The Adaptive Particle Representation (APR) for Simple and Efficient Adaptive Resolution Processing, Storage and Simulations

DISSERTATION

Zur Erlangung des akademischen Grades
Doctor of Philosophy
(Ph. D.)

vorgelegt

der Fakultät Mathematik und Naturwissenschaften
der Technischen Universität Dresden

von

Bevan Leslie Cheeseman

geboren am 7. February 1987 in Auckland, New Zealand

Eingereicht am 11.08.2017
Die Dissertation wurde in der Zeit von 22.07.2013 bis 11.08.2017 im
Chair of Scientific Computing for Systems Biology, Faculty of Computer
Science, TU Dresden
Center for Systems Biology Dresden, Max Planck Institute of Molecular
Cell Biology and Genetics angefertigt.

Gutachter1: Prof. Dr. Ivo F. Sbalzarini
Gutachter2: Prof. Dr. Jens H. Walther

Abstract

This thesis presents the Adaptive Particle Representation (APR), a novel adaptive data representation that can be used for general data processing, storage, and simulations. The APR is motivated, and designed, as a replacement representation for pixel images to address computational and memory bottlenecks in processing pipelines for studying spatiotemporal processes in biology using Light-sheet Fluorescence Microscopy (LSFM) data.

The APR is an adaptive function representation that represents a function in a spatially adaptive way using a set of Particle Cells $\mathcal{V}$ and function values stored at particle collocation points $\mathcal{P}^*$. The Particle Cells partition space, and implicitly define a piecewise constant Implied Resolution Function $R^*(\mathbf{y})$ and the spatial locations that particles are sampled. As an adaptive data representation, the APR can be used to provide both computational and memory benefits by aligning the number of Particle Cells and particles with the spatial scales of the function. The APR allows reconstruction of a function value at any location $\mathbf{y}$ using any positive weighted combination of particles within a distance of $R^*(\mathbf{y})$. The Particle Cells $\mathcal{V}$ are selected such that the error between the reconstruction and the original function, when weighted by a function $\sigma(\mathbf{y})$, is below a user-set relative error threshold E. We call this the Reconstruction Condition and $\sigma(\mathbf{y})$ the Local Intensity Scale. $\sigma(\mathbf{y})$ is motivated by local gain controls in the human visual system, and for LSFM data can be used to account for contrast variations across an image.

The APR is formed by satisfying an additional condition on $R^*(\mathbf{y})$; we call the Resolution Bound. The Resolution Bound relates the $R^*(\mathbf{y})$ to a local maximum of the absolute value function derivatives within a distance $R^*(\mathbf{y})$ or $\mathbf{y}$. Given restrictions on $\sigma(\mathbf{y})$, satisfaction of the Resolution Bound also guarantees satisfaction of the Reconstruction Condition. In this thesis, we present algorithms and approaches that find the optimal Implied Resolution Function to general problems in the form of the Resolution Bound using Particle Cells using an algorithm we call the Pulling Scheme. Here, optimal means the largest $R^*(\mathbf{y})$ at each location. The Pulling Scheme has worst-case linear complexity in the number of pixels when used to represent images. The approach is general in that the same algorithm can be used for general (α,m)-Reconstruction Conditions, where α denotes the function derivative and m the minimum order of the reconstruction. Further, it can also be combined with anisotropic neighborhoods to provide adaptation in both space and time.

The APR can be used with both noise-free and noisy data. For noisy data, the Reconstruction Condition can no longer be guaranteed, but numerical results show an optimal range of relative error E that provides a maximum increase in PSNR

over the noisy input data. Further, if it is assumed the Implied Resolution Function satisfies the Resolution Bound, then the APR converges to a biased estimate (constant factor of E), at the optimal statistical rate.

The APR continues a long tradition of adaptive data representations and represents a unique trade off between the level of adaptation of the representation and simplicity. Both regarding the APRs structure and its use for processing. Here, we numerically evaluate the adaptation and processing of the APR for use with LSFM data. This is done using both synthetic and LSFM exemplar data. It is concluded from these results that the APR has the correct properties to provide a replacement of pixel images and address bottlenecks in processing for LSFM data. Removal of the bottleneck would be achieved by adapting to spatial, temporal and intensity scale variations in the data. Further, we propose the simple structure of the general APR could provide benefit in areas such as the numerical solution of differential equations, adaptive regression methods, and surface representation for computer graphics.

Contents

1 Thesis Introduction 3
1.1 Format of thesis . . . 3
1.1.1 Reading aids . . . 3
1.1.2 Notes regarding language . . . 4
1.1.3 Statistical data and error bars . . . 4

2 APR Motivation 5
2.1 Towards studying spatio-temporal processes in biology . . . 6
2.2 Using Light Sheet Fluorescence Microscopy (LSFM) to track objects . . . 7
2.2.1 Flouresence as a localisation technique . . . 7
2.2.2 Sampling notation . . . 8
2.2.3 Imaging a sample . . . 9
2.2.4 Image formation . . . 9
2.2.5 Processing on images . . . 11
2.3 The problem . . . 13
2.4 Current approaches to processing for LSFM . . . 17
2.4.1 Concepts . . . 18
2.4.2 Summary . . . 20
2.5 Motivation from the visual system . . . 21
2.6 Summary and main points . . . 22

3 Representation criteria and previous work 25
3.1 Introduction . . . 25
3.2 Representation Criteria (RC) . . . 26
3.2.1 Representation Criteria 1 (RC1) . . . 27
3.2.2 Representation Criteria 2 (RC2) . . . 28
3.2.3 Representation Criteria 3 (RC3) . . . 28
3.2.4 Representation Criteria 4 (RC4) . . . 29
3.2.5 Representation Criteria 5 (RC5) . . . 29
3.3 Review of existing multi-resolution and adaptive representations 30

3.3.1 Processing augmentation techniques 31
3.3.2 Adaptive sparse transform domain techniques 36
3.3.3 Adaptive sparse collocation techniques 40
3.4 Discussion 47
3.4.1 Lack of local gain control (RC1) 47
3.4.2 Use across a wide range of tasks (RC4) 47
3.5 Summary and main points 48

4 The Adaptive Particle Representation (APR) 49
4.1 Adaptive Particle Representation (APR) 50
4.1.1 Main results 51
4.1.2 Motivation of formulation 52
4.1.3 Extensions 53
4.2 APR description 54
4.2.1 Reconstruction Condition and Resolution Bound 54
4.2.2 Particle Cells and Implied Resolution Functions 59
4.2.3 Pulling Scheme 63
4.2.4 Choosing the Particle Set $\mathcal{P}$ 68
4.2.5 Forming the APR=$\{\mathcal{V}, \mathcal{P}^*\}$ 71
4.2.6 Summary 71
4.3 Practical considerations 71
4.3.1 Discrete sampling 72
4.3.2 Impact of noisy Local Resolution Estimate $L(\mathbf{y})$ 72
4.3.3 Impact of noisy particles $\hat{f}(y_p)$ 73
4.3.4 Comparison with continuous resolution functions 74
4.3.5 Particle Cells and smoothness of the Local Intensity Scale 75
4.3.6 Reconstruction of higher order derivatives 75
4.4 APR transform steps summary 76
4.5 Summary and main points 77

5 General Dimension APR and Technical Details 81
5.1 General dimension APR 82
5.1.1 Result 1 83
5.1.2 Result 2 83
5.1.3 Result 3 83
5.2 Reconstruction Condition and Resolution Bound 83
5.3 Particle Cells and the Implied Resolution Function 85
5.3.1 Particle Cell definitions 85
5.3.2 Implied Resolution Function 88
5.3.3 Local Particle Cell set 89
5.3.4 Optimal Valid Particle Cell sets 90

5.4 Pulling Scheme 92
5.4.1 Self-similarity and production of individual solutions . . 92
5.4.2 Separability 94
5.4.3 Redundancy of Particle Cells 96
5.4.4 Equivalence 96
5.4.5 Algorithm description 100
5.4.6 Generic algorithm 100
5.4.7 Memory and computational complexity 101
5.5 Particle sampling 104
5.5.1 APR as $\{\mathcal{V}_n, \mathcal{P}^*\}$ 105
5.5.2 Optimality 105
5.5.3 Integral neighborhood optimization 106
5.6 Technical additions 108
5.6.1 Function reconstruction and interpolation 109
5.6.2 APR particle graph 110
5.7 Summary and main points 111

6 APR Validation 113
6.1 1D benchmarks 115
6.1.1 Implementation 115
6.1.2 1D example 115
6.1.3 Reconstruction Condition 117
6.1.4 Numeric vs. symbolic gradient 119
6.1.5 Number of particles 120
6.1.6 Gradient 120
6.1.7 Discontinuities 121
6.1.8 Summary 122
6.2 3D florescence image implementation 122
6.2.1 Gradient estimation $|\nabla I|$ 123
6.2.2 Local Intensity Scale σ 124
6.2.3 Intensity estimation I_p 125
6.2.4 Reconstruction methods 125
6.2.5 Pipeline and parameters 125
6.3 3D synthetic data 126
6.3.1 Object function 126
6.3.2 Image formation 127
6.3.3 Summary 128
6.4 APR properties benchmarks 129
6.4.1 Noise-free Reconstruction Condition 130
6.4.2 Noise corrupted Reconstruction Condition 132
6.4.3 Increasing information content 135

        6.4.4 Increasing image size 136
        6.4.5 Increasing sampling rate 136
    6.5 APR performance benchmarks 137
        6.5.1 Computational Ratio benchmark data 139
        6.5.2 Exemplar benchmark data 139
        6.5.3 Data structures 140
        6.5.4 Execution time 141
        6.5.5 Memory Cost 148
        6.5.6 Storing the APR 148
    6.6 Summary and main points 150
        6.6.1 Limitations of the benchmark results 151
        6.6.2 Reflection on Representation Criteria 152

7 APR Processing 157
    7.1 Interpretations of the APR for processing 158
        7.1.1 Collocation points and spatial partition 158
        7.1.2 Particle graph 158
        7.1.3 Continuous representation 160
        7.1.4 Tree structure 160
    7.2 Evaluating performance 160
    7.3 Performance benchmarks 162
        7.3.1 Limitations 162
        7.3.2 Neighbor access 164
        7.3.3 Separable pixel filtering 166
        7.3.4 Segmentation 171
    7.4 Visualization 175
        7.4.1 By slice 175
        7.4.2 By racasting 177
        7.4.3 By particle rendering 177
    7.5 Novel algorithms 180
        7.5.1 Adaptive APR filters 180
        7.5.2 Segmentation and visualization 183
    7.6 Summary and main points 185
        7.6.1 Reflection on Representation Criteria 185

8 Comparison with previous methods and optimality 189
    8.1 Similarity to other methods 190
        8.1.1 APR form and Resolution Function 190
        8.1.2 Reconstruction Condition 191
        8.1.3 Local Intensity Scale 191
        8.1.4 Resolution Bound 191

8.1.5 Particle Cells and Implied Resolution Function 192
8.1.6 Pulling Scheme 192
8.1.7 Wavelet thresholding 193
8.1.8 Computational and Memory Cost 194
8.1.9 Summary and originality 194
8.2 Comparison to wavelet optimality 195
8.2.1 Optimal error convergence 195
8.2.2 Optimal representation of noisy signals 197
8.3 Optimal ϵ convergence of the APR 204
8.3.1 APR reconstruction 207
8.3.2 Result summary 210
8.4 Summary 211

9 APR extensions in space 213
9.1 Higher-order reconstruction APR 214
9.1.1 General derivative conditions 215
9.1.2 Multiple resolution conditions 217
9.2 1D validation of higher-order reconstructions 217
9.2.1 Adding higher order reconstructions 217
9.2.2 Restricted higher order reconstructions 218
9.2.3 Derivative reconstruction 220
9.2.4 Summary 221
9.3 Potential use-cases 221
9.3.1 Computational partial differential equations 222
9.3.2 Using the APR without a Reconstruction Condition . . . 224
9.3.3 Anisotropic APR neighborhood 225
9.4 Summary and main points 226

10 Extensions in time (APR$_t$) 227
10.1 APR$_t$ 228
10.1.1 Theory 228
10.1.2 Algorithm 229
10.2 Benchmark 1D 230
10.2.1 Time Reconstruction Condition 231
10.2.2 Temporal adaptation 232
10.2.3 Relative size of update, add and remove 235
10.2.4 Summary 235
10.3 Extension to LSFM data 237
10.3.1 Complication of noise 237
10.3.2 Real-time optimizations 237
10.4 Summary and main points 237

11 Outlook and Discussion **239**
11.1 Executive summary 239
11.1.1 Chapter Result Summary 243
11.2 The APR for representation of LSFM data 246
11.2.1 Critical evaluation of the Representation Criteria 247
11.2.2 Conclusions and outlook 253
11.3 APR as a general purpose adaptive data representation 254
11.3.1 Other image modalities 254
11.3.2 Numerical solution of differential equations 254
11.3.3 Statistical adaptive regression 255
11.3.4 Surface approximation in computer graphics 255
11.4 Concluding remark 256

Bibliography **257**

A Appendix **272**
A.1 Thought experiment: eye as a camera 272
A.2 Besov space definition 272
A.2.1 Reconstruction Condition vs. Resolution Bound 273
A.2.2 Bounds for Implied Resolution Function 273
A.2.3 Bounds on particle sampling 274
A.2.4 Optimal continuous $\mathcal{O}(N^{\frac{d-1}{d}+1})$ algorithm 277
A.3 Impact of noisy Local Resolution Estimate $L(\mathbf{y})$ 278
A.4 APR 3D processing pipeline 280
A.4.1 Implementation 280
A.5 Data structures 287
A.5.1 Sparse APR (SA) data structure 288
A.5.2 Sparse APR Random Access data structure 290
A.6 APR writing, output and compression 292
A.7 Pipeline parameter Summary 292
A.7.1 Image parameters 293
A.7.2 Reconstruction parameters 293
A.8 Synthetic image generation 295
A.8.1 Implementation 295
A.8.2 Template image 295
A.8.3 Ground truth image 295
A.8.4 Original (noisy) image 296
A.8.5 Benchmark parameter selection 296
A.9 Evaluation benchmarks 297
A.9.1 Noise-free Reconstruction Condition 297
A.9.2 Reconstruction error (noisy) 298

A.9.3 Reconstruction image quality (noisy) 298
A.9.4 Increasing information content 299
A.9.5 Increasing image size 300
A.9.6 Increase sampling 300
A.10 Benchmark data 300
A.10.1 Computational Ratio (CR) benchmark data 301
A.10.2 Exemplar datasets 301
A.10.3 Performance benchmarks 301
A.10.4 Neighbor access 304
A.10.5 Separable pixel filtering 305
A.10.6 Graph cuts segmentation 306
A.10.7 Adaptive APR Filters 308
A.11 APR visualization examples 309
A.11.1 2D slice reconstruction 309
A.11.2 Perspective ray-cast 309
A.11.3 Direct particle rendering 310

Declaration **311**

Main Acronyms

- Adaptive Particle Representation (APR) (See Section 4.1 on Page 50 for 1D and Section 5.1 on Page 82 for general)
- Light-sheet Fluorescence Microscopy (LSFM) (See Section 2.2.1 on Page 7)
- Spatiotemporal processes in biology (STB) (See Section 2.1 on Page 6)
- Representation Criteria (**RC**) (See Section 3.2 on Page 26)
- Computational Ratio (CR) (See Section 6.11 on Page 138)
- Memory Compression Ratio (MCR) (See Section 6.12 on Page 149)
- Pixel to Particle Speed Ratio (PP) (See Section 7.3 on Page 161)
- Speed Up (SU) (See Section 7.1 on Page 160)
- Sparse APR Random Access Data Structure (SARA) (See Section 6.5.3 on Page 140 and Section A.5.2 on Page 290)
- Sparse APR Data Structure (SA) (See Section 6.5.3 on Page 140 and Section A.5.1 on Page 288)
- Mean Squared Error (MSE) (See Section 6.10 on Page 134)
- Peak Signal to Noise Ratio (PSNR) (See Section 6.9 on Page 133)

Key concepts and definitions

- Particle Cells (See Section 4.2.2 on Page 60 for 1D and Section 5.3 on Page 85 for general)
- Resolution Function $R(\mathbf{y})$ (See Section 4.1 on Page 50 for 1D and 5.1 for general)
- Implied Resolution Function $R^*(\mathbf{y})$ (See Section 4.2.2 on Page 60 for 1D and Section 5.3.2 on Page 88 for general)
- Optimal continuous Resolution Functions ($R_b(\mathbf{y})$ and $R_c(\mathbf{y})$) (See 4.2.1,Section 4.3.4 on Page 74 and Section A.2.1 on Page 273)

- Local Resolution Estimate $L(\mathbf{y})$ (See Section 4.2.1 on Page 57 for 1D and Section 5.13 on Page 85 for general for classic case. See Section 9.17 on Page 216 for (α, m) extensions)
- Reconstruction Condition (See Section 4.2 on Page 50 for 1D and 5.2 for general)
- Resolution Bound (See Section 4.3 on Page 51 for 1D and Section 5.3 on Page 83 for general)
- Local Intensity Scale $\sigma(\mathbf{y})$ (See Section 4.2.1 on Page 56 (1D) and Section 5.11 on Page 84 (general) for continuous assumption and Section 4.3.5 on Page 75 for effective APR assumption)
- Local Particle Cell (LPC) set $\mathcal{L}$ (See Section 4.19 on Page 62 for 1D and Section 5.28 on Page 89 for general)
- Optimal Valid Particle Cell (OVPC) set $\mathcal{V}$ (See Section 4.2.2 on Page 62 for 1D and Section 5.28 on Page 89 for general)
- Particle sampling $\mathcal{P}^*$ (See Section 4.1.1 on Page 52 in 1D and Section 5.1.3 on Page 83 for general definition and Section 4.2.4 on Page 68 and Section 5.5 on Page 104 for sampling strategies)
- Pulling Scheme (See Section 4.2.3 on Page 63 in 1D and Section 5.4.5 on Page 100 in general and Section A.4.1 on Page 285 for implimentation details.)

Important variables

- $f\{\bar{x}\}$ sampled function (See Section 2.2.2 on Page 8)
- $f(y)$ continuous function (See Section 2.2.2 on Page 8)
- N_p number of particles in APR.
- N number of pixels in original image (See Section 2.2.2 on Page 8)
- Ω spatial domain
- d spatial dimension
- $\mathcal{N}$ Interaction Neighborhood (See Section 4.1 on Page 50 in 1D and 5.1 in general)

- E Relative error bound (See Section 4.2 on Page 50 in 1D and 5.2 in general)
- E^* Observed reconstruction error (See Section 6.8 on Page 130)
- $c_{\mathbf{i},l}$ particle cell at level l and location $\mathbf{i}$. (See Section 5.3 on Page 85)

Foreword and Acknowledgments

This thesis represents the culmination of over four years of work. It was a privilege to be given the opportunity by Professor Ivo F. Sbalzarini to be able to explore my ideas with both support and freedom. This journey has been filled with both many highs and similarly many lows and sleepless nights. A key theme for me has been the constant struggle to find a balance between thinking and doing. However, it is through this struggle that the core ideas of my work arose.

As is often the case in science, the origin of this project and its ideas are steeped in serendipity. As part of my Master's program at the University of Melbourne, I took a course with Associate Professor Steve Carnie on the computational solution of differential equations. As part of this course, I had to make a selection of an advanced topic to review and present. From a list, the selection I made was the adaptive solution of partial differential equations using the method of adaptive method of lines. It seems from this simple decision that the inspiration for my eventual thesis topic grew. I also owe thanks to Professor Jan Huisken, who, on top of providing valuable feedback through my yearly TAC meetings, presented the problem of processing of LSFM data to me in the selection week of the Dresden Ph.D. program. Coincidently, in concurrent meetings at the selection week with Professor Ivo F. Sbalzarini, my now supervisor, presented his own research on adaptive Lagrangian particle methods. It is from these interactions, and my fond memories of presenting adaptive methods in my masters, that the desire to apply adaptive methods from computational differential equations to LSFM data arose.

There are many people that I need to thank because they helped me through these four years and supported me both personally and professionally. First, thank you to the MOSAIC group and alumni who shared some, or all, of my last four years in Dresden. They have provided a source of ideas, inspiration, and lasting friendship that has been invaluable. Scientifically, I would like to thank Dr. Michael Hecht for his discussions and help with mathematical notation, Ulrik Günther for his discussions and collaboration on visualization, and Mattius Sasuik for his help with parallelization and code de-

velopment. Again, I would like to thank Professor Ivo F. Sbalzarini for leading the group and his support, enthusiasm, and gift of freedom throughout my research. I would also like to thank DIGS-BB for their funding. The scholarship support they provided allowed for the purchase of a benchmark machine ('The Particulator') which was invaluable to my work. A special thank you to all research groups that provided LSFM data for which I could use to benchmark and develop this work. This was an invaluable resource, and without it, the project would not have developed. In particular, thanks to the Huisken Lab (Especially Stephan Daetwyler), and various members of the Tomancak Lab at MPI-CBG for allowing me access to their data. I also extend my appreciation and thanks to Professor Jens Walther for agreeing to review this work.

Lastly, and importantly, I would like to thank my family and friends for putting up with me and my obsessions and argumentative bent. Special thanks to my parents and the extended Cheeseman clan, for their support through the years and putting up with their absent and forgetful son, brother, and uncle. Moving to Dresden and Germany has presented its challenges, but has been made worth it for the friends I have made during this time. Most of all, it presented the opportunity to meet Maj Svea, who has been a constant source of support, escape, and shared joy over the last few years.

1 Thesis Introduction

In this thesis, we introduce the Adaptive Particle Representation (APR). The APR is a novel adaptive data representation designed to reduce computational and memory costs of processing tasks. The APR is a general adaptive data representation. However, it was motivated by and developed for the high-throughput processing of Light-sheet Fluorescence Microscopy (LSFM) data for the study of spatiotemporal processes in biology (STB).

1.1 Format of thesis

This thesis is structured as follows. In Chapter 2 we motivate the use of an adaptive data representation for studying STB using LSFM data. In Chapter 3, we discuss the desired properties of such an adaptive data representation and review existing methods. In Chapter 4, we introduce the APR and its core ideas, concepts and algorithms using 1D for didactic exposition. Complimenting this, in Chapter 5 we then provide a general dimensional treatment of the APR, including all technical details and proofs of the main results. In Chapter 6, we then empirically validate and explore the properties of the APR, focusing on the representation of LSFM data. In Chapter 7, we evaluate the memory and computational performance of using the APR for image processing tasks. In Chapter 8, we then discuss the similarities of the APR with existing adaptive representations and reflect on optimality results from wavelet thresholding. In Chapter 9, we discuss extensions of the classic APR to more general representations and briefly explore applications. Then in Chapter 10, we provide an extension of the APR to adapt in time and provide preliminary results. In Chapter 11, we conclude by summarizing and the critically evaluating the results of this thesis and discussing future work and applications.

1.1.1 Reading aids

At the end of each chapter, to aid the reader, we provide a conclusion and summary of the main results of the chapter in a table format. Any information that was deemed important but did not fit the flow of the text was placed in the Appendix (A) following the references. Due to the large size of this thesis, it is recommended that for reading in pdf form, such that the hyperlinks can be used for navigation between different areas of this thesis. Also, for reference at the beginning of this thesis, we have provided a summary of key terms and abbreviations that are frequently used in this thesis and provided links to their appropriate definitions in the text.

1.1.2 Notes regarding language

Throughout this thesis, I use extensive use of ”we”. The use of ”we” was a stylistic choice and is intended in the sense of you the reader, and myself. In all cases, unless explicitly stated, the work presented was my own work. I do also make use of ”I”, this usually restricted to situations where I wish to make it clear that I am expressing a personal opinion. I also employ liberal use of casual statements, words, or phrases in single commas ”. These statements are not meant to be interpreted literally but are used in an attempt to communicate the ’gist’ of an idea in a more concise way.

1.1.3 Statistical data and error bars

Empirical results are presented in this these both, with, and without error bars. In the case where error bars are included, they represent the estimated standard error of the data. However, more often than not, they have been omitted. Where omitted, I have made a judgment that the error estimates were not meaningful in the given context.

2 APR Motivation

Contents

2.1 Towards studying spatio-temporal processes in biology 6

2.2 Using Light Sheet Fluorescence Microscopy (LSFM) to track objects 7

2.2.1 Flouresence as a localisation technique 7

2.2.2 Sampling notation 8

2.2.3 Imaging a sample 9

2.2.4 Image formation 9

2.2.5 Processing on images 11

2.3 The problem 13

2.4 Current approaches to processing for LSFM . . . 17

2.4.1 Concepts 18

2.4.2 Summary 20

2.5 Motivation from the visual system 21

2.6 Summary and main points 22

In this chapter, we introduce the motivation, and scope, of the work presented in this thesis and formulate the scientific problem addressed. We begin by describing the need for high-resolution spatiotemporal data to study spatiotemporal processes in biology (STB) at the cellular level. We then introduce Light-sheet Fluorescence Microscopy (LSFM) and describe how the image data could be used as the basis for studying STB. Following this, we highlight, how features of LSFM data, result in processing the data becoming a critical bottleneck to its use. These features include the large data size and dynamic spatial, temporal and local intensity scales in the data. We then review current approaches to processing LSFM data and their common features. From this, we highlight the need for an alternative representation for LSFM image data that can be used across specimens and processing tasks. Lastly, we discuss the human visual system, as an example of features of a desired general alternative representation, and seek inspiration from its features. Namely, the use of adaptive sampling and a local gain control.

2.1 Towards studying spatio-temporal processes in biology

Gaining a mechanistic understanding of spatiotemporal processes in biology (STB) requires the ability to both observe and quantify these processes throughout time and space [80]. These observations and quantifications are essential for both hypothesis generation, and testing. Such quantification requires the capacity to track and observe specific structures involved in a process in a live developing organism consistently in space (3D) and time. For example, understanding of the spatiotemporal development of a specific organ, *e.g.* the gut, requires the ability to track all cells' movement, division, and lineage relationships, simultaneously through time across multiple specimens [8]. To achieve this, high-resolution data in space and time is required. Although useful, data from individual cell tracking, or, spatial data at distinct time points, do not provide a sufficient description of the spatiotemporal dynamics, especially in the presence of stochasticity [28]. Unfortunately, attaining high-resolution spatiotemporal data of developing processes is difficult, and only recently possible [136, 49].

Fortunately, recent developments in fluorescence microscopy [60], chemistry [87], and genetics [65] now provide tools that have the promise of allowing researchers to extract high-resolution spatiotemporal data for a broad range of specimens and processes in a high-throughput manner [108, 118]. However, these fluorescence microscopes do not directly output the shape, and location, of objects through time, instead they produce raw data (3D images through time) from which the desired spatiotemporal information is extracted through computational processing. This field is often called Image-based Systems Biology [106]. This step of extracting information from the raw microscopy data is non-trivial and currently creates a significant bottleneck for research into STB [136, 96, 108].

In this thesis, I develop a novel data representation, namely the Adaptive Particle Representation (APR), designed to alleviate the bottleneck in the processing of LSFM data and enable future research into spatiotemporal processes in biology. The next section provides a brief introduction to the required concepts from fluorescent microscopy and image formation necessary to understand this work.

2.2 Using Light Sheet Fluorescence Microscopy (LSFM) to track objects

Light Sheet Fluorescence Microscopy (LSFM) is currently the most promising and growing class of microscopes for studying STB [86, 118]. This fit and popularity are due to LSFM's abilities to image with both high spatial and temporal resolution deep into tissues while simultaneously causing limited photodamage to the specimen [60]. Further, they allow long-term imaging of developing embryos. There has been considerable specialization and development across different types of LSFM, *e.g.* [67, 29, 99], providing improvements and trade-offs, we direct the reader to the following recent comprehensive reviews for an overview [86, 118]. However, the basic principles of the original Selective Plane Illumination Microscope (SPIM), introduced by Huisken et al. [60] hold across these microscopes, and its description is used here to illustrate the general principles of modern fluorescence microscopy as used to study STB.

2.2.1 Flouresence as a localisation technique

Fluorescent microscopes, such as LSFM, allow the inference of the location of objects through space and time using the localization of fluorescent molecules. Fluorescent molecules can be localized, or stuck, to structures of interest in a specimen through the use of genetic or chemical techniques. For example, for development biology embryos can be genetically engineered to attach fluorescent proteins on a particular structure of all of their cells, such as a cell's nuclei, as it develops. A cell usually has one nucleus per cell located in its interior that often has a spherical shape making them ideal for tracking. The fluorescent molecules attached to the nuclei become 'excited' when exposed to a light of particular wavelengths, causing them to emit light at different (longer) wavelengths. The location and shape of the labeled structures can then be inferred, by observing, or recording the spatial distribution of the brightness (intensity) of the emitted light signal.

LSFMs allow the localization of the emitted light signal in 3D in high spatial and temporal resolution deep into samples using a technique known as *optical sectioning* that can be then used to infer position and shape of labeled objects. To understand how this process works, we will use a simplified example: consider the task of tracking a population of M cells through time for a particular specimen and developing process. For simplicity, we can ignore cell death (removal of cells), or cell division (addition of cells). Therefore, we wish to find the set of functions $C = \{x_1(t), x_2(t), ..., x_M(t)\}$, where $x_i \in \Omega \subset \mathbb{R}^3$, which for all $t \in \Omega_t \subset \mathbb{R}$ tells us the 3D location of the centers of all M

cells. As we cannot directly observe C, we first label the cell nuclei densely with a fluorescent molecule. We assume the labeling is dense enough that we can represent it as a piece-wise continuous function with compact support $O : \Omega \times \Omega_t \rightarrow \mathbb{R}$. We will call this the Object function. In our simple example, let

$$O(x, y, z, t) = \sum_{i=1}^{M} O_i(x, y, z, t) \tag{2.1}$$

where $O_i(x, y, z, t)$ is a piece-wise constant function, non-zero only at the locations occupied by the labeled cell nuclei i. Given the cells are distinct, we assume that there is no overlap between the support of the individual $O_i(x, y, z, t)$. Then, given $O(x, y, z, t)$, finding the cell tracking C, would entail identifying and tracking all the compactly supported regions $O_i(x, y, z, t)$ through time, and defining a suitable center. However, we cannot directly observe $O(x, y, z, t)$ from the microscope. Instead, fluorescence microscopes allow us to obtain regularly sampled image data I_{3D+t} that we can process to estimate $O(x, y, z, t)$.

2.2.2 Sampling notation

We now introduce some notation to allow us to deal with functions, pixel images, and adaptive representations. Although some definitions may be obvious, we define them here to remove ambiguity.

As above, when dealing with a function, round brackets, for example for $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, then $f(x, y)$ denotes the function f evaluated at (x, y). Later, we also use $f(\mathbf{y})$ and multi-index notation, where for the same example in 2D, $\mathbf{y} = (x, y)$ and $f(x, y) = f(\mathbf{y})$.

We consider a pixel image as the set generated from evaluating a function at a set of regularly spaced collocation points. For example, we define a 1D image of size N with spatial resolution h sampled from $[a, b]$ from some function $f : [a, b] \rightarrow \mathbb{R}$ as

$$f\{\bar{x}\} = \{f(x_i) | x_i \in \bar{x}\} \tag{2.2}$$

where $\bar{x} = \{x_i | x_i = a + h(i-1), i = 1, 2, .., N\}$ are the evenly spaced collocation points, and $h = (b - a)/N$. The notation holds not just for pixel images, but also for arbitrary vectors and samplings with $\bar{x} \in \mathbb{R}^N$. To avoid ambiguity with vector norms, we denote the cardinality of a set by $\#$, so in the 1D image example we have $\#\bar{x} = \#f\{\bar{x}\} = N$.

Pixel images in higher dimensions follow the convention as follows, with a 3D+t pixel image, being defined for $f : [a_x, b_x] \times [a_y, b_y] \times [a_z, b_z] \times [a_t, b_t] \rightarrow$

$\mathbb{R}$, being defined as $f\{(\bar{x},\bar{y},\bar{z},\bar{t})\} = \{f(x_i,y_i,z_i,t_i)|(x_i,y_i,z_i,t_i) \in \bar{x}\times\bar{y}\times\bar{z}\times\bar{t}\}$. Where the number of samples in each direction is N_x, N_y, N_z, N_t, with h_x, h_y, h_z, h_t, for the x, y, z, t directions respectively. Therefore, the total number of samples in the pixel image $N = \#f\{(\bar{x},\bar{y},\bar{z},\bar{t})\} = N_xN_yN_zN_t$. When written as $f\{x,y,z,t\}$, where the arguments are not explicitly vectors, or defined constants, the expression can be interpreted as to hold for any of sampled collocation points.

2.2.3 Imaging a sample

Figure 2.1 shows a schematic of a standard SPIM microscope, a type of LFSM, the figure has been reproduced from Huisken et al. [60]. In the schematic, the labeled specimen is represented by the bright green fish inside a chamber filled with an agarose gel that keeps it in place while allowing the sample to grow. The image data $I\{\bar{x},\bar{y},\bar{z},\bar{t}\}$ is produced one 2D (x,y) plane at a time. Where we achieve optical sectioning of one plane by illuminating a given (x,y) section with a thin sheet of light. The labeled fluorescent molecules in this plane, are excited, and emit light at a different wavelength that is focused perpendicular to the light sheet on a camera (shown in the figure by the detection arrow). The signal is integrated over a fixed time and area of size $h_x \times h_y$ and used to form an image $I\{\bar{x},\bar{y},z_0,t_0\}$ for a fixed z_0 and time t_0. The size of $h_x = h_y$, and is set by the effective area of the sensor on the camera chip. This integrated signal is known as the *intensity.* The full volume of the specimen is then regularly sampled, by moving the sample stepwise with displacement h_z so that for each step a new plane of the sample is illuminated and imaged. The series of images $I\{\bar{x},\bar{y},z_i,t_0\}$ are combined into a stack, i.e. $I\{\bar{x},\bar{y},\bar{z},t_0\} = \bigcup_{i=0}^{N_z} I\{\bar{x},\bar{y},z_i,t_0\}$ and treated as a 3D image (Shown in bottom left of Figure 2.1). We ignore the difference in time t_0 between imaging each slice. This process is then repeated at regular intervals in time, h_t, for N_t steps to produce the 3D+t image dataset $f\{(\bar{x},\bar{y},\bar{z},\bar{t})\} = \bigcup_{i=0}^{N_t} I\{\bar{x},\bar{y},\bar{z},t_i\}$. The signal is scaled and quantized most often to a 16bit integer, given it a range $\{0, 65536\}$

2.2.4 Image formation

The relationship between the 3D+t image dataset $I\{x,y,z,t\}$ at each location and our desired object function $O(x,y,z,t)$ can be modelled in the following

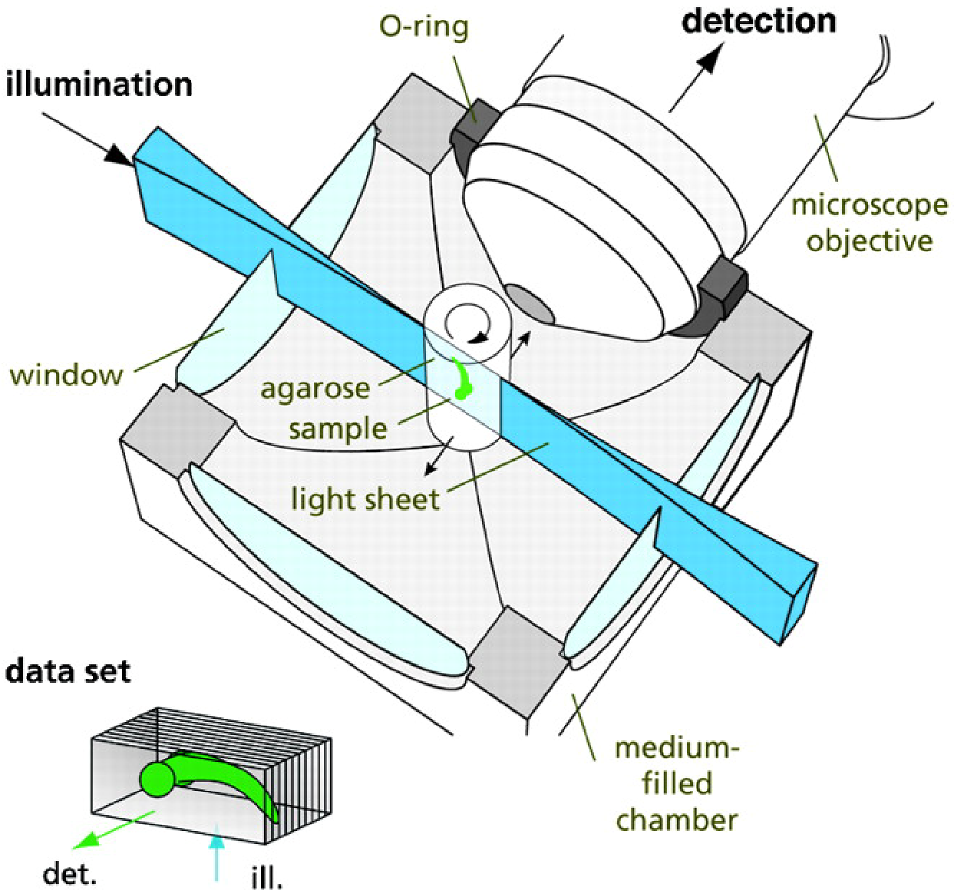


Figure 2.1: The figure shows a schematic of the optical setup for a Selective Plane Illumination Microscope (SPIM), the original design for a Lightsheet Fluorescence Microscope (LSFM) (Reproduced from Huisken et al. [60]). First, the labeled sample is secured in a movable sample chamber using agarose gel. A thin 2D sheet of laser light of fixed wavelength is then used to illuminate only a thin section of the specimen. The labeled structures in the fluorescent sample that the 2D sheet passes through are excited, emitting light of another wavelength that then is collected by the detection path (perpendicular to the light sheet) and recorded by a camera. The fluorescence signal is integrated over a fixed time and is used to construct a 2D image. This image gives a read out of the spatial distribution of the fluorescence label within the 2D sheet. The sample can then be moved step-wise through the light sheet, illuminating sections of the sample, slice by slice. When done successively, at a fixed interval, the sequence of 2D images, or slices, known as a *stack*, is computationally combined to create a 3D image, as shown in the lower left corner of the figure.

way

$$I^*(x, y, z, t) = \iiint_\Omega (O(u, v, w, t) + \\ b(u, v, w, t)) PSF(x, y, z, t, x - u, y - v, z - w) dudvdw \tag{2.3}$$

and then sampled at pixel locations as

$$I\{x, y, z, t\} = \int_{x-h_x/2}^{x+h_x/2} \int_{y-h_y/2}^{y+h_y/2} \int_{t}^{t+\delta_t} I^*(u, v, z, s) dudvds + \eta(x, y, z, t) \tag{2.4}$$

where for our example $\Omega \subset \mathbb{R}^3$ would be all points in the sample chamber, $b(u, v, w, t)$ is additional background signal, $PSF(x, y, z, t, u, v, w)$ is called the point spread function (PSF), and $\eta(x, y, z, t)$ models the pixel wise distortion of the image by noise [133] and δ_t is the integration time (not h_t). The additional background signal $b(u, v, w, t)$ comes from non-labelled structures either inside, or outside, the specimen that also produce light through a process known as auto-fluorescence. The point spread function, $PSF(x, y, z, t, u, v, w)$, represents a spatially varying blur kernel. This blur results from the optical properties of the microscope due to diffraction, and also the changing refractive index of the sample and chamber. The shape and size of the PSF depends on the specific optical setup and lenses used in the microscope for both illumination and detection. When using a single view, as in our example, this PSF is anisotropic being up to three times wider in the z direction than x, y. A truncated anisotropic Gaussian function (compact support) is often used to approximate the kernel. The image noise $\eta(x, y, z)$ results from both, the quantum nature of light, and from noise introduced by the camera and its sensor [130]. The quantum noise results in the signal following a Poisson distribution and is unavoidable. The other sources can follow Gaussian or Poisson, distributions and are camera specific. Also, these additional noise sources can have a spatially varying mean (non-zero) and variance. Figure 2.2 shows two examples of both a specific (x, y) slice, and a maximum z projection of a stack at a particular time step, from A a developing fish (*Danio rerio*), and B a beetle (*Tribolium castaneum*), imaged with two different SPIM microscopes. The "maximum z projection" is simply the image of the largest intensity across all z for specific x, y.

2.2.5 Processing on images

The next step, given the 3D+t images; is to return to the original task of tracking the population of M cells. This inference task may, or may not be practically feasible. In the sense that, recovery of $O(x, y, z, t)$ depends on the relative contributions and size of distortions resulting from b, PSF and η (dropping arguments). In an ideal microscope, the PSF would be a Dirac delta, and both b and η zero allowing us to measure the object function directly. Indeed, reducing these factors is the focus of most microscope development. The processing steps of taking the 3D+t images and obtaining information on the labeled objects, in our case the cell trajectories C, are done in multiple steps. First, the 3D image for each time point is often enhanced, by denoising, deconvolution, and or background subtraction. The aim of each task is to remove, or at least reduce, the effects of η, PSF, and b respectively. The next step is typically isolating objects in the image, a process known as segmentation. The segmentation can be either, binary, or multi-label. A binary segmentation is effectively trying to find a binary image $S_1\{\bar{x}, \bar{y}, \bar{z}, t_i\}$ from $I\{\bar{x}, \bar{y}, \bar{z}, t_i\}$ such that

$$S_1\{x, y, z, t_i\} = \begin{cases} 1 & O(x, y, z, t_i) > 0 \\ 0 & otherwise \end{cases} \tag{2.5}$$

for some fixed t_i. A multi-label task is higher level and involves allocating a binary segmentation to different labels. For example, in our task, a multi-label task could be assigning each pixel to a particular cell

$$S_M\{x, y, z, t_j\} = \begin{cases} k_j(i) & O_i(x, y, z, t_j) > 0 \\ 0 & otherwise \end{cases} \tag{2.6}$$

where $k_j : \{1, 2, ..M\} \to \{1, 2, ..M\}$ is a bijection assigning each cell to a unique label. The index j is added to denote that this mapping will often change between time steps. The final task in our example is known as tracking and involves making the assignments across the separate time steps t_j, such that there is then one global k for all time steps for each region. After this, the final task is to estimate the center of each object and infer the position of all cells through time, $\hat{C}$. Ideally, this process would be highly repeatable and robust, allowing the collection of numerous samples of $\hat{C}$ with high accuracy from multiple samples.

In reality, the processing steps described above can be done jointly or in a different order. Also, additional steps are often involved in the formation of $I\{\bar{x}, \bar{y}, \bar{z}, \bar{t}\}$, involving multiple views being 'fused' together to provide a smaller and isotropic PSF, also registration can be required either between different views or between time steps to correct for the sample moving. Other important processing tasks include visualization and storage of the image through compression. Figure 2.2 shows examples of two different visualization techniques, namely a single slice view and maximum z projection. Of course, in practice, the task may not be cell-tracking, but tracking, or estimating the structure of other features of the sample. However, they likely require similar steps as our example; requiring the estimation and then inference from some object function $O(x, y, z, t)$ obtained from $I\{\bar{x}, \bar{y}, \bar{z}, \bar{t}\}$.

The tasks outlined above are not unique to fluorescence microscopy or images, and have been, and continue to be, addressed in both the image processing and computer vision communities and literature. However, there are many unique features to the 3D+t image datasets that result in existing algorithms and software not being sufficient or adequate. We outline these features below, discuss current approaches, and then formulate the problem and approach that is the focus of the rest of this thesis.

2.3 The problem

The first feature of 3D+t image datasets is the number of pixels that are required to capture spatial-temporal processes in biology using pixels. In both time and space, the sampling resolutions h_x, h_y, h_z, h_t are set so that the finest details of the process in both space and time are captured, in 'some' Nyquist sampling sense (It is not always precise). However, the spatial structure of the specimens, and the temporal dynamics, of biological processes vary across many length scales across the domain.

In space, the imaging domain Ω is a rectangle; however, the sample usually is not, resulting in large areas of the domain where the Object function O is zero. Further, as shown for the two example data sets in Figure 2.2, within the sample there is a large range of spatial scales describing the Object function. Combined, this results in a significant portion of pixels contributing little information about the localization of O. The maximum values of N_x and N_y are effectively set by the camera, and typical values range from $1000 - 2000$ for both. The value for N_z is typically set based on the degree of anisotropy in the PSF. The values of N_z typically range from $50 - 2000$, with the tendency towards isotropic sampling (higher N_z) for state of the art LSFMs. Combined, for one-time step, this results in a single data set with between 10^7 and 10^9 pixels and hence raw size in memory between 100 MegaBytes (MB) and 8 GigaBytes (GB) when stored is unsigned 16bit integers (as is standard) [1]. Hence, a single 3D image for one time-step can have a size that is over a thousand times larger than typical 2D images in traditional image processing.

The large size of data is further exacerbated with the addition of time. As with space, the temporal dynamics of an Object function for a biological

---

[1] However processing is often done using single floating point precision (32bit)

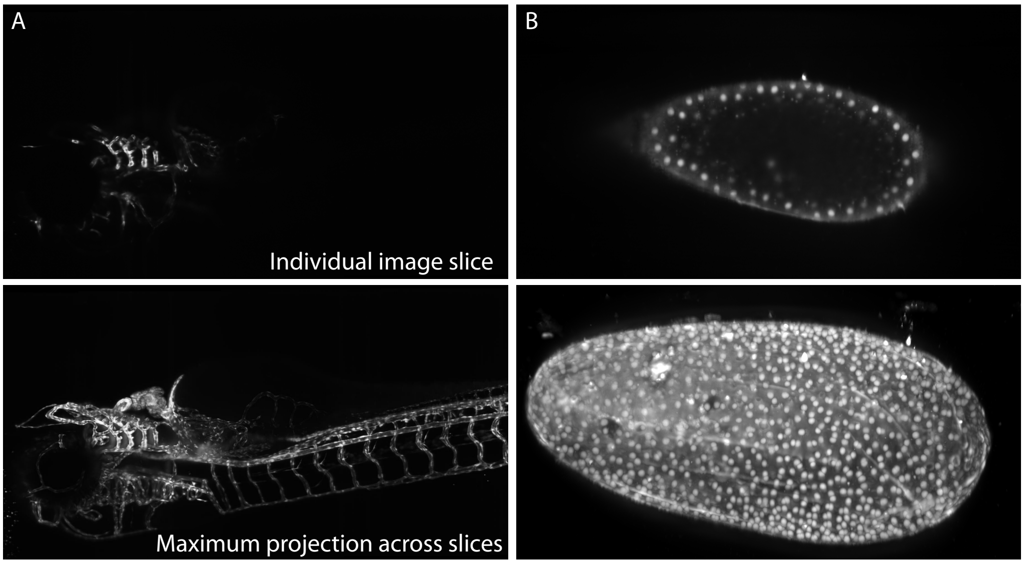


Figure 2.2: The top panels show a single image slice $I(x, y, z0)$ of a stack for two different specimens and labels from two different LSFMs. The bottom panels show a maximum projection of the full 3D image stack. The maximum projection $I_{max}(x, y)$ is the image formed by taking the maximum value across z for each location x, y. **A** shows images of the GFP-labeled vasculature of a developing Zebrafish (*Danio rerio*) imaged using a custom built single view SPIM,(Courtesy of Stephan Daetwyler, Huisken Lab - MPI-CBG, Dataset Number: 2 Table A.1). **B** shows images of GFP-labelled (LifeAct) nuclei of a developing flour beetle embryo (*Tribolium castaneum*) imaged using a commercial Ziess Lightsheet Z.1 microscope (Images courtesy of Akanshka Jain, Tomancak Lab, MPI-CBG, Dataset Number: 16 Table A.1).

process usually exhibits a range of time scales. For cells, for example, both their movement and division (cell replication) are characterized by random processes with small sudden changes, amongst larger periods of little activity. The total time domain under study can vary greatly, with the required total time of imaging ranging from minutes to days. In practice, N_t can range from one to thousands, therefore pushing the largest potential datasets up to 10^{12} pixels and TeraBytes (TB) of data. These data volumes can be further increased by the use of more than one channel for simultaneous imaging of different labelings, or, by the use of multiple views per time-point to make the PSF smaller and more isotropic [60].

