several rasterization passes to shape filters before cached framelets are resampled to display resolution. Pipeling the two steps allows the serial memory copy step to occur in parallel with other computation as indicated in Figure 3.13. The reconstruction stage performs both spatial and temporal filtering; framelets are resampled spatially to output pixel resolution, and framelets that overlap spatially but are rendered at different times are resampled to the leading edge of time.

### 4.2.1 Reconstruction Artifacts

Reconstruction artifacts arise in regions with discontinuities in the age of framelets as well as regions where the spatial sampling rate changes and does not adequately capture fine detail in the scene. Both types of artifacts are addressed by increasing the amount of smoothing in problematic transition regions.

Inadequate or incorrect adaptive response is the principal cause of artifacts during reconstruction. While oversampling of the underlying image signal in the spatial domain usually only results in inefficiency, in the case that the total amount of computational capacity of the rendering system is limited, the inefficiency may redundantly sample one region while starving another region, which may result in spatial or temporal undersampling. Undersampling in the temporal domain, i.e. delayed sampling of the underlying image signal, is a more difficult artifact to overcome because the cache might not contain information about moving features at any level of spatial detail. Both

spatial and temporal undersampling may result in addition during reconstruction of high frequency detail or edge artifacts to the rendered image. In the spatial domain, these artifacts appear as rough high contrast jagged edges when the discrete input samples lack the resolution to determine where the high frequency detail is located. The analogous artifact in the temporal domain is tearing, where one region of the image contains samples older than a neighboring region. If the image signal contains motion, undersampling in the temporal domain may cause the object in motion to appear in more than one place across the image at a time.

The ambiguity introduced by either spatial or temporal undersampling, i.e. as to where in the image, or from samples rendered at which time, a high frequency detail should be reproduced, is addressed by smoothing or blurring the detail. This results in motion blur in the temporal domain. In fast changing regions of the image, or in regions with improper temporal oversampling, excessive motion blur, or smoothing in the temporal domain, may result in the loss of spatial detail as well.

The principal challenge faced by the reconstruction stage of the TSAR algorithm is different than the resampling problem addressed in many computer graphics applications such as image resizing or video resampling. In these applications, image pixels are produced by a low pass filtering process over sample data collected at a spatial sample rate, e.g. a physical scene exposed to a camera with optics that integrate light over each element in the sensor, or a photorealistic rendering engine in which a large number of secondary rays sample the illumination of a hemisphere. In these instances, low pass filtering of high resolution samples eliminates errant high frequency or noise and produces smoothly varying image features, in many cases with significant redundancy. In the adaptive rendering case, the input is the result of a sampling process which attempts to eliminate as much redundant detail as possible. The reconstruction stage often upsamples rendered samples to output pixel resolution or resamples to output resolution with little additional information. In this case, information about redundant detail in the discrete samples is not available during reconstruction.

#### **4.2.2 Algorithm**

The reconstruction stage of the TSAR pipeline uses a spatial and temporal weighting scheme to combine the contents of the framelet cache and produce the output image for display. The algorithm uses the graphics processor to determine regions of the cache likely to introduce high frequency temporal edges and then estimates temporal weights to smooth these discontinuities. This is performed in four steps, shown in Figure 4.4. First, the age of the most recent framelet in each pixel is determined by rasterizing the framelet cache and using framelet age as depth. The resulting depth buffer is referred to as an age buffer. Next, discontinuities between the age of framelets within the age buffer are smoothed. In the third step, the framelet cache is rasterized against the smoothed age values in the age buffer; samples newer than the value in the buffer pass the depth test and are

accumulated to the color buffer along with a temporal weight based on the difference between the framelet age and the value in the buffer. In the last step of the reconstruction algorithm, the color buffer is normalized and resulting image is displayed.

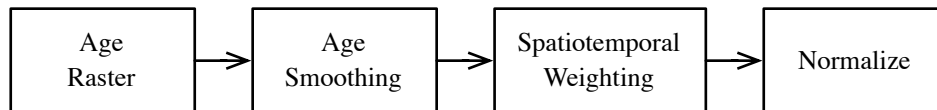
The four steps of the reconstruction algorithm are predominately concerned with shaping temporal filters such that tearing artifacts are reduced without sacrificing excessive spatial fidelity due to the introduction of motion blur. This is especially important in fast changing regions with higher temporal sample rates where a small temporal filter width may combine many framelets. Spatial filtering is required to avoid the introduction of spatial high frequencies, especially in regions where the resolution of available framelets changes; however, the set of framelets which may contribute to the reconstruction of the image in a given region depends of the temporal characteristics of the image signal and the contents of the framelet cache. In regions with significant temporal change, and therefore high temporal refresh rates, although the cache may contain a large number of overlapping framelets in the region, only the newest framelet is likely to be available due to the shape of temporal filter weights. In this case, since this output must reflect the newest information in the framelet cache, spatial filtering is limited by the constraints of temporal reconstruction process.

The temporally coherent collection of samples contained within each framelet largely simplifies the temporal reconstruction problem; within the extent of a cached framelet tile, signal reconstruction at the leading edge of time consists of spatial reconstruction at the temporal plane of the newest framelet; along the boundaries of the newest framelet where the age of framelets in the cache changes, reconstruction must smooth artifacts caused by temporal discontinuities.

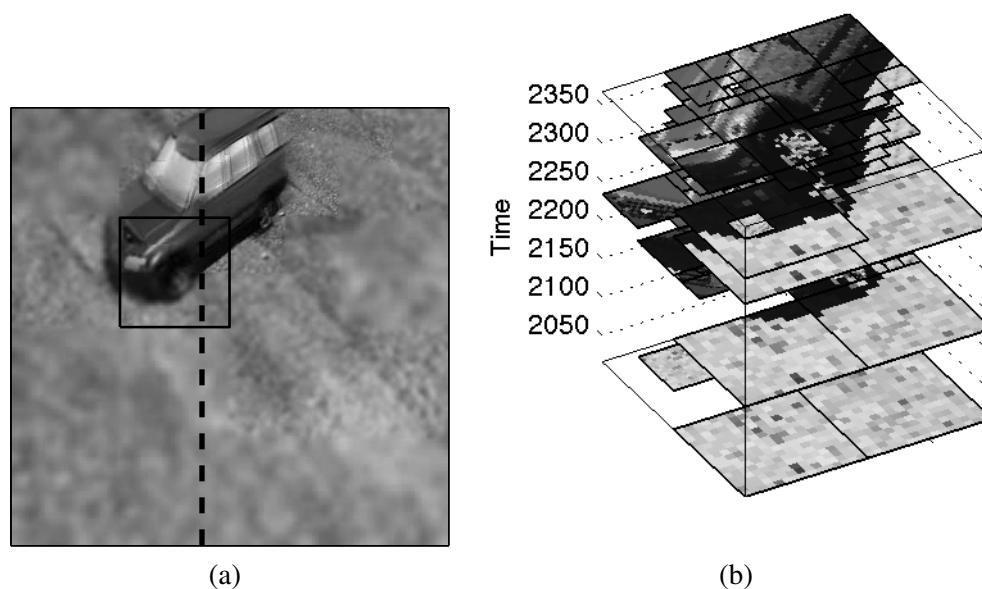
### 4.2.3 Temporal Filtering

Reconstruction of framelet samples in the temporal dimension consists of a downsampling operation which must bias the newest framelets available while smoothing transitions between newer and older framelets. Figure 4.5 contains a reconstructed frame as well as the subset of the framelet cache used to reconstruct a square region of the frame. The framelets within the square region indicated in Figure 4.5(a) are shown in Figure 4.5(b) with time running along the vertical axis. The dashed line in the figure indicates a single column of the image examined in greater detail in subsequent figures. The framelets inside this region of the image were rendered at several different temporal refresh rates, e.g. the framelets in the upper part of the region which the vehicle is moving through are refreshed frequently and are spaced close together along the vertical axis, while the framelets in the lower part of the region are spaced much further apart.

In regions with animation, the newest framelets contain the most accurate samples of the motion. If these framelets alone are used for reconstruction, in cases like Figure 4.5(b), where the newest framelets differ by several hundred milliseconds, the temporal difference between newest framelets



**Figure 4.4.** Rasterization passes during reconstruction.



**Figure 4.5.** Reconstruction and the framelet cache. The result of temporal smoothing based on an age heuristic is shown in (a), the framelets within the square region of the image are plotted in (b).

will result in a tearing artifact where the motion will appear to be incomplete or the moving object will appear to be in two places at once.

The large difference in framelet age shown in Figure 4.5(b) is due to an inadequate granularity of temporal adaptive response. Because the lower portion of the highlighted region contains low spatial frequencies and is dominated by static geometry, it is sampled at a lower spatial sample rate by a large framelet tile. The high frequency temporal change caused by the vehicle wheel and movement of the vehicle crosses the upper left corner of the low spatial frequency tile. In this case, the amount of temporal change was likely not sufficient to result in either a subdivision of the tile or an increase in the temporal refresh rate of the whole tile.

Reconstruction filters along the temporal dimension are applied at each pixel in the output image to resample framelets overlapping that pixel to a single color at the leading edge of time. Temporal filter shape is based on the framelet sample age field computed in the first step of the reconstruction stage shown in Figure 4.4. The filter shape is determined by the difference in the temporal dimension



between each framelet sample and the age function.

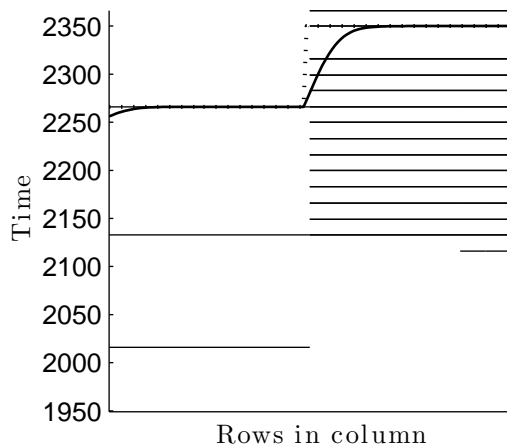
To shape the filter more broadly at discontinuities in the age of the newest framelets available, the age field is smoothed on the newer side of a temporal discontinuity between framelets. This process is illustrated in Figure 4.6 which shows the framelets overlapping the column highlighted in Figure 4.5(a) and Figure 4.7 in two dimensions; the vertical dimension of the figure indicates the spacing of framelets along the temporal axis. The age field, given by the time stamp of the newest framelet available in the cache for reconstruction, is indicated by the dashed line.

Framelet samples at the temporal value equal to or greater than the age field are given positive temporal weights, while samples less than, or older than, the age field are given a weight of zero. In the implementation used in the prototype, older samples are culled by a depth buffer operation and do not contribute to reconstruction. Smoothing around temporal edges in the age field is accomplished by decreasing the age field on the newer side of the discontinuity, as indicated by the solid curve in Figure 4.6.

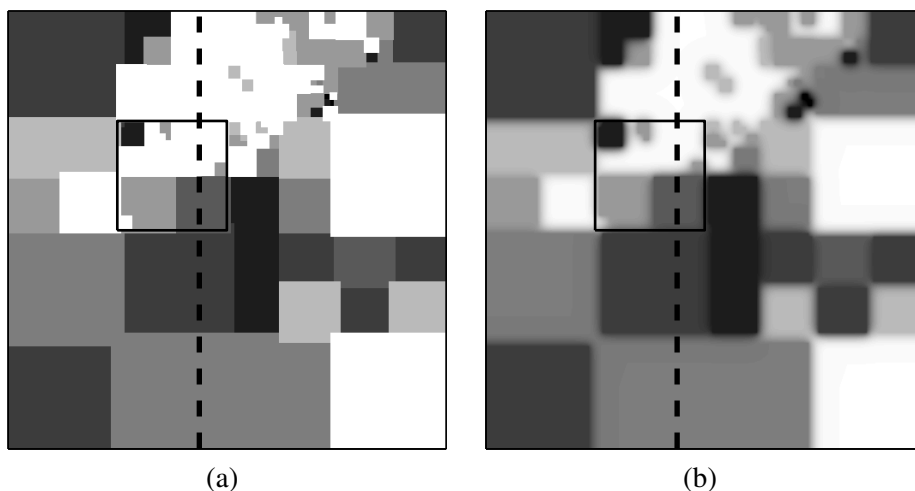
The age field is manipulated by applying a smoothing convolution which is constrained to only decrease the value of the field; therefore, on the edge shown in the figure, it only decreases the age value on the right side of the discontinuity, but does not increase the age value on the left side. This results in older framelet samples on the right, or newer, side of the temporal discontinuity contributing to the reconstructed image and eases the transition between image samples rendered at different times. The smoothing operation on the age field must be performed using a one-sided operator since the age function must always be less than or equal to the age of the most recent framelet.

The result of age smoothing for the frame shown in Figure 4.5(a) is shown in Figure 4.7. Regions in the figure with greater luminance are newer framelets. Overlap between framelets, e.g. between the two light gray framelets in the lower left corner, is due to the increased framelet splat size used to perform spatial reconstruction across framelet boundaries. The temporal age of a framelet effects the age buffer across the entire spatial extent of all samples in the framelet.

The depth buffer and depth test available on conventional graphics hardware may be used to produce the age field, and also to efficiently cull framelets older than the field. In the first pass of the reconstruction algorithm, the depth buffer is used to create an age field by rendering each framelet in the cache at its spatial position and at a depth proportional to the time stamp of the framelet relative to the oldest and newest framelets in the cache. The second pass of the reconstruction algorithm is performed by smoothing the age field using the depth buffer computed in the first step. Before the third and fourth passes, the depth test is reconfigured to only allow samples greater than the age field stored in the depth buffer. Samples rendered after the age smoothing pass are tested against



**Figure 4.6.** Temporal reconstruction in two dimensions. Framelets overlapping the column indicated in Figure 4.5(a) and Figure 4.7 are shown in profile.



**Figure 4.7.** Reconstruction age fields. The input age field based on the contents of the cache is shown in (a), and the result of smoothing sharp edges to smooth temporal artifacts is shown in (b).

the depth buffer, but do not modify it. To prevent framelet samples equal to the age field from being culled, the framelets are shifted in the temporal dimension by half the maximum temporal resolution supported by the rendering system.

#### 4.2.4 Spatial Filtering

Reconstruction in the temporal dimension always results in downsampling, i.e. the reduction of all of the overlapping samples in the image within a temporal window to a single output at the leading edge of time, given the amount of available redundancy in most workloads, and the distribution of sample densities, upsampling from low resolution framelets to output pixels at a

higher resolution is more common than spatial smoothing.

Spatial reconstruction is performed using a scattering approach where framelets are splatted to the output buffer as square quads with enlarged extent to cover the support of all samples within the framelet. The effect of the enlarged framelet extent may be observed in Figure 4.7 which shows the contents of the age buffer consisting of framelets rendered at different times. Framelets in the lower left region of the image appear to have overlapping corners due to the enlarged spatial extent of each framelet. The spatial location of each framelet sample remains the same, and the extent of the framelet quad is enlarged by half the spatial resolution of the framelet, i.e. half the distance between neighboring samples in the framelet. When composited to the output buffer, both color and combined spatiotemporal weight are accumulated. The spatial weight about each framelet sample is given by a bilinear filter centered at the spatial location in the output image of each framelet sample. During the final step of the reconstruction stage, the accumulated weight is used to normalize the accumulated color at each pixel in the output image.

The result of this reconstruction algorithm is shown in the examples given in Chapter 7. Since reconstruction is the last stage in the TSAR pipeline, it is principally concerned with compensating for artifacts or inadequacies introduced by earlier stages, e.g. Figure 7.24 which shows the effect of modulating response parameters to exacerbate rendering artifacts. Much image processing can be performed during reconstruction to correct adaptive response mistakes by smoothing temporal artifacts or sharp transitions between levels of spatial detail. Unlike the earlier stages of the TSAR pipeline, however, the cost of reconstruction scales with the number of output pixels in the final image, not the number of samples in the framelet cache, or indirectly, the spectral complexity of the underlying imagery. The reconstruction approach presented here provides a mechanism to display the contents of the framelet cache in a direct manner with a minimal amount of work performed to smooth spatiotemporal sampling artifacts. Temporally and spatially adaptive sampling is the primary focus of this dissertation, not reconstruction; the principal mechanism available in the TSAR pipeline to correct temporal artifacts in rendered imagery is the adaptive response process, not the smoothing and blurring performed during reconstruction.

## CHAPTER 5

### ERROR ESTIMATION

The sample rate error estimate across the spatiotemporal image is computed by the third stage of the adaptive rendering pipeline, shown in Figure 1.2, and used by the fourth stage to adapt the sampling rate. This chapter defines the ideal sample rate and sample rate error in the frequency domain. Two approaches are described to approximate the spectral formulation, the first in the spatiotemporal domain, and the second using wavelet statistics.

Error estimation and spectral characterization of the image signal described in this chapter is performed at discrete locations across the image domain: these estimates may be interpreted as locally greedy control responses to adapt the sample density. Chapter 6 describes how the individual responses are combined, and how the sample rates used for rendering are changed, without violating constraints and limitations on the system as a whole.

#### 5.1 Sample Rate Error

Sample rate error is defined as the signed amount which the spatial or temporal sample densities must change to perform both sufficient sampling of the underlying image signal, and to reduce the amount of redundant sampling work performed during rendering. The necessary spatial and temporal sample density across the image changes over time as user interaction or scene animation causes the underlying image signal to change. Sample rate error is a signed quantity. Regions where redundant sampling work is performed, or the image is oversampled, have negative error. Regions where the sample rate is low, or the image signal is undersampled, are defined as having positive error. The sign of the sample rate error estimate indicates the direction in which the current sample rate should change to obtain the ideal sample rate. The magnitude of the error estimate is proportional to the distance between the current rate and the ideal, or optimal, rate in both dimensions.

The ideal spatial and temporal sample rates, which vary across the image, are unknown due to the discrete sampling theorem, and the assumption that the underlying image signal for the graphics scene is not bandlimited. The error estimate  $\phi$  must be formed in relative terms by guessing the direction and distance from the current sample density to the ideal sample rate. Consider a case

where the spatial sample density is already close to the maximum rate the renderer is capable of sampling, and there is very little change in color between samples; since the graphics scene is not band limited, a very small, and very high frequency, image feature may still occur between neighboring samples. This feature is out of phase with the sample pattern and will not be discovered until the sample density is sufficiently increased, or the feature happens to enter the same phase as the sampling function.

### 5.1.1 Schematic Interpretation

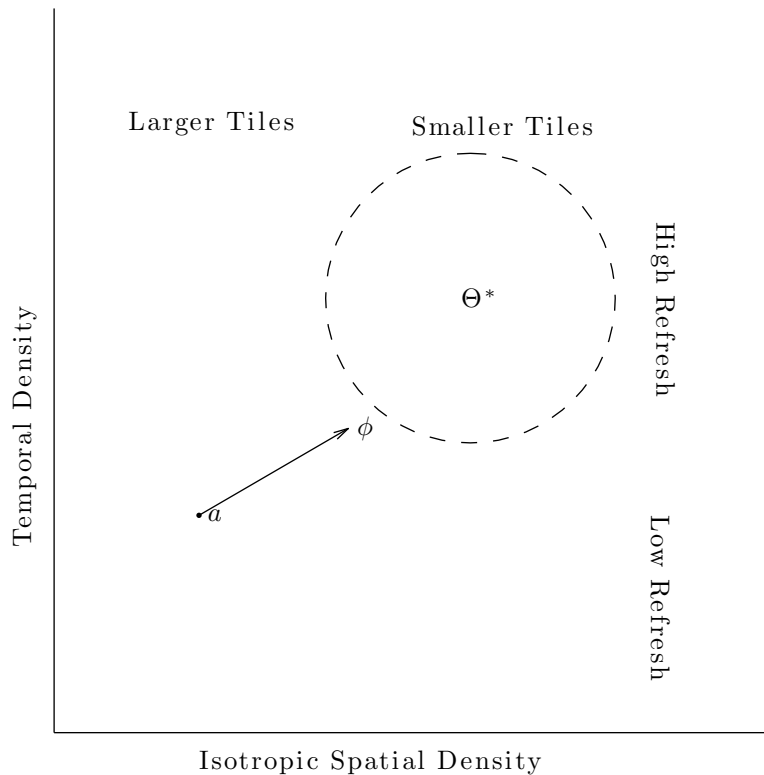
The spatiotemporal sample density error estimate  $\phi$  may be interpreted in a schematic figure using an illustration of a two-dimensional space, if the spatial sample resolution is assumed to be isotropic. The spatial sample density in both the horizontal and vertical dimensions of the image may be plotted along the horizontal axis of the figure, increasing from left to right. The temporal sample density may be plotted along the vertical axis, increasing from bottom to top. Higher overall sample rates, i.e. small framelet tiles with frequent refresh, occur in the upper right corner of the space, while lower sample densities occur in the lower left near the origin.

Figure 5.1 illustrates the error estimation problem at one element, or discrete location, in the image domain; point  $a$  is the current spatial and temporal density for the element, located at the coordinate  $a = \langle \theta_s^n, \theta_t^n \rangle$ , and unknown ideal sample density  $\Theta^*$  for the element is indicated as a set of possible values. In the sample density space, the error estimate is the vector  $\phi$  pointing from the current density  $a$  towards a point in the estimated set  $\Theta^*$  which contains the ideal sample rate. In this example, the set is plotted as a dotted circle to indicate that the exact value of  $\Theta^*$  is unknown. The shape of the estimated set of possible  $\Theta^*$  depends on the mechanism used by the control algorithm to characterize redundancy in the signal. The actual unknown ideal spatial and temporal sample density, which adequately samples the signal with minimal redundancy, is a single point in the space, not a set of points.

The magnitude of the spatial and temporal error, and therefore the precise direction, i.e. within a quadrant, are approximate given the degree of uncertainty as to the location of  $\Theta^*$ . Depending on the direction and magnitude of  $\phi$ , sample density may be add to or removed from the element; also, if the current density is close to a maximum or minimum, the error vector may need to be constrained before adaptive response is performed; these operations are performed by the adaptive response control decision solver, described in Chapter 6.

### 5.1.2 Ideal Sample Density

The accuracy of the estimated set of possible  $\Theta^*$ , and the error estimate vector  $\phi$  depend on the approximation used to characterize the redundancy, or efficiency, of the current sampling rate. In



**Figure 5.1.** Sample rate error estimation at one element in the image. Point  $a$  is located at the current spatial and temporal sample densities, the set  $\Theta^*$  indicates the unknown ideal spatiotemporal sample rate based on the underlying image signal. The error estimation procedure computes vector  $\phi$ , based on characteristics of rendered imagery.

Figure 5.1, the ideal spatial and temporal sample density in the spatiotemporal domain is indicated by a circular set  $\Theta^*$  to illustrate the uncertainty regarding its value. With only information about the signal obtained from the adaptive sampling process, the actual ideal sample density may not be determined exactly, and while the control algorithm implicitly estimates a set of possible ideal densities, the shape of the set is likely not a circle.

Within a region of the image, such as a framelet tile, the ideal sample density  $\Theta^*$  must strike a balance between using a higher sample rate to capture high frequency energy at sharp image features, and the amount of redundant sampling work performed in low frequency areas of the tile. Given the expected power law of natural imagery described in Section 1.3.4, lower frequency features are expected to dominate the image. After the sample rate is high enough to capture the energy of these features, the amount of additional energy captured by continuing to increase the sample rate asymptotically approaches zero, while the amount of redundant sampling work increases. The ideal sample density  $\Theta^*$  occurs at a point after a sufficient amount of low frequency energy has been captured by the sampling process.

### 5.1.2.1 Frequency Domain Interpretation

To illustrate this definition of  $\Theta^*$  in the frequency domain, consider the test scene shown in Figure 5.2 which has spectral characteristics that are representative of an entertainment graphics workload. The example consists of a half second animation sequence from the car scene from which the framelet tile number 1442, shown in black, and enlarged in the upper right corner, is sampled at a resolution of 1024 samples in each dimension. The sample rate corresponds to  $64^2$  samples per pixel at the  $512^2$  pixel resolution shown in the figure, and a refresh latency  $\Delta t < 1\text{ms}$ . This sample rate is significantly higher than the TSAR prototype is capable of either rendering or analyzing interactively. The maximum spatial rate of  $4^2$  samples per pixel, and 16ms refresh latency, is a typical configuration for the real-time system.

In the high resolution discrete case, the ideal sample density  $\Theta^*$  may be determined by identifying the spatial and temporal sample rate at which *most* of the energy, i.e. within a certain percent of the total energy, is captured. In the frequency domain, the ideal spatiotemporal domain sample rate  $\Theta^*$  is determined by finding the corresponding frequencies  $f_s$  and  $f_t$ , below which a certain percent of the total energy in the high resolution discrete signal is represented.

The norm operator  $\text{Norm}_{\text{ps}}(s, t)$ , subsequently defined, is a measure of the total energy in the three-dimensional power spectrum contained in the bands  $[1, s]$  in the spatial  $x$  and  $y$  directions, and  $[1, t]$  in the time direction. The operator performs a sum in the power spectrum over a square band of spatial frequencies, since the spatial sample density is assumed to be isotropic, and over a rectangular band of temporal frequencies.

Let  $\mathcal{F}[x, y, t]$  be the discrete Fourier transform of the high resolution example signal shown in Figure 5.2, the power spectrum  $\mathcal{F}^2[x, y, t]$  contains coefficients in eight octants corresponding



**Figure 5.2.**  $\Theta^*$  example scene. To illustrate the definition of  $\Theta^*$  the framelet highlighted is rendered at a spatial resolution of 1024 samples per pixel and 1024 frames per second.

to each combination positive and negative frequencies  $1, \dots, N/2$ , the power of these octants is combined by the function  $\bar{\mathcal{F}}^2 [x, y, t]$ :

$$\bar{\mathcal{F}}^2 [x, y, t] = \sum_{\substack{l \in \{-x, +x\} \\ m \in \{-y, +y\} \\ n \in \{-t, +t\}}} \mathcal{F}^2 [l, m, n] \quad (5.1)$$

Then the norm is computed by summing the total power in rectangular bands  $s \times s \times t$  in size:

$$\text{Norm}_{\text{ps}} (s, t) = \sum_{z=1 \dots t} \left( \sum_{x=1 \dots s} \sum_{y=1 \dots s} (\bar{\mathcal{F}}^2 [x, y, z]) \right) \quad (5.2)$$

The spatial power summation operation is illustrated in Figure 5.3(a) for the first slice along the temporal axis. The scalar field in the figure is a color mapped image of the log of the total power at each spatial frequency across all eight octants of the three-dimensional FFT, produced by equation 5.1. The spatial power norm consists of summing the total power in each square band along the diagonal, indicated by the dashed line, e.g.  $\text{Norm}_{\text{ps}} (f_s, 1)$ , which is the sum of all power in the square band outlined in the figure. Figure 5.3(b) contains a log/log plot of the spatial power norm taken across the diagonal of the slice. The spatial derivative in this plot decreases very quickly.

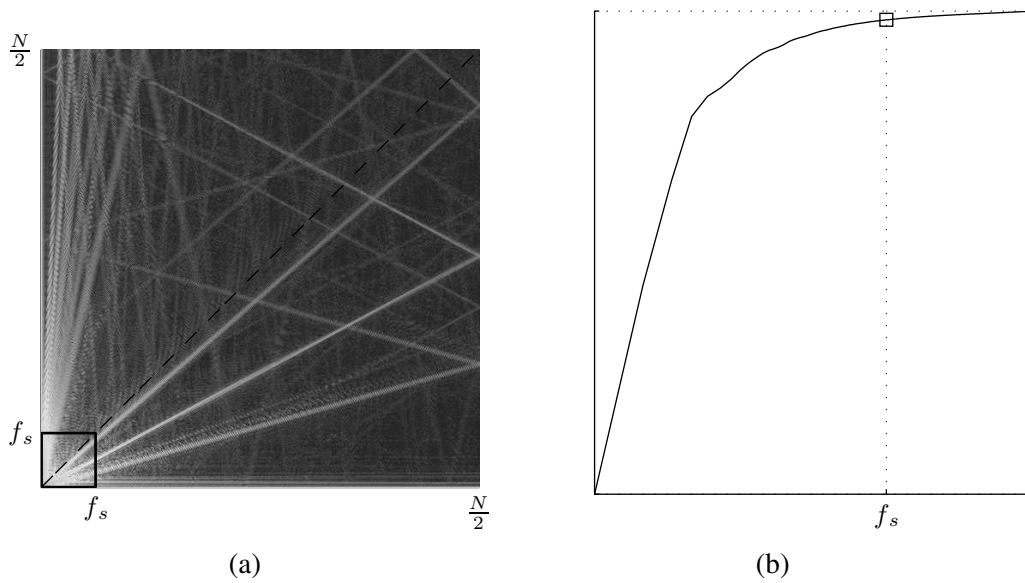
The frequencies  $f_s$  and  $f_t$  are the minimum at which the gradient magnitude of the norm, i.e.  $\left\| \frac{\partial}{\partial s}, \frac{\partial}{\partial t} \right\|$ , passes below the threshold  $\tau$ :

$$\min_{f_s, f_t} \left| \left\| \nabla (\text{Norm}_{\text{ps}} (f_s, f_t)) \right\| - \tau \right| \quad (5.3)$$

By the Nyquist sampling theorem, the minimum spatiotemporal sample rate necessary to obtain the energy in the power spectrum bounded by frequencies  $f_s$  and  $f_t$  is:

$$\Theta^* = \langle 2f_s, 2f_t \rangle \quad (5.4)$$

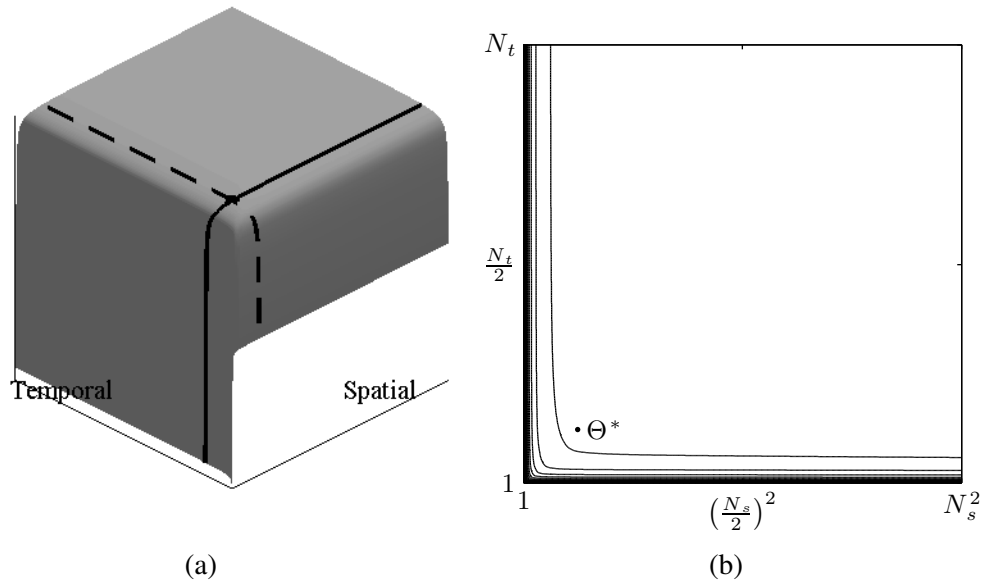




**Figure 5.3.**  $\Theta^*$  example power spectrum norm.

The shape of  $\text{Norm}_{\text{ps}}(s, t)$  on a log scale is shown in Figure 5.4(a); the point labeled  $\Theta^*$  in Figure 5.4(b) indicates the location of  $\langle f_s, f_t \rangle$ . The norm is shown in three dimensions in Figure 5.4(a) and as a contour plot in Figure 5.4(b). The surface climbs steeply in low spatial and temporal frequencies before reaching a plateau which extends through the remainder of the domain with a very small derivative. Figure 5.3(a) shows the spatial extent of the frequency band  $f_s$ ; while many features in  $\bar{\mathcal{F}}^2$  are visible outside of the square band indicated, based on the shape of the norm shown in Figure 5.4(b), the power in those features does not significantly contribute to the amount of energy already captured at band width  $f_s$ .

This example, with the output pixel resolution  $512^2$  shown in Figure 5.2,  $\Theta^*$  corresponds to a spatial sample rate of  $8^2$  samples per pixel with an 8ms refresh latency. The ideal sample rate falls outside the capability of the TSAR prototype, indicated by the the region  $\Omega_\theta$  in Figure 5.5(a), which contains an enlarged view of the lower frequencies in Figure 5.4(b). Realizable discrete rates within this region are described in Section 6.1.1. Since the ideal sample rate is well outside  $\Omega_\theta$ , the adaptive response control decision will eventually split framelet tile 1442 into smaller subregions within which the individual ideal sample rate may be realizable. The upper left subtile of 1442 shown in Figure 5.2 contains a very low frequency signal, which may indeed fall within  $\Omega_\theta$ , while the lower right subtile contains several sharp edges; the ideal sample density in this subregion will likely remain outside of  $\Omega_\theta$ .

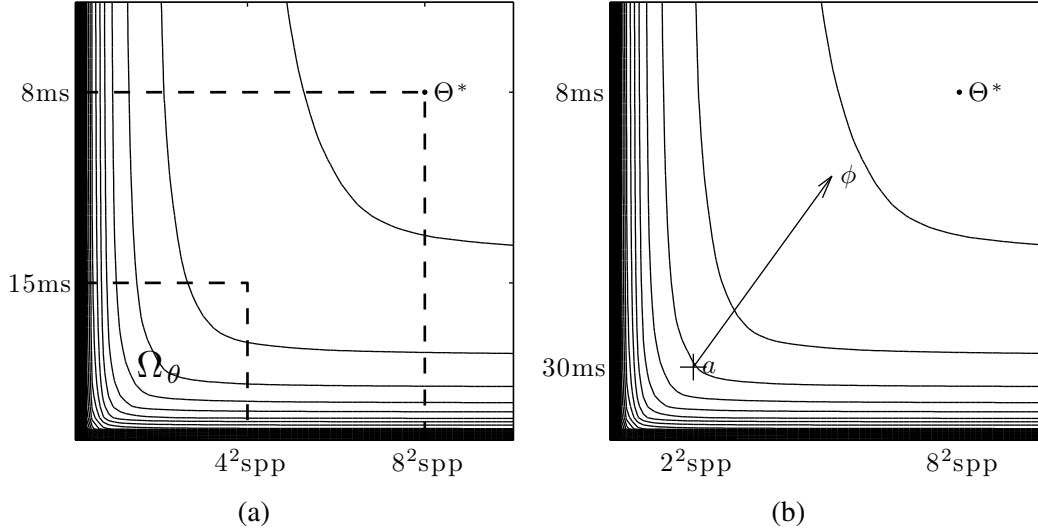


**Figure 5.4.**  $\Theta^*$  example surface.

### 5.1.2.2 Error Estimation

The objective of the sample rate error estimation stage is shown in terms of the power spectrum norm in Figure 5.5(b); the point  $a$  indicates the boundary of the frequency band corresponding to the spatiotemporal sample rate of rendering the example framelet, tile 1442, shown in Figure 5.2. For each framelet, the error estimation mechanism attempts to approximate the local power spectrum norm surface and the location of point  $a$  based on the current spatiotemporal sample rate. From this point, the adaptive response should follow the surface gradient to the ideal sample rate  $\Theta^*$ . The error response vector  $\phi$  is the subsequent direction and magnitude estimate following the gradient.

Real-time computational limitations prevent the error estimation mechanism from approximating the power spectrum norm surface as accurately as the analysis described in Section 5.1.2.1; instead, the surface gradient in the immediate vicinity of point  $a$  is approximated by a variety of statistical approaches in the spatiotemporal and frequency domains. These approaches approximate the power spectrum by projecting the signal across different sets of basis functions in the spatial and temporal domains. Section 5.2 describes the use of sample variance between luminance and the spatial and temporal directions, and Section 5.3.2 describes the use of a discrete wavelet transform to approximate the shape of the power spectrum norm.



**Figure 5.5.**  $\Theta^*$  example contour plot of the surface shown in Figure 5.4(a). The region  $\Omega_\theta$  indicates the sample resolutions realizable by the TSAR prototype.

## 5.2 Spatiotemporal Domain Approximations

The principal motivation for employing first order statistics in the spatiotemporal domain is that they are computationally inexpensive and produce a concise, if inaccurate, characterization of the image signal. Higher order spectral statistics may more accurately approximate the power spectrum analysis described in Section 5.1.2.1, but require a greater amount of computation and produce more information which must be eventually reduced to a single sample rate error estimate.

### 5.2.1 Variance

Sample population variance is a first order statistic widely used to estimate the amount of noise within a small image region such as a pixel. Since the spatial extent of the sample population is sufficiently small, variance is assumed to be inversely proportional to convergence of the iterative rendering process.

The variance of a collection of samples is given by the average distance between each sample and the mean:

$$\text{Var}(x) = \text{E} \left[ (x - \bar{x})^2 \right] = \frac{1}{N} \sum_{i=0}^N (x_i - \bar{x})^2 \quad (5.5)$$

Using samples from a framelet, variance may be interpreted as an approximation of average power, or unlocalized power magnitude across the entire extent of the framelet at all scales. With

this interpretation, the difference between each luminance sample and the mean is expressed as a potential difference, the squared value of which may be taken as approximate power. The sum of squared differences in equation 5.5 is then a norm

$$(x_i - \bar{x}) = V \quad (5.6)$$

$$\frac{1}{N} \sum_{i=0}^N (x_i - \bar{x})^2 = \frac{1}{N} \|V^2\|_1 \quad (5.7)$$

$$\approx \frac{1}{N} \|P\|_1 \quad (5.8)$$

where  $V$  may be considered analogous to voltage and  $P$  analogous to power.

Variance does not take into account any information about the relative position of individual samples. It provides only a single estimate for a spectral band between the minimum and maximum frequencies within the sample collection. This extent is bounded by the smallest and largest distance between two samples in the collection.

While these statistics provide qualitative information about the underlying image signal, it is difficult to deduce quantitative information necessary to determine an adaptive response control decision.

### 5.3 Frequency Domain Approximations

Statistics in the spatiotemporal domain are inexpensive and permit efficient incremental computation; however, they lack the ability to distinguish features at different scales and are a poor approximation to the power spectrum norm. Information about structure and scale, or the amount of energy at different frequencies, enables the sample rate error estimate to be less conservative and produce a much tighter bound on the sample rate error within a tile. For example, the ability to distinguish spatial features within a collection of samples permits a region with a smooth gradient across a luminance range to be distinguished from another region containing low magnitude noise in the same range.

Spatiotemporal statistics, especially variance, are ubiquitous in rendering and computer graphics in most cases because the sample population over which the statistics are computed, i.e. a subpixel region, is not large enough to contain complex structures. Subpixel regions may be well-represented by only a single constant value given by the mean. In the TSAR model, the cost of sample rate error estimation is amortized over a region much greater than a pixel. Adaptive framelet tilings produce sample populations which vary greatly in extent and include image components that range

in size from subpixel image features such as individual edges to much larger components of the scene geometry. The ability of the sample rate error estimate to distinguish between the size of these features and their location within the broad extent of a framelet allows for a more accurate characterization of signal behavior across the set of samples.

### 5.3.1 Localization

Analysis of the ideal sample rate  $\Theta^*$  and the error estimate  $\phi$  at the high resolution used in the example described in Section 5.1.2.1 is prohibitively expensive for real-time application. Decreasing the resolution of the analysis, i.e. taking the Fourier Transform of the signal downsampled to a lower resolution, might appear to be an attractive alternative; however, although the FFT could be used to estimate sample rate error, its discretization of the spatiotemporal and frequency domains are inefficient, and the opposite of what is desired to estimate sample rate error and varying adaptive response across the image. The FFT localizes, or isolates, frequencies using periodic basis functions with very finegrained differences in period, i.e. from  $1, \dots, N/2$ , but the consequence of the periodic basis with infinite support is that the transform does not perform any spatial localization. As a result, the transform produces more spectral information than necessary, i.e. significantly more frequencies are included in the transform than the renderer is capable of employing. Additionally, no information about spatially varying behavior, i.e. across the extent of a framelet, is obtained.

While the range of sampling frequencies realizable by the renderer, labeled  $\Omega_\theta$  in Figure 5.5(a), remains the same, the Fourier analysis performed in Section 5.1.2.1 causes the power spectrum norm, and therefore the ideal sample density  $\Theta^*$ , to vary with the position and extent of the framelet tile. Since the Fourier basis functions are periodic and assume that the input signal is stationary, the value of  $\Theta^*$  cannot be localized to a spatial coordinate more specific than the extent of the framelet tile. As the spatial extent of the framelet tiling adapts to the image signal, features within the support of coarse resolution basis functions may be excluded from the power spectrum of tiles at fine resolutions, resulting in a change to the power spectrum norm surface shape, and the corresponding location of  $\Theta^*$ .

In the case of tile 1442, shown in Figure 5.5(a), the position of  $\Theta^*$  across the whole tile is relatively far away from the current sample density  $a$ , shown in Figure 5.5(b). If the tile was subdivided, the smaller tile 5771 occupying the upper left corner of the region would contain much lower frequency features, and the ideal sample density estimate in that subregion of the tile would move closer to the point  $a$ . Ideally,  $\Theta^*$  would be by a continuously varying field across the image domain, instead of a tiling at framelet resolution, and would be defined as the ideal sample rate at a specific point in the image independent of the extent over which it is estimated.

### 5.3.1.1 Wavelet Transform

Wavelet transforms are a family of operators which project an input signal against basis functions at different scales translated across the spatiotemporal domain, and approximate the localization of signal behavior in both the spatiotemporal and frequency domains. This formulation of a discrete spatiotemporal wavelet approximation to the power spectrum norm follows the terminology and notation of Mallat [39].

Wavelet basis functions are a set of functions which integrate to zero across their domain:

$$\int_{-\infty}^{\infty} \psi(t) dt = 0 \quad (5.9)$$

These basis functions may be scaled by parameter  $s$  and translated by  $u$  to provide a parameterized transform:

$$\psi_{us}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{1}{s}(t-u)\right) \quad (5.10)$$

The continuous wavelet transform of a function  $f$  is then the correlation between  $f$  and  $u$ :

$$W[f(t)]_{us} = \int_{-\infty}^{\infty} f(t) \cdot \psi_{us}(t) dt \quad (5.11)$$

$$= f(t) * \psi_{us}(t) \quad (5.12)$$

The wavelet transform  $W[f(t)]_{us}$  is a continuous two-dimensional field over the parameters  $u$  and  $s$ . Modulation of the scale parameter  $s$  attempts to isolate energy in  $f(t)$  at frequencies approximately corresponding to the spectrum of the wavelet function  $\psi(t)$ , i.e. the convolution  $f(t) * \psi_{us}(t)$  in the spatiotemporal domain, to produce a scalar at coordinate  $u, s$ , is equivalent to the product of  $f$  and  $\psi$  in the frequency domain, which isolates frequencies within the spectrum of  $\psi$ , which is likely not bandlimited. In this way, the parameter  $s$  is an independent variable of *scale space*, denoted  $\omega$ , rather than the frequency domain, indicated by  $f$  in figures, e.g. Figure 5.6 which shows a discretization of the function. The terms scale and frequency will be used interchangeably

in this formulation; however, the wavelet scale parameter does not isolate individual frequencies in the same manner as the Fourier transform parameter.

The Haar basis, shown in Figure 5.7(a), used in this formulation has two sharp features which provide compact support and spatiotemporal localization, compared with a similar FFT basis shown in Figure 5.7(b); however, the power spectrum of the Haar basis, shown in Figure 5.7(c), is much broader than the Fourier basis, whose energy, by definition of the transform, is contained within a single frequency.

Interpreted as a difference operator, the Haar basis at different scales may correspond to power spectrum in a similar manner as equation 5.8. Unlike sample variance, the parameterization of the basis function enables the detection of oriented structure at different scales in the signal.

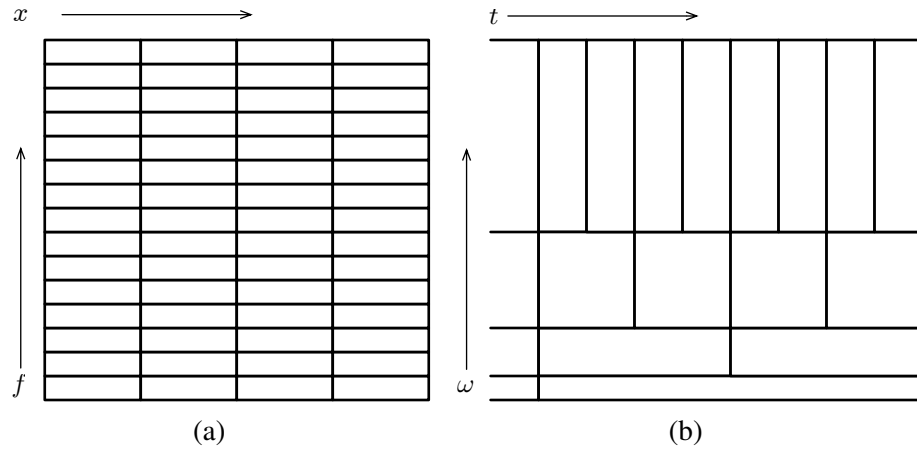
Discretization of the spatiotemporal and scale space plane based on a hierarchical power-of-two subdivision grid results in a transform known as the discrete wavelet transform (DWT). The DWT discretizes  $u$  and  $s$  with a hierarchical grid, shown in Figure 5.6(b), such that the finest scale of the wavelet is evaluated with the greatest spatial resolution and measures the highest frequency; the next finest scale is twice as large and evaluated at half the spatial resolution. The power of two relationship results in each discrete scale capturing energy in a decreasing octave of the signal spectrum. The parameter  $s$  is over discrete octaves  $\omega$  in the figure, and the parameter  $u$  over positions in the spatiotemporal domain  $t$ . Only one spatial dimension is shown in Figure 5.7.

The DWT may be expressed as a recurrence where coefficients in the highest octave are used to compute the coefficients of the second highest octave and so forth. At each octave, the signal is convolved with a wavelet function  $g$  and a scaling function  $h$ . The operator  $g$  is a high pass filter and the convolution produces  $\frac{N}{2}$  detail coefficients, while the operator  $h$  is a low pass filter producing  $\frac{N}{2}$  approximation coefficients of the signal. The DWT of the next slower frequency octave is computed from the approximation coefficients.

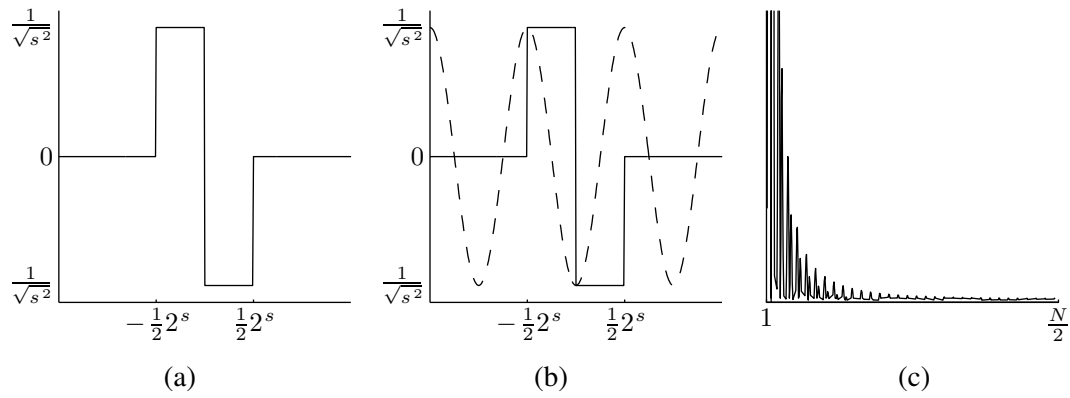
$$a_{s+1}[t] = a_s * h[2t] \quad (5.13)$$

$$d_{s+1}[t] = a_s * g[2t] \quad (5.14)$$

The discrete Haar basis function  $g$ , and the low pass filter  $h$  are:



**Figure 5.6.** Frequency domain discretizations of the windowed FFT and DWT.



**Figure 5.7.** The Haar wavelet function.

$$g[t] = \begin{cases} 1 & \text{if } 0 \leq t < \frac{1}{2} \\ -1 & \text{if } \frac{1}{2} \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.15)$$

$$h[t] = \begin{cases} \frac{1}{2} & \text{if } 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.16)$$

With a multidimensional signal, separable convolution is performed along each dimension with  $h$  and  $g$ .

Compared to the continuous algorithm, the DWT algorithm trades continuous localization in the spatiotemporal and scale domains for an efficient implementation. Compared to the FFT, the DWT provides precise spatial localization while allowing energy from a broader set of frequencies to contribute to each scale. In the case of TSAR, the low computational complexity of the transform



is attractive, and the decreased spectral localization is less of a concern since the transform is only used to characterize the signal rather than to represent, compress, or encode it.

### 5.3.1.2 Spatiotemporal DWT

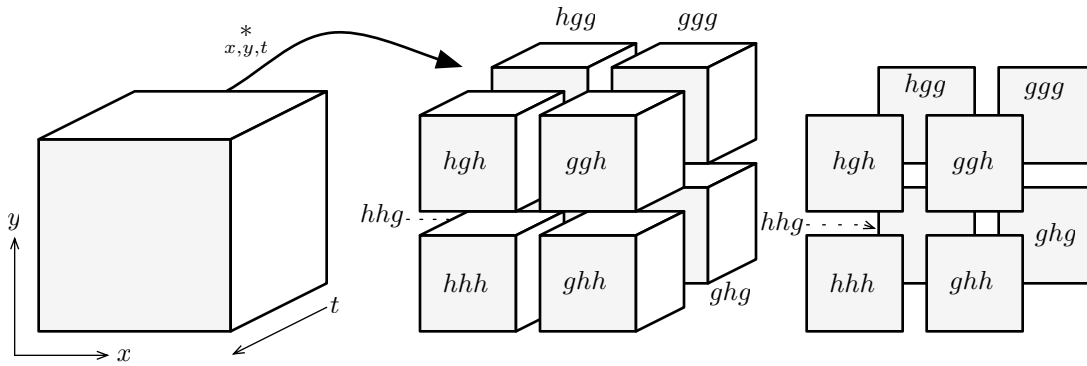
The DWT may be applied to spatiotemporal data by treating the signal as a three-dimensional function. At each octave, the approximation coefficients from the previous octave are convolved with  $g$  and  $h$ , along each axis separating the signal into detail and approximation coefficients. To simplify notation, the result of convolution with the  $h$  operator will simply be written  $h$  instead of  $a_{s+1} = a_s * h$ ; similarly  $g$  refers to the operation producing  $d_{s+1}$  from equation 5.14. After separable convolution in three dimensions, the coefficients in each octave are divided between eight *octants*, labeled  $hhh$  through  $ggg$  in the order of binary numbers, i.e. the coefficients in octant  $hhh$  are the result of the low pass filter applied in each dimension,  $ghg$  contains high pass in the first dimension, low pass in the second, and high pass in the third, etc. Figure 5.8 indicates the notation of octants and organization of coefficients in the DWT algorithm.

The eight octants at each octave of the spatiotemporal DWT characterize the magnitude and scale of different types of features in the signal. Octave five,  $hgh$ , detects horizontal edges,  $ggh$  spatial corners, and  $hgg$  corners in  $y$  and time. The octant  $hhh$  contains the approximation coefficients used to compute the next octave in the transform.

### 5.3.1.3 Leading Edge of Time

The canonical form of the DWT convolves each complete dimension with the  $h$  and  $g$  filters to produce a set of coefficients equal in size to the input; however, much less than the complete temporal dimension changes with each refresh of the rendering pipeline. In the canonical form, for  $N^3$  input luminance values, eight octants of  $(N/2)^3$  coefficients are produced, seven containing detail coefficients and one containing the low pass approximation of the signal. If the spatiotemporal transform is performed at each temporal refresh, only the input on the spatial plane at the leading edge of time may change, and only coefficients in the  $N^3$  volume that are a function of luminance values at the leading edge need to be updated. The leading edge of time is shaded in Figure 5.8, which shows a single octave; given the input of  $N^2$  new luminance values in the spatial plane,  $2N^2$  coefficients are produced, utilizing  $N^2$  cached coefficients. Coefficients in the canonical transform before the leading edge of time have already been computed by the algorithm prior to the current time.

The number of coefficients that must be retained depends on the support of the wavelet basis function used in the temporal dimension. In the case of the Haar basis, with a support of two, only one set of spatial coefficients must be retained at each octave. Figure 5.9 shows the discretization of



**Figure 5.8.** Notation for the DWT in three dimensions of a scalar volume into eight octants.

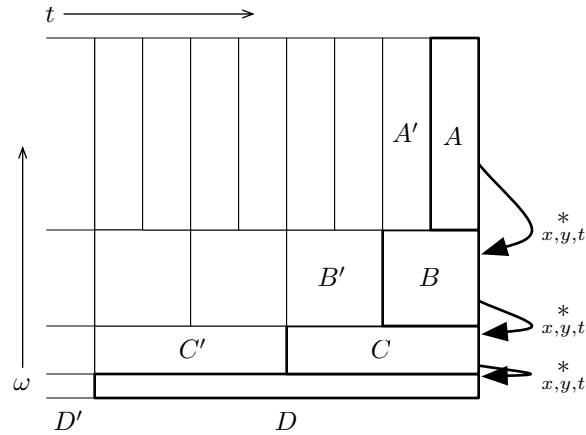
the scale and time plane; the spatial dimensions are not shown. Along the temporal axis, the highest frequency wavelet basis  $A$  is convolved with the cached coefficients  $A'$ , producing coefficients in eight octants. At octave  $A$ , the low pass octant  $hhh$  is the input  $B$  at the leading edge of time. These coefficients are combined with the cached values  $B'$ , and so forth for octaves  $A$  through  $D$ .

This relationship between coefficients of the complete transform needed to incrementally update the DWT along the leading edge of time is the basis for the streaming DWT algorithm described in Section 5.3.4. The algorithm operates on data stored in Morton order in framelet tiles, instead of the implicit row major indexing scheme shown in equations 5.13 and 5.14.

### 5.3.2 Localized Levelwise Norm

The spatiotemporal DWT applied at the leading edge of time provides spatial localization proportional to feature scale, and in the temporal dimension, discretization of scale space at the leading edge of time. The transform does not reduce the dimensionality of the input data, and its results must be converted to a measurement of sample rate adequacy, analogous to equation 5.2, to produce an estimate of spatial and temporal sample rate error. This will be formulated in two steps: first, a norm operator is shown to reduce the spatial and temporal dimensionality of the transform and divide the information between spatial and temporal features; then, an operator is given to produce an error estimate by comparing the norm with a threshold. Both operations are formulated in one dimension and then applied in the spatiotemporal domain.

The localized levelwise norm (LLN) operation reduces the dimensionality of the DWT while preserving or emphasizing certain characteristics of the data. The LLN is the application of a vector norm to a set of coefficients at each scale within a specific spatiotemporal extent. The approach reduces the dimensionality of the input while the choice of norm operator allows filtering of different characteristics, e.g. a nonmetric operator such as the infinity norm may be used to provide a bound on the magnitude of power within a region, while an  $L_2$  norm reports the total amount of power.



**Figure 5.9.** Heisenberg boxes for the DWT along the temporal axis; the spatial axes are not shown. Atoms intersecting the leading edge of time are highlighted.

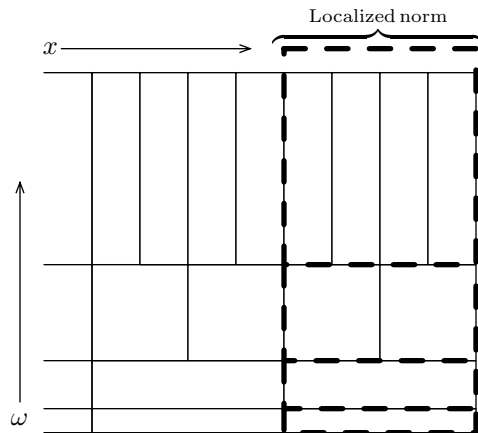
Figure 5.10 shows the extent in scale space of an LLN operator applied in one spatial dimension. Coefficients overlapping the LLN extent at each scale are treated as a vector to which the norm is applied. In the figure, the highest frequency scale contains four coefficients, the next highest two, etc.; at the lowest frequency, the wavelet basis is broader along the spatial axis than the extent of the LLN, and a proportional fraction of the energy is included in the norm. In the example shown, the eight coefficients are reduced to four scalar measures, one at each scale; in higher dimensional cases, the reduction is greater and the norm may be designed to isolate specific behavior by only including energy from specific octants.

At each octave, octants  $ghh$ ,  $hgh$ , and  $ggh$  are included in the spatial LLN, and  $hhg$ ,  $hgg$ , and  $ghg$  are included in the temporal error norm. The eighth octant  $ggg$  detects corners in all dimensions; since this octant combines high pass energy from both space and time, it does not assist in distinguishing spatial and temporal change, and is not used.

The LLN transforms the spatiotemporal DWT into a set of vectors describing the spatial and temporal power spectrum of the signal at each point across the spatial plane, at the leading edge of time. The shape and magnitude of these vectors are used to determine sample rate adequacy across the image.

### 5.3.3 DWT Error Estimation

The error estimation stage uses a DWT to approximate the power spectrum of the sampled image signal in each framelet, and then uses a LLN-based comparison to approximate the local shape of the  $\Theta^*$  surface. In terms of Figure 5.5(b), the point  $a$  is given by the current spatial and temporal sample rate of the framelet and the error vector  $\phi$  is produced by using an LLN based comparison to approximate the local shape of the power spectrum norm surface.



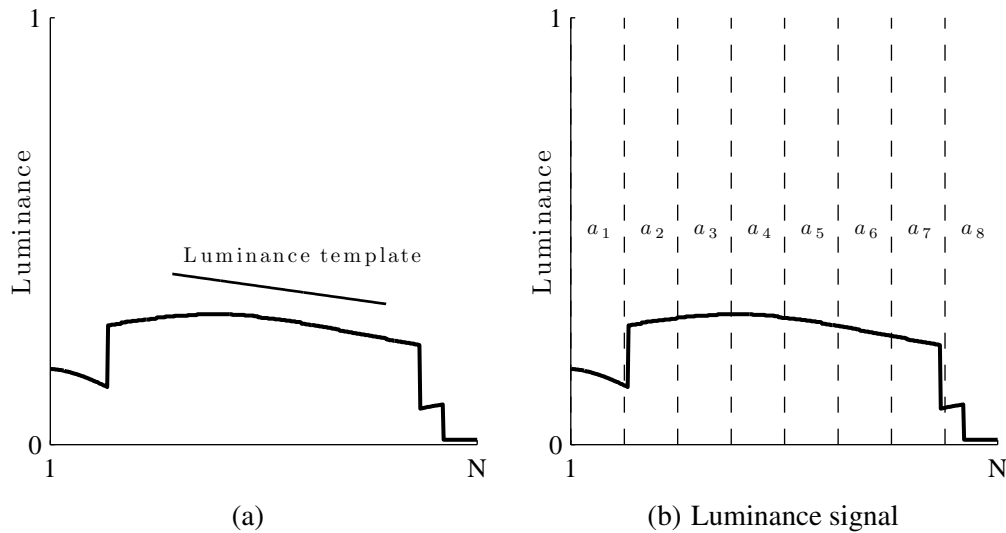
**Figure 5.10.** Localized levelwise norm in scale space.

### 5.3.3.1 Signed LLN Difference

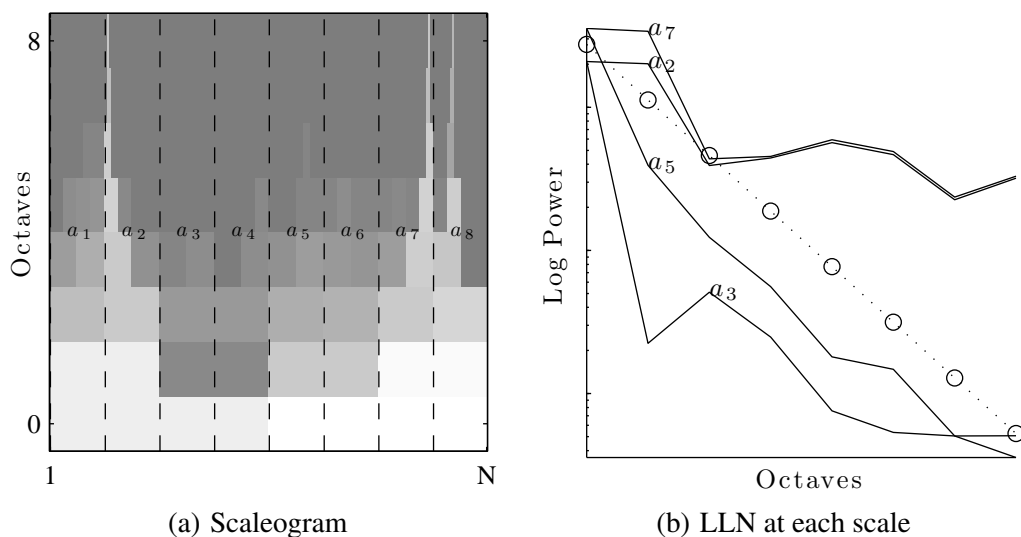
Ideally, the desired power spectrum of the sampled image, e.g. the point  $\Theta^*$  in Figure 5.5(b), would be dictated by properties of the reconstruction process, such as the spectral support of resampling filters in the manner of the filter design process of Mitchell [42]. Without a wavelet convolution theorem, neither the reconstruction process, nor its consequences in frequency space, may be related directly to the DWT power spectrum. Instead, the DWT power spectrum is compared to a threshold amount of energy at each octave using an empirically defined threshold template power spectrum. The comparison is analogous to the conventional band pass analysis performed in frequency space, but the two operations are not mathematically equivalent. The template power spectrum may be generated by taking the DWT power spectrum of a specifically formulated luminance signal which represents desired sampled signal behavior in the vicinity of  $\Theta^*$ .

The LLN-based error estimation approach for a one-dimensional luminance signal is shown in Figures 5.11 through 5.13. In the error estimation stage of the TSAR pipeline, the same procedure is performed for both spatial and temporal error estimates using the LLN over each set of high frequency spatial and high frequency temporal DWT octants, instead of over a one-dimensional luminance signal in the example figures. Since the norm reduces coefficients in three of the DWT octants at each scale to a single scalar value, the approximation is analogous to the one-dimensional example.

The error estimation procedure approximates the local shape of the signal power spectrum within the framelet by taking the signed norm between the template function power spectrum and the signal DWT power spectrums. The signed norm is sum of the differences between the LLN across three spatial or three temporal octants, depending the component of the error vector, over all octaves. Two LLN measures, for spatial and temporal signal behavior, respectively, are computed



**Figure 5.11.** Characterization using the localized levelwise norm in one dimension.

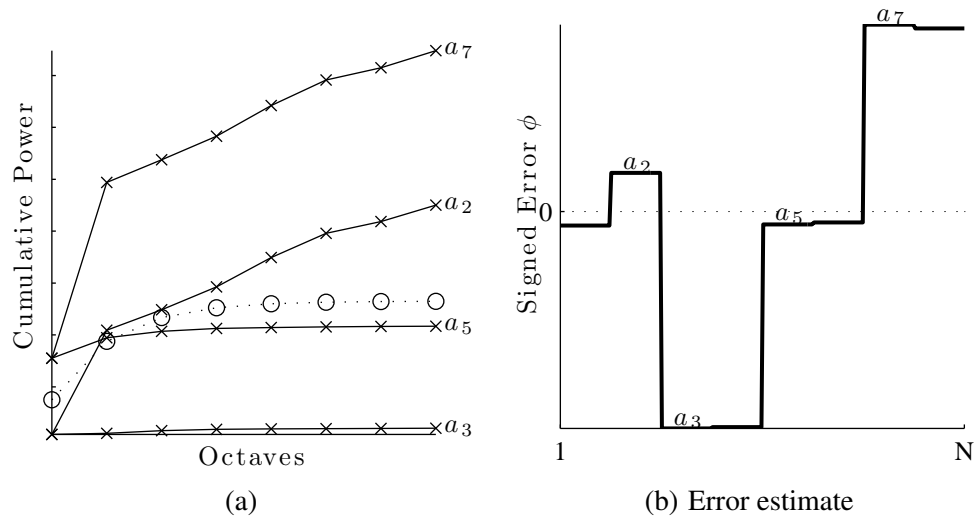


**Figure 5.12.** Example signal scaleograms

across the framelet. Since the spatial sample density is assumed to be isotropic, and is controlled by a single parameter, it is not necessary to distinguish between vertical and horizontal spatial features; therefore, energy within high pass spatial features must only be distinguished from energy in high pass temporal, and low pass spatial features.

### 5.3.3.2 Threshold Function

The template DWT power spectrum is assumed to have exponential shape both due to the spectral characteristics of the expected graphics workload, related to those of natural imagery and



**Figure 5.13.** Example signal error summation.

described in Section 1.3.4, and the behavior of the DWT whose low pass averaging operator, given in equation 5.13, tends to accumulate power in coarser scales.

Linear luminance functions produce a DWT power spectrum which is uniform in space with exponentially increasing magnitude in coarser scales. This characteristic may be observed by considering the application of the high pass detail coefficient operator of the Haar wavelet, equation 5.14, which takes the form of a difference operator, to luminance functions of different shapes. When the detail coefficient operation is applied to a luminance template function with varying slope across space, it produces coefficients of varying magnitude. However, a template luminance function with a constant slope produces the same difference everywhere and at coarser scales, the difference operator is applied over a wider extent of the signal hierarchically which produces an exponentially larger magnitude difference at increasing octaves.

Figure 5.11(a) shows an example luminance function plotted along with a linear luminance template function. In this example, the slope of the luminance template function is determined empirically by selecting the slope of the luminance signal function which is adequately and efficiently reconstructed at sample rate  $N$ ; i.e. the spatial coordinate of  $\Theta^*$  occurs near sample density  $1/N$  for this imagery.

The template power spectrum for both the spatial and temporal domains is defined as a linear function in a logarithmic space with two parameters, a vertical offset and a negative slope. In a linear space of the LLN power spectrum, the template function is an exponentially decreasing distribution. Although the LLN power spectrum norm space is not mathematically equivalent to the frequency domain power spectrum, the template function vertical offset parameter is analogous to the DC

component of the signal in the frequency domain and the slope parameter analogous to the desired distribution, e.g. following Figure 1.3,  $-2$  for natural imagery. The implications of this design and examples of the algorithm's sensitivity to the parameters is discussed in Section 7.3.2.

### 5.3.3.3 Computing the LLN

The example luminance signal in Figure 5.11 was taken from a simple scene and consists of four luminance gradients produced by Phong illumination separated by sharp edges. In Figure 5.11(b), the horizontal coordinate axis of the plot is divided into eight regions enumerated  $a_1$  through  $a_8$ ; a levelwise norm and error estimate will be localized to each of these regions. The expected error response in this case should distinguish regions containing sharp features from those containing low frequency gradients.

Figure 5.12(a) shows the LLN power spectrum scaleogram of the luminance signal. The scaleogram is computed by squaring the DWT coefficients of the signal and then taking the levelwise norm over each of the eight regions. Fine octaves, approximating high frequencies, are at the top of the figure and coarse octaves, approximating lower frequencies, are at the bottom. Pyramid shaped energy features may be observed in the scaleogram about regions of the luminance signal containing sharp features, in the regions  $a_1$ ,  $a_7$ , and  $a_8$ . Lower power features may be distinguished starting in the middle scales 5 and 6 in regions  $a_4$  and  $a_5$ .

The LLN power spectrum scaleogram magnitudes for regions  $a_2$ ,  $a_3$ ,  $a_5$ , and  $a_7$  are plotted in Figure 5.12(b) on a log scale along with the DWT power spectrum of the luminance template function from Figure 5.11(a). Regions  $a_2$  and  $a_7$  contain sharp features at either side of the signal and therefore have more power at finer scales, to the right of the plot, than regions  $a_3$  and  $a_5$  which contain smooth features. In general, all regions of the signal will contain energy at every scale; even in cases like  $a_3$  which is clearly oversampled, the smoothly varying gradient contains low magnitude power at fine scales.

The signed difference between the template and the signal DWT for the spatial and temporal error estimates are computed incrementally octave by octave using the approach as described in Section 5.3.4. The result is a cumulative summation of the difference between the signal and template from fine to coarse scales. The cumulative power from coarse to fine scales in the four example regions is shown in Figure 5.13(a) on a linear scale. This cumulative summation of the signal and template DWT power spectrums provides similar information about the  $\Theta^*$  surface shape as the power spectrum norm shown in Figure 5.5(b). Both the shape and magnitude of each power spectrum distinguish the sample rate error estimate and the eventual adaptive response in each region. All regions have similar shape in coarse scales. In the midrange scales, the energy captured at finer scales levels off in regions  $a_3$  and  $a_5$ , while the amount of power continues to

increase through the finer scales in regions  $a_2$  and  $a_7$ . While the LLN of regions  $a_3$  and  $a_5$  are shaped similarly to the template thresholds, behavior within the two regions is distinguished by the magnitude of power within them. Overall, region  $a_3$  has less signal power than  $a_5$ , or the template, which indicates more redundancy between samples in the framelet and results in a greater negative magnitude error estimate.

The result of the LLN-based error estimate procedure for the one-dimensional test signal is shown in Figure 5.13(b). The estimate is piecewise constant over the interval of the luminance function based on the resolution of the eight localized error estimates.

### 5.3.4 Streaming Discrete Wavelet Transform

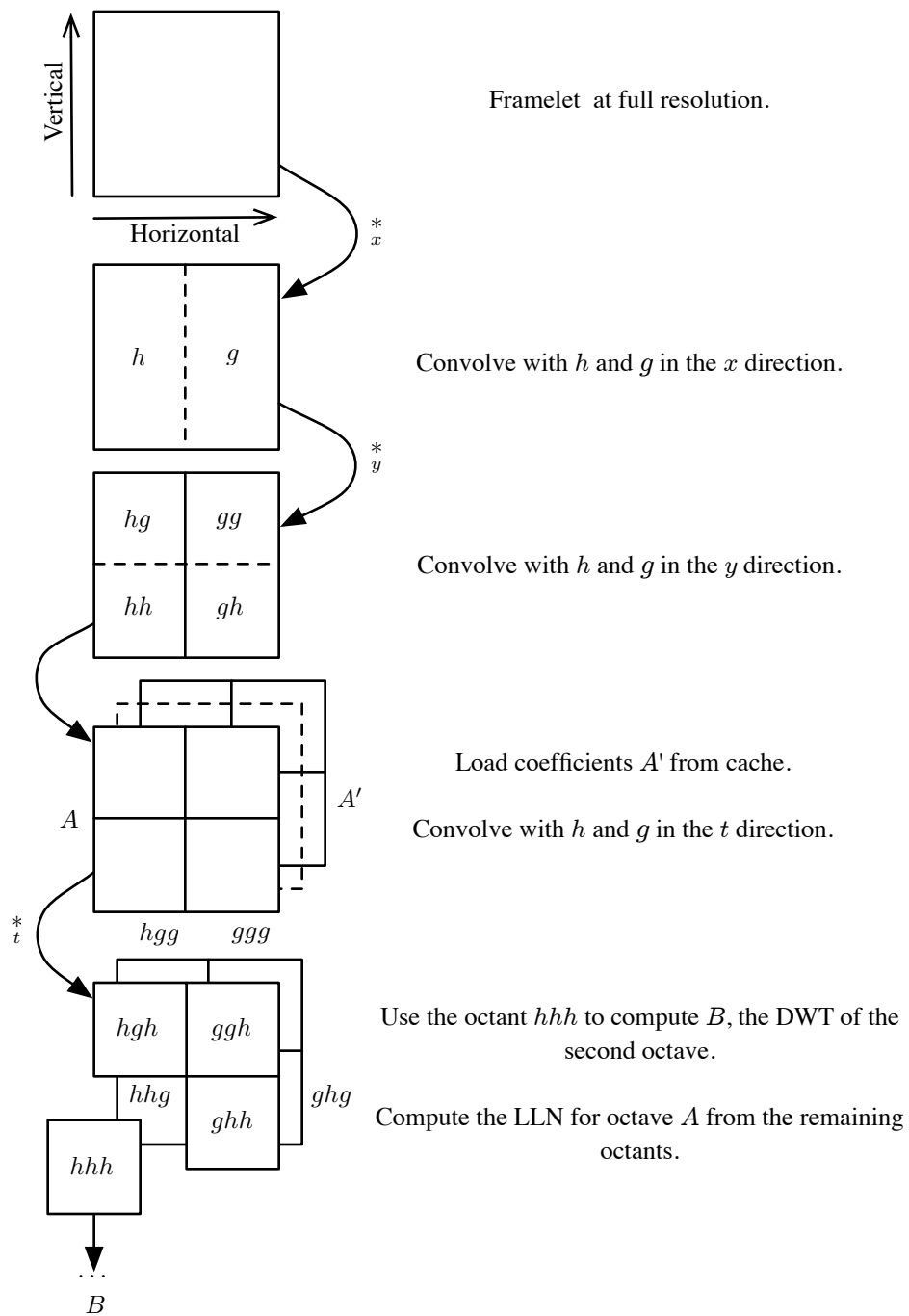
The localized levelwise norm may be applied to a spatial temporal signal by combining coefficients within a certain spatial extent at the leading edge of time. The LLN produces a spatially varying measure of energy at each octave resulting in a set of coefficients at each point across the image similar to the one-dimensional example shown in Figure 5.13.