The potential for data generation over a day by a typical LSFM is shown in Figure 2.3. The figure compares current LSFM techniques (SPIM with an sCMOS camera) to old microscopy techniques and cameras (reproduced from Reynaud et al. [96]). This large data volume has been called the *Big Data Challenge* by Reynaud et al. [96] and impacts every part of the processing steps outlined above, causing issues even for the storage and transferring of data. Previously the bottleneck to studying spatiotemporal processes was the microscope and labeling technology. Now, with the development of LSFM, it is the data volume and processing challenges that create a bottleneck and limit the scale and type of experiments [108].

Let us return to our example of tracking a population of M cells by estimating C to illustrate the *Big Data Challenge*. Let us assume that in our example, we imaged for a day capturing images once a minute of $M = 10000$ cells (again ignoring division and death to keep the calculation simple). Such a population is large, and we assume it would require at least the sampling values previously discussed to capture the full process ($N_x = N_y = 2000, N_z = 1000, N_t = 1440$). Therefore, the 3D+t data set could be as large as 8 TB, while our final result $\hat{C}$ would only require 400 MB when stored as in floating point precision, a factor of 2000.

The second feature of these image datasets adds to the complexity of processing tasks [63, 138]. The Object function (and image), not only include different spatial (where) and temporal (when) scales but also, differing scales in 'how' the information is encoded. In detail, this means that each object O_i can have a different range of function values in both the Object function and Image. We call this local range the *local intensity scale*. The local intensity scale has many contributions. The first is the Object function itself. The density of fluorescent labels (what the Object function is a proxy for) can vary across objects, in our case, cell nuclei. The change in the range of labeling can result from random fluctuations, systematic difference between the cells level of fluorescence molecule (e.g. different expression levels), and reduction of labeling by a process known as photo-bleaching. The second factors are a result of the image formation process, where all three of the functions PSF, b and η can contribute. The width and height of the PSF impacts how the density at each location in the object function translates into image intensity. Therefore, if the PSF varies across the image, this will also change the imaged brightness across these regions. The background signal b contributes additional signal to the local range in the image I that is independent of O. Lastly, the fluctuations in the camera can result in random and deterministic offsets in how the light that hits the camera is translated into a signal and intensity value.

The combination of the dynamic spatial, temporal, and local intensity scales combined with the high pixel number in the 3D+t images often leads to the direct application of software, algorithms, and methods in image processing and computer vision being unable to solve the discussed processing tasks. This deficiency comes both in practice, due to high computational cost, and in principle, by the lack of acounting for varying local intensity scales in the underlying models. Although, other image modalities, often have varying spatial and local intensity scales the magnitude of these variations across a simple

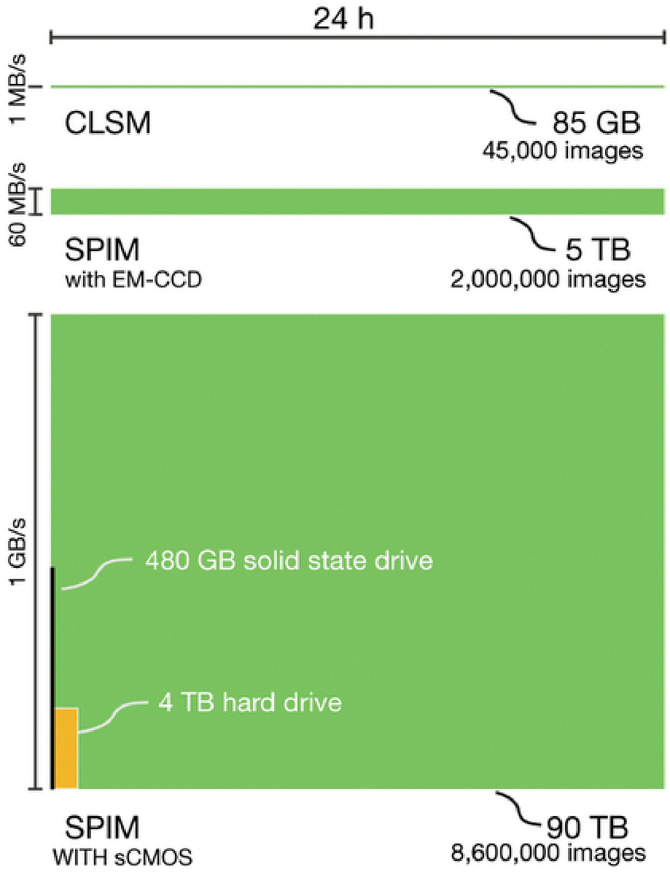


Figure 2.3: Schematic showing the data produced in 24 hours at maximum data-rates of a current LSFM (SPIM with sCMOS Camera) producing up to 1 GB/s of image data, compared to a SPIM with older camera technology (EMCCD Camera) producing 60 MB/s, and older optical sectioning microscopy technique, confocal laser scanning microscope, 1 MB/s. For perspective the schematic gives the relative sizes of this data volume compared to typical hard-ware sizes (circa, 2013). (Reproduced from Reynaud et al. [96] which was changed color and adapted from Schmid et al. [111])

data set is often smaller. For example, regarding the local intensity scale, this is directly reflected in need for, and use of, unsigned 16-bit images, as opposed to 8-bit images that are often sufficient in other image modalities (greyscale). The smaller magnitude of changes can mean that constant approximations, or simple models, are adequate for solving most processing tasks. In cases where they are not adequate, the relatively smaller size of the data sets permits more sophisticated methods to be successfully used even if they result in a significant relative increase in computational and memory cost. Whereas, for LSFM data, such a relative increase in computational and memory cost often exceeds capabilities of current hardware and software [5]. These problems are exacerbated by the large spatial and temporal scales combined with the high pixel count. Effectively, any algorithm has to 'find' the locations of the data set that are informative, requiring processing across every pixel, before more sophisticated steps, can be run. Although such steps may be 'cheap' for small datasets, they can become prohibitive at the data sizes discussed here. Further, they have to be repeated across different tasks, or even steps, within an algorithm.

Therefore, these features of 3D+t LSFM data result in the need for the development of custom algorithms, and methods, to deal with the processing challenges. From the discussion above, the requirement of such algorithms would be to have for low computational and memory cost per pixel, while still allowing for the use of models that can account for the variations in the temporal, spatial, and intensity scales. Indeed, there has been significant research and development, into the design of methods and software to meet these requirements. In the next section, we provide an overview of the current approaches and methods for processing on LSFM data.

2.4 Current approaches to processing for LSFM

At a high level, we can split the research into LSFM processing into two groups, those focusing on specific processing steps, such as deconvolution, and those providing pipelines from image to a specific result, e.g. cell tracking, therefore combining a range of processing steps. We will briefly overview the literature in the two groups. Following we generalize and discuss the different approaches and how these relate to the challenges raised above.

The research in the first group has been specifically designed to deal with general features of LSFM data relevant to a processing task. Implementations of the methods are often provided as a Plugin for FIJI [109], a popular image-processing tool used by biologists. The specific processing tasks include image registration [90, 88], image fusion [121, 100, 89, 91, 88, 100], image deconvolution [91, 132, 123, 138, 27, 110, 48, 122], image segmentation

[5, 71, 63, 10, 9, 53, 117, 75], visualization [64, 84, 98, 11, 82], cell tracking [140, 10, 117], and storage and compression [56, 111, 11].

The second group of processing pipelines can be further split. First, there are those pipelines that have been designed for a custom microscope, specimen and task [132, 70, 125, 119, 127]. Second, there are those pipelines designed for a wider use and scope [113, 11, 10, 117, 11, 49, 134]. The goal of the general pipelines is to combine the individual processing methods mentioned above into workflows and software that is more accessible to biologists [113].

2.4.1 Concepts

Instead of focusing on individual processing tasks, or reviewing each work in detail, we have attempted to identify the key ideas and elements across the research. With specific attention on those methods that provide solutions to address the large size of the images and the spatial, temporal, and local intensity scales.

Paralellism

A common method for approaching the computational cost of such large image datasets is by increasing the throughput of the method by designing algorithms that can be parallelized and run on graphics processing units (GPUs) [110, 27, 138, 9, 10, 91, 98, 82]. GPUs are designed to perform thousands of simple computational operations simultaneously. Utilizing GPUs can provide significant (For example 25 times in [110]) speed up regarding total execution time when compared to similar code executed in serial on a CPU. Utilization of GPUs has been very successful in reducing computational time. With their use particularly well suited for processing tasks such as deconvolution and image fusion. However, GPUs are less suited to higher level tasks, such as segmentation and cell tracking, where branching or more irregular, computational patterns may be required. Further, GPUs require the data that is being computed on to be transferred to and stored in its local memory. The local memory requirement can limit the size of datasets that can be processed, as the size of this local memory is often an order of magnitude less, than that available to a CPU through RAM. The memory restriction can become an issue for higher level tasks that may require the simultaneous storage of multiple variables per pixel [5].

In contrast, for acceleration of a general segmentation algorithm, Afshar and Sbalzarini [5] utilized distributed parallelism. Distributed parallelism, simultaneously addresses both computational and memory costs, by computing the solution jointly across multiple computer systems. Using distributed parallelism allows the amount of computational power and memory to be scaled to the size of the problem. Therefore, more complicated algorithms can be run on large datasets. However, a drawback is that development and implementation of algorithms that solve problems on distributed systems can be difficult.

Real-time processing

A second common concept, often linked to the use of parallelism, is the concept of real-time or online processing [108, 137, 111, 110, 27, 5, 98, 117, 99]. For LSFM, real-time processing is the processing of image data sets during the acquisition process of the microscope. Real-time, in this context, means being able to perform a processing task on the data set for a given time point before the imaging of the next time point. The classification of a method being real-time therefore depends on the particular experiment.

Real-time processing appears to provide two major benefits. The first benefit is that information in the acquired images can be used to monitor or adjust the experiment while it is running. The monitoring and adjustment can be done manually by a user viewing the data set, using real-time visualization such as ClearVolume [98], or automatically, with algorithms adjusting and monitoring the acquisition [99], resulting in what has been termed a *smart* microscope [108]. The second benefit involves reducing the data set size as it is acquired. By doing this, the full data-set never has to be stored or processed on all at once. Real-time processing is particularly useful for the fusion of multiple views, as the dataset can be immediately reduced to one view [110]. A second approach is that the full image data can be no longer stored. Instead, only storing a result that has a smaller memory footprint such as the segmentation [5] or alternative data representation [111].

Modelling the image formation process

Algorithms have also been developed to remove the effects of b, PSF, or η that occur during image formation (see 2.3). By removing, or reducing, their effects they make the input image I 'closer' to the desired Object function O and hence simplify further processing. In particular, methods have been introduced that reduce the impact of PSF, by performing deconvolution with an estimated spatially varying PSF [48, 122]. Weigert et al. [138], goes further, providing both deconvolution, and 'super-resolution' using methods from deep learning. The approach allows the reconstruction of isotropically sampled datasets, from anisotropically sampled and blurred input data, in a data-specific way. Alternatively, Jensen et al. [63], introduce new image models, to allow for different local intensity scales for segmentation.

Data representations

The final approach we discuss is using alternative data representations. That is using alternative representations of the image for processing other than or in addition to a pixel image.

For visualization the BigDataViewer [84], utilizes multiple down-sampled versions of the image, with smart-caching schemes, to allow visualization and basic processing of large datasets. For segmentation and tracking, [10, 117] use the concept of super-voxels, a 3D extension of super-pixels [2] using optical flow [9]. The use of super-voxels helps address the different spatial and local intensity scales, while also reducing computational complexity. The method groups adjacent pixels together into groups, called super-voxels, where the number of super-voxels is much less than the number of original pixels N. Segmentation and cell tracking can then be done on this smaller dataset, giving improvements in both computational and memory costs. Further, the methods use local descriptors to group pixels, allowing it to account for changes in local intensity scale. However, rather than replacing a pixel image, super-pixels effectively *augment* it. With the original image still being required for further processing tasks such as visualization.

Schmid et al. [111] introduces a different approach that translates the original image into an alternative data representation in real-time. Specifically designed for early developing Zebrafish embryos it makes use of their spherical shape. The 3D images are acquired by the microscope, and in real-time, the data is processed and projected onto a sphere using a radial maximum projection operation. The full image data is never stored. Therefore, all subsequent processing steps are performed on the projected image providing significant memory and computational benefits. Unfortunately, this approach is limited to only a subset of species and developmental processes. Further, it is unclear whether such a maximum radial projection retains all relevant information on the Object function from the original images. The idea of using 2D projections has generalized and further extended to work on more general geometries by Heemskerk and Streichan [56].

2.4.2 Summary

Although the approaches above have been successful in providing methods and algorithms that have allowed the utilization of LSFM for studying spatial and temporal processes, there is still room for much development. Unfortunately, many of these approaches are specific or do not scale sufficiently to make full use of, current, and likely future, advancements in LSFM. Indeed, although methods are sufficient to provide analysis of a single large time course of a single embryo, the ability to then scale consistently to a larger number of samples is still hindered by the processing steps. A lacking feature of current approaches is that the algorithms deal with the large size, and varying scales individually and by time point. This duplicates the process of identifying and accounting for the scales and high data volume. The exception to this trend is the projection methods used by [111].

A successful approach we believe would combine a majority of the features above. Providing a more general solution in the spirit of Schmid et al. [111] by providing an alternative data representation that accounts for the different scales in the dataset while simultaneously reducing its computational and memory cost, while still fully capturing the information of the Object function O. By handling the features of the dataset once in a general way, the representation could then be utilized across the whole pipeline without the need for original pixel data. The approach would ideally also scale to larger data sets and be able to account for temporal scales.

In the next section, we reflect on biological visual systems and argue that they provide examples of an approach with the features discussed above. We then use two key features from visual systems as motivation for our work here.

2.5 Motivation from the visual system

As humans, our visual system continuously acquires high-resolution information on spatial-temporal processes occurring in the world around us. Similar to fluorescence microscopy, the visual information is gained through the detection and delineation of objects through local variations in photon counts. However, instead of objects having to be specifically labeled, changes in the photon levels reflect the specific optical properties of the object, its location, and local luminescence. Hence, the problem can be framed similar to that of processing on LSFM data, with an Object function O, which we wish to infer information about from an input 'image' I that we perceive.

The comparison with the visual system can be taken further as it shares many of the same features found in LSFM data and its processing. Typical visual scenes feature both varying spatial and temporal scales in the Object function. In space, a visual scene is composed of objects, which feature many spatial scales, with information on their shape and size largely determined by their boundary. They also exhibit a range of temporal scales, encountering both stationary, slow and fast objects at any one time. Visual scenes also show large variations in their local intensity scale, with luminescence levels varying over up to nine orders of magnitude in a typical day [116].

To compare the 'size' of the problem compared to LSFM, we consider the following thought experiment. How much data would the human visual system have to acquire and process if it used a homogenous sampling and processing strategy as in an LSFM? We provide a 'back of the envelope calculation', to estimate the order of magnitude. To estimate a hypothetical rate, we assume the sampling resolution in space and time to be the maximum resolution, 120 degrees field of view, three color channels stored, across at 64-bit precision. This would result in a data rate of around 2000 GB/s, or 180,000 TB a day (See A.1 for calculations). This rough calculation indicates that the effective data size the visual system deals with is at least comparable to LSFM. However, it is obvious the visual system does not function in this way, in fact, estimates on the data-rate from the retina to the visual cortex is as low as 1 MB/s [68].

Two core features of the human visual system are adaptive sampling and local gain control. This adaptive sampling works by selectively focusing the attention of the eye on areas with potentially high information content [57]. This adaptation occurs at a higher level, such as detecting faces, but also at lower levels, adapting to information-rich areas of the scene such as edges [95]. The selective focus enables efficient inference of information about the scene at a high effective resolution by focusing the processing capacity of the retina and visual cortex. The local gain control, allows the eye to effectively adapt to a scene while accounting for luminance ranges of over nine orders of magnitude, even though the firing rate of an optic nerve varies over less than two [116]. Hence, the local gain control effectively normalizes the signal efficiently accounting for variations in the local intensity scale. Further, it has been argued; these features have evolved to provide an efficient data representation, and the adaptivity optimizes the information transmission efficiency [23]. It is from the two features of adaptive sampling and local gain control, that we take inspiration from, for the development of the adaptive representation in this thesis.

2.6 Summary and main points

In summary, the eye efficiently accounts for the large spatial and temporal scales through adaptive sampling, while handling varying local intensity scales through a local gain control. The adaptivity occurs at the lowest level (first step) in the visual system, allowing all subsequent, and higher level, processing tasks to benefit from it.

Given the challenges of processing on LSFM data, shares many features with the visual system. It would seem that a similar approach could be useful. Indeed, the idea of adaptivity is a feature of both the successful super-voxels and projection approaches. We follow this approach designing an alternative representation that shares these properties. In the next chapter, we formally outline the desired properties for an adaptive representation and review the literature. The table below briefly summarizes the chapter.

Summary of the chapter

- Introduced LSFM and how it can be used to study spatiotemporal processes in biology.
- Discussed the challenging features of LSFM data and outlined the current bottleneck in processing raw image data
- Identified variations in spatial, temporal, and local intensity scales as key issues contributing to this bottleneck
- Reviewed and reflected on current processing techniques and concluded the need for an alternative data representation
- Compared the problem and solution from the human-visual system for inspiration for an efficient data representation
- Concluded adaptive sampling and local gain control as key concepts for an alternative data representation

3 Representation criteria and previous work

Contents

3.1 Introduction **25**
3.2 Representation Criteria (RC) **26**
3.2.1 Representation Criteria 1 (RC1) 27
3.2.2 Representation Criteria 2 (RC2) 28
3.2.3 Representation Criteria 3 (RC3) 28
3.2.4 Representation Criteria 4 (RC4) 29
3.2.5 Representation Criteria 5 (RC5) 29
3.3 Review of existing multi-resolution and adaptive representations **30**
3.3.1 Processing augmentation techniques 31
3.3.2 Adaptive sparse transform domain techniques 36
3.3.3 Adaptive sparse collocation techniques 40
3.4 Discussion **47**
3.4.1 Lack of local gain control (RC1) 47
3.4.2 Use across a wide range of tasks (RC4) 47
3.5 Summary and main points **48**

3.1 Introduction

In this chapter, motivated from the previous, we further develop the requirements of an alternative representation for LSFM images that could alleviate processing bottlenecks. First, we outline five *representation criteria* (RC). We form the RC from both reflections on the demands of processing, successful existing methods and algorithms, and ideas of efficient representations from the visual system. Following this, we review the literature on multi-resolution and adaptive-representations, focusing on areas are of most relevance. In concluding, we reflect on existing work and identify those ideas from existing literature that still require development to fulfill the RC.

3.2 Representation Criteria (RC)

Here, we present five criteria that I believe an alternative image representation should fulfill to be able to be used as a general solution for processing LSFM images for studying spatiotemporal processes in biology. The alternative representation would be formed from the pixel image data after, or during, acquisition. Then all preceding processing steps and storage would be done using this alternative representation without the need for the raw pixel image data. Motivated by the visual system, the representation would adaptively sample the image domain while taking into account the local intensity scale. This sampling would adapt to the range of spatial and temporal scales in the Object function O while taking into account variations in the local intensity scale. All following processing steps could then directly benefit from the reduction in computational complexity, and identification of scales.

We will first state the five criteria, then describe each and discuss it in detail. The *representation criteria (RC)* are as follows:

- **RC1**: The size of the representation $\#R$ must be proportional to the information content of an image, accounting for varying spatial scales, and not scale with the number of pixels.
- **RC2**: The representation must guarantee a user-controllable representation accuracy E for noise-free images, relative to a local intensity scale σ, and not reduce the signal-to-noise ratio of noisy images.
- **RC3**: It must be possible to rapidly convert a given pixel image to the representation with a computational cost at most proportional to the number of pixels.
- **RC4**: The representation must reduce the computational cost of image-processing tasks without resorting to the original pixel representation.
- **RC5**: The representation must also be able to similarly account for varying temporal scales.

Of course, such criteria are subjective and broad, and showing a representation satisfies them in any precise way does not seem possible. However, we find they do provide a useful tool, both for guiding the development of the work here and evaluation of the results.

Although, these criteria, may not be sufficient, and it is likely that there are others that such an alternative representation should fulfill. Also, though we are attempting to develop the method to be of general use, there are likely problems where the use of any adaptive representation, regardless of the RC, would not be appropriate. For example when the exact pixel noise distribution is meaningful.

In this section and the immediate chapters, we do not immediately address adaptation through time, instead focusing on a single time-step and RC1-4. Later, in Chapter 10 we address RC5 and show how in addition to the spatial and local intensity scales, how temporal scales can also be addressed using the same framework.

For now, let us consider a representation R of image I, and we utilize concepts, notation, and examples from Chapter 2. We again consider that we have labelled M biological structures, which we model by an Object function $O(\mathbf{y}) = \sum_{i=1}^{M} O_i(\mathbf{y})$. Where we consider a single time point and $\mathbf{y} = (x, y, z)$. Then, the information in the Object function regarding object i, is contained in the support of the object function of i, that is, $supp\{O_i(\mathbf{y})\} = \{\mathbf{y} \in \Omega | O_i(\mathbf{y}) \neq 0\}$. Again, instead of observing O directly we have an input image $I\{\bar{\mathbf{y}}\}$, where $\bar{\mathbf{y}} = (\bar{x}, \bar{y}, \bar{z})$, with a total number of sample points $N = N_x N_y N_z$, where the image is formed by a process as described by Equation 2.3.

We discuss more details and motivation for each of the RC below.

3.2.1 Representation Criteria 1 (RC1)

Following the visual system, our representation should adaptively sample the domain Ω. Adapting so that all processing tasks can focus their computational and memory costs proportionally to information rich areas.

We define the information content in I as the data that can be used to infer the support of O. Therefore, $\#R \leq N$, the number of elements in, or size of the representation, should reflect the amount of information content in the image, and not simply the smallest spatial scale multiplied by the domain size as in pixel images. Since in most cases, algorithms computational and memory complexity are of the form $\mathcal{O}(f(\#R))$, by aligning the size of $\#R$ with the information content, then the computational and memory costs to also scale with information content. However, as discussed, the information in O is encoded in local changes in I, where the scale of these changes, we call the local intensity scale $\sigma(\mathbf{y})$. As in the visual system, we want the representation R to account for the changing scale σ, as in the eye. Therefore, ideally, the sampling would scale as

$$\#R \sim \int_{\Omega} g(I(\mathbf{y}), \sigma(\mathbf{y})) d\mathbf{y}. \tag{3.1}$$

where $g : \mathbb{R} \to \mathbb{R}$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is an unknown function that describes the change in local intensity scale. Included, should be the ability to efficiently losslessly compress the APR for file storage with a cost that reflects $\#R$.

3.2.2 Representation Criteria 2 (RC2)

We assume this adaptive sampling will be *lossy.* That is $||\hat{I_R}\{\bar{\mathbf{y}}\}-I\{\bar{\mathbf{y}}\}||_{l_1} \geq 0$, where $\hat{I}$ is the sampled image reconstructed in some way from R. However, we wish to be able to control the size of this reconstruction error, and hence control the loss of information about O.

a priori, we do not know the degree that points in Ω of I contain information about O. Therefore, we require that we control the reconstruction error at all points in the domain similarly. Hence, the use of the infinity norm, would seem justified. However, as with the adaptation, we know that the absolute value of I, and also the reconstruction errors ϵ at any point, to not have 'meaning' without taking into account the local intensity scale σ. Therefore, we propose that our representation should be able to adapt to the information content while controlling error as

$$|\frac{\hat{I}\{\bar{\mathbf{y}}\} - I\{\bar{\mathbf{y}}\}}{\sigma(\mathbf{y})}|_\infty \leq E \tag{3.2}$$

where $|\bar{x}|_\infty = \max_{x \in \bar{x}} x$, $\hat{I}\{\bar{\mathbf{y}}\}$ is the reconstruction from the representation and E is a user-controlled parameter. That is, the reconstruction error is controlled pointwise across the domain. For noise-free circumstances, we propose that E can be set arbitrarily small.

However, in the presence of noise, as occurs during image formation for LSFM, minimizing the difference between the observed noisy image and the reconstructed image is not desired. Instead, it is the reconstruction error relative to the noise-free image I^* we would wish to control. Therefore, the representation must be able to account for noise corruption, with an adaptation that is controlled by E, while endeavoring not to lose information on O. That is instead the representation should adapt such that, as E decreases the reconstruction error of the representation relative to the noise-free image I^*, should be equal to, or less, than noise level of the original noisy image.

Ideally, both this adaptation and the noisy reconstruction should be done optimally (in some sense).

3.2.3 Representation Criteria 3 (RC3)

To be used in general we require to be able to generate R from $I\{\hat{\mathbf{y}}\}$ in a fast an efficient way. Further, this process must, scale, such that if datasets continue to grow, the cost of R will scale at the same rate. Hence, the computational cost of generating R should be at most linear in N, i.e. $\mathcal{O}(N)$.

Also, this process should be amenable to utilizing current and future parallel hardware. Ideally being able to utilize, GPU, shared memory and distributed parallelism.

Lastly, the process should be fast, regarding execution time, where 'fast' depends on the application. Specifically, it should be fast enough, with the use of appropriate hardware, that it can be used in a range of real-time applications. That is, able to compute R, during acquisition between acquiring each time step.

3.2.4 Representation Criteria 4 (RC4)

We propose that R is to be used, instead of the pixel image data, for all processing tasks after its formation. For example, visualization, segmentation, tracking, and storage. Replacing pixels with an adaptive representation of the data for all processing tasks, as motivated by the human visual system, and the methods of Schmid et al. [111]. In the best case, all existing algorithms and software could be directly used with R, while fully benefitting from its adaptive properties from **RC**1-3. However, this is unrealistic, given most implementations of algorithms, implicitly depend on the structure of a pixel image. Although, a more reasonable goal, is that the underlying methods, and algorithms, can be adapted from their pixel formulation to one for R, with minimal effort.

Different algorithms and methods, interpret a pixel image in varied ways. For example, collocation points of a continuous function, voxel elements, a pixel graph, or nodes in a tree structure. Therefore, we propose that the representation R should be 'close' enough to a pixel image, that similar interpretations of R are natural and possible. By sharing the interpretations, existing algorithms are likely to be able to be more easily adapted.

Furthermore, the computational, and memory performance of these adapted algorithms should be able to benefit from the adaptation, both regarding size, since $\#R \leq N$, and through the information about the spatial scale and local intensity intrinsic to R through adaptation. For this to be possible requires that any increase in memory or computational cost of data structures, and algorithm access patterns, for adapted algorithms, do not exceed the benefits gained from adaptation.

3.2.5 Representation Criteria 5 (RC5)

Lastly, in addition to adapting to spatial scales, and the local intensity scale, the representation should extend to time. That is, it should also in a way consistent with **RC**1-4, also be able to adapt sampling in time to account for the temporal scales in Object function that exist. Further, we suppose that the time adaptation should be causal. Both, to preserve temporal ordering of events and allow real-time processing.

Again, note that we first treat space, then address time in Chapter 10.

3.3 Review of existing multi-resolution and adaptive representations

In this section, we review the relevant existing literature on alternative data representations used for processing. First, we introduce our definitions of multi-resolution and adaptive methods used here.

Representation definitions We define a multi-resolution representation as:

Multi-resolution representation. A method that represents a spatial data set with a representation with more than one (multi) length-scale in each spatial direction.

The size of the multi-resolution representation R, i.e. the number of free parameters $\#R$, or coefficients $\{c_i\}_{i=1}^{\#R}$, that describe it, need not be less than the number of samples in the original data N. Such representations can be *lossless* or *lossy*. A lossless representation allows perfect reconstruction (per sample) of the original dataset from R. Whereas, a loss-less representation does not.

A simple example using a 2D image would be the combination of the original image and a downsampled image by a factor two. The size of the representation $\#R$ for this example would be $N + N/4$. Whereas, a pixel image by itself has only one spatial scale in each direction determined by the sampling distance between pixels.

Building on this defintion, we define an adaptive representation as,

Adaptive representation. A multi-resolution representation that also involves data-dependent selection of non-zero free parameters relating to the spatial scales.

The size of the representation, need not be less than N, but the determination of the number of non-zero parameters to describe the representation depends on the dataset in some way, beyond scaling with N (or simply the result of the data set being zero). In this definition, there is slight ambiguity for lossless methods that could have zero coefficients for a given signal, without any active *selection*. For example coefficients for a lossless transform. In this case, if the non-zero coefficients are treated the same as non-zero coefficients, we consider it only a multi-resolution representation, if however, special treatment is given to these coefficients ('in some way') we would define the representation as adaptive. Therefore, this ties our definition of representations also to their use.

Review approach Methods and techniques using both multi-resolution and adaptive representations, have been developed for, and are used with success, across a range of fields. These areas include image processing, computational differential equations, statistics and computer graphics (find a range of specific references below). Across these different applications, there are similar methods, ideas, and concepts used. However, there are also differences, diversity, and specialization, across the fields. It is likely this diversity reflects that the methods are customized to address the specific demands resulting from the features of the data and the nature of memory and computational costs in the processing involved. As discussed in Chapter 2, LSFM data has unique features compared to classical natural image data, and processing tasks. Hence, it would seem necessary also to review literature originally designed for other applications, as they may share common features with LSFM data or processing. Rather than focus on each field and their approaches separately, here we have attempted to identify and group similar methods and ideas.

At the highest level, we split techniques into three groups. The groups are *Augmentation Techniques*, *Sparse Transform Techniques*, and *Sparse Collocation Techniques.* We define and discuss each group separately in the section below. In each subsection, we highlight the main ideas, and methods, we have identified, and then finish in each case by reflecting on the representation criteria **RC**1-4. These definitions are tied to how the specific method is used, and a particular method can be in more than one group. For example, Wavelet methods feature in all three. These groupings were heuristically made to allow comparison with the representation criteria easier.

In all cases, we consider a function $f(x)$, sampled at locations $\bar{x}$, and stored as N samples $f\{\bar{x}\}$. In most cases, we will give the notation for a $1D$ notation for simplicity, but it is to be assumed the ideas can be applied in $3D$ unless otherwise stated.

3.3.1 Processing augmentation techniques

Here, we review multi-resolution representations are not adaptive (not data dependent) and adaptive techniques that are used in addition to the original data. Often, the multi-resolution or adaptive representations are used to

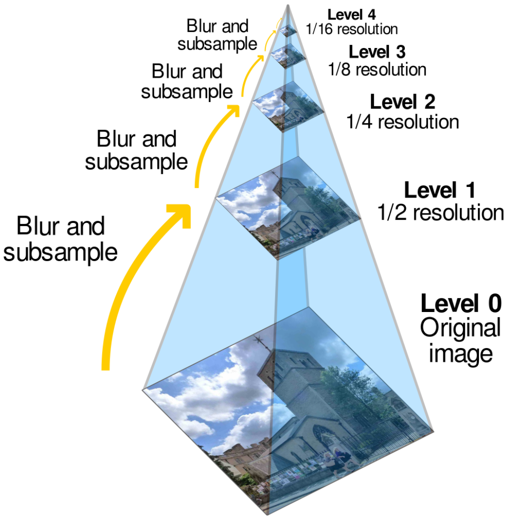


Figure 3.1: Schematic of a five level image-pyramid, and example of a non-adaptive multi-resolution representation R of an image. Levels greater than one correspond to blurred down-sampled versions of the original image. (Source: `https://commons.wikimedia.org/wiki/File:Image_pyramid.svg`, Used under Creative Commons Attribution-Share Alike 3.0 Unported, Created by user: Cmglee)

augment a processing step. This is done by simplifying, or regularizing, a specific processing step. Peyré [83] provides an excellent review of adaptive and multi-resolution representations for image-processing. In the review, the representations are viewed as regularizers for ill-posed inversion problems.

Pyramid techniques

Some of the most natural multi-resolution representations are image pyramid techniques [4]. These techniques involve augmenting the original image with down-sampled images by a factor of two as shown in Figure 3.1. These down-sampled images may either be blurred versions of the original image or, the result of another filtering operation such as Laplacian image pyramids [26]. These multi-resolution representations can aid processing to help identify features on different scales and have also been motivated by visual systems [4]. Related to these techniques, are scale-space techniques [139]. Scale-space techniques, also form multiple blurred versions of an image. A convolution with a Gaussian with smoothing scale s is often used for blurring. A set of images is then created for a range of values of s. In the case of a $1D$ signal, the signal is augmented to 2D with the additional dimension being scale s. This 2D augmented signal can then be combined with tree structures decompositions and used to identify different resolution scales in the data.

Change of basis

Another important approach to identifying different length scales in a function is the use of a change of basis. If the new basis has different spatial scales, the size of different coefficients can be used to identify spatial scales in the data.

Discrete Fourier Transform The 'classic' multi-resolution representation for signals is the discrete Fourier transform (DFT). The discrete Fourier transform represents a signal in the following way,

$$f\{x_n\} = \frac{1}{N}\sum_{k=0}^{N-1} c_k e^{\frac{i2\pi kn}{N}} \tag{3.3}$$

where $x_n \in \hat{x}$, and the c_k are calculated as,

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1} f_n e^{\frac{i2\pi kn}{N}} \tag{3.4}$$

so $\#R = N$ and $i = \sqrt{-1}$. The DFT effectively transforms the signal into a frequency domain representation. Where $\{e^{\frac{i2\pi kn}{N}}\}_{n=0}^{N-1}$ forms an orthogonal basis of $\mathbb{C}^N$. Where the $|c_k|$, corresponds to the contribution of a certain frequency to the original signal. It is multi-resolution in the sense that the basis functions $e^{\frac{i2\pi kn}{N}}$ have different wavelengths that describe them. In addition to giving frequency information on a signal, the Fourier transform also can simplify and speed up certain processing tasks such as digital filtering or convolutions. Rather than calculating the DFT using the above definition, the Fast Fourier Transform is often used allowing the coefficients c_k to be computed with a computational complexity of $\mathcal{O}(N \log N)$ instead of $\mathcal{O}(N^2)$ (the result of the application of the definition above). Though the DFT allows identification of frequency, and hence spatial scales in the solution, it does not allow us to locate where the spatial scales occur in space.

Wavelets Wavelets, in contrast, allow both the identification of length scales and their location [74, 34]. Wavelets are based on the decomposition of a signal into basis functions of different sized support placed at regular intervals over the domain. The basis functions are formed by combinations of two special functions $\phi_i(x) = \phi(x - i)$ and $\psi_{i,j}(x) = 2^{\frac{j}{2}}\psi(2^{\frac{j}{2}}x - i)$ known as the scale and wavelets functions respectively, where i determines the location in space and l the length scale. The combination of these, $\{\phi_i\}_{i\in\mathbb{Z}} \bigcup \{\psi_{i,j}\}_{i,j\in\mathbb{Z}}$, forms an orthonormal basis of $L^2(\mathbb{R})$ [30].

When sampling a discrete sampled function f, assuming that $N = 2^{l_{max}}$, then wavelets allow the following decomposition of data in what is known as a Multi-Resolution Analysis (MRA) [74] as

$$f(x) = \sum_{j=j_{min}}^{j_{max}} \sum_{i=1}^{2^j} d_{i,j}\psi_{i,j}(x) + \sum_{i=1}^{2^{j_{min}}} a_i\phi_i(x) \tag{3.5}$$

where $d_{i,j}$ are called the detail coefficients and a_i are known as the approximation coefficients. Figure 3.2, shows an example of such a decomposition. The decomposition can be interpreted in the following way; the approximation coefficients generate a low-resolution representation of the function, determined by j_{min}, then the detail coefficients, level by level, add details progressively to the function. The detail coefficients with lower j, respond to longer length scales, and higher j, smaller length scales. Further, for smooth regions of a function, as j increases, the size of $|d_{i,j}|$ decreases. Whereas, if there is a local discontinuity or non-smooth region, the $|d_{i,j}|$ will not decay, or not as rapidly. Hence, different scales and discontinuities can be readily detected in a signal by analyzing the behavior of $|d_{i,j}|$ across the domain [73].

In contrast, to the pyramid multi-resolution techniques above, the size of the wavelet transform $\#R = N$, and the coefficients can be computed with linear computational complexity in N using the Discrete Wavelet Transform (DWT) [72] ($\mathcal{O}(N)$). A variety of processing steps can then be formulated in the Wavelet domain, to regularize, or improve various processing tasks [83]. However, wavelets have many additional features when used as an adaptive representation, through thresholding of coefficients, which we discuss in the next section.

Superpixels

Superpixels group together similar, by some measure, pixels in an image into regions larger than a single pixel [2]. In this way, superpixels can be viewed as a form of over-segmentation. Figure 3.3 shows examples of superpixels generated with different parameters of two images using the SLIC method in Achanta et al. [2]. In the figure, the black lines decern the boundaries between superpixels. Superpixels are especially well suited for segmentation tasks, where they reduce the number of computational elements, and hence memory costs, and help identify spatial, and local intensity scales. These features have been utilized by for segmentation and tracking using LSFM data using the method for 3D data, or voxels, introduced by Amat et al. [9]. However, the method is not designed as a general representation. Instead, the full dataset is retained and used for all other processing and storage tasks, and may indeed still be

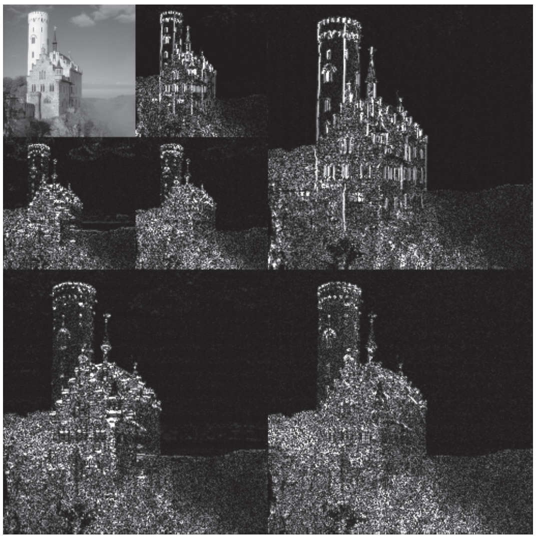


Figure 3.2: An example of a wavelet decomposition using the wavelet transform used by JPEG2000 [31]. The small image in the top left corner, represents the approximation coefficients a_i and the other six images represent the detail coefficients $d_{i,l}$ at two different resolutions. (Source: `https://commons.wikimedia.org/wiki/File:Jpeg2000_2-level_wavelet_transform-lichtenstein.png`, Used under Creative Commons Attribution-Share Alike 3.0 Unported, Created by user: Alejo2083)

utilized while processing with the superpixels. So effectively $\#R \geq N$, since the original image, is still required in addition to the superpixels.

Reflection on representation criteria

All of the above methods have been highly successful in using multi-resolution concepts to identify different spatial, and local intensity scales, to improve processing tasks. However, they do not address our representation criteria. In particular, since in all cases $\#R \geq N$, then **RC**1 is not satisfied. Further, as lossless methods, it is unclear of how **RC**2 is to be satisfied. Effectively, they are not adaptive representations. Though, adaptive versions of these representations do exist, in particular for wavelets, which we discuss below.

These methods not fulfilling the representation criteria reflect the features of the data and problems they were designed for and not a deficiency in the methods. For example, many of the methods are designed for the typically small size of natural images. Hence, the methods are not focused on reducing the computational and memory costs through a reduction in the size of the data representation. Instead, they are focused on providing 'higher quality' solutions to various processing tasks.

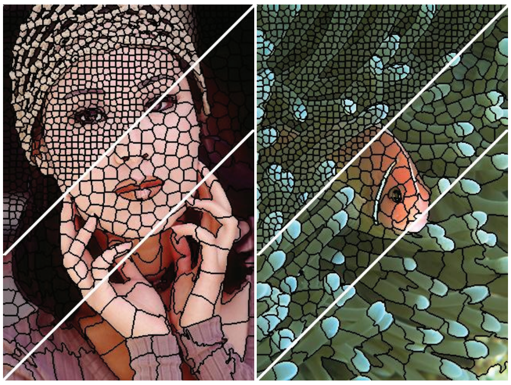


Figure 3.3: Example of superpixels, reproduced from Achanta et al. [2]. The black borders define the edges of the Super-pixels. The three sections represent superpixels set to an average size of 64, 256, and 1024 pixels respectively.

3.3.2 Adaptive sparse transform domain techniques

In this section, we consider adaptive representations, where $\#R \leq N$, that is, they are adaptive and have some non-zero coefficients that are 'selected' and describe R that is less than N. Such representations we call sparse. These representations are sparse in the transform domain, as the coefficients $\{c_i\}_{i=1}^{\#R}$, can not be interpreted as function values of the original data set.

Thresholding of wavelets

Wavelets can be used as an adaptive representation, through a process known as wavelet-shrinkage, or thresholding Donoho and Johnstone [47]. Thresholding involves first calculating the wavelet transform and set of detail $d_{i,j}$ and approximation coefficients a_i for a particular wavelet and scaling function, as shown in 3.5. Wavelet thresholding, or shrinkage, then involves the reduction of, or removal, of particular detail coefficients $d_{i,j}$ based on a given criteria based on their value. This produces a new set of coefficients $\Gamma = \{\hat{d}_{i,j}\}$ which now provide a lossy representation of the function $\hat{f}$. If the procedure sets certain detail coefficients m coefficients to zero, then, the new $\hat{f}$ has a representation with $\#R = N - m$. This reduced set of coefficients can then be used to reduce computational and memory costs and is in a sense optimal for compression [42].

Because wavelets form an orthonormal basis the effect of the removal of coefficients on the squared error is as follows

$$\sum_{i=1}^{n} |f\{x_i\} - \hat{f}\{x_i\})|^2 = \sum_{d_{i,j} \notin \Gamma} |d_{i,j}|^2 \tag{3.6}$$

giving an over-all impact on the approximation error [72]. Although simple, the representations resulting from such thresholding operations have many optimal properties and are successful with wide use, including in bio-medical imaging [128]. We outline some of these optimal properties below.

Optimal error convergence Here we discuss the methods used for comparing different compression schemes through a framework around classing functions in smoothness spaces known as Besov spaces as presented by DeVore et al. [42]. A Besov space $\mathcal{B}_q^{\alpha,1}(L^q(\Omega))$ is a vector space equipped with the norm $\|.\|_{B_q^{\alpha}(L^p(\Omega))}$ (or quasi-norm depending on p and q). The definition is involved, and leave the full definition to Appendix A.2. DeVore et al. [42] indicate that a Besov norm with α is 'roughly' equivalent to having α defined derivatives in $L_p(\Omega)$

Given a function in the α Besov space, they prove, that the optimal rate of convergence of the error compare to the number of coefficients $\#R \to 0$, for an adaptive representation asymptotically follows

$$||f - \hat{f}||_{L_p} \sim \mathcal{O}(\#R^{-\frac{\alpha}{d}}) \tag{3.7}$$

where $||.||_{L_p}$ represents a chosen p norm with $1 \leq p \leq \infty$ and $\frac{1}{q} = \frac{1}{p} + \frac{\alpha}{2}$.

They show that for some threshold ϵ, and using a wavelet that can reproduce all polynomials up to degree α that by setting all coefficients

$$||c_{i,l}\psi_{i,l}||^q_{L_p(\Omega)} \leq \epsilon \tag{3.8}$$

to zero produces an adaptive representation with an error convergence as

$$||f - \hat{f}||_{L_p(\Omega)} \leq C_1^{\alpha/2} C_2 \#R^{-\alpha/2} ||f||_{B_q^{\alpha,1}(L^q(\Omega))} \tag{3.9}$$

which is the stated optimal rate.

Such a scheme does not hold for the infinity norm. However, DeVore et al. [43] provide an alternative iterative scheme, designed for surface approximation for setting of coefficients that find optimal representations following

$$\|f(x) - \hat{f}(x)\|_{L_\infty} \leq \epsilon \tag{3.10}$$

for a given ϵ, similar to the form required in **RC**1.

Optimal representation of noisy functions A second optimal property of wavelet thresholding is that it allows for adaptive representations that have near-optimal de-noising properties [47].

If we consider a noisy signal $f\{x\} = g(x) + \eta(x)$ where $\eta \sim N(0, \sigma)$ is a normally distributed noise process with zero mean and standard deviation σ. Then define an estimated MSE error of $\hat{f}$ for the truncated wavelets as

$$\mathcal{R}(g, \hat{f}) = \frac{1}{n}\mathrm{E}[\sum_{i=1}^{n}(g(x_i) - \hat{f}\{x_i\}))^2] \tag{3.11}$$

and consider the assymptotics of the expected error E as $n \to \infty$, where E is the expectation over all realizations of the noise process. Then the optimal rate achievable by any adaptive scheme is $\frac{1}{n}$, and for a constant width kernel (fixed bandwidth) will at best achieve $\frac{1}{n^{1/2}}$. Donoho and Johnstone [47], show that wavelets using particular shrinkage schemes of the detail coefficients, lead to near optimal ($\mathcal{O}(\frac{\log(n)}{n})$) convergence and hence near optimal representation of noisy functions. (note we used n in replace of N to avoid confusion with the normal distirbution)

Extensions and applications Due, to these properties, amongst others, thresholding of wavelets have been used successfully across a wide range of fields, and applications [128, 114, 85] (The most recognized application being the wavelet thresholding used in JPEG2000 [31]). Because of their wide use and success, the literature is immense, in both application, theoretical work, and extensions. We do not review it here, and direct the reader to Plonka [85] for a recent survey of developments with a focus on image-processing.

Dictionary learning techniques

Wavelet thresholding can be used to form compression schemes that have the optimal asymptotic rate for functions in a given Besov space. However, having optimal asymptotic convergence does not indicate that that the methods produce the best compression rate of any basis [35]. This point can be illustrated by considering a function that is exactly the wavelet for some interval and hence requires only one coefficient. A one-pixel translation of this function will result in the use of many more coefficients. Ideally, such a translation could also be represented by one coefficient. However, such basis functions would no longer be orthogonal. Methods that are developed to find the best basis from a set of non-orthogonal and redundant basis functions are known as dictionary learning techniques. In such a way, very sparse representations (small $\#R$) for a specific dataset can be created. Allowing potentially for significantly greater rates of compression than those obtained through use of an orthonormal basis such as wavelets. However, as the basis function no longer form an orthogonal basis, the coefficients can no longer be found through using the inner-product. In fact, Davis et al. [35] prove that this problem is, in general, NP-Hard. However sub-optimal methods such as greedy schemes can be used [35], with Tosic and Frossard [126] providing a recent review.