The LLN operation performs a significant reduction over all of the DWT octant coefficients at each octave, combined with the signed difference comparison between the template and framelet signal. The storage overhead of the streaming algorithm is minimal. For each octave of the DWT, detail coefficients are produced in seven two-dimensional octants at the leading edge of time, plus one two-dimensional octant of low pass coefficients. Six of the seven octants are combined by the LLN into two norms, one spatial and one temporal, while the seventh octave is unused.

The streaming DWT algorithm performs three convolutions in the spatial and temporal dimensions at each octave; the procedure is shown in Figure 5.14 using the canonical DWT indexing scheme for the purpose of illustration. Given a framelet at full resolution, the first octave of fine scale coefficients is obtained applying equations 5.13 and 5.14 to the luminance value of each sample across the horizontal  $x$  dimension. The detail coefficients given by equation 5.14 are packed immediately following the approximation coefficients given by equation 5.13 in the output. Next, the high and low pass operators are applied along the  $y$  direction and the results are packed similarly. The result of the two spatial convolutions with the new framelet consists of the same number of coefficients as samples in the input framelet, these coefficients are defined as field  $A$  for the first octave.

Temporal convolution is performed by loading a matching set of coefficients  $A'$  and applying the high and low pass operators. The third convolution produces coefficients in eight octants over which the spatial and temporal LLNs are performed. Coefficients in the first octant  $hhh$  are used as the input at the leading edge of time to the transform at the next highest octave. After convolution





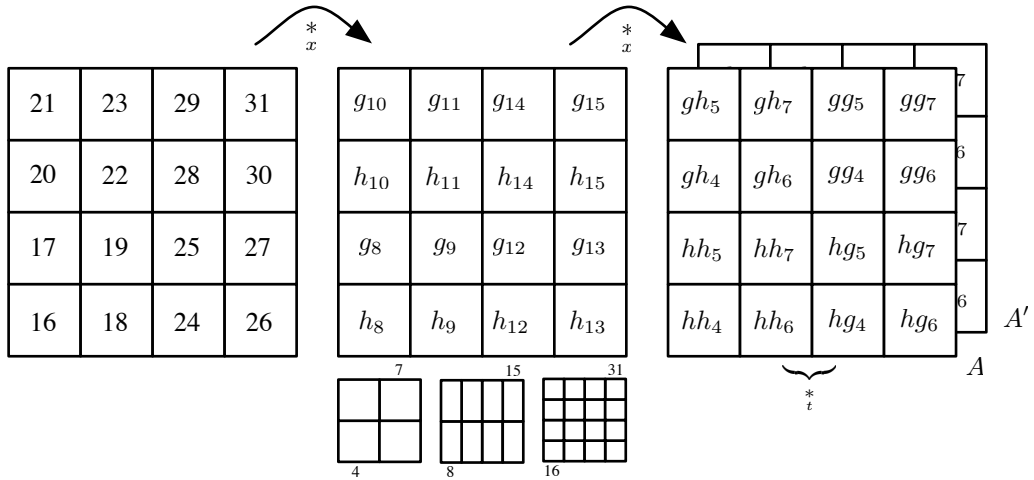
**Figure 5.14.** Computation of one octave in the streaming DWT.

in the temporal dimension, the coefficients  $A$  replace  $A'$  in the cache. In instances where the cached coefficients  $A'$  do not exist, only spatial error estimates are reported.

The DWT is performed on framelet samples stored in Morton order and require modification of the canonical DWT algorithm indexing scheme given in equations 5.13 and 5.14. In Morton order, described in Section 4.1.2, coefficients are stored along a space filling curve instead of a row major, or column major layout. Although the latter data layouts are more efficient for general separable convolution, the extent of the Haar wavelet operators is only two elements so each convolution in the DWT may be applied to a  $2 \times 2$  neighborhood of the input.

The indexing scheme for the two spatial convolutions of the DWT is shown in Figure 5.15 for a  $4 \times 4$  example framelet; the actual framelet resolution used for testing the prototype is either  $16 \times 16$  or  $32 \times 32$ . Coordinates are addressed by subscripts based on Morton quad-tree tile indices described in Section 4.1.3, so the first index in the  $4 \times 4$  example is tile 16. After the first two spatial convolutions, the input will be divided into four quadrants containing coefficients of the  $hh$ ,  $gh$ ,  $hg$ , and  $gg$  operations; coefficients within each quadrant are arranged in Morton order. The first convolution along the  $x$  direction stores the result using an interleaved packing within the same  $2 \times 2$  neighborhood. The indexing scheme after the first convolution follows the Morton kd-tree tiling order using the next coarser level of the tree in the first dimension, i.e. the result of convolving  $F_{[16]}$  and  $F_{[17]}$  with  $h$  and  $g$  is stored in two elements starting at the location of kd-tree tile  $8 = 16/2$ , labeled  $g_8$  and  $h_8$  in the figure. The second convolution along the  $y$  direction operates within the same neighborhood of coefficients, across  $g_8$ ,  $h_8$  and  $g_9$ ,  $h_9$ , but stores its results in the uninterleaved packing of separate quadrants.

The result of the temporal convolution in the seven high pass octants is immediately combined with the incremental LLN for the octave, while the uninterleaved ordering ensures that the first octant containing low pass coefficients in each dimension is in the same sample order of the input at half the resolution.



First convolution:

$$\left. \begin{aligned}
 h(F_{[16]}, F_{[17]}) &\rightarrow F_{[16]} = h_8 \\
 g(F_{[16]}, F_{[17]}) &\rightarrow F_{[17]} = g_8 \\
 h(F_{[18]}, F_{[19]}) &\rightarrow F_{[18]} = h_9 \\
 g(F_{[18]}, F_{[19]}) &\rightarrow F_{[19]} = g_9
 \end{aligned} \right\} \text{Interleaved packing.}$$

Second convolution:

$$\left. \begin{aligned}
 h(F_{[16]}, F_{[18]}) &\rightarrow F_{[16]} = hh_4 \\
 g(F_{[16]}, F_{[18]}) &\rightarrow F_{[20]} = hg_4 \\
 h(F_{[17]}, F_{[19]}) &\rightarrow F_{[24]} = gh_4 \\
 g(F_{[17]}, F_{[19]}) &\rightarrow F_{[28]} = gg_4
 \end{aligned} \right\} \text{Uninterleaved packing.}$$

Figure 5.15. Interleaved coefficient packing.

## CHAPTER 6

### ADAPTIVE RESPONSE CONTROL DECISION

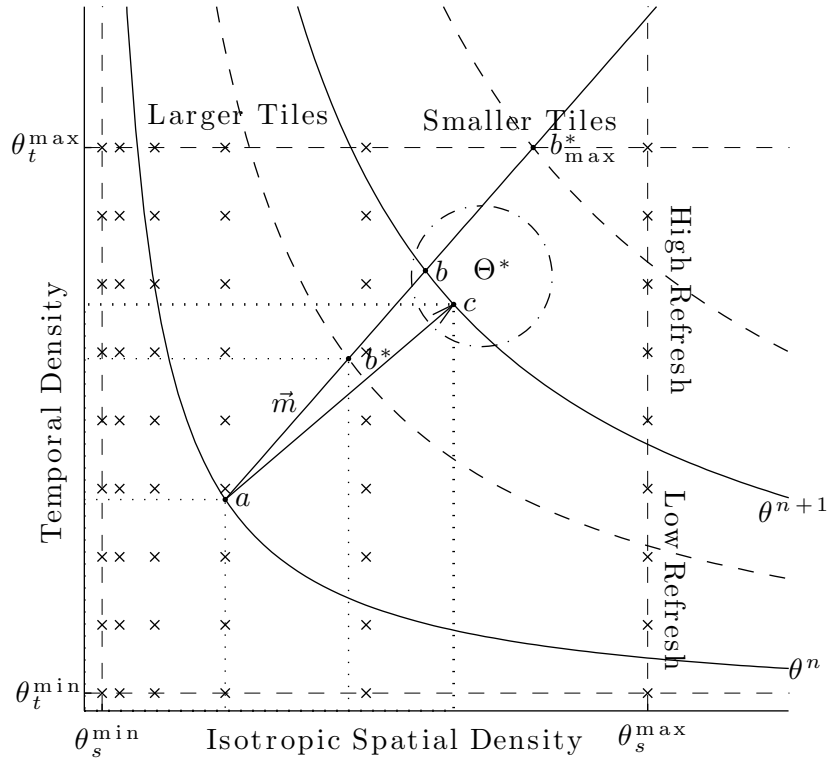
The adaptive response control decision modulates spatial and temporal sample density across the image based on a sample rate error estimate produced by the algorithms described in Chapter 5. Sample density error values characterize the amount of redundancy between samples as a measure of sampling process efficiency. The estimated error vector describes the difference between the current sample density and the optimal sample density across the image. The response control algorithm applies a series of local and global constraints to the sample rate error estimate to safely update the sample densities and respond to changes in the graphics scene.

#### 6.1 Decision Problem

The adaptive response control decision manipulates sample rates used for rendering by changing spatial and temporal sample densities of elements across the image. If modeled in a two-dimensional space with a single isotropic spatial density along one axis and the temporal sample density along the other, the control decision estimates a transformation between an initial coordinate, the current sample density of each element, and a point at an unknown coordinate corresponding to the ideal sample density in the underlying signal. Figure 6.1 shows the space of sample densities, termed a  $\theta$ -space, for a single element.

Point  $a$  represents the current spatial and temporal sample density for the element, with densities at coordinates  $a_s$  and  $a_t$ , respectively, and combined spatiotemporal sample density  $\theta^n$ . The curve in the figure passing through point  $a$  contains other combinations of spatial and temporal density equal to  $\theta^n$ . Point  $a$  is the current estimate of the desired sample density for the element, measured on a real valued scale, based on the convergence of the adaptive response control process before the leading edge of time. Other points in the figure are determined by the control algorithm and contribute to the eventual control decision for the algorithm. The space is bounded by the minimum and maximum possible spatial and temporal sample densities of the system,  $\theta_s^{\min}$ ,  $\theta_s^{\max}$ ,  $\theta_t^{\min}$ ,  $\theta_t^{\max}$ , respectively. Larger values along the horizontal axis correspond to higher spatial densities and smaller framelet tiles. Larger values along the vertical axis correspond to higher temporal refresh.

While the sample density space in Figure 6.1, for one element is continuous, the adaptive



**Figure 6.1.** The adaptive response control decision problem for one element. Point  $a$  is the initial density of the element, points  $b^*$  and  $b$  are intermediate solutions, and point  $c$  is the new sample density. The vector  $\vec{ca}$  is the result of the control decision for the element.

renderer is only able to place samples at specific spatial and temporal sample rates indicated by the irregular grid of crosses. Information about the estimated ideal sample density and the control decision itself are made at a much higher numerical precision. These values are approximated by the discrete sample rates realizable by the renderer during a quantization step when a discrete framelet tiling is built over the sample density fields, described in Section 4.1.6.

The objective of the adaptive response control decision for each element is to select a pair of spatial and temporal sample densities as close as possible to the ideal sample density  $\Theta^*$  which is an unknown property of the underlying image signal. The new sample density for the element  $b^*$  must fall within a set of local and global constraints on the sample density values. The control decision algorithm has three stages: first, local constraints are enforced on the local sample rate error estimate to determine the amount of change in combined sample density at the element; next, global constraints such as conservation of combined sample density are enforced across all of the elements; and lastly, the constrained amount of combined sample density change is compared once more to the sample rate error estimate and modulated by any spatiotemporal fidelity bias. The first and last stages consist of finding solutions to equations locally at each element in the space shown

in Figure 6.1, the second stage is solved using a transport equation over all elements.

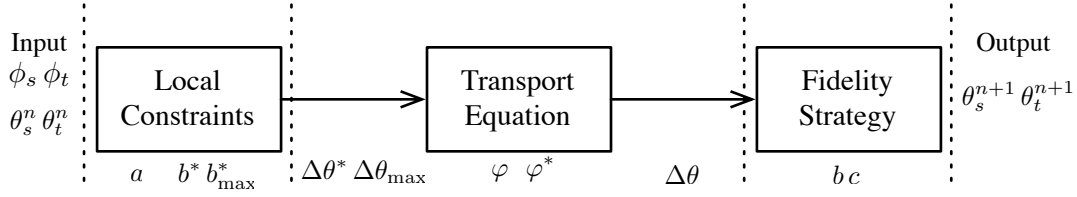
### 6.1.1 Schematic Interpretation

In terms of the illustration in Figure 6.1, the control decision algorithm is formulated as follows. The direction and distance between  $a$  and  $\Theta^*$  is estimated by an error vector  $\phi$ , with direction  $\vec{m}$ , computed by the mechanisms described in Chapter 5. Individual error estimates only characterize local behavior of the sampled signal and do not adhere to local constraints on the sample densities at the element or global constraints on the sample density across the image as a whole. To enforce local constraints on the sample density of the element, the control algorithm determines point  $b^*$ , the desired new sample density for the element. The difference between the combined spatiotemporal sample densities at  $b^*$  and  $a$  is the locally constrained net change of sample density  $\Delta\theta^*$  at the element. The control algorithm enforces global constraints by modulating the amount of change in combined sample density permitted at each element, and determines a new combined sample density  $\theta^{n+1}$  and constrained  $\Delta\theta$ . The local constraints are communicated to the second stage of the algorithm by specifying a maximum amount of combined change  $\Delta\theta_{\max}$  between  $a$  and  $b^*_{\max}$ . In the final stage of the control decision algorithm, a specific point  $b$  on the curve given by  $\theta^{n+1}$  is selected based on  $\vec{m}$ ; further modulation, for the implementation of a biased fidelity management strategy, is possible by moving the solution  $b$  along the curve  $\theta^{n+1}$  to point  $c$ . The vector  $\vec{ca}$  is the control decision for the element. These stages are illustrated in Figure 6.2 with the per-element variables related to each stage indicated.

In the case shown in Figure 6.1, the control decision increased the spatial sample density slightly more than the temporal sample density. The response might occur after the disocclusion of an object, requiring a higher refresh rate in an area of the image, and additional spatial samples to reconstruct the object's features. Depending on the selection of discrete sample densities from the realizable grid, this control decision may double or quadruple the sample rates used by the adaptive renderer.

The new spatiotemporal sample rate of the element at point  $c$  likely occurs between nodes in the grid of realizable sample rates. The selection of an optimal sample rate between the realizable candidates closest to  $\Theta^*$ , the four highest grid points in this case, depends on the bias of the fidelity management strategy. If temporal fidelity is more important than spatial fidelity, the upper right node might be selected, or conversely the lower right node.

The sample rate must move between nodes on the realizable grid to alter sample placement in the renderer and improve the fidelity of the graphics process. The grid may be exponentially spaced along the horizontal axis, due to the constraints of framelet tiling, and uniformly spaced along the temporal axis based on the cost of rendering a batch of new samples and executing the control algorithm.



**Figure 6.2.** Control decision stages and related variables.

### 6.1.2 Proportional Response

The control decision may be expressed as the time derivative of the spatial and temporal sample density across the image. At the leading edge of time, this derivative is approximated by sample rate error, an estimate of the direction and distance between the current sample rate and the ideal sample rate for the signal. The spatiotemporal error fields  $\phi_s$  and  $\phi_t$  are approximately the derivative of their respective sample density fields  $\theta_s$  and  $\theta_t$ :

$$\left. \begin{aligned} \frac{d}{dt}\theta_s &\approx \phi_s \\ \frac{d}{dt}\theta_t &\approx \phi_t \end{aligned} \right\} \quad (6.1)$$

On notation, an arbitrary scalar value of *field theta spatial* is written  $\theta_s(m)$  for element  $m$  in Morton order (see: 4.1.2), or simply  $\theta_s$  in an elementwise operation; equations of the whole field, such as reduction over the whole field, are written  $\theta_s$ , i.e. *the entire field  $\theta_s$  indexed by  $m$* . Equations 6.1 and 6.2 refer to elementwise operations; an example of the latter form will be equation 6.24, concerning a sum over all elements in the field.

The relation in equation 6.1 is approximate because the error estimate  $\phi$  is a characterization of the local signal and is unconstrained. Direct application of  $\phi$  as a derivative might violate constraints on the sample density field; instead the derivative is expressed as a function  $\hat{k}$  of the error estimate:

$$\frac{d}{dt}\theta_{s,t} = \hat{k}(\phi_s, \phi_t) \quad (6.2)$$

The function  $\hat{k}$  must enforce constraints on the sample density and is the principle design choice of the control decision algorithm, for example, the choice of how global constraints are enforced, such as positivity of the sample density and conservation of the total amount of sampling work across the image. In Section 6.3.1.2 a system of equations with relational uncertainty is used to

enforce global constraints; Section 8.4 describes the use of a partial differential equation based on a physical model. Both formulations of  $\hat{k}$  address local constraints on the sample density as the solution to a system of equations with linear relations and a pair of nonlinear terms constituting the mixture of spatial and temporal density change. The fidelity management strategy of the adaptive rendering process is implemented by manipulating bias between of the mixture components.

## 6.2 Local Constraints

As illustrated in Figure 6.1, the spatial and temporal sample density of an element may be represented as a point in Cartesian coordinate space, where a certain element at position  $m$  in the image has spatiotemporal sample density represented by point  $a = \langle \theta_s(m), \theta_t(m) \rangle$ . The combined sample density of  $a$  is the product of the spatial and temporal dimensions, or the area of rectangle between point  $a$  and the origin:

$$\text{Area}(a) = \theta_s(m) \theta_t(m) \quad (6.3)$$

$$= a_s a_t \quad (6.4)$$

$$= \theta^n \quad (6.5)$$

In Section 3.1.1, an element of the sample density field is defined as a three-dimensional volumetric entity at the leading edge of the spatiotemporal volume; the same element is considered here in two dimensions since the spatial sample densities are assumed to be isotropic in so far as horizontal and vertical sample rates are the same, i.e. both spatial densities  $\sqrt{\theta_s}$ ; therefore  $\text{Area}(a)$ , indicated by dotted lines in Figure 6.1, is the same as the quantity illustrated in Figure 3.3 projected to two dimensions.

The variable  $\theta^n$  is the combined spatiotemporal sample density volume of element  $m$  at time  $n$ . The objective of the local constraint enforcement stage of the control algorithm  $\hat{k}$  is to determine an appropriate value:

$$\Delta\theta = \theta^{n+1} - \theta^n \quad (6.6)$$

for each element in the sample density field. Substituting equation 6.4 into equation 6.6  $\Delta\theta$  is expressed as a difference between points  $b^* = \langle b_s, b_t \rangle$  and  $a$ :



$$\begin{aligned}
\Delta\theta &= (\theta_s^{n+1}\theta_t^{n+1}) - (\theta_s^n\theta_t^n) \\
&= (b_s^*b_t^*) - (a_s a_t) \\
&= \text{Area}(b^*) - \text{Area}(a)
\end{aligned} \tag{6.7}$$

Where the point  $b^*$  is the desired new spatial and temporal sample densities for element  $m$ . Both areas are indicated by dotted lines in Figure 6.1.

To implement proportional response, the location of  $b^*$ , and the net change in the total sample density of the element  $\Delta\theta$ , should be proportional to the spatial and temporal sample rate error estimate produced using the mechanisms described in Chapter 5. The error mechanism produces two signed error estimates, nominally  $\phi_s, \phi_t \in [-1, 1]$ ; the *mixture ratio* between spatial and temporal error may be expressed as a normalized vector:

$$\vec{m} = \frac{1}{\|\langle \phi_s, \phi_t \rangle\|_2} \cdot \langle \phi_s, \phi_t \rangle \tag{6.8}$$

Then the set of solutions  $b^*$  with the mixture ratio occur at a distance  $k$  along a ray extending from the point  $a$  in the direction of  $\vec{m}$ :

$$b^* = a + \vec{m}k \tag{6.9}$$

The adaptive renderer has a minimum and maximum sample density constraint for both spatial and temporal dimensions, written  $\theta_s^{\min}, \theta_s^{\max}, \theta_t^{\min}, \theta_t^{\max}$ , from which constraints on the value  $\theta^{n+1}$ , and  $\text{Area}(b^*)$  may be derived. These constraints may be expressed by linear relations:

$$\begin{aligned}
\theta_s^{\min} &\leq b_s^* \leq \theta_s^{\max} \\
\theta_t^{\min} &\leq b_t^* \leq \theta_t^{\max}
\end{aligned} \tag{6.10}$$

### 6.2.1 Response Speed

The overall speed of the adaptive response is the rate at which the sample density changes in reaction to changes in the graphics scene or animation within each adaptive response cycle, i.e.  $\frac{\|\Delta\theta_s, \Delta\theta_t\|}{\Delta t}$ , where the control decision algorithm is executed at  $\Delta t$  intervals. The response speed depends on the accuracy of sample rate error estimation in detecting features and the amount which the spatial and temporal density change is modulated by local and global constraints.

The proportional response criteria makes the assumption that the ideal spatial and temporal sample density, shown as the possible set  $\Theta^*$  in Figure 6.1, is an unknown distance from  $a$  along  $\vec{m}$ . Although the actual distance is unknown, it is assumed to be proportional to the length of  $\phi$ . This ensures that elements with high magnitude error respond more quickly than elements with low magnitude error, even if they have the same error direction. The choice of the ray parameter  $k$  in equation 6.9 determines the response speed at the element, before modulation by global constraints, and may determine whether the response control decision overshoots  $\Theta^*$  resulting in underdamped control behavior, or approaches  $\Theta^*$  too slowly, resulting in overdamped response.

Adaptive response that is proportional to the error estimate  $\phi$  occurs if  $b^*$  falls along the ray equation 6.9, a distance proportional to the magnitude of  $\phi$ :

$$k_\phi = \|\phi\|_2 \quad (6.11)$$

The maximum value of the ray parameter  $k$  is given by the intersection of equation 6.9 with the linear and nonlinear constraints equations 6.10. Solving the four linear constraints on sample density for  $k$  yields four maximum values:

$$\left. \begin{aligned} k_0 &= \frac{\theta_s^{min} - a_s}{\vec{m}_s} \\ k_1 &= \frac{\theta_s^{max} - a_t}{\vec{m}_s} \\ k_2 &= \frac{\theta_t^{max} - a_t}{\vec{m}_t} \\ k_3 &= \frac{\theta_t^{max} - a_t}{\vec{m}_t} \end{aligned} \right\} \quad (6.12)$$

These bounds determine the potential spatiotemporal change at each element, and constitute a significant constraint on the error vector  $\phi$ . By definition,  $\phi$  is a characterization of redundancy in the underlying signal; nonbandlimited spatial or temporal features such as high contrast edges will produce large magnitude positive error estimates regardless of sample density. The constraints

in equation 6.12 limit the potential change caused by  $\phi$  based on the current sample density at the element in relation to the bounds of the local sample density space.

The ray parameter  $k$  corresponding to  $b^*$  is the minimum positive distance along  $\vec{m}$ :

$$k_{\max} = \max(0, \min(k_0, k_1, k_2, k_3)) \quad (6.13)$$

$$k_{\min} = \min(k_{\max}, k_{\phi}) \quad (6.14)$$

$$b^* = a + k_{\min}\vec{m} \quad (6.15)$$

Equation 6.13 gives the maximum distance along the ray to a linear boundary, and equation 6.14 is the minimum of this distance and the proportional response distance  $k_{\phi}$ .

While equation 6.14 ensures that  $b^*$  adheres to local constraints on the sample density of the element, the density transport stage enforces global constraints like conservation of the total sample density across the image. Conservation is enforced by modulating the total amount of change in sample density at each element. The amount of combined sample density flux between point  $a$  and  $b^*$  is obtained from equation 6.9 and equation 6.7:

$$\Delta\theta^* = \text{Area}(a + k_{\min}\vec{m}) - \text{Area}(a) \quad (6.16)$$

This is the change in the combined spatiotemporal sample rate for solution  $b^*$ , whose response is proportional to the spatiotemporal mixture and magnitude of  $\phi$ .

### 6.2.2 Maximum Change in Density

Local constraints are communicated to the subsequent density transport stage, which enforces global constraints on all of the elements, by specifying a maximum positive modulation amount that the value  $\Delta\theta^*$  may change at each element such that a solution to equation 6.9 exists within constraints 6.12. The transport stage may modulate  $\Delta\theta^*$  by a positive scalar, the change may be increased or decreased, but the direction of the response, the sign of  $\Delta\theta$ , written  $\text{sgn}(\Delta\theta^*)$  may not change.

In terms of Figure 6.1, the transport stage selects a new combined spatiotemporal sample density for the element  $\theta^{n+1} = \theta^n + \Delta\theta^*w$  by adjusting modulation term  $w$  based on global constraints

across the image. After the transport stage determines  $\theta^{n+1}$ , the last stage in the decision control algorithm selects a new point  $b$ ,  $\text{Area}(b) = \theta^{n+1}$  proportional to  $\phi$ .

The modulation term  $w$  must be bounded such that the response improves the sample density, i.e.  $\text{Area}(b)$ , is no further from  $\text{Area}(\Theta^*)$  than  $\text{Area}(a)$ ; the sign of the response must not change,  $\text{sgn}(\text{Area}(b) - \text{Area}(a)) = \text{sgn}(\Delta\theta^*)$ ; and a combined sample density along the curve  $\theta^{n+1}$  occurs within the local constraints 6.12. Two cases must be considered to determine bounds on the modulation term, when the spatial and temporal components of mixture ratio  $\vec{m}$  have the same sign or when they have opposite sign. The first case is illustrated by the point  $a_0$  in Figure 6.3(a) where the function  $\Delta\theta$  increases with  $k$  along the ray and is monotonic, plotted in Figure 6.3(b). In the monotonic case, the maximum change between  $\text{Area}(a)$  and a point along the ray occurs at distance  $k_{\max}$ , given by equation 6.13. The second case, shown in Figure 6.3(c), occurs when the signs of  $\vec{m}$  do not match, and the function  $\Delta\theta$  along the ray has both positive and negative sign at different distances.

In the nonmonotonic case, the maximum combined sample density occurs at the unknown point  $b_{\max}^*$  a distance  $k_{\max}^*$  along the ray:

$$b_{\max}^* = a + \vec{m}k_{\max}^* \quad (6.17)$$

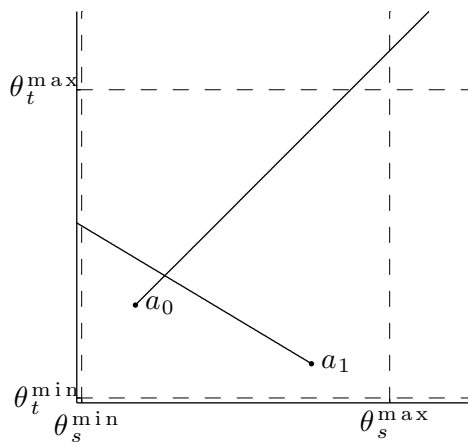
$$\begin{aligned} \text{Area}(b_{\max}^*) &= (a_s + \vec{m}_s k_{\max}^*) (a_t + \vec{m}_t k_{\max}^*) \\ &= a_s a_t + a_s \vec{m}_t k_{\max}^* + a_t \vec{m}_s k_{\max}^* + \vec{m}_s \vec{m}_t (k_{\max}^*)^2 \end{aligned} \quad (6.18)$$

This function is a parabola with maximum value at distance  $k_{\max}^*$  where the derivative along the ray is zero:

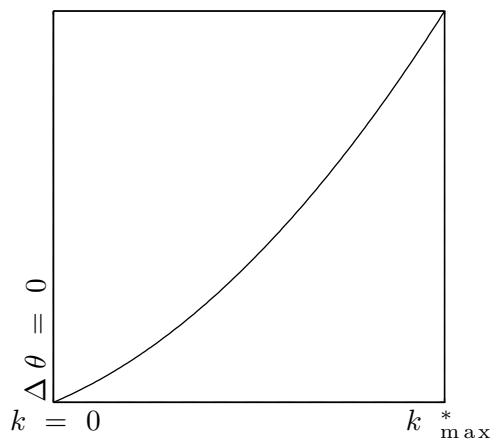
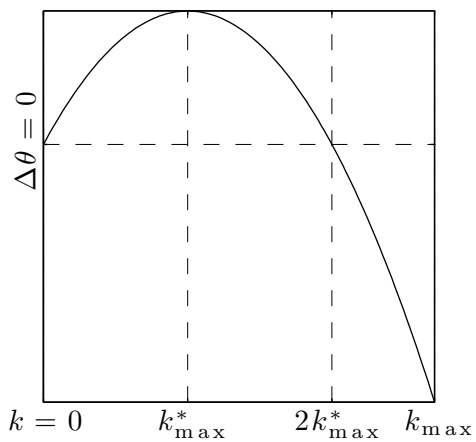
$$\frac{d}{dk} \text{Area}(b_{\max}^*) = a_s \vec{m}_t + a_t \vec{m}_s + 2\vec{m}_s \vec{m}_t k_{\max}^* = 0 \quad (6.19)$$

$$k_{\max}^* = -\frac{a_s \vec{m}_t + a_t \vec{m}_s}{2\vec{m}_s \vec{m}_t} \quad (6.20)$$

At point  $a$ , distance  $k = 0$  along the ray,  $\Delta\theta = 0$  and increases with the ray parameter  $k$  to the extrema of the parabola at  $k_{\max}^*$ , and the sign of  $\Delta\theta$  changes at the distance  $2 \cdot k_{\max}^*$  along the ray, indicated by lines along the horizontal axis in Figure 6.3(c). Point  $b^*$  and  $\Delta\theta^*$  occur at a distance  $k_{\min}$  along the ray; therefore, the distance along the ray to the location of maximum change in the



(a) Densities and mixture ratios.

(b)  $\Delta\theta$  for  $a_0$ .(c)  $\Delta\theta$  for  $a_1$ 

**Figure 6.3.** Nonmonotonic and monotonic cases of  $\Delta\theta_{\max}$ .  $\Delta\theta_{\max}$  is evaluated along the rays starting at points  $a_0$  and  $a_1$ .

same direction  $\text{sgn}(\Delta\theta^*)$  is:

$$k_{\text{sgn}}^* = \begin{cases} k_{\text{max}}^* & \text{if } k_{\text{min}} < 2 \cdot k_{\text{max}}^* \\ k_{\text{max}} & \text{otherwise} \end{cases} \quad (6.21)$$

The actual amount of change from  $\theta^n = \text{Area}(a)$  is:

$$\Delta\theta_{\text{max}} = \text{Area}(a + \vec{m}k_{\text{sgn}}^*) - \text{Area}(a) \quad (6.22)$$

In cases where  $\vec{m}$  is axis aligned, the denominator of equation 6.17 is zero, but the  $\Delta\theta$  function 6.7 is monotonic and  $k_{\text{sgn}}^* = k_{\text{max}}^*$ .

### 6.3 Density Transport

The application of local constraints to the error estimate vector  $\phi$  at each element produces an intermediate solution  $b^*$ , which is a coordinate in the space of spatial and temporal sample densities, and combined spatiotemporal density change  $\Delta\theta^*$  between  $b^*$  and  $a$ , given by equation 6.16. The combination of each  $\Delta\theta^*$  over all elements might violate global constraints like conservation of the total sample density, i.e.  $\sum \Delta\theta^*$  is not constrained. Conservation is enforced by modulating the quantity  $\Delta\theta^*$  at each element such that their sum is within certain bounds; the manipulation must not violate already enforced constraints on the spatial and temporal density.

Unlike the first and last stages of the control decision which balance spatial and temporal sample density individually, shown in Figure 6.2, the transport stage operates on the total combined change, or density flux, at each element. To obtain a conservative solution, the total amount of positive change, i.e. the sum over elements with  $\Delta\theta^* > 0$ , must equal the total amount of negative change, which is the sum over all elements with  $\Delta\theta^* < 0$ . The control decision may be able to improve the sampling rate in a region even if the combined response at an element is modulated to zero by redistributing density between the spatial and temporal dimensions without adding or removing density from the element. The combined change in the positive direction is termed a *positive response* and the negative combined change is termed a *negative response*. Positive responses increase the sample density and improve image fidelity as the dynamic scene changes, negative responses decrease the sample density and exploit redundancy, freeing capacity for sampling work to be performed elsewhere. Figure 6.4(a) shows several image features at a single animation frame

with the corresponding locally constrained spatiotemporal response vector shown in figure 6.4(b), and the quantity  $\Delta\theta^*$  for each element highlighted by grid lines in Figure 6.4(c). The monotonic curve in Figure 6.4(c) shows the non-zero  $\Delta\theta^*$  values for all of the elements in the image, sorted in ascending order. The density transport stage must modulate the curve  $\Delta\theta^*$  to find a similar curve within the local and global constraints on sample density.

The adequacy, or *speed*, of an adaptive response control decision depends on how much the *response magnitude*  $|\Delta\theta^*|$  is increased or decreased to obtain a conservative solution. Rather than the length of the error estimate  $\|\phi\|$ , the magnitude of the response in the density transport stage depends on the total amount of combined density flux at each element. All locally constrained response vectors in Figure 6.4(b) have approximately the same length, i.e.  $\|b^* - a\|$ ; however, the transport response magnitude in Figure 6.4(c) varies considerably. Consider points  $a_1$  and  $a_3$ ; due to the Area function use to compute combined sample density, the response magnitude shown in Figure 6.4(c) of point  $a_3$  is very small compared to  $a_1$  even though the distance between  $a_3$  and  $b_3^*$  in Figure 6.4(b) is almost the same as the distance between  $a_1$  and  $b_1^*$ . In the case of  $a_3$ , nearly the entire change in sample density may be accomplished by redistributing the current sample density Area ( $a_3$ ) between the spatial and temporal dimensions; because the contours of  $a_3$  and  $b_3^*$  on the Area are very similar, the contour of  $b_1^*$  is much greater than  $a_1$ , so a substantial density flux is necessary. In response to image features in the animation,  $a_1$  likely requires a greater response because its location, shown in Figure 6.4(a), is at the leading edge of an object in the animation, the opposite of  $a_3$ , element  $a_2$  occurs on a spatial edge away from temporal change.

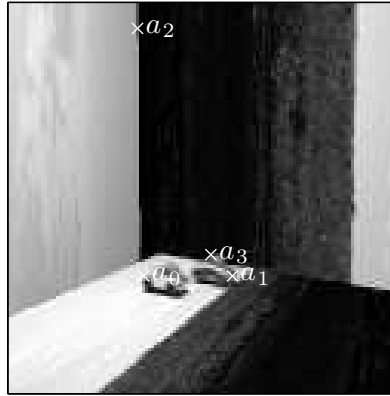
### 6.3.1 Global Constraints

The principal global constraint is conservation of the total sample rate over space and time, such that the total rendering capacity of the adaptive renderer is accounted for at each moment in time. Sample density is either distributed between framelets in the tiling, or accounted for by a surplus of unused density. Sample density is considered to be proportional to rendering work or computational capacity.

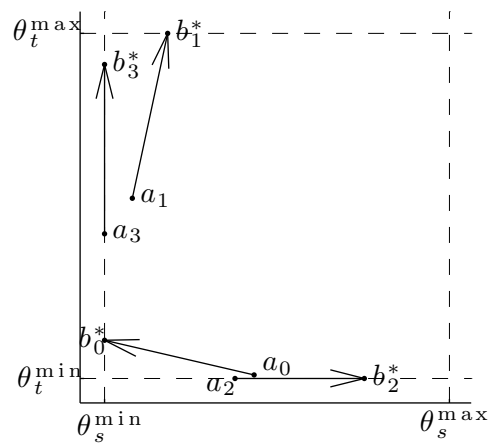
#### 6.3.1.1 Conservation

The transport stage must determine a globally constrained combined sample density change  $\Delta\theta$  at each element, and then a new combined sample density  $\theta^{n+1}$ , within the bounds given by equations 6.16 and 6.22:

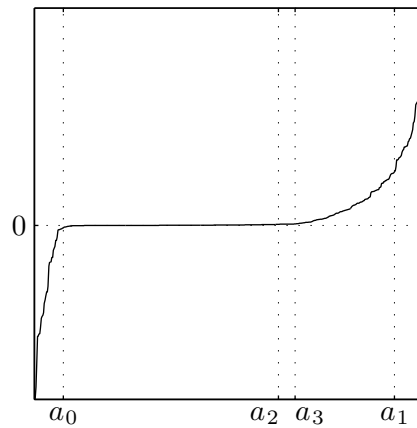
$$0 \leq |\theta^{n+1} - \theta^n| \leq |\Delta\theta_{\max}| \quad (6.23)$$



(a) Animation frame.