Compressed sensing

Related to thresholding of sparse bases, are methods known as compressed sensing [46]. Given a function which is sparse in some transform domain, compressed sensing provides a method by which $M < N$ samples can be used to recover the original function sampled at N locations using appropriate (possibly nonlinear) algorithms. These M samples are chosen non-adaptively. In contrast to thresholding of wavelets or other classic adaptive methods which depend on the function. That is, their number and location are not set with any information of the particular function. However, these samples are not collocation points of the original function, such as pixels, but instead, could be thought of as a specific set of random samples from the transform coefficients. Surprisingly, given knowledge of the class of signal, and distribution of coefficients, the number of M required samples can be shown to be very close to that which could be achieved by optimal adaptive sampling when using knowledge of the function [46].

Reflection on representation criteria

From the discussions above, thresholded wavelet transforms appear to have many of adaptive properties. Including proofs of optimality, that with extensions would allow them to satisfy **RC**1-3. However, the main drawback of these methods, including dictionary learning and compressed sensing, is that the sparse representation exists in the transform domain. Given most processing tasks in LSFM are designed for the pixel domain, many tasks would require the reconstruction of the original image and therefore return to a data size of N. It is true that many tasks, such as deconvolution [48], or image fusion [100] can be done efficiently in the wavelet domain. It is not clear, however, how existing approaches for tasks such as tracking and segmentation could be adapted when the approaches rely on pixel graphs. A similar argument also applies to other adaptive compression schemes, such as Zhao et al. [142], although these methods are highly effective at solving the individual processing task of compression, it is unclear how the spatial adaptation and reduction in time could be used for general processing tasks.

3.3.3 Adaptive sparse collocation techniques

The last group of adaptive representations is those that are sparse $\#R \leq N$ and the coefficients $\{c_i\}_{i=1}^{\#R}$ can be interpreted as collocation points of the function. That is, each coefficient c_i can be mapped to a location $\mathbf{y}$ and $c_i = f(\mathbf{y})$.

From surveying the literature, historically, the main theoretical contributions of sparse collocation representations have been in fields outside of image processing. Namely, the representation of functions for computational differential equations and statistics. In computational differential equations, the drive for adaptive collocation representation arises when the problems being solved have a large range of spatial and temporal scales [58]. As for LSFM data and processing, for numerically differential equations exhibiting large differences in spatial scales, the processing task in can become very costly and inefficient when using homogeneously sampled grids. As previously, we have made an effort to isolate the main ideas and attribute references to the original methods.

Equidistribution techniques for solving differential equations

The first idea we discuss is the general principle of equidistribution that was originally introduced by de Boor [37], Burchard [25] as a method for the adaptive placement of knots for splines. Splines are piecewise polynomials. In general, the problem that is solved by equidistribution is choosing the location, and potentially number, of collocation points $\hat{x}$ such that are then used to represent a sampled function $f\{\hat{x}\}$ for some processing task. Depending on the application, task, and method, a monitor function is defined $M(x)$ over the domain Ω. The monitor function is often related to the error, some other property of the function, or prior information of the solution. The idea is that the nodes should be placed such that

$$\int_{x_i}^{x_{i+1}} M(y)dy = A \tag{3.12}$$

for $i = 0, ..N-1$, and A is some constant.

The concept can be understood by the following simple example. Consider, the monitor function $M(x)$ as a measure of the error local error 'density'. In this way, the error between two points is simply then the integral between the two points. Then by finding a $\bar{x}$ that satisfies Equiation 3.12, coincides with making the error between any two points the same. Therefore, in areas of the domain where the solution error is likely to be high, the function will be sampled with high density, and areas with the solution error is likely to be low, are sampled with lower density. This is exactly the problem that was solved in 1D by a simple algorithm in De Boor [38] for the placement of knots for splines. The algorithm was derived by the following error estimate for splines of order k

$$\|f - \hat{f}\|_\infty \leq \frac{1}{k!} \left(\int_{x_i}^{x_{i+1}} |f^{(k)}(r)|^{1/k} dr \right)^k, \tag{3.13}$$

where $\|f - \hat{f}\|_\infty$ is the maximum error over the interval x_i and x_{i+1} and $f^{(k)}$ is the k-th derivative of f. From this, they derived the monitor function of $M(x) = |f^{(k)}(r)|^{1/k}$. De Boor [38]'s proposed method is simple and effective, however, does not have a simple extension to higher dimensions [58]. Similar ideas of equidistribution are used by a class of methods known as Adaptive Method of Lines [105], however, again these methods are largely in 1D.

Additional methods have been developed that use different approaches and monitor function, but share the equidistribution principle and can extend to arbitrary dimensions. For example, the class of methods known as Moving Mesh Partial Differential Equation (MMPDE) methods [59]. MMPDE methods are similarly based on a monitor function and equidistribution but are focused on time evolution and extendable to arbitrary dimensions. MMPDE's define an additional evolution equation for mesh nodes, that equidistributes the mesh through time [58]. Instead, using pseudo-forces and node addition and removal rules, Reboux et al. [93] provided an equidistribution method, again for arbitrary dimension, based on Lagrangian particles for the adaptive solution of advection-diffusion equations. See Huang and Russell [58], for a recent review of different equidistribution techniques and references.

Separately, another class of adaptive methods have been developed based on a similar equidistribution principle [78] for mesh generation for adaptive finite element methods [17]. Adaptive Finite ElementMethods involve the solution of weak-form partial differential equations over elements, often triangulations on mesh nodes [78]. In the approach, instead of setting some fixed constant A, the problem is posed as finding the placement of fixed N, in such a way that integrals, of a similar form to Equiation 3.12, all have some *a posteriori* set average value. In addition to adapting by placement of the mesh nodes (known as h-adaptation), methods also adjust the degree of the approximation on a given element to again reduce the solution. Babuška and Guo [15], showed for certain applications the combination, hp-adaptation, this can result in exponential convergence. These adaptive ideas have also been generalized to a larger class of methods, without distinct elements, known as Partition of Unity Finite Element Methods (PUFEM) [15].

Adaptive Mesh Refinement (AMR)

Another class of successful adaptive collocation schemes is Adaptive Mesh Refinement (AMR) methods [18]. AMR methods use nested grids of different resolutions that are refined, or coarsened, through time using an estimate of the local truncation error of the time stepping error.

Mesh-based adaptive image representations

In recent years, there has been the development of adaptive image representations using linear triangulations [135, 3, 62, 141, 40, 104]. These methods rely on representing the image using meshes as often used in FEM discussed above. These methods have, (with the exception of Yang et al. [141] that was designed for tomographic projection), been developed as compression methods. Often, motivated by the poor performance of some wavelet methods for approximating sharp image edges [40].

Figure 3.4, shows an example of a mesh-based adaptive image representation, called adaptive thinning, presented in Demaret et al. [40]. These methods have the advantage of being able to provide highly anisotropic adaptation to spatial scales. Different methods have been used for the generation of these meshes, including concepts similar to error equidistribution [141, 3] using error diffusion [50], greedy point removal schemes [40], and the use of binary spanning trees [104]. Sarkis and Diepold [104] reviews a range of these methods. They find that the greedy point removal method of Demaret et al. [40] provides the highest compression rates. However, this comes at a significant increase in relative computational cost. This high compression performance, with high computational cost, was the motivation of the approaches in Adams [3], that uses an error equidistributed initialization for the greedy point removal scheme of Demaret et al. [40] to reduce memory and decrease execution time. Similar methods, using more general polygons and binary space partitionings have also been previously developed, see Radha et al. [92].

On an aside Agarwal and Suri [6], in the context of computer graphics, proves that finding the optimal mesh triangulation similar to above, that satisfies, $\|f - \hat{f}\|_\infty \leq \epsilon$ is a form of an NP-Hard problem.

Collocation based wavelet methods

In the previous section, we discussed sparse adaptive representations in the wavelet domain. These methods have excellent adaptation properties are can be proven to be in certain aspects optimal. However, the sparsity is in the transform domain, and therefore to be utilized processing tasks must be done there. This requirement can cause issues for specific problems that are more

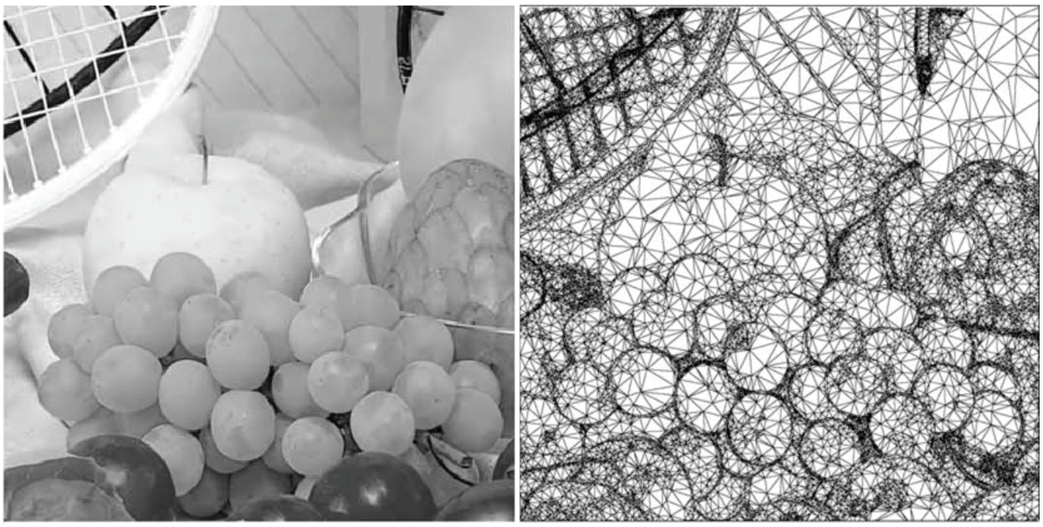


Figure 3.4: Adaptive image representation using triangulated mesh using the Adaptive Thinning Method introduced in Demaret et al. [40]. The *left* image shows the reconstructed pixel image from the representation and *right* a representation of the triangulated mesh. (Reproduced from Demaret et al. [41])

suited to the original function domain. To overcome such issues for solving PDE's [19], new wavelet techniques [55, 19, 131, 97] have been developed based on concepts such as interpolating [45], and second generation, wavelets [131, 120]. These methods, often called collocation wavelet methods, utilize the adaptivity properties of wavelets, but the approximation coefficients $a_{i,j}$ are collocation points of the function, and detail coefficients $d_{i,j}$ a measure of error in some interpolation scheme. Thresholding of detail coefficients can then be used to form a grid, that can utilize both wavelet, and function domain methods. Schneider and Vasilyev [114] provides a recent review of these methods.

Collocation wavelet methods, often use dyadic multi-resolution grids [131], meaning each point at a lower grid resolution is also in all higher grid resolutions. Figure 3.5, shows an example of such an adaptive dyadic grid and solution for the 1D Burgers equation solved by methods from Vasilyev and Bowman [131]. The methods allow localization in the function domain, concentrating grid points, and hence computational effort, adaptively across the spatial domain. An example of this adaptation is shown in Figure 3.6, showing the adaptation grid in $2D$ for solving for a shock problem from gas dynamics reproduced from Regele and Vasilyev [94]. Instead of simply relying on thresholding, the methods usually require additional heuristic steps of adding support nodes, and ghost layers [131, 97], around the solution to make the spatial adaptation from the wavelets usable for the processing steps. The use

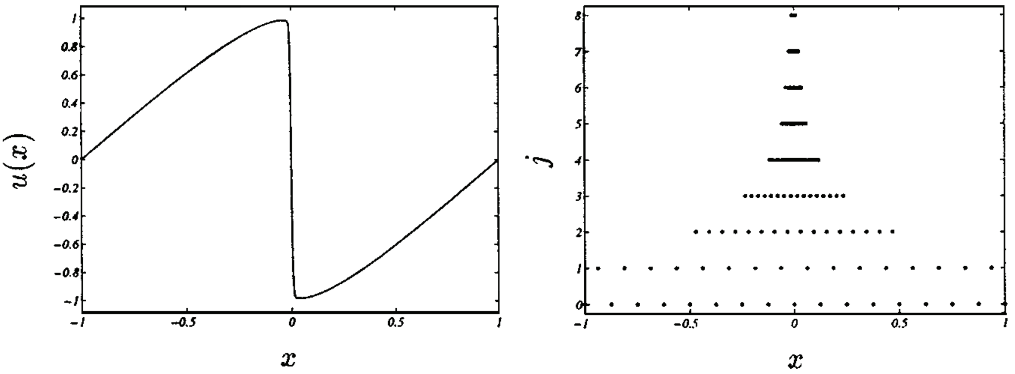


Figure 3.5: Example of an adaptive dyadic grid produced by the collocation wavelet method presented in Vasilyev and Bowman [131] for the $1D$ Burgers equation. The *left* plot shows the solution $u(x)$ at $t = \frac{1.5}{\pi}$ and *right* plot shows the dyadic adaptive grid also at $t = \frac{1.5}{\pi}$, where j, represents the resolution proportional to 2^{-j}. (Reproduced from Vasilyev and Bowman [131])

of interpolating wavelets [45] also allows for the computational of the detail coefficients independently, allowing for distributed parallel implementations such as Rossinelli et al. [97], that would likely not be possible with traditional wavelet domain techniques.

The error of these methods, can be controlled proportional to the threshold parameter ϵ across the domain by bounds like

$$||f - \hat{f}||_{L_2} \leq C_1 \epsilon ||f||_{L_2} \tag{3.14}$$

where C_1 is a constant that depends on f [131]. f [131]. The method can be extended for sufficiently regular functions to bound higher order derivatives [131].

Adaptive statistical methods

In statistics, there has been the development of methods for adaptive smoothing and regression of noisy data using adaptive data-representations [47]. Donoho and Johnstone [47], surveys these methods, showing that they can be interpreted as a combination of some adaptive sampling $f\{\bar{x}\}$ and a smoothing function $\delta(x)$, and a reconstruction scheme using the two. Since $\#\hat{x} < N$, these methods can be considered as sparse collocation adaptive representations.

The different methods align with using different reconstruction functions and definitions of $\delta(x)$. The classification and regression trees method (CART) [22], utilizes a piece-wise constant reconstruction and smoothing function. The

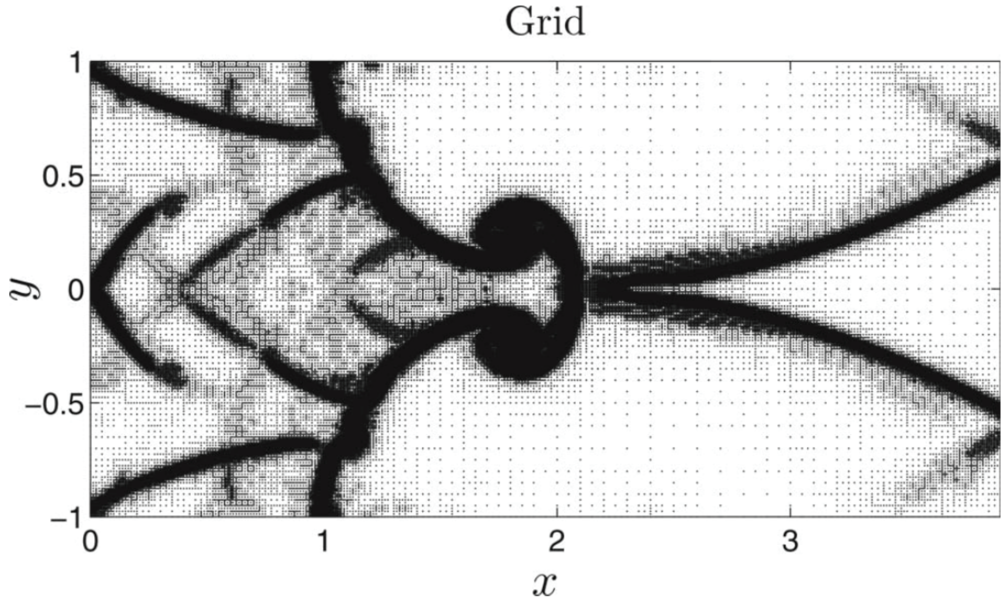


Figure 3.6: Example of a 2D adaptive dyadic grid produced by the collocation wavelet method presented in Regele and Vasilyev [94] for solving two-dimensional Richtmyer Meshkov instability problem at Atwood number of 0.67 with $t = 2.65$ units. (Reproduced from [94])

adaptive knot placement for spline regression [52, 51], also called MARS models, use piece-wise polynomial reconstructions, with piece-wise constant $\delta(x)$. Kartal Koc and Bozdogan [66] provides a recent review. Lastly, there are adaptive bandwidth approaches for kernel regression [24]. These methods allow more general form of reconstruction using compact supported kernel functions, and continuous $\delta(x)$. Köhler et al. [69] provides a recent review.

Reflection on RC

The last group of methods appears to be closer to satisfying our representation criteria. They incorporate adaptation to different spatial scales, while adaptively sampling to reduce computational and memory costs to $\#R < N$. Also, the adaptive sampling is in the function domain, likely making implementation of classical 'pixel' algorithms 'easier' across a wide range of tasks.

Across the methods, there is usually a user-specified threshold or parameter that allows proportional adjustment of the approximation error. However, this control is usually global, and with exceptions, do not pointwise bound the error as specified in **RC**1. Further, as is, they do not provide a means for the incorporation of a local intensity scale or gain control.

Regarding computational cost and **RC**3, we require the methods to be

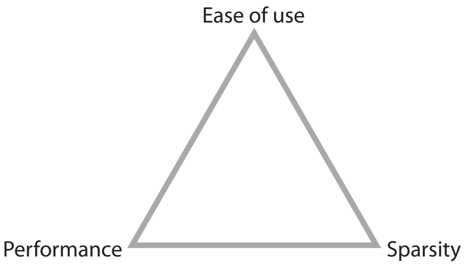


Figure 3.7: Schematic representing the trade-offs between a representation having high sparsity, i.e. requiring a low number of coefficients, high performance regarding individual operations, and ease of use regarding developing and implementing new algorithms.

able to scale to large datasets ($> 1000^3$). From the execution time analysis of Sarkis and Diepold [104] and Adams [3], the adaptive mesh methods, appear likely to be too computationally costly. With the methods ranging from $.5 - 40$ seconds to compute the solution on as small as a 512×512 test image when optimized and implemented in C++. Making them unlikely to efficiently scale without further development to images up to 4000 times larger. The methods that have been developed for PDE's, such as the wavelet collocation methods, AMR, and AFEM, likely exhibit better scaling behavior. These methods often have distributed parallel memory implementations [97] allowing them to scale to very large problems. A caveat here is, is that these algorithms a largely designed for adaptation of the solution through time. Therefore, the input for the next time-step is simply the adaptive representation for the last dataset. Hence they are not optimized, or implemented, for transformation to an adaptive representation from an arbitrary solution on large regular grids as is the case for LSFM data.

A final point relates to **RC**4 and an adaptive representations ability to translate the sparsity of representation into decreases in memory and computational cost for a range of algorithms. As discussed by, Rossinelli et al. [97] and Berger and Colella [18], there is often a trade-off between the sparsity of the representation $\#R$ and the complexity and cost of implementing, and performing processing tasks, using R. For example, the higher the degree of relative sparsity a representation can achieve, often comes at the expense of more complicated data structures, implementation complexity, and costly individual operations over elements. Hence, for certain representations and applications, it is more efficient to reduce the level of adaptivity to simplify data-structures and speed up costs of individual basic operations [97, 18].

3.4 Discussion

The first observation from the review and discussion is the high popularity and success of adaptive representations across different areas of application. With developments ranging from, image processing, computer graphics, statistics, and computational differential equations. With many core concepts, such as the use of tree data structures, being common across many approaches, but the exact implementation and emphasis reflecting the particular features of the application and data. Hence, although no representation from the review meets our representation criteria in full, this not through some deficiency. Instead, reflecting that the combination of features and intended use-case set out by the representation criteria not matching those of those intended for existing methods. With LSFM data combining the challenge of varying spatial scales in image processing with the computational and memory issues faced when numerically solving particular partial differential equations.

From the three groups of methods discussed above, it seems that sparse collocation adaptive methods are closest to satisfying the representation criteria. I conclude however, that there are two main concepts are not adequately addressed in the literature and discuss them below.

3.4.1 Lack of local gain control (RC1)

The first is how to incorporate a local intensity scale into the error control as outlined in RC1. However, with additional work, it is likely that many of the above methods could be extended to include such a concept.

3.4.2 Use across a wide range of tasks (RC4)

The second relates to **RC**4, and determining the appropriate trade-off between sparsity, performance, and ease of use to all a representation to be used across a range of tasks originally designed for pixel images. We highlight the trade-off between these factors in Figure 3.7 with a schematic triangle. Where the basic intuition is, the greater sparsity the method achieves, the greater complexity of the basis representation resulting in the higher computational cost per coefficient and complexity in use. Further, the more simple the representation is to use likely comes at the cost of not using highly optimized data-structures and algorithms. Of course, the ideal representation would provide the highest of all three. Assessing **RC**4 is not aided in that the goal of is rather ambiguous. However, in the discussion of **RC**4, we introduce the notion of different interpretations of pixels that are utilized by different methods. **RC**4 can be satisfied by a representation both naturally sharing the interpretations of pixels while reducing computational and memory costs across them. In essence, be as close to pixels as possible, while still achieving the other representation criteria.

Interpreted in this way, subjectively, it seems that representations requiring the use of basis functions or triangulations for their use, may be 'too far' from the original pixel images to be widely used. Ideally, the algorithm would allow the most simple interpretation of pixels, as local piece-wise constant patches, and also allow continuous representation as a smooth function, a particle graph, and also as a tree structure.

3.5 Summary and main points

In the above chapter we have first outlined, and then discussed, five key properties that we propose a representation should have to as a general representation solution for studying spatiotemporal processes in biology (STB) using Light-sheet Fluorescence Microscopy (LSFM) data as described in Chapter 2. We named these five criteria, the *representation criteria* (**RC**). Following this, we reviewed the different ideas and approaches of existing multi-resolution and adaptive representations and reflected on how they could meet the representation criteria. We concluded by discussing the results and highlighting two areas, local error adaptation and integration of a gain control (**RC**1), and the ability to easily be used across a range of process tasks (**RC**4) as the main areas requiring further development.

Summary of the chapter

- Introduced the five Representation Criteria **RC**, outlining desired properties of an adaptive representation for processing on LSFM data
- Provided definitions and surveyed the literature of multi-resolution and adaptive-representations, focusing on core ideas, across research areas
- Identified sparse collocation adaptive representations as closest match to the representation criteria
- Concluded further development was needed to satisfy **RC**1 through local error adaptation including a 'local gain control', and increased simplicity and similarity to pixel images for the satisfaction of **RC**4

4 The Adaptive Particle Representation (APR)

Contents

4.1 Adaptive Particle Representation (APR) 50
4.1.1 Main results 51
4.1.2 Motivation of formulation 52
4.1.3 Extensions 53
4.2 APR description 54
4.2.1 Reconstruction Condition and Resolution Bound . . 54
4.2.2 Particle Cells and Implied Resolution Functions . . . 59
4.2.3 Pulling Scheme 63
4.2.4 Choosing the Particle Set $\mathcal{P}$ 68
4.2.5 Forming the APR=$\{\mathcal{V},\mathcal{P}^*\}$ 71
4.2.6 Summary 71
4.3 Practical considerations 71
4.3.1 Discrete sampling 72
4.3.2 Impact of noisy Local Resolution Estimate $L(\mathbf{y})$ . . 72
4.3.3 Impact of noisy particles $\hat{f}(y_p)$ 73
4.3.4 Comparison with continuous resolution functions . . 74
4.3.5 Particle Cells and smoothness of the Local Intensity Scale 75
4.3.6 Reconstruction of higher order derivatives 75
4.4 APR transform steps summary 76
4.5 Summary and main points 77

In this chapter, we introduce the Adaptive Particle Representation (APR), an adaptive sparse collocation representation designed for LSFM data. The development of the APR represents the main theoretical contribution of this thesis. The APR is motivated by the processing bottleneck and issues raised in Chapter 2 and designed to fulfill the representation criteria (**RC**) outlined in Chapter 3.

The chapter is structured as follows. First, I present the mathematical problem to which the APR is a solution. The explanation takes multiple steps, first with the derivation of two conditions, the *Reconstruction Condition*, and *Resolution Bound*, that we require the representation to satisfy. Then using an introduced concept of Particle Cells we derive the *Pulling Scheme* an efficient algorithm for satisfying the conditions. The solution is first derived and explained in 1D, with formal proofs for the general dimensional problem presented later in the next chapter. I have structured the chapter this way to first focus on the ideas and concepts, and later focus on the technical details.

4.1 Adaptive Particle Representation (APR)

The Adaptive Particle Representation (APR) takes a regularly sampled input function, such as pixel images, and resamples it as a set of particles $\mathcal{P}$ and a Resolution Function $R(y)$ defined for all locations y in the domain Ω. Where particles p are collocation points in space, x_p that 'carry' properties evaluated at that location, for example the function value $f_p = f(x_p)$ (or an estimate). In this way, then $\mathcal{P}$ is the set carrying all information for describing N_p particles, i.e. $\mathcal{P} = \{\{x_p\}_{p=1}^{N_p}, \{f_p\}_{p=1}^{N_p}\}$, being extended to include sets of other properties if required. The Resolution Function $R : \Omega \to \mathbb{R}$ defines a local isotropic neighborhood $\mathcal{N}$ at each point in the domain $\Omega \subset \mathbb{R}$ defining a subset of particles that can be used in the reconstruction of the function at that y. Therefore, $R(y)$ defines a isotropic spatial length scale at every point in the domain.

Formally, we consider a differentiable function $f : \Omega \to \mathbb{R}$, that is known at sampled location with fixed spacing, and is denoted as $f\{\bar{x}\}$ where $\#\bar{x} = N$. We represent the function for a given $\mathcal{P}$ and $R(y)$ in the following way

$$\hat{f}(y) = \sum_{x_p \in \mathcal{N}(y,R(y))} f(x_p)\xi_p(y) \tag{4.1}$$

where $\mathcal{N}(y, R(y)) = \{x \in \Omega : |x-y| \leq R(y)\}$ and $\xi_p(y) = \xi(y, x_p)$ are constants that satisfy $\sum_{x_p \in \mathcal{N}(y,R(y))} \xi_p(y) = 1$ with $\xi_p(y) \geq 0$[1]. Where $x_p \in \mathcal{N}(y, R(y))$, means all particles in $\mathcal{P}$ that are in the neighborhood defined by $\mathcal{N}(y, R(y))$. The Resolution Function is set such that the reconstruction follows,

$$\|\frac{f - \hat{f}}{\sigma}\|_\infty \leq E \tag{4.2}$$

---

[1] This positivity constraint can be relaxed, with a slight adjustment to the results, with addition of reconstruction dependent constant

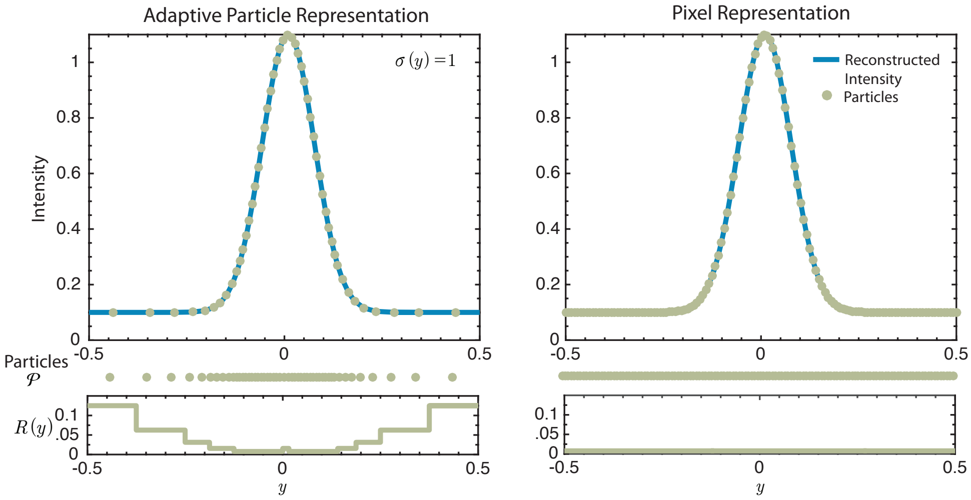


Figure 4.1: The *left* panel shows the Adaptive Particle Representation (APR) ($E = 0.1$ and $\sigma(y) = 1$) (left) and pixel representation sampled with $h = 0.0078$ (*right*) of a $1D$ guassian pulse ($I(y) = e^{\frac{-(y-0.01)^2}{0.009}} + 0.1$) in terms of a resolution function $R(y)$ (*bottom*), and set of particles $\mathcal{P}$ (*middle, top*).

where $\|\mathbf{x}\|_\infty = \max_{x_i \in \bar{x}} x_i$, E is a user-specified relative error threshold and $\sigma(y)$ called the Local Intensity Scale ($\sigma : \Omega \to \mathbb{R}$, and is required to satisfy an additional 'smoothness constraint' given in 4.15). We call this bound the *Reconstruction Condition*, and this holds for any ξ satisfying the conditions above and any $\mathcal{P}$, where $\#(x_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))) > 0$ for all $y \in \Omega$ (That is, there is at least one support particle in the isotropic neighborhood set by $R(y)$). Figure 4.1, provides an example of a 1D Gaussian function represented as described by $\mathcal{P}$ and $R(y)$. The *left* pane shows the Adaptive Particle Representation, and the *right* shows a pixel image also intepreted as above.

4.1.1 Main results

Now we will briefly describe the main result of this thesis. The APR places two further restrictions on the problem defined above. First, we constrain the resolution function $R(y)$ to also satisfy

$$R(y) \leq \min_{x \in \mathcal{N}(y, R(y))} (L(x)) \tag{4.3}$$

where $L(y) = |\frac{E\sigma(y)}{\frac{\partial f}{\partial y}}|$ is called the *Local Resolution Estimate* and we assume we have access to $\frac{\partial f}{\partial y}$. We call the constraint 4.3 the *Resolution Bound*. If $R(y)$ satisfies the esolution Bound, then it also satisfies the Reconstruction

Condition (for special σ). We note, that when $\frac{\partial f}{\partial y} = 0$, and hence $L(y)$ is not defined, we can interpret this simply as the point not placing any constraint on the global resolution function at that point. Therefore, practically, divergent $L(y)$ does not present an issue. Second, we constrain $R(y)$ to be an *Implied Resolution Function* $R^*(y)$, that is, it is constructed out of piecewise constant blocks we call *Particle Cells.*

Result 1

Here we present a worst-case linear complexity in N algorithm, known as the Pulling Scheme, that can find the optimal $R^*(y)$ and particle set $\mathcal{P}$ that satisfy problems in the form of the Resolution Bound (4.3) for general $L(y)$. Where the optimal Implied Resolution Function is the R^* that satisfies

$$\underset{R^* \in \mathcal{R}^*}{\arg\max} \int_\Omega R^*(y) dy \tag{4.4}$$

where $\mathcal{R}^*$ is the set of all Implied Resolution Functions $R^*(y)$ that satisfy the Resolution Bound 4.3. The optimal $\mathcal{P}^*$ is then the particle set that satisfies, $\#(x_p \in \mathcal{N}(y, R(y))) > 0$, and $\#\mathcal{P} = \int_\Omega \frac{1}{R(y)} dy$.

Result 2

Given the local resolution function $\sigma(y)$ is sufficiently slowly varying (see 4.15), and $L(y) = |\frac{E\sigma(y)}{\frac{\partial f}{\partial y}}|$, then the reconstructions $\hat{f}$ formed using $R^*(y)$ and $\mathcal{P}$ will satisfy the Reconstruction Condition 4.2.

Result 3

The Implied Resolution Function $R^*(y)$ and Particle Cell set $\mathcal{P}$, can be completely described by a set of Particle Cells $\mathcal{V} = \{\{c_{i_p,l_p}\}_{p=1}^{N_p} | (i_p, l_p) \in \mathbb{Z}\}$ (i.e. it can be defined by N_p length sets of integers) and $\mathcal{P}^* = \{\{f_p\}_{p=1}^{N_p}\}$. The combination of these two sets $\{\mathcal{V}, \mathcal{P}^*\}$ we call the Adaptive Particle Representation (APR). [2]

4.1.2 Motivation of formulation

Before explaining and proving the above results in detail, we shall first briefly reflect on the motivations/advantages for formulating the APR in this way.

---

[2] With the difference between $\mathcal{P}$ and $\mathcal{P}^*$ being the explicit storage of the particle locations in the former

Reconstruction Condition

First, we consider the Reconstruction Condition and form of function representation. First, the Reconstruction Condition allows direct fulfillment of **RC**2 and makes the inclusion of a local intensity scale central to the representation and pointwise error control. Second, the explicit inclusion of Resolution Function gives a direct measure of a minimum local length scale at each point. This Resolution Bound is analogous to the smoothing scale discussed by [47] and discussed in the previous chapter. We show later in Chapter 6 this adaptation allows the APR to scale linearly with the information content regarding the Object function O (**RC**1).

Function reconstruction

The choice of function reconstruction in 4.1 differs from most representations discussed in Chapter 3 in that it does not depend on a specific reconstruction basis or kernel. This form was motivated by **RC**4, and the need for the APR to be used across a wide range of processing tasks. This represents an attempt to be as 'close to pixel images as possible' while still satisfying the RC. The formulation of 4.1 allows a wide range of interpolation, or reconstruction, methods to be used to form $\hat{f}$. Ranging from piecewise constant representations of the image to smooth differentiable representations possibly used B-splines, wavelets, or Partition of Unity [16] formulations. Formulated in this way the APR guarantees that all such reconstruction methods will satisfy the Reconstruction Condition if the coefficients satisfy the stated conditions and only sample locations within that defined by the Resolution Function. This general form of reconstruction comes at the cost of the sparsity of the representation.

Implied Resolution Function and Particle Cells

As is detailed below, the use of Particle Cells is central to the development of the Pulling Scheme and satisfaction of **RC**3. Further, the use of Particle Cells allows for efficient storage in memory of both the Resolution Function and particle locations.

4.1.3 Extensions

The APR as discussed in this and the following three chapters represent the most simple out of a class of representations that can be formulated similarly to above. One extension is a series of representations that require the reconstruction coefficients satisfy an increasing number of moment conditions, and adapt to higher order derivatives of the function. Other extensions include additional reconstruction conditions placed on higher derivatives of the function and anisotropic resolution functions. These representations are discussed in Chapter 9. Although these extensions can result in greater sparsity, they come at the sacrifice of simplicity and generality (The trade-off shown in Figure 3.7). Because our goal here is a general-use representation, we focus our attention in the next chapter to the simplest model as proposed above.

Lastly, an important extension is the incorporation of time to allow for satisfaction of **RC**5. We discuss the extension of the APR to account for time in Chapter 10, and it follows the ideas presented here, but adapted to the unique features of time. The extension is done such that it directly uses this space-adapted APR presented in this Chapter.

4.2 APR description

In this section, we describe and explain the derivation of the main results stated above in 1D for simplicity of notation and explaining ideas. The following chapter presents the general dimension case and provides the proofs of theorems referred to in this section. First, we go through the derivation of the Resolution Bound, and how it relates to the Reconstruction Condition. Next, we present Particle Cells, and how they can be used to find Implied Resolution Functions that satisfy the Resolution Bound. Lastly, we present the Pulling Scheme, the algorithm that allows efficient formation of the APR.

4.2.1 Reconstruction Condition and Resolution Bound

Let us continue with the problem as outlined above, and consider the function represented as

$$\hat{f}(y) = \sum_{x_p \in \mathcal{N}(y, R(y))} f_p \xi_p(y) \tag{4.5}$$

where we do not assume any particular distribution of particles $\mathcal{P}$, but assume there is at least one $p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))$ for all y. We then consider the reconstruction error at each point $y \in \Omega$ as

$$\epsilon(y) = f(y) - \hat{f}(y), \tag{4.6}$$

which by assuming the function has a continuous derivative can express this by taking Taylor series expansions of f_p centered at y and using the integral form of the remainder [103] as,

$$\begin{aligned}\epsilon(y) &= f(y) - \sum_{x_p \in \mathcal{N}(y,R(y))} f_p \xi_p(y) \\ &= f(y) - \sum_{x_p \in \mathcal{N}(y,R(y))} \Bigg(f(y)\xi_p(y) + \\ &(y - x_p)\xi_p(y) \int_0^1 \frac{\partial}{\partial y} f(y + s(x_p - y)) ds \Bigg)\end{aligned} \tag{4.7}$$

now by using that $\sum_{x_p \in \mathcal{N}(y,R(y)))} \xi_p(y) = 1$

$$\epsilon(y) = \sum_{x_p \in \mathcal{N}(y,R(y))} (y - x_p)\xi_p(y) \int_0^1 \frac{\partial}{\partial y} f(y + s(x_p - y)) ds. \tag{4.8}$$

We can bound this exact expression of the error, using a uniform estimate, by bounding each integral using the maximum gradient over the interval and using the triangle inequality and the fact that by definition $|(y - x_p)| \leq R(y)$ we get

$$|\epsilon(y)| \leq \left(\sum_{x_p \in \mathcal{N}(y,R(y))} |\xi_p(y)| \right) R(y) \max_{x^* \in \mathcal{N}(y,R(y))} |\frac{\partial f(x_i)}{\partial y}| \tag{4.9}$$

and in now assuming[3] also $\xi_p > 0$ therefore $\left(\sum_{x_p \in \mathcal{N}(y,R(y))} |\xi_p(y)| \right) = 1$ so we get

$$|\epsilon(y)| \leq R(y) \max_{x^* \in \mathcal{N}(y,R(y))} |\frac{\partial f(x^*)}{\partial y}|. \tag{4.10}$$

Now returning to the Reconstruction Condition, we can re-write the infinity norm as a bound on each $y \in \Omega$ as

$$|\epsilon(y)| \leq E\sigma(y). \tag{4.11}$$

So, using 4.10, 4.11 will be satisfied, if

$$R(y) \max_{x^* \in \mathcal{N}(y,R(y))} |\frac{\partial f(x^*)}{\partial y}| \leq E\sigma(y) \tag{4.12}$$

---

[3] The procedure from here can be done without this assumption, however this leaves the sum of the coefficients in the resulting expressions

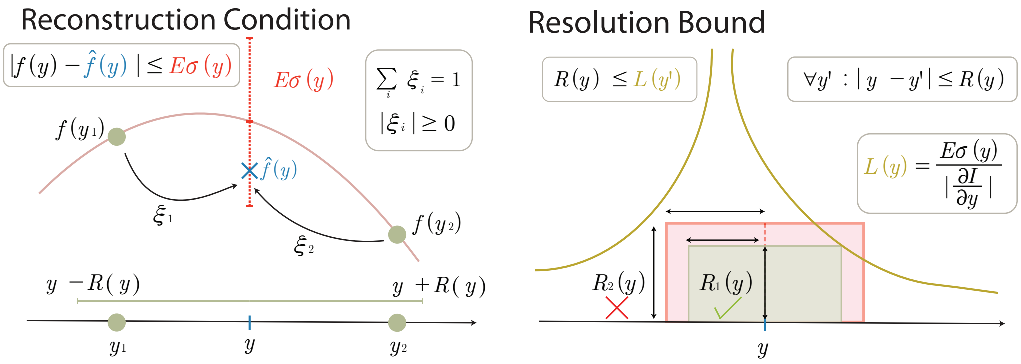


Figure 4.2: Schematics showing the Reconstruction Condition 4.2 (*left*) and Resolution Bound 4.3 (*right*). The Reconstruction Condition, requires that at each y the reconstruction error for $\hat{f}(y)$ is below $E\sigma(y)$ for any coefficients ξ_i satisfying that stated conditions. The Resolution Bound, instead places a geometric bound on the relationship between the Local Resolution Estimate $L(y)$. Requiring that a rectangle centered at y, with height $R(y)$ and width $2R(y)$ does not intersect $L(y^*)$ for all $y^* \in \Omega$. Two possible values of the resolution function at y are shown. One that satisfies the Resolution Bound, $R_1(y)$ and one that does not $R_2(y)$.

which we can formulation in terms of the Resolution Function as

$$R(y) \leq \frac{E\sigma(y)}{\max_{x^* \in \mathcal{N}(y,R(y))} |\frac{\partial f(x^*)}{\partial y}|}. \tag{4.13}$$

This we can then re-write as

$$R(y) \leq \sigma(y) \min_{x^* \in \mathcal{N}(y,R(y))} (g(x^*)) \tag{4.14}$$

where $g(x) = \frac{E}{|\frac{\partial f(x)}{\partial y}|}$ which we can see is almost the Resolution Bound, only the local intensity scale $\sigma(y)$ is outside the max.

Restriction on Local Scale Function

To get 4.14, into the form the Resolution Bound requires an assumption that

$$\sigma(y) \min_{x \in \mathcal{N}(y,R(y))} (g(x)) = \min_{x \in \mathcal{N}(y,R(y))} (\sigma(y)g(x)) \tag{4.15}$$

this, therefore, provides a constraint for the information scale $\sigma(y)$ to ensure this approximation is valid. For this to approximately hold, $\sigma(y)$ must be sufficiently slowly varying. That is it must be approximately constant over $\mathcal{N}(\mathbf{y}, R(\mathbf{y}))$. In general, this can not be guaranteed except in the case where $\sigma(y) = \sigma_0$ is a constant. However, in our examples, results are shown in Chapter 6 reflect that the reconstruction condition still holds when $\sigma(y)$ is a smoothed local estimate of the range of $f(y)$. Further, the restriction is slightly relaxed, through the use of Particle Cells, as discussed in 4.3.5.

Resolution Bound

Then, given this being satisfied, or an approximation, we have the Resolution Bound,

$$R(y) \leq \min_{x \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} (L(x)) \tag{4.16}$$

where, $L(y) = \frac{E\sigma(y)}{|\frac{\partial f(y)}{\partial y}|}$ is called the Local Resolution Estimate. The specific form 4.16 is required for the algorithms used below.

Intuition for Resolution Bound

We shall briefly reflect on the Reconstruction Condition and Resolution Bound to gain some intuition on their difference. The Reconstruction Condition is the easier of the two to interpret, and is illustrated in *left* panel of Figure 4.2. The Reconstruction Condition requires that at each point $y \in \Omega$, the reconstruction by any combination of positive coefficients ξ summing to one should be within $E\sigma(y)$ of the function $f(y)$. In Figure 4.2 ((right)), we illustrate the Resolution Bound. The Resolution Bound requires that a rectangle centered at y of height $R(y)$ and width $2 * R(y)$ does not anywhere intersect with the curve $L(y)$. To give further intuition on how L then relates to the derivative, and a bound on f, we show them in Figure 4.3. In the top panel we show an example of a value of $R(y^*)$ that violates the Resolution Bound in red, and a sub-optimal $R(y^*)$ that satisfies the Resolution Bound in blue. It is sub-optimal, as a larger $R(y^*)$ would also satisfy the Resolution Bound. The optimal $R(y^*)$ is given in the *bottom* panel in green.

Continuous solutions

From the formulation above, we know that the Resolution Bound represents an equal or tighter bound on $R(y)$ then the Reconstruction Condition. That is the optimal solution (largest everywhere) to the Resolution Bound $R_c(y)$ will always be less than or equal to the optimal solution to the Reconstruction Condition $R_b(y)$. This difference comes from the constant approximation to the integral in 4.8. In the limit as $R(y) \to 0$, the two become equivalent. The dashed lines in Figure 4.4 show this for the $1D$ Gaussian example. Where by continuous in practical examples we mean the solution can take any value corresponding to $i * h$, where h is the sampling resolution, and i is a positive integer. These solutions have been calculated using a brute-force approach. The Reconstruction Condition can be solved by ensuring the worst-case reconstruction has an error less than $E\sigma$. This can be done by finding the value

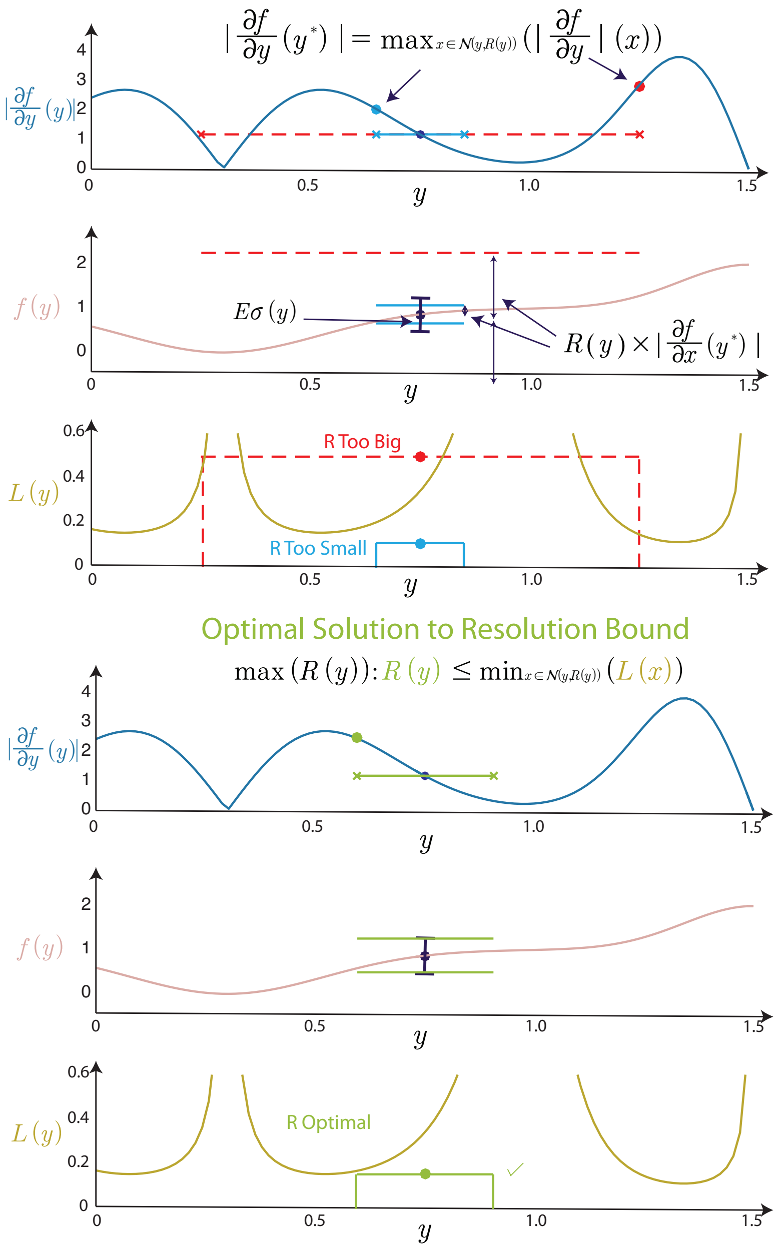


Figure 4.3: An example of the optimization problem for finding the optimal resolution $R(y)$ for a given point y and function $f(y)$ that satisfies the Resolution Bound and how it relates to the function and its derivative. The *top* plots shows a $R(y)$ that is too small in blue, and too large in red. The uppermost plots show the absolute value of the gradient of the function, and its maximum value in the isotropic interaction neighborhood $\mathcal{N}(y, R(y))$. The *second* plots show the function $f(y)$ and also the two error bounds for the reconstruction at the two different resolutions $R(y)$. The third plots shows the Local Resolution Estimate $L(y)$ and also shows a graphical description of the Resolution Bound. To be satisfied, the line must always be below $L(y)$ within $R(y)$ ($L(y)$ cannot intersect with the rectangle). In this way, the (*top*) red solution does not satisfy the Resolution Bound, although the blue solution does. However, this is not the largest possible $R(y)$ and hence, not optimal $R(y)$. We can see that the green solution (*bottom*) also satisfies the Resolution Bound, and does it optimally, as any increase in $R(y)$ would violate the Resolution Bound.

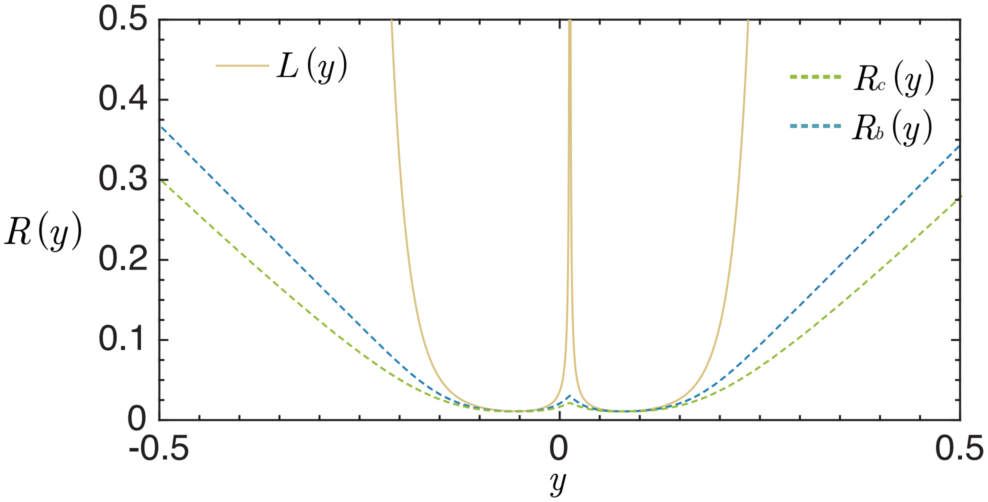


Figure 4.4: The Local Resolution Estimate, optimal continuous resolution function satisfying the Resolution Bound $R_c(y)$ and the continous optimal resolution function satisfying the Reconstruction Condition $R_b(y)$ for the 1D Guassian shown in Figure 4.1

for which $|f(y) - f(y^*)|$ is largest for all y^* within $R(y)$ of y. This is a result of the positivity constraint on the coefficients. Using this, a brute-force approach iterates over a sampling of the function $f\{\hat{x}\}$ and checks the values of $|f(y) - f(y^*)|$ increasing the neighbourhood of the search $(R(y))$ until it is larger than $E\sigma$. Similarly, for the continuous resolution bound, the brute-force approach iterates over $L\{\hat{x}\}$, increasing the neighborhood at each point until $R(y) \geq L(y^*)$. Unfortunately, both of these approaches are $O(N^2)$ worst case complexity and scale very poorly to even moderately sized problems (See A.2.1 for more discussions relating to continuous solutions). Further, how to find the optimal $\mathcal{P}$ for a general $R(y)$ beyond 1D is, to my knowledge, an unsolved problem. In 1D an approach similar to that in [38] can be used, however in practice $R\{\bar{x}\}$ would also have to be stored, meaning the representation would not be sparse.

4.2.2 Particle Cells and Implied Resolution Functions

The problem is formulated as the Resolution Bound because this allows it to be solved optimally using a linear in N number of operations from $f\{\bar{x}\}$ if we restrict the resolution function $R^*(y)$ to be composed of a particular set of square blocks. Such a function is shown by the bold black line in Figure 4.5. In this section, we describe these Resolution Functions and blocks and relate them to the Resolution Condition.

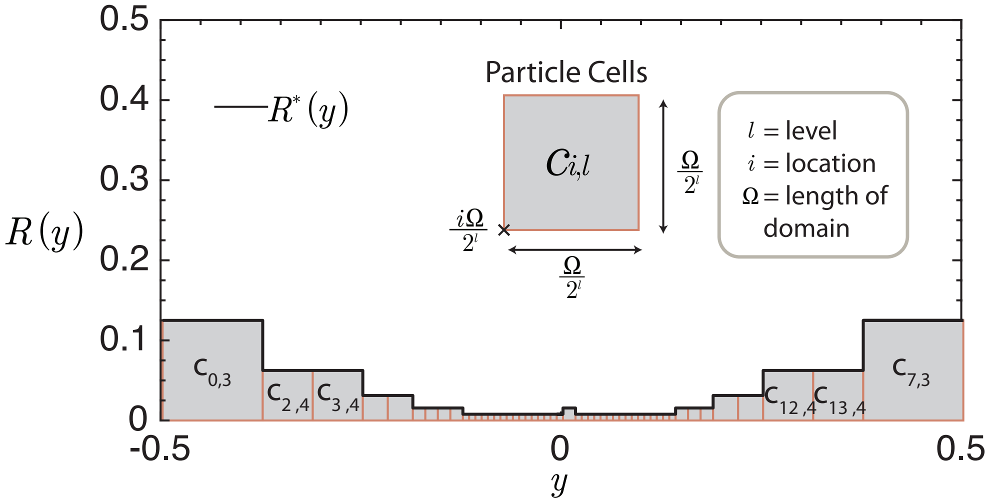


Figure 4.5: An Implied Resolution Function $R^*(y)$ (black line) constructed out of blocks called Particle Cells $c_{i,j}$. Particle Cells are defined by their location i and level l, as shown in the inset. Implied Resolution Functions require that there is only **one** Particle Cell placed in any location y (no overlap).

Implied Resolution Function

The resolution function composed out of these 'blocks', $R^*(y)$, we call the Implied Resolution Function, and the square blocks we call Particle Cells (see Figure 4.5). The Implied Resolution Function is the piecewise constant function described by the uppermost edge of the box, as shown in Figure 4.5. Particle cells have sides of length $\frac{\Omega}{2^l}$, where Ω is the size of the domain, and l is a positive integer we call the Particle Cell level. Hence, particle cell (block) $c_{i,l}$, is uniquely determined by its level l and location i. Figure 4.5 (*inset*) illustrates these definitions for a single Particle Cell. The size of the blocks, and hence levels, are set such that the lowest level is half the size of the domain Ω ($l_{min} = 1$), and the highest level l_{max} aligned with the dimension of the original pixels. For domains that are not powers of 2, this requires extending the domain Ω. If we define the following characteristic function

$$\phi(y, c_{i,l}) = \begin{cases} 1 & y \in [i2^{-l}\Omega, (i+1)2^{-l}\Omega) \\ 0 & otherwise \end{cases} \tag{4.17}$$

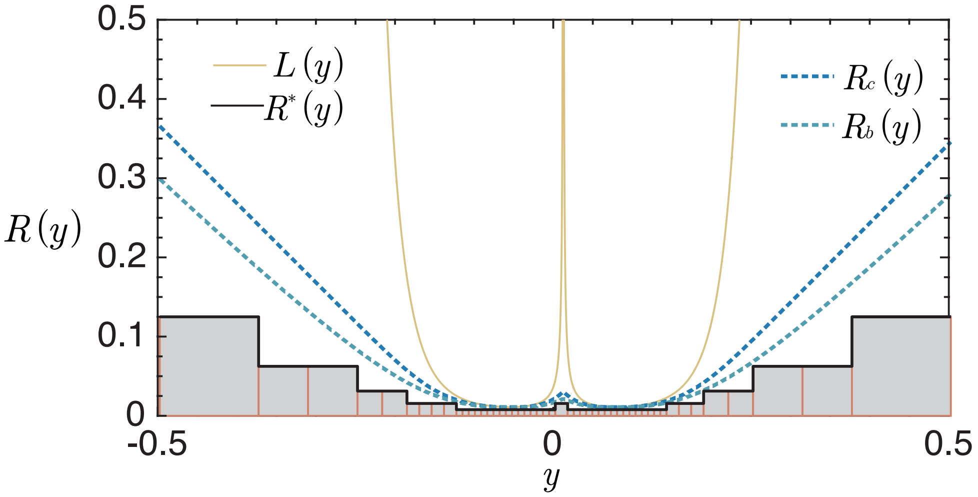


Figure 4.6: Comparison between the optimal Implied Resolution Function R^* and the optimal continuous solutions to the Resolution Bound R_c and Reconstruction Condition R_b.

then the Implied Resolution Function $R^*(y)$ for a set of Particle Cells $\mathcal{V}$ is

$$R^*(y) = \sum_{c_{i,l} \in \mathcal{V}} \phi(y, c_{i,l}) \frac{\Omega}{2^l}. \tag{4.18}$$

The problem of finding the optimal (largest everywhere) Implied Resolution Function, can be reduced to finding the smallest set of blocks $\mathcal{V}$ that defines a Resolution Function $R^*(y)$ that satisfies the Resolution Bound. We call such a set of Particle Cells $\mathcal{V}$ the Optimal Valid Particle Cell (OVPC) set. Given $R^*(y)$ also satisfies the Resolution Bound, then necessarily, $R^*(y) \leq R_c(y) \leq R_b(y)$ as shown in Figure 4.6 for the 1D Gaussian example. The Implied Resolution Function is only defined for Particle Cell sets $\mathcal{V}$ that form a spatial parition of Ω, thats is $\bigcup_{c_{i,l} \in \mathcal{V}} \text{supp}\{\phi(., c_{i,l})\} = \Omega$ and $\text{supp}\{\phi(., c_{i_1,l_1})\} \cap \text{supp}\{\phi(., c_{i_2,l_2})\} = \emptyset$ for all pairs of Particle Cells in $\mathcal{V}$ (i.e. $(i_1, l_1) \neq (i_2, l_2)$).

To construct an algorithm that finds the OVPC set for a given Local Resolution Estimate $L(y)$ requires the formulation of the Resolution Bound in terms of Particle Cells. This formulation first requires arranging the set $\mathcal{C}$ of all possible Particle Cells $c_{i,l}$ by level l and location i as a tree structure as shown in Figure 4.7. In 1D this is a binary tree, in 2D a quad-tree and 3D an oct-tree. This is the same structure as used in the Haar wavelet and pyramid image formulations [4]. When arranged as a tree structure, we can naturally define both

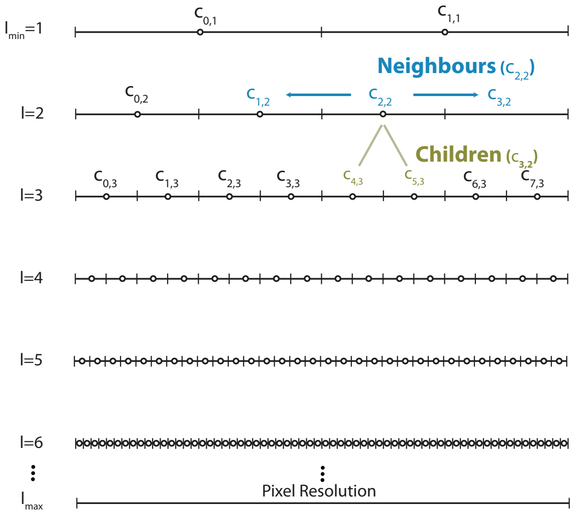


Figure 4.7: TThe dyadic decomposition of the Particle Cell Set $\mathcal{C}$ of $\Omega \times \Omega$ represented as a tree structure. The set of neighbors of a Particle Cell are shown in blue, and children in green.

children and neighbor relationships between Particle Cells, as shown in green and blue in the example. Also, we define the descendants of a Particle Cell, as the set of all children, and children's children up to the maximum level l_{max}. Given these definitions the Local Resolution Estimate can be represented as a set of Particle Cells $\mathcal{L}$ by iterating over each pixel y, and adding the Particle Cell with level $l = \lfloor \log_2 \frac{\Omega}{L(y)} \rfloor$ and location $i = \lfloor \frac{y2^l}{\Omega} \rfloor$ to $\mathcal{L}$ if it is not already in $\mathcal{L}$. Figure 4.8 shows a representation of how $\mathcal{L}$ relates to $L(y)$, with $\mathcal{L}$ also represented in the tree. We call this set of Particle Cells the Local Particle Cell (LPC) set $\mathcal{L}$. Formally,

$$\mathcal{L} = \{c_{i,j} \in \mathcal{C} | \exists (y^*, L(y^*)) : y^* \in (i\frac{\Omega}{2^l}, (i+1)\frac{\Omega}{2^l}], L(y^*) \in [\frac{\Omega}{2^l}, \frac{\Omega}{2^{l-1}})\} \tag{4.19}$$

defines the LPC. Where $L(y) \leq \frac{\Omega}{2^{l_{max}}}$ are set to $\frac{\Omega}{2^{l_{max}}} + \delta$ such that these areas with generate Particle Cells at l_{max}. In practice an alternative definition is used, effectively, taking Particle Cells one lower level than above, but gives the same R^* (see 4.2.3). But above, is the more natural definition, and coincides with taking a truncated lower bound on $L(y)$ at each point. Using this LPC set $\mathcal{L}$ we can now represent the Resolution Bound using Particle Cells.

Theorem 1. A set of Particle Cells $\mathcal{V}$ will define an Implied Resolution Function that satisfies the Resolution Bound for a given $L(y)$, iff, the following statement is true: *for every Particle Cell in $\mathcal{V}$ none of the Particle Cells descendants, or neighbours descendants, are in the LPC set $\mathcal{L}$*

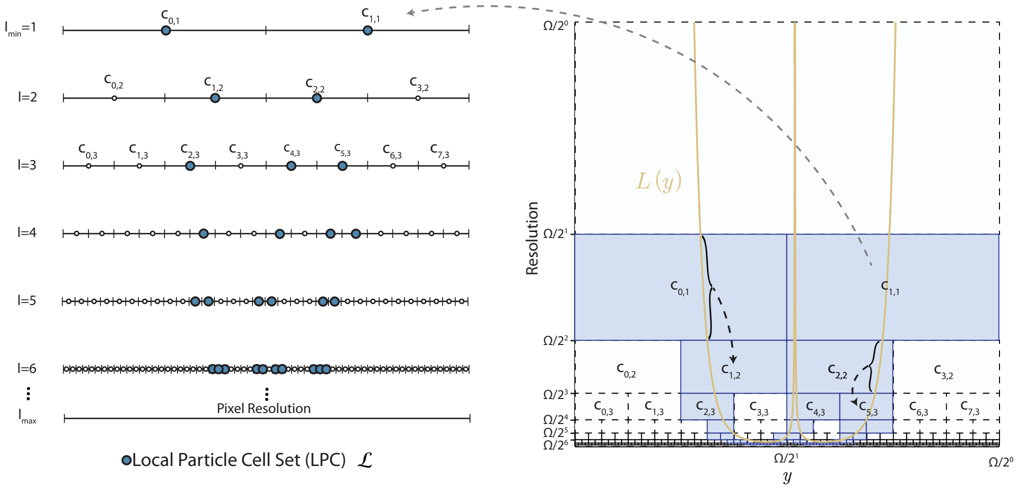


Figure 4.8: Schematic showing the formation of a Local Particle Cell (LPC) set $\mathcal{L}$ from the Local Resolution Estimate $L(y)$

See the following chapter Theorem 1 for the proof. We call any set of Particle Cells satisfying this statement *valid*. The Optimal Valid Particle Cell (OVPC) set $\mathcal{V}$, is then defined as the set for which the replacement of any group of Particle Cells with a larger Particle Cell would result in $\mathcal{V}$ no longer being valid (Theorem 2).From these definitions, the OVPC set can be found using a simple algorithm that explicitly checks the validity and optimality condition of potential particle configurations. The algorithm involves iterating from the lowest level l to higher levels, and replacing larger Particle Cells with smaller Particle Cells until the whole set is valid. However, such an approach, although linear in N, does not utilize the regularity and redundancy of Particle Cells, and as a worst case checks the validity and optimality of every Particle Cell.

4.2.3 Pulling Scheme

Here we present a novel algorithm for finding the OVPC set called the Pulling Scheme. The name comes from the way a single small Particle Cell in $\mathcal{L}$, pulls the Resolution Function down to smaller values (by forcing smaller Particle Cells) across the domain like a weight placed on a trampoline (See Figure 4.10 and Figure 4.11). The Pulling Scheme finds the OVPC set $\mathcal{V}$ directly, without explicitly checking validity or optimality. Three properties of OVPC sets are used to derive the algorithm. We list them below:

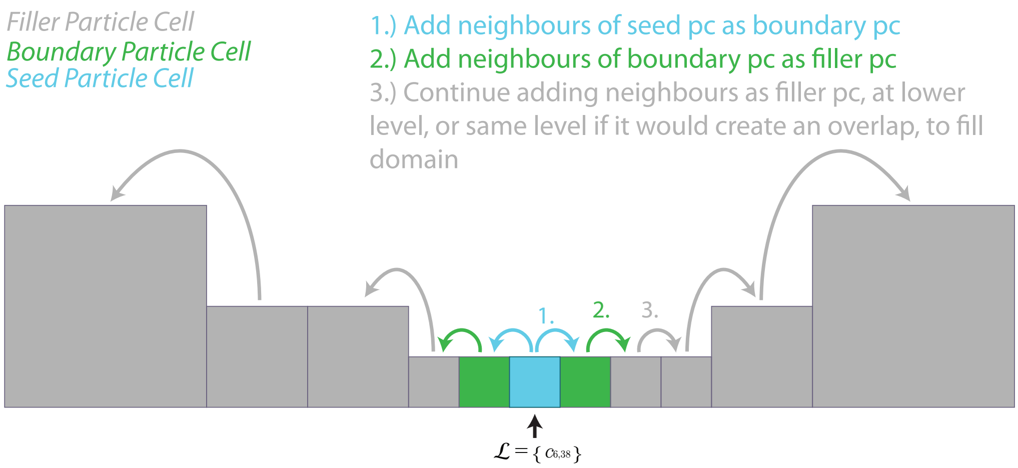

Filler Particle Cell
Boundary Particle Cell
Seed Particle Cell
1.) Add neighbours of seed pc as boundary pc
2.) Add neighbours of boundary pc as filler pc
3.) Continue adding neighbours as filler pc, at lower level, or same level if it would create an overlap, to fill domain
1.
2.
3.
$\mathcal{L} = \{ c_{6,38} \}$

Figure 4.9: Schematic how an OVPC set $\mathcal{V}$ can be generated when $\mathcal{L} = \{c_{i,l}\}$ has only one Particle Cell. We give Particle Cells an additional property called type, based on how the Particle Cell was added to the set $\mathcal{V}$. Particle cells that are in both $\mathcal{V}$ and $\mathcal{L}$ are of type *seed*. Particle cells that are neighbors to a seed cell are type *boundary*. All others, are of type *filler*. $\mathcal{V}$ is created by first adding $c_{i,l}$ (1.), and its neighbors (2.) on the same level and their neighbors (3.). The domain is then filled, allowing only one level change at once, and ensuring the resulting set forms a spatial partition.

1. OVPC sets have a predictable and self-similar structure (See Figure 4.9 and Figure 4.10, is best seen in 2D as in Figure 5.2). The predictability is reflected in neighboring Particle Cells never differing by more than by one level and having a common pattern of Particle Cells around the smallest Particle Cells in the set (highest level (resolution)). This local structure is independent of absolute level l and therefore adds the self-similar structure to the sets. Using these structural features, the OVPC set $\mathcal{V}$ for a LPC set $\mathcal{L}$ with only one Particle Cell $c_{i,l}$ can be generated directly for any i and l. This is done by giving each Particle Cell added to $\mathcal{V}$ an additional property called type $= \{seed, boundary, filler\}$. Figure 4.9 illustrates the process. The solution is generated starting by with $c_{i,l}$ and adding this Particle Cell to $\mathcal{V}$ with type *seed*. Next, the two neighbours of $c_{i,l}$ are added to $\mathcal{V}$ with type *boundary*. Then for these *boundary* Particle Cells their vacant neighbours are added with type *filler*. The last step fills the rest of the domain with Particle Cells of type *filler*. Adding adjacent Particle Cells increasing by at most one level, adding a Particle Cell at the same level if the higher level Particle Cell would cause an overlap between cells in $\mathcal{V}$.

2. OVPC sets are what we call *separable* (Lemma 1). This Lemma says that we can find the OVPC set given an LPC set $\mathcal{L}$, by solving for the optimal Particle Cell set of each Particle Cell in $\mathcal{L}$ separately and then

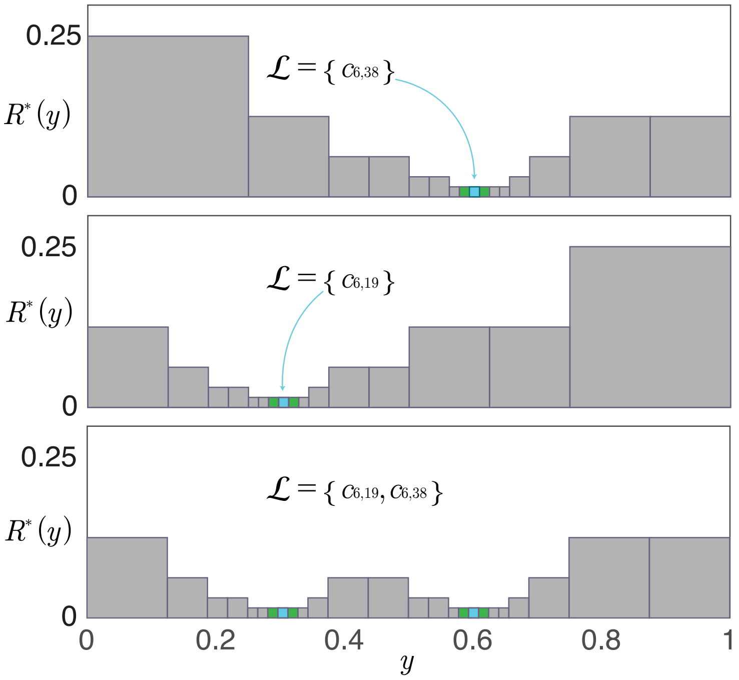


Figure 4.10: Illustrates three OVPC sets $\mathcal{V}$ for three different LPC sets $\mathcal{L}$. The colors of the Particle Cells indicate their type. The three together illustrate the **separable** property of OVPC sets, as the OVPC set of the two Particle Cells (*bottom*), can be formed by taking the smallest Particle Cell in each location of the top two OVPC sets.

taking the smallest Particle Cells from this set that covers the domain. Figure 4.10 illustrates this for two Particle Cells in $\mathcal{L}$ (See Figure 5.2 for illustration in 2D).

3. From Lemma 2 when constructing $\mathcal{V}$ we can ignore all Particle Cells in $\mathcal{L}$ that also have descendants in $\mathcal{L}$. We call this property *redundancy* and is a consequence of the descendant Particle Cell providing an equal or tighter constraint on the resolution function everywhere in the domain then its ascendant Particle Cell.

The Pulling Scheme uses all these properties to directly construct $\mathcal{V}$ by propagating solutions from individual Particle Cells in $\mathcal{L}$ using property 1, one level at a time starting from the highest level of the Particle Cells in $\mathcal{L}$. Figure 4.10 shows a schematic of two solutions being propagated from two Particle Cells. When two solutions meet at a Particle Cell, the precedence of one solution depends on the Particle Cells type where they meet. Precedence is ordered from seed>boundary>filler. This order represents the solution that

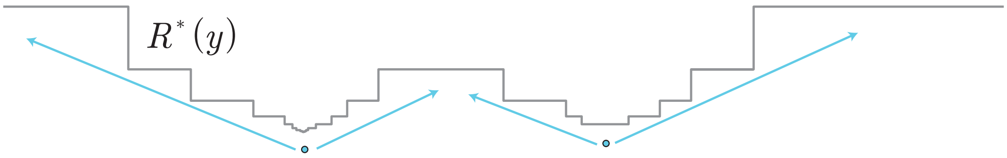


Figure 4.11: The basic idea of the **Pulling Scheme**. R^* is propagated outwards from higher levels to lower levels utilizing property 1 and property 2 of OVPC sets. When two solutions meet, only one needs to be propagated. Therefore, by propagating solutions from l_{max} to l_{min} and propagating the solution to higher levels using the filler type Particle Cells, the solution can be constructed directly, without checking the validity.

provides the 'tighter' constraint on the resolution function. Then only the solution with precedence needs to be propagated. The Pulling Scheme can be implemented in many different ways, and see the next chapter for a general description 5.4.5. Here, we describe an implementation that uses a data structure that explicitly stores $\mathcal{C}$ the full Particle Cell tree. This is the form of Pulling Scheme used for the benchmarking sections below, see A.4.1. However, other forms are possible that do not require the explicit storing of the tree structure (see 5.4.5 for a discussion). [4]

Algorithm 1 describes the four steps of the algorithm on each level iterating from the highest level l_{max} to lowest level l_{min}. Figure 4.12 gives a schematic representation for each of the steps. This requires extending the definition of type to also include, *ascendant*,*ascendant neighbour*, and *propogate*, which are used for Particle Cells that are not in $\mathcal{V}$ but used in the algorithm. The use of these allows the efficient propagation of the solutions with minimal operations.

Equivalence optimization

In practice, we use an optimization to the above scheme, which produces an identical solution but reduces the computational and memory cost by a factor of 2^d, where d is the dimension (Lemma 3). See 5.4.4 for the proofs extended to this case. The optimization involves utilizing the difference between the boundary and seed Particle Cells and filler Particle Cells and uses the exact pulling scheme as above. However, it uses a different $\mathcal{L}$. We additionally define the natural LPC $\mathcal{L}_n$ as

$$\mathcal{L}_n = \{c_{i,j} \in \mathcal{C} | \exists (y^*, L(y^*)) : y^* \in (i\frac{\Omega}{2^l}, (i+1)\frac{\Omega}{2^l}], L(y^*) \in [\frac{\Omega}{2^{l+1}}, \frac{\Omega}{2^l})\} \quad (4.20)$$

---

[4] The implementation of the algorithm described here, and its parallelization were developed in collaboration in Mattius Sasuik

note the change in the last interval from $[\frac{\Omega}{2^l}, \frac{\Omega}{2^{l-1}})$ to $[\frac{\Omega}{2^{l+1}}, \frac{\Omega}{2^l})$ and $\hat{l}_{max} = l_{max} - 1$. This coincides with a Particle Cell being in $\mathcal{L}_n$ if the function $L(y)$ passes through it, when represented as in Figure 4.8. If the pulling scheme is run on $\mathcal{L}_n$ and produces $\mathcal{V}_n$ then, all Particle Cells $c_{i,l}$ with type seed and boundary, are replaced by their children to form $\hat{\mathcal{V}}$, this is identical to the solution formed by the pulling scheme directly on $\mathcal{L}$ to produce $\mathcal{V}$. That is $\hat{\mathcal{V}} = \mathcal{V}$ (Lemma 3).

Computational and memory complexity

Here we address the computational and memory complexity of the Pulling Scheme using explicitly storage of $\mathcal{C}$ as described above. We will discuss the equivalence optimized version described in the previous section as this is used in practice. Although the above is given in 1D, all results hold in general dimension d, so we will give this analysis for general dimension d.

First we consider the size of $\mathcal{C}$, for a given problem with l_{min} and l_{max}, where $N = 2^{dl_{max}}$, then storing $\mathcal{C}$ requires a data-structure with

$$\begin{aligned} N_{\mathcal{C}} &= \sum_{l=l_{min}}^{l_{max}-1} 2^{dl} \\ &= \frac{2^{d(l_{max}+1)} - 2^{dl_{min}}}{(2^d - 1)} \end{aligned} \tag{4.21}$$

entries, because the highest level in the structures is $l_{max} - 1$. If we then consider the ratio of the size of the data-structure to the original data size N, we get

$$\frac{N_{\mathcal{C}}}{N} \leq \frac{1}{2^d - 1}\left(1 - \frac{1}{N}\right) \tag{4.22}$$

where we have set $l_{min} = 0$ as a worst case. Therefore, in the large N limit, we get $\frac{N_{\mathcal{C}}}{N} \approx \frac{1}{2^d - 1}$. Which gives us upper bounds of N in 1D, $\frac{N}{3}$ in 2D, and $\frac{N}{7}$ in 3D for the size of the required data-structure. Given there are only seven unique values that are needed for the algorithm, then each only requires 3 bits of information to be stored. Although this is not likely in practice, due to available data types, the Pulling Scheme requires at most $\frac{3N}{2^d - 1}$ bits in memory.

For the worst-case computational complexity, we can consider $\mathcal{L}^* = \mathcal{C}^*$, where $\mathcal{C}^*$ is $\mathcal{C}$ restricted to $l_{max} - 1$. That is every Particle Cell is in $\mathcal{L}^*$. Now each, step requires iteration over the data-structure given $\mathcal{O}(N_{\mathcal{C}})$ operations. All parent and neighbour operations scale with dimension d, and therefore for fixed d, have a fixed cost. Therefore, again we can get an upper bound on all steps taken across Algorithm 1 as $\mathcal{O}(N_{\mathcal{C}})$. Therefore, the whole algorithm is worst-case $\mathcal{O}(N_{\mathcal{C}})$ which is $\mathcal{O}(N)$.

In practice, the performance of this algorithm is more complicated depending on N, $\mathcal{L}$ and the spatial distribution of Particle Cells. From the steps above, we can see that the number neighbor searches at the highest resolution are the number of seed Particle Cells at that level. The most costly steps scale with the number of seed Particle Cells ($\#(\mathcal{L} \cap \mathcal{V})$). This is compared to the neighbor and filler Particle Cells that incur proportionally fewer operations. Hence tentatively we would expect the performance to scale as $\mathcal{O}(N + \#(\mathcal{L} \cap \mathcal{V}))$, with the different term dominating depending on situation. Further, the exact cost also would depend on the spatial distribution of $\mathcal{L}$. We benchmark the behavior of these scheme in Chapter 6 below.

Therefore, the above shows how the Pulling Scheme satisfies constructed $R^*(y)$ in worst case N steps as in Result 1. The last point missing is the formulation of $\mathcal{P}$ that we address in the next section.

4.2.4 Choosing the Particle Set $\mathcal{P}$

Given the Implied Resolution Function R^* (represented by $\mathcal{V}$) computed by the Pulling Scheme, the last step of forming the APR (and outlining Results 1-3) is determining the sampling of particles $\mathcal{P}$. We must sample particles such that at all pixel locations y there is at least one particle within a distance of $R(y)$ ($\#(x_p \in \mathcal{N}(y, R(y))) > 0$). We satisfy this by placing one particle at the center of each Particle Cell in $\mathcal{V}$. Specifically, for each Particle Cell $c_{i,l}$ in $\mathcal{V}$ we add a particle p to $\mathcal{P}$ with location

$$y_p = \frac{\Omega}{2^l}(i + 0.5) \tag{4.23}$$

where we ignore any constant offsets. Within a Particle Cell $c_{i,l}$ in $\mathcal{V}$ the Implied Resolution Function $R^*(y)$ is equal to the width of the Particle Cell. Therefore, a particle placed in the center is within $R^*(y)$ of all y within the Particle Cell, guaranteeing $\#(y \in \mathcal{N}(y, R(y))) > 0$. Given the particle locations y_p the last step is to store the intensity at the point $I_p = I(y_p)$ using an estimate from the original image. We store these sampled particle intensities in $\mathcal{P}^*$.

Although simple, such a sampling is also in a sense optimal for a given Implied Resolution Function. We define an optimal sampling of a given $R(y)$ as the sampling that satisfies

$$\#\mathcal{P} = \int_{\Omega} \frac{1}{R(y)} dy \tag{4.24}$$

```
Data: Particle Cell set $\mathcal{L}$
Result: Optimal Valid Particle Cell set $\mathcal{V}(\mathcal{L})$
Function pulling_scheme($\mathcal{L}$)
    Represent all possible Partile Cells $\mathcal{C}$ from $l_{max}$ to $l_{min}$ in
      multi-resolution pyramid mesh and set all Particle Cells to
      EMPTY;
    forall Particle Cells $c \in \mathcal{C}$ where $c \in \mathcal{L}$ do
        $p$.type = SEED
    end
    for $l_c = l_{max} : l_{min}$ do
        /* Fill neighbors (Step 1)                                  */
        forall neighbors $n$ of $c \in \mathcal{C}(l_c)$ where c.type is (SEED or
          PROPOGATE) do
            if n.type is EMPTY then
                $n$.type = BOUNDARY
        end
        /* Set Parents (Step 2)                                     */
        forall parents $p$ of $c \in \mathcal{C}(l_c)$ where c.type is (SEED,
          PROPOGATE, or ASCENDANT) do
            $p$.type = ASCENDANT
        end
        if $l_c > l_{min}$ then
            /* Set Ascendant Neighbors (Step 3)                     */
            forall neighbors $n$ of $c \in \mathcal{C}(l_c - 1)$ where c.type is
              ASCENDANT do
                if n.type is EMPTY then
                    $n$.type = ASCENDANT_NEIGHBOR
                if n.type is SEED then
                    $n$.type = PROPAGATE
            end
            /* Set Fillers (Step 4)                                 */
            forall children $d$ of $c \in \mathcal{C}(l_c - 1)$ where c.type is
              (ASCENDANT_NEIGH or PROPOGATE) do
                if (d.type is EMPTY then
                    $d$.type = FILLER
            end
    end
    return all type SEED, BOUNDARY and FILLER Particle Cells in $\mathcal{C}$
      as $\mathcal{V}$;
```

Algorithm 1: An example of a **Pulling Scheme** algorithm for generating a OVPC set $\mathcal{V}$ from local Particle Cell set $\mathcal{L}$ using a temporary pyramid mesh data structure

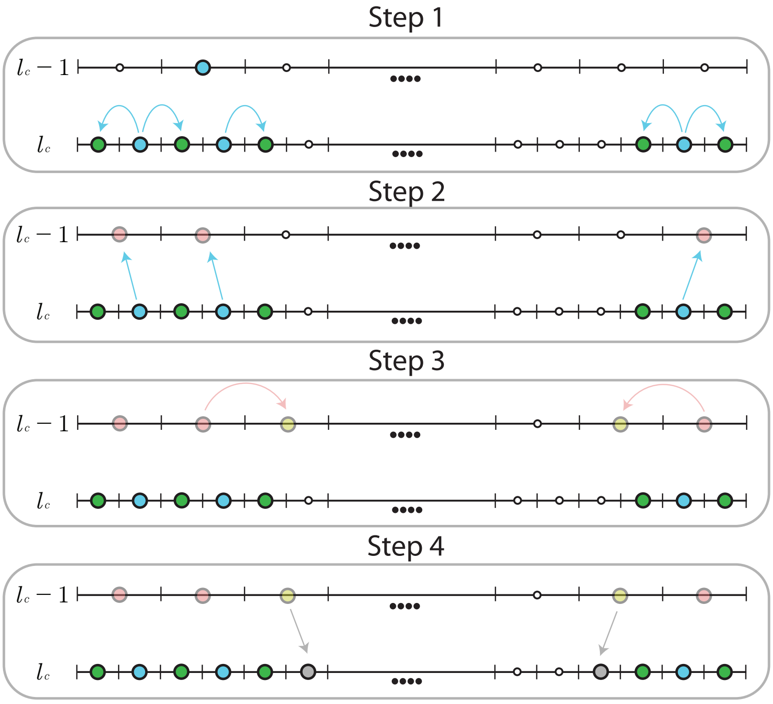


Figure 4.12: Schematic illustrating the four different steps in Algorithm 1 for the Pulling Scheme. The colour of the dots, identifies the type of Particle Cell. Blue dots represent *seed*, *boundary* in green, *filler* in grey, *ascendant* in red, *ascendant neighbour* in yellow. These four steps occur on each level from the highest level l_{max} to lowest l_{min}. Step 1, seed Particle Cells, or propogate, add neighbour cells as boundaries on level l_c. Step 2, seed and ascendant Particle Cells set their parents $(l_c - 1)$ to ascendant. Step 3, the ascendant particles on $l_c - 1$ set their vacant neighbours to ascendant neighbours. Step 4, Particle Cells of type ascendant neighbours and propogate on level $l_c - 1$ set empty children in l_c to filler.

and $\#(y \in \mathcal{N}(y, R(y))) > 0$ for all $y \in \Omega$. We have no proof for this being optimal. Indeed, for arbitrary $R(y)$ you can construct samplings that satisfy $\#(y \in \mathcal{N}(y, R(y))) > 0$ and not 4.24. However, intuitively, if we consider $\frac{1}{R(y)}$ as the point-wise required density, then ignoring edge effects, this means that satisfying 4.24 leads to this density being realized. Further, this integral is also satisfied for a constant regular sampling such as pixels.

If we now consider, the integral 4.24 for the Implied Resolution Function $R^*(y)$,

$$\begin{aligned}\int_\Omega \frac{1}{R^*(y)} dy &= \int_\Omega \frac{1}{\sum_{c_{i,l} \in \mathcal{V}} \phi(y, c_{i,l}) \frac{\Omega}{2^l}} dy \\ &= \sum_{c_{i,l} \in \mathcal{V}} 1 \\ &= \#\mathcal{V} = \#\mathcal{P} \end{aligned} \tag{4.25}$$

as required, and therefore $\mathcal{P}^*$ is optimal in the sense of 4.24. Hence, the Pulling Scheme in addition to providing an optimal Implied Resolution Function also provides an inherent 'optimal' sampling.

4.2.5 Forming the APR=$\{\mathcal{V}, \mathcal{P}^*\}$

The combination of the Particle Cell set $\mathcal{V}$ and $\mathcal{P}^*$ fully define the APR. Where $\mathcal{V} = \{\{c_{i_p,l_p}\}_{p=1}^{N_p} | (i_p, l_p) \in \mathbb{Z}\}$, and requires storing the integer i_p and l_p, and $\mathcal{P}^* = \{f_i = f(y_i)\}_{i=1}^{N_p}$. The Implied Resolution Function $R^*(y)$ and the particle locations y_p do not need to be directly stored as they are both directly calculatable from $\mathcal{V}$. Alternatively, $\mathcal{V}_N$ can be stored instead of $\mathcal{V}$, however this requires also the additional storage of the type of each Particle Cell (seed,boundary, or filler).

4.2.6 Summary

Therefore, the above concludes the presentation of the Results 1-3 presented at the beginning of the chapter. We focused on the 1D case for ease of exposition. The same results with the formal proofs are given in a more general formulation in the next chapter.

4.3 Practical considerations

In the above, we have ignored particle considerations of, how do we estimate $\frac{\partial f}{\partial y}$ and the impact of noise. Here we will briefly discuss these issues, including a discussion on the continuous resolution functions. Again, the results here are presented in 1D but apply to the general dimension case.

4.3.1 Discrete sampling

First, we consider the what the ideal sampling of $\frac{\partial f}{\partial y}\{\bar{x}\}$ would be that would allow reconstruction of all $y \in \Omega$ then the samples

$$\frac{\partial f}{\partial y}\{x_i\} = \max_{x \in [x_i - h/2, x_i + h/2)} \frac{\partial f}{\partial y}(x) \tag{4.26}$$

where h is the sampling distance between points for $\bar{x}$. These estimates would guarantee the APR reconstructs the function $y \in \Omega$, and not just at sample locations. This follows from the fact that this would produce an upper bound, on the true derivative across every interval.

4.3.2 Impact of noisy Local Resolution Estimate $L(\mathbf{y})$

However, even in noise-free situations, we do not have the ability to sample the derivative directly. Instead, we observe $|\widehat{\frac{\partial f}{\partial x}}| = |\frac{\partial f}{\partial x}| + \epsilon$. Therefore, it is interesting how errors from the estimation of the derivative translate into the violation of the Reconstruction Condition for a given relative error bound E.

Therefore, we consider how an error in $L(y)$ translates into the error in the solution compared to the user-set relative error bound E. That is we assume that instead of $L(y)$ we observe,

$$L^*(y) = \frac{E\sigma(y)}{|\nabla f|(1-\alpha)} \tag{4.27}$$

where α represents the maximum relative error in $|\frac{\partial f}{\partial x}|$ (We assume here the $0 \geq \alpha < 1$). We need only consider reductions of the magnitude of the gradient as increases will not impact the Reconstruction Condition (they simply increase the resolution wastefully). So then if we consider what the worst-case observed E^* is relative to the desired E for a given α (See A.3 for derivation) we get

$$\frac{E^* - E}{E} = \frac{1}{1-\alpha} - 1 \tag{4.28}$$

where the error is taken to occur at a local maximum of the derivative such that the error has an impact on the solution. Interpreting this, we can see that if $\alpha = 0.1$, i.e. ten percent absolute error in the gradient, then the ratio $\frac{E^* - E}{E} = 0.111$, and so if $E = 0.1$ then the observed relative error worst case would be 0.111. So a ten percent error has been related to an eleven percent increase in the realizable relative error.

This bound is insightful, as it tells us that if we have a given α and want to guarantee some realized $\hat{E}$, we can increase the user set E, to retain the bound despite the error (*i.e.* such that $E^* = \hat{E}$). However, this is at the cost of a higher number of particles. Alternatively, we can re-arrange the bound, as

$$\alpha = 1 - \frac{1}{1 + \frac{(E^* - E)}{E}}. \tag{4.29}$$

which then tells us how large a relative error in our derivative we can tolerate if we wish to have a set accuracy for the relative error bound.

The analysis above is based on relative errors. How do we then consider absolute errors ϵ? For a given ϵ, the relative error will be greatest with the derivative is small. Interestingly, these are the regions of our solution where it is likely $R^*(y) \leq L(y)$. That is, the large relative error will not impact the solution. The $y \in \Omega$ that are most likely to contribute to $\mathcal{V}$ will then also have a 'relatively' smaller α.

4.3.3 Impact of noisy particles $\hat{f}(y_p)$

Now if we consider that we are only able to sample noisy estimates of the function for our particles, $\hat{f}_p = f(\mathbf{x}_p) + \eta(\mathbf{x}_p)$, where η is some noise process. If we assume that we are still able to estimate $L(\mathbf{y})$ such that R^* is the true optimal solution we will find our observed error is

$$E^* = \frac{|f(\mathbf{y}) - \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} (f(\mathbf{x}_p) + \eta(\mathbf{x}_p)) \xi_p(y)|}{\sigma(\mathbf{y})} \tag{4.30}$$

then given $L(\mathbf{y})$ is the noise-free solution then the Reconstruction Condition holds and we get

$$E^* = |A + \frac{1}{\sigma(\mathbf{y})} \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} (\mathbf{x}_p) \xi_p(y)| \tag{4.31}$$

where we have $|A| \leq E$. Therefore as $E \to 0$, $A \to 0$ and therefore the infinity norm of the observed relative error $|E^*|_\infty$ will tend to the maximum $|\frac{1}{\sigma(\mathbf{y})} \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \eta(\mathbf{x}_p) \xi_p(y)|$ across the domain.

Therefore, the noisy input data, provides an upper bound of the observed relative error E^* regardless of adaptation, and user set E. Note, we provide a more detailed analysis of the impact of noise later in 8.3 showing that the

APR converges at an optimal rate to a bias estimate of the noise-free APR given a non-trivial Resolution Function that satisfies the Resolution Bound. This is consistent with the simple analysis here.

4.3.4 Comparison with continuous resolution functions

Although we can solve our restricted problem optimally using Particle Cells, this has been at the cost of fulfillment of the tighter Resolution Bound and not allowing continuous adaptation. We provide some brief worst-case bounds and numerical results in A.2.1. Unfortunately, for arbitrary functions, I know of no bound between the optimal continuous Resolution Function for the Reconstruction Condition $R_b(y)$, and the Resolution Bound $R_c(y)$ (Note, the c was intended to be interpreted as for continuous). However, the ratio of the two should converge to one as $E \rightarrow 0$ for an infinitely differentiable function. We can, however, find worst-case bounds between the Implied Resolution Function R^* and R_c. These are insightful, as they indicate the upper bound on the 'cost' of restricting ourselves to Resolution Functions constructed using Particle Cells. By considering the largest possible $L(y)$ on a per Particle Cell basis, we find that $\frac{R^*(\mathbf{y})}{R_c(\mathbf{y})} \leq 4\sqrt{d}$. Which corresponds to a worst case pointwise ratio of 4, ≈ 5.65 and ≈ 6.93 in 1D, 2D and 3D respectively.

Also, we could use the same method to provide an upper bound on the ratio $\frac{\#\mathcal{P}}{\#\mathcal{P}_c}$, where $\mathcal{P}_c$ corresponds to the evaluation of the sampling integral (4.24 in 1D, or see 5.75 for general dimension d) with R_c. We find the following upper bounds on the ratios of 3.47, 24.3 and 221.25 in 1D, 2D, and 3D. We note that even given R_c I know of no means of sampling in general dimension to achieve $\mathcal{P}_c$, and that the bounds are likely not to be tight for practical $L(\mathbf{y})$. Further, use of a continuous solution numerical solution, without analytical form, would require storage of N values, and essentially eliminate any reductions in the size of the representation through $\mathcal{P}$. However, there may be situations where this is useful.

To test these bounds, we numerically computed optimal continuous solutions. This can be done using the $\mathcal{O}(N^2)$, 'brute force' approach. However, unfortunately, the high computational cost limited the investigations. With the solution taking several orders of magnitude longer than the Pulling Scheme.

First, we found that the bound between R_c and R_b in 1D was close to one for reasonable ranges of $E \lll .3$, and tends to one as $E \rightarrow 0$. In 3D using the later discussed implementation, we found the observed mean ratio of the implied and continuous resolution functions in 3D was between 2 and 3, depending on the image content and level of noise. Further, we find that the worst-case bounds for particle ratio $\frac{\#\mathcal{P}}{\#\mathcal{P}_c}$ do not appear to be tight in practice, finding ratios of less than 11 in 3D benchmark examples (compared to the worst-case bound of 225.21).

4.3.5 Particle Cells and smoothness of the Local Intensity Scale

In practice, the use of an Implied Resolution Function relaxes the smoothness assumption on the Local Intensity Scale σ (4.2.1). For a continuous Resolution Function an equality for the expression

$$\sigma(y) \min_{x \in \mathcal{N}(y,R(y))} (g(x)) \approx \min_{x \in \mathcal{N}(y,R(y))} (\sigma(y) g(x)) \tag{4.32}$$

would require a constant $\sigma(y)$ for general f. However, we note when using an Implied Resolution Function as in the APR we only will detect changes that would change the Particle Cell level l. Now for a given problem the Particle Cell level can be calculated as,

$$l = \lceil \log \left(\frac{|\Omega| \sigma(x)}{g(x)} \right) \rceil \tag{4.33}$$

with a -1 if the equivalence optimization is being used. We find that 4.32 is then equivalent to

$$\lceil \log \left(\frac{|\Omega|}{\sigma(x) g(x)} \right) \rceil = \lceil \log \left(\frac{|\Omega|}{\sigma(y) g(x)} \right) \rceil \tag{4.34}$$

for all $x \in \mathcal{N}(y, R(y))$ and $y \in \Omega$. This is a weaker bound then 4.32 potentially allowing non-constant $\sigma(y)$. Hence, the situation is not quite as restrictive as 4.32 implies when using an Implied Resolution Function. Further exploration of this bound in combination with the results on the noisy $L(y)$ (A.3) would seem a fruitful future research direction.