(b) Local constraints

(c) Sort ( $\Delta\theta^*$ )

**Figure 6.4.** Animation features with corresponding spatiotemporal change and unconstrained combined response. (a) shows a low resolution nearest neighbor reconstruction of the animation with three points selected. The local spatiotemporal response is shown in (b). (c) shows the quantity of  $\Delta\theta^*$  across all elements with non-zero error in ascending order. Values above zero are positive response, values below are negative response.



Such that the sum of the change in combined sample density over all of the elements is zero, or is less than a certain allowed surplus  $C$ :

$$\sum_m \theta^{n+1}(m) - \sum_m \theta^n(m) = 0 \quad (6.24)$$

$$\sum_m \Delta\theta = 0 \quad (6.25)$$

$$\text{or } 0 \leq \sum_m \Delta\theta \leq C \quad (6.26)$$

If the sum of negative  $\Delta\theta$  is greater than the sum of positive  $\Delta\theta$ , there is a net decrease in the total combined sample density required across the image producing a surplus  $C$ . If a running surplus is employed, subsequent control decisions may allow the total negative and positive to response to differ by at most surplus  $C$  accumulated over time:

$$C^{n+1} = \sum_m \Delta\theta - C^n \quad (6.27)$$

$$C^{n+1} \geq 0 \quad (6.28)$$

### 6.3.1.2 Relational Linear Constraints

The density transport stage may be formulated as a search for the optimal modulated response within two relational constraints, global sample density conservation in equation 6.24, and the maximum change at each element 6.23.

Instead of operating on  $\Delta\theta^*$  directly as a scalar field on the image plane, a vector, or ordered set,  $\varphi$  is defined as a permutation of elements in the field  $\Delta\theta^*$ , e.g. sorted in the ascending order of  $\Delta\theta_{\max}^*$  at each element:

$$\varphi = \text{Sort}_{\Delta\theta_{\max}^*}(\Delta\theta^*) \quad (6.29)$$

The set  $\varphi$  may be divided into two subsets  $A$  and  $B$  based on response direction above and below zero:

$$A = \{\varphi_m : \varphi_m > 0\} \quad (6.30)$$

$$B = \{\varphi_m : \varphi_m < 0\} \quad (6.31)$$

$$Z = (\varphi) \cap (A \cup B) \quad (6.32)$$

The third subset  $Z$  consists of elements with zero sample rate error  $\Delta\theta^* = 0$ . The subset  $A \cup B$ , excluding  $Z$ , is shown in Figure 6.5. All subsets are mutually exclusive and consist of intervals along the curve  $\varphi$ ;  $B$  occurs to the left of the cross in plots 6.5(b)-6.5(f), and  $A$  to the right. Because the error estimate  $\phi$  is only computed for framelets in the most recent rendered batch, only elements within the extent of the batch are potential members of  $A$  and  $B$ .

Equation 6.24 may be written in terms of  $A$  and  $B$  and scalar surplus  $C$ :

$$0 \leq \sum \Delta\theta = \sum A + \sum B \leq C \quad (6.33)$$

$$0 \leq \sum |A| - \sum |B| \leq C$$

$$0 \leq \|A\|_1 - \|B\|_1 \leq C \quad (6.34)$$

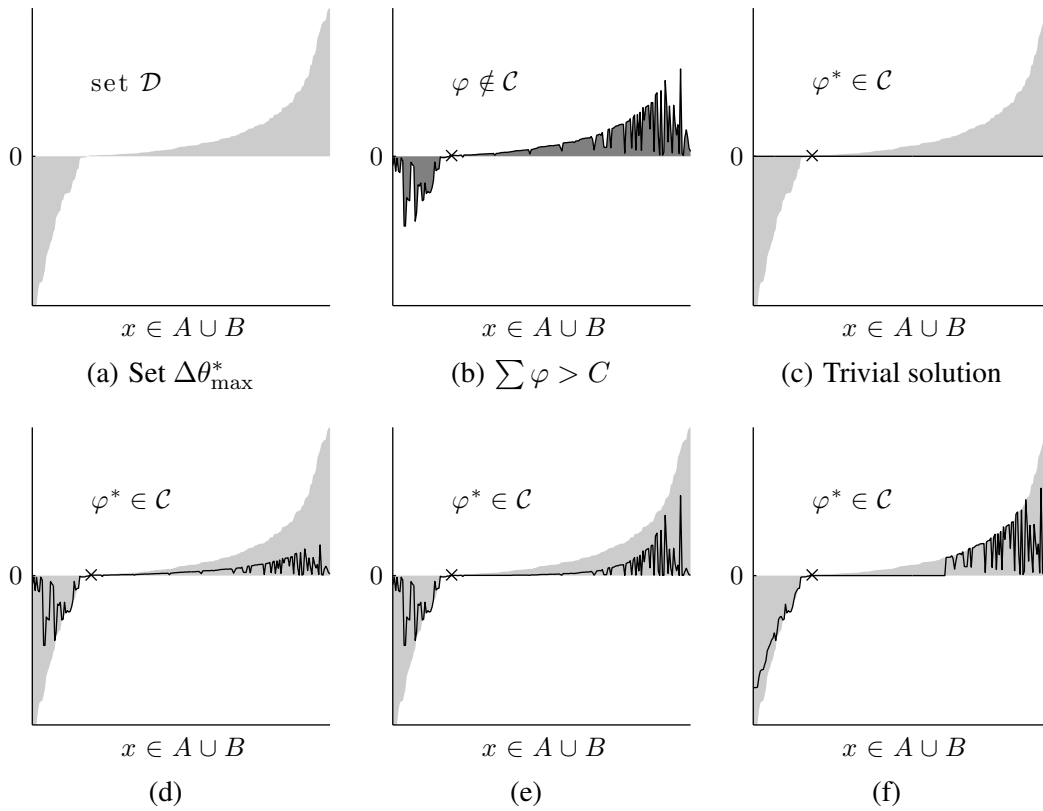
The second sum over  $B$  contains only negative elements, so its positive valued norm must be subtracted. The set  $Z$  contains only elements with zero density change and does not contribute to the conservation equation.

The density transport stage will produce a solution  $\varphi^*$  by modulating  $\varphi$  to balance  $\|A\|_1$  and  $\|B\|_1$ , or stated differently, by searching the set of functions within the constraints defined by equations 6.22 and 6.24.

The  $\Delta\theta_{\max}^*$  constraint bounds the magnitude of individual elements such that they meet local constraints on the sample density and defines a set  $\mathcal{D}$ :

$$\mathcal{D} = \left\{ \varphi^* \in \mathfrak{R}^1 : \bigwedge_m \varphi_m^* \leq \Delta\theta_{\max}^*(m) \right\} \quad (6.35)$$

The conservation equation bounds the total sum over the solution, and comprises a subset of  $\mathcal{D}$



**Figure 6.5.** Set of solutions to the transport equation. Elements with  $|\varphi| \geq 1$  are shown along the  $x$  axis in ascending order of  $\Delta\theta_{\max}^*$ . (a) shows set  $\mathcal{D}$  which enforces local constraints on the sample density at each element. (b) shows the integral of  $\varphi$ ; the cross indicates the zero crossing dividing membership between  $A$  and  $B$ . (c) shows the trivial solution to the transport problem,  $\varphi^* = 0$ . (d)–(e) show other solutions with  $C = 0$ .  $\mathcal{C}$  is the set of all solutions  $\varphi^*$ , four of which are shown. All plots have the same scale.

in which  $\varphi^*$  must reside, the set of functions in  $\mathcal{D}$  which integrate to less than  $C$ :

$$\mathcal{C} = \left\{ \varphi^* \in \mathcal{D} : 0 \leq \sum_m \varphi^* \leq C, \right\} \quad (6.36)$$

The shaded area in Figure 6.5(a) shows the set of functions  $\mathcal{D}$ ; this space must be searched for a function  $\varphi^* \in \mathcal{C}$ . Membership in  $\mathcal{C}$  is a relational constraint based on the integral over  $\varphi^*$ . In the case shown in Figure 6.5(b), the integral over the positive response, to the right of the cross, is much greater than the negative response to the left. In Figure 6.5,  $\varphi$  and  $\varphi^*$  are plotted in the ascending order of  $\Delta\theta_{\max}^*$ , resulting in an appearance that is not smooth. The order in which  $\varphi$  is sorted is not relevant for most operations; however, ordering by ascending  $\Delta\theta^*$  produces a smoother boundary to the set  $\mathcal{D}$  in figures. Certain operations employ partial derivatives along  $\varphi$  to determine the desired shape of  $\varphi^*$ ; in these cases, the set is ordered  $\varphi = \text{Sort}_{\Delta\theta^*}(\Delta\theta^*)$ .

### 6.3.1.3 Example Solutions

Each density transport solution  $\varphi^*$  shown in Figures 6.5(c)-6.5(f) modulates  $\Delta\theta^*$  in different arbitrary ways to obtain a conservative solution  $\varphi^* \in \mathcal{C}$ . The amount that the response at various magnitudes is increased or decreased effects how quickly the adaptive renderer is able to response to changes in the scene. Figure 6.5(c) shows the trivial solution  $\varphi^* = \bar{0}$ , which is the modulation of  $\Delta\theta^*$  to zero across the whole domain such that  $\theta^{n+1} = \theta^n$ . This solution is only necessary if it is not possible to obtain another solution within  $\mathcal{C}$  because  $\varphi$  has only one response direction and no surplus is available, i.e.  $\|B\|_1 = 0$  and  $\|A\|_1 > C$ . If the trivial solution to the density transport stage is employed, the response control decision might still modify the spatial and temporal sample densities by adjusting the mixture between dimensions in the third stage of the decision algorithm.

The remaining examples in Figure 6.5 show arbitrary solutions in the absence of any surplus, i.e.  $\sum \varphi = C = 0$ ; less modulation would be necessary if a surplus were included; however, the size of  $\mathcal{C}$  is still infinite and the search problem no easier. In the first example in Figure 6.5(d),  $\|A\|_1$  is larger than  $\|B\|_1$ , so  $A$  is multiplied by a constant factor to normalize to the sum of  $B$ . The result is that the faster response,  $A$  in this case, is dramatically slowed while  $B$  remains unchanged. This approach will be referred to as *case B* in Section 6.3.2.3. The example in Figure 6.5(e) squares  $A$  before normalizing  $\|A^2\|_1$  to  $\|B\|_1$ ; squaring the initial response  $\varphi$  introduces a bias towards larger magnitude responses during normalization. The last example shown in Figure 6.5(f) modulates both  $A$  and  $B$ ; here, the magnitude of  $B$  is increased towards the boundary of set  $\mathcal{D}$ , and then a

cumulative summation over elements in  $A$ , in descending order, is performed until the sum is equal to the modulated sum of  $B$ ; the remaining elements in  $A$  are set to zero.

These examples maintain similarity between  $\varphi$  and  $\varphi^*$  in different ways and employ operations of varying computational complexity and parallel efficiency; however, without additional criteria, it is not clear which solution is best. Figure 6.6 shows  $|\varphi^*|$  for each example along with a reconstructed image with elements in the 70th percentile of  $|\varphi|$  highlighted, a reference plot containing  $|\varphi|$  at the same scale is shown in Figure 6.7.

Although the fastest example, shown in Figure 6.6(f), slows the large magnitude positive response the least, over half of the elements in  $A$  are modulated to zero response, and many elements in  $B$  have been sped past the magnitude of  $\varphi$ . In  $\varphi^*$ ,  $\|B\|_1 > \|A\|_1$  even though the opposite was true in the input  $\varphi$ . While Figures 6.6(d) and 6.6(e) show a significantly slower response for large magnitude elements, small improvements are still made across the both sets. In these examples  $\|A\|_1$  and  $\|B\|_1$  differ by about one order of magnitude; however, if  $\|A^2\|_1 \gg \|B\|_1$ , normalizing by  $\frac{\|B\|_1}{\|A^2\|_1}$  would undesirably attenuate lower magnitude responses in  $A$  to nearly zero. The proportional response criteria may provide some guidance of how to choose an optimal solution between these examples and all of the other  $\varphi^* \in \mathcal{C}$ .

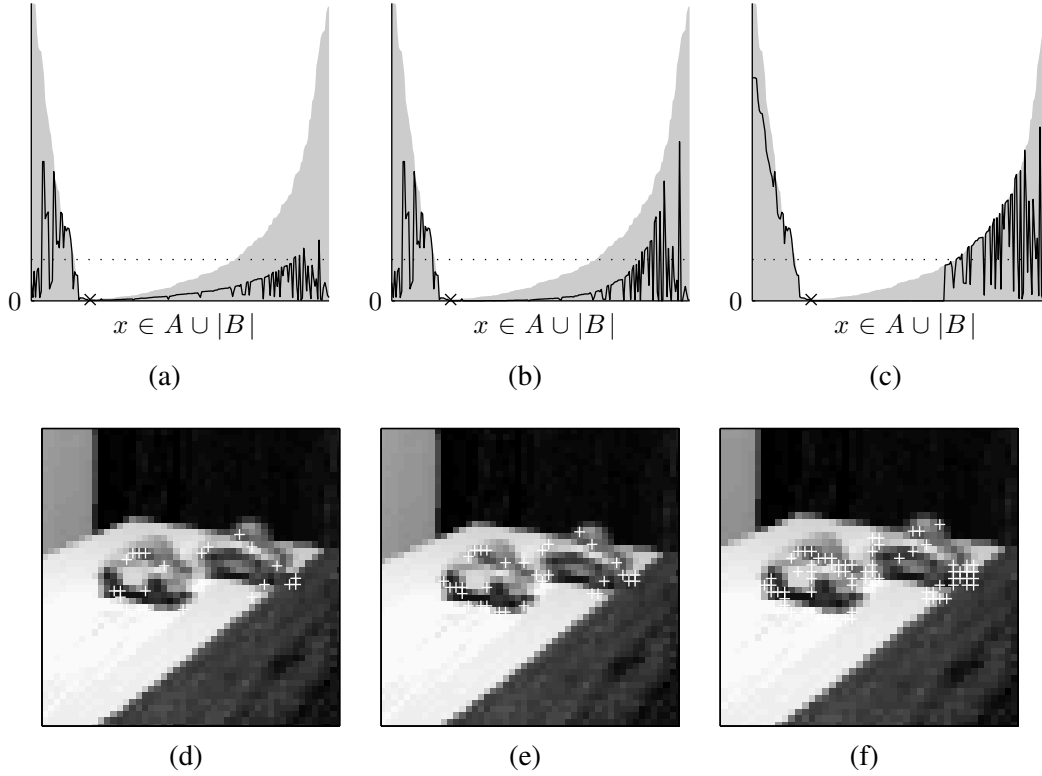
Equation 6.36 produces a set of solutions defined by linear relations, in which the optimal solution is unknown. Determinization is the selection of one solution from the set of possible solutions, i.e. the heuristic for the selection of  $\varphi^*$  from the set  $\mathcal{C}$ . Two approaches are considered; in Section 6.3.2, functional solutions are obtained by converting the relational constraints  $\mathcal{C}$  into equivalences, i.e. solving an equation in terms of and equivalence with a specific point or boundary, e.g.  $\sum \varphi^* = C$ , instead of a relation defining all points inside the boundary  $0 \leq \sum \varphi^* \leq C$ ; and in Section 6.3.3, a heuristic, usually an iterative approximation, is used to select a solution in the interior of the set, i.e. an algorithm leading to a solution  $0 \leq \sum \varphi^* \leq C$ .

### 6.3.2 Two-Dimensional Functional Solutions

Instead of modulating the  $N$  dimensional vector  $\varphi$  to obtain a solution in  $\mathcal{C}$ , consider a simpler choice of applying two uniform weights  $w^A$  and  $w^B$  to the subsets  $A$  and  $B$  of  $\varphi$ . The solution  $\varphi \in \mathcal{C}$  is then given by the weighted combination of  $A$  and  $B$ :

$$\varphi^* = w^A A \cup w^B B \quad (6.37)$$

This type of operation was shown in Figure 6.5(d) where the subset  $A$  is weighted by a constant

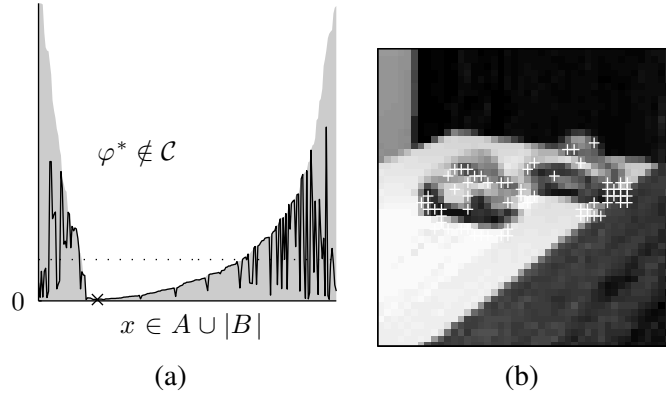


**Figure 6.6.** Implication of density modulation to response. Figures (a)-(c) show the response magnitude of example solutions in  $\mathcal{C}$  from figures 6.5(d)-6.5(f) with a threshold indicated by the dotted line. Figures (d)-(f) show nearest neighbor reconstructed frames with the elements above the threshold highlighted. The first three plots have the same scale.

factor such that  $\|A \cdot w^A\|_1 = \|B\|_1$ , in that case with  $w^B = 1$ . The solution uses a functional form of the equation defining membership in set  $\mathcal{C}$ , the weight  $w^A$  is given by an equivalence with an analytical expression in terms of  $A$ ,  $B$ , and  $C$ , although  $C = 0$  in the examples given in Section 6.3.1.3.

Many functional solutions in terms of these parameters are possible, including the four examples given in Section 6.3.1.3; however, only Figure 6.5(c), the trivial solution, and Figure 6.5(c), mentioned above, are possible in the two-dimensional simplification. The set of other possible solutions may be expressed more concisely than in terms of membership in  $\mathcal{C}$  and  $\mathcal{D}$ , constraints applied to the whole set  $\varphi^*$ , by placing constraints corresponding to set membership on  $w^A$  and  $w^B$ .

First, membership in set  $\mathcal{D}$  is written in terms of  $\Delta\theta^*$  and  $\Delta\theta_{\max}$ , given by equations 6.16 and 6.22, respectively, to produce a bound  $w^{max}$  on the modulation term  $w$ :



**Figure 6.7.** Magnitude of unconstrained response. Figure (a) shows the response magnitude of the input  $\varphi$  and a threshold indicated by the dotted line. Figure (b) shows nearest neighbor reconstruction with elements above the threshold highlighted.

$$w^{\max} = \frac{\Delta\theta_{\max}}{\Delta\theta^*} \quad (6.38)$$

$$0 \leq w \leq w^{\max} \quad (6.39)$$

The bound  $w^{\max}$  is positive regardless of the element's membership in  $A$  or  $B$ , since the signs of  $\Delta\theta^*$  and  $\Delta\theta_{\max}$  match in either case. For each element in the sets  $A$  and  $B$ , the minimum over the maximum scaling factor within the local constraints is given by:

$$w_{\max}^A = \min_{m \in A} (w_m^{\max}) \quad (6.40)$$

$$w_{\max}^B = \min_{m \in B} (w_m^{\max}) \quad (6.41)$$

$$0 < w^A \leq w_{\max}^A \quad (6.42)$$

$$0 < w^B \leq w_{\max}^B \quad (6.43)$$

The conservation equation 6.24 may be simplified to:

$$0 \leq \|w^A \cdot A\|_1 - \|w^B \cdot B\|_1 \leq C \quad (6.44)$$

Then the combination of relations 6.40 and 6.44 is equivalent to membership in set  $\mathcal{C}$ ; however,

the set of solutions  $\mathcal{C}^{2D}$  having only two weights is significantly smaller:

$$\mathcal{C}^{2D} = \{\varphi^* = w^A A \cup w^B B : 0 \leq \|w^A \cdot A\|_1 - \|w^B \cdot B\|_1 \leq C, 0 < w^A \leq w_{\max}^A, 0 < w^B \leq w_{\max}^B\} \quad (6.45)$$

### 6.3.2.1 Schematic Interpretation

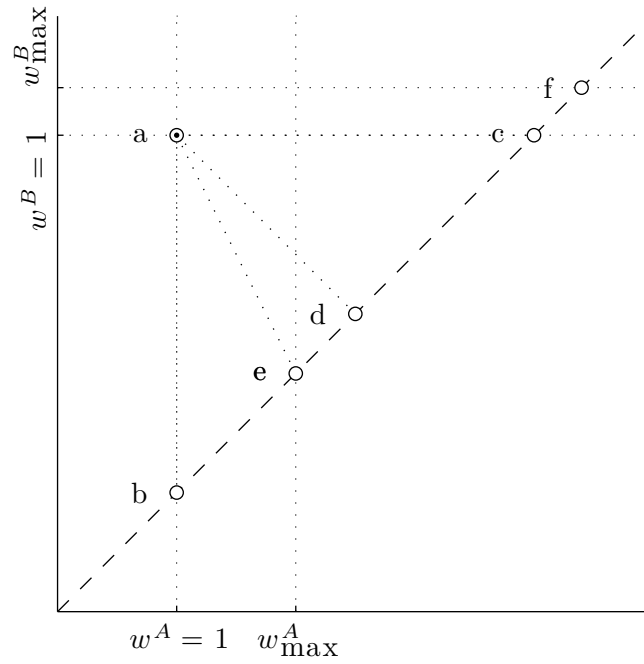
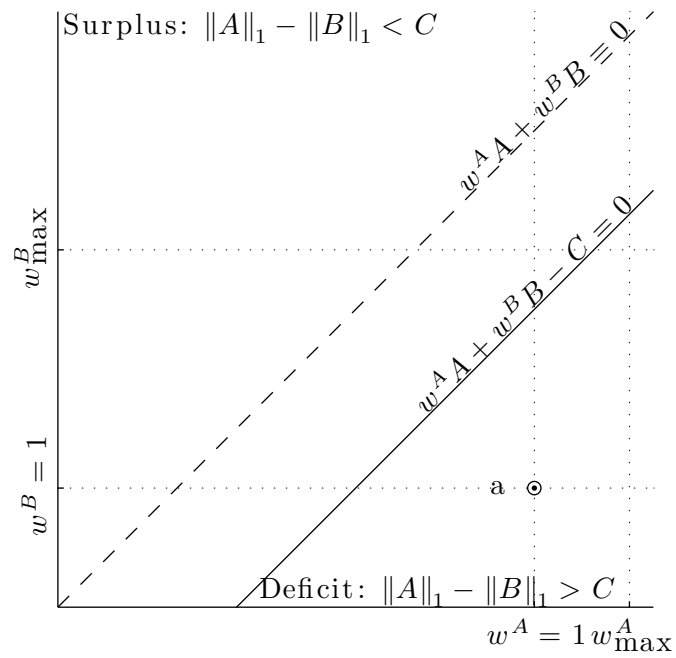
The density transport stage may be interpreted in a two-dimensional space containing points with horizontal coordinates of total positive direction response  $\|A \cdot w^A\|_1$  and vertical coordinates of total negative direction response  $\|B \cdot w^B\|_1$ , shown in Figure 6.8(a). The horizontal and vertical axes are parameterized by the weights  $w^A$  and  $w^B$ , such that the initial configuration  $\varphi = A \cup B$  occurs at  $w^A = w^B = 1$ , point  $a$  in the figure. With the exception of the trivial solution  $w^A = w^B = 0$ , modulation weights are positive; values less than one slow down the magnitude of the response, in the positive and negative directions, respectively, e.g. resulting in  $\|A \cdot w^A\|_1 < \|A\|_1$ , and values greater than one increase the response speed.

In the strictly conservative case, i.e.  $\|w^A \cdot A\|_1 - \|w^B \cdot B\|_1 = C = 0$ , solutions in  $\mathcal{C}^{2D}$  occur on the diagonal line  $\|A \cdot w^A\|_1 - \|B \cdot w^B\|_1 = 0$ , below and to the left of constraints  $w_{\max}^A$  and  $w_{\max}^B$ . With  $C > 0$ , the shaded region in Figure 6.8(b) above the line  $\|w^A \cdot A\|_1 - \|w^B \cdot B\|_1 = C$  contains the set of solutions. Points within this region either produce an additional surplus if they occur above the diagonal where  $\|w^B \cdot B\|_1 > \|w^A \cdot A\|_1$ , or consume some or all of the sample density surplus  $C$ , i.e.  $0 < \|w^B \cdot A\|_1 - \|w^B \cdot B\|_1 \leq C$ .

Response speed of a solution in the density transport stage across the whole image is taken as  $\frac{\|\Delta\theta\|}{\Delta t}$ , which is proportional to the integral over the magnitudes of  $A$  and  $B$ , and in the two parameter formulation, determined by  $w^A$  and  $w^B$ . Solutions in the quadrants surrounding point  $a$  modulate the response speed differently; below and to the left slow the negative and positive response, respectively, while points above and to the right increase the speed of the response. Various combinations of behaviors are shown in Figure 6.8. Since possible solutions must occur in the set  $\mathcal{C}^{2D}$ , bounded by the off-diagonal line  $C$  and the constraints  $w_{\max}^A$  and  $w_{\max}^B$ , some behaviors are not possible in many configurations, e.g. in Figure 6.8(a), no solutions occur in the *faster positive and faster negative* quadrant, above and to the right of  $a$ , while a small number of solutions in that quadrant are possible in Figure 6.8(b) due to the available surplus  $C > 0$ . Given the configuration of the space, solutions with slower response are always possible by modulating the response towards the trivial solution.

The higher dimensional case when  $\varphi$  is modulated by more than two parameters to produce a



(a) Functional solutions  $C = 0$ .(b) Solution set with surplus  $C > 0$ .**Figure 6.8.** Relational constraints on the density transport equation.

solution  $\varphi \in \mathcal{C}$  may still be interpreted geometrically by taking  $w^A$  and  $w^B$  as vectors equal in size to  $A$  and  $B$ , instead of constant scalar weights. The point  $a$  is the same; however, the solution is written  $\langle \|A \cdot w^A\|_1, \|B \cdot w^B\|_1 \rangle$  by taking the norm of the dot product between the response direction vector and the respective weight vector. The solution in  $\mathcal{C}$  is then  $\varphi^* = (w^A \cdot A) \cup (w^B \cdot B)$ . Functional solutions in terms of  $A$  or  $B$  in the higher dimensional case would be written in terms of systems of equations, e.g. linear operators on  $A$  or  $B$  taken as vectors.

### 6.3.2.2 Certainty

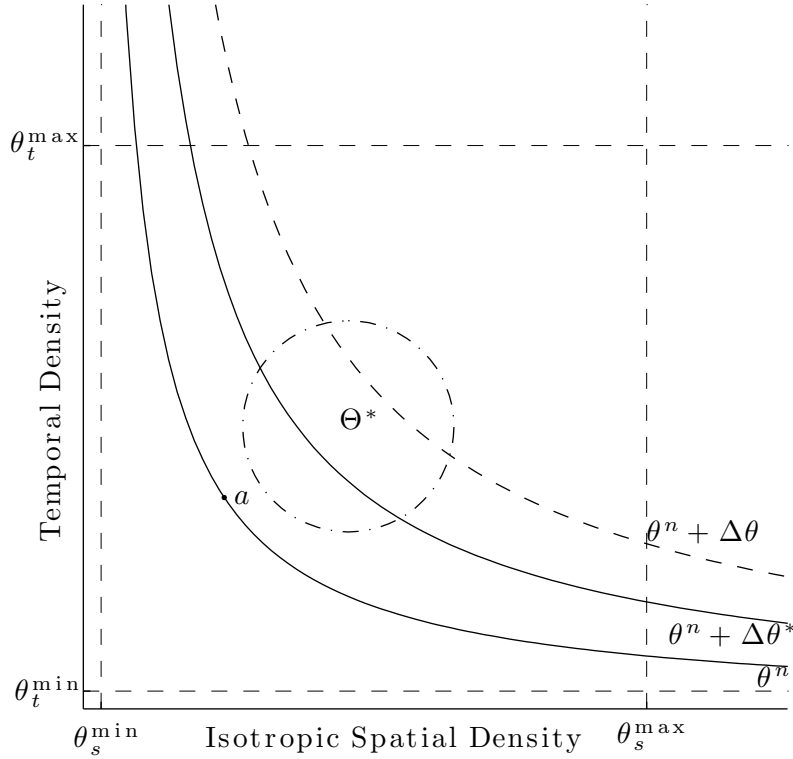
Uncertainty in the adaptive response control decision algorithm stems from the unknown ideal sampling rate  $\Theta^*$ , a dynamic property of the underlying image, which may not be determined from an instantaneous error estimate based on a set of discrete samples at the leading edge of time. The transport stage of the control decision algorithm does not directly effect the movement of the sample density at each element towards  $\Theta^*$ ; instead, the transport stage modulates the speed of the overall response in the positive and negative directions by determining the sample density flux  $\Delta\theta$  at each element, from which individual spatial and temporal response is derived. Since the exact value of  $\Theta^*$  is unknown, shown as a set in Figure 6.9, the amount which  $\Delta\theta^*$  should be modulated to produce  $\theta^{n+1} = \theta^n + \Delta\theta$  is uncertain.

This modulation effects the total change in sample density at each element; optimal modulation would minimize:

$$\min_{\Delta\theta} \|\text{Area}(\Theta^*) - \theta^{n+1}\| \quad (6.46)$$

where  $\theta^{n+1}$  is given by equation 6.6. If the modulation weights  $w^A$  and  $w^B$  are too small, the curve  $\theta^{n+1}$  will be further away from the ideal sample density than was possible, and if the modulation is too great, the curve may overshoot  $\Theta^*$ .

Consider the initial value  $a$  in Figure 6.10(b), within the region defined by the relational constraints 6.44 and 6.40 is an unknown point  $g$  at which the modulation of  $\Delta\theta$  across the image by vectors  $w^{A*}$  and  $w^{B*}$ ,  $g = \langle \|A \cdot w^{A*}\|_1, \|B \cdot w^{B*}\|_1 \rangle$  minimizes equation 6.46. Based on the assumption that  $\Delta\theta$ , as a function of the error estimate vector  $\phi$ , accurately characterizes the signal and provides an accurate response direction to improve fidelity, the ideal modulation vectors  $w^{A*}$  and  $w^{B*}$  are assumed to only increase response magnitude:



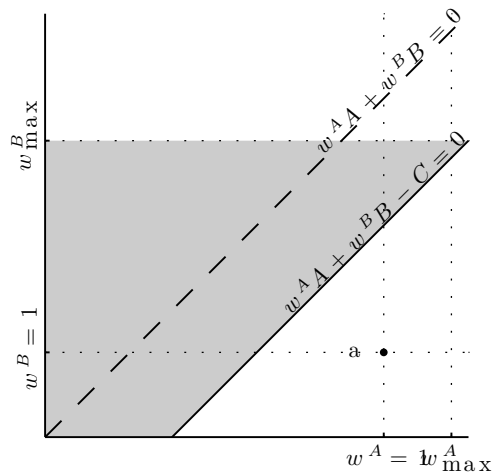
**Figure 6.9.** Uncertainty in the transport stage for a single element. The density transport stage modulates the locally constrained quantity  $\Delta\theta^*$  at each element to produce a change in combined density  $\theta^{n+1} = \theta^n + \Delta\theta$ . The new spatial and temporal sample densities will occur on the curve  $\theta^{n+1}$  shown. The optimal  $\theta^{n+1}$  is uncertain because  $\Theta^*$  is unknown.

$$\|A\|_1 \leq \|A \cdot w^{A*}\|_1 \leq \|A \cdot w_{\max}^A\|_1 \quad (6.47)$$

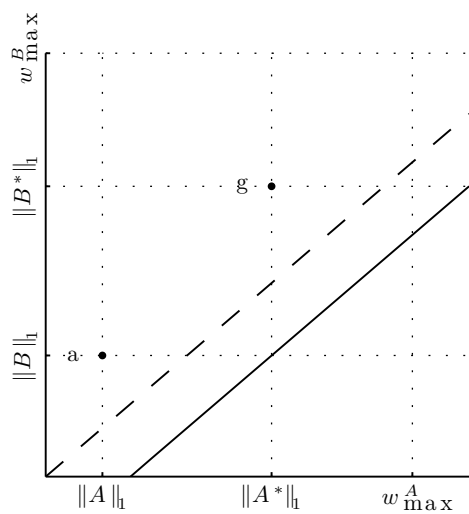
$$\|B\|_1 \leq \|B \cdot w^{B*}\|_1 \leq \|B \cdot w_{\max}^B\|_1 \quad (6.48)$$

This places  $g$  in the quadrant above and to the right of point  $a$ . While the point  $g$  is unknown, its position may be represented using a relational description, comprised of equations 6.44, 6.48, and 6.40, or an analogous constraints for higher dimensional parameter spaces.

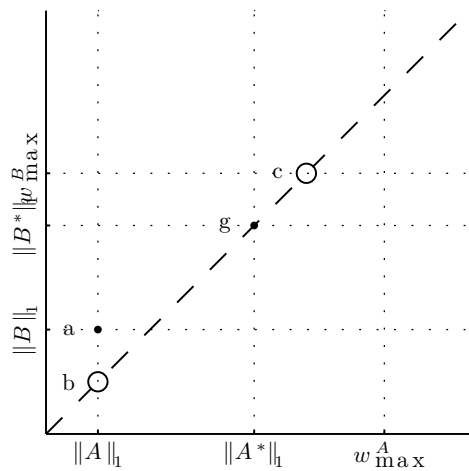
In Figure 6.10(b), the set of possible solutions for  $g$  comprises most of the space given the surplus constraint  $C > 0$ . The optimal solution is uncertain in both cases when  $C$  is zero or greater than zero. Consider that in Figure 6.8(a), a solution may occur anywhere along the main diagonal between the origin and point  $e$ ; however, since the surplus case  $C > 0$  is significantly less constrained, the certainty distribution is more intuitive. In an example like Figure 6.8(a), the distribution between  $A$  and  $B$  is very skewed and there is great subjective certainty that any solution, even between  $b$  and  $e$ , is very poor since the magnitude of  $B$  is very far from the diagonal.



(a) Relational constraints.



(b) Ideal solution  $g$ ,  $C > 0$ .



(c) Ideal solution  $g$ ,  $C = 0$ .

**Figure 6.10.** Example cases for solutions to the transport equation.

### 6.3.2.3 Functional Solutions

The set  $\mathcal{C}^{2D}$ , given by equation 6.45, provides a relational description of the solution to the density transport stage; functional solutions are obtained by finding equations for specific points in the sets in terms of the initial point  $a$  and the constraints. Points  $b$ ,  $c$ ,  $d$ ,  $e$ , and  $f$  in Figure 6.8(a) are examples of possible functional solutions within the relational constraints; only points  $b$  and  $e$  are legitimate solutions within all of the constraints shown in the figure. Each solution occurs at the intersection of pairs of constraints, or at coordinates expressed as a function of point  $a$  and a constraint.

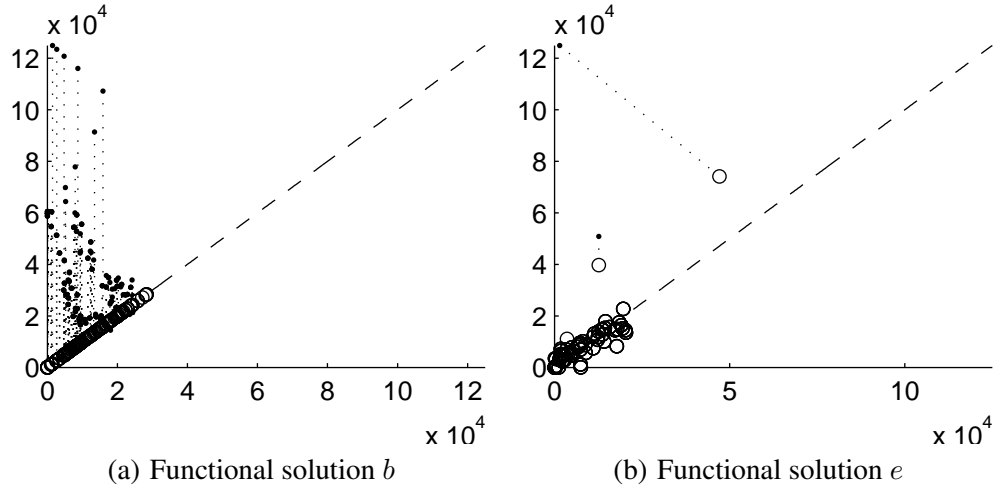
Consider the labeled points in Figure 6.8(a); point  $a$  is the locally constrained initial condition, and solutions within the conservation constraint occur along the main diagonal. Point  $b$  does not change the speed of the lower magnitude response, in the positive  $A$  direction, but slows the larger magnitude negative response. Point  $c$  does the opposite; the larger magnitude response, in the negative direction is unchanged and the positive response is accelerated, violating the  $w_{\max}^A$  in this example. Point  $e$  occurs at the intersection of  $w_{\max}^A$  and  $C = 0$ ; this is the largest magnitude, conservative, positive response. Point  $f$  is the largest magnitude conservative negative response. Point  $d$  increases the positive response and decreases the negative response by the same amount to the conservative diagonal. Of these points, only  $b$  and  $e$  occur within all constraints;  $b$  is the safest, slowest response since it does not increase either magnitude; point  $e$ , or  $f$  if  $w_{\max}^B < w_{\max}^A$ , is less safe and may overshoot the idea solution since one response direction is increased to its maximum magnitude.