4.3.6 Reconstruction of higher order derivatives

Later in Chapter 9 we discuss a generalized form of the APR that can be designed to also guarantee bounds on arbitrary function derivatives. However, can we, in general, provide any bounds on the reconstruction of the derivatives of the function for the APR by only guaranteeing the Reconstruction Condition? Here we shall quickly consider the gradient in 1D. However, the principle extends to higher dimensions and derivatives.

Lets consider we have constructed the APR for a function f. Now we consider,

$$\epsilon_1 = |\frac{\partial f}{\partial y}(y) - \frac{\partial f}{\partial y}\hat{}(y)| \tag{4.35}$$

the point wide reconstruction error of the derivative, where $\frac{\partial f}{\partial y}\hat{}(y)$ is a first order in h estimate of the gradient (satisfying conditions of a DC-PSE operator [115]). Following a similar approach to above we get (see later in 9.14 for derivation of a similar bound from which this can be inferred)

$$\epsilon_1 \leq AR(y) \max_{x^* \in \mathcal{N}(y,R(y))} \left(\frac{\partial^2 f}{\partial y^2}(x^*) \right) \tag{4.36}$$

where A is a constant based on the choice of particles and operator. Now given the Resolution Bound holds, it can be shown also that

$$\epsilon_1 \leq A \left(\frac{E\sigma}{\max_{x^* \in \mathcal{N}(y,R(y))} \left(\frac{\partial f}{\partial y}(x^*) \right)} \right) \max_{x^* \in \mathcal{N}(y,R(y))} \left(\frac{\partial^2 f}{\partial y^2}(x^*) \right). \tag{4.37}$$

where we assume $R(y)$ is bounded. If we assume an Implied Resolution Function, this is trivially satisfied. Hence we see that as $E \to 0$, the error should go to zero at a rate proportional to E. However, we have no guarantee on the absolute value as in the Reconstruction Condition. Given restrictions to a function with appropriate Lipschitz constants, one could achieve a global bound. However, this would unlikely to be of little practical use.

The same principle holds for higher derivatives and dimensions. So in conclusion, the APR satisfying the Reconstruction Condition, does unfortunately not also bound the error of the derivatives of the function.

4.4 APR transform steps summary

Therefore, here we have described the APR that resamples an image by adapting a set of Particle Cells $\mathcal{V}$ (representing the Implied Resolution Function $R^*(y)$) and set of particles $\mathcal{P}$ to the content of a function. In Algorithm 2, we outline the main steps for taking a sampled input function $f\{\bar{x}\}$ and forming the APR. These steps are the same for arbitrary dimension; the only main change is the definition of the Local Resolution Estimate $L(y)$. Which in general dimension d, for $f : \mathbb{R}^d \to \mathbb{R}$ becomes

$$L(\mathbf{y}) = \frac{E\sigma(\mathbf{y})}{|\nabla f(\mathbf{y})|} \tag{4.38}$$

where ∇f represents the gradient of the function, and $\mathbf{y} \in \Omega \subset \mathbb{R}^d$. Also, to the extension of the concepts for Particle Cells to higher dimensions. An important observation is that the steps requiring interaction across all pixels are those in step 2. Except step 5, that requires some estimation at particle locations. All other steps require only operations on Particle Cells which have lower memory costs than the original image.

For a detailed description on how all these steps were implimented for the emperical results shown in Chapter 6 and Chapter 7 see A.4.

```
Data: Sampled input function $f\{\bar{x}\}$
Result: APR = $\{\mathcal{V}, \mathcal{P}^*\}$
1.) Set relative error bound $E$;
2.) Compute discrete estimate of the gradient magnitude $|\nabla f|$ and $\sigma$ at
    locations $\bar{x}$;
3.) Compute $L(y)$ and create Local Particle Cell set $\mathcal{L}$;
4.) Compute Optimal Valid Particle Cell set $\mathcal{V}$ from $\mathcal{L}$ using the Puling
    Scheme;
5.) Sample particle intensities $\mathcal{P}^*$ from image at center of Particle Cells
    in $\mathcal{V}$
```

Algorithm 2: Summary of the steps of computing the APR for a given function f, sampled homogenously at $\bar{x}$.

4.5 Summary and main points

In the above chapter, we have introduced the Adaptive Particle Representation (APR) and the three theoretical and algorithmic results at its core. Following, we described the ideas and methods that are introduced to produce them, and a brief motivation on their form relating to the representation criteria presented in Chapter 3. This has attempted to be done in a didactic style, introducing the new ideas, concepts, and principles, using a 1D formulation and schematics for ease of explanation. The following Chapter 5, provides a more technical description of the general dimension case. It is intended, that these two chapters complement each other, and are to be used in unison. Next, I briefly summarize these concepts discussed in this chapter involved in forming the APR.

The APR involves the representation of a function, originally given as N sampled points $f\{\bar{x}\}$, in a lossy manner, with a user-defined relative error E to be met point-wise across the domain relative to a Local Intensity Scale σ. The representation involves a collection of sampled points $\mathcal{P}$ and a Resolution

Function $R(y)$. The reconstruction constraint is named the Reconstruction Condition. The function reconstruction is controlled everywhere by the Resolution Function $R(y)$ that dictates the size of the isotropic neighborhood of points, from the sampling $\mathcal{P}$, which are permitted to be used for function reconstruction. However, when constructing the APR, we do not directly satisfy the Reconstruction Condition; instead, we satisfy the Resolution Bound. The Resolution Bound places a direct restriction on the Resolution Function, based on the magnitude of the derivative and given restrictions on the local intensity scale, guarantees the fulfillment of the Reconstruction Condition.

In this chapter, we showed that if the Resolution Function is restricted to a special class of piecewise constant functions, called Implied Resolution Functions, we provide an algorithm for finding an optimal Implied Resolution Function $R^*(y)$, and associated particle sampling $\mathcal{P}$. This algorithm is called the Pulling Scheme and has linear complexity in the original number of sampled point N. The algorithm relies on the concept of Particle Cells, blocks that use tree-structures common to multi-resolution and adaptive methods. Instead of explicitly storing the Implied Resolution Function and particle locations y_p, the APR is fully described by $\{\mathcal{V}, \mathcal{P}\}$, where $\mathcal{V}$ is a set of Particle Cells, and $\mathcal{P}^*$ is a set of the function intensity values sampled at collocation points of the function.

Following this, we briefly touched on practical considerations of how to evaluate the derivative and the impact of noise on the formation of the APR and satisfaction of the Reconstruction Condition. Further, we also discussed the cost of restricting the resolution function to an Implied Resolution Function. Lastly, we briefly outlined the practical steps required for the formation of the APR, from a sampled input function.

Summary of the chapter

- Introduced the Adaptive Particle Representation (APR), using 1D, and the main theoretical and algorithmic results of this thesis.
- Derived the Resolution Bound and Reconstruction Conditions in 1D
- Introduced Particle Cells and how they can be used to construct an Implied Resolution Function
- Outlined the Pulling Scheme, that effeciently produces the optimal Particle Cell set $\mathcal{V}$ and particle sampling $\mathcal{P}^*$ that form the the APR.

- Discussed practical issues arising from discrete sampling and noise piece-wise constant resolution function
- Outlined the main steps in practice for transforming a given $f\{\bar{x}\}$ into the APR $\{\mathcal{V}, \mathcal{P}^*\}$.

5 General Dimension APR and Technical Details

Contents

5.1 General dimension APR 82
  5.1.1 Result 1 83
  5.1.2 Result 2 83
  5.1.3 Result 3 83
5.2 Reconstruction Condition and Resolution Bound 83
5.3 Particle Cells and the Implied Resolution Function 85
  5.3.1 Particle Cell definitions 85
  5.3.2 Implied Resolution Function 88
  5.3.3 Local Particle Cell set 89
  5.3.4 Optimal Valid Particle Cell sets 90
5.4 Pulling Scheme 92
  5.4.1 Self-similarity and production of individual solutions 92
  5.4.2 Separability 94
  5.4.3 Redundancy of Particle Cells 96
  5.4.4 Equivalence 96
  5.4.5 Algorithm description 100
  5.4.6 Generic algorithm 100
  5.4.7 Memory and computational complexity 101
5.5 Particle sampling 104
  5.5.1 APR as $\{\mathcal{V}_n, \mathcal{P}^*\}$ 105
  5.5.2 Optimality 105
  5.5.3 Integral neighborhood optimization 106
5.6 Technical additions 108
  5.6.1 Function reconstruction and interpolation 109
  5.6.2 APR particle graph 110
5.7 Summary and main points 111

This chapter extends the main results for the Adaptive Particle Representation (APR) presented in the previous chapter to general dimension. Here, we also include technical details and proofs. For a more didactic introduction to the ideas and concepts used in forming the APR, the reader is directed first to the previous chapter. It is intended for this current chapter to be used in *addition* to the previous, and therefore, some repeated explanations or definitions have been omitted.

This chapter is structured as follows. First, we restate the definition of the APR and the main results from the previous section in a general dimension setting. We then derive and discuss, the Resolution Condition, Particle Cells and Implied Resolution Function, and the Pulling Scheme in a general dimension setting. Definitions are given in a more general way than the previous, to allow consideration of extensions beyond those discussed here. Lastly, we introduce possible reconstruction functions and the APR particle graph.

5.1 General dimension APR

Here we re-state the formulation of the APR in a general dimension d setting.

As in the 1D setting the Adaptive Particle Representation takes a regularly sampled input function, such as pixel images, and resamples it as a set of particles $\mathcal{P}$ and a resolution function $R(\mathbf{y})$. Where particles p are now collocation points in d dimensions, $\mathbf{x}_p$ and carry function values $f_p = f(\mathbf{x}_p)$. The resolution function $R : \Omega \to \mathbb{R}$ defines a local isotropic neighborhood $\mathcal{N}$ at each point in the domain $\Omega \subset \mathbb{R}^d$.

We consider a once differentiable function $f : \Omega \to \mathbb{R}$, that is sampled on a grid with fixed spacing as $f\{\bar{\mathbf{x}}\}$ where the number of samples is $\#\bar{\mathbf{x}} = N$. We represent the function for a given $\mathcal{P}$ and $R(\mathbf{y})$ in the following way

$$\hat{f}(\mathbf{y}) = \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} f(\mathbf{x}_p)\xi_p(\mathbf{y}) \tag{5.1}$$

where $\mathcal{N}(\mathbf{y}, R(\mathbf{y})) = \{\mathbf{x} \in \Omega : |\mathbf{x} - \mathbf{y}| \leq R(\mathbf{y})\}$ and $\xi_p(\mathbf{y}) = \xi(\mathbf{y}, \mathbf{x}_p)$ are constants that satisfy $\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \xi_p(y) = 1$ with $\xi_p(\mathbf{y}) \geq 0$. and assuming $\#(\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))) > 0$ for all $\mathbf{y} \in \Omega$. The reconstruction follows the Reconstruction Condition,

$$\|\frac{f - \hat{f}}{\sigma}\|_\infty \leq E \tag{5.2}$$

where $\sigma : \Omega \to \mathbb{R}$, and satisfies a smoothness assumption 5.11. The Resolution

Bound is then

$$R(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} L(\mathbf{x}). \tag{5.3}$$

We quickly re-state the results of the previous sections, providing the slight adjustments as required.

5.1.1 Result 1

The Pulling Scheme can find the optimal $R^*(\mathbf{y})$ and particle set $\mathcal{P}$ that satisfy problems in the form of the Resolution Bound (5.3) for general $L(\mathbf{y})$. Where the optimal implied resolution function is the R^* that satisfies

$$\arg\max_{R^* \in \mathcal{R}^*} \int_{\Omega} R^*(\mathbf{y}) d\mathbf{y} \tag{5.4}$$

where $\mathcal{R}^*$ is the set of all Implied Resolution Functions $R^*(\mathbf{y})$ that satisfy the Resolution Bound 5.3. The optimal $\mathcal{P}^*$ is then the particle set that satisfies, $\#(\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))) > 0$, and $\#\mathcal{P} = \int_\Omega \frac{1}{R(y)^d} d\mathbf{y}$.

5.1.2 Result 2

Given the local resolution function $\sigma(\mathbf{y})$ is sufficiently slowly varying (see 5.11), and then $L(\mathbf{y}) = \frac{E\sigma(\mathbf{y})}{|\nabla f|}$ where ∇f represents the gradient of the function then the reconstructions formed using $R^*(\mathbf{y})$ and $\mathcal{P}$ will satisfy the Reconstruction Condition 5.2.

5.1.3 Result 3

The Implied Resolution Function $R^*(\mathbf{y})$ and particle cell set $\mathcal{P}$, can be completely described by a set of particle cells $\mathcal{V} = \{\{c_{\mathbf{i}_p, l_p}\}_{p=1}^{N_p} | \mathbf{i}_p \in \mathbb{Z}^d, l_p \in \mathbb{Z}\}$ (i.e. it can be define by N_p length sets of integers $d+1$ integers) and $\mathcal{P}^* = \{\{f_p\}_{p=1}^{N_p}\}$. The combination of these two sets $\{\mathcal{V}, \mathcal{P}^*\}$ we call the Adaptive Particle Representation (APR).

5.2 Reconstruction Condition and Resolution Bound

Here we present derivation of the Resolution Bound, it differs little from the one-dimensional case. We begin with the Reconstruction Condition state point wise as

$$|f(\mathbf{y}) - \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} f_p \xi_p(\mathbf{y})| \leq \frac{E}{\sigma(\mathbf{y})} \tag{5.5}$$

which holds must hold for all $\mathbf{y} \in \Omega$. Therefore again we proceed by considering the exact formulation of the error as

$$\epsilon(\mathbf{y}) = f(\mathbf{y}) - \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} f_p \xi_p(\mathbf{y}) \tag{5.6}$$

now if we again assume function is C^1 can express by taking Taylor series expansions of f_p centered at $\mathbf{y}$ and using the integral form of the remainder for the Taylor series [103]

$$\epsilon(\mathbf{y}) = \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \sum_{|\mathbf{k}|=1} (\mathbf{y} - \mathbf{x}_p)^{\mathbf{k}} \int_0^1 \frac{\partial}{\partial \mathbf{x}^{\mathbf{k}}} f(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y})) ds \xi_p(\mathbf{y}) \tag{5.7}$$

where $\mathbf{k}$ is using multi-index notation (See [103] for a brief description). In this case, it simply denotes summing over each spatial direction. Which we note is equivalent to the fundamental theorem of calculus and can be written as a path integral, and again using the triangle inequality

$$|\epsilon(\mathbf{y})| \leq \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} |\xi_p(\mathbf{y})||(\mathbf{y} - \mathbf{x}_p)| \int_0^1 |\nabla f(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y}))| ds \tag{5.8}$$

where $\nabla f(\mathbf{x})$, represents the gradient operator. Now again given that $|\mathbf{y} - \mathbf{x}_p| \leq R(\mathbf{y})$ then

$$|\epsilon(\mathbf{y})| \leq \left(\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} |\xi_p(\mathbf{y})| \right) R(\mathbf{y}) \max_{\mathbf{x} \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} (|\nabla f(\mathbf{y})|) \tag{5.9}$$

and so then again given we assume $\xi_p(\mathbf{y}) > 0$ then using this bound, 5.5 will hold if

$$R(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \left(\frac{E\sigma(\mathbf{y})}{|\nabla f(\mathbf{x})|} \right), \tag{5.10}$$

which then assuming sufficient smoothness of $\sigma(\mathbf{y})$, such that the approximation

$$\max_{\mathbf{x} \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \left(\frac{|\nabla f(\mathbf{x})|}{\sigma(\mathbf{y})} \right) = \max_{\mathbf{x} \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \left(\frac{|\nabla f(\mathbf{x})|}{\sigma(\mathbf{x})} \right) \tag{5.11}$$

holds then

$$R(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \left(\frac{E\sigma(\mathbf{x})}{|\nabla f(\mathbf{x})|} \right). \tag{5.12}$$

which is of the required form

$$R(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \left(L(\mathbf{x}) \right). \tag{5.13}$$

where the local resolution estimate is $L(\mathbf{y}) = \frac{E\sigma(\mathbf{y})}{|\nabla f(\mathbf{y})|}$.

5.3 Particle Cells and the Implied Resolution Function

In this section, we introduce the general dimension treatment of Particle Cells. We begin with several definitions that will be useful.

5.3.1 Particle Cell definitions

For our given domain $\Omega \subset \mathbb{R}^n$, with maximum side length Ω_0. We begin by extending the Ω to a square domain $\Omega^* \in \mathbb{R}^d$, with edge length Ω_0, such that $\Omega \subseteq \Omega^*$.

Next we introduce Particle Cells $\mathcal{C}$ that form a partition of the extended spatial domain Ω^* and $\mathcal{R}$ the range of the possible resolution functions, $\mathcal{R} = \{R : \Omega \to \mathbb{R}^+\}$, which we call the resolution domain. Formally we enumerate the set $\mathcal{C}$, as

$$\mathcal{C} = \{c_{\mathbf{i},l}, \forall (\mathbf{i}, l) : l \in \mathbb{N}, i_k = 0, .., 2^l - 1\} \tag{5.14}$$

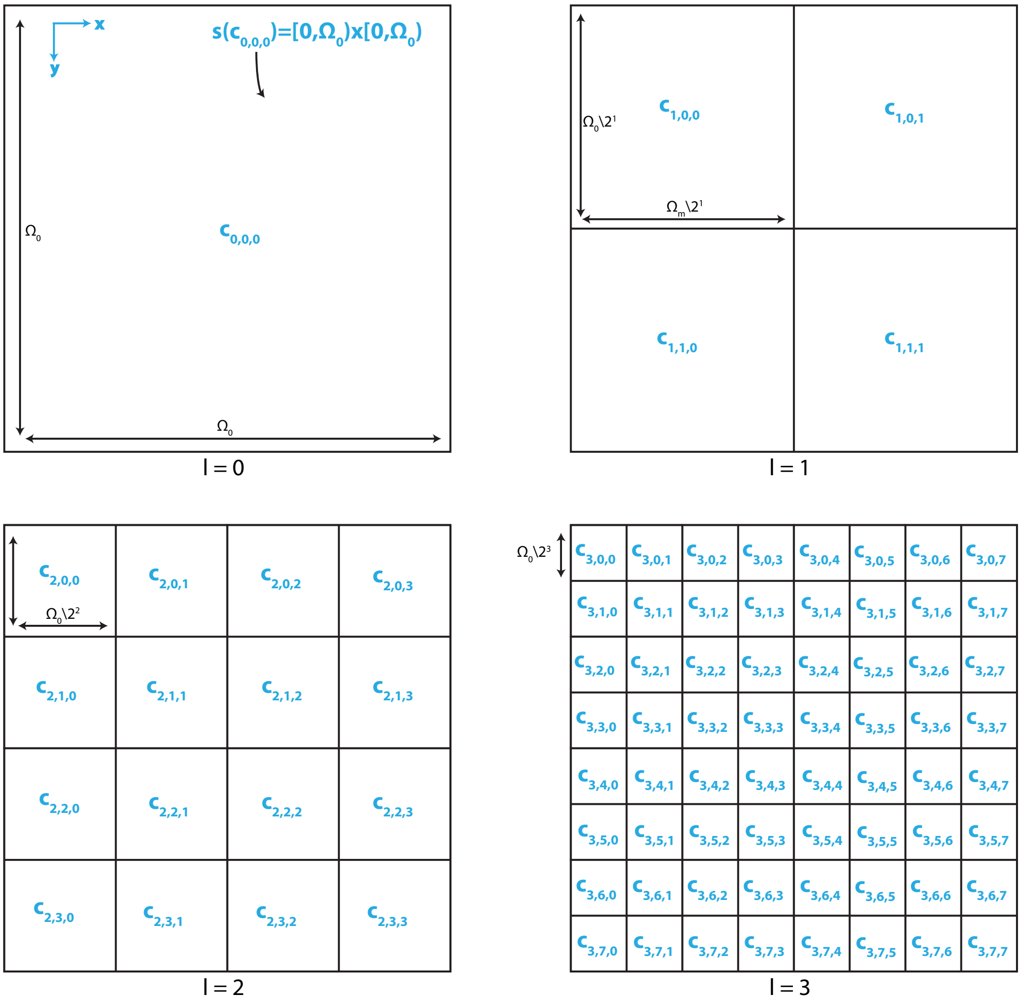
where $\mathbf{i} = i_1, .., i_n$ is multi-index notation for the spatial indices in each direction, and l indicates the level of the Particle Cell resolution. These Particle Cells form a partition using divisions of powers of 2, as follows,

$$\gamma(c_{\mathbf{i},l}) = [\frac{\Omega_0}{2^l}, \frac{\Omega_0}{2^{l+1}}) \times \prod_{\mathbf{i}} [i_k \frac{\Omega_0}{2^l}, (i_k + 1)\frac{\Omega_0}{2^l}) \tag{5.15}$$

where the product is over all spatial indices and therefore,

$$\bigcup_{c_{\mathbf{i},l} \in \mathcal{C}} \gamma(c_{\mathbf{i},l}) = \Omega^* \times \mathcal{R}. \tag{5.16}$$

Each Particle Cell forms regular elements, rectangles in 1D, a half-cubes in 2D, and half-hypercubes in 3D. The 1D example of these rectangles are given in


x
y
s(c0,0,0)=[0,Ω0)x[0,Ω0)
Ω0
c0,0,0
Ω0
Ω0\2^1
c1,0,0
c1,0,1
Ωm\2^1
c1,1,0
c1,1,1
c2,0,0
Ω0\2^2
c2,0,1
c2,0,2
c2,0,3
c2,1,0
c2,1,1
c2,1,2
c2,1,3
c2,2,0
c2,2,1
c2,2,2
c2,2,3
c2,3,0
c2,3,1
c2,3,2
c2,3,3
Ω0\2^3
c3,0,0 c3,0,1 c3,0,2 c3,0,3 c3,0,4 c3,0,5 c3,0,6 c3,0,7
c3,1,0 c3,1,1 c3,1,2 c3,1,3 c3,1,4 c3,1,5 c3,1,6 c3,1,7
c3,2,0 c3,2,1 c3,2,2 c3,2,3 c3,2,4 c3,2,5 c3,2,6 c3,2,7
c3,3,0 c3,3,1 c3,3,2 c3,3,3 c3,3,4 c3,3,5 c3,3,6 c3,3,7
c3,4,0 c3,4,1 c3,4,2 c3,4,3 c3,4,4 c3,4,5 c3,4,6 c3,4,7
c3,5,0 c3,5,1 c3,5,2 c3,5,3 c3,5,4 c3,5,5 c3,5,6 c3,5,7
c3,6,0 c3,6,1 c3,6,2 c3,6,3 c3,6,4 c3,6,5 c3,6,6 c3,6,7
c3,7,0 c3,7,1 c3,7,2 c3,7,3 c3,7,4 c3,7,5 c3,7,6 c3,7,7

l = 0

l = 1

l = 2

l = 3

Figure 5.1: Four levels l of $\mathcal{C}$ showing how Particle Cells in 2D $c_{\mathbf{i},l}$ partition the domain Ω on levels $l = \{0, 1, 2, 3\}$. Each square in the figure represents the spatial domain $s(c_{\mathbf{i},l})$ of a given Particle Cell $c_{\mathbf{i},l}$.

Figure 4.8. We note, that this is a different, but complimentary interpretation of Particle Cells to when constructing the implied resolution function out of particle cells as given in the previous chapter and shown in Figure 4.5. This partitioning is similar to those often used in quad and octree data structures, and as used in adaptive particle cell lists [13].

We define further properties of Particle Cells, reflecting their spatial domain, and resolution domain separately. The spatial domain of a Particle Cell is defined as,

$$s(c_{\mathbf{i},l}) = \prod_{\mathbf{i}} [i_k \frac{\Omega_0}{2^l}, (i_k + 1)\frac{\Omega_0}{2^l}) \tag{5.17}$$

Figure 5.1 shows an example of the spatial domain, $s(c_{\mathbf{i},l})$ of different Particle Cells for a range of l in 2D. The spatial domain of a Particle Cell $s(c_{\mathbf{i},l})$ is the area, or volume, of the domain Ω^* of which it partitions. Effectively forming dyadic cubes of the domain Ω [81]. The resolution domain of a Particle Cell is defined as

$$r(c_{\mathbf{i},l}) = [\frac{\Omega_0}{2^l}, \frac{\Omega_0}{2^{l+1}}). \tag{5.18}$$

Further, we define $l(c_{\mathbf{i},l}) = l$, to denote level of $c_{\mathbf{i},l}$ and $\mathbf{i}(c_{\mathbf{i},l}) = \mathbf{i}$ for the spatial coordinate of $c_{\mathbf{i},l}$. Now given these definitions we can now define relationships between the Particle Cells considering them as constructing a tree structure as shown in Figure 4.7. We define the set of descendants of a particle cell $c_{\mathbf{i},l}$ as

$$\mathcal{D}(c_{\mathbf{i},l}) = \{c^d_{\mathbf{i},l} \in \mathcal{C} : s(c^d_{\mathbf{i},l}) \subset s(c_{\mathbf{i},l})\}, \tag{5.19}$$

which is the set of all Particle Cells who's spatial domain overlaps with $c_{\mathbf{i},l}$ but have a smaller resolution than $r(c_{\mathbf{i},l})$. The first set of descendants, called children, are shown for a cell in green in Figure 4.7. Formally, children of $c_{\mathbf{i},l}$ are those $c^c_{\mathbf{i},l} \in D(c_{\mathbf{i},l})$ such that $l(c^c_{\mathbf{i},l}) = l(c_{\mathbf{i},l}) - 1$. We also then denote the parent of $c_{\mathbf{i},l}$, as $c_{\mathbf{i}/2,l-1}$, where $c_{\mathbf{i},l}$ is simply then the child of $c_{\mathbf{i}/2,l-1}$.

We also define the set of neighbors of a Particle Cell $c_{\mathbf{i},l}$, by first defining the interaction Particle Set

$$\mathcal{I}(c_{\mathbf{i},l}) = \{c^n_{\mathbf{i},l} \in \mathcal{C} : \exists \mathbf{x} \in s(c^n_{\mathbf{i},l}), \mathbf{y} \in s(c_{\mathbf{i},l}) : \mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))\} \tag{5.20}$$

which is the set of all Particle Cells $c^n_{\mathbf{i},l}$ for which there is exists a $\mathbf{x}$ in its spatial domain and also a $\mathbf{y}$ in the spatial domain of $c_{\mathbf{i},l}$ such that they could interact, i.e. $\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))$. Then using the interaction Particle Cell set, we define the neighbor Particle Cell set as

$$\mathcal{B}(c_{\mathbf{i},l}) = \{c^n_{\mathbf{i},l} \in \mathcal{I}(c_{\mathbf{i},l}) : \nexists c^{n'}_{\mathbf{i},l} \in \mathcal{I} : s(c^n_{\mathbf{i},l}) \subset s(c^{n'}_{\mathbf{i},l})\} \tag{5.21}$$

which is the set of all neighboring Particle Cells of highest level that $c_{\mathbf{i},l}$ can interact with (including $c_{\mathbf{i},l}$). This definition and the theorems proven below hold across general definitions of the interaction neighborhood $\mathcal{N}(\mathbf{y}, R(\mathbf{y}))$. For simplicity of explanation, here we present examples with the isotropic interaction neighbourhood $\mathcal{N}(\mathbf{y}, R(\mathbf{y})) = \{\mathbf{x} \in \Omega : |\mathbf{x} - \mathbf{y}| \leq R(\mathbf{y})\}$, as introduced earlier. For the isotropic interaction neighborhood, the neighbor Particle Cell set is simply the neighboring Particle Cells of $c_{\mathbf{i},l}$ on the same level. This is illustrated in Figure 4.7 with a 1D example of a neighbor Particle Cell set $\mathcal{B}(c_{\mathbf{i},l})$ in blue.

Using these we define a set $\mathcal{ND} \in \mathcal{C}$ that contains all descendants of a particular Particle Cell $c_{\mathbf{i},l}$ and its neighbors as

$$\mathcal{ND}(c_{\mathbf{i},l}) = \bigcup_{c^n_{\mathbf{i},l} \in \mathcal{B}(c^v_{\mathbf{i},l})} \bigcup_{c^d_{\mathbf{i},l} \in \mathcal{D}(c^c_{\mathbf{i},l})} c^d_{\mathbf{i},l}. \tag{5.22}$$

Then any Particle Cell set $\mathcal{V} \subset \mathcal{C}$ forms a partition of the spatial domain Ω^* iff,

$$\bigcup_{c^v_{\mathbf{i},l} \in \mathcal{V}} s(c^v_{\mathbf{i},l}) = \Omega^*. \tag{5.23}$$

Then we can also define the set of Particle Cell sets $\mathcal{V}$ that form a spatial partition as

$$\mathcal{S} = \{\mathcal{V} : \mathcal{V} \subset \mathcal{C}, \bigcup_{c^v_{\mathbf{i},l} \in \mathcal{V}} s(c^v_{\mathbf{i},l}) = \Omega^*\}. \tag{5.24}$$

Lastly, we formally introduce an additional property of a Particle Cell called *type*, $t(c_{\mathbf{i},l})$, discussed in the previous section for Particle Cells when compared to a Particle Cell set $\mathcal{T}$ in the following way

$$t(c_{\mathbf{i},l}, \mathcal{T}) = \begin{cases} 1, & c_{\mathbf{i},l} \in \mathcal{T} \\ 2, & c_{\mathbf{i},l} \notin \mathcal{T} \text{ and } \exists c^n_{\mathbf{i},l} \in \mathcal{B}(c_{\mathbf{i},l}) : c^n_{\mathbf{i},l} \in \mathcal{T} \\ 3, & otherwise \end{cases}$$

where we name the three different Particle Cell types as *seed*, *boundary*, and *filler* respectively.

5.3.2 Implied Resolution Function

Now we define the Implied Resolution Function for a set of Particle Cells $\mathcal{V}$ that forms a spatial partition. We begin by now defining a characteristic function in general dimension as

$$\phi(\mathbf{y}, c_{\mathbf{i},l}) = \begin{cases} 1 & \mathbf{y} \in s(c_{\mathbf{i},l}) \\ 0 & otherwise \end{cases} \tag{5.25}$$

that is, it is non-zero only at point within the spatial domain of the Particle Cell. Using this the Implied Resolution Function $R^*(\mathbf{y})$ for a set of Particle Cells $\mathcal{V}$ forming a partition of the spatial domain is defined as

$$R^*(\mathbf{y}, \mathcal{V}) = \sum_{c_{i,l} \in \mathcal{V}} \phi(\mathbf{y}, c_{\mathbf{i},l}) \frac{\Omega_0}{2^l} \tag{5.26}$$

where we often drop the dependence on $\mathcal{V}$ below, unless required. One can interpret this Resolution Function as being built out of cube blocks of length $\frac{\Omega_0}{2^l}$, as shown in Figure 4.5 for 1D. In 1D the blocks are squares, 2D cubes, and 3D hypercubes. (Note: this is different from how Particle Cells are used to partition the resolution domain)

5.3.3 Local Particle Cell set

Given these definitions, we can now represent the Local Resolution Estimate $L(\mathbf{y})$ as Particle Cells. We assume that we have the following inequality to satisfy

$$R(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} L(\mathbf{x}). \tag{5.27}$$

We introduce the general dimension Local Particle Cell (LPC) set $\mathcal{L} \subseteq \mathcal{C}$ that has members such that

$$\mathcal{L} = \{c_{\mathbf{i},l} \in \mathcal{C} | \exists \, (L(\mathbf{y}), \mathbf{y}) : \mathbf{y} \in s(c_{\mathbf{i},l}), L(\mathbf{y}) \in r(c_{\mathbf{i}/2,l-1})\}, \tag{5.28}$$

where $c_{\mathbf{i}/2,l-1}$ indicates the parent of $c_{\mathbf{i},l}$. In words, this takes the Local Resolution Estimate $L(\mathbf{y})$ and finds those Particle Cells $c_{\mathbf{i},l}$ whose parents intersect with $L(y)$ at locations inside the spatial domain $s(c_{\mathbf{i},l})$. An example was given in the previous section for 1D in Figure 4.8. We also define another set we call the natural Local Particle Cell (nLPC) set

$$\mathcal{L}_n = \{c_{\mathbf{i},l} \in \mathcal{C} : \exists \, (L(\mathbf{y}), \mathbf{y}) \in \gamma(c_{\mathbf{i},l})\}, \tag{5.29}$$

in words takes the Local Resolution Estimate $L(\mathbf{y})$ and finds those Particle Cells that the function intersects. The second definition comes in use slightly later for the equivalence optimization and is called 'natural' due to its simpler definition. In all except special cases, $\mathcal{L}$ does not form a partition of the spatial domain.

Maximum resolution level

In practice it is often useful to specify a minimum level l_{min} and maximum level l_{max}. For a given function we can define a minimum $L_{min}(\mathbf{y}) = \frac{\Omega_0}{2^{l_{min}}}$ and maximum value $L_{max}(\mathbf{y}) = \frac{\Omega_0}{2^{l_{max}}}$. Where for both $\mathcal{L}$ and $\mathcal{L}_n$ this effectively truncates any values with l below l_{min} to l_{min} and above l_{max} to l_{max}. (See A.4.1 for a description of implementation and constructing these sets).

5.3.4 Optimal Valid Particle Cell sets

Now we have a way to relate, a Particle Cell set to a Resolution Function, if now we re-formulate 5.27, in terms of this Implied Resolution Function we have

$$R^*(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} L(\mathbf{x}). \tag{5.30}$$

we can now use the Implied Resolution Function and present the following theorem:

Theorem 1. $\mathcal{V}$ will define an Implied Resolution Function $R^*(\mathbf{y})$ thats satisfies Cond. 5.30 for all $\mathbf{y} \in \Omega^*$, for a given $\mathcal{L}$, and called *valid* iff it forms a spatial partition and

1. $\forall c^v_{\mathbf{i},l} \in \mathcal{V}$ then $\{\mathcal{L} \cap \mathcal{ND}(c^v_{\mathbf{i},l})\} = \emptyset$

In words, for all Particle Cells $c_{\mathbf{i},l}$ in $\mathcal{V}$, the set is *valid*, if and only if, there are no Particle Cells that are descendants of $c_{\mathbf{i},l}$ or its neighbors in $\mathcal{L}$.
Proof:
Given a *valid* Particle Cell set $\mathcal{V}$, we suppose there exists at least one combination of $\mathbf{y} \in \Omega^*$ and $\mathbf{y}^* \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))$, such that

$$L(\mathbf{y}^*) < R^*(\mathbf{y}) \tag{5.31}$$

is true and therefore condition 5.30 is violated. In addition, there must exist $c^v_{\mathbf{i},l} \in \mathcal{V}$ such that $\mathbf{y} \in s(c^v_{\mathbf{i},l})$. From 5.26, we have

$$R^*(\mathbf{y}) = \frac{\Omega^*}{2^{l(c^v_{\mathbf{i},l})}} \tag{5.32}$$

and therefore if $L(\mathbf{y}^*) < \frac{\Omega^*}{2^{l(c^c_{\mathbf{i},l})}}$ then there must exist some $c^*_{\mathbf{i},l} \in \mathcal{L}$, for which $l(c^*_{\mathbf{i},l}) < l(c^v_{\mathbf{i},l})$ and $\mathbf{y}^* \in s(c^*_{\mathbf{i},l})$. Now since $\mathbf{y}^* \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))$ and $l(c^*_{\mathbf{i},l}) < l(c^v_{\mathbf{i},l})$ it implies that

$$c^*_{\mathbf{i},l} \in \mathcal{ND}(c^v_{\mathbf{i},l}) \tag{5.33}$$

and therefore

$$\exists c^a_{\mathbf{i},l} \in \mathcal{V} : \{\mathcal{L} \cap \mathcal{ND}(c^a_{\mathbf{i},l})\} \neq \emptyset \tag{5.34}$$

and proves Theorem. 1 by contradiction. □

Now we consider conditions on $\mathcal{V}$ that would define it as *optimal.* Consider $\mathbf{V}$ to be the set of all Particle Cell sets $\mathcal{V}$ that satisfy Theorem 1 and are valid. Then a Particle Cell set $\mathcal{V}$ will be optimal if it satisfies

$$\operatorname*{arg\,max}_{\mathcal{V}^* \in \mathbb{V}} \int_{\Omega^*} R^*(\mathbf{y}, \mathcal{V}^*) d\Omega^*. \tag{5.35}$$

which is equivalent to finding the largest everywhere $R^*(y)$ that satisfies 5.30. Which is equivalent to

$$\operatorname*{arg\,max}_{R^* \in \mathcal{R}^*} \int_{\Omega^*} R^*(\mathbf{y}) d\Omega^*. \tag{5.36}$$

where $\mathcal{R}^*$ is the set of all Implied Resolution Functions defined as 5.26 ($R^* : \Omega^* \to \mathbb{R}^+$) that satisfy 5.30. We can now state the following theorem for satisfying 5.35,

Theorem 2. Given $\mathcal{V} \subset \mathcal{C}$, that is *valid*, $\mathcal{V}$ will satisfy Cond. 5.35 and be *optimal*, iff, there does not exist a $\mathcal{W} \subset \mathcal{C}$ where $\mathcal{W} \neq \mathcal{V}$ and is *valid* for $\mathcal{L}$ and where $\mathcal{V}$ can be formed from the elements of $\mathcal{W}$ and its descendants. Formally, $\mathcal{V}$ is optimal, if there does not exist any valid $\mathcal{W}$ such that for any $c^w_{\mathbf{i},l}$ or $c^v_{\mathbf{i},l}$ the following holds

$$\left(c^w_{\mathbf{i},l} \in \mathcal{W}, c^v_{\mathbf{i},l} \in \mathcal{V}\right) : c^v_{\mathbf{i},l} \in D(c^w_{\mathbf{i},l}). \tag{5.37}$$

In words, $\mathcal{V}$, is optimal, if there does not exist another arrangement of Particle Cells that form a spatial partition and is valid while having a larger resolution anywhere in the domain.

Proof:

Lets consider two Particle Cell sets $\mathcal{V}$ and $\mathcal{W}$, where both are *valid* with respect to $\mathcal{L}$, and $\mathcal{W} \neq \mathcal{V}$, and $\mathcal{V}$ is *optimal.* Now we suppose that, Cond. 5.35 is violated, that is

$$\int_{\Omega^*} R^*(\mathbf{y}, \mathcal{V}) d\Omega^* < \int_{\Omega^*} R^*(\mathbf{y}, \mathcal{W}) d\Omega^*, \tag{5.38}$$

that we can re-write as

$$\int_{\Omega^*} \sum_{c_{i,l} \in \mathcal{V}} \phi(\mathbf{y}, c_{\mathbf{i},l}) \frac{\Omega_0}{2^l} d\Omega^* < \int_{\Omega^*} \sum_{c_{i,l} \in \mathcal{W}} \phi(\mathbf{y}, c_{\mathbf{i},l}) \frac{\Omega_0}{2^l} d\Omega^*. \tag{5.39}$$

Given the above inequality to hold, there must exist $\mathbf{y} \in \Omega^*$ where the following holds for some $c^v_{\mathbf{i},l} \in \mathcal{V}$ and $c^w_{\mathbf{i},l} \in \mathcal{W}$,

$$0 < \phi(\mathbf{y}, c^v_{\mathbf{i},l}) \frac{\Omega_0}{2^{l(c^v_{\mathbf{i},l})}} < \phi(\mathbf{y}, c^w_{\mathbf{i},l}) \frac{\Omega_0}{2^{l(c^w_{\mathbf{i},l})}} \tag{5.40}$$

which implies that $l(c^w_{\mathbf{i},l}) < l(c^v_{\mathbf{i},l})$ and further that $s(c^v_{\mathbf{i},l}) \subset s(c^w_{\mathbf{i},l})$ and hence

$$c^v_{\mathbf{i},l} \in D(c^w_{\mathbf{i},l}), \tag{5.41}$$

which violates Theorem 2, and thus concludes the proof through contradiction. □.

Therefore a Particle Cell set $\mathcal{V}$ is an Optimal Valid Particle Cell (OVPC) set if it satisfies both Theorem 1 and Theorem 2.

5.4 Pulling Scheme

Here we present the additional results based on the above that are used by the Pulling Scheme. These are the general definitions of the three properties from the previous chapter. We begin by defining

Definition 1. $\mathcal{V}^*(c_{\mathbf{i},l})$ is the optimal Particle Cell set for $\mathcal{L} = \{c_{\mathbf{i},l}\}$

That is the OVPC set for a LPC set with only one Particle Cell $c_{\mathbf{i},l}$.

5.4.1 Self-similarity and production of individual solutions

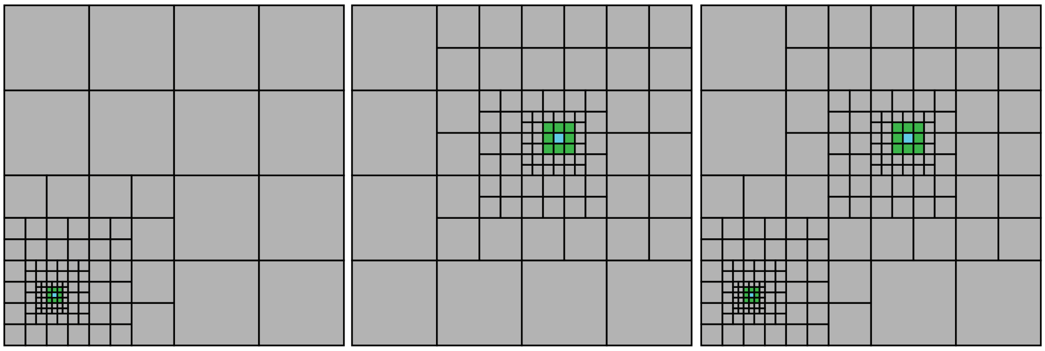
The first is an observation that the solution $\mathcal{V}^*(c_{\mathbf{i},l})$ is highly predictable and shows self-similarity regarding its relative local structure. This is shown for two different $c_{\mathbf{i},l}$ in 2D in Figure 5.2, where the Particle Cells are colored by their type. As in 1D, the solutions are defined by a central *seed* Particle Cell, surrounded by a layer of *boundary* and then *filler* cells. The remainder of the domain is then filled with Particle Cells increasing by one level across neighbors, adding Particle Cells on the same level when needed to maintain a spatial partition. In Algorithm 3 we provide a possible pseudo-code for this process.

```
Data: $\mathcal{L} = \{c_{\mathbf{i},l}\}$
Result: $\mathcal{V}(c_{\mathbf{i},l})$
1.) add $c_{\mathbf{i},l}$ to $\mathcal{V}$;
2.) add $c^b_{\mathbf{i},l} \in \mathcal{B}(c_{\mathbf{i},l})$ to $\mathcal{V}$;
3.) add $c^f_{\mathbf{i},l} \in \mathcal{B}(c^b_{\mathbf{i},l})$ for each neighbour of these $c^b_{\mathbf{i},l}$ not already in $\mathcal{V}$ add
  to $\mathcal{T}$ and $\mathcal{V}$;
for $l_c = l(c_{\mathbf{i},l}) : -1 : 0$ do
    propogate_filler;
    propogate_filler;
end
```

Algorithm 3: Produces the OVPC $\mathcal{V}$ for an LPC $\mathcal{L}$ with only one Particle Cell $c_{\mathbf{i},l}$ with level $l(c_{\mathbf{i},l})$

```
Function propogate_filler
    foreach $c^c_{\mathbf{i},l} \in \mathcal{T}$ do
        foreach $c^n_{\mathbf{i},l} \in \mathcal{B}(c^c_{\mathbf{i},l})$ do
            if $c^n_{\mathbf{i},l} \notin \mathcal{V}$ and $D(c^n_{\mathbf{i},l}) \in \mathcal{V} = \emptyset$ then
                if $s(c^c_{\mathbf{i},l}) \notin s(c^n_{\mathbf{i}/2,l})$ then
                    add the parent $c^n_{\mathbf{i}/2,l}$ of neighbour $c^n_{\mathbf{i},l}$ ;
                    Add $c^n_{\mathbf{i}/2,l}$ to $\mathcal{T}_t$ and $\mathcal{V}$
                else
                    add the neighbour $c^n_{\mathbf{i},l}$, as adding parent would cause
                      over-lap;
                    Add $c^n_{\mathbf{i},l}$ to $\mathcal{T}_t$ and $\mathcal{V}$
                end
            end
        end
    end
    Set $\mathcal{T} \leftarrow \mathcal{T}_t$;
    Set $\mathcal{T}_t \leftarrow \emptyset$;
```

Algorithm 4: Propogates filler Particle Cells through the domain



$\mathcal{V}$ $(c_{(10,10),6})$ $\mathcal{V}$ $(c_{(20,20),5})$ $\mathcal{V}$ for $\mathcal{L}=\{c_{(10,10),6}, c_{(20,20),5}\}$

Figure 5.2: Example of the optimal particle cell set $\mathcal{V}$ (right) for $\mathcal{L} = \{c_{(10,10),6}, c_{(20,20),5}\}$ in 2D, and the individual optimal solutions $\mathcal{V}^*(c_{(10,10),6})$ (left) and $\mathcal{V}^*(c_{(20,20),5})$ (right) that can be used to combined using the *separability* property to construct $\mathcal{V}$. The particle cells are colored in the following way, a particle cell is *blue* if its type is a *seed* if it is in the local particle cell set, $c_{\mathbf{i},l} \in \mathcal{L}$, a cell is *green* if it is of type *boundary* and therefore has a neighbor that is in the local particle cell set and is *grey* if it as of type *filler*.

5.4.2 Separability

We present Lemma 1, that is the basis of the separability property used in Pulling Scheme.

Lemma 1. Given $\mathcal{V} \subset \mathcal{C}$ is optimal, with respect to $\mathcal{L}$, and let $\mathcal{V}^*(c_{\mathbf{i},l}) \subset \mathcal{C}$ be optimal for the local set $\mathcal{L}^*(c_{\mathbf{i},l}) = \{c_{\mathbf{i},l}\}$. Then,

$$\mathcal{V} = \text{minhull}\left(\bigcup_{c^l_{\mathbf{i},l} \in \mathcal{L}} \mathcal{V}^*(c^l_{\mathbf{i},l})\right) \tag{5.42}$$

where for $\mathcal{T} \subseteq \mathcal{C}$,

$$\text{minhull}(\mathcal{T}) = \{c^s_{\mathbf{i},l} \in \mathcal{T} : \{D(c^s_{\mathbf{i},l}) \cap \mathcal{T}\} = \emptyset\}. \tag{5.43}$$

In words, Lemma 1 states that the optimal solution $\mathcal{V}$, for a given LPC set $\mathcal{L}$, can be constructed by forming the valid and optimal set for each Particle Cell in $\mathcal{L}$ separately $\mathcal{V}^*(c_{\mathbf{i},l})$, and then forming a set with the Particle Cells $c_{\mathbf{i},l}$ at each point y with the smallest Implied Resolution Function $R^*(\mathbf{y}, \mathcal{V}^*(c_{\mathbf{i},l}))$ (highest level l). We call the above property, *separability*. Figure 5.2 shows the property in 2D and Figure 4.10 in 1D. One can intuitively confirm that the configurations are optimal, by replacing any Particle Cell by its parent, and then checking if Theorem. 1 holds.

Proof:
Lets consider $\mathcal{V}$, which is *optimal* for $\mathcal{L}$, and $\hat{\mathcal{V}} = \text{minhull}\left(\bigcup_{c^l_{\mathbf{i},l} \in \mathcal{L}} \mathcal{V}^*(c^l_{\mathbf{i},l})\right)$. Now propose that there exists some $\hat{c_{\mathbf{i},l}} \in \hat{\mathcal{V}}$ such that

$$\exists c^*_{\mathbf{i},l} \in \mathcal{L} : c^*_{\mathbf{i},l} \in \mathcal{ND}(\hat{c_{\mathbf{i},l}}) \tag{5.44}$$

and therefore $\hat{\mathcal{V}}$ would not be *valid* by Theorem 1. However, given that $\mathcal{V}^*(c^*_{\mathbf{i},l})$ is valid, it forms a spatial partition, and $\hat{c_{\mathbf{i},l}} \notin \mathcal{V}^*(c^*_{\mathbf{i},l})$, therefore

$$\exists \bar{c_{\mathbf{i},l}} \in \mathcal{V}^*(c^*_{\mathbf{i},l}) : \bar{c_{\mathbf{i},l}} \in D(\hat{c_{\mathbf{i},l}}) \tag{5.45}$$

and since $\mathcal{V}^*(c^*_{\mathbf{i},l}) \subset \bigcup_{c^l_{\mathbf{i},l} \in \mathcal{L}} \mathcal{V}^*(c^l_{\mathbf{i},l})$ then,

$$\begin{aligned} D(\hat{c_{\mathbf{i},l}}) \cap \bigcup_{c^l_{\mathbf{i},l} \in \mathcal{L}} \mathcal{V}^*(c^l_{\mathbf{i},l}) &\supseteq \bar{c_{\mathbf{i},l}} && (5.46)\\ &\neq \emptyset && (5.47) \end{aligned}$$

therefore violating Lemma 1 as $\hat{c_{\mathbf{i},l}} \in \hat{\mathcal{V}}$. Therefore, by contradiction, given Lemma. 1 holds, $\hat{\mathcal{V}}$ will be *valid.*

Now, let us propose, that $\hat{\mathcal{V}}$ is not *optimal*, that is there exists some $\mathcal{W}$ such that

$$\int_{\Omega^*} R^*(\mathbf{y}, \hat{\mathcal{V}}) d\Omega^* < \int_{\Omega^*} R^*(\mathbf{y}, \mathcal{W}) d\Omega^*, \tag{5.48}$$

following the arguments for the proof of Theorem 2 above this implies there would exist some $c^w_{\mathbf{i},l} \in \mathcal{W}$ and some $\hat{c_{\mathbf{i},l}} \in \hat{\mathcal{V}}$ such that

$$\hat{c_{\mathbf{i},l}} \in D(c^w_{\mathbf{i},l}). \tag{5.49}$$

Given that $\hat{c_{\mathbf{i},l}} \in \hat{\mathcal{V}}$ there exists some $\bar{c_{\mathbf{i},l}} \in \mathcal{L}$ such that $\hat{c_{\mathbf{i},l}} \in \mathcal{V}^*(\bar{c_{\mathbf{i},l}})$. However, given $c^w_{\mathbf{i},l} \notin \mathcal{V}^*(\bar{c_{\mathbf{i},l}})$ and $\mathcal{V}^*(\bar{c_{\mathbf{i},l}})$ is *optimal* then,

$$\left(\mathcal{ND}(c^w_{\mathbf{i},l}) \cap \mathcal{V}^*(\bar{c_{\mathbf{i},l}})\right) \supseteq \hat{c_{\mathbf{i},l}} \neq \emptyset. \tag{5.50}$$

Since $\bar{c_{\mathbf{i},l}} \in \mathcal{L}$, then $\mathcal{W}$ cannot be valid. Implying that $\hat{\mathcal{V}}$ must be *optimal* for $\mathcal{L}$ and given the *optimal* solution is unique implies

$$\hat{\mathcal{V}} = \mathcal{V} \tag{5.51}$$

$\square$.

5.4.3 Redundancy of Particle Cells

The third property relates to the redundancy of Particle Cells in $\mathcal{L}$ that have descendants in $\mathcal{L}$,

Lemma 2. Given any two Particle Cells $c_{\mathbf{i},l}$ and $c^p_{\mathbf{i},l}$, where $c_{\mathbf{i},l} \in D(c^p_{\mathbf{i},l})$ then

$$\text{minhull}(\{\mathcal{V}^*(c_{\mathbf{i},l}), \mathcal{V}^*(c^p_{\mathbf{i},l})\}) = \mathcal{V}^*(c_{\mathbf{i},l}). \tag{5.52}$$

In words, the optimal valid solution of Particle Cells for which one is the descendant of the other will be the individual valid solution of the descendant Particle Cell.

Proof:

Lets suppose that,

$$\exists \hat{c_{\mathbf{i},l}} \in \text{minhull}(\{\mathcal{V}^*(c_{\mathbf{i},l}), \mathcal{V}^*(c^p_{\mathbf{i},l})\}) : \hat{c_{\mathbf{i},l}} \notin \mathcal{V}^*(c_{\mathbf{i},l}) \tag{5.53}$$

and then

$$\hat{c_{\mathbf{i},l}} \in \mathcal{V}^*(c^p_{\mathbf{i},l}) \tag{5.54}$$

and Lemma 2 is violated. However, given the definition of the minhull operation, and the fact that $\mathcal{V}^*(c_{\mathbf{i},l})$ must form a spatial partition this implies that

$$\exists c^*_{\mathbf{i},l} \in \mathcal{V}^*(c_{\mathbf{i},l}) : \hat{c_{\mathbf{i},l}} \in D(c^*_{\mathbf{i},l}). \tag{5.55}$$

However, given that $\mathcal{V}^*(c^p_{\mathbf{i},l})$ is optimal by definition, then if $c^*_{\mathbf{i},l} \notin \mathcal{V}^*(c^p_{\mathbf{i},l})$, then by Theorem 2

$$c^p_{\mathbf{i},l} \in \mathcal{ND}(c^*_{\mathbf{i},l}) \tag{5.56}$$

and since $c_{\mathbf{i},l} \in D(c^p_{\mathbf{i},l})$ by construction then also

$$c_{\mathbf{i},l} \in \mathcal{ND}(c^*_{\mathbf{i},l}) \tag{5.57}$$

but this results in a contradiction, as then $\mathcal{V}^*(c_{\mathbf{i},l})$ would not be valid by Theorem. 1 □.

5.4.4 Equivalence

Here we show that an equivalent solution can be obtained, by solving for a smaller set of Particle Cells, that can then later be used to directly form $\mathcal{V}$. First, let us define

Definition 2. Let $\mathcal{V}_n$ be the *optimal* Particle Cell set for the natural Local Particle Cell set $\mathcal{L}_n$ formed from $L(\mathbf{y})$ as in 5.29

Then we have the following result,

Lemma 3. Given $\mathcal{V}_n$ is *optimal* and *valid* for $\mathcal{L}_n$ then

$$\mathcal{V} = \left\{ c_{\mathbf{i},l} \in \mathcal{C} \middle| (c_{\mathbf{i},l} \in \mathcal{V}_n, t(c_{\mathbf{i},l}, \mathcal{L}_n) = 3) \text{ or } \left(c_{\mathbf{i}/2,l-1} \in \mathcal{V}_n, t(c_{\mathbf{i}/2,l-1}, \mathcal{L}_n) < 3\right) \right\} \tag{5.58}$$

where $\mathcal{V}$ is the optimal valid Particle Cell set for $\mathcal{L}$ and $c_{\mathbf{i}/2,l-1}$ denotes the parent Particle Cell of $c_{\mathbf{i},l}$.

In words, $\mathcal{V}$ is constructed by taking all those Particle Cells that have type filler in $\mathcal{V}$, and taking the children of all Particle Cells in $\mathcal{V}_n$ that are of type seed or boundary (Where type is defined relative to $\mathcal{L}_n$).

Which means that finding for $\mathcal{V}_n$ with respect to $\mathcal{L}_N$ is equivalent to $\mathcal{V}$ for $\mathcal{L}$. This is useful because, $\#\mathcal{L} > \#\mathcal{L}_n$ and the maximum level l^n_{max} in $\mathcal{L}_n$ is one level less than l_{max} of $\mathcal{L}$ by construction (See 4.2.3). See Figure 5.3 below for a 2D example. *Proof*:
Here we need to show that if $\mathcal{V}_n$ is the OVPC for $\mathcal{L}_n$ and we define

$$\hat{\mathcal{V}}(\mathcal{V}_n) = \left\{ c_{\mathbf{i},l} \in \mathcal{C} \middle| (c_{\mathbf{i},l} \in \mathcal{V}_n, t(c_{\mathbf{i},l}, \mathcal{L}_n) = 3) \text{ or } \left(c_{\mathbf{i}/2,l-1} \in \mathcal{V}_n, t(c_{\mathbf{i}/2,l-1}, \mathcal{L}_n) < 3\right) \right\} \tag{5.59}$$

then $\hat{\mathcal{V}} = \mathcal{V}$ where $\mathcal{V}$ is the OVPC for $\mathcal{L}$. We do this by relying on Lemma 1. That we can decompose our solution of

$$\mathcal{V}_n = \text{minhull}\left(\bigcup_{c^l_{\mathbf{i},l} \in \mathcal{L}_n} \mathcal{V}^*(c^l_{\mathbf{i},l}) \right) \tag{5.60}$$

and

$$\mathcal{V} = \text{minhull}\left(\bigcup_{c^l_{\mathbf{i},l} \in \mathcal{L}} \mathcal{V}^*(c^l_{\mathbf{i},l}) \right) \tag{5.61}$$

then since by construction for $c^*_{\mathbf{i},l} \in \mathcal{L}$ there exists a $c^{*n}_{\mathbf{i},l} \in \mathcal{L}_n$ such that $c^*_{\mathbf{i},l} \in D(c^{*n}_{\mathbf{i},l})$, and $l(c^*_{\mathbf{i},l}) = l(c^{*n}_{\mathbf{i},l}) + 1$ that if we can show that $\hat{\mathcal{V}}(\mathcal{V}^*(c^{n*}_{\mathbf{i},l})) = \mathcal{V}^*(c^*_{\mathbf{i},l})$

then

$$\begin{aligned}\mathcal{V} &= \text{minhull}\left(\bigcup_{c^l_{\mathbf{i},l}\in\mathcal{L}_n} \hat{\mathcal{V}}(\mathcal{V}^*(c^l_{\mathbf{i},l}))\right) \\ &= \hat{\mathcal{V}}\left(\text{minhull}\left(\bigcup_{c^l_{\mathbf{i},l}\in\mathcal{L}_n} \mathcal{V}^*(c^l_{\mathbf{i},l})\right)\right) \\ &= \hat{\mathcal{V}}(\mathcal{V}_n)\end{aligned} \tag{5.62}$$

the operations can be taken out of the union and minhull, due to the operation always taking the smallest Particle Cell and the direct correspondence between the two sets (Note: I have no formal proof of this property, but it seems to follow from the definitions).

Therefore, we consider such a $\mathcal{L} = \{c^*_{\mathbf{i},l}\}$ and $\mathcal{L}_n = \{c^{*n}_{\mathbf{i},l}\}$ and consider $\mathcal{V}_n = \mathcal{V}^*(c^{*n}_{\mathbf{i},l})$ and $\bar{\mathcal{V}} = \hat{V}(\mathcal{V}^*(c^{*n}_{\mathbf{i},l}))$. Now lets assume that $\mathcal{V}_n$ is valid and optimal solution with respect to $\mathcal{L}_n$, and assume that $\bar{\mathcal{V}}$ is not valid with respect to $\mathcal{L}$.

If $\bar{\mathcal{V}}$ in not valid, then there must exist $c^v_{\mathbf{i},l} \in \bar{\mathcal{V}}$ such that

$$c^*_{\mathbf{i},l} \in \mathcal{ND}(c^v_{\mathbf{i},l}). \tag{5.63}$$

Given this holds, then we consider the validity of $\mathcal{V}_n$ with respect to $\mathcal{L}_n$. We treat this in two cases.

First, suppose that $l(c^*_{\mathbf{i},l}) < l(c^v_{\mathbf{i},l}) + 1$, and $c^v_{\mathbf{i},l} \in \mathcal{V}_n$ which implies that $l(c^{n*}_{\mathbf{i},l}) < l(c^v_{\mathbf{i},l})$. Also, given $c^*_{\mathbf{i},l} \in D(c^{n*}_{\mathbf{i},l})$ then

$$c^{n*}_{\mathbf{i},l} \in \mathcal{ND}(c^v_{\mathbf{i},l}). \tag{5.64}$$

If $c^{vn}_{\mathbf{i},l} \in \mathcal{V}_n$, where $c^v_{\mathbf{i},l}$ is the child of $c^{vn}_{\mathbf{i},l}$, then since $\mathcal{ND}(c^v_{\mathbf{i},l}) \subset \mathcal{ND}(c^{nv}_{\mathbf{i},l})$ then also

$$c^{n*}_{\mathbf{i},l} \in \mathcal{ND}(c^{nv}_{\mathbf{i},l}). \tag{5.65}$$

This leads to $\mathcal{V}_n$ violating Theorem 1 with respect to $\mathcal{L}_n$.

Now in the second case, $l(c^*_{\mathbf{i},l}) = l(c^v_{\mathbf{i},l}) + 1$, implying that $l(c^{n*}_{\mathbf{i},l}) = l(c^v_{\mathbf{i},l})$. However, this implies that $c^{vn}_{\mathbf{i},l} \in \mathcal{V}_n$ as otherwise $t(c^v_{\mathbf{i},l}, \mathcal{L}_n) = 1$ and $l(c^v_{\mathbf{i},l}) = l(c^*_{\mathbf{i},l})$. Therefore again since $\mathcal{ND}(c^v_{\mathbf{i},l}) \subset \mathcal{ND}(c^{nv}_{\mathbf{i},l})$ then also

$$c^{n*}_{\mathbf{i},l} \in \mathcal{ND}(c^{nv}_{\mathbf{i},l}) \tag{5.66}$$

and $\mathcal{V}_n$ is invalid with respect to $\mathcal{L}_n$.

Therefore, given we assume $\mathcal{V}_n$ is valid w.r.t $\mathcal{L}_n$ then $\bar{\mathcal{V}}$ must also be valid w.r.t $\mathcal{L}$.

The second step is to show that when $\bar{\mathcal{V}}$ is optimal w.r.t $\mathcal{L}$ then also $\mathcal{V}_n$ is to $\mathcal{L}_n$. So again we assume that $\mathcal{V}_n$ is optimal, but $\bar{\mathcal{V}}$ is not. We follow on from the proof of Theorem 2 which gives us that therefore there exists some $\mathcal{W}$ such that $c^{v}_{\mathbf{i},l} \in \bar{\mathcal{V}}$ and $c^{w}_{\mathbf{i},l} \in \mathcal{W}$ and $s(c^{v}_{\mathbf{i},l}) \subset s(c^{w}_{\mathbf{i},l})$ and hence

$$c^{v}_{\mathbf{i},l} \in D(c^{w}_{\mathbf{i},l}). \tag{5.67}$$

For this there are again two cases, one where $c^{v}_{\mathbf{i},l} \in \mathcal{V}_n$ and the other where $c^{nv}_{\mathbf{i},l} \in \mathcal{V}_n$ where $c^{v}_{\mathbf{i},l} \in D(p^{nv})$ and $l(c^{nv}_{\mathbf{i},l}) = l(c^{v}_{\mathbf{i},l}) - 1$.

First, let us consider the case of $c^{c}_{\mathbf{i},l} \in \mathcal{V}_n$. We then have that $c^{v}_{\mathbf{i},l} \in D(c^{w}_{\mathbf{i},l})$. However, since $c^{*}_{\mathbf{i},l} \in D(c^{n*}_{\mathbf{i},l})$ and $c^{w}_{\mathbf{i},l} \notin \bar{\mathcal{V}}$ means that

$$c^{*}_{\mathbf{i},l} \in \mathcal{ND}(c^{w}_{\mathbf{i},l}) \tag{5.68}$$

which directly implies that

$$c^{n*}_{\mathbf{i},l} \in \mathcal{ND}(c^{w}_{\mathbf{i},l}). \tag{5.69}$$

However, this would make $\mathcal{V}_n$ not valid.

Now in the second case we have $c^{nv}_{\mathbf{i},l} \in \mathcal{V}_n$, and hence $t(c^{nv}_{\mathbf{i},l}, \mathcal{L}_n) < 3$. So we know that,

$$c^{*n}_{\mathbf{i},l} \in \mathcal{B}(c^{nv}_{\mathbf{i},l}) \tag{5.70}$$

such that

$$c^{*}_{\mathbf{i},l} \in \mathcal{ND}(c^{w}_{\mathbf{i},l}). \tag{5.71}$$

This now contradicts that $\mathcal{W}$ can be valid for $\mathcal{L}$. Hence given $\mathcal{V}_n$ is optimal for $\mathcal{L}_n$, $\bar{\mathcal{V}}$ must also be optimal for $\mathcal{L}$.

Now given there is a unique optimal solution then necessarilly,

$$\hat{V}(\mathcal{V}^{*}(c^{*n}_{\mathbf{i},l})) = \mathcal{V}^{*}(c^{*n}_{\mathbf{i},l}) \tag{5.72}$$

for any $c^{*}_{\mathbf{i},l} \in D(c^{*n}_{\mathbf{i},l})$, and $l(c^{*}_{\mathbf{i},l}) = l(c^{*n}_{\mathbf{i},l}) + 1$, and from our arguments above this leads to $\hat{\mathcal{V}}(\mathcal{V}_n) = \mathcal{V}$ and concludes the proof. $\square$

5.4.5 Algorithm description

The above Theorems and Lemmas are used to form a class of algorithms we call the Pulling Scheme that computes an optimal particle cell set $\mathcal{V}$ from an LPC set $\mathcal{L}$. In the previous chapter, in Section 4.2.3, we introduced the ideas and concepts, and do not repeat this here. The algorithm descriptions in the previous chapter focused on an implementation using explicit storage of the full particle cell set $\mathcal{C}$. A full algorithmic description of such an approach is given in A.4.1, and it is the algorithm used in the benchmarking sections below.

As an alternative, we provide an additional pseudo-code for a more general form of the algorithm that does not rely on explicit storage of the Particle Cell tree. This form could be useful in situations where the memory cost of storing the tree becomes prohibitive, or the APR is being iterated through time. Alternatives to using such a tree include the use of hash tables, or sparse data-structures to handle the Particle Cells. I foresee that such approaches could be useful if the Pulling Scheme were used for other applications such as the numerical solution of differential equations. In fact, in early iterations of this work, the Pulling Scheme was implemented using a hash-tablestructure. The mesh-implementation was later developed for its ease of parallelization and cache-efficiency. In the case of LSFM data, the size of the full Particle Cell structure is small in comparison to the raw images. Hence, the memory cost is of little impact to its use; this does not apply in general.

5.4.6 Generic algorithm

The pseudo-code for a generic approach is outlined in Algorithms 5-8 and we give a description below. We assume that a method exists for propagating individual solutions on a given level, as in Algorithm 6, and do not describe a specific implementation here. In the description, either $\mathcal{L}$ could be used or $\mathcal{L}_n$ with then the application of Lemma 3. Therefore, if $\mathcal{L}$ is used l_{max}, and if $\mathcal{L}_n$, $l^n_{max} = l_{max} - 1$.

Given that we can construct individual optimal solutions, the separability property (Lemma 1), tells us that the optimal solution for the set $\mathcal{L}$ can be constructed by taking the Particle Cell with the highest level l of all individual solutions at each point in space $\mathbf{y}$. Further, from Lemma 2 we can ignore all $c_{\mathbf{i},l} \in \mathcal{L}$ that have descendants in $\mathcal{L}$. If we construct the set $\mathcal{V}$ by starting from our maximum level l_{max}. We then know from Lemma 2, that if we add a Particle Cell, this will be the optimal Particle Cell $c_{\mathbf{i},l}$ for that location as any Particle Cells in $\mathcal{L}$ for which $c_{\mathbf{i},l}$ is a descendant cannot be in $\mathcal{V}$. Therefore, we can ignore these $c_{\mathbf{i},l} \in \mathcal{L}$ when constructing $\mathcal{L}$. We can achieve this by introducing a new property, *status*, which indicates if the cell has descendants

[1].

The pseudo-code for the Pulling Scheme in Algorithm 5. In the algorithm, a Particle Cell is set to IN-ACTIVE, if it has descendants in $\mathcal{L}$, this is done each time a seed type Particle Cell is checked (Algorithm 7). The next step involves propagating the individual solution $\mathcal{V}^*(c_{\mathbf{i},l})$ of all seed type Particle Cells on that level, and adding the Particle Cells in $c^i_{\mathbf{i},l}\mathcal{V}^*(c_{\mathbf{i},l})$ to $\mathcal{V}$ if they are not already in $\mathcal{V}$. If the propagation step for $c_{\mathbf{i},l}$ results in adding a Particle Cell to $\mathcal{V}$, $c_{\mathbf{i},l}$ is added to a temporary particle set $\mathcal{T}_{next}$ (Algorithm 6). Further, if $c^*_{\mathbf{i},l}$ is added, any $c_{\mathbf{i},l} \in \mathcal{L}$, that for which $c^*_{\mathbf{i},l}$ is the descendant, are set to PROPOGATE, unless already IN-ACTIVE (Algorithm 8). This step is required because, although these particle cells cannot be in $\mathcal{V}$ due to Theorem 1, they can impact the solution through Lemma 1. The algorithm then iterates to the next level l_{max-1}., with first assigning $\mathcal{T}_{current} = \mathcal{T}_{next}$, and setting $\mathcal{T}_{next} = \emptyset$. The steps of adding ACTIVE Particle Cells from $\mathcal{L}$ to $\mathcal{V}$ and propagating the solution for $c_{\mathbf{i},l}$ with status $ACTIVE$ or $PROPOGATE$ is repeated. Then a final step propagating the solution of those $c_{\mathbf{i},l}$ in the temporary Particle Cell set $\mathcal{T}_{current}$, and adding them to $\mathcal{T}_{next}$ if they again add $c^*_{\mathbf{i},l} \in \mathcal{V}^*(c_{\mathbf{i},l})$ to $\mathcal{V}$. This process is repeated until the minimum level l_{min} is reached, and $\mathcal{V}$ has been constructed.

5.4.7 Memory and computational complexity

The memory and computational complexity depend on the exact implementation and data structures used. In 4.2.3 we discussed the complexity of the mesh pyramid implementation used here. Here we, briefly sketch some arguments that the memory and computational cost for a mesh-free implementation should be $\mathcal{O}(\#\mathcal{V})$ and this is worst-case $\mathcal{O}(N)$.

Let's assume that the data structure has $\mathcal{O}(1)$ access for adding and checking for Particle Cells. Also, we have the ability to iterate over Particle Cells by level; the computational complexity should scale with $\mathcal{O}(\#\mathcal{V})$. This scaling can be seen from the fact that if we assume that all $c_{\mathbf{i},l}$ in $\mathcal{L}$ that have descendants in $\mathcal{L}$ have been removed (Lemma 2). Then $\#\mathcal{L} \leq \#\mathcal{V}$, since $\mathcal{V}$ contains either the elements of $\mathcal{L}$ or their children. Therefore, given the operations described per Particle Cell above, are all $\mathcal{O}(1)$ respect to $\#\mathcal{L}$, then the total number of operations should be $\mathcal{O}(\#\mathcal{V})$. Then also if the data structure has an $\mathcal{O}(1)$ over-head per Particle Cell concerning $\#\mathcal{V}$ then the memory overhead should also be $\mathcal{O}(\#\mathcal{V})$. Where we use the fact that the additional entries to $\mathcal{L}$

---

[1] In the implementation in the previous section, instead of introducing a new property, we extend the type property (with ascendant, ascendant neighbor, and propagate) their function is the same

```
Data: particle cell set $\mathcal{L}$
Result: particle cell set $\mathcal{V}$
Function pulling_scheme($\mathcal{L}$)
    /* Initialize status property of all particle cells in $\mathcal{L}$
       to ACTIVE                                                  */
    foreach $c_{\mathbf{i},l} \in \mathcal{L}$ do
        $status(c_{\mathbf{i},l}) \leftarrow ACTIVE$
    end
    $l_c \leftarrow l_{max}$
    $\mathcal{T}_{next} \leftarrow \emptyset$  Temporary particle cell sets for propogating individual solutions;
    $\mathcal{T}_{current} \leftarrow \emptyset$
    /* Loop over the resolution levels, from finest to
       coarsest (from the maximum level $l$)                      */
    while $l_c > l_{min}$ do
        /* Loop over all particle cells in $\mathcal{L}$ at level $l_c$    */
        foreach $c_{\mathbf{i},l} \in \mathcal{L} : l(c_{\mathbf{i},l}) == l_c$ do
            if $status(c_{\mathbf{i},l}) == ACTIVE$ then
                If particle cell is still active, add to set and propogate solution;
                $\mathcal{V} \leftarrow \{c_{\mathbf{i},l}, \mathcal{V}\}$
                propogate_individual_solution_for_level($c_{\mathbf{i},l}$,$\mathcal{L}$,$\mathcal{T}_{next}$,$\mathcal{V}$,$l_c$)
            if $status(c_{\mathbf{i},l}) \neq INACTIVE$ then
                set_parents_inactive($c_{\mathbf{i},l}$, $\mathcal{L}$)
        end
        foreach $c_{\mathbf{i},l} \in \mathcal{T}_{current} : l(c_{\mathbf{i},l}) == l_c$ do
            propogate_individual_solution_for_level($c_{\mathbf{i},l}$, $\mathcal{L}$, $\mathcal{T}_{next}$,$\mathcal{V}$,$l_c$)
        end
        $\mathcal{A}_{current} \leftarrow \mathcal{T}_{next}$
        $\mathcal{A}_{next} \leftarrow \emptyset$
        $l_c - -$  Level done, move to next;
    end
    return $\mathcal{V}$
```

Algorithm 5: Generating a optimal valid Particle Cell set $\mathcal{V}$ from the Local Particle Cell set $\mathcal{L}$

```
Function propogate_individual_solution_for_level($c_{\mathbf{i},l}$,$\mathcal{L}$,$\mathcal{T}$,$\mathcal{V}$,$l$)
    temp ← FALSE
    foreach $c^*_{\mathbf{i},l} \in \mathcal{V}^*(c_{\mathbf{i},l}) : l(c^*_{\mathbf{i},l}) == l$ do
        if $c^*_{\mathbf{i},l} \notin \mathcal{V}$ then
            $\mathcal{V} \leftarrow \{c^*_{\mathbf{i},l}, \mathcal{V}\}$
            set_parents_propogate($c^*_{\mathbf{i},l}, \mathcal{L}$)
            temp ← TRUE
    end
    if temp == TRUE then
        If the solution adds new elements, add to temporary set to be propogated to next level;
        $\mathcal{T} \leftarrow \{c_{\mathbf{i},l}, \mathcal{T}\}$
```

Algorithm 6: Add those Particle Cells in $\mathcal{V}^*(c_{\mathbf{i},l})$ at level l

```
Function set_parents_inactive($c_{\mathbf{i},l}$,$\mathcal{L}$)
    $l_p \leftarrow l(c_{\mathbf{i},l})$
    $\mathbf{i}_p \leftarrow \mathbf{i}(c_{\mathbf{i},l})$
    while $l_p > l_{min}$ do
        $l_p \leftarrow l_p - 1$
        $\mathbf{i}_p \leftarrow \lfloor \frac{\mathbf{i}_p}{2} \rfloor$
        if $\mathbf{c}_{\mathbf{i}_p,l_p} \in \mathcal{L}$ then
            if $status(\mathbf{c}_{\mathbf{i}_p,l_p}) == INACTIVE$ then
                Has already been set to in-active, so parents have already been checked;
                BREAK;
            else
                $status(\mathbf{c}_{\mathbf{i}_p,l_p}) \leftarrow INACTIVE$
    end
```

Algorithm 7: Sets the status of all parent Particle Cells of a seed $c_{\mathbf{i},l}$ in $\mathcal{L}$ to INACTIVE

```
Function $set\_parents\_propogate(c_{\mathbf{i},l},\mathcal{L})$
    $l_p \leftarrow l(c_{\mathbf{i},l})$
    $\mathbf{i}_p \leftarrow \mathbf{i}(c_{\mathbf{i},l})$
    while $l_p > l_{min}$ do
        $l_p \leftarrow l_p - 1$
        $\mathbf{i}_p \leftarrow \lfloor \frac{\mathbf{i}_p}{2} \rfloor$
        if $\mathbf{c}_{\mathbf{i}_p,l_p} \in \mathcal{L}$ then
            if $status(\mathbf{c}_{\mathbf{i}_p,l_p}) \neq ACTIVE$ then
                Has already been set to in-active, so parents have already been checked;
                BREAK;
            else
                $status(\mathbf{c}_{\mathbf{i}_p,l_p}) \leftarrow PROPOGATE$
    end
```

Algorithm 8: Sets the status of all parent Particle Cells of a boundary or filler Particle Cell in $\mathcal{L}$ to PROPOGATE

that do not feature in $\mathcal{V}$ must be a parent of a node in $\mathcal{V}$ and thus represents a constant cost (based on the isotropic neighborhood restricted one level change for neighbors).

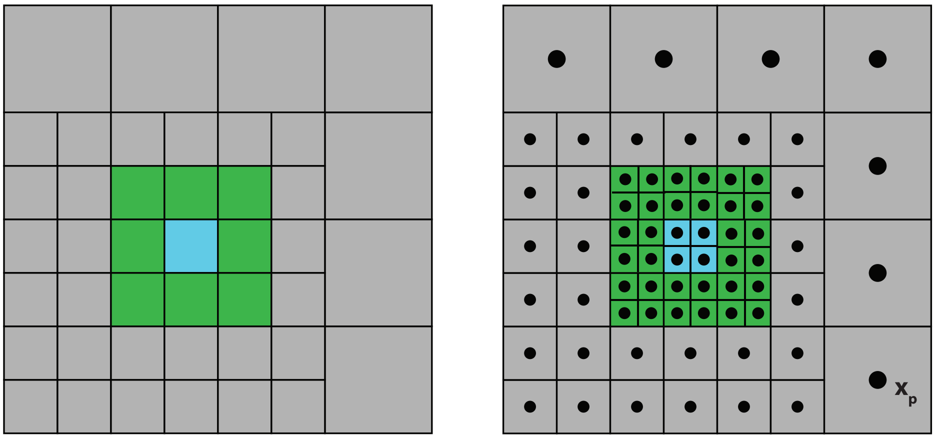
In the worst case $\#\mathcal{V} = \frac{N}{2^d}$ (where d is the dimension and N is full maximal sampling) if we assume that Lemma 3 is being used, or $\#\mathcal{V} = N$ if it is not. Therefore, worst-case scales in both as $\mathcal{O}(N)$.

5.5 Particle sampling

As in the 1D case, the last step given $\mathcal{V}$, is to determine the particle locations and sample them forming $\mathcal{P}^*$ and the APR. In general dimension, we take the identical approach to 1D 4.2.4. The set of points in $\mathcal{P} = \{\mathbf{x}_p\}_{p=1}^{N_p}$ are chosen such that for each Particle Cell $c_{\mathbf{i},l} \in \mathcal{P}$ a particle p is added to $\mathcal{P}$ as

$$\mathbf{x}_p(c_{\mathbf{i},l}) = \{\frac{\Omega^*}{2^l}(i_k + 1/2)\} \tag{5.73}$$

for $i_k = 1,..,d$ and $N_p = \#\mathcal{V}$. The function, is then sampled at locations $f_p = f(\mathbf{x}_p)$ to form $\mathcal{P}^* = \{f_p\}_{p=1}^{N_p}$. Such a sample satisfies the requirement that $\#(\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))) > 0$. Figure 5.3, shows an example of $\mathcal{V}_n$ on the *left* and then $\mathcal{V}$ and $\mathcal{P}$ on the (*right*). $\#(\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y})))$ is in all cases greater than one, with maximum resolution areas, producing a local grid identical to a pixel image representation. If different constraints on the reconstruction



$\mathcal{V}_n$ $\{\mathcal{V},\mathcal{P}\}$

Figure 5.3: *Left* the nOVPC $\mathcal{V}_n$ generated using $\mathcal{L}_n$ and the corresponding $\mathcal{V}$ and particle sampling $\mathcal{P}$ (*right*).

function are required, i.e. a different number of particles, or different layout, these could also be used. Here, we again present the simplest case.

5.5.1 APR as $\{\mathcal{V}_n, \mathcal{P}^*\}$

From Figure 5.3 and Lemma 3, there is a redundancy in directly storing $\mathcal{V}$, instead $\mathcal{V}_n$ could be stored along with each $t(c_{\mathbf{i},l})$ for each cell (w.r.t $\mathcal{L}_n$). In this way, the particles could be sampled from $\mathcal{V}_n$ using

$$\mathbf{x}_p(c_{\mathbf{i},l}) = \begin{cases} \prod_{\mathbf{i}}\{\frac{\Omega^*}{2^l}(i_k + 1/4), \frac{\Omega^*}{2^l}(i_k + 3/4)\} & t(c_{\mathbf{i},l}) = \{1,2\} \\ \{\frac{\Omega^*}{2^l}(i_k + 1/2)\} & t(c_{\mathbf{i},l}) = 3 \end{cases} \tag{5.74}$$

where then the locations of $\mathcal{P}$ are still implicit now from $\mathcal{V}_n$ and their type $t(c_{\mathbf{i},l})$. This representation results in lower memory overhead, but at the cost of complexity. Again, preference depends on the use-case, but as a solution to the **RC**, I believe the more costly, but simpler combination of $\{\mathcal{V},\mathcal{P}^*\}$ is preferable. (However, we use this form for file storage of the APR).

5.5.2 Optimality

For the 1D case, we introduced the concept of $\mathcal{P}$ being 'optimal' for $R^*(y)$, and hence $\mathcal{V}$. We can extend this concept by considering a similar integral where a sampling for a given resolution function $R(\mathbf{y})$ that satisfies

$$\#\mathcal{P} = \int_{\Omega} \frac{1}{R(\mathbf{y})^d} d\mathbf{y} \tag{5.75}$$

and $\#(\mathbf{y} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))) > 0$ for all $\mathbf{y} \in \Omega$. Again, intuitively, if we consider $\frac{1}{R(y)}^d$ as the point-wise required density (defined now for the 'hyper-volume' dependent on the dimension d), then again ignoring edge effects, this means that satisfying 5.75 leads to this density being everywhere exactly realized.

So again if we consider, the integral 5.75 for the implied resolution function $R^*(\mathbf{y})$, as

$$\int_{\Omega} \frac{1}{R^*(\mathbf{y})^d} dy = \int_{\Omega} \frac{1}{\left(\sum_{c_{\mathbf{i},l} \in \mathcal{V}} \phi(\mathbf{y}, c_{\mathbf{i},l}) \frac{\Omega}{2^l}\right)^d} d \tag{5.76}$$

$$\begin{aligned} y & \\ &= \sum_{c_{\mathbf{i},l} \in \mathcal{V}} 1 \\ &= \#\mathcal{V} = \#\mathcal{P} \end{aligned} \tag{5.77}$$

as required, and therefore $\mathcal{P}^*$ is optimal in the sense of 5.75. Hence, the Pulling Scheme in addition to providing an optimal Implied Resolution Function also provides an inherent 'optimal' sampling in general dimension.

5.5.3 Integral neighborhood optimization

From Figure 5.3, we observe that a single $c_{\mathbf{i},l}$ results in a large, high-resolution area in the solution. If we instead take $\mathcal{V}_n$ and create $\mathcal{V}$ in the following way

$$\mathcal{V}_i = \left\{ c_{\mathbf{i},l} \in \mathcal{C} \,\middle|\, (c_{\mathbf{i},l} \in \mathcal{V}_n, t(c_{\mathbf{i},l}, \mathcal{L}_n) > 1) \,\text{or}\, \big(c_{\mathbf{i}/2,l-1} \in \mathcal{V}_n, t(c_{\mathbf{i}/2,l-1}, \mathcal{L}_n) = 1\big) \right\} \tag{5.78}$$

where now boundary Particle Cells are also kept at their original resolution, then, if we use the alternative neighborhood of

$$\mathcal{N}(\mathbf{y}, R(\mathbf{y}))_i = \{\mathbf{x} \in \Omega : |(\mathbf{y} - \mathbf{x})| \int_0^1 \frac{1}{R(\mathbf{y} + s(\mathbf{x} - \mathbf{y}))} ds \leq 1\} \tag{5.79}$$

for the representation of the function as in 5.1, instead of the isotropic neighborhood, then $R^*(\mathbf{y}, \mathcal{V}_i)$ will also satisfy the Reconstruction Condition 5.2. This then results in a smaller $\mathcal{P}^*$ as shown in Figure 5.4. In practice this

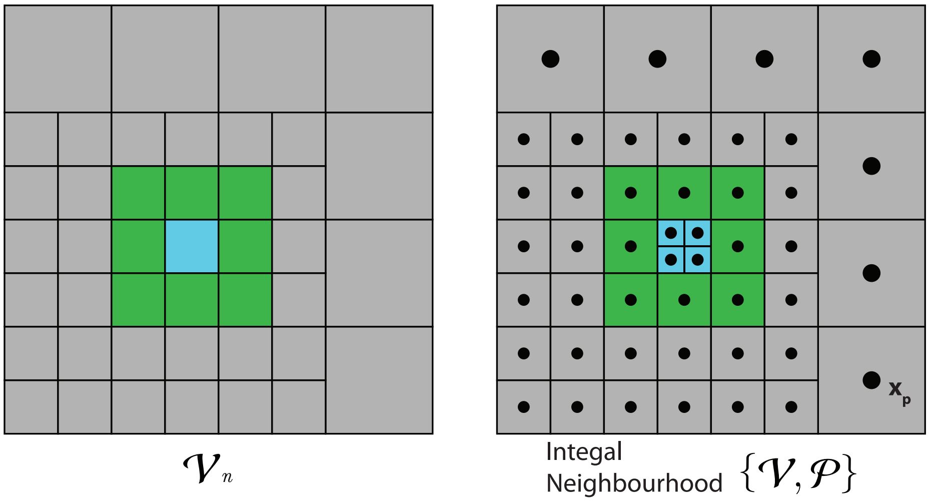


Figure 5.4: *Left* the nOVPC $\mathcal{V}_n$ generated using $\mathcal{L}_n$ and the corresponding $\mathcal{V}$ and particle sampling $\mathcal{P}$ (*right*) for the integral neighborhood optimization that has been used in Chapter 6 and Chapter 7

results in a $\approx 10 - 30\%$ reduction in $\#\mathcal{V}$, and is used in Chapter 6 and Chapter 7 below. However, although useful for single time-points, I believe again the isotropic neighborhood may be superior for satisfying the **RC** especially once time is integrated.

As in the isotropic neighborhood case, we can sample directly using $\mathcal{V}_n$, with only slight adjustment

$$\mathbf{x}_p(c_{\mathbf{i},l}) = \begin{cases} \prod_{\mathbf{i}} \{\frac{\Omega^*}{2^l}(i_k + 1/4), \frac{\Omega^*}{2^l}(i_k + 3/4)\} & t(c_{\mathbf{i},l}) = \{1\} \\ \{\frac{\Omega^*}{2^l}(i_k + 1/2)\} & t(c_{\mathbf{i},l}) = 2, 3 \end{cases} \tag{5.80}$$

Fulfillment of Reconstruction Condition

Now we briefly show that Reconstruction Condition is satisfied for the integral neighborhood definition and $R^*(\mathcal{V}_i, \mathbf{y})$.

We have the integral interaction neighborhood

$$\mathcal{N}(\mathbf{y}, R(\mathbf{y}))_i = \{\mathbf{x} \in \Omega : |(\mathbf{y} - \mathbf{x})| \int_0^1 \frac{1}{R(\mathbf{y} + s(\mathbf{x} - \mathbf{y}))} ds \leq 1\} \tag{5.81}$$

and show that if we are using the local resolution estimate $L(\mathbf{y}) = \frac{E\sigma(\mathbf{y})}{|\nabla f(\mathbf{y})|}$ that this neighborhood guarantees satisfaction of the Reconstruction Condition 5.1, given that $R(\mathbf{y}) \geq L(\mathbf{y})$ and the assumption on the Local Intensity Scale $\sigma(\mathbf{y})$ being sufficiently smooth over the integral path that $\sigma(\mathbf{y}) \approx \sigma(\mathbf{x})$ can be used.

Starting from the following bound as presented above,

$$\epsilon(\mathbf{y}) \leq \sum_{p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} |(\mathbf{y} - \mathbf{x}_p)| \int_0^1 |\nabla f(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y}))| ds \xi_p(\mathbf{y}) \tag{5.82}$$

which we wish to satisfy the Reconstruction Condition, so

$$E\sigma(\mathbf{y}) \geq \sum_{p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} |(\mathbf{y} - \mathbf{x}_p)| \int_0^1 |\nabla f(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y}))| ds \xi_p(\mathbf{y}) \tag{5.83}$$

which we can re-write as

$$\frac{1}{E\sigma(\mathbf{y})} \sum_{p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} |(\mathbf{y} - \mathbf{x}_p)| \int_0^1 |\nabla f(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y}))| ds \xi_p(\mathbf{y}) \leq 1 \tag{5.84}$$

and substituting for $L(\mathbf{y})$ and assuming $\sigma(\mathbf{y})$ is $O(1)$ over the interval gives

$$\sum_{p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} |(\mathbf{y} - \mathbf{x}_p)| \int_0^1 \frac{1}{L(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y}))} ds \xi_p(\mathbf{y}) \leq 1 \tag{5.85}$$

now given our reconstruction kernel conditions, this will hold if for every point,

$$|(\mathbf{y} - \mathbf{x}_p)| \int_0^1 \frac{1}{L(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y}))} ds \leq 1 \tag{5.86}$$

now given the assumption that $R(\mathbf{y}) \leq L(\mathbf{y})$ then the above will hold if the following also holds

$$|(\mathbf{y} - \mathbf{x}_p)| \int_0^1 \frac{1}{R(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y}))} ds \leq 1 \tag{5.87}$$

which is the integral interaction neighborhood stated above.

5.6 Technical additions

Here we discuss two additional issues that are useful for the following chapters. First, we discuss possible choices of reconstruction schemes for f from the APR. Second, we discuss the APR graph, which is used later for processing.

5.6.1 Function reconstruction and interpolation

In the discussions in this and the previous chapter, we have not specified $\xi_p(\mathbf{y})$. Instead, just the two conditions is must fulfill

$$\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \xi_p(\mathbf{y}) = 1, \tag{5.88}$$

$$\xi_p(\mathbf{y}) \geq 0.$$

In essence, any average over points within the neighborhood $\mathcal{N}(\mathbf{y}, R(\mathbf{y}))$ is valid. Here we briefly describe three different approaches that are used in the following chapters, that produce reconstructions $\hat{f}$ that satisfy the Reconstruction Condition. Many other possible approaches exist, including using B-Splines or Wavelets for reconstruction, however, we stick to the following three simple cases here. Also, Chapter 9 discusses conditions for higher order reconstruction, these can also be used for the APR as described above.

Piecewise constant reconstruction

This first approach, is practically, the most simple and efficient. A piecewise constant reconstruction $\hat{f}_{pc}$ that satisfies 5.88 can be constructed as

$$\hat{f}_{pc}(\mathbf{y}) = \sum_{c_{\mathbf{i},l} \in \mathcal{V}} f_p \phi(\mathbf{y}, c_{\mathbf{i},l}) \tag{5.89}$$

where $\phi(\mathbf{y}, c_{\mathbf{i},l})$ is defined as in 5.25. Due to its simple structure, 5.89 can be very efficiently implemented and has low computational cost. Despite its simplicity, it seems to produces subjectively 'good' reconstructions for visualization purposes. Because of these properties, we use it as the default reconstruction throughout the rest of this work. The draw back of this approach is not a 'smooth' reconstruction.

Smooth reconstruction

Instead, in the second approach, smooth reconstructions $\hat{f}_s$ can be used by utilizing a kernel function $\psi(\mathbf{x}) \geq 0$ in the following way

$$\hat{f}_s(\mathbf{y}) = \frac{\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} f_p \psi(\mathbf{x} - \mathbf{y})}{\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \psi(\mathbf{x} - \mathbf{y})}. \tag{5.90}$$

Such smooth reconstructions could be useful for visualization purposes, when piecewise constant 'artifacts' may not be wanted, or for processing applications requiring a smooth representation.

Worst-case reconstruction

For the analysis below, it is useful to be able to create the worst-possible reconstruction that satisfies 5.88, so we can show empirically that the Reconstruction Condition holds. If we consider any point $\mathbf{y} \in \Omega$, let $f_{min} = \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))}(f_p)$ and $f_{max} = \max_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))}(f_p)$, then any reconstruction satisfying 5.88 follows

$$f_{min} \leq \hat{f}(\mathbf{y}) \leq f_{max} \tag{5.91}$$

therefore, we define the minimum $\hat{f}_{min}$ and maximum $\hat{f}_{max}$, worst case reconstructions as

$$\hat{f}_{min}(\mathbf{y}) = \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))}(f_p), \tag{5.92}$$

and

$$\hat{f}_{max}(\mathbf{y}) = \max_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))}(f_p) \tag{5.93}$$

which represent upper and lower bounds on any reconstruction.

5.6.2 APR particle graph

Many processing tasks on images require the formulation of pixel images as a graph, for example, graph-cut methods [21]. Pixels are set as the nodes, and edges are created between adjacent pixels. Although the APR has changing resolution across the domain, a similar symmetric particle graph can be constructed from the APR by using adjacent particle cells, or formally, an integral interaction neighborhood. Figure. 5.5 (left), shows such a graph restricting neighbors across the faces of particle cells, analogous to a Von Neumann (or face-connected) neighborhood. In a classical pixel image graph, each node (pixel) would have the same number of neighbors. However, due to the adaptive sampling in the particle graph, the number of neighbors can vary as the resolution adapts. However, the maximum number and the minimum number of neighbors is bounded. The minimum number of neighbors is $2d$, as in the comparable pixel graph, and the maximum $n2^d$ where d is the dimension (However, empirically in 3D benchmark data the average number of neighbors is less than 6.3).

In the face connected particle graph is done on a given APR across neighboring Particle Cells, particle p_0 and p_1 will be neighbors if the line integral of the inverse of the Implied Resolution Function is below a certain threshold

$$|(\mathbf{x}_{p_1} - \mathbf{x}_{p_0})| \int_0^1 \frac{1}{R^*(\mathbf{x}_{p_0} + s(\mathbf{x}_{p_1} - \mathbf{x}_{p_0}))} ds \leq \frac{1}{3}\sqrt{9 + d - 1} \tag{5.94}$$

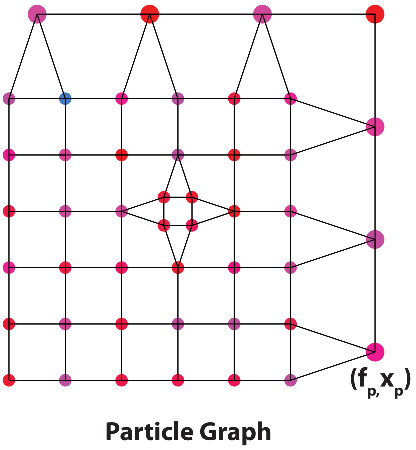


Figure 5.5: The APR paritcle graph, shown in 2D. This aligns with connecting the particles that are in face-connected Particle Cell neighbours of the current Particle Cell.

where d is the dimension. In 1D this bound is 1, and ≈ 1.054 in 2D and ≈ 1.105 in 3D. Particle neighbors in the particle graph can be interpreted as those particle pairs for which the difference between the two value will be approximately $\frac{E}{\sigma(\mathbf{y})}$ where y is a position on the line segment between the points. Note, that these points in dimension greater than 1 exceed the integral neighbor bound by a small factor. If it is wished that the neighbours be guarantee a Reconstruction Condition $\hat{E}$, the APR can be construction with $E = \frac{3\hat{E}}{\sqrt{9+d-1}}$. Due to the isotropic nature of the Implied Resolution Function, this will guarantee that extending the neighborhood, the particles on the edge can be used for any reconstruction, within E.