Functional solution  $b$  to the density transport equation slows the response direction of greater magnitude to a conservative solution on the main diagonal. Figure 6.11(a) shows the application of solution  $b$  to several dozen frames from the car test scene. Small black dots indicate initial conditions, and white dots indicate solutions. Each white and black dot pair constitutes one control decision.

Functional solution  $b$  slows the greater magnitude response to that of the slower magnitude response, in the example shown  $\|B\|_1 \gg \|A\|_1$ , and so the magnitude of  $B$  is drastically decreased to conservative solutions along the diagonal; no surplus is allowed in this example  $C = 0$ .

If a surplus was allowed in the example shown in Figure 6.11(a), then most of the initial points, which occur above the main diagonal, would be members of  $\mathcal{C}^{2D}$  without modulation; however, if these solutions were accepted, the total sample rate across space and time would decrease dramatically.

Density transport control decisions using solution  $e$  are plotted in Figure 6.11(b) for the same car test scene tracking a running surplus over the sample density. Surplus tracking allows the algorithm



**Figure 6.11.** Scatter plot of functional solutions  $b$  and  $e$  for several hundred frames of the car scene. Unconstrained initial conditions are represented by filled dots and are connected to circles at corresponding solution coordinates. Plots show two parameter transport equation solutions.

to make a small number of large magnitude responses, such as the first response from the upper left corner in the figure, and then use the available surplus to reduce the magnitude of later responses.

Many more functional solutions may be obtained if a surplus is allowed and  $C \geq 0$  since the set of possible solutions is much larger; in the two parameter case, the possible set is a convex polygon instead of a line. All of the function solutions in Figure 6.8(a) along the conservative diagonal where  $C = 0$  are still valid, as are analogous solutions along the off-diagonal  $C$ . It is possible to express functional solutions in the interior of the set  $\mathcal{C}^{2D}$ , above the diagonal  $C$  in the figure, in terms of analytic expressions of the constraints; however, because the difference between these solutions are more finegrained, selection between them may be better performed using a relational or nonfunctional approach. Such an approach, based on shape similarity between  $\varphi$  and  $\varphi^*$ , is described in Section 6.3.3.

#### 6.3.2.4 Limitations

The two-parameter formulation of the transport equation given in equation 6.44 is a projection of the high number of individual control decisions to two dimensions that are convenient for illustration, but the approach overly constrains the magnitude of the overall response. Due to equation 6.40, a single highly constrained element may restrict the entire response in the positive  $A$  direction or negative  $B$  direction. In the average case for the test scene, approximately two thirds of the elements have a non-zero sample density flux, i.e.  $|\Delta\theta^*| > 0$ , and they are members of sets  $A$  and  $B$ , of which only 3 percent have proportional response solutions  $b^*$  within the boundaries of the

local constraints allowing their response magnitude to be increased, i.e.  $\left| \frac{\Delta \theta_{\max}^*}{\Delta \theta^*} \right| > 1$ , the speed of the remaining elements may only be slowed.

The minimum and maximum reduction operators in equation 6.40 cause the 97 percent of highly constrained elements to slow the response of the 3 percent in the interior of the bounds. 3 percent of a frame is a significant portion of the image, equivalent to about a  $128^2$  region at one Mpixel output resolution. This is approximately the size of the leading edge of occluding features in the test scenes.

### 6.3.3 Relational Heuristics

While the definition of set  $\mathcal{C}$  in equation 6.36 provides enough information to determine if a certain response vector is a solution to the transport problem, it provides little guidance about how to obtain potential solutions. Application of the proportional response criteria in equation 6.1, originally used to relate the magnitude and direction of the spatial and temporal response with that of the error estimate, to the change in combined sample density, allows comparison between both the quality of different solutions, and the benefit of different perturbations of an intermediate solution. This comparison allows the transport stage to be formulated more explicitly as a search or gradient descent problem over vectors  $\varphi^*$  in the set  $\mathcal{D}$  towards an optimal solution within the set  $\mathcal{C}$ . While the dimensionality of  $\varphi$  is close to the order of output image resolution, a size which prevents the practical implementation of a descent algorithm, the general formulation as an optimization problem provides a model for realizable approximations to be compared.

After global constraints are applied, the relative magnitude, or *relative shape* of the response, of  $\varphi^*$  should be *similar to*  $\varphi$ . Shape similarity is given by a function  $\text{Shape}(\varphi)$ , to be defined later, and similarity given by the distance between the shape of two vectors. While shape is vaguely defined, it provides a model to determine the optimality of different solutions  $\varphi^* \in \mathcal{C}$  and allows for the formulation of an optimization problem:

$$\text{Distance}(\varphi^*, \varphi) = \|\text{Shape}(\varphi^*) - \text{Shape}(\varphi)\| \quad (6.49)$$

$$\min_{\varphi^* \in \mathcal{C}} \text{Distance}(\varphi^*, \varphi) \quad (6.50)$$

Consider the  $N$  dimensional vector  $\varphi^*$ , in an iterative search for an optimal  $\varphi^* \in \mathcal{C}$ , element  $m$  at iteration  $t$  is written:  $\varphi_m^{*(i)}$ ; the vector  $\varphi^{*(i+1)}$  may be obtained by evaluating the gradient, i.e.  $\frac{\partial}{\partial i} \text{Distance}(\varphi_m^{*(i)}, \varphi)$ , in each direction  $m \in \{1, \dots, N\}$ , to find the largest negative change.

## 6.4 Density Mixture

The density transport stage applies global constraints to the intermediate solution  $b^*$  at each element and produces a new constrained change in sample density  $\Delta\theta$ . The point  $b^*$  determined by the first stage of the control algorithm might not fall on the curve  $\theta^{n+1} = \theta^n + \Delta\theta$ , determined by the density transport stage after modulating  $\Delta\theta^*$  at each element, so a new solution  $b$ , proportional to  $\phi$ , must be computed. The density mixture stage, the final step in the response control decision algorithm, determines the new point  $b$  and may also bias the spatial and temporal sample densities by sliding  $b$  along the curve, in the increasing direction of either the spatial or temporal sample densities, to a final decision  $c$ . If  $\Delta\theta$  is zero, the mixture stage may still effect a response by redistributing sample density at the element without changing the combined amount.

The coordinate  $b$  is found by intersecting a ray from point  $a$  in the direction  $\vec{m}$  with the curve given by  $\theta^{n+1}$ ; following equation 6.9, the ray equation with parameter  $k$  is:

$$b = a + \vec{m}k \quad (6.51)$$

$$b_s = a_s + \vec{m}_s k \quad (6.52)$$

$$b_t = a_t + \vec{m}_t k \quad (6.53)$$

The curve  $\theta^{n+1}$  is given by equation 6.4:

$$\begin{aligned} \theta^{n+1} &= \text{Area}(b) \\ &= b_s b_t \end{aligned} \quad (6.54)$$

The system defined by equations 6.52, 6.53, and 6.54, has up to two roots. In the case that  $\vec{m}$  is not axis aligned, the roots  $k_0$  and  $k_1$  are given by:

$$c = \sqrt{\vec{m}_t^2 a_s^2 + \vec{m}_s^2 a_t^2 + 4\vec{m}_s \vec{m}_t \theta^{n+1} - 2\vec{m}_s \vec{m}_t a_s a_t} \quad (6.55)$$

$$k_0 = -\frac{a_t - \frac{\vec{m}_s a_t - \vec{m}_t a_s + c}{2\vec{m}_s}}{\vec{m}_t} \quad (6.56)$$

$$k_1 = -\frac{a_t - \frac{-\vec{m}_s a_t + \vec{m}_t a_s + c}{2\vec{m}_s}}{\vec{m}_t}$$



The solution  $b$  is computed by equation 6.51 using the minimum positive root. In the axis aligned cases,  $b$  is given by  $b_s = \theta^{n+1}/b_t$  or  $b_t = \theta^{n+1}/b_s$  for the horizontal and vertical directions of  $\vec{m}$ , respectively.

Bounds on the density mixture problem are enforced by the nonlinear constraint  $\Delta\theta_{\max}$  computed in the local constraint stage of the control algorithm. The constraint  $\Delta\theta_{\max}$  is given by equation 6.22 which enforces the linear bounds on spatial and temporal sample density in equations 6.12. The threshold  $\Delta\theta_{\max}$  ensures that a solution along the curve  $\theta^{n+1}$  occurs within the linear bounds. Equation 6.21 avoids the zero root case of equation 6.51, which would occur if the direction of  $\vec{m}$  from initial point  $a$  was such that the ray missed the curve  $\theta^{n+1}$ .

## CHAPTER 7

### EXPERIMENTAL RESULTS

The TSAR algorithm attempts to increase a combination of the spatial fidelity, temporal fidelity, or overall efficiency of a renderer by adding an adaptive response component to the rendering pipeline. This component adjusts the sampling work performed by the renderer and in doing so, consumes computational resources. In order to be beneficial to the overall rendering system, the added overhead of the adaptive response and sampling control mechanism must be compensated for by an improvement in generated imagery. The TSAR algorithm improves the overall rendering system in many cases, especially in terms of temporal fidelity; however, in certain situations, the overhead of adaptive response or adaptive sampling is too great and an improvement is not obtained.

The relationship between algorithmic performance and the graphics scene input, algorithm parameters, and rendering situation distinguish the TSAR approach from the fidelity independent, or free trade off, approaches described in Chapter 2. The adoption of free trade off approaches does not result in a compromise between both the spatial and temporal fidelity of produced imagery. For example, selecting one acceleration structure over another will not effect the spatial fidelity of individual images; however, it may allow for faster refresh and greater temporal fidelity. Since the TSAR algorithm presents a significant trade off between both types of fidelity, it must be evaluated differently than other high performance graphics techniques.

Instead of describing the average temporal refresh rate achieved by the prototype system, the experimental results presented in this chapter describe the relative amount of error produced by the TSAR approach compared to a nonadaptive uniform renderer with equal or greater computational cost. The ratio of error, measured as a difference between the respective rendering output and reference high spatiotemporal resolution imagery, is used to determine the sets of scene input and algorithm parameters where TSAR approach is more suitable. The result of each test run is presented as a distribution of the error ratio across the temporal extent of the test run. Suitable and unsuitable cases are distinguished by the spatial and temporal characteristics of the generated imagery and the cost of the sampling mechanism.

This chapter describes the formulation of suitability, and then the search for it across the space of algorithmic inputs. This consists of a reduction of the large number of algorithm parameters,

the assumptions made while estimating the cost of the adaptive and nonadaptive approaches, and finally, the result of the evaluation.

## 7.1 Suitability

Since the TSAR algorithm poses a fundamental compromise between spatial and temporal fidelity, the tradeoff along with computational cost of the approach must be included in an evaluation to identify suitability. Certain situations are clearly not suitable, e.g. cases where the nonadaptive per-pixel rendering cost is trivial or the imagery does not exhibit appropriate spectral characteristics for sparse sampling and reconstruction. Due to the cost of adaptive sampling and response in these cases, the benefit of the algorithm might be low or the effect might be harmful. The cost of adaptive sampling and response may be unimportant at the other end of the spectrum too, e.g. where the cost of adaptive response would be insignificant compared to that of a high quality noninteractive renderer. The evaluation of the TSAR algorithm presented in this chapter identifies a set of suitable cases between these two extremes where the approach results in an improvement in fidelity compared to a conventional nonadaptive framed rendering approach. This chapter provides an existence proof that situations exist in interactive graphics where the TSAR algorithm is beneficial, and describes the relationship between the suitable cases, algorithm parameters, characteristics of the input imagery, and the overall cost of the approach.

### 7.1.1 Graphics Scene Input and Algorithm Parameters

This evaluation of the TSAR system constitutes a search for situations where the approach is suitable, i.e. cases where the combination of graphics scene input and algorithmic parameters produce imagery at a lower cost computational cost and higher quality than the nonadaptive alternative. The space of graphics scenes, algorithmic parameters, and definition of computational cost and image quality are defined in the following sections.

Analysis of the TSAR algorithm is performed in a hardware-independent manner by counting floating point operations in the algorithm to determine the complexity of each component. The cost constraint, expressed in terms of computational requirements relative to a nonadaptive approach, ensures that both approaches may be implemented on equivalent hardware.

Both computational cost and the quality of output imagery are sensitive to parameters of the algorithm and spectral characteristics of the graphics workload such as the amount of motion and the size and complexity of spatial features. Each suitable case may be described by a certain type of graphics imagery paired with a specific configuration of the TSAR algorithm. The principal challenge in the evaluation is to limit the number of potential cases necessary for testing since the

graphics input to the system and the number of possible algorithmic parameter values is nearly unbounded.

#### **7.1.1.1 Test Scenes**

While the possible number of different graphics scenes is unbounded, the practical size of the space of input to the TSAR prototype is limited by several components of the prototype system: the capabilities of the scene modeling and animation tool, constraints of the scene description format, constraints of the underlying ray tracing system, and the expense of creating test scenes. Since the adaptive rendering algorithm was layered on top of the Manta rendering engine, without any changes made to the underlying software, the material model is limited to textured surfaces with Phong illumination, shadows, and reflections. Both the cost in terms of labor of modeling dynamic scenes and limitations of the scene marshalling software restricted the possible graphics input to rigid body motion of objects, cameras, and lights in the scene.

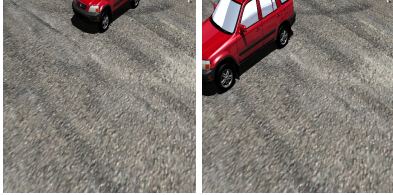


Limitations on the complexity of test scenes and the amount of time necessary for evaluation resulted in a single test world scene containing five different animation sequences, each with different spatial and temporal characteristics. The five different animation sequences are referred to as different test input scenes in this chapter. Representative frames from each test input are shown in Figures 7.1 and 7.2.

Throughout the evaluation process, these five test input scenes produced over two hundred test runs. Each test scene consists of a vehicle moving across a paved surface with textures, reflections, and shadows. The vehicle chassis position, wheel rotation, camera position, and orientation are animated based on time. The animation sequences, consisting of different relative positions of the camera and vehicles, are selected to represent situations encountered in visual simulation or entertainment applications.

The initial vehicle geometry was obtained from the Google 3D Warehouse, attributed to an anonymous author. The geometry was improved to correct surface normals and animated before being added to the test scene. Textures in the test world scene were extracted from landscape photographs.

Two representative frames from input zero are shown in Figure 7.1(a). The test case consists of a fixed camera frame into which a vehicle moves through the upper left of the frame. Only the upper left of the image contains any temporal change; the foreground contains coarse spatial detail. When the vehicle is not visible in the camera frame, there is no temporal change and little difference in spatial detail.

The second input is a chase camera position where the viewer is positioned a fixed distance from the vehicle and moves with it through the scene. Although the relative location of the vehicle



Example Frames	Description
	Fixed camera. Localized temporal, uniform spatial. Transient detail.
(a) Test input scene 0	
	Chase camera. Consistent spectral characteristics.
(b) Test input scene 1	
	Fly-by camera. Transient spatial and temporal features.
(c) Test input scene 2	

**Figure 7.1.** Representative frames test input scenes produced by a nonadaptive framed renderer.

in the frame does not change, the reflections on the vehicle, the wheels, and the background change as the vehicle moves through the scene. Two representative frames from input 1 are shown in Figure 7.1(b). The spectral characteristics of different spatial regions of the imagery do not change over time because the camera tracks the vehicle from a fixed distance. Spatial detail in the scene varies between coarse and fine features across the foreground and background.

The third input is a fly-by camera scenario, shown in Figure 7.1(c). The camera is located at a fixed position but pivots to follow the vehicle as it moves past. Most of the image changes from one frame to another as the ground and background rotate around the camera, although the spectral characteristics of the image are consistent throughout the sequence since the vehicle always occupies the center of the frame. This results in a test case with consistent spectral features, i.e. the horizon and the texture in the foreground are always at approximately the same position, but changing spatial detail since the shape of the horizon and region of the texture within the frame change.

The fourth input is a fixed camera frame where the vehicle moves towards the camera but the

Example Frames	Description
 <p data-bbox="488 443 740 474">(a) Test input scene 3</p>	<p data-bbox="824 306 1214 407">Dolly camera. Largely static spatial detail, enlarging region of temporal change.</p>
 <p data-bbox="488 743 740 774">(b) Test input scene 4</p>	<p data-bbox="824 642 1214 701">Fixed camera. Rapid transient features.</p>

**Figure 7.2.** Representative frames test input scenes produced by a nonadaptive framed renderer for the remaining two scenes.

orientation of the camera frame remains unchanged. Since only the vehicle is moving, most of the image remains unchanged. Input 3 is shown in Figure 7.2(a). The vertical edge apparent in the background of the images is a texture mapping artifact.

The last input case consists of a fixed camera frame across which the vehicle moves very quickly. Input 4 is shown in Figure 7.2(b). This is a difficult case for adaptive response because the speed of the movement creates rapid occlusions and disocclusions of regions with markedly different spectral characteristics and the response component of the pipeline is only able to observe information up to the leading edge of time.

### 7.1.1.2 Parameters

Besides the graphics scene input, the space of possible algorithmic parameters is particularly large because the TSAR prototype system allows one hundred arguments to be set at runtime, although nearly all have empirically set constant values. The set of possible configuration parameters must be significantly culled to produce a tractable number which may be tested with the input scenes and compared to the nonadaptive renderer.

The search over the parameter space for algorithmic configurations to test suitability is conducted using a four step process to constrain the space of possible parameters and input. First, a cost model is formulated in terms of a number of independent parameters to reason about the relative computational complexity of different components of the algorithm. Next, the cost model is

projected into a low dimensional space which may be empirically mapped to algorithm parameters. Third, a lower cost subspace is selected from the simplified cost model based on the relative cost of the nonadaptive approach. Finally, the cost model dimensions are transformed into the space of system parameters, and the lower cost subspace is sampled by executing test runs of the system with each combination of parameters. The aggregate quality of each run is compared against a competitive nonadaptive approach to identify instances where the quality of the TSAR approach is greater.

The hardware-independent cost model employed for evaluation is described in Section 7.1.2 and quantitative evaluation of the sample points identified within the relative cost constraints and the characteristics of the test scene is described in Section 7.2.

### 7.1.2 Cost Model

The TSAR prototype used for evaluation provides interactive performance, depending on the type and amount of information recorded during experimentation. However, a wider variety of experiments may be performed if the system is run in simulated time, where the upper limit on the pace of temporal change in the imagery is not dictated by the constraints of computer hardware. In this way, the prototype may be seen to provide a fast simulation of a real-time rendering system. Several factors contribute to the need to employ simulated time: instrumentation of the pipeline, in the form of software event counters, such as the size of the tiling or the number of sample density scalars changed by the adaptive response stage, require pipeline components to be executed in a serial emulation mode on the host processor instead of in parallel on the graphics device processor; recording the state of the error and density fields, as well as the new framelet sample batch and the output reconstructed frame, require data to be copied each pipeline cycle from the device to the host memory and eventually to persistent storage on a disk; and limitations of the scene modeling and loading pipeline result in inefficient rendering by the ray tracing engine. The combination of all of the instrumentation, profiling, and recording overhead prohibits quantitative real time analysis in every test case. Nevertheless, the difference between real time and the temporal rate achievable by the prototype is less than an order of magnitude.

Since evaluation of the TSAR prototype must be performed on different hardware than the intended real time implementation, a model of computational cost and complexity in terms of aggregate operations is used, instead of a hardware-specific cost analysis. The cost model is formulated by manually counting the total number of floating point operations that would be performed by the most significant stages of the pipeline in a serial and hardware-independent manner. Several simplifications to the operation count are made: only floating point operations are counted, the execution path with the greater number of operations was counted in instances where conditionals

caused divergence, also transcendental floating point operations such as *pow*, *sqrt*, *log*, etc. were counted as a single operation; and generalizations were made in particularly complicated divergent cases, such as the number of traversal steps of a spatial acceleration structure during ray tracing, or the amount of overdraw incurred during reconstruction.

The stages of the TSAR pipeline included in the cost model are: framelet rendering by ray tracing, error estimation, sample density update, framelet tiling rebuild, and reconstruction. In each stage, the total number of operations per-entry, i.e. in aggregate by all threads, is counted, in addition to a per-output pixel cost, which is the entry cost in floating point operations divided by the number of pixels, i.e. output frame size, reconstructed in each pipeline cycle. The cost model for the nonadaptive renderer consists of only the per-pixel ray tracing cost.

### 7.1.2.1 Model Parameters

The number of operations performed in each pipeline stage is a function of several parameters, some of which are constant in the implementation and others which may be specified at execution time, or as properties of the scene. The independent parameters in each stage along with the default values assumed for cost analysis are shown in Table 7.1; all but the last two rows are possible command line parameters. The default values were determined empirically while developing the prototype.

Expressions for the entry cost of each stage per pipeline cycle are shown in the following set of equations, in terms of the individual components of each stage. Constant integers indicate approximate operation counts in the implementation.

**Table 7.1.** Cost model independent parameters

Parameter	Default value	Description
<i>batch_size</i>	128	Framelets rendered per pipeline cycle.
<i>sdf_size</i>	128 <sup>2</sup>	Resolution of the $\theta$ and $\phi$ scalar fields.
<i>framelet_size</i>	16 <sup>2</sup>	Number of samples in each framelet.
<i>recon_res</i>	512 <sup>2</sup>	Reconstruction output resolution in pixels.
<i>cache_size</i>	2048	Number of framelets stored in the cache.
<i>recon_age_support</i>	128	Width of the framelet age smoothing operator.
<i>recon_spa_support</i>	3 <sup>2</sup>	Size of the reconstruction spatial filter.
<i>trace_segments</i>	5	Number of ray segments per pixel.
<i>tiling_size</i>	512	Number of framelets in the tiling.
<i>dwt_coeffs</i>	448	Number of coefficients contributing to the LLN measure.
<i>tree_size</i>	1024	Number of spatial structure tree nodes.



$$\begin{aligned}
\textit{selection\_per\_entry} &= (\textit{tiling\_size} * 1 + \textit{batch\_size} * 2) \\
\textit{error\_per\_spa\_conv} &= 16 * \textit{dwt\_coeffs} \\
\textit{error\_per\_tem\_conv} &= 37 * \textit{dwt\_coeffs} \\
\textit{error\_per\_framelet} &= \textit{per\_spa\_conv} + \textit{per\_tem\_conv} \\
\textit{error\_per\_entry} &= \textit{batch\_size} * \textit{per\_framelet} \\
\textit{update\_per\_compute\_dtheta} &= \textit{sdf\_size} * 49 \\
\textit{update\_per\_scale\_error} &= 19 + 96 * \textit{sdf\_size} \\
\textit{update\_per\_entry} &= \textit{per\_compute\_dtheta} + \textit{per\_scale\_error} \\
\textit{retiling\_per\_sum\_tables} &= \textit{sdf\_size} * 2 \\
\textit{retiling\_per\_tiling\_build} &= 2 * \textit{tiling\_size} * \log_2(\textit{tiling\_size}) \\
\textit{retiling\_per\_entry} &= \textit{per\_sum\_tables} + \textit{per\_tiling\_build} \\
\textit{recon\_per\_age\_buffer} &= \textit{framelet\_size} * \textit{cache\_size} \\
\textit{recon\_per\_age\_smooth} &= \textit{recon\_res} * \textit{recon\_age\_support} * 4 \\
\textit{recon\_per\_framelet\_filter} &= (16 * \textit{recon\_spa\_support} + 6) * \\
&\quad \textit{avg\_recon\_overdraw} * \textit{recon\_res} \\
\textit{recon\_per\_normalize} &= \textit{recon\_res} \\
\textit{recon\_per\_entry} &= \textit{per\_age\_buffer} + \textit{per\_age\_smooth} + \\
&\quad \textit{per\_framelet\_filter} + \textit{per\_normalize} \\
\textit{tracing\_per\_bvh\_node} &= 16 \\
\textit{tracing\_per\_triangle} &= 64 \\
\textit{tracing\_per\_shader} &= 96 \\
\textit{tracing\_per\_traversal} &= \log_2(\textit{tree\_size}) * \\
&\quad \textit{per\_bvh\_node} * \textit{per\_triangle} \\
\textit{tracing\_per\_ray\_tree} &= \textit{trace\_segments} + \\
&\quad (\textit{per\_traversal} + \textit{per\_shader}) \\
\textit{tracing\_per\_entry} &= \textit{batch\_size} * \\
&\quad \textit{framelet\_size} * \textit{per\_ray\_tree} \\
\textit{naive\_per\_pixel} &= \textit{tracing\_per\_ray\_tree}
\end{aligned}$$

The per-entry and per-pixel cost vary widely between the different stages. Reconstruction, which is a multistage image-based algorithm, contains four subcomponents corresponding to the approximate number of floating point operations in each pass. Error analysis has the greatest per-entry cost, but it is parameterized on the batch size and framelet resolution. These quantities are an order of magnitude smaller than their divisor, the output pixel resolution, so the per-pixel cost is much lower. The output resolution is the chief independent parameter of the reconstruction stage and the parameter divides out of all but the first subcomponent of the stage, so the computational cost is not effectively amortized across image.

### 7.1.2.2 Simplified Model

While less than the hundred possible command line parameters, the number of independent parameters listed in Table 7.1 is still too great for a tractable search for suitable configurations. To limit the number of potential parameter combinations, the cost model is projected from a eleven-dimensional space to a three-dimensional space.

The three parameters, batch size, number of ray traced segments, and output resolution size, are selected as the independent parameters of a simplified cost model which is obtained by fixing the other parameters to their default values. The batch size parameter controls the number of framelets rendered per pipeline cycle in the adaptive renderer which is equivalent to the number of output image pixels in a framed renderer. The number of ray traced segments determines the cost of rendering each image pixel or framelet sample; one segment is counted for each ray traced to color the pixel, e.g. a pixel containing the reflection of a diffuse surface with a single light source would require four segments: a primary ray cast from the camera, a reflection ray from the initial intersection, and two shadow rays, one from the primary and one from the secondary intersection points. The model assumes that each ray segment has equal cost. The last parameter of the simplified model is the resolution of the output imagery; in the adaptive renderer this is the resolution at which reconstruction is performed and in the nonadaptive renderer it is simply the number of pixels colored per frame. These three independent parameters are most likely to be dictated by the requirements of a graphics situation, e.g. the degree of indirect illumination, or implementation hardware limitation.

Cost comparison between different stages of the TSAR pipeline and between the adaptive approach and a nonadaptive framed rendering approach is performed in terms of the difference between producing a single complete image from one cycle through the rendering pipeline. In the nonadaptive pipeline, the size of the output image is simply the number of pixels ray traced in each frame. In the TSAR pipeline, the size of the output is image is the resolution at which reconstruction is performed.

The output imagery resolution in pixels versus the cost per-pixel is shown in Figure 7.3. The other independent parameters of the cost model assume default values. The solid thick line is the sum of the cost of all stages in the adaptive pipeline and the dashed thick line is the cost of rendering an image at the same output resolution using a nonadaptive framed rendering algorithm. The vertical axis of the plot is on a log scale.

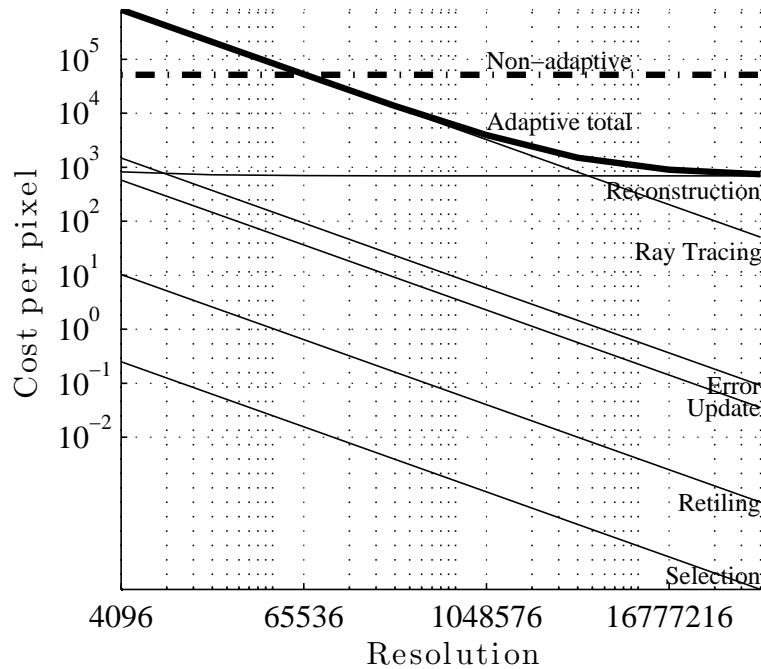
Since the cost of reconstruction is dominated by operations performed at output pixel resolution, it is nearly constant as resolution increases. At higher output resolutions, the cost of reconstruction dominates that of other stages in the pipeline. As the cost of the other pipeline stages approach zero, the total adaptive cost approaches that of the reconstruction stage, which is more than an order of magnitude less than the cost of the nonadaptive renderer due to the default batch size which is listed in Table 7.1 and fixed in this figure. In cases where the output resolution is less than  $256^2$ , the cost of ray tracing framelet samples alone is greater than the cost of framed rendering.

The batch size parameter of the TSAR algorithm controls the number of framelets which are selected for rendering from the tiling in each pipeline cycle. Based on the default parameter value in Table 7.1, each framelet contains 256 image samples. The total number of image samples rendered in each pipeline cycle is this number multiplied by the batch size. Batch size versus the cost per-pixel of each pipeline stage is plotted in Figure 7.4. At batch size 1024, the same number of ray traced image samples is the same in both the adaptive and nonadaptive renderers; the difference between the nonadaptive and adaptive cost, which is greater, shows the overhead of the adaptive approach. Just to the left of this batch size, the cost of both approaches is the same. The reconstruction cost function increases very slowly with batch size while the update and retiling functions are constant and independent.

The number of ray segments traced per image pixel versus per-pixel cost is shown in Figure 7.5. This independent parameter is proportional to the amount of nonlocal shading effects used to synthesize the image. Common interactive ray traced scenes may employ between two and sixteen ray segments per image sample, while high quality indirect illumination may require significantly more. The zero ray segment point along the left side of the plot indicates the cost model behavior when the cost of rendering a sample is trivial or the sampling mechanism is free. As the complexity of each image sample increases, the difference between the adaptive and nonadaptive approaches converges to the ratio between the batch size and the output resolution, in this case, one quarter the number of image samples, while the cost of the other pipeline stages is constant.

### 7.1.3 Cost Constraint

The suitability definition specifies that the cost of the adaptive approach must be less than or equal to the cost of the competing nonadaptive approach, and separately, that the quality must

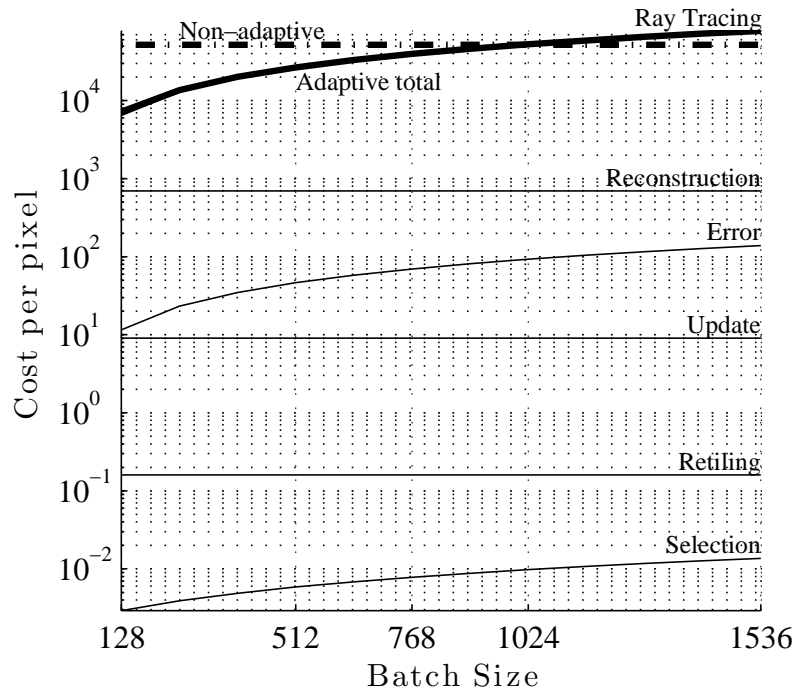


**Figure 7.3.** Cost per pixel versus output resolution. The solid thick line indicates the total cost of adaptive rendering, the dashed thick line indicates the cost of a nonadaptive framed renderer. Each thin line indicates the cost of a stage in the TSAR pipeline.

be better. This cost constraint significantly limits the space of possible parameter values. The computational cost of both approaches may be expressed as functions of the simplified model, and the constraint as a relationship between either function. The simplified cost model of the adaptive algorithm may be expressed as a three-dimensional function with independent parameters batch size, number of ray segments, and output resolution. The cost model of the nonadaptive framed renderer is a function with two varying parameters, the number of ray segments and the output resolution. The cost constraint is satisfied when the two functions are equal or the nonadaptive per-pixel cost is greater than the adaptive per-pixel cost.

The set of parameters over which the cost of the algorithms is the same is shown in Figure 7.6 for intervals of parameters consistent with the capabilities of the TSAR prototype. The isosurface shown is drawn where the ratio is equal to one. The region of parameter space above the isosurface satisfies the cost constraint and the region below does not. Along the right edge of the figure at the zero ray segments coordinate, there is no region of the parameter space above the isosurface. Likewise, before batch size 256, nearly all of the parameter space is within the constraint.

The space above the equal cost isosurface must be sampled to identify parameter combinations within the suitability constraints; these may be tested. Several potential coordinates are marked by boxes in Figure 7.6. These occur at ray segment coordinate four which is the approximate number



**Figure 7.4.** Cost per pixel versus batch size. The solid thick line indicates the total cost of adaptive rendering, the dashed thick line indicates the cost of a nonadaptive framed renderer. Each thin line indicates the cost of a stage in the TSAR pipeline. The total number of image samples rendered per pipeline cycle is equal to 256 times the batch size. The total cost of the adaptive approach exceeds the nonadaptive approach along the right edge of the figure.

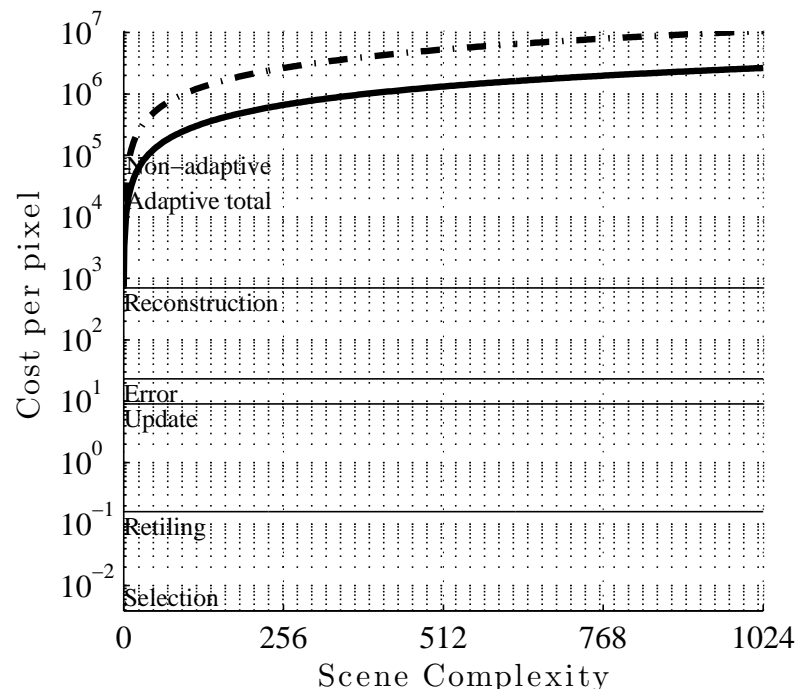
of ray segments per pixel in the test input scenes.

## 7.2 Quantitative Evaluation

The equal cost isosurface shown in Figure 7.6 constrains the space of possible parameter combinations to be considered in the search for suitable configurations. The parameter space above the isosurface in the figure contains parameter combinations which are less expensive than a nonadaptive framed renderer, but the cost model does not provide any information about the relative quality of imagery produced by these parameters, compared to nonadaptive imagery. This section formulates a quality analysis for quantitative comparison of imagery produced by parameters combinations within constraints of the cost model.

### 7.2.1 Measuring Quantitative Quality

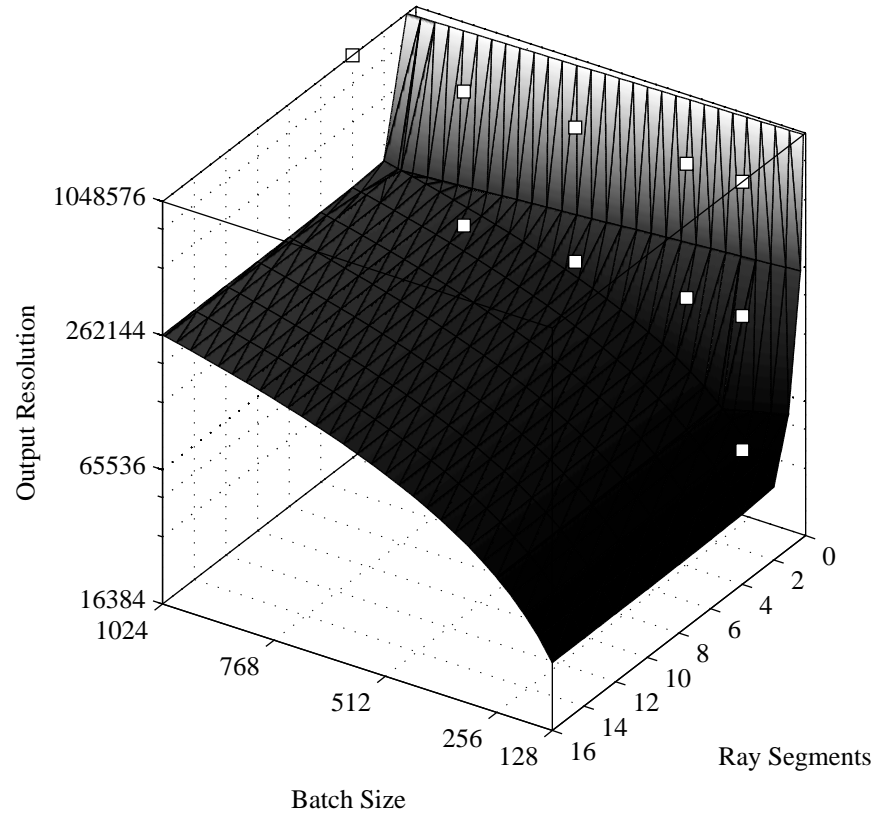
Quantitative approaches to measure the perceptual quality of still images and video imagery may be divided into three categories: measures based on pixel error or difference between images; approaches which estimate the similarity between features or structures in the images, the overall



**Figure 7.5.** Cost per pixel versus  $\theta$  resolution. The solid thick line indicates the total cost of adaptive rendering, the dashed thick line indicates the cost of a nonadaptive framed renderer. Each thin line indicates the cost of a stage in the TSAR pipeline. Since the cost of tile selection does not change with the resolution of the error and sample density fields, it appears along the bottom edge of the figure.

perceptual likeness; and approaches which estimate the prevalence of an anticipated type of artifacts in the imagery. The suitability of each approach depends on the definition of image quality in the application at hand. In the field of image compression, expected type of quality degradation is well known and measurement of specific artifacts is possible, e.g JPEG quantization artifacts assessed by Süsstrunk et al. [62]. The second category is suitable in instances where quality is defined in terms of the overall perceptual similarity of a synthetic image and an actual object. Approaches in this category include Wang et al. [69] and Winkler et al. [72], and are specifically motivated by shortcomings of error or difference-based quality assessment, such as a sensitivity to minor translations or high frequency aliasing artifacts. The definition of quality in terms of visual similarity to an actual object seems very compatible with the definition of fidelity given in Section 1.3.1, but the relative insensitivity to minor changes such as translation decreases the response of these approaches to common types of temporal change, such as objects moving across the image.

The ideal quantitative quality measure for the evaluation of the TSAR algorithm would judge the adequacy of the spatial and temporal sample rates for reconstruction of imagery. While a pixel



**Figure 7.6.** Equal cost per-pixel isosurface. Scalar volume is the ratio between the per-pixel cost of the nonadaptive and adaptive approaches over dimensions framelet batch size, ray segments per-pixel, and output pixel resolution. Isosurface indicates equal cost, i.e. ratio equal to one, of the adaptive and nonadaptive approaches. The adaptive approach is less expensive above the surface and more expensive below. Marked points above the surface indicate the parameter values of test cases.

difference-based quality measure, like mean squared error, is not able to distinguish between various types of image differences and may be more sensitive than the human visual system, as illustrated by Wang et al. [68], it does provide a reliable distance measure between two images and is sensitive to differences due to small temporal changes. Earlier approaches that are more sophisticated and computationally expensive than pixel difference measurements, as summarized by the Video Quality Experts Group [54], produce statistically equivalent results when compared to subjective quality measurements. Quantitative analysis based on frame to frame mean squared error between test run and reference imagery at a high temporal resolution is used in the evaluation of the TSAR algorithm principally due to its computational efficiency and sensitivity to differences caused by temporal sample rate artifacts.

### 7.2.2 Reconstruction Artifacts

Two principal factors contribute to differences between test run and reference imagery, differences in the temporal refresh rate or age of images, or differences in the spatial sampling and reconstruction approaches used during rendering. Differences in the rate and time of temporal refreshes cause motion in the test run imagery to proceed at a slower rate than in the reference. Substantial differences in frame rate may introduce sharp temporal features, artifacts where animated motion is aliased. Spatial artifacts decrease the fidelity of test run imagery when the size and detail of spatial features reconstructed by the test run differ from analogous spatial features in the test run imagery.

In general, the TSAR algorithm decreases the spatial or temporal sample rates used to render the test run, and most of the artifacts which reduce fidelity are caused by inadequate spatially and temporal up sampling, which is unable to reconstruct enough detail compared to the reference imagery. Spatial up sampling artifacts produce smooth regions of individual frames which lack high frequency detail; other spatial artifacts may occur at the boundary between framelets where the sample rate changes. Temporal up sampling artifacts produce regions of frames containing older sample information; boundaries between older and newer regions of the image are smoothed through the addition of motion blur.

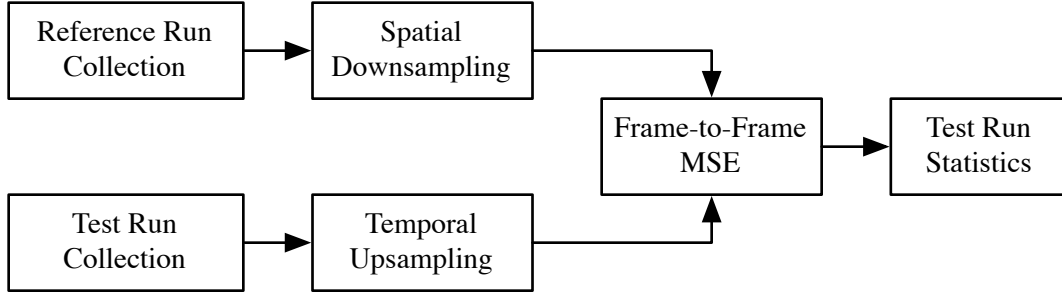
Test run imagery for evaluation of the TSAR algorithm is compared to high resolution reference imagery produced by rendering each test input scene at a high spatial and temporal sampling rate. Test run imagery is collected separately from the adaptive renderer and the nonadaptive renderer and individually compared to a single high resolution reference. The result of this comparison for each test run produces two quality measures from which the suitability of the adaptive test run may be determined.

### 7.2.3 Comparison Pipeline

The quantitative quality comparison pipeline consists of four stages: test run collection, resampling, error measurement, and result aggregation, which are illustrated in Figure 7.7.

First, individual frames are recorded by the renderer in the test run collection stage. The TSAR prototype may be configured to download fully reconstructed frames from the front buffer of the graphics processor during execution. This overhead is one factor that contributes to the requirement that analysis be based on simulated time test runs. In both the TSAR prototype and the nonadaptive framed renderer, the animation time stamp, either a real-time or simulated time value, is recorded when the front and back buffers are swapped by the display hardware. The per-frame timestamp allows different test runs to be temporally resampled for frame to frame comparison either qualitatively with each other or quantitatively with a high resolution reference sequence.





**Figure 7.7.** Quantitative comparison pipeline.

The second stage of the comparison pipeline prepares the reference and test imagery for pixel to pixel comparison and is implemented differently for each type of imagery. Test run imagery is up sampled along the temporal dimension to the temporal resolution of the reference imagery, and reference imagery is down sampled spatially to the output resolution of the experiment, which is defined as the test run imagery spatial resolution. Temporal up sampling is performed by determining the image visible on the test run display device at each reference image sample coordinate. The previous test run image is duplicated for each temporal sample in the reference run between subsequent test run temporal sample locations.

Spatial down sampling of the reference is a low pass filtering operation that resizes individual reference images to the spatial resolution of the test run imagery. The test run imagery is produced at a specific spatial pixel resolution with the rendering engine configured for super sampling. Down sampling in the quality comparison pipeline is analogous to the filtering operation performed to resample subpixel samples to pixel resolution. The spatial resampling stage allows a single high resolution set of reference imagery to be compared with test run imagery at several different resolutions.

The third stage of the pipeline computes the mean squared error, expressed:

$$m_t = \frac{1}{N} \sum_{y=1}^N \sum_{x=1}^N (\text{Run}(x, y, t) - \text{Ref}(x, y, t))^2 \quad (7.1)$$

which is the average difference squared, between matching pairs of pixels in the test run and reference image sequences.

Test runs may be summarized in the fourth stage of the comparison pipeline by computing descriptive statistics across each error value across the entire run to describe the relative behavior of the adaptive approach. Relative behavior is expressed as a ratio between the run-to-reference

mean squared error of adaptive TSAR runs divided by the run-to-reference mean squared error of the nonadaptive runs, for each reference time stamp:

$$\frac{m_t(\text{TSAR})}{m_t(\text{nonadaptive})} \quad (7.2)$$

Quality ratio values greater than one indicate that the TSAR run error is greater than the non-adaptive error, and that even for a set of parameters within the cost constraints, the case is not suitable. For parameter combinations within the cost constraints, values less than one indicate suitability since the TSAR error is less than the nonadaptive error.

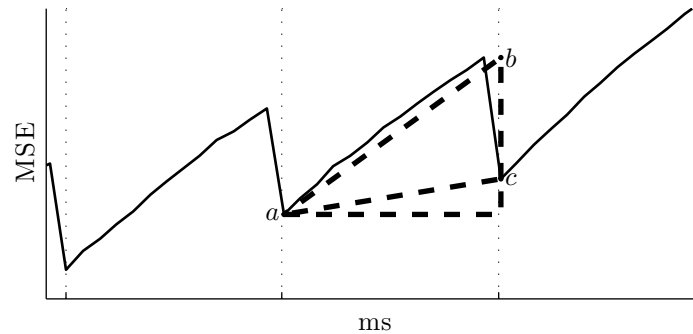
#### 7.2.4 Quality Index Behavior

Within the constraints described in Section 7.2.1, the relative spatial and temporal quality of rendered imagery may be deduced from the output of the comparison pipeline. The difference between pairs of pixels in a test run image versus a reference image is the sum of the difference between the reconstruction of spatial features and the reconstruction of temporal features in the test run. The quality of the reconstruction of both spatial and temporal features is determined by the adequacy of the respective spatial or temporal sample rate used to render the imagery. Although the mean squared error measure is dimensionless, over a sequence of images, it exhibits a characteristic behavior which may be used to understand the relative spatial and temporal quality of the imagery.

At the time of a temporal refresh in a nonadaptive renderer, indicated by vertical grid lines in Figure 7.8, the difference between the run and reference imagery due to temporal age or latency is minimized. The MSE value at the temporal refresh location is dominated by differences in spatial feature reconstruction; these locations indicate the level of spatial quality in the test run compared to the reference image.

Between temporal refreshes, the MSE value consists of the combination of pixel differences due to the age of the most recent test run image, and pixel differences attributed to the level of spatial quality of that image. These two factors are illustrated in Figure 7.8. At points *a* and *c*, which occur at temporal refreshes, the difference between the run and the reference images is due to the spatial quality level of the test run. The rate of change between points *a* and *b* relative to the rate of change between points *a* and *c* indicates the adequacy of the temporal refresh rate and the level of temporal quality.

Figure 7.9 shows the MSE error of two input scenes to a nonadaptive renderer; the first scene exhibits this characteristic behavior while the second scene does not. In the first example, shown in



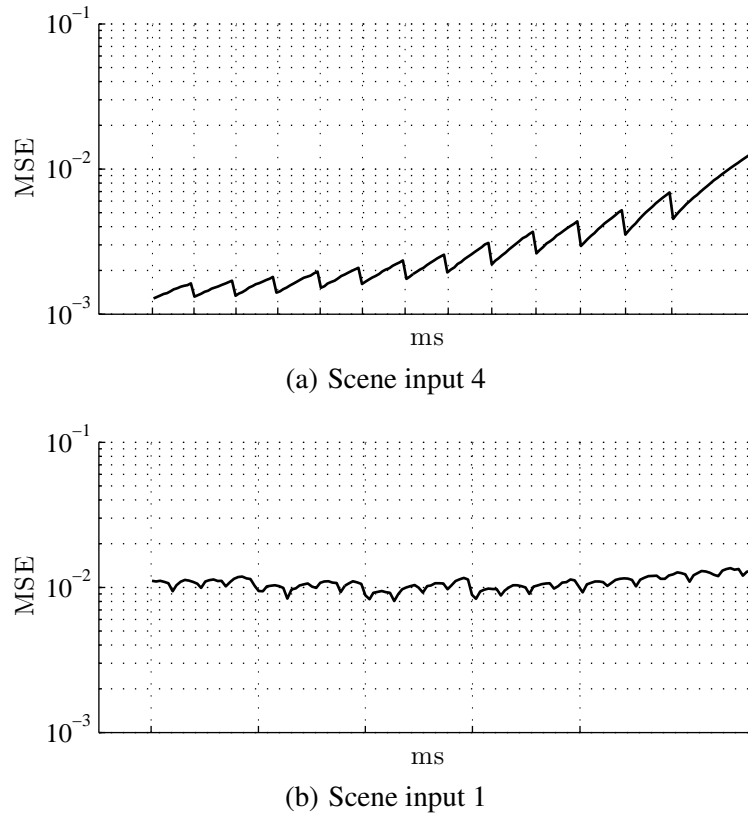
**Figure 7.8.** Characteristic behavior of image sequence MSE.

Figure 7.9(a), the spatial quality level decreases over time, but the temporal quality level, indicated by the slope of the MSE measure between temporal refreshes (at vertical grid lines), remains the same at the beginning of the sequence before increasing gradually towards the end. This example was taken from a nonadaptive framed renderer of test input scene 3 which contains consistent motion. The decreasing level of spatial quality is likely due to an increased amount of higher frequency details in the imagery. Figure 7.9(b) shows an example from input scene 1 which does not fit this characterization; here, the MSE is approximately the same at each temporal refresh; however, the value does not increase monotonically between refreshes. Temporal change in input scene 1 is predominantly periodic in nature because the camera leads the vehicle by a fixed amount and the highest frequency changing features are spinning wheels. The local minimums between test run refreshes are likely due to the wheels of the vehicle in the reference imagery, which is constantly changing, reaching an orientation similar to the previous test run refresh.

### 7.2.5 Suitability Results

For each of the five test input scenes, certain parameter combinations exist within the cost model bounds that produce imagery with a suitable quality ratio, i.e. where the MSE of adaptive TSAR generated imagery is less than the MSE of the framed nonadaptive imagery, both compared to a high resolution reference. Eighty test runs within the cost model constraints are shown together in Figure 7.10; each of the five columns in the figure contains sixteen test runs, indicated by box plots, for each test input scene using input parameters in the order of the rows listed in Table 7.2. Each box plot represents the distribution of quality index ratio, given by equation 7.2, across a test run. Due to the cost constraint, the total estimated computational cost of the TSAR approach in each test run is not greater than the nonadaptive renderer at the same output resolution.

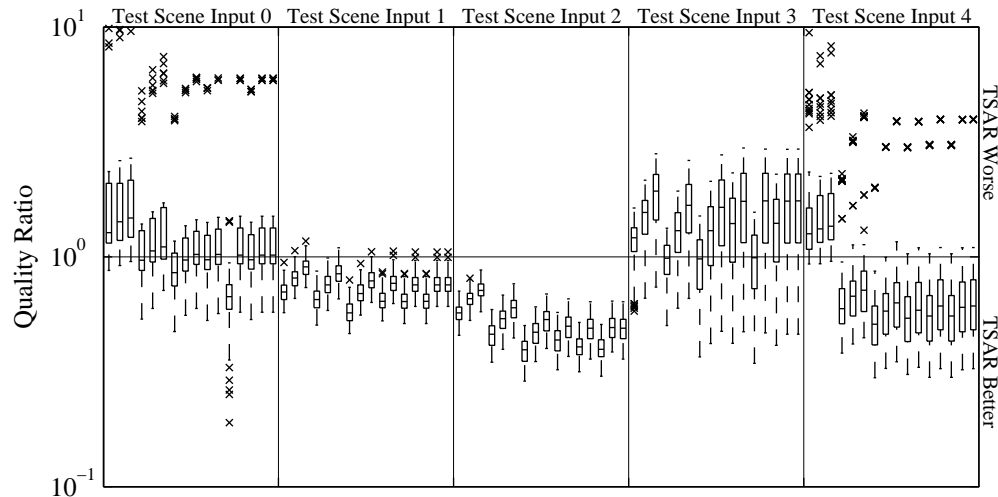
The figure shows distribution across each test run of the quality index ratio defined in Section 7.2.3 as the relative MSE between the adaptive and nonadaptive output imagery, respectively,



**Figure 7.9.** Mean squared error of nonadaptive runs. The vertical grid lines intersect the horizontal axis at the temporal coordinates of frames in the test run image sequence. Figures (a) and (b) show different indications of low temporal quality based on the behavior of the MSE value between temporal refreshes.

and high spatial temporal resolution reference imagery. Individual results for each test input scene are shown in more detail in subsequent figures. The parameters of each test run within an input scene are listed in Table 7.2 in order from left to right, e.g. the first run on the left side of the figure is of framelet batch size 32 and output pixel resolution  $128 \times 128$ ; the last run in the first column, the sixteenth box plot, is of framelet batch size 1024 and output pixel resolution  $1024 \times 1024$ . Values occurring below  $10^0$  indicate that the adaptive TSAR approach is better than the nonadaptive approach and that the test run is a suitable case. Values above  $10^0$  do not indicate suitability.

The relative suitability of the rendered imagery varies throughout the course of each test run as the animation causes the spatial and temporal characteristics of the scene to change. Only test input scene two produced imagery which met the suitability requirement across the entire test run for each combination of parameters tested. Test scenes one and four produced imagery that overall was suitable with the exception of some outlying instances, or certain ranges of parameters. Test scenes zero and three produce imagery with a quality ratio, and suitability, varying across the test



**Figure 7.10.** Overall quality comparison across all test runs within the cost constraint based on the frame to frame ratio between the quality index of the adaptive TSAR test run and the nonadaptive framed test run. Values in the bottom half of the figure are suitable and values in the top half are unsuitable. Each box plot indicates the distribution of quality index ratios across a single animation test run. The whiskers extend to the most extreme nonoutlier values. Outliers, described in Section 7.2.5, are indicated by crosses. Test runs using different input scenes are delimited by vertical grid lines.

runs.

Although only one scene produced suitable results for all parameter combinations, the majority of test runs produced suitable imagery most of the time. Across all test runs combined, the TSAR algorithm produces suitable results 66 percent of the time. The five right columns of Table 7.2 indicate the greatest percentile of each test run at a suitable quality ratio, i.e. in test run 11, input scene zero, the 80<sup>th</sup> percentile is the greatest percentile with a value less than or equal to one, therefore, 80 percent of the quality ratio measurements in the test run produced suitable results. Of the eighty parameter combinations tested within the cost model constraints, which correspond to different output resolutions and ray trace sampling budgets, one third are suitable all of the time, 46 percent are suitable 90 percent of the time, and 67 percent of the runs are suitable half of the time.

The 75<sup>th</sup> percentile of each run's quality ratio is selected to broadly distinguish suitable and unsuitable test scenes. Figure 7.11 and Figure 7.12 show the quality ratio distributions of test run scenes where the 75<sup>th</sup> percentile of each run is suitable, i.e. the quality ratio measures in the test run are less than or equal to one at least 75 percent of the time. Likewise, Figure 7.13 shows cases where the 75<sup>th</sup> percentile is greater than one. The twelfth test run in Figure 7.13(a) is an exception where only outliers quality ratio values are unsuitable.

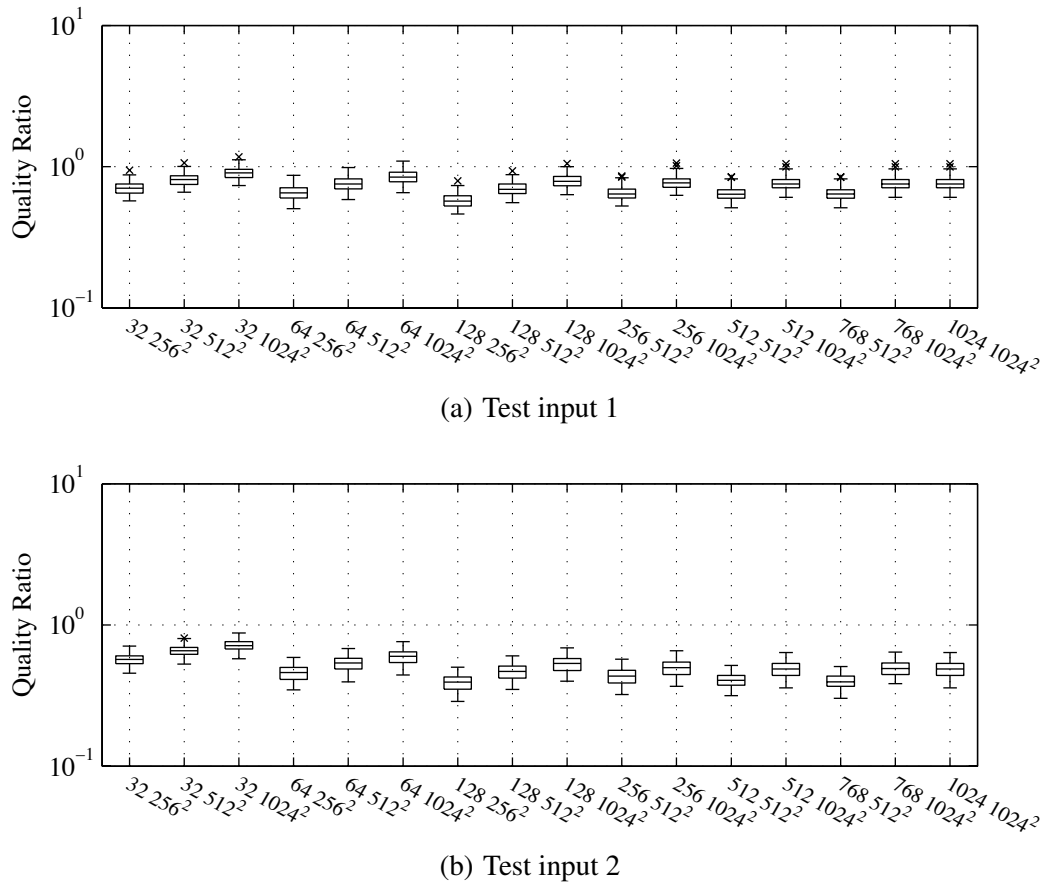
Test input scenes one and two indicate a sensitivity of the quality ratio to resolution more so

**Table 7.2.** Suitability test runs in Figure 7.10 listed in order for each set of sixteen test runs per input test scene. Parameter combinations not listed in the table are not within the cost model constraints shown in Figure 7.6. Columns to the right indicate the greatest suitable percentile, i.e. with quality ratio less than or equal to one, of each test run for all five test input scenes.

Run	Parameters		Suitable Percentile				
	Batch Size	Output Resolution	Scene 0	Scene 1	Scene 2	Scene 3	Scene 4
0	32	256 <sup>2</sup>	16	100	100	20	11
1	32	512 <sup>2</sup>	11	99	100	10	16
2	32	1024 <sup>2</sup>	6	86	100	8	6
3	64	256 <sup>2</sup>	53	100	100	53	90
4	64	512 <sup>2</sup>	38	100	100	21	87
5	64	1024 <sup>2</sup>	32	95	100	15	84
6	128	256 <sup>2</sup>	70	100	100	55	91
7	128	512 <sup>2</sup>	57	100	100	31	91
8	128	1024 <sup>2</sup>	45	99	100	20	88
9	256	512 <sup>2</sup>	56	100	100	28	91
10	256	1024 <sup>2</sup>	44	99	100	20	88
11	512	512 <sup>2</sup>	80	100	100	52	91
12	512	1024 <sup>2</sup>	45	100	100	19	88
13	768	512 <sup>2</sup>	55	100	100	28	91
14	768	1024 <sup>2</sup>	45	100	100	19	88
15	1024	1024 <sup>2</sup>	45	100	100	19	88

than to framelet batch size, and test scene four contains many outlier values which must be specially interpreted. In two of the five test scenes, certain outlier values, indicated by crosses in the figure, occur well above the upper quartile and in many cases are unsuitable quality ratios. The source of these outliers may be identified by comparing the behavior of the quality ratio and individual MSE values with the spectral behavior of the test input animations over time. Consider the plot of MSE values for test input scene zero shown in Figure 7.14. The MSE values of the nonadaptive framed run are drawn with a thick line, and the MSE values for each of the adaptive framed runs are drawn with thin lines. In this test scene, the camera is positioned at a fixed location looking down at the vehicle as it moves through the camera frame. The animation continues for a short time after the vehicle completely exits the frame and there is no more temporal change in the imagery. While the adaptive run MSE values generally follow the same shape as the nonadaptive MSE, towards the end of the sequence starting at the third to last refresh of the framed renderer (indicated by vertical grid lines), the nonadaptive run decreases to almost zero, and the MSE of each adaptive run stops changing substantially and is fixed at a certain error value. This behavior produces a number of quality ratio values outside of the midspread of the distribution.

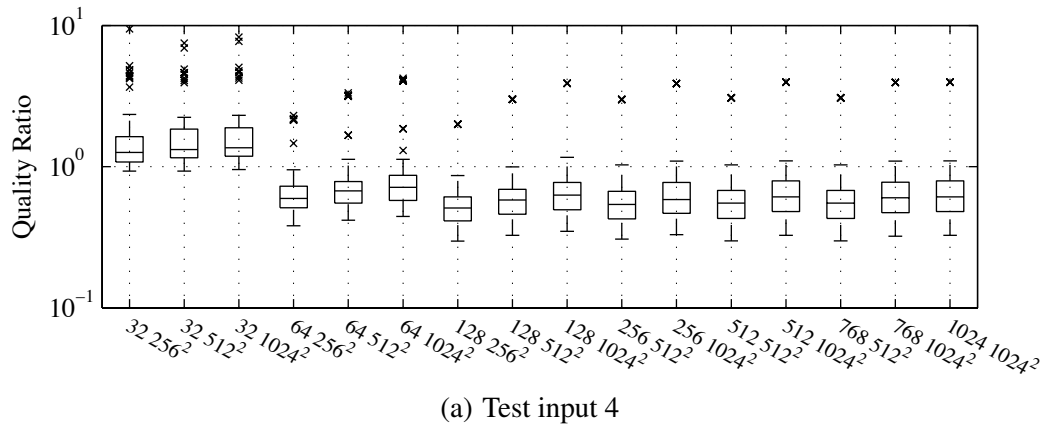
Figure 7.15 shows the quality ratio over time between the first test run of input scene zero and the nonadaptive renderer. Frame refresh of the nonadaptive renderer is indicated by vertical dotted



**Figure 7.11.** First two suitable cases.

lines. The upper and lower quartiles of the adaptive renderer quality ratio are indicated by dashed horizontal lines; outliers occur above the solid horizontal line. The quality ratio function jumps to an extreme value immediately following the third to last refresh. The MSE value of the adaptive runs in this outlier region correspond to the maximum spatial quality the configuration of each run is able to deliver in the absence of any temporal change.

The outlier behavior seen in test input cases zero and four suggests a limitation of the TSAR approach, and other adaptive approaches which attempt to find some predetermined balance between spatial and temporal fidelity; in instances where the scene stops changing, the sensitivity of the adaptive response mechanism must be adjusted to significantly bias the response to spatial features. In these cases, the TSAR algorithm detects a combination of spatial and temporal sample rate error for the majority of the animation sequence, and a balanced sensitivity to spatial and temporal signal features is appropriate; however, at the end of the sequence, the spectral characteristics of the scene change significantly but the sensitivity of the response algorithm does not follow suit. Test scene input two, shown in Figure 7.11(b), produces the best quality ratio distributions for all parameter



**Figure 7.12.** Third suitable case.

combinations and has the most consistent spectral characteristics throughout the animation. The same spatial and temporal sample rate error sensitivity was used for all of the suitability experiment test runs.

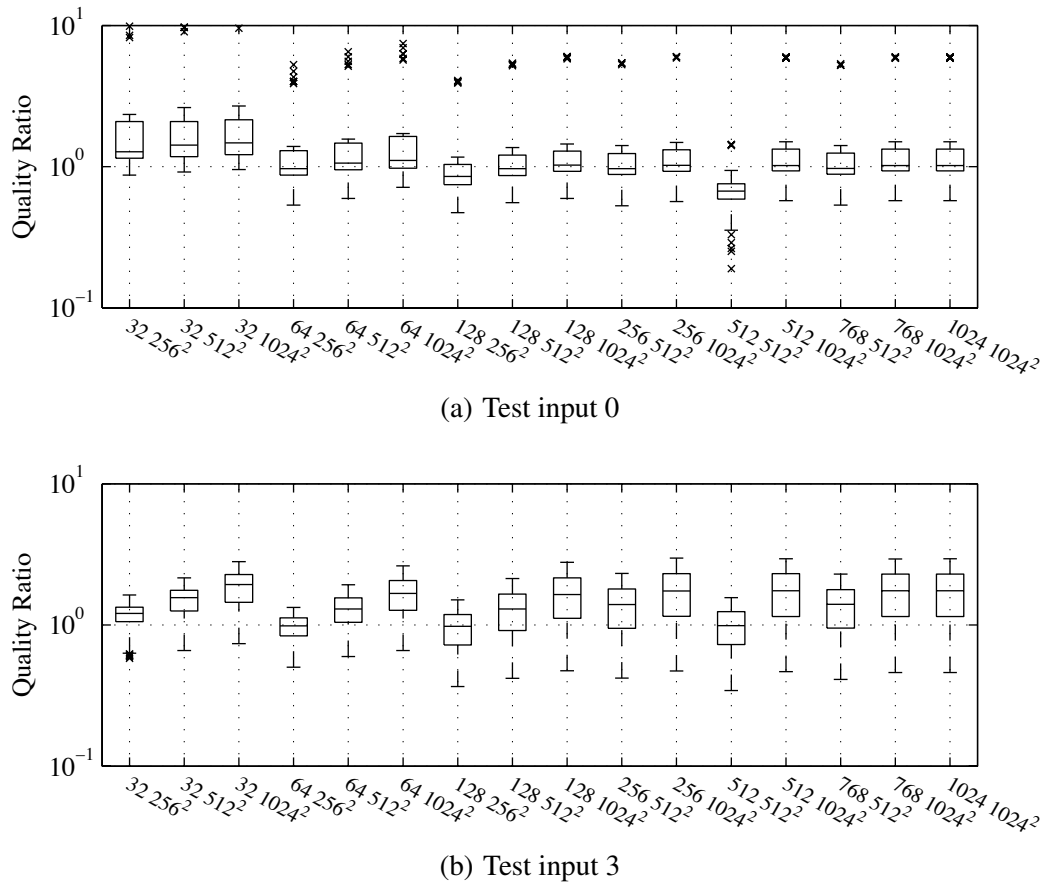
The quality ratio and pixel to pixel mean squared error measurement are more sensitive to output resolution than to batch size, evident by the grouping of the ratio values by reconstruction output resolution in Figures 7.16(a), 7.17(a), and 7.17(b) and in the behavior of quality ratio distributions shown in Figure 7.11 where the more quickly changing output resolution parameter appears to distinguish runs more so than the batch size parameter. In the case of test input scene four, the batch size parameter seems to reach a plateau between sizes 32 and 64 after which it has minimal influence on the quality distribution.

Sensitivity to output resolution is due in part to the comparison pipeline as well as the type of artifacts produced by the TSAR approach. The quality comparison pipeline down samples the high resolution reference image spatially to the test run resolution. The down sampling process is a low pass filter and serves to smooth high frequency features in the reference image. In the nonadaptive framed test run, these high frequency spatial features are likely aliased, even at the  $1024^2$  reconstruction resolution, compared to the super sampled reference. Reconstruction of the adaptive test run imagery in the TSAR pipeline down samples regions with sample rates greater than one sample per pixel and up samples other regions of the image, producing smooth low frequency artifacts that are likely to be closer to the smoothed reference image than the aliased framed run.

### 7.3 Qualitative Evaluation

Qualitative evaluation of the imagery produced by the TSAR algorithm provides an alternative to the objective analysis described in the first two sections of this chapter. The cost model described





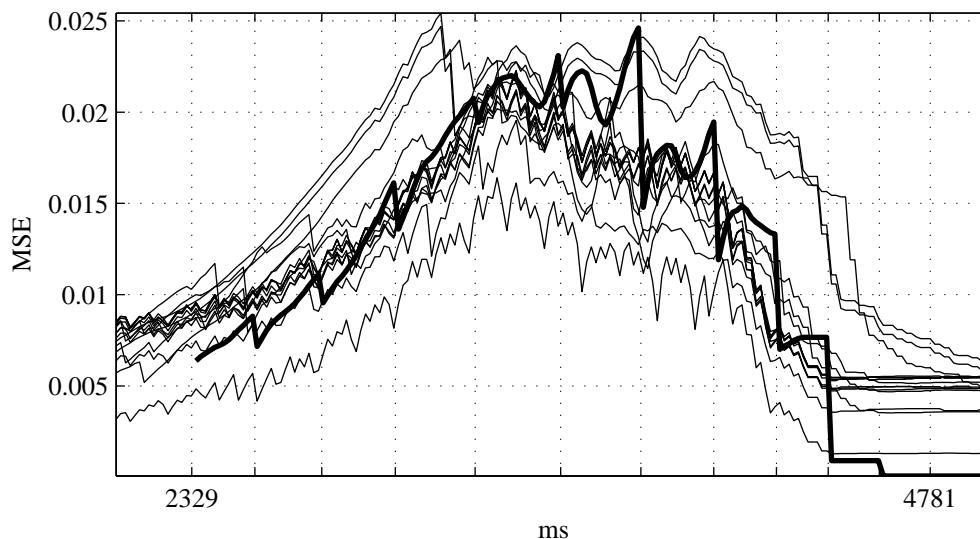
**Figure 7.13.** Unsuitable cases.

in Section 7.1.2 indicates the relative computational complexity of individual pipeline stages in comparison with a nonadaptive renderer; it does not, however, provide any information about the relative quality of imagery produced by algorithmic configurations with different costs. The quantitative analysis described in the previous section compares the relative difference between different configurations and a high resolution standard using a pixel to pixel mean squared error measure, the limitations of which are described in Section 7.2.1.

This section qualitatively describes the result of varying algorithmic parameters within the confines of the cost model and the suitability analysis. The behavior of the TSAR algorithm is characterized by example images from specific instances in the animation, instead of by descriptive statistics taken over the length of a test input scene animation.