5.7 Summary and main points

In this chapter, we have introduced the Adaptive Particle Representation (APR) for arbitrary dimension and provided the technical details and proofs for the main theoretical and algorithmic results in this Thesis that were described using 1D in Chapter 4.

First, we stated the definition of the APR and the main results from the previous section in a general dimension setting. Then we derived the Resolution Bound. Following this, we provided the formal definitions of Particle Cells, Implied Resolution Functions, and the Local and Optimal Valid Particle Cell sets. Given these definitions, we then formally stated and proved the theorems and lemmas on which the formulation of the APR relies. Based on these results we gave a generic algorithmic description of the Pulling Scheme and potential optimizations for particle sampling. Lastly, we discussed three different reconstruction functions and the APR particle graph.

An attempt to provide a general formulation was made in this section, to allow future development and extensions to be able to utilize the above results. We provide steps in this direction with the discussions of extensions in Chapter 9 (space) and Chapter 10 (time).

The ideas and methods given in the last two sections have many similarities with existing techniques described in Chapter 3 or elsewhere, we discuss this for the different features of the APR in Chapter 8.

Summary of the chapter

- Formulated APR and stated main results in general dimension
- Derived the Resolution Bound in general dimension
- Formally introduced Particle Cells and their Implied Resolution Functions
- Stated and proved the main theorems and lemmas used to form the APR
- Provided a generic description of the Pulling Scheme
- Discussed particle sampling including optimizations
- Detailed three reconstruction approaches used in this work
- Introduced the APR particle graph

6 APR Validation

Contents

6.1 1D benchmarks 115

6.1.1 Implementation 115

6.1.2 1D example 115

6.1.3 Reconstruction Condition 117

6.1.4 Numeric vs. symbolic gradient 119

6.1.5 Number of particles 120

6.1.6 Gradient 120

6.1.7 Discontinuities 121

6.1.8 Summary 122

6.2 3D florescence image implementation 122

6.2.1 Gradient estimation $|\nabla I|$ 123

6.2.2 Local Intensity Scale σ 124

6.2.3 Intensity estimation I_p 125

6.2.4 Reconstruction methods 125

6.2.5 Pipeline and parameters 125

6.3 3D synthetic data 126

6.3.1 Object function 126

6.3.2 Image formation 127

6.3.3 Summary 128

6.4 APR properties benchmarks 129

6.4.1 Noise-free Reconstruction Condition 130

6.4.2 Noise corrupted Reconstruction Condition 132

6.4.3 Increasing information content 135

6.4.4 Increasing image size 136

6.4.5 Increasing sampling rate 136

6.5 APR performance benchmarks 137

6.5.1 Computational Ratio benchmark data 139

6.5.2 Exemplar benchmark data 139

6.5.3 Data structures 140
6.5.4 Execution time 141
6.5.5 Memory Cost 148
6.5.6 Storing the APR 148
6.6 Summary and main points 150
6.6.1 Limitations of the benchmark results 151
6.6.2 Reflection on Representation Criteria 152

In this chapter, we empirically test and explore the properties of the APR presented in the previous two chapters. We do this using two implementations. The first, a simple 1D implementation for analytically defined functions written in Matlab, and second, a shared memory parallel 3D implementation designed for 3D LSFM data.

Using the 1D implementation, we first give examples of an APR with a simple test case and explore the effect of changing the relative error E (Where we use a constant local intensity scale). Doing this, we evaluate whether the APR obeys the Reconstruction Condition. Following, we then look at the reconstruction error of the gradient of the function and then explore how the APR handles discontinuous functions.

The introduction and results for the 3D implementation a separated into multiple sections. First, we outline the implementation and algorithmic choices used in constructing the APR, where we use a varying Local Intensity Scale, based on an estimate of the local range of the function (Using the statistical definition of range). Second; we describe the process of how synthetic data is created for the benchmarks. Then in the third and fourth section, we provide the benchmark results using this implementation and synthetic data. In the first benchmarks, we explore the properties of the APR, regarding the level of information content, the Reconstruction Condition, and image size. Then we introduce two benchmark datasets that we use as a proxy for different information content and image size ratios. The first set of datasets is a collection of images and APR with a fixed ratio of information content to image size. The second dataset is a collection of nineteen real LSFM images of varying size, specimen, and label. We also introduce the APR data structures that are used in 3D. Lastly, in the second set of benchmarks, we explore the computational cost and scaling of forming the APR and the memory cost of storing the APR in file storage.

6.1 1D benchmarks

In this section, we explore and test the APR using 1D functions. We use the most simple case where the Local Intensity Scale $\sigma(y)$ is a constant. First, we will briefly describe the algorithms that were used and then follow with a discussion of various results.

6.1.1 Implementation

The results in this section were produced using scripts in Matlab. The code takes a function f over a fixed domain Ω that can be queried at any point $y \in \Omega$. Given a user-set relative error E and the input function, the APR is then be computed.

The Pulling Scheme was implemented as outlined in 4.2.3 and A.4.1, using storage of a full representation of the Particle Cell set $\mathcal{C}$ from l_{min} to $l_{max} - 1$, where $\frac{\Omega}{2^{l_{max}}}$ represents the smallest distance between sampled particles. The equivalence optimization 4.2.3 was used. However, the integral neighborhood optimization was not used. l_{max} was set by finding the numerical maximum of the absolute value of the gradient, computed using central differences, and finding its associated particle cell level using $l = \max(l_{min}, \lfloor \log_2(\frac{\Omega}{L(y)}) \rfloor))$ and l_{min} was set to one. The natural Local Particle Set, $\mathcal{L}_n$ was then calculated by iterating over the domain at a sampling defined by $\frac{\Omega}{2^{l_{max}}}$. $\mathcal{L}_n$ was created by calculating $L(y)$ and then determining the associated Particle Cells setting and then setting the values in the $\mathcal{C}$ structure to one (For more details see the description of the 3D pipeline in A.4.1).

We tested this pipeline, using a numeric, and symbolic version. In the symbolic version, all the function and gradient calls were symbolically evaluated. In the second, the numeric version, the function sampled at a spacing $\frac{\Omega}{2^{l_{max}}}$ was the input of the function. $L(y)$ was computed either using central differences for the numerical version. The pulling scheme was then used to calculate $\mathcal{V}_n$ and from this $\mathcal{V}$. Lastly, the particle set $\mathcal{P}^* = \{f_p\}$ was formed by sampling the function. As particle locations do not align with the sampling used in the previous steps when using the numerical computation, linearly interpolated values were used. Unless explicitly stated, all results are shown for the numerical version.

6.1.2 1D example

Here, we explore the APR for a simple 1D function for a function composed of a narrow negative and broad positive Gaussian function. Figure 6.1, shows the APR represented as particles at f_p in green, and a piecewise linear interpolation

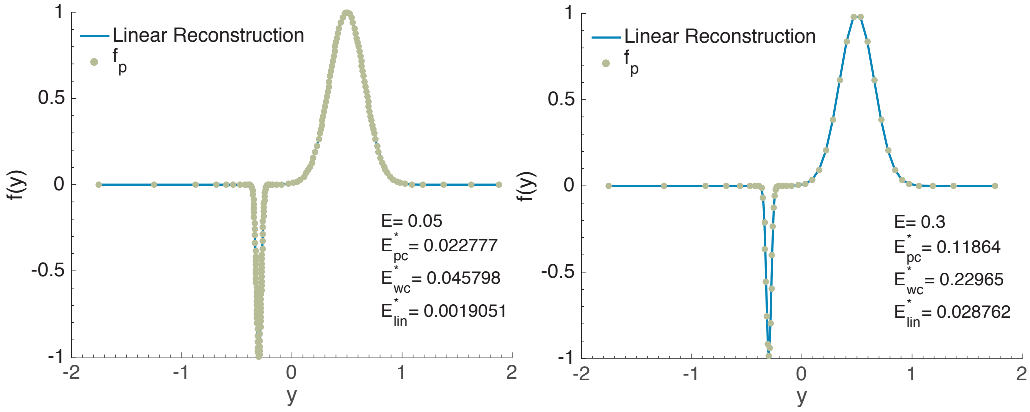


Figure 6.1: The APR with $E = 0.05$ (*left*) and $E = 0.3$ (*right*) for $f(y) = e^{\frac{-(x-0.5)^2}{0.05}} - e^{\frac{-(x+0.3)^2}{0.001}}$ with $\sigma(y) = 1$ on the domain $\Omega = [-2, 2]$. The observed reconstruction errors (normalized infinity norm) are given inset for E^*_{pc} a piecewise constant reconstruciton, E^*_{wc} worst case reconstruction, and E^*_{lin} piecewise linear reconstruciton. For $E = 0.05$, $\#\mathcal{P}^* = 176$, and $l_{max} = 12$, for $E = 0.3$, $\#\mathcal{P}^* = 51$, and $l_{max} = 9$.

in blue, for a high relative error $E = 0.05$ and low relative error $E = 0.3$ (Function definition in caption). We only use this example here; however, the results are consistent across general differentiable functions that have been tried. From the two plots, we can see that the particles are adapting to the different length scales in the problem, having a low density of particles in the flat areas and resolution increasing near the two peaks. Further, we can see that the impact of increasing the relative error is an increase in the resolution in the already higher resolution areas. In the inset, we show the observed reconstruction errors E^* of the two APRs for a range of different reconstruction methods. We define the observed reconstruction error E^* for a set of points $\bar{x}$ as

$$E^* = \max_{x \in \bar{x}} \left(\frac{|\hat{f}(x) - f(x)|}{\sigma(x)} \right) \tag{6.1}$$

where $\hat{f}$ is the reconstructed value from the APR. A subscript is usually given to indicate which reconstruction method was used, and $\bar{x}$ is the set of all points sampled at a spacing of $\frac{\Omega}{2^{l_{max}}}$. For the 1D examples, we use three different constructions. E^*_{pc} is based on a piecewise constant nearest neighbor reconstruction $\hat{f}_{pc}$, E^*_{wc} is the worst-case taking the maximum reconstruction error for both $\hat{f}_{min}$ and $\hat{f}_{max}$ as described in 5.6.1, and E^*_{lin} is from a piecewise linear (between particles) reconstruction. Figure 6.2 shows the reconstructions for the three cases for the APR with $E = 0.3$. For the case of $E = 0.05$ the reconstructions, except the worst-case, are indistinguishable by eye from the

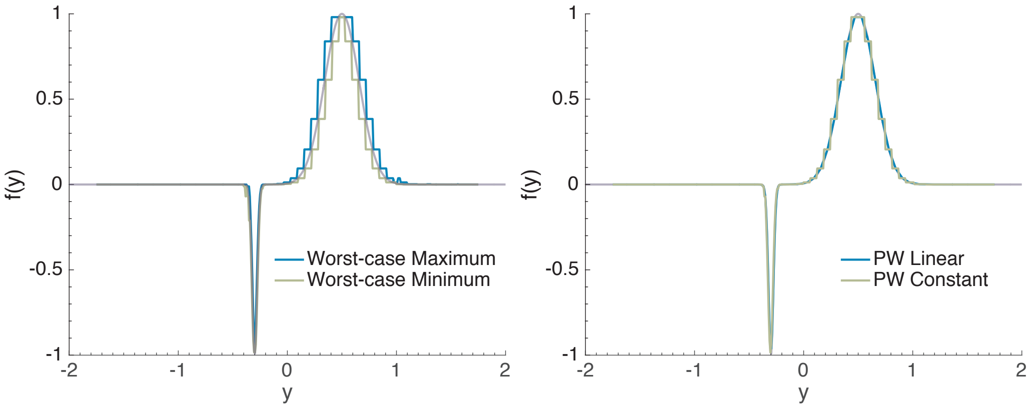


Figure 6.2: Examples of three different reconstruction methods for the $E = 0.3$ APR from Figure 6.1. The *left* plot shows the maximum $\hat{f}_{max}$ (light blue) and minimum $\hat{f}_{min}$ (grey green) worst-case reconstructions (5.6.1) with the original function plotted in transparent blue. The *right* plot shows piecewise constant interpolation $\hat{f}_{pc}$ (green) and piecewise linear inteprolation. The original function is plotted in transparent blue.

function. From theory, this observed reconstruction error should be less than or equal to E for all of these methods. Returning to the values in Figure 6.1, we see that this is the case. As expected, the worst-case reconstruction has the highest value, followed by the piecewise constant, and then piecewise linear reconstructions. Next, we show details of the APR formation, and how the change of resolution between the two relative error values arises. The increase in E, from 0.3 to 0.05, results in a scaling of the Local Resolution function $L(y)$, shifting it to a smaller value. The lower values then result in a more constrictive Resolution Bound resulting then in a smaller Implied Resolution Function. This is shown in Figure 6.3 where the Implied Resolution Function $R^*(y)$ (green) and the Local Resolution Estimate $L(y)$ (blue). However, across these figures, discerning the changes in resolution in high-resolution areas (small R^*) is difficult. However, this is easily done instead by directly visualizing the particle cell level l. Figure 6.4, shows the changes in resolution by particle cell level l for the two different relative errors. We note that the particle cell level l for the higher relative error $E = 0.05$ seems to be more responsive to the features of the function then $E = 0.3$.

6.1.3 Reconstruction Condition

Above we showed that the Reconstruction Condition holds for two values of relative error E. But, does this also hold for arbitrary values of E? To address this, we computed the APR and reconstruction errors E^* for 200 values from

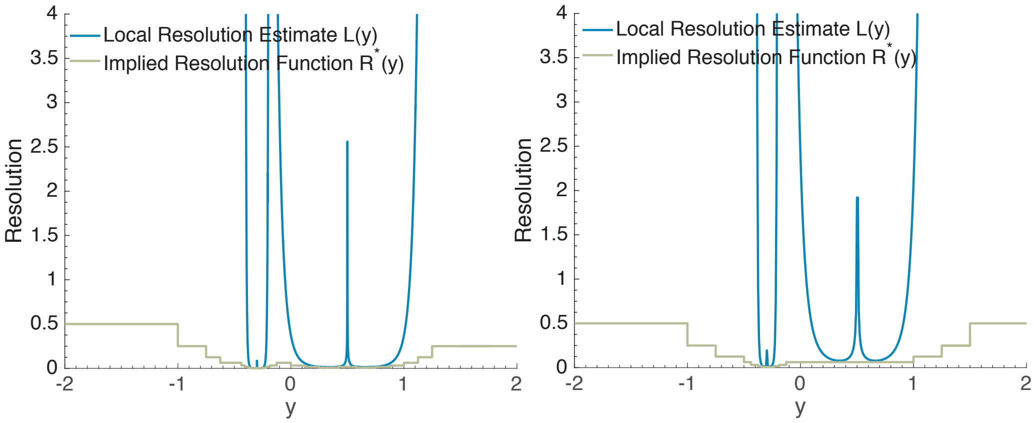


Figure 6.3: Local Resolution Estaimte $L(y)$ (blue) and Implied Resolution Function $R^{*}(y)$ (green) for the $E = 0.05$ (*right*) and $E = 0.3$ (*left*) examples from Figure 6.1.

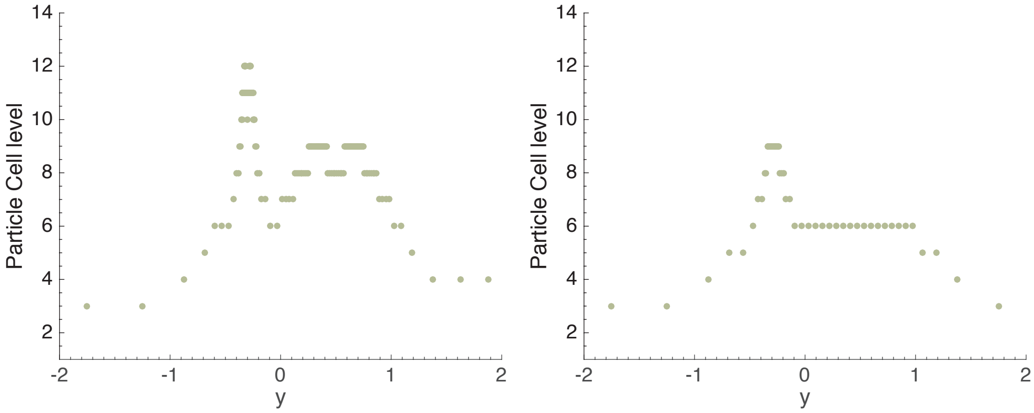


Figure 6.4: Particle Cell level l, for all $c_{\mathbf{i},l} \in \mathcal{V}$ for the $E = 0.05$ (*left*) and $E = 0.3$ (*right*) examples from Figure 6.1.

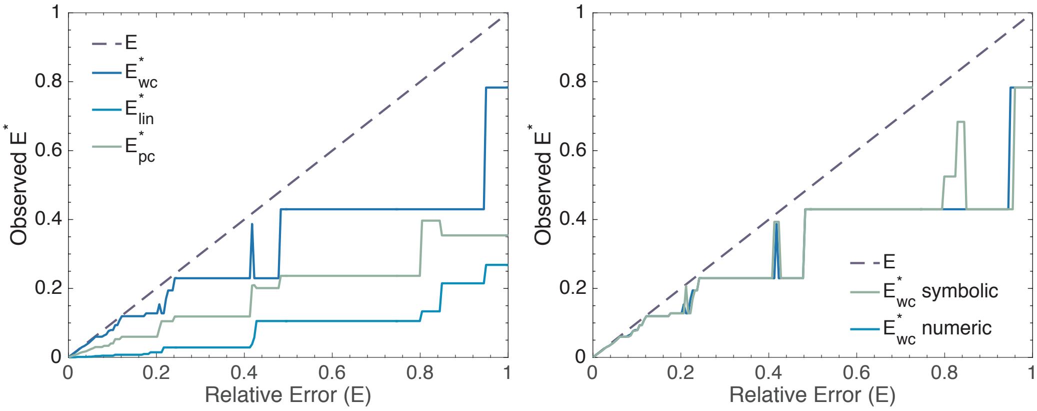


Figure 6.5: The observed reconstruction errors for the APR and function as in Figure 6.1 for a linear range of 200 values of relative error E from 0.001 to 1. In both plots the dotted dark blue line indicates $E^* = E$, the representing Reconstruction Condition that the APR reconstruction should be below. The *left* plot shows the observed reconstruction errors for worst case E^*_{wc} (blue), piecewise linear E^*_{lin} (light blue) and piecewise constant E^*_{pc} (green). The *right* plot shows the worst case reconstruction error E^*_{wc} when the gradient is computed analytically (green) and numerically using central differences (blue).

0.001 to 1 for the three reconstruction methods. The results are plotted in the *left* plot in Figure 6.5. For small values of E the reconstruction errors show a linear response to E, and for higher values show a piecewise constant response. Across all values, the worst-case reconstruction, as predicted, is the highest. Further, although it comes close to the bound, represented by the dotted line, it never crosses it. These results, therefore, confirm that the Reconstruction Condition holds across E for our test function.

6.1.4 Numeric vs. symbolic gradient

The derivation of the APR assumes full knowledge of the gradient of the function $\frac{\partial f}{\partial x}$. In 4.3.2, we briefly discussed some theoretical arguments on how errors in the gradient would affect the observed reconstruction error E^*. To test the impact of this, we compared the worst-case reconstruction error of the APR computed with exact knowledge of f through symbolic evaluation of $\frac{\partial f}{\partial x}$ and the numeric version computed from knowledge of f only at samples of distance $\frac{\Omega}{2^{l_{max}}}$. All previous results have been with the numeric version. The result is shown in the right plot of Figure 6.5. For small values of E the results appear identical, however, for a few points at higher E, there are some differences. Indeed, the reconstruction error for the numeric code is smaller, except at isolated points for E near 1. Arguably, since the bound only requires E be below the bound, the lower value results from more particles being used, and therefore the higher analytical solution is 'better'. However, in this example, the difference between the two regarding the number of particles was small (1-2).

6.1.5 Number of particles

Intuitively, we should expect that the smaller the E, the more particles that should be required to form the optimal solution to the Resolution Bound with Particle Cells, and therefore the number of particles should increase with increasing E. This is the case, and is shown in the *left* plot of Figure 6.6. The plot shows both the numeric and analytic version number of particles against the relative error E. Only one curve can be seen because the differences are indistinguishable when visualized this way. The plot shows that not only does the number of particles increase with decreasing E, that it does so in a non-linear way. To explore this, in the inset of Figure 6.6 *left* we show the same results in a log-log plot. We find what appears to be two different regimes, corresponding to linear regions in the log-log plot. In the figure, we also show linear fits for these two regions. For small values of E, the number of particles N_p, appears to scale like $E^{-0.86}$ and for higher values like $E^{-0.56}$.

6.1.6 Gradient

As briefly discussed in 4.3.6, satisfying the Reconstruction Condition, only guarantees the reconstruction of the function f at a specified relative error E, and does not bound the derivative. However, the error of the gradient should still scale with E. We empirically explore the gradients reconstruction error, defined as

$$E^{*}_{grad} = \max_{x \in \bar{x}} \left(\frac{|\frac{\hat{\partial f}}{\partial y}(x) - \frac{\partial f}{\partial y}(x)|}{\sigma_{grad}} \right) \tag{6.2}$$

where we set σ_{grad} to be equal to the maximum absolute value of the gradient across the interval. The normalization by the maximum absolute value of the gradient is to make the results comparable to the E bound for f. The results are shown in the *right* plot of Figure 6.6, where the gradient is computed using both 1st order, and 2nd order in h DC-PSE [115] derivative estimates. We find that in both cases as the error decreases in E. For the first order derivative, the error is above the bound set by E, however, for the higher order 2nd derivative, we see that the reconstruction error in the gradient is always below E.

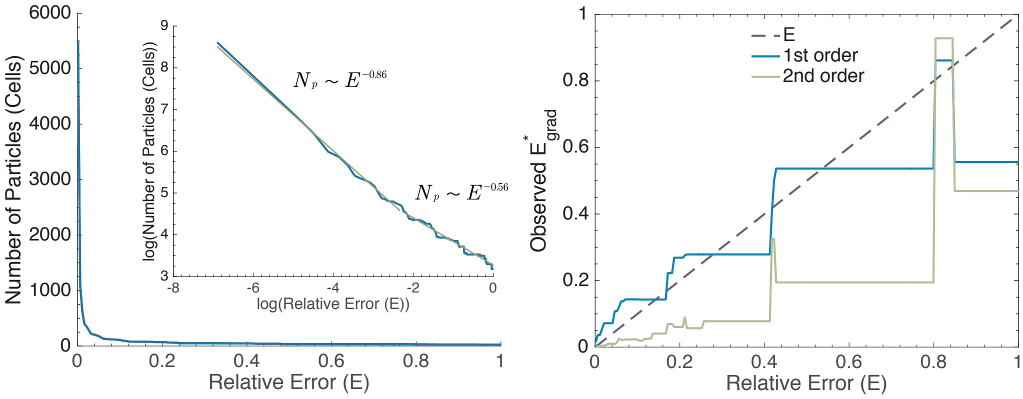


Figure 6.6: The *left* plot shows the number of particles ($N_p = \#\mathcal{V}$) for the APR and function as in Figure 6.1 for a linear range of 200 values of relative error E from 0.001 to 1. Inset is the same data (blue) on a log-log plot, with two linear fits (green). The first fit is for $E \leq 0.05$, with exponent -0.86 and R-Square: 0.994 and second for $E > 0.05$ with exponent -0.56 and R-Square: 0.975. The ***right*** plot shows the observed reconstruction error of the gradient computed on the same series of APRs. The observed reconstruction error of the gradient is the infinity norm of the gradient normalized by the maximum absolute value of the gradient. The dotted line shows the relative error bound, the light blue shows a first order gradient, and green second order gradient.

Later in Chapter9, we discuss, and show examples of, how using the same framework the APR can be extended to guarantee the observed reconstruction error E^*_{grad} in addition to E^*.

6.1.7 Discontinuities

Lastly, for the 1D case, we explore the case where f is no longer in C^1 and contains discontinuities. We do this by adding two Heaviside step functions to the previously used example from Figure 6.1. The existence of discontinuities violates the assumptions of the formulation of the APR. However, practically discontinuities can be handled when using the numerical version, given the introduction of a fixed maximum level l_{max} for the initial sampling. We do this by using sampling set by the previous example for the input $f\{\bar{x}\}$, but letting l_{max} for the APR be determined by the numerical computation of the derivative and $L(y)$. We show the resulting APR's for the same relative errors $E = 0.05$ and $E = 0.3$ in Figure 6.7, with the observed reconstruction errors again inset. We find that the two piecewise constant reconstruction methods still satisfy the Reconstruction Condition, but the worst-case method does not. The piecewise reconstruction methods meeting the bound is the result of only computing E^* at the sampling points given by $\bar{x}$, which coincides with the highest sampling distance in the APR. Therefore, at the high-resolution regions, the reconstruction is simply the particle values f_p for these meth-

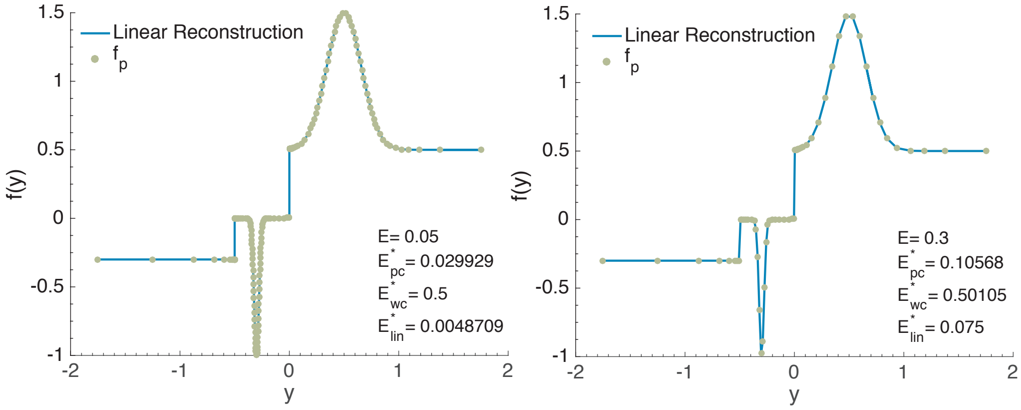


Figure 6.7: The APR with $E = 0.05$ (*left*) and $E = 0.3$ (*right*) for $f(y) = e^{\frac{-(x-0.5)^2}{0.05}} - e^{\frac{-(x+0.3)^2}{0.001}} + 0.5 *$ Heaviside$(x) - 0.3 *$ Heaviside$(-.5 - x)$ with $\sigma(y) = 1$ on the domain $\Omega = [-2, 2]$. The observed reconstruction errors (normalized infinity norm) are given inset for E^*_{pc} a piecewise constant reconstruciton, E^*_{wc} worst case reconstruction, and E^*_{lin} piecewise linear reconstruciton. For $E = 0.05$, $\#\mathcal{P}^* = 200$, and $l_{max} = 18$, for $E = 0.3$, $\#\mathcal{P}^* = 57$, and $l_{max} = 9$.

ods. However, the worst-case reconstruction effectively uses all points within $R^*(y)$. In this case, the reconstruction fails at the discontinuity. However, the same would occur for any discrete sampling across the discontinuity using an isotropic kernel with support greater than one point.

6.1.8 Summary

In the above, we have briefly shown that the results from the previous methods section hold, at-least for the basic noise-free example we have shown. We return to using this implementation in Chapter 8 in a noisy scenario for comparison of optimality results with wavelets. However, the results above are representative of all 1D functions I have explored in work not presented here. We explore issues regarding how the adaptation relates to information content, noise, and computational issues, data structures, and storage in the 3D case we present next.

6.2 3D florescence image implementation

In this section, we briefly outline how we have implemented the steps for forming the APR for noisy $3D$ fluorescent images. We make use of the optimizations for the integral neighborhood sampling (5.5.3) and equivalence optimization (4.2.3). For the Pulling Scheme, we use explicit storage of $\mathcal{C}$ as described in

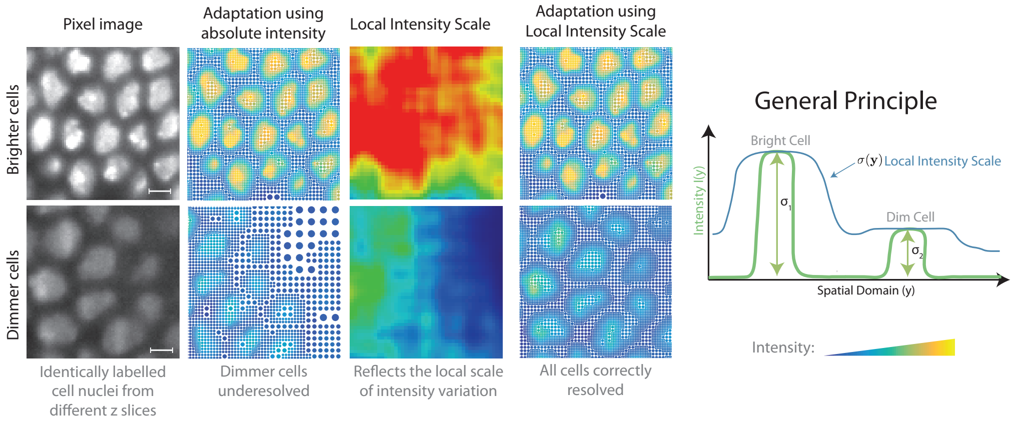

Pixel image
Adaptation using absolute intensity
Local Intensity Scale
Adaptation using Local Intensity Scale
Brighter cells
Dimmer cells
Identically labelled cell nuclei from different z slices
Dimmer cells underesolved
Reflects the local scale of intensity variation
All cells correctly resolved
General Principle
Bright Cell
σ(y) Local Intensity Scale
Dim Cell
Intensity I(y)
σ1
σ2
Spatial Domain (y)
Intensity:

Figure 6.8: *Left* compares the adaptive sampling of two regions of labeled cell nuclei in the same image stack (Dataset number 6 from Table A.1). One of the regions is brighter than the other (*left* panel). The *centre left* panel shows adaptive representations sampling based on the absolute intensity. The *right* panel shows adaptation using a Local Intensity Scale calculated from the image and shown in the *centre left* panel. The use of the Local Intensity Scale allows both regions to be correctly resolved (*right*). The schematic on the *right* shows the basic idea behind the Local Intensity Scale we use here. The local intensity scale should be slowly varying and reflect the range of intensities (from highest to lowest) of objects within a set length scale.

4.2.3 in 1D and given in with algorithm steps given in A.4.1. Here we provide an outline of the implementation choices that differ from the 1D case above. A.4.1 gives additional technical details for the steps. Note, in this section as we are now dealing with images, we will use I to represent the original noisy input image, instead of f as previously used. When implementing the APR for 3D LSFM data, three main choices had to be made. First, how to calculate the gradient magnitude $|\nabla I(\mathbf{y})|$, second, what form of Local Intensity Scale $\sigma(\mathbf{y})$ to use and how to calculate it, and last, how to sample the image intensity at particle locations $I_p = I(\mathbf{y}_p)$. All decisions have been made with the objective of meeting the Representation Criteria through optimizing both robustness to noise and computational efficiency.

6.2.1 Gradient estimation $|\nabla I|$

To calculate the gradient magnitude $|\nabla I|$ from the image we use smoothing cubic B-splines [129]. Smoothing cubic B-splines provide robust gradient estimation in the presence of noise. However, they require the setting of a smoothing parameter λ to be set according to the noise level. Further, we have implemented the fitting of the B-Splines using the recursive IIR approach [129]. Using the recursive approach provides a computational cost that is $\mathcal{O}(1)$ concerning λ, i.e. the computational cost is constant regarding a change in the

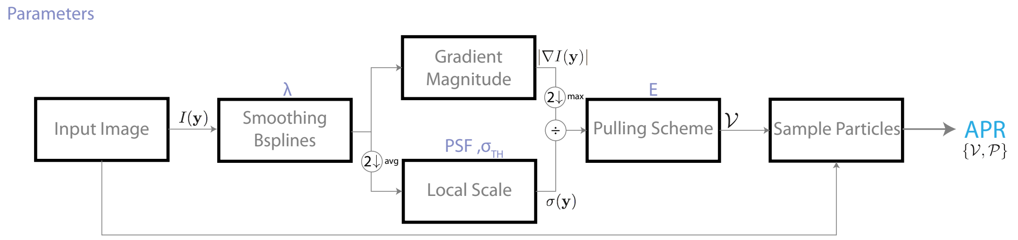


Figure 6.9: Flow chart showing APR pipeline for fluorescent images, first smoothing B-splines are fit to the image, then the gradient magnitude $|\nabla f|$ and local scale $\sigma(\mathbf{y})$, is computed. The Local Resolution Estimate $L(\mathbf{y})$ is then computed and used to construct the input for the Pulling Scheme that then computes the OVPC set $\mathcal{V}$. The particles are then sampled from the original image, forming the APR. Required parameters are given above the boxes in purple.

smoothing scale λ.

6.2.2 Local Intensity Scale σ

For the effective adaptation to the content of the LSFM data requires the addition of a non-constant Local Intensity Scale $\sigma(\mathbf{y})$. We illustrate this need in the *left* panel of Figure 6.8. It shows two regions of an LSFM image of cell nuclei that show a large difference in brightness, despite both showing comparable cells. The difference is a result of the varying local intensity scale in LSFM data that was discussed in detail in Chapter 2. In the absence of a dynamic Local Intensity Scale, only the bright region is correctly resolved, with the dim nuclei being under-sampled (or no adaptation for a lower constant scale). To correct for this, we use a spatially varying Local Intensity Scale that is a smooth estimate of the local range of the image as shown in a schematic on the *right* of Figure 6.8. This Local Intensity Scale compensates for the varying brightness levels across the image and allows for the adaptation to both bright and dim regions, as seen in the last two panels in the figure. The Local Intensity Scale for the two different regions is also shown, displaying the large difference between the two areas of the same image.

To compute and define a local range of intensity in the image requires the addition of a length scale. For this, we use the intrinsic length scale in the image from the image formation process. The smoothing window in each direction is set proportional to the average width of the point spread function (PSF) of the microscope used. Further, two minimum threshold parameters σ_{TH} are introduced to prevent the resolving of background noise that would occur in their absence (Figure A.4). As mentioned, for the Resolution Bound to hold, the Local Intensity Scale must be sufficiently smooth (See 4.3.5). Practically, we can not guarantee this condition for a nonconstant $\sigma(y)$. However, setting the window proportional to the PSF and calculating $\sigma(\mathbf{y})$ from a once down-sampled image satisfy this condition sufficiently for the Reconstruction Condition to hold empirically (see benchmark section below). The calculation requires a series of local mean window estimates. These are implemented using the ideas of integral images, also known as summed area tables [33]. Summed area tables are a recursive approach resulting in the computational cost to be independent of the length scale (window size) used. A detailed description of the Local Intensity Scale is given in A.4.1.

6.2.3 Intensity estimation I_p

Two methods are used to estimate intensities I_p from the image $I\{\bar{\mathbf{y}}\}$. Because the image is noisy, a direct evaluation of the closest pixel value no longer provides the best estimate of the noise-free intensity value at $\mathbf{x}_p$. For particles in particle cells at pixel resolution, the intensities are median filtered in each direction and then sampled. The use of a median filter is based on the edge preserving properties of the filter and that the high-resolution areas are localized near large gradients. For particles in a larger particle cells, i.e. with level $l > l_{max}$ we use the average intensity of the pixels contained in the particle cell for the value of I_p. This then allows for an adaptive estimate of the intensity I_p that uses the scale information inherent in $\mathcal{V}$. A more detailed description is given in A.4.1.

6.2.4 Reconstruction methods

For the comparison of the APR with images, a reconstruction method must be used. In 5.6.1 we discussed the reconstruction methods used in this section. Unless otherwise explicitly mentioned, it should be assumed that the piecewise constant reconstruction method is used. This was chosen for its computational efficiency, simplicity, and effectiveness.

6.2.5 Pipeline and parameters

A summary of the algorithmic steps required to form the APR from an input image $I\{\bar{\mathbf{y}}\}$ are shown in Figure 6.10. The parameters that must be set are shown in purple. These are the smoothing parameter λ for gradient estimation, the threshold parameters for the Local Intensity Scale σ_{TH}, the point speed function width PSF_w, and the desired relative error E. A detailed discussion of all parameters, their interpretation, and how they have been and can be, set is given in A.7. For all of the benchmarks given below, the parameters that

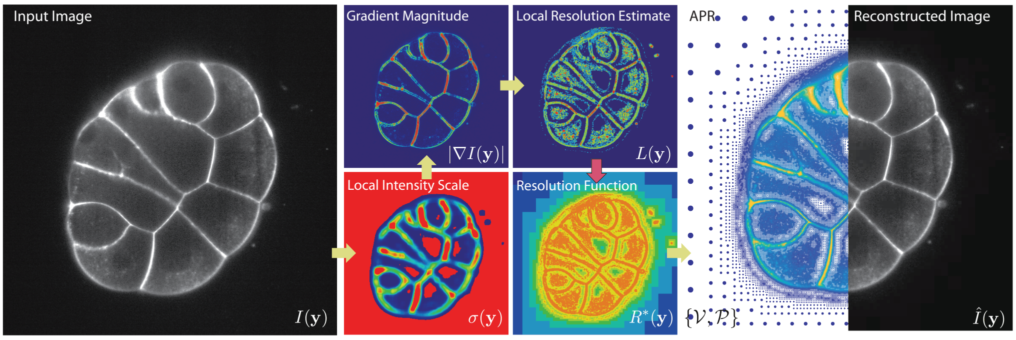


Figure 6.10: The figure shows the main steps in the pipeline for creating the APR using a 2D example image (Dataset number 10 Table A.1). First the Local Scale Function $\sigma(\mathbf{y})$ (Local Scale Function is red where the value is below the minimum threshold σ_{th}) and gradient magnitude $|\nabla I(\mathbf{y})|$ are calculated, and then combined to compute the Local Resolution Estimate $L(\mathbf{y})$ (See $*$). This is then used to form $\mathcal{L}$ and input to the Pulling Scheme, the *red* arrow, to form $\mathcal{V}$. The Implied Resolution function $R^*(\mathbf{y})$ is shown (See $*$). $\mathcal{V}$ is then used to define the particle locations and sample the function and create the APR *left* panel. The *left* side of the last panel shows the APR piecewise constant reconstructed image. (($*$) for both the Local Resolution Estimate and the Implied Resolution Function, the particle cell level has been interpolated to each pixel, to allow better visualization)

have been used are described in a section for each in A.9. To give the reader some intuition of the steps required to form the APR, Figure 6.10 shows the main steps for a single slice of an LSFM image.

6.3 3D synthetic data

To be able to test the properties of the APR for 3D LSFM data we use synthetically generated image data. We generate synthetic images following our Object function and image formation model described in 2.2.1 and 2.2.4. Synthetic data is used as it allows us to control image parameters, such as image size, content, and quality in addition to full knowledge of the noise-free, ground-truth image and Object function. We provide an overview below and give additional technical details in A.8. The synthetic image generation was implemented in C++ using the ArrayFire GPU library [12].

6.3.1 Object function

We follow the model of an LSFM image as discussed previously in 2.2.1. Where we define our Object function (ignoring time) defined on $\Omega \subset \mathbb{R}^3$ with M

objects as

$$O(x, y, z) = \sum_{i=1}^{M} O_i(x, y, z) \tag{6.3}$$

where the function is set to zero outside of Ω for simplicity. Each object is composed as

$$O_i(x, y, z) = B_i O^*(x - x_i, y - y_i, z - z_i) \tag{6.4}$$

where $O^*(x, y, z)$ is a piecewise constant function of compact support, that we call the template object, and B_i is a constant we call the brightness of object i. In all but one case, the template object used below is a sphere.

6.3.2 Image formation

Given a particular Object funtion, we form an image $I\{\mathbf{y}\}$, approximating the image formation process described previously in 2.2.4. The first step involves the simplified version of Eq 2.3 and discrete approximation of

$$\begin{aligned} I^*(x, y, z) = \iiint_\Omega & (O(u, v, w) + b) \\ & PSF(x - u, y - v, z - w) du dv dw \end{aligned} \tag{6.5}$$

where b is set to be a constant and PSF as set as a non-spatially varying Gaussian with a standard deviation in each direction of PSF_i. For efficiency, the convolution is only done once over the template object, allowing a high sampling approximation to the Object function, without explicitly storing it. The ground truth image is then formed by integrating over the pixel (voxel) volume (h_x, h_z, h_y) to create the pixel intensity for each location as

$$I_{gt}\{x, y, z\} = \int_{x-h_x/2}^{x+h_x/2} \int_{y-h_y/2}^{y+h_y/2} \int_{z-h_z/2}^{z+h_z/2} I^*(u, v, w) du dv dw \tag{6.6}$$

for fixed locations $\bar{\mathbf{y}}$, the spacing of pixels and pixel volumes does not need to be the same (isotropic). However, this is the case for the benchmarks here. Note, we have also integrated of z dimension as a simplification. Again a discrete approximation to the integral is used. The last step involves the corruption of the image by noise as

$$I\{x, y, z\} = I_{gt}\{x, y, z\} + \eta(x, y, z, I_{gt}\{x, y, z\}) \tag{6.7}$$

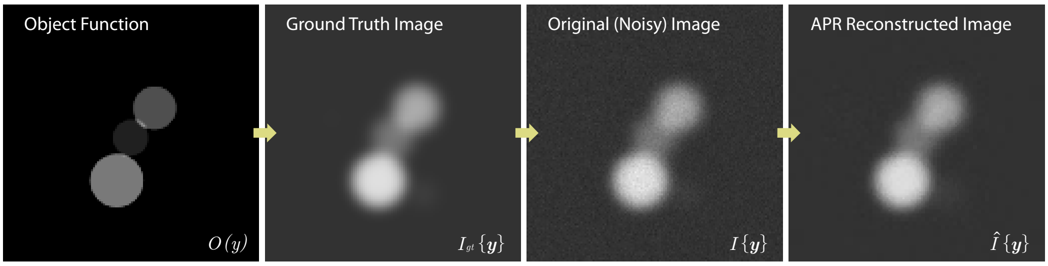


Figure 6.11: Flow chart showing the generation of synthetic images used for benchmarking the APR. First template objects are generated of a certain size, given locations (x_i, y_i, z_i) , and brightness B_i, to define the Object function $O(\mathbf{y})$ (*left*). The Object function is then blurred through convolution with a Gaussian kernel PSF, and then sampled to produce the Ground Truth Image $I_{gt}\{\mathbf{y}\}$ (*center left*). This ground truth image is then corrupted by a Gaussian approximation to Poisson Noise η, to generate the Original Image $I\{\mathbf{y}\}$ (*center right*). This original image is then transformed into an APR. The APR can be then used to produce a reconstructed image $\hat{I}\{\mathbf{y}\}$ that can be compared with both the original and ground truth image for benchmarking.

where $\eta(x, y, z, I_{gt}\{x, y, z\}) \sim \mathcal{N}(I_{gt}\{x, y, z\}, I_{gt}\{x, y, z\})$ a Gaussians noise with mean and variance equal to the intensity of the pixel as an approximation to Poisson noise [76]. The image $I\{x, y, z\}$ at locataions $\hat{\mathbf{y}}$ we denote as $I\{\hat{\mathbf{y}}\}$, it is this image that is transformed into the APR.

6.3.3 Summary

Figure 6.11 provides an example of 2D slices of the steps in the synthetic image generation pipeline. Throughout the benchmarks below, we alter the synthetic images regarding image size, information content, quality, and sampling. We will briefly describe how this is done for each, relating to the parameters mentioned above. Figure 6.12, provides examples of what the original image of a fixed sphere template looks like under different conditions.

Image size

The image size can be changed by setting the appropriate size of the domain Ω, pixel locations $\mathbf{y}$ and pixel size h_x, h_y, h_z.

Information content

Given our Object function model of the image, we can define the level of information content to be proportional to the number of objects M in the image. Therefore, we can scale the image content for a given size image and sampling, by increasing the number of objects M. The objects are given random locations uniformly distributed across the domain and often have a random uniform distribution of brightness B_i, within the range B_{min} and B_{max}. An example is shown in the *right* image in Figure 6.12.

Image blur

We alter the degree of image blur and its shape using the width of the Gaussian kernel and its standard deviation parameters PSF_i. Here, we show results for three levels of blur; we call small, medium, and large blur. They correspond to a standard deviation in terms of pixels of 1, 3, and 6 respectively. Figure 6.12 in the *left* most column provides an example of how these blur kernels impact the same template.

Image quality

Here we consider either noise-free, that is η is set to zero, or the noisy case using a Poisson noise approximation. The image quality can be then altered, by changing the relative magnitude of the η compared to the object brightnesses B_i. The mean of η within any object i can be approximated by a combination of $b+B_i$. Therefore, we can increase or decrease the image quality by increasing or decreasing the ratio of $\frac{B_i}{B_i+b}$. This is done by keeping a fixed average object brightness B_i and then changing the background b. Hence, we are altering the average Peak Signal to Noise Ratio (PSNR) of the image. We show results here for three levels of image quality we call low, medium and high Image Quality (Abbreviated to Qual in figures). Figure 6.12, *third column* gives examples for these levels of image quality.

Sampling

Lastly, the degree of sampling can be changed. This involves decreasing the pixel size h_x, h_y, h_z and sampling $\mathbf{y}$ while keeping all other variables fixed in real variables. Practically, this means the PSF width PSF_i defined in pixels has to be appropriately increased. The increase in sampling can be thought of as zooming in on the object, as with a camera lens.

6.4 APR properties benchmarks

All benchmarks were run using the 3D pipeline described above that has been implemented in a C++ Library using OpenMP shared memory parallelism for most algorithm steps. A workstation running Ubuntu, Xeon E5-2660 v3 (25M Cache, 2.60 GHz), 64 GB ram was used to run all benchmark and applications

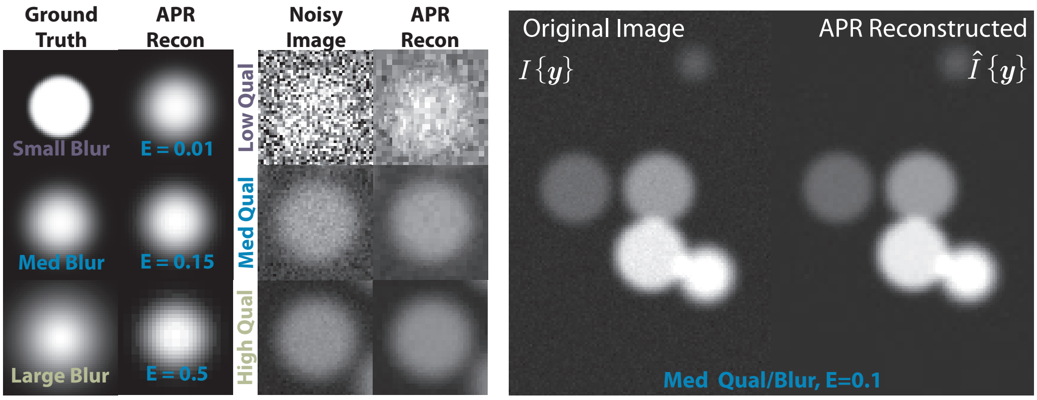


Figure 6.12: A summary of the different groups of synthetic images used in the benchmarks. The *first* column shows the three levels of blur used. The *second* column shows examples of the APR reconstruction using $\hat{I}_{pc}$ for three relative errors E, for the medium blur noise-free case for a close up of a sphere object template. The *third* and *forth* columns show the original image, and APR reconstruction ($E = 0.1$) for noisy images for the medium low and high image quality levels used. The *left* image shows the original image and reconstruction for medium blur and image quality with $E = 0.1$, showing the random distribution of object location and brightness B_i.

below. In this section, we will give a description of each benchmark and describe the results and give the exact parameters used in A.9.

6.4.1 Noise-free Reconstruction Condition

In the first set of benchmarks, we assess if the Reconstruction Condition holds for noise-free synthetic images. The benchmarks are similar to those run in the 1D case above. The details of all the parameters used are given in A.9.1. For these benchmarks, we again show the observed reconstruction error which as for the 1D case above is

$$E^* = \max_{\mathbf{x} \in \bar{\mathbf{y}}} \left(\frac{|\hat{I}\{\mathbf{x}\} - I_{gt}\{\mathbf{x}\}|}{\sigma(\mathbf{x})} \right) \tag{6.8}$$

that is the infinity norm of the pointwise reconstruction error relative to the computed Local Intensity Scale σ at that point. The observed reconstruction error was calculated at all pixel locations in the original image $\bar{y}$.

In these benchmarks, the number of objects is held fixed (five), and for a given E an image is generated with randomly placed objects with brightnesses that vary randomly over an order of magnitude. The first plot in Figure 6.13 shows the observed reconstruction error E^* for piecewise constant reconstruction and for 40 values of E, and 40 repetitions for small, medium and large blur. For all blurs and E, the reconstruction error is below the dotted line that

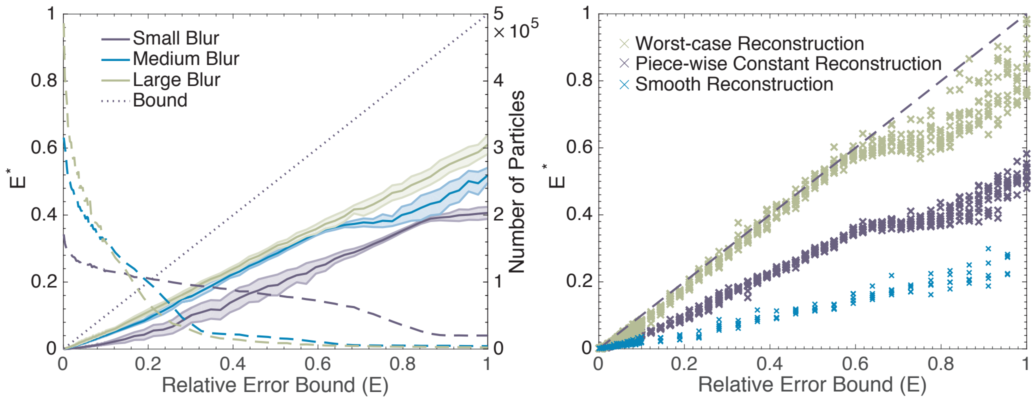


Figure 6.13: The first plot, on the *left* axis shows the observed reconstruction error E^*, I_{gt} the ground truth and piece-wise constant reconstructed image $\hat{I}_{pc}$ for a noise-free input image, the dotted line is $E^* = E$, representing the Reconstruction Condition. The shaded areas represent an estimate of the standard deviation and the solid line the mean. The *right* axis shows the mean number of particles in the APR. Both are plotted against the relative error bound E shown for synthetic data small, medium, and large levels of blur as shown in Figure 6.12. The second plot shows the medium level of blur benchmark, but showing the observed reconstruction error for worst-case, piecewise constant and smooth reconstruction methods (See 5.6.1). Each cross represents an individual APR and Image comparison.

represents $E = E^*$ the Reconstruction Condition. The second plot in the figure shows the medium blur benchmark but using the three different reconstruction methods. The piecewise constant and smooth reconstruction methods obey the Reconstruction Condition. However, for the worst-case reconstruction, we find three points out of 1600 realizations do not satisfy the bound, and they sit slightly above it the dotted line. This breaking of the Reconstruction Condition could arise from the smoothness assumption for the Local Intensity Scale failing at these points, or enter through a numerical error in $L(y)$ or other transform steps. Another important observation is the tightness of the Reconstruction Condition when assessed by the worst-case reconstruction. This implies that the APR is effectively adapting, as the worst-case reconstruction is almost exactly E. These two benchmarks used sphere template objects, to test whether the results were affected by the geometry of the template, we used an anisotropic 'Octopus' template (See Figure A.4), shown in the first plot in Figure 6.14. Comparing the results to the sphere template results, we see little qualitative, except the response appearing more linear for higher E for the octopus template. These results are consistent with running other benchmarks with a variety of templates. Based on this, for the rest of the benchmarks presented here, we only use our sphere templates, due to their computational simplicity.

In summary, except a few points, we find the Reconstruction Condition

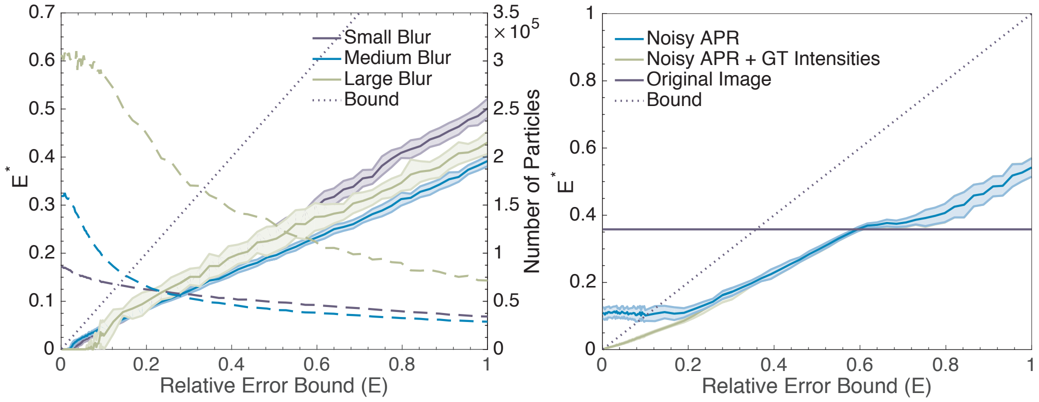


Figure 6.14: The first plot is a repetition of noise-free benchmark in Figure 6.13 with octopus template image (see Figure A.4) used instead of the sphere template for the Object function. The second plot shows the observed reconstruction error E^* for piecewise constant reconstruction for a noisy image. The *blue line* shows the infinity norm of the observed relative error E^* in response to changing relative error bound E with a noisy image (medium quality and blur) as input to the APR against the desired bound E. The Reconstruction Condition $E = E^*$ given by *dark blue dotted* line. We see the observed reconstruction error reaches a lower bound which it does not decrease below and therefore is above the bound for small E. The *green line* shows the result when noisy intensities f_p are replaced with noise-free ground-truth intensities $f_{gt,p}$, but the noisy sampling is used. In this case, the observed relative error now obeys the bound for all E. The *solid blue line* shows the average observed infinity norm of the observed relative error E^* for the original images

holds, despite no guarantee on this due to the non-constant Local Intensity Scale σ. Because, the worst-case reconstruction (almost) meets the bound, then any reconstruction method using a weighted average of points will also meet the Reconstruction Condition. Therefore, in the noise-free case, the APR guarantees a user set reconstruction error E relative to a local intensity scale, and therefore we conclude that for these cases it satisfies **RC**2. In addition to the reconstruction error, for the first and third plot, we also provide the average number of particles for the APR for a given relative error E. As expected intuitively, like in the 1D case, the average number of particles monotonically decreases as E increases. However, in these examples, instead of having two regimes, there appear to be multiple scaling regimes that depend on both the template and the size of the blur kernel used. One observation is that for low blur, the number of particles is much less sensitive to E, then for the higher blur images that show much larger gradients in response to E.

6.4.2 Noise corrupted Reconstruction Condition

In reality, however, LSFM data is corrupted by noise. In 4.3.3, we gave an argument that we would expect the observed reconstruction error E^* to reach a lower bound as $E \to 0$. Indeed, we see this in practice. In the first plot of

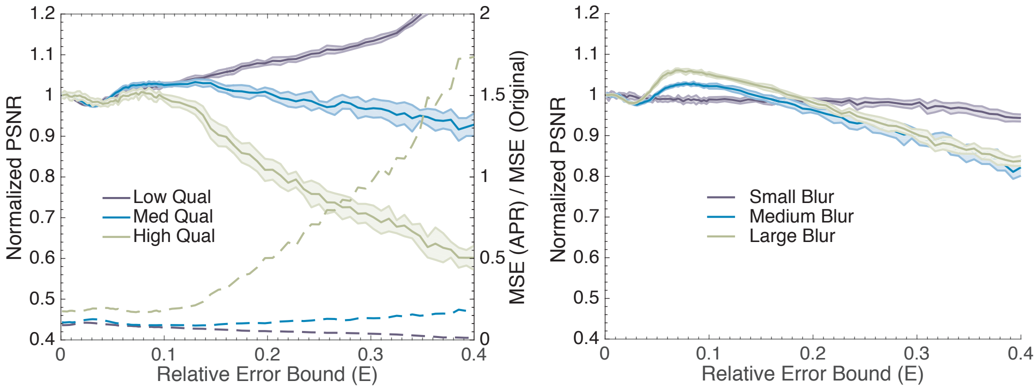


Figure 6.15: The *left* axis of the first plot shows the PSNR of the reconstructed APR from a noise corrupted image, normalized by the PSNR of the APR with $E = 0.001$. The *right* axis shows the ratio of the mean squared error (MSE) of the APR reconstructed image over the MSE of the original image compared to the ground truth, plotted against relative error bound E. The second plot also shows the observed PSNR against the relative error but for medium image quality and small, medium and large image blur.

Figure 6.14 we repeat the same experiment for medium blur case presented in Figure 6.13 *right*, but for a noisy input image with medium image quality. We see that for values of $E > 0.2$ the observed reconstruction error decreases with E, however, beyond this point it reaches a lower bound and no longer increases. For values of $E < 0.1$, the observed reconstruction error is then above E violating the Reconstruction Condition. Although, the analysis, relies on the adaptation being correct, and the lower bound arising only from the uncertainty of the noise-free values of f_p. To discriminate the source of the bound, we also calculated the observed reconstruction error when the noisy f_p are replaced with intensities from the ground truth image $f_{gt,p}$ (i.e. without noise), but still with the noisy construction of $\mathcal{V}$ and particle placement. We find that in this case (shown by the green curve) the Reconstruction Condition is again satisfied. We find similar results for the low and high image quality results, with the lower bound shifting appropriately. These results indicate that the approximation of $\mathcal{V}$, and the Implied Resolution Function $R^*(y)$ are relatively robust to noise in our benchmarks. Then, given the observed reconstruction error E^* reaches a lower bound, how should we choose E in the presence of noise? To explore this question, we look at the observed Peak Signal to Noise Ratio (PSNR) of the reconstructed image. Here, we have calculated the observed PSNR as

$$PSNR = 10\log_{10}(\frac{64000}{MSE}) \tag{6.9}$$

where MSE is the Mean Squared Error which we calculated as

$$MSE = \frac{1}{N} \sum_{\mathbf{y}_i \in \bar{\mathbf{y}}} (\hat{I}\{\mathbf{y}_i\} - I_{gt}\{\mathbf{y}_i\})^2 \tag{6.10}$$

where as previously N is the number of pixels in the original image. As we are only concerned about the relative change of the observed PSNR with respect to E, we use the Normalized PSNR, which is simply the observed PSNR at E divided by the average observed PSNR of an APR with $E = 0.001$ for the same type of image. The Normalized PSNR allows us to compare the results across different image qualities and blur levels.

The first plot in Figure 6.15 shows Normalized PSNR against relative error E for low, medium, and high image quality with medium blur. For the different image qualities, we see different dynamics. However, the cases have some common features. Interestingly, the maximum PSNR does not occur as $E \to 0$. Instead, all qualities show a nonlinear behavior as E gets small. For the low and medium quality images, the PSNR has a maximum in E between $0.08 - 0.15$. For the low image quality case, there is a positive relationship for larger E and the Normalized PSNR. This is likely due to the image being so noisy, that the improvement in image quality from downsampling the image dominates.

In the second plot, we show the same benchmark, but for medium image quality, and changing blur. Here, the medium and high blur images show similar dynamics, reaching a maximum PSNR between $E = 0.05 - .1$. The low blur images, however, show almost no dependence on E. This low dependence is similar to that seen in the average particle number. Therefore, for noisy images, there seems to be an optimal range of E between $0.05 - 0.15$ across medium to high image blur and medium to high image quality. Fortunately, from observation, this appears to align with the image quality and blur of common LSFM data. Because higher values of E are preferred as this leads to a lower number of particles being used by the APR, in cases where non-prior information is known we use a default value of $E = 0.1$ seems appropriate.

Lastly, we address the how the image quality of the reconstruction from the APR compares to the original noisy image. For the optimal range of E do we find a higher or lower image quality than the original? To address this on the *right* axis of the first plot in Figure 6.15, we show the ratio of the MSE of the reconstructed APR image, to the MSE of the original image for the medium image quality and varying blur. The results show that in the optimal range the MSE of the APR is four to five times lower than the MSE of the original image. We find similar results across image qualities (not shown). Therefore, we can conclude, that in this range, the errors made due to optimization are 'within' the error of the noise of the original image.

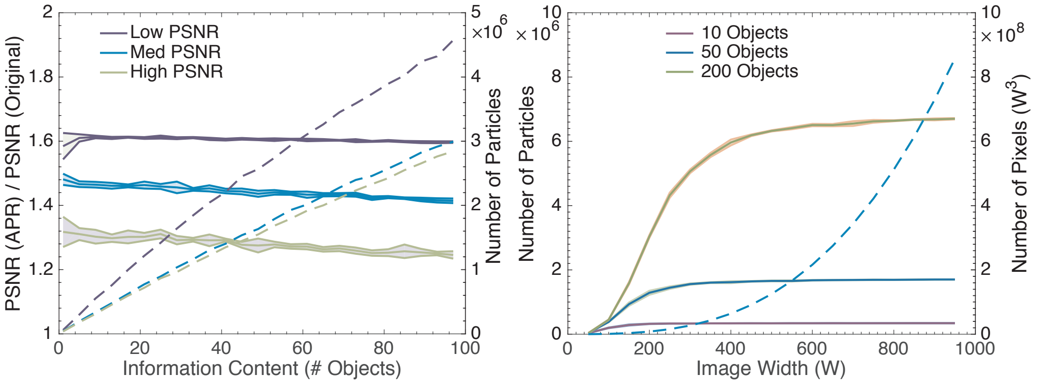


Figure 6.16: The first plot on the *left* axis shows the ratio of the PSNR of the APR reconstructed image, compared to the PSNR of the noisy original image (Only computed for areas above local information scale threshold σ_{TH}). On the *right* axis, the number of particles, against the information content set by the number of template objects M in the original image (parameter details: A.9.4). Only the mean is given as the confidence interval was indistinguishable on the plot. The second plot on the *left* axis shows the number of particles, and the right *right* axis the number of pixels, plotted against the original image width, for $10, 50$ and 100 objects in the image (parameter details: A.9.5).

These results indicate that for our benchmarks the APR can satisfy the second part of **RC**2, in that given an appropriate E, the APR adapts, while not reducing the signal-to-noise ratio compared to the original noisy image.

6.4.3 Increasing information content

In the following benchmark, we address how the APR adapts to image content. We can test this through increasing the number of objects M we use to construct the Object function O for a fixed size image. The first plot of Figure 6.16 on the *left* axis shows the average number of particles for an increasing number of objects for low, medium, and high image quality with medium blur. We find an approximately linear relationship between the number of objects and the average number of particles across image quality. Notably, the low image quality images require a higher number of particles on average, likely reflecting over-sampling due to uncertainty in $L(y)$. The relationship is not exactly linear, due to the fixed size of the domain. As the number of objects increases the likelihood of them over-lapping increases, given that the maximum resolution is fixed to pixel resolution this reduces the required number of particles, leading to a sub linear relationship. Therefore, for our benchmarks, the APR appears to be adapting to the information content of the image.

However, how does the image quality change with the number of objects? We address this, by plotting the observed PSNR of the reconstructed image normalized by the observed PSNR of the original image for non-background regions (see Figure 6.16). The *right* axis of the plot shows that this ratio is nearly constant when plotting against the number of objects, with a slight negative trend. However, across all, the ratio is always greater than 1, with lower quality images shower a higher ratio. We restrict the PSNR to non-background regions because otherwise the PSNR is dominated by the proportion of the image that does not have any objects. This is due to the reconstruction error being near zero in these flat regions. The near zero reconstruction error in the background results in a strong negative correlation with the number of objects and is un-informative. Instead, the restricted ratio provides us an informative lower bound. From the constant response, we, therefore, can conclude that for our benchmark data the APR effectively adapts to the information content while maintaining image quality.

6.4.4 Increasing image size

We have shown that for a fixed image size, the size of the APR scales with the information content, but what about for fixed information content and increasing size? The second plot in Figure 6.16 addresses this question, plotting the number of particles for 10, 50 and 100 objects against increasing image width W ($N = W^3$). That is, the size and number of the objects are held constant and the size of the domain they are placed in increases. The three curves all show similar, but scaled and shifted, dynamics. In all cases, the number of particles increases towards a limiting fixed number of particles. That is, beyond a given image size, the number of particles in the APR becomes constant. The growing number of particles for lower image sizes likely reflects the objects overlapping due to the confined domain, resulting in the increase in the number of particles as the objects become less over-lapped on average. To provide perspective, the number of pixels N is also plotted on the *right* axis. From the above, we conclude that the size of the APR reflects the number and distribution of objects and not the original image size N. Therefore, when combined with the conclusions of the last benchmark, we assess that the APR fulfills **RC**1 for the data presented here.

6.4.5 Increasing sampling rate

In the last benchmark of this section, we explore how the APR of a fixed object distribution and blur responds to a change in sampling. That is, does the APR becomes independent of the originally chosen sampling? Conceptually, this is equivalent to deciding what resolution of the image to set when using a digital camera to capture a fixed scene. Here, we ignore any practical issues that would result from the reduction in the number of photons per pixel. For this benchmark, we place a single object in the center of the image and then increase the sampling by decreasing $h_x = h_y = h_z$, while keeping the PSF and domain constant in real terms (but increasing in terms of pixels). The smoothing parameter λ is also increased as to represent a fixed smoothing length in real terms. In the first plot of Figure 6.17 we show the number of particles plotted against the width W. For small image sizes, we see a linear increase in the number of particles. However, above an image width W of approximately 100, the number of particles oscillates around a fixed value. The equivalent noise-free benchmark shows similar dynamics; where it is clear the oscillations show a distinct pattern increasing between consecutive widths that are a power of 2. Therefore, these oscillations appear to be reflecting the changing relationship between the quantization and $L(y)$. Hence, on average the number of particles reaches a 'maximum' sampling. We note that an image with high blur was used to result in the APR reaching a maximum resolution l at a low image size to aid computational simplicity. This effectively decreases the image size for which the fixed size 'kicks in' (See A.9.6 for additional details on the benchmark).

Given that some limiting APR is reached, how is the image quality of the reconstruction affected by further increases in sampling? We address this question in the second plot in Figure 6.17, showing the PSNR of the same benchmark against image width. Here, we find that the reconstructed image quality increases as the sampling rate is increased until an image width of approximately 400. Beyond this width, the value begins to oscillate, possibly indicating that the gains from in improvement of image quality from an increasing number of samples are smaller error induced by the sampling technique. The above results show that the APR can utilize an increase in sampling to increase image quality while keeping the resulting size of the representation constant.

6.5 APR performance benchmarks

In this next section, we give results of 'performance' benchmarks for the transforming, and the memory and file storage costs of the APR. From the results in the previous section, we know that the size of the APR, in terms of the number of particles (or Particle Cells), depends on the information content. However, the cost of transforming, and representing the APR in memory, also depends on the original image size N. The dependence of memory and storage cost on N arises through the pixel sampling limiting the highest resolution l_{max} of the representation. Although in the previous benchmark we showed that

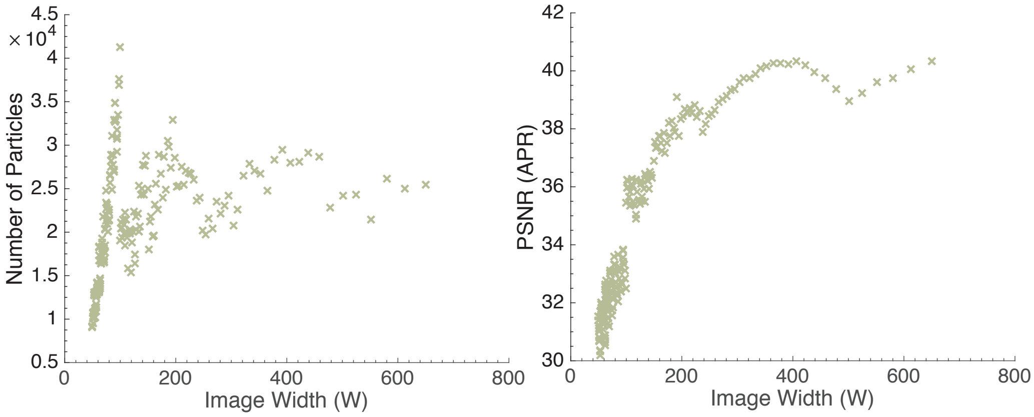


Figure 6.17: The two plots show the number particles N_p, and PSNR of the APR for an image with fixed image content and blur size, but increased sampling, and hence width W. The benchmark is equivalent to choosing the resolution of a natural image for a fixed scene. (See A.9.6 for parameters)

the APR becomes independent of the initial sampling resolution, most images do not appear to be sampled in this regime. Therefore, to understand the performance requires understanding the dependence on both the level of information content, the number of objects, and the original image size N. Instead of discussing absolute particle numbers, it seems more intuitive to discuss the ratio of particles N_p to original pixels N. Hence we define the Computational Ratio (CR) as

$$
\begin{aligned}
CR &= \frac{\text{number of input pixels}}{\text{number of output particles}} \\
&= \frac{N}{N_p}
\end{aligned}
\tag{6.11}
$$

where an image with a high amount of information content for its size will have a small CR, and an image with a low amount of content for its size a large ratio.

In the analysis below, and in Chapter 7 which explores processing with the APR, we use two groups of test datasets. The first, we call the Computational Ratio benchmark data, and consists of synthetic data, with the CR set to three levels, high, medium and low. The second, we call the Exemplar benchmark data, consists of nineteen LSFM datasets of various size, species, and labeling.

Following we will first describe the CR and Exemplar benchmark data, then introduce the APR data structures we have used and then provide results for the computational cost of forming the APR, and its storage cost.

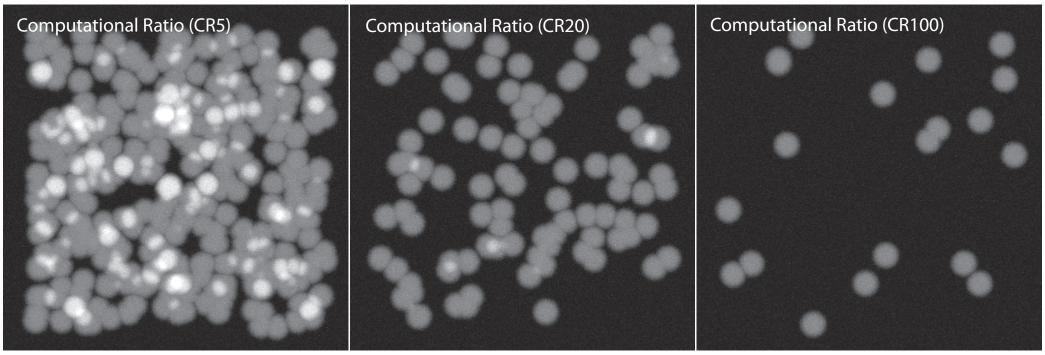

Computational Ratio (CR5)
Computational Ratio (CR20)
Computational Ratio (CR100)

Figure 6.18: Maximum intensity projection of examples of the Computational Ratio (CR) benchmark data that are used to represent low (CR100), medium (CR20) and high (CR5) levels of information content ($N = 400^3$).

6.5.1 Computational Ratio benchmark data

To represent low, medium, and high image content relative to image size, we generate data sets for varying image size N and number of objects M that approximately correspond to CRs of 100, 20, and 5 respectively. In Figure 6.18, we show the maximum projection for examples of a CR5, CR20, and CR100 synthetic data sets with image size $N = 400^3$ (We use this CR'X' notation in figures and the remaining text). However, we could not determine a procedure for generating a precise CR for a given image. Instead, the datasets were generated using a linear estimate of the number of objects required to reach a certain ratio (See A.10.1). Generated in this way, the CR does vary across N, and the average CR values for $N = 200^3$ to $N = 1000^3$ are 5.8, 19.3, and 89.4 for the CR5, CR20, and CR100 cases respectively. However, I do not believe this detracts from the analysis. The values of CR were set as to span realistic values as seen in the Exemplar benchmark data discussed next.

6.5.2 Exemplar benchmark data

To add to the analysis using the CR benchmark data, we also run benchmarks on a corpus of nineteen LSFM datasets. The Exemplar data is intended to give real examples of ratios and times that can be achieved for the APR with our current implementation. In Figure 6.19 we show an example of a 2D section of the original image, APR in particles, and reconstruction for data set 7 of nuclei from an early developing Zebrafish. The datasets are summarized in Table A.1 and parameters used in Table A.2. How parameters was set is discussed in A.10.2. The first plot in Figure 6.21, shows the CR for the exemplar benchmark data. The CRs for the exemplar data sets range from

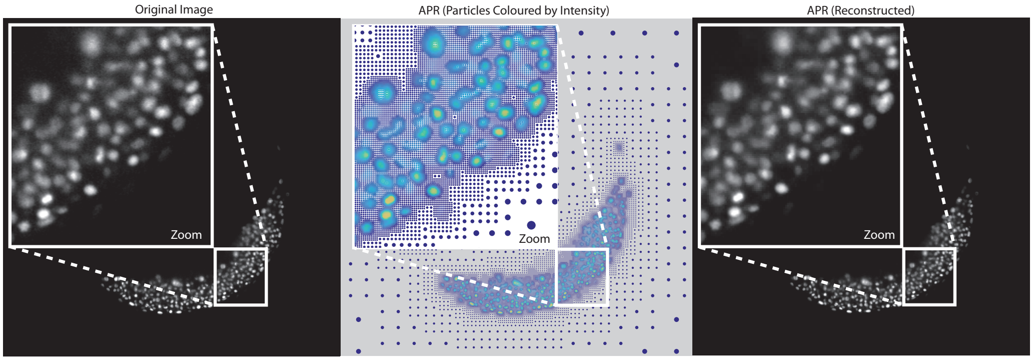


Figure 6.19: A 2D slice showing the original image I (*left*), particles of the APR coloured by intensity (*middle*), and reconstructed image $\hat{I}_{pc}$ (*right*) of labelled cell nuclei for a developing Zebrafish (Images courtesy of Gopi Shah, Huisken Lab, MPI-CBG Dataset number: 7 Table A.1) The insets show a close up of the same region. The particle rendering was created by rendering all particles from Particle Cells from which the image plane intersects.

a minimum of 5.6 to a maximum of 180, with a mean of 42.1 and median of 28.5. The numbering in figures corresponds to the number in Table A.1.

6.5.3 Data structures

Appropriate data structures must be used to be able to store, and process on, the APR. Ideally, these structures allow fast memory access at low overhead. Here, we propose two different multi-level data structures for the APR, with each level, encoded similar to sparse matrix schemes [101] and are described in detail in A.5. Alternative data structures could be used for the APR, and we do not claim those we use here are optimal. However, I have tested alternative data structures (not shown) including those based on hash tables and tree data structures and found the two proposed here provided superior all-around performance. The choice of data-structure is critical to the performance of using a data representation for processing, and I leave further investigations and comparisons for future research.

The data structures both efficiently encode $\mathcal{V}$ and $\mathcal{P}^*$ in memory by only storing one spatial coordinate per Particle Cell in a sparse matrix like representation. The Particle Cells are stored in structures by level l. On each level, the Particle Cells $c_{\mathbf{i},l}$ are stored in vectors according to the last two of the coordinates of $\mathbf{i}$, and ordered by the first spatial coordinate. We refer to the first coordinate as y, and this represents the memory access direction. The data structures are split into two components, one representing $\mathcal{V}$ called the access data structure, and the second storing the particle intensities $\mathcal{P}^*$ with in an identical memory layout. The data structures are called the Sparse APR (SA), described in A.5.1, and shown in Figure A.5, and Sparse APR Random Access (SARA) data structure, described in A.5.2 and shown in Figure A.6. The SA requires linear in y iteration for neighbor access, and the second permits random access of neighbors at the cost of a higher memory overhead. The SARA achieves random access by an efficient storing or neighbors and also stores the Particle Cell type.

The memory cost of storing the APR for the two data structures is discussed in detail in A.5. In summary, when storing intensity as float, the SA data structure requires 50 % extra memory in addition to the particle intensity, and the SARA structure 200 %, compared to storing an image that had N_p pixels. Therefore, the memory cost of storing the APR will only be less than the original image in memory if the CR is greater than 1.5 for the SA, and 3 for the SARA. However, this high overhead is not incurred on any additional properties to the particle intensity that needs to be stored (there is a slight overhead see A.5.1). For example, when performing processing tasks, additional variables are needed to be stored per pixel, or for the APR per particle. For an image, these additional variables scale directly with N. However, for the APR these directly scale with N_p and therefore will have a lower memory cost will be lower by the factor approximately equal to the CR. Therefore, in practice, the access over-head per particle is usually amortized by the use of multiple particle properties. Examples of this are shown in the next chapter.

In this chapter, we only provide benchmarking of the computational cost of forming the SARA structure. However, the formation of the SA structure has a lower cost. Further benchmarks regarding the computational performance and memory cost of using the SA and SARA structures for processing is the subject of Chapter 7. Although neighbor access can be done using the SA, the SARA data structure was used in all benchmarks where neighbor access was required.

6.5.4 Execution time

In this section, we analyze the computational cost, or execution time, for forming the APR from an original image with N pixels. The steps of forming the APR have been summarized the schematic in Figure 6.9 earlier in the chapter. The pipeline can be broadly grouped into three steps, first calculating the Local Resolution Estime $L(y)$ using filtering operations on the original image, then forming $\mathcal{L}_n$ and finding $\mathcal{V}_n$ using the Pulling Scheme and lastly constructing an APR data structure. Below, we first address the overall cost and then provide more detailed analysis for the three steps. We have not included the time taken to load the image in memory in the analysis here.

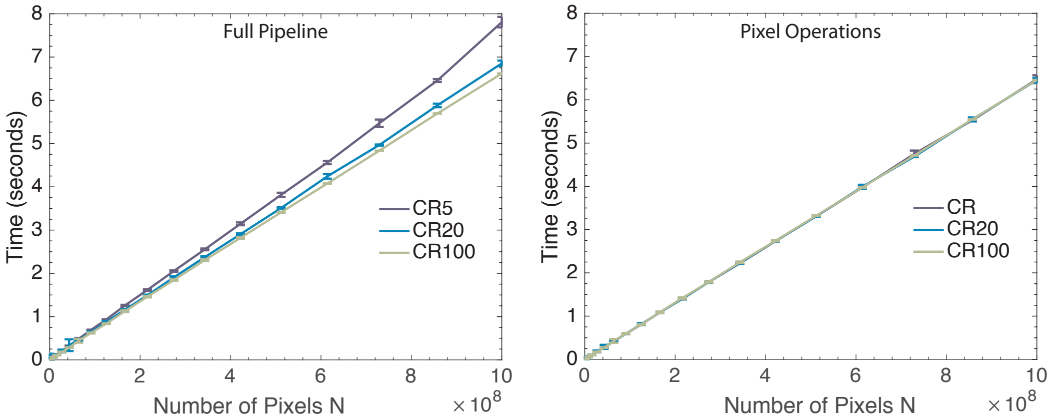


Figure 6.20: The first plot shows the execution time to produce the APR from the original image, to the APR data structure in seconds, for images from size N from 50^3 to 1000^3 using the CR5 (dark blue), CR20 (light blue) and CR100 (light green) benchmark data. The second plot shows the time taken for the filtering steps on the original image for the same benchmark data. These are the steps to calculate $L(y)$ and median filtering on the whole image. The standard deviation is shown by the error bar.

Full pipeline

The first plot in Figure 6.20 shows the total execution time for forming the APR from an input image for the CR benchmark data. The time taken shows a linear dependence on N, with a slight increase in computational cost for decreasing CR. The time corresponds to an average of 130 MegaPixels per second for processing CR5 input images or taking 7.2 seconds to form the APR from an input image of size $N = 1000^3$ and CR5. The execution time for the Exemplar data is shown in the second plot in Figure 6.21. The execution times range from 1 second to 16 seconds for the largest datasets. With an average execution time of 6.7 seconds.

Pixel steps

In the second plot of Figure 6.22, we show the execution time of the filtering steps on the original image. From comparison with the first plot, we can see that this represents a majority of the computational cost and that these steps do not depend on the CR. These steps include the calculation of the gradient magnitude, calculation of $\sigma(y)$, construction of $\mathcal{L}$ and median filtering for intensity estimation. The contribution of each of these steps to the total execution cost is shown in the first plot of Figure 6.22. On average, these steps account for over 80% of the total computational cost. With the calculation of the gradient magnitude using B-splines accounting for over 50% of the computational cost. I believe this relatively high cost of the full image

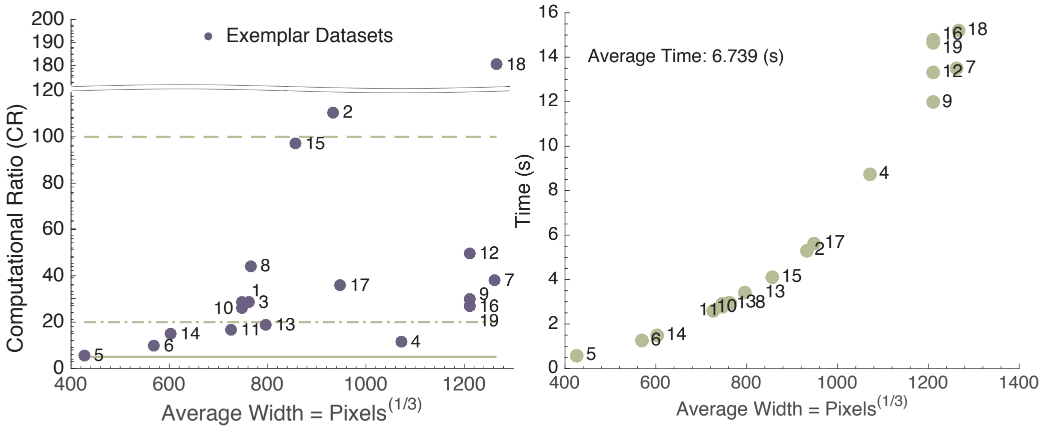


Figure 6.21: The first plot shows the Computational Ratio (CR) of the Exemplar benchmark data, plotted against the images average width ($N^{1/3}$)). The number corresponds to that given for each dataset in Table A.1. The second figure shows the total execution time to form the APR for the Exemplar benchmark data, again plotted against the average width.

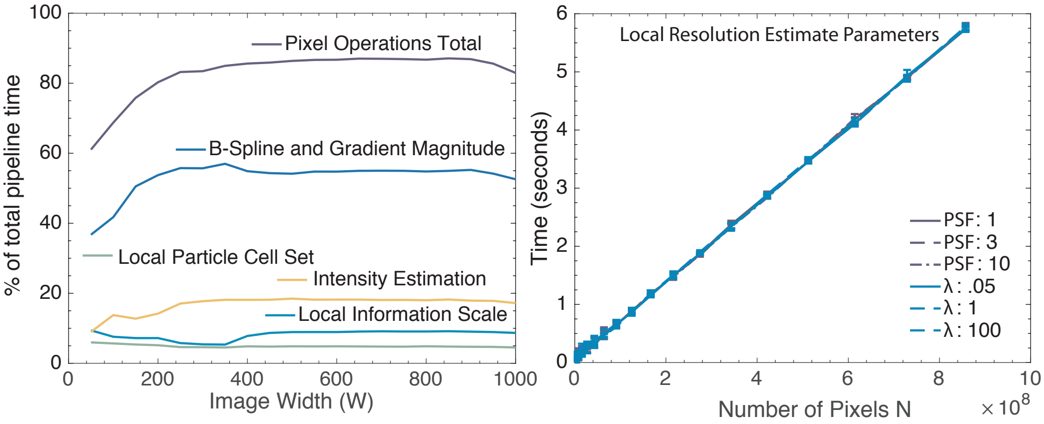


Figure 6.22: The first plot shows the proportion of the total execution time taken by each the different pixel operation steps against image width W for the CR benchmark data. The second plot shows the total execution time for steps forming $L(y)$, for a range of different parameter values, plotting against the number of pixel in the original image. The lines are indistinguishable, reflecting that the steps execution time is independent of the parameter values.

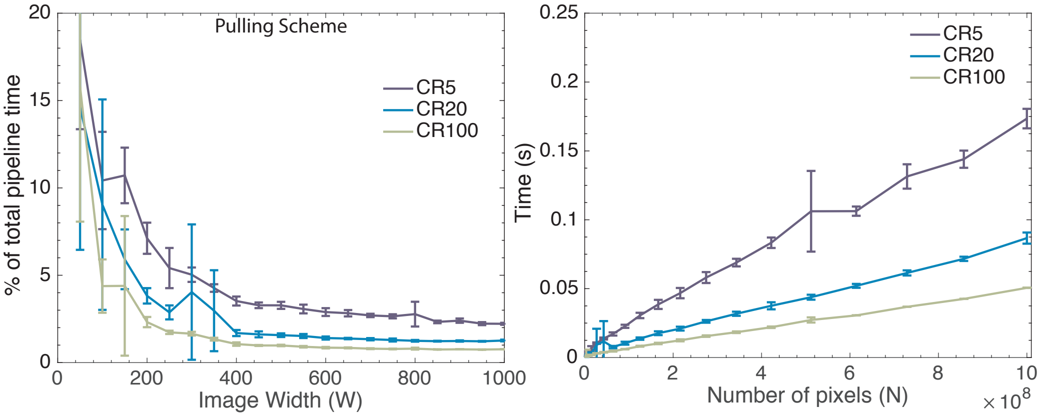


Figure 6.23: The first plot shows the percentage of the total execution time taken by the Pulling Scheme algorithm, that takes $\mathcal{L}_n$ as input and calculates $\mathcal{V}_n$ against image width W for the CR benchmark data. The second plot shows the same data but plots instead the average execution time. The error bars in both cases represent the estimated standard deviation. (Note the dynamics for small image sizes are an artifact of the CR for small images being inflated to near 5 for all datasets.)

pixel operations is not a consequence of expensive operations (as all filters have efficient algorithms and implementations). Instead, it is a reflection of the algorithms scaling with N, for very large N, compared to the pulling scheme and data structure formation scaling with the number of required particles.

Pulling Scheme

In the first plot of Figure 6.23 we show the proportion of the total execution time taken by the Pulling Scheme taking the natural Local Particle Cell set $\mathcal{L}_n$ and forming $\mathcal{V}_n$. For large images $N > 600^3$ we find that across different CR levels the step accounts for less the 2.5% of the total execution time. Where as expected the higher CR leads to a higher cost, as the number of operations scale with the number particle cells in $\#\mathcal{L}_n$. In the second plot, we plot the absolute execution time for the CR benchmark data. We note the small execution times even at the largest image sizes for CR5. Further, the scaling for each dataset shows a weak sub-linear scaling in practice. However, we note that both of these plots show sporadic variation in the observed timing that hinders analysis. This variation is a result of a deficiency in the OpenMP parallelism used here. However, it does not result in an incorrect solution. Unfortunately, I have not yet addressed this issue.

Additional benchmarks To avoid this issue, and to allow for further analysis, we also present additional benchmarks here using only serial execution.

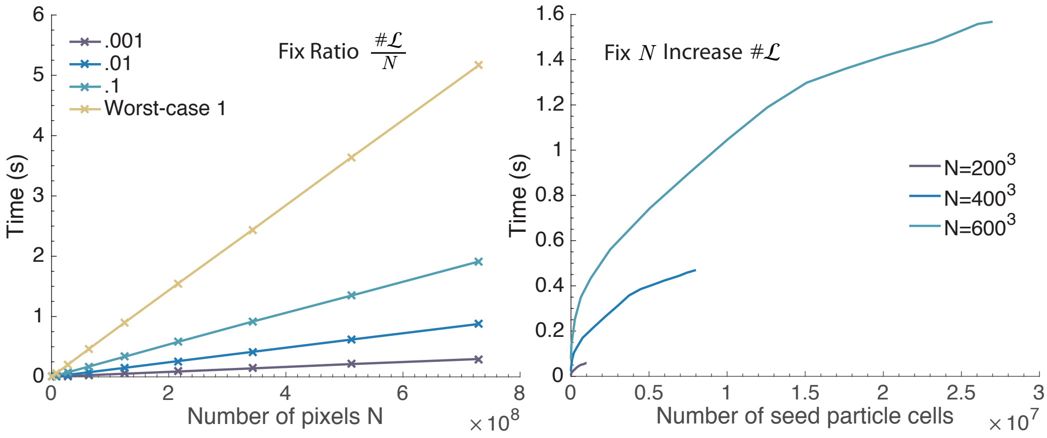


Figure 6.24: In the first plot, we show the Pulling Scheme execution time against image size for four different fixed ratios of $\frac{\#\mathcal{L}}{N}$. The Particle Cells in $\mathcal{L}$ where randomly generated and the results averaged across 10 realizations. All ratios showed linear scaling that was confirmed in a log-log plot. The yellow line corresponds to $\mathcal{L}$ containing all particle cells between l_{min} and l_{max}, representing the worst-case performance. The second plot shows the average execution time for three different fixed image sizes N, plotted against the number of seed Particle Cells in $\mathcal{V}$. The number of seed Particle Cells is simply $\#(\mathcal{V} \cap \mathcal{L})$ and hence increases linearly with $\#\mathcal{L}$. The relationship does not represent a polynomial scaling.

Independently from the CR benchmark analysis in this section, we also tested the Pulling Scheme for randomly generated $\mathcal{L}$. This allows the ability to directly alter inputs to the Pulling Scheme without having to consider the whole pipeline and how to generate the appropriate synthetic image. In the first plot of Figure 6.24, we show the scaling of the Pulling Scheme for a fixed ratio of $\#\mathcal{L}$ and N. In all benchmarks are run with Particle Cells in $\mathcal{L}$ sampled uniformly and randomly from level l_{min} to l_{max} with a set probability. This benchmark is similar, but not exactly equivalent, to the fixed CR benchmarks. First, we find confirmation of the worst case linear scaling represented by the Worst-case curve. This benchmark corresponds to the largest $\#\mathcal{L}$ for a given image size N. We also ran three other ratios, .1, .01, and .001, finding linear scaling for all. Each of these benchmarks corresponds to scaling together both the image size and the number of particle cells in $\mathcal{L}$.

In the second benchmark, we fix the image size and increase the number of Particle Cells in $\mathcal{L}$ that are randomly generated. In the second plot of Figure 6.24, we plot the number of seed Particle Cells in $\mathcal{V}$ for three different image sizes N. The number of seed Particle Cells is the number of Particle Cells given by $\#(\mathcal{L} \cap \mathcal{V})$. We find this is the appropriate variable when compared to $\mathcal{V}$ because the number of neighbor search operations is directly proportional to the number of seed Particle Cells, and not simply the absolute $\#\mathcal{V}$. For all image sizes N we find non-polynomial scaling. With the rate of increase

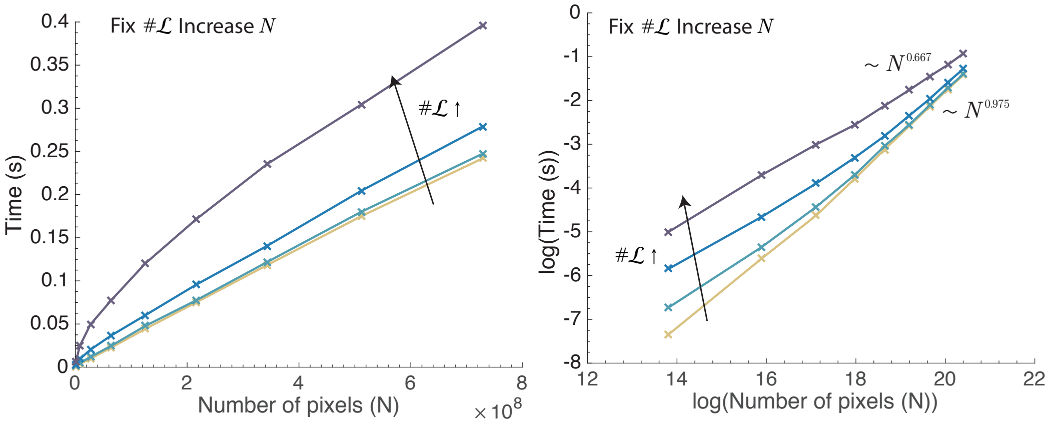


Figure 6.25: In the first plot, we show the average execution time against N four sizes of $\#\mathcal{L}$. The values were set at fixed ratios of the number of Particle Cells in the $N = 100^3$ case. The arrow notes the direction of increasing $\#\mathcal{L}$. The second plot shows the same data on a log-log plot. Linear regression shows sub-linear scaling for all plots for large N. We find that the polynomial scaling coefficient appears to decrease with increasing $\#\mathcal{L}$. The yellow line for the smallest value of $\#\mathcal{L}$ for the highest five N values had a gradient of 0.975 and Rsquare of 1, and the purple curve representing the largest value of $\#\mathcal{L}$ had a gradient of 0.667 and Rsquare again of 1.

in execution time decreasing as $\#(\mathcal{L} \cap \mathcal{V})$ increases. We note that the same relationship is seen for $\#\mathcal{L}$ (they are proportional), but arguably its is the number of seed cells that is the relevant variable. Empirically the relationship does not seem to be a polynomial nor logarithmic. In the last benchmark, we instead fix the total number of Particle Cells in $\mathcal{L}$ and consider the execution time as the image size N is increased. The results are shown in the first plot of Figure 6.25 for four different numbers of Particle Cells. The four levels were set at ratios of $0.001, 0.01, 0.1$ and 1 of the maximum number of Particle Cells in the $N = 100^3$ image. For all four levels, we find sub-linear