### 7.3.1 Individual Scene Analysis

While the cost model and the quantitative quality measure allow an objective notion of suitability to be formulated in terms of cost efficiency and relative improvement over a nonadaptive approach

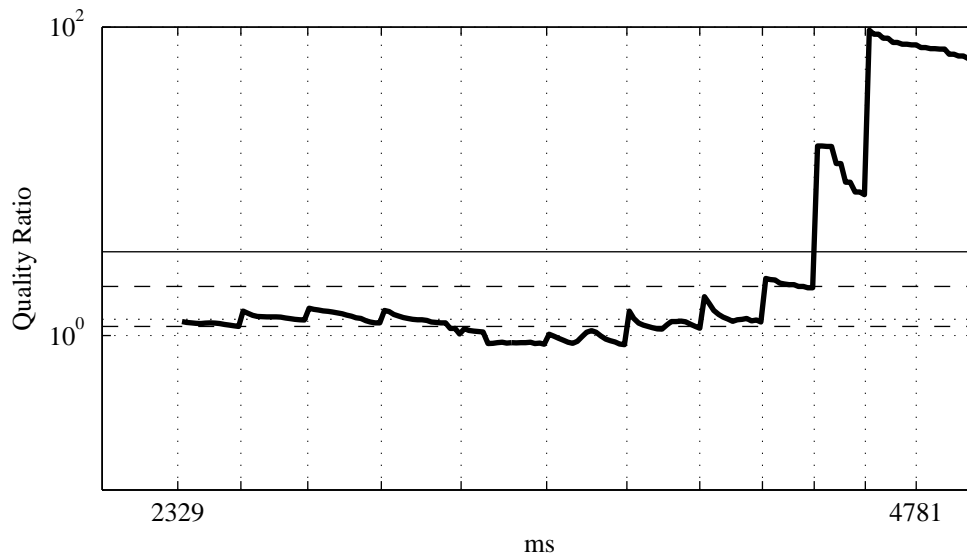


**Figure 7.14.** MSE corresponding to outlier behavior over time of scene input zero is shown. The nonadaptive framed run in bold along with sixteen adaptive runs.

across a test input animation sequence, the analysis does little to describe the varying level of imagery quality in perceptual or subjective terms. Limitations of the suitability analysis are due to constraints of the MSE measure and the reduction of the behavior of the system across an entire test sequence to a single distribution of error values. Unlike the suitability analysis, the principal parameter addressed in this section is framelet batch size which did not produce substantial variation in the suitability analysis of the previous section but is shown to have a substantial effect on the quality of imagery, especially on the degree of spatial and temporal reconstruction artifacts.

The qualitative examples presented in this section for each test input scene are shown in Figures 7.18 through 7.21. Each contains seven subfigures, one large reference image at the specified time stamp, followed by six smaller images from the framed nonadaptive renderer and different configurations of the TSAR algorithm. Each image contains the displayed image of the respective renderer at the time stamp specified, which is the frame output before the time stamp. The reference image is one megapixel and the test runs are either one megapixel or one quarter megapixel. All of the images are taken from test runs conducted for the suitability analysis described in the previous section.

Figure 7.18 contains an example frame from the end of test input scene zero. The frame is taken at time stamp 4554 after the vehicle has exited the frame and all temporal change in the reference imagery has ceased. This temporal region of the animation produced the outlier MSE values described in Section 7.2.5 due to inadequate response or spatial reconstruction. Figure 7.18(b) shows the output of the framed nonadaptive renderer in which the rear of the vehicle is

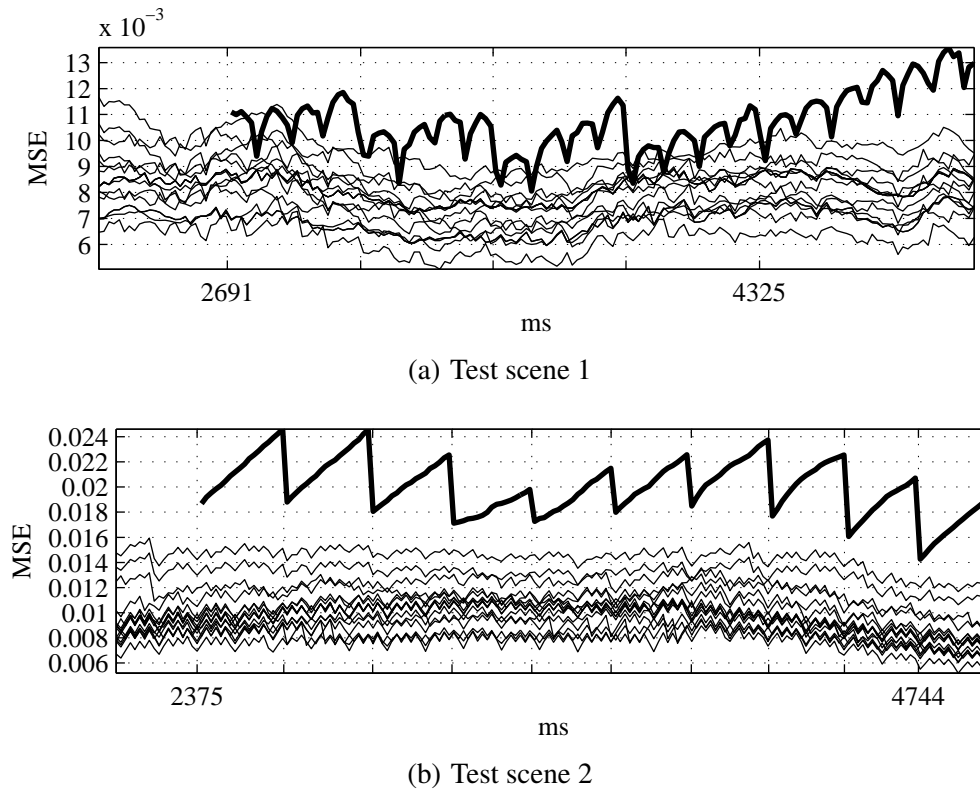


**Figure 7.15.** Quality ratio outlier behavior over time of scene input zero. The quality ratio for batch size 128 and resolution  $1024^2$  is shown. The dotted line indicates the median of the quality ratio values, the dashed lines indicate the upper and lower quartiles, and the solid line indicates the first nonoutlier value. The vertical axis of both plots is on a log scale.

still visible on the left side of the frame; this image is approximately ten milliseconds old, due to the lower refresh rate of the framed renderer compared to the reference or TSAR test runs. Likewise, significant temporal error is observed in Figure 7.18(c) which shows the output of the TSAR algorithm with framelet rendering batch size 32. In this case, because the framelet batch size is so small, i.e.  $1/32$  the number of samples per pipeline refresh compared to the nonadaptive renderer, the algorithm is unable to keep up with fast motion in regions with fine detail.

This temporal artifact nearly vanishes when the batch size doubles, and does not occur when the batch size quadruples; as seen in Figures 7.18(d) and 7.18(e), respectively. This temporal behavior may explain the difference in magnitude between the batch size 32 and 64 test runs and the other test runs with greater batch sizes seen in Figure 7.13(a).

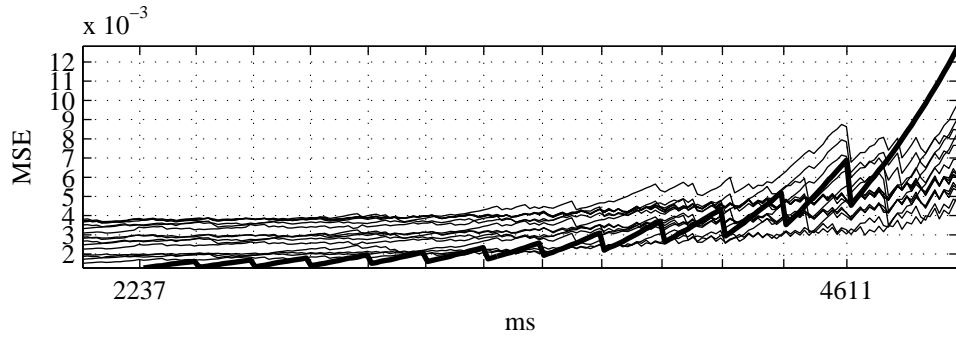
The temporal reconstruction error due to the use of old framelets stored in cache is due to both the design of the TSAR algorithm and constraints placed on the independent parameters for the suitability experiments. The magnitude and speed of spatial and temporal adaptive response are principally controlled by four parameters which adjust the overall magnitude and steepness of the desired spatial and temporal power spectrums; this mechanism is described in Chapter 5. These parameters control the response sensitivity to different sized features. Other parameters such as the maximum spatial and temporal sample rates also effect the response behavior. Test runs conducted for the suitability experiments limited the independent parameters to the output



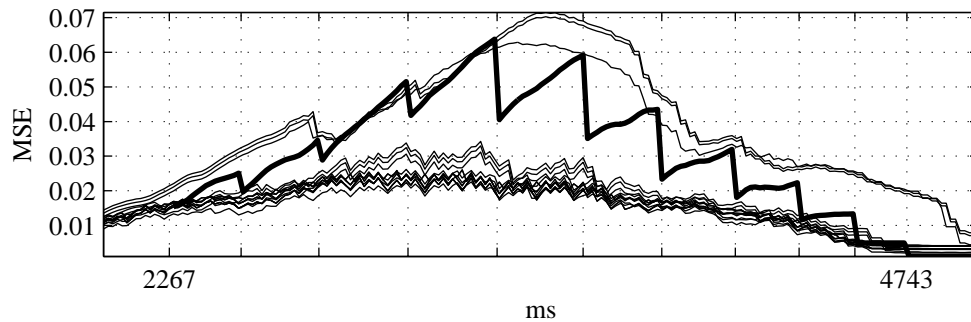
**Figure 7.16.** MSE values varying over time for scenes 1 and 2. The box plots shown in previous figures describes the quality ratio between each of the sixteen test runs in each test input scene, indicated by thin lines, and the thick line which is the nonadaptive framed run.

resolution and framelet batch size of the simplified cost model, which do not include the algorithmic parameters related to response sensitivity; those parameters might improve temporal sampling and reconstruction in this case.

The behavior is caused by an implicit bias towards spatial response in cases where the framelet batch size cannot adequately cover the number of framelet tiles which, according to the adaptive temporal response rate, must be refreshed across the image. The behavior may be addressed by modifying the pipeline to add a bias to constrain spatial response in regions with significant temporal change. The TSAR approach includes three mechanisms which may be used to add this type of bias. Mostly simply, the depth of the framelet tiling, i.e. the maximum spatial sample rate, may be constrained in situations with small framelet batch sizes. This constraint prevents the small tiles from consuming a large portion of the framelet batch but also prevents reconstruction of fine spatial details across the whole image. Second, the response sensitivity parameters which control the magnitude of spatial and temporal sample rate error based on the size of features detected may be adjusted in small batch size cases. This has the effect of reducing the framelet tile sizes in



(a) Test scene 3



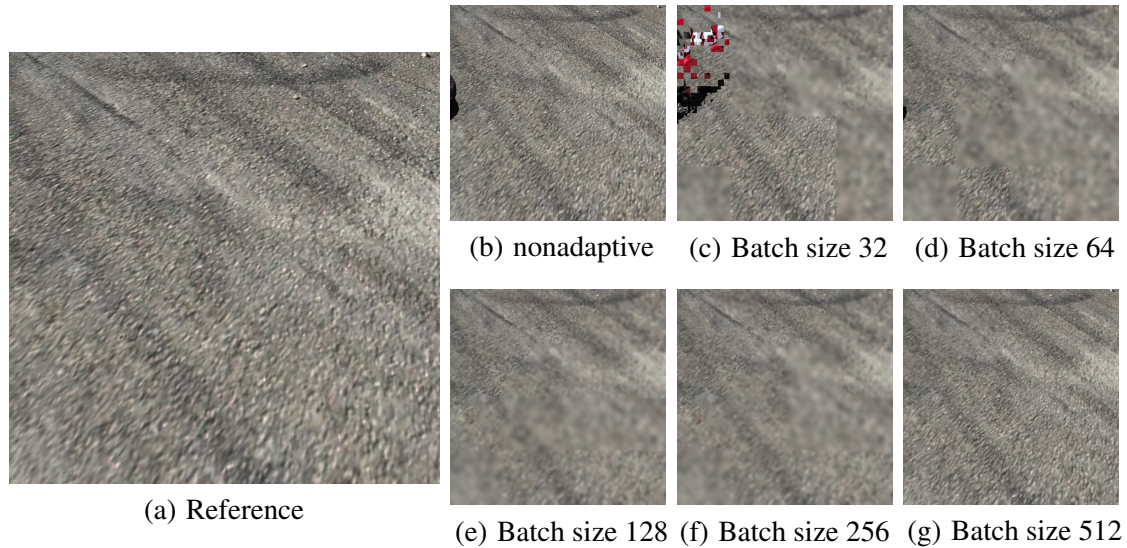
(b) Test scene 4

**Figure 7.17.** MSE values varying over time for scenes 3 and 4.

regions with fine spatial features; however, particularly large magnitude features may still induce large magnitude error and increased spatial sampling rate. The third mechanism is to add an explicit response bias in the last stage of the adaptive response control decision, i.e. define a certain portion of the upper right corner of the sample density space (small tiles, faster refresh) in Figure 6.1 as undesirable and modulate response decisions along iso-contours outside of that space.

Figure 7.22(c) contains another example of the implicit spatial bias and illustrates a side effect that is not as apparent in the first test scene; because the batch size is so small it takes the renderer a considerable amount of time to render framelets across the entire image and perform adaptive response, the sample rate in the foreground of the batch size 32 example is much higher than in the other adaptive test cases; it matches the nonadaptive renderer.

The low frequency spatial reconstruction artifacts visible in the lower right region of the TSAR runs in Figure 7.18 vary due to the different time of detection and magnitude of sample rate error based on the framelet batch size. In general, the batch size parameter does not alter the manner in which the framelet tiling adapts to spatial and temporal features, except that in extreme cases, such as Figure 7.18(c), where too small a batch size prevents the adaptive response mechanism from sufficiently sampling the scene temporally and delay its response to changing features.

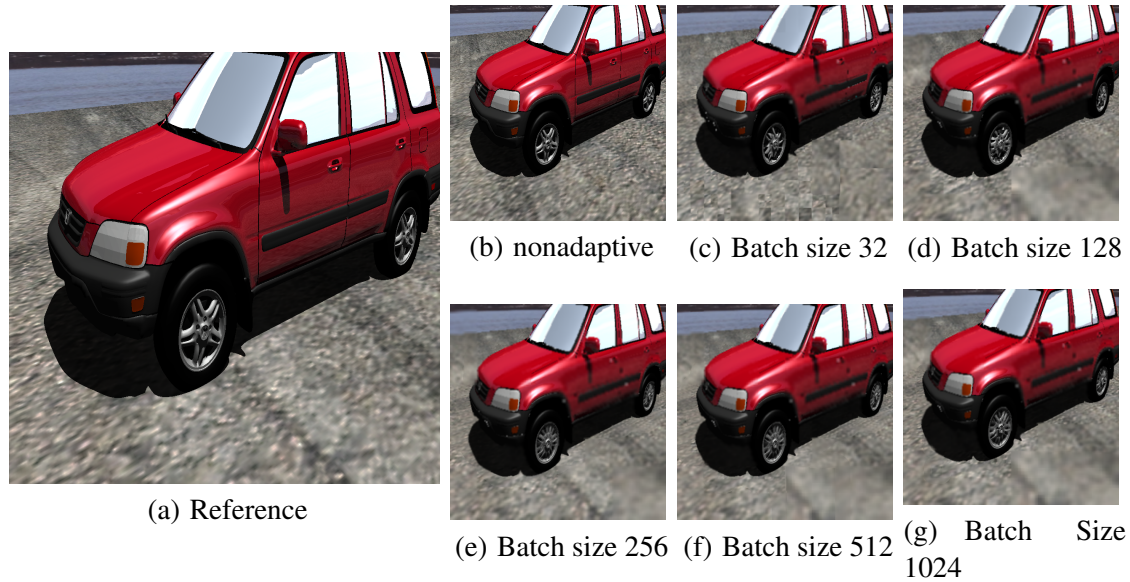


**Figure 7.18.** Examples from test input scene 0 at time stamp 4554.

Figure 7.18(g) shows an example frame from the exceptional case in test input scene zero, the twelfth test run,  $512 \times 512^2$  which produced suitable results and a small number of outlying error values. At the time stamp shown, this test run differs from the nonadaptive run in two principal ways. First, unlike the other test runs of input scene zero, the spatial sample rate, judged from the level of detail of reconstructed features, is approximately the same as the framed render and in some cases is higher. Also, due to the overall faster refresh rate of the TSAR pipeline, i.e. in this case the ray tracer is rendering half the number of image pixel samples of the nonadaptive renderer, the temporal error seen in Figure 7.18(b) is not present.

It is difficult to determine from the qualitative examples alone why the suitable behavior of the twelfth test run  $512 \times 512^2$  does not occur for the four remaining test runs at resolution  $1024^2$ . At a higher reconstruction output resolution it is likely that the spatial reconstruction error induced by framelet reconstruction was greater than the amount of temporal error caused by the visible bumper in the older nonadaptive image. It is also possible in these cases that the framelet tiling did not converge to the same spatial sample rate as the lower resolution case due to differences in the extent and magnitude of the spatial features detected.

The second test input scene was second most successful for the TSAR algorithm based on the suitability analysis; qualitative examples of varying the batch size parameter are shown in Figure 7.19. In this scene, because the motion of the vehicle is periodic and the orientation and location of the camera relative to the vehicle are fixed, it is difficult to observe differences related to temporal delay. The temporal delay between the reference image in Figure 7.19(a) and the nonadaptive output in Figure 7.19(b) are most noticeable in the reflections at the top of the driver's window on

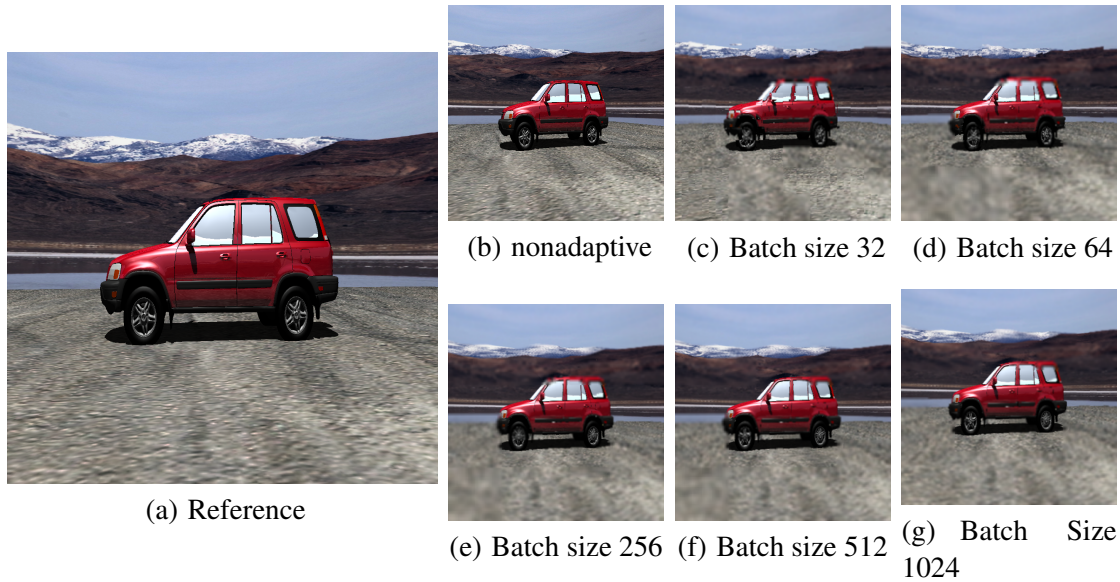


**Figure 7.19.** Examples from test input scene 1 at time stamp 4125.

the vehicle and the orientation of the wheels based on the rotation of the hub cap logo. In this case, the nonadaptive image was displayed at the same time stamp, 4125, as the reference image, but the scene animation was sampled earlier due to the pipeline latency of the framed renderer. The principal difference between the nonadaptive image and the TSAR test runs is the reconstruction of regions with fast motion and sharp spatial features and the spatial reconstruction of low frequency regions at the bottom of the image.

This test case is characterized by consistent spectral behavior throughout the course the the animation; differences between the test run are caused by the behavior of the TSAR reconstruction stage as well as the framelet selection mechanism. The reconstruction stage attempts to reduce artificial temporal edges between framelets of different ages in the framelet cache by increasing the extent of motion blur in regions where the age of the newest framelet changes. This effect is most noticeable between batch sizes 128 and 256, Figures 7.19(d) and 7.19(e), respectively. Beyond this range, a sufficient number of framelets are available in cache to produce a well motion blurred reconstruction. At batch size 1024 shown in Figure 7.19(g), the current orientation of the wheel, matching the reference image, begins to emerge because a sufficient number of framelets in the batch sample the wheel at the leading edge of time. In the low sample rate case of Figure 7.19(c), similar to the first test scene, the framelet mechanism biases spatial reconstruction within framelets surrounding the wheels of the vehicle at the expense of temporal quality. Unlike the first test scene, the artifact is less noticeable because the relative position of the wheels in the frame remains unchanged and the result appears to be a noisy reconstruction of wheel spokes.





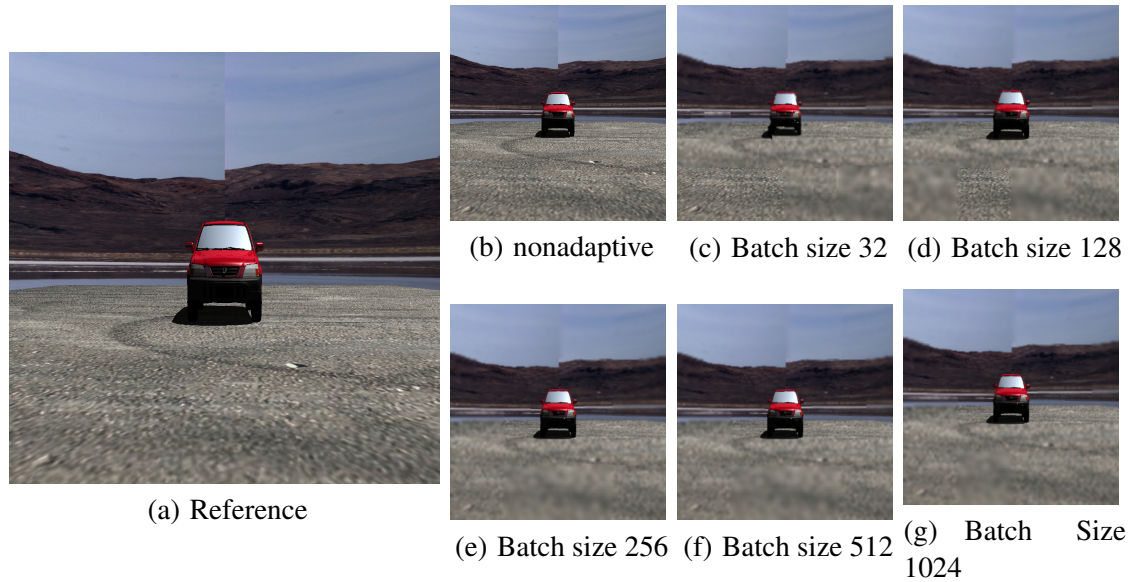
**Figure 7.20.** Examples from test input scene 2 at time stamp 2828.

The third input test scene, shown in Figure 7.20, is similar to the second scene in that the spatial and temporal spectral characteristics are generally consistent throughout the animation sequence; however, individual spatial and temporal features change position, orientation, or move across the image. Since the vehicle is moving past the camera, the extent of the projection of the vehicle on to the image changes slightly, requiring the adaptive response mechanism to adjust the spatial sample rate as the occluded region of the background changes. The changing orientation of the vehicle is the best indication of the temporal delay between the reference image in Figure 7.20(a) and the nonadaptive test run in Figure 7.20(b); more of the front end of the vehicle is visible in the latter.

Many of the reconstruction artifacts in the third test input example fall into the same categories of the first two examples, as shown in Figure 7.20(c) and in the fast motion regions around the wheels of the other images; the principal artifact shown in the third test case is the behavior of the algorithm in newly occluded regions, such as the region above the vehicle which is slowly occluded by the roof. This region is reconstructed poorly in instances with lower batch sizes such as Figure 7.20(d).

The third test scene is the second case deemed unsuitable for all parameters combinations within cost model constraints; example frames from the scene are shown in Figure 7.21. Temporal delay dominates the difference between the nonadaptive framed image, shown in Figure 7.21(b), and the reference image in Figure 7.21(a). The temporal delay causes the projection of the vehicle on to the image plane to be slightly smaller in the nonadaptive version which may be observed by considering the location of the vehicle relative to features on the ground plane texture. Poor response to the static

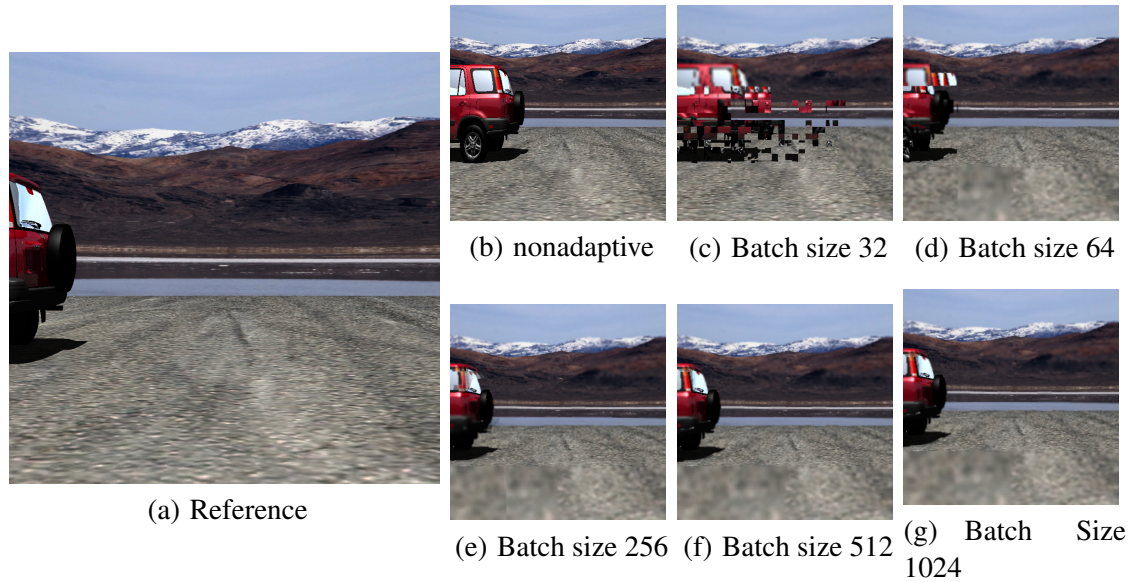




**Figure 7.21.** Examples from test input scene 3 at time stamp 3953.

spatial features in the scene dominates the difference measure between the adaptive test runs and the reference image. In the smallest batch size, Figure 7.21(c), one region of the vehicle shadow exhibits the spatial bias problem described for the previous scenes. The larger batch sizes do not exhibit the spatial bias problem; instead, the spatial sample rate is too low in the foreground and almost all high frequencies are lost. This problem may be addressed by increasing the spatial error sensitivity.

Like the first test input scene, the fourth test scene consists of a fixed camera frame through which the vehicle moves at a high rate of speed; unlike the first case the static regions of the image have varying spatial spectral characteristics. The difference between the reference and the nonadaptive run is considerable due to the temporal latency of the framed renderer, e.g. the amount of the vehicle visible in Figures 7.22(a) and 7.22(b), respectively. The difference in the position of the vehicle due to the fast motion between the reference and adaptive test runs is greater than in the other test input scenes. Batch sizes 256 through 1024 best illustrate this in Figures 7.22(e) through 7.22(g), respectively. The reconstruction stage of the TSAR algorithm compensates for the quickly varying age of framelets with different spatial features by increasing the amount of motion blur; in cases with fast motion, this results in several copies of spatial features from a range of time all blended together. The smaller framelet batch sizes, such as Figure 7.22(c), exhibit the spatial response bias almost to the same extent as Figure 7.18(c).



**Figure 7.22.** Examples from test input scene 4 at time stamp 4290.

### 7.3.2 Response Sensitivity

Response sensitivity, or the degree to which the TSAR algorithm changes the spatial or temporal sample rate when features are detected in the rendered signal, is the characteristic which best describes the deficiency observed in many of the examples shown in Section 7.3.1. Although several stages in the TSAR pipeline effect the response to changing signal features, interpretation of rendered samples occurs in the error estimation stage. Here, the spatial and temporal power spectrum of new framelets is approximated using a DWT approach and compared to a predetermined threshold. The algorithmic parameters used to shape the power spectrum threshold function, described in Section 5.3.3.2, control the magnitude of sample rate error produced by energy at different octaves in the DWT, which in conjunction with the adaptive response control decision stage determine the magnitude of response. This section describes a qualitative evaluation of the result of varying these parameters, specifically, varying the shape or slope parameter of the power spectrum threshold function.

Independent error sensitivity parameters are one aspect of the TSAR algorithm which distinguish it from other approaches. These parameters are avoided in systems such as AFR which use a global data structure to determine the relative priority of sampling each region of the image. AFR employs a hierarchical tiling over which individual samples are distributed and sample rate error is approximated. Adaptive response in AFR and the mechanism analogous to sample density transport in the TSAR algorithm is accomplished by building a priority queue of leaves in the tiling based on the relative amount of sample variance in each tile. In the terminology of the TSAR algorithm, the

priority queue has the effect of transporting sample density independent of the absolute magnitude of error measurements which are only needed to determine relative global order of image tiles in the queue. Sample rate error is defined as a dimensionless variance with an implicit zero point defined by the median priority in the queue. Implicit definition of the zero point and of the sample rate error range is the principal advantage of the priority queue mechanism. Without this mechanism in TSAR, a mapping between the amount of energy in a certain octave of the DWT and a specific sample rate error value must be supplied independently by the power spectrum threshold function. The disadvantage of the queue approach is that the data structure is global and its maintenance requires parallel communication and synchronization, steps that are avoided if the local error measurement may be directly translated into a desired local adaptive response independent of the global state of the system. Although the TSAR error sensitivity parameters are fixed for the suitability test runs shown here, the parameter values are appropriate for a broad class of imagery, absent the need to induce bias between spatial and temporal sampling.

The implicit spatial bias behavior observed in small framelet batch size test cases, e.g. Figures 7.18(c) and 7.22(c), is due in part to design limitations of the framelet-based adaptive sampling mechanism used by the TSAR. The effect is not encountered in similar low sample rate circumstances with single sample-based techniques such as AFR. In test runs of small batch sizes, the total number of samples available is highly constrained compared to the nonadaptive renderer, i.e.  $1/32$  of the framed renderer, since the number of samples contained within a framelet is fixed, each framelet makes up a large percentage of the total number of samples which may be refreshed in a single pipeline instance. Due to the fixed extent of framelets within the framelet tiling, the small number of total image samples cannot be distributed as broadly about the image as it might be in the single sample case. Instead of a distribution of individual samples covering the whole tiling, the available samples must be distributed to individual framelets within the tiling. In the case that the tiling contains a large number of small high spatial rate tiles, the small batch size might not be sufficient to refresh all of the tiles necessary to adhere to the adaptive temporal refresh rate. The problem is less likely to occur in situations with larger sample rate budgets and greater framelet batch sizes, or in situations with less temporal change, because the number of framelet tiles requiring temporal refresh is likely a smaller percent of the framelet batch size.

The DWT threshold function slope parameter effects the slope of the linear function defined in a logarithmic space, which when transformed into the linear space of the power spectrum approximation has exponential shape. The parameter is negatively valued such that the power spectrum threshold decreases exponentially from coarse to fine scales; larger magnitude parameter values cause the threshold function to decrease faster, resulting in the algorithm tolerating much less energy

at finer scales in the wavelet transform. The high resolution reference image for one time stamp from test input scene 4 is shown in Figure 7.23; in this example, the vehicle moves across a fixed camera frame at a high rate of speed. Figure 7.24 shows the effect of varying the spatial and temporal power spectrum threshold slope parameter on the reconstructed output at this time stamp. The most sensitive case, with the steepest threshold function, is shown in the upper left corner of the figure. The default parameter values, identified empirically and used in the suitability analysis, are shown at the center of the figure indicated by asterisks.

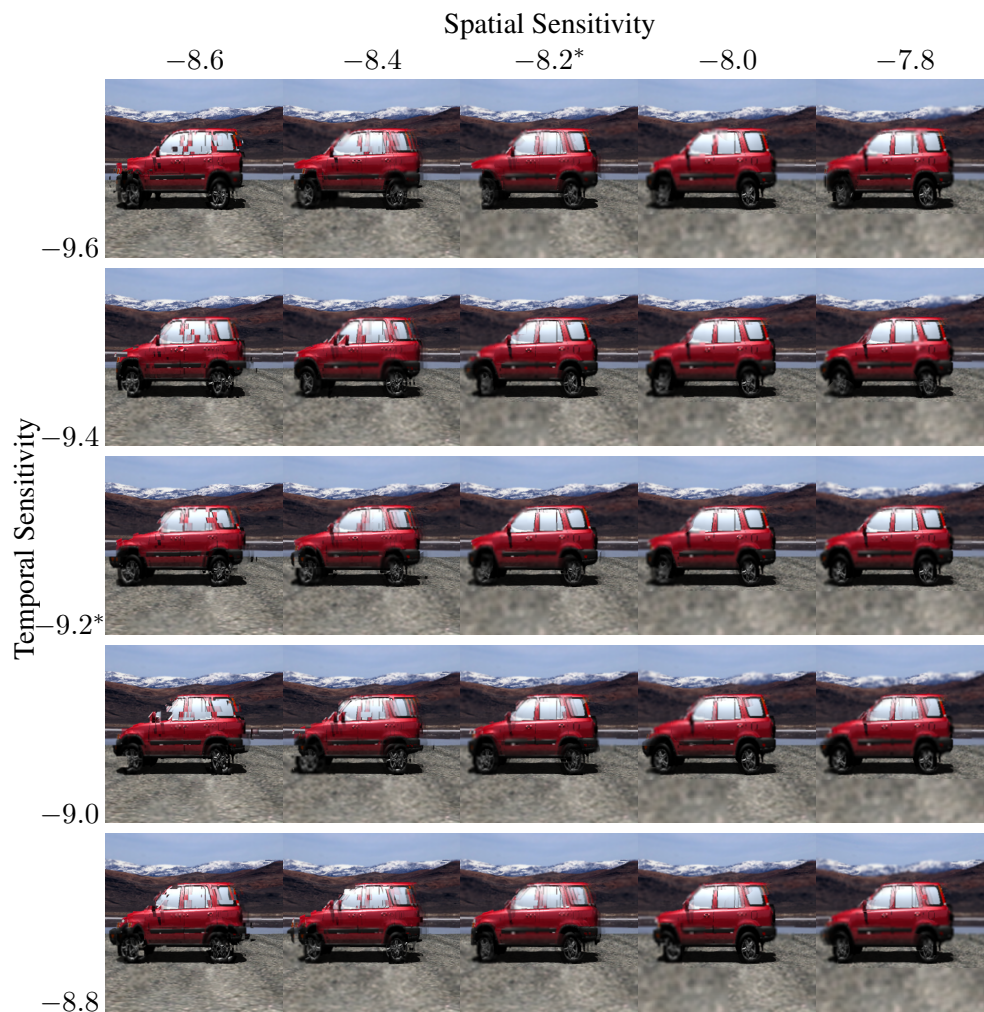
The spatial and temporal sample rates of the imagery shown in Figure 7.24 is shown in Figure 7.25; the spatial sample rate is indicated by the extent of each framelet tile; each framelet contains 256 samples; the color of each tile indicates the temporal refresh rate; darker tiles refresh at a lower rate than brighter tiles.

The effect of varying spatial sensitivity, shown decreasing in the horizontal direction in Figure 7.24, is apparent; the column of test runs with the steepest power spectrum threshold  $-8.6$  exhibit spatial reconstruction of the highest frequency features. High spatial sample rates are less pronounced in several cases in Figure 7.24 due to temporal smoothing during reconstruction, but may be observed in Figure 7.25 which shows the sample rate employed at the time stamp for each test run instead of the reconstructed imagery. The high spatial sensitivity results in a similar problem to the implicit spatial bias exhibited previously for test runs with small batch sizes. In the test runs shown in the left most column, the high spatial sensitivity results in a large number of small framelet tiles with high spatial sampling rates; in cases with high temporal sensitivity, the number of small high resolution tiles which must be refreshed at the leading edge of time exceeds the framelet batch size, resulting in the reconstruction of many small high resolution framelet at varying time stamps. The behavior is most apparent at the front end of the vehicle in the upper left corner at  $-8.6$ ,  $-9.6$  where many copies of high frequency spatial features from different time stamps are blended together during reconstruction. As the temporal sensitivity decreases, this behavior becomes less pronounced.

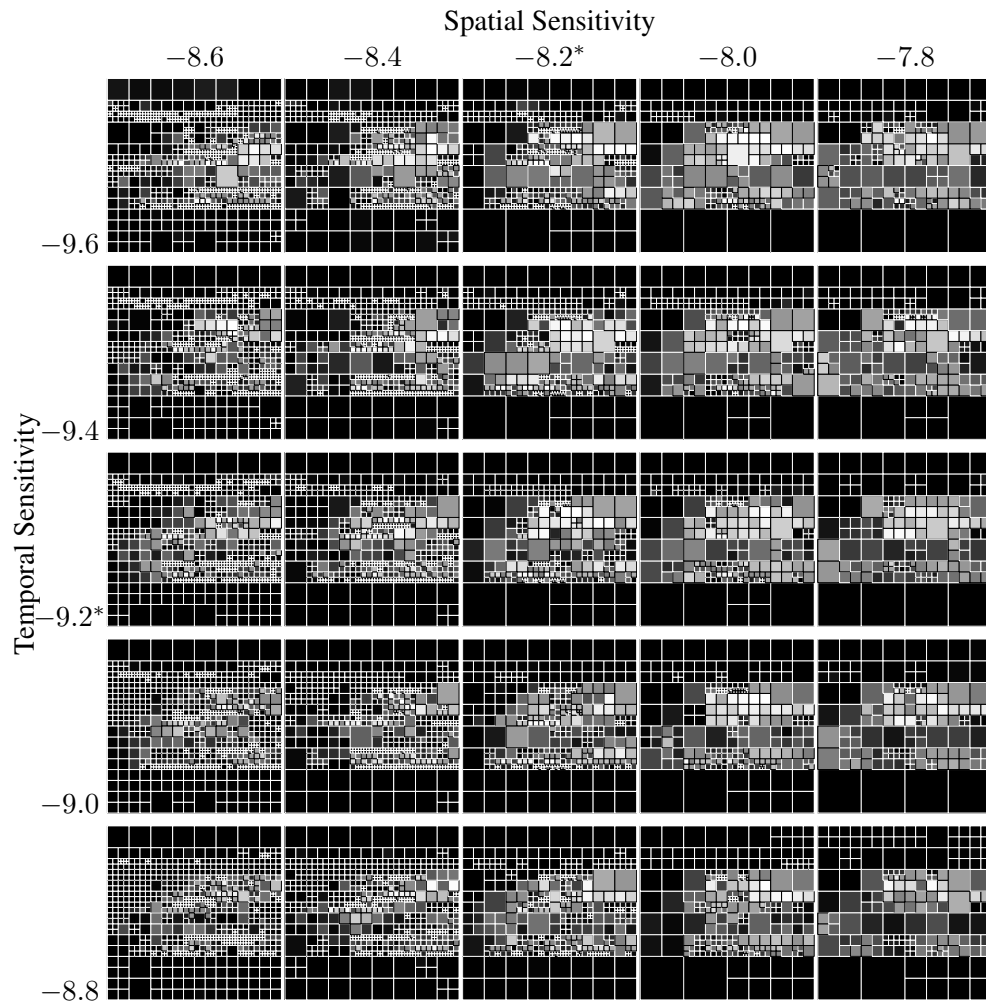
Varying temporal sensitivity, the vertical parameter in Figure 7.24, effects the latency of features in motion and the amount of temporal smoothing, or motion blur, applied to smooth differences in the temporal age of available framelets. Like the spatial sensitivity parameter, the temporal power spectrum threshold slope alters the degree to which energy is tolerated at finer temporal scales. In the case of test input scene four, this results in a varying amount of propagation of high temporal refresh rate framelets as occluding features of the vehicle move from right to left.



**Figure 7.23.** Reference image for error sensitivity parameters.



**Figure 7.24.** Spatial and temporal error sensitivity. Representative frames from test input scene 4 at time stamp 3300. Each parameter determines the slope of the DWT power spectrum template function, steeper slopes produce greater error response. The parameter combination used for the suitability experiment is indicated by an asterisk.



**Figure 7.25.** Error sensitivity effect on sample rate. Spatial and temporal sample rates for the imagery shown in Figure 7.24. Tile extent indicated spatial sample rate, each tile contains 256 samples; color indicates temporal refresh rate, darker is less frequent.



## CHAPTER 8

### CONCLUSION

The experimental results presented in the previous chapter consisting of a cost-based suitability analysis and the qualitative evaluation show the benefit of the TSAR algorithm in several animation situations. The chief conclusion, and principal contribution of this dissertation, is that the tile-based adaptive sampling scheme, used to divide the renderers computational capacity between spatial and temporal detail, is able to identify and exploit sufficient redundancy to improve fidelity. When compared to a nonadaptive framed renderer, the TSAR algorithm is capable of producing higher fidelity imagery at the same computational cost in three out of the five test input scenes; in one scene, the approach was better throughout the entire test animation, in the other two, the algorithm was better for more than half of the animation. The other central contributions of this work are the fidelity analysis framework, which motivates the response model, and the sampling and reconstruction mechanisms in the rendering pipeline which manipulate framelet tiles.

The adaptive sampling algorithm is distinguished from other techniques in three principal areas, the use of spatially and temporal coherent framelet containers for sampling, the signed estimate of sample density error derived from wavelet analysis, and the adaptive response mechanism based on image space statistical manipulation. These characteristics produce an adaptive rendering system which is capable of detecting and responding separately along the spatial and temporal dimensions. In the qualitative analysis, separable response allowed the renderer to sample and reconstruct fine spatial details in slow changing regions of the image. For example, in input scene four, the background near the horizon contained a high frequency horizontal edge which moved more slowly across the frame than the vehicle. Separable response allowed the renderer to sample this region of the frame with small framelet tiles refreshed at a low rate.

The separable adaptive response is possible due to the fidelity analysis framework employed by the TSAR approach. The framework provides control of the spatial and temporal sampling rate across the image by approximating the difference between the power spectrum of current generated imagery and expected or desired imagery. The framelet tiles provide the structured samples necessary to approximate the spectral behavior of generated imagery.

## 8.1 Interpretation

Chapter 7 provides both a quantitative and qualitative analysis of the system as a whole. The result is that the TSAR approach is most suitable for graphics scenes with consistent or gradually changing spatiotemporal spectral characteristics over the animation. This does not imply that motion in the scene should be slow, or isolated to specific regions of the frame, only that the amount of redundancy in various locations in the frame should be consistent. Test input scene two, the most suitable of the input tested, exhibits this characteristic. Although the background and object are constantly moving, and no two images in the animation sequence are the same, the position of the object within the frame is consistent as is the horizon between the foreground and background textures. The property of having consistent spatiotemporal spectral content is shared by a wide variety of different animated scenes, including test input scenes one, two, and four which each test a different type of animation.

There is a minimally sufficient framelet batch size, due to the spatial bias described in Section 7.3.1. Beneath this batch size, the TSAR algorithm is not able to distribute framelets across the image broadly enough to respond to widespread temporal change, while maintaining the spatial sample rate necessary to adequately reconstruct spatial detail. The first three test runs in each input scene with batch size parameter 32, or 8192 new samples per refresh, fall beneath this threshold in all test scenes. These are the most unsuitable runs in each test input scene, although in scenes one and two, the misperception occurs below a quality index ratio of one where the TSAR approach is better. Suitability quickly improves with the batch size until the threshold at which point the improvement plateaus. Therefore, if the TSAR approach was to be employed for arbitrary graphics scene input, across all potential input, not just in cases with consistent spectral characteristics, a batch size no greater than the minimum threshold need be used.

The overall quantitative results, which are presented in Figure 7.10 and Table 7.2, are that in one third of the animation test cases, an improvement was obtained all of the time; in nearly half of the cases, the TSAR approach was better 90 percent of the time; and in two thirds of the cases, the algorithm was better half of the time. These results may be interpreted either as a likelihood that the application of the TSAR approach to a certain graphics solution will yield an improvement, or as an indication of the sensitivity of the algorithm to its input. Unlike fidelity independent techniques described in Section 2.4.1, the TSAR algorithm does not at all times improve performance in all cases, e.g. the situation described in Section 7.2.5 where the scene did not permit a spatiotemporal trade off.

The criteria for determining successful application of the TSAR algorithm is different compared to the application of other algorithms because the TSAR algorithm enables a dynamic fidelity



trade off, but does not effect the intrinsic rendering capability of the system. For example, in the test cases included in the previous chapter, the TSAR algorithm did not effect the shadow or reflection algorithm used by the renderer, or the underlying ray tracing algorithm itself. It did not substitute a more expensive rendering algorithm to obtain greater spatial quality throughout the entire test sequence, or substitute a less expensive algorithm to obtain greater temporal quality throughout. The control algorithm is sufficiently decoupled that in certain instances, the underlying renderer may have even produced higher quality effects, such as reflections or shadows, which were smoothed away or otherwise discarded by the TSAR algorithm. This decoupling of the TSAR pipeline from the underlying rendering engine is also a strength of the approach; provided that the lower level rendering engine is capable of placing rendered samples in framelets, the adaptive sampling algorithm is independent of the image synthesis mechanism. It would be possible to completely uncouple the TSAR pipeline from rendering engine in cases where it does not result in an improvement, e.g. in the absence of sufficient spatial and temporal error in different regions of the image to permit conservation during response. The results should be interpreted to indicate is that in half of the cases tested, the adaptive approach should be turned on 90 percent of the time, or in two thirds of the cases, it should be activated half of the time. The remainder of the time, better results would be obtained using a nonadaptive sampling approach.

The analysis presented here does not attempt to compare artifacts caused by the TSAR algorithm leading to fidelity degradation, with those caused by alternative techniques to increase spatial or temporal quality. For example, considering the test scenes used in this dissertation, it is not known how an explicit static trade off in spatial quality, such as removing reflections or shadows from the imagery, to increase temporal refresh, would compare to artifacts introduced by TSAR in unsuitable cases.

This interpretation of the quantitative results in terms of the amount of time in a graphics application at which the TSAR algorithm should be activated or deactivated is not a substitute for the design decision of whether or not to include the algorithm in a system, or which components to extract of the prototype described in this dissertation. The decision to include a temporally and spatially adaptive rendering algorithm in a graphics system should be made by classifying the spectral characteristics of the intended workload. Ideally, the workload will fall as close as possible to the test scenes in the first third of the test cases where the algorithm produces an improvement all of the time.

## 8.2 Characteristics of Appropriate Workloads

The graphics test scenes used for evaluation in Chapter 7 are examples of imagery that would be found in a certain type of entertainment or visual simulation application. Although the scenes are different in appearance than imagery that would be produced by a visualization application or other types of video games, the test input scenes contain several different types of spatial and temporal situations, such as when the camera is fixed in one place, chasing and object, or tracking an object as it moves past. These situations are found in a broader set of graphics workloads. The evaluation of these five test input scenes may be used to describe characteristics of appropriate graphics workloads for use with the TSAR algorithm.

The decision to employ the TSAR algorithm must be made both at the level of the application use case, and at the lower level of the suitability of the rendered imagery. The test input scenes used in Chapter 7 are examples of different types of scene input and output imagery; the evaluation addresses the lower level question of whether for these examples, the TSAR algorithm produces higher fidelity results. The question of suitability to specific applications, e.g. video games versus visual simulation, is not addressed by the evaluation. At the application level, the use of graphics must permit a trade off between spatial and temporal fidelity, and the type of artifacts introduced by the approach must be acceptable. The application must benefit from a specific trade off between spatial and temporal fidelity, i.e. there must be a fidelity objective which can be expressed in terms of relative spatial and temporal response sensitivity parameters. Reconstruction artifacts, specifically motion blur due to temporal smoothing, must be acceptable to the graphics application. At the rendering level, the TSAR algorithm must produce higher fidelity imagery of the graphics scenes than the nonadaptive approach. The imagery produced by the graphics application must contain features of varying detail to which the algorithm may adapt.

The graphics application must have a specific objective regarding the relative importance of spatial or temporal fidelity; while the TSAR algorithm is able to control both, it exposes only a mutable trade off; it does not determine the best trade off to make or contain any inherent basis for making the trade off. Within an application, the bias towards spatial or temporal fidelity might depend on the use of the application and might even vary over time. For example, in a visual simulation application, temporal fidelity might be more important when reaction time is critical, but spatial fidelity more important when detailed information is needed to make a more considered decision. The application may bias sampling in either direction by manipulating the spatial and temporal error sensitivities of the response component of the pipeline.

The imagery must contain changing features to which the algorithm may adapt; response to new features in the underlying image signal requires an overall balance of detail flux in underlying

imagery. The fly-by camera in test input scene, which was the most suitable, contained sufficient change in individual spatial and temporal features while maintaining consistent spatiotemporal spectral characteristics across the frame. In order to enforce conservation of overall sample density, the amount of temporal or spatial detail in input scenes must decrease in one region of the image if it is to increase in another region over the course of an animation. Test input scene three, where the amount of spatial detail only increased over time, was among the least successful. In situations such as test input scenes zero and four, which exhibited the outlier example described in Section 7.2.5, where the spectral characteristics of the underlying image stop changing, the adaptive response mechanism will be unable to transport sample density across the image when a feature appears or disappears.

The graphics scenario should allow a slight bias towards temporal fidelity at the expense of high frequency spatial fidelity. This bias is introduced by the TSAR approach in regions bordering different temporal refresh rates during reconstruction. Scenes which include intentional motion blur effects from fast moving objects, or cases such as the spinning wheel in test input scene one where motion blur is plausible, already implicitly accept this bias.

These characteristics imply that it is acceptable to make a trade off between spatial and temporal fidelity in the graphics application domain. In entertainment graphics and in many types of scientific visualization, the loss of one fidelity dimension inherent in the trade off is acceptable; however, graphics applications which produce quantitative information or require high spatial or temporal precision are not candidates for this type of fidelity trade off.

### 8.3 Algorithms

Considered beyond the scope of the prototype and test application, the adaptive sampling mechanism consisting of a spatiotemporal framelet tiling, the reconstruction approach, and the adaptive response model, are well encapsulated and extractable pieces of the TSAR system. The first two components of the approach are particularly extractable from the larger system described here and may have practical application to other system. The third is a component for which significant effort was expended, but may be implemented very differently if additional constraints were placed on the overall system, such as a specific rendering application, or imagery with a specific spectral characteristic. The final chapter in this dissertation is organized in a forward looking manner as a review of the components which are most likely to be used in the implementation of a future system, and the conclusions which support their reuse. Aspects of the design which are particularly related to contemporary hardware considerations are also described, as is the potential space of alternative designs.

### 8.3.1 Spatiotemporal Sampling with Framelets

In practical terms, the experiment posed by the TSAR prototype is whether the rendering work performed to refresh a whole frame at a single instance in time may be divided into smaller pieces and distributed across the image over an interval of time to sample and reconstruct the underlying image signal with greater fidelity and efficiency. The sample rate error estimate, adaptive response model, and reconstruction approaches are by-products of the framelet tile-based spatiotemporal adaptivity experiment.

The use of framelet tiles for sample placement and as sample containers throughout the pipeline is the novel contribution of this work. The design of the spatiotemporal sampling mechanism of the TSAR algorithm avoids single sample control decisions or rendering operations, instead grouping samples into spatially coherent framelets and temporally coherent batches of framelets. The spatiotemporal sampling mechanism distinguishes the TSAR algorithm from other temporally adaptive approaches such as Adaptive Frameless Rendering. The motivation for using a tile-based approach rather than a single sample, or nonuniform sample placement mechanism, was to increase amortization and computational efficiency during rendering. The coherency of the framelet tile in image space matched coherent ray tracing methods such as ray packets; although in the Manta ray tracing system, each framelet contains several primary ray packets.

The framelets, the tiling, and the scalar fields used to construct them, allow the computational expense of rendering and of the adaptive response control decision to be amortized over large extents of the output image. This design enables the cost of the extra components added to the graphics pipeline to scale independently from the number of pixels in the output image. The spectral content of the underlying image signal dictates the resolution of framelets in the tiling, and the number of framelets needed to adequately sample the image signal and reconstruct output imagery. The cost of the adaptive response control decision scales in the resolution of the scalar fields used for error estimation which are a parameter of the algorithm which must be empirically selected based on the anticipated graphics workload.

Framelets had a significant impact on the entire TSAR pipeline, as a container of structured uniform samples, they serve as the basis for the error estimation model and are the fundamental data type throughout most of the pipeline. In respect to error estimation, the framelet may be seen as an extension of the cross hair sampling pattern employed by AFR to compute spatial and temporal derivatives. Since the extent of a framelet and number of samples contained within it is much greater than the pattern used in AFR, more coherence may be exploited by the rendering engine and higher order derivatives may be computed for error estimation.

The structured coherence inherent in the framelet allows the algorithm to avoid searching for

neighboring samples or maintaining a spatial search structure. The framelet contains enough information to estimate sample rate error within its extent over a smaller number of octaves in the underlying image signal; in cases where communication beyond the extent of a framelet is needed, such as the adaptive response control decision or reconstruction of the final output image, framelets are efficiently resampled to uniform resolution scalar fields.

The framelet tiling approach, including the mechanism to divide the image into tiles, group samples within the tiles into renderer fragments, and obtain the rendered result, does not by itself decrease the rendered sample throughput of the Manta renderer. In certain cases, the assignment of rays to framelet samples in Morton order increases the sampling throughput compared to the default block linear approach used by the renderer which was designed to avoid false sharing of cache lines on multiprocessor systems.

The framelet tiling is a piecewise constant approximation of the continuously varying sample density field, although the framelet may be able to adequately sample fine details at a resolution higher than the tiling, the square shape of the piecewise constant tiles, and the coarse change in spatial and temporal sample density between tiles of different sizes and refresh rates, constrain response to small spatial or temporal features. Reconstruction, the pipeline stage which is responsible for smoothing discontinuities, is much more successful at addressing spatial artifacts than temporal artifacts, since in the spatial case, a framelet containing samples at the wrong resolution at least contains current information about the underlying image signal; in the temporal case, discontinuities or errors in the temporal sampling rate result in missing information or samples which are too old to be of use. Since the response to a temporal discontinuity during reconstruction is to introduce motion blur on the more recent side of the temporal edge, inadequate temporal sampling due to the granularity of the framelet tiling will result in a smoothing of fine spatial details as well.

### 8.3.2 Reconstruction

Reconstruction received the least attention in this dissertation, which focused on the sampling components in the front end of the pipeline, although the last pipeline stage is the most expensive component of the TSAR algorithm because it operates on each pixel in the output image. While the arrangement of samples into framelet tiles does not increase rendering overhead at the beginning of the pipeline, expensive operations are required during reconstruction to address artifacts caused by the framelet tiling. For example, sharp discontinuities in the spatial sample resolution might occur when only neighboring framelets of different sizes are available in the cache. Temporal discontinuities, which result in popping artifacts, occur between batches of framelets which are rendered at the same time. The cost of reconstruction scales linearly with the number of output pixels; expensive operations cannot be amortized across large extents of the output image, unlike

other stages in the TSAR pipeline whose costs scale with the number of framelets.

One unexplored alternative to the reconstruction approach described in this work would be to push the smoothing of discontinuities between framelets of different sample densities to a stage earlier in the pipeline where the cost per framelet would still be amortized over a larger extent of the output image. For example, in transition regions between framelets at different spatial resolutions, smoothing might be performed on affected framelets during error estimation. The operation cannot be moved too far forward in the pipeline since framelets on the higher resolution side of the discontinuity provide information about signal behavior which would be lost by the smoothing operation.

### 8.3.3 Adaptive Response Model

The adaptive response component of the TSAR algorithm consists of the set of algorithms which transform a batch of rendered framelets into a manipulation of the spatial and temporal sample rate used in subsequent sampling.

The adaptive response model follows from the observation that rendered imagery not only exhibits temporal coherence, but also that in many circumstances, the localization and degree of temporal coherence is consistent over an interval of time. That is, the rate of change, of the amount of change in regions of the image is consistent over time. In many instances, this quantity, similar perhaps to a second, or higher, derivative of temporal change, has a much smaller magnitude than simply frame-to-frame difference. Consider the examples described in the evaluation section, even in the tracking camera situations; even though the entire image changes between any two moments in time, the rate of change in specific regions of the image remains largely the same. This allows the adaptive response mechanism to progressively update the desired sample rate across the image.

This observation motivates the spectral definition of spatial and temporal sample rate error, and the use of a wavelet transform, which explicitly computes the second and higher order temporal differences, in its analysis. It also reduces the importance of reprojection, a mechanism which relies on first order temporal difference to reuse image features over time. Reprojection was central component of the AFR algorithm; however, it was not practical with the framelet mechanism of the TSAR approach because a mechanism to transform sample coordinates while amortizing the cost of expensive operations over framelets was not found; neither was an approach to amortize the decision about which samples could be reused. Both problems are due to the greater number of coherent samples contained within the framelet, versus a single sample or small number of samples operated on at one time in the AFR approach. Instead of reprojection, the TSAR algorithm relies on temporal smoothing, or motion blur, during reconstruction to exploit first order image difference coherence.

## 8.4 Future Work

Several components of the temporally and spatially adaptive rendering algorithm described and evaluated in this dissertation present an opportunity for further research. The most extractable components for further investigation are the framelet tile-based adaptive sampling mechanism, the spatiotemporal reconstruction algorithm, and the algorithm used to implement the adaptive response control decision.

The structured sample placement using framelet tiles is a central component of the TSAR algorithm which ultimately proved successful, although constraints placed on the framelet tiling contributed to reconstruction artifacts, especially in regions on the border between two spatial sample densities. Future work might explore the use of framelet tilings which allow overlap, or tilings which permit a more gradual change in spatial sample rate. The key challenge of this direction would be to identify efficient data structures and algorithms to manipulate the tiling.

Adaptive sampling, not reconstruction, was the focus of this dissertation. Given the success of the tile-based scheme, future work might examine how to improve the fidelity of rendered imagery by decreasing the amount of spatial and temporal smoothing required during reconstruction. One approach might be to use information collected about the image during error estimate to effect reconstruction filters. Reconstruction is a difficult stage in the TSAR pipeline because it is the only component whose computational complexity scales with the resolution of the output image instead of the amount of spatial and temporal detail in the imagery.

The adaptive response control decisions stage of the TSAR algorithm is another potential area for further investigation. During the design of this component, several attempts were made to model the global constraint stage by modeling sample transport around the domain through the simulation of a physical process. Behavior prediction based on an analogous physical system was thought to be an advantage in some situations, e.g. if a high-frequency feature moves into the image, sample density may be transported into the surrounding region and require a lower magnitude response as the feature continues to move; however, in other cases, the physical system results in undesirable response behavior such as the formation of waves in the sample density field. Modeling adaptive response as a physical process was the original motivation for eliminating the hierarchical kd-tree sampling structure from the AFR design. The intention was that many PDEs had well known parallel solutions which are well suited to conventional graphics hardware and intrinsically more parallelizable than updates to a large hierarchical structure. This gave way to modeling sample rate error and sample density in both spatial and temporal dimensions as scalar fields suitable for differential operators rather than other data structures.

Rather than an adaptive response model based on differential properties of a analogous physical

system, the component described in Section 6.3 is a first order manipulation which is analogous to statistical histogram manipulation. The mechanism to adjust the rate of change of the sample density is shown in Figure 6.5 as if it was a cumulative histogram, or more specifically as a pair of histograms containing the positive and negative values of the scalar field. The techniques described to apply global constraints on adaptive response are implemented in terms of these functions which are derived from the amount of sample density increase and decrease across the image.



## REFERENCES

- [1] AILA, T., AND LAINE, S. Alias free shadow maps. In *Proceedings of Eurographics Symposium on Rendering 2004* (2004), Eurographics Association, pp. 161–166.
- [2] BALBOA, R., AND GRZYWACZ, N. Power spectra and distribution of contrasts of natural images from different habitats. *Vision Res.* (2003), 2527–2537.
- [3] BAXTER, B., AND CORRIVEAU, P. PC display resolution matched to the limits of visual acuity. *Journal of the Society for Information Display* 13 (February 2005).
- [4] BERGMAN, L., FUCHS, H., GRANT, E., AND SPACH, S. Image rendering by adaptive refinement. *SIGGRAPH Comput. Graph.* 20, 4 (1986), 29–37.
- [5] BIGLER, J., STEPHENS, A., AND PARKER, S. G. Design for parallel interactive ray tracing systems. In *IEEE Symposium on Interactive Ray Tracing* (2006), pp. 187–196.
- [6] BISHOP, G., FUCHS, H., MCMILLAN, L., AND ZAGIER, E. J. S. Frameless rendering: double buffering considered harmful. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM, pp. 175–176.
- [7] BOLIN, M. R., AND MEYER, G. W. A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 299–309.
- [8] BOULOS, S., EDWARDS, D., LACEWELL, J. D., KNISS, J., KAUTZ, J., SHIRLEY, P., AND WALD, I. Packet-based whitted and distribution ray tracing. In *GI '07: Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM, pp. 177–184.
- [9] BURT, P. J. Moment images, polynomial fit filters, and the problem of surface interpolation. In *Computer Vision and Pattern Recognition* (June 1988), IEEE Computer Society Press, pp. 144–152.
- [10] CAMPBELL, F. W., AND GREEN, D. G. Optical and Retinal Factors Affecting Visual Resolution. *Journal of Physiology* 181 (1965), 576–593.
- [11] COOK, R. L. Stochastic sampling in computer graphics. *ACM Trans. Graph.* 5, 1 (1986), 51–72.
- [12] COOK, R. L., PORTER, T., AND CARPENTER, L. Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM, pp. 137–145.
- [13] DAYAL, A. *Temporally Adaptive Rendering*. PhD thesis, Northwestern Univeristy, 2006.
- [14] DAYAL, A., WOOLLEY, C., WATSON, B., AND LUEBKE, D. P. Adaptive frameless rendering. In *Rendering Techniques* (2005), pp. 265–275.

- [15] DIETRICH, A., AND SLUSALLEK, P. Adaptive spatial sample caching. *Symposium on Interactive Ray Tracing 0* (2007), 141–147.
- [16] DIETRICH, A., WALD, I., BENTHIN, C., AND SLUSALLEK, P. The OpenRT Application Programming Interface - Towards a Common API for Interactive Ray Tracing. In *Proceedings of the 2003 OpenSG Symposium* (2003). Available at <http://www.openrt.de>.
- [17] DIPPE, M. A. Z., AND WOLD, E. H. Antialiasing through stochastic sampling. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM Press, pp. 69–78.
- [18] DRORI, I., COHEN-OR, D., AND YESHURUN, H. Fragment-based image completion. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM Press, pp. 303–312.
- [19] FERNANDO, R., FERNANDEZ, S., BALA, K., AND GREENBERG, D. P. Adaptive shadow maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 387–390.
- [20] FIELD, D. J. What the statistics of natural images tell us about visual coding. *Human Vision, Visual Processing, and Digital Display 1077* (1989), 269–276.
- [21] FIELD, D. J. Wavelets, vision and the statistics of natural scenes. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences 357*, 1760:2527 (1999).
- [22] FUNKHOUSER, T. A., AND SÉQUIN, C. H. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM, pp. 247–254.
- [23] GAUTRON, P., BOUATOUCH, K., AND PATTANAIK, S. N. Temporal radiance caching. *IEEE Trans. Vis. Comput. Graph.* 13, 5 (2007), 891–901.
- [24] GLASSNER, A. S. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.
- [25] GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 43–54.
- [26] GROSSMAN, J. P., AND DALLY, W. J. Point Sample Rendering. In *9th Eurographics Workshop on Rendering* (August 1998), pp. 181–192.
- [27] HAVRAN, V., DAMEZ, C., MYSZKOWSKI, K., AND SEIDEL, H.-P. An efficient spatio-temporal architecture for animation rendering. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 106–117.
- [28] HECKBERT, P. S. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM Press, pp. 145–154.
- [29] JOHNSON, G. S., LEE, J., BURNS, C. A., AND MARK, W. R. The irregular z-buffer: Hardware acceleration for irregular data structures. *ACM Trans. Graph.* 24, 4 (2005), 1462–1482.

- [30] KAJIYA, J. T. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 143–150.
- [31] KALLONIATIS, M., AND LUU, C. Psychophysics of vision. Published in document: <http://webvision.med.utah.edu/>.
- [32] KELLER, A. Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 49–56.
- [33] KILGARIFF, E., AND FERNANDO, R. The geforce 6 series gpu architecture. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), ACM, p. 29.
- [34] KRAUS, M., AND ERTL, T. Adaptive texture maps. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 7–15.
- [35] LAINE, S., SARANSAARI, H., KONTKANEN, J., LEHTINEN, J., AND AILA, T. Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007* (2007), Eurographics Association, pp. xx–yy.
- [36] LAMBRECHT, C., AND VERSCHEURE, O. Perceptual quality measure using a spatio-temporal model of the human visual system. In *SPIE* (1996), vol. 2668, pp. 450–461.
- [37] LEFOHN, A., SENGUPTA, S., KNISS, J. M., STRZODKA, R., AND OWENS, J. D. Dynamic adaptive shadow maps on graphics hardware. In *ACM SIGGRAPH 2005 Conference Abstracts and Applications* (Aug. 2005).
- [38] LUEBKE, D., WATSON, B., COHEN, J. D., REDDY, M., AND VARSHNEY, A. *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.
- [39] MALLAT, S. *A wavelet tour of signal processing*. Academic Press, 1998.
- [40] MATUSIK, W., PFISTER, H., BRAND, M., AND MCMILLAN, L. Efficient isotropic brdf measurement. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 241–247.
- [41] MITCHELL, D. Advanced topics in ray tracing, course notes,. In *SIGGRAPH '90: Material presented at the ACM SIGGRAPH 1990 conference* (New York, NY, USA, 1990), ACM Press.
- [42] MITCHELL, D. P. Generating antialiased images at low sampling densities. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 65–72.
- [43] MITCHELL, D. P., AND NETRAVALI, A. N. Reconstruction filters in computer-graphics. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM Press, pp. 221–228.
- [44] MOHSENIAN, N., RAJAGOPALAN, R., AND GONZALES, C. A. Single-pass constant- and variable-bit-rate MPEG-2 video compression. 489–509.
- [45] MONTRYM, J. S., BAUM, D. R., DIGNAM, D. L., AND MIGDAL, C. J. Infinitereality: a real-time graphics system. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 293–302.

- [46] MORTON, G. *A computer oriented geodetic data base and a new technique in file sequencing*. Internation Business Machines, Ottawa, 1966.
- [47] NEHAB, D., SANDER, P. V., LAWRENCE, J., TATARCHUK, N., AND ISIDORO, J. R. Accelerating real-time shading with reverse reprojection caching. In *GH '07: Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 25–35.
- [48] NG, R., RAMAMOORTHY, R., AND HANRAHAN, P. All-frequency shadows using non-linear wavelet lighting approximation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 376–381.
- [49] NVIDIA. *NVIDIA CUDA Programming Guide 2.0*. 2008.
- [50] OVERBECK, R., BEN-ARTZI, A., RAMAMOORTHY, R., AND GRINSPUN, E. Exploiting temporal coherence for incremental all-frequency relighting. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering* (June 2006), pp. 151–160.
- [51] PAINTER, J., AND SLOAN, K. Antialiased ray tracing by adaptive progressive refinement. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1989), ACM Press, pp. 281–288.
- [52] PALMER, S. E. *Vision Science*. MIT Press, 1999.
- [53] REINHARD, E., SHIRLEY, P., ASHIKHMIN, M., AND TROSCIANKO, T. Second order image statistics in computer graphics. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization* (New York, NY, USA, 2004), ACM Press, pp. 99–106.
- [54] ROHALY, A. M., LIBERT, J., CORRIVEAU, P., AND WEBSTER, A. Final report from the video quality experts group on the validation of objective models of video quality assessment. Tech. rep., 2000.
- [55] SCHMITTLER, J., WALD, I., AND SLUSALLEK, P. Saarcor: a hardware architecture for ray tracing. In *HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 27–36.
- [56] SITTHI-AMORN, P., LAWRENCE, J., YANG, L., SANDER, P. V., AND NEHAB, D. An improved shading cache for modern gpus. In *GH '08: Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2008), Eurographics Association, pp. 95–101.
- [57] SITTHI-AMORN, P., LAWRENCE, J., YANG, L., SANDER, P. V., NEHAB, D., AND XI, J. Automated reprojection-based pixel shader optimization. *ACM Trans. Graph.* 27, 5 (2008), 1–11.
- [58] SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 527–536.
- [59] SPRINGER, J. P., BECK, S., WEISZIG, F., REINERS, D., AND FROELICH, B. Multi-frame rate rendering and display. *Virtual Reality Conference, IEEE 0* (2007), 195–202.

- [60] SPRINGER, J. P., LUX, C., REINERS, D., AND FROEHLICH, B. Advanced multi-frame rate rendering techniques. In *VR (2008)*, pp. 177–184.
- [61] STEPHENS, A., BOULOS, S., BIGLER, J., WALD, I., AND PARKER, S. G. An Application of Scalable Massive Model Interaction using Shared Memory Systems. In *Proceedings of the 2006 Eurographics Symposium on Parallel Graphics and Visualization (2006)*, pp. 19–26. (to appear).
- [62] SÜSSTRUNK, S. E., AND WINKLER, S. Color image quality on the Internet. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series (Dec. 2003)*, S. Santini and R. Schettini, Eds., vol. 5304 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 118–131.
- [63] THIYAGALINGAM, J., BECKMANN, O., AND KELLY, P. H. J. Is morton layout competitive for large two-dimensional arrays yet. *Concurr. Comput. : Pract. Exper.* 18, 11 (2006), 1509–1539.
- [64] VELÁZQUEZ-ARMENDÁRIZ, E., LEE, E., BALA, K., AND WALTER, B. Implementing the render cache and the edge-and-point image on graphics hardware. In *GI '06: Proceedings of Graphics Interface 2006 (Toronto, Ont., Canada, Canada, 2006)*, Canadian Information Processing Society, pp. 211–217.
- [65] WALD, I., MARK, W. R., GÜNTHER, J., BOULOS, S., IZE, T., HUNT, W., PARKER, S. G., AND SHIRLEY, P. State of the Art in Ray Tracing Animated Scenes. In *Eurographics 2007 State of the Art Reports*.
- [66] WALTER, B., DRETTAKIS, G., AND GREENBERG, D. Enhancing and optimizing the render cache. In *In Proceedings of the 13th Eurographics Workshop on Rendering (2002)*, vol. 10.
- [67] WALTER, B., DRETTAKIS, G., AND PARKER, S. Interactive rendering using the render cache. In *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering) (New York, NY, Jun 1999)*, D. Lischinski and G. Larson, Eds., vol. 10, Springer-Verlag/Wien, pp. 235–246.
- [68] WANG, Z., BOVIK, A., SHEIKH, H., AND SIMONCELLI, E. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on* 13, 4 (April 2004), 600–612.
- [69] WANG, Z., LU, L., AND BOVIK, A. C. Video quality assessment based on structural distortion measurement. *Signal Processing: Image Communication* 19, 2 (2004), 121 – 132.
- [70] WESTERINK, P. H., RAJAGOPALAN, R., AND GONZALES, C. A. Two-pass MPEG-2 variable bit-rate encoding. 471.
- [71] WHITTED, T. An improved illumination model for shaded display. *Commun. ACM* 23, 6 (1980), 343–349.
- [72] WINKLER, S., AND MOHANDAS, P. The evolution of video quality measurement: From PSNR to hybrid metrics. *Broadcasting, IEEE Transactions on* 54, 3 (Sept. 2008), 660–668.
- [73] WISE, D. S., FRENS, J. D., GU, Y., AND ALEXANDER, G. A. Language support for morton-order matrices. In *PPoPP '01: Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming (New York, NY, USA, 2001)*, ACM, pp. 24–33.

- [74] WOOLLEY, C., LUEBKE, D., WATSON, B., AND DAYAL, A. Interruptible rendering. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics* (New York, NY, USA, 2003), ACM Press, pp. 143–151.
- [75] YANG, L., SANDER, P. V., AND LAWRENCE, J. Geometry-aware framebuffer level of detail. *Computer Graphics Forum (Special issue of Eurographics Symposium on Rendering 2008)* 27, 4 (2008), 1183–1188.
- [76] ZAGIER, E. J. S. Defining and refining frameless rendering. Tech. rep., Chapel Hill, NC, USA, 1996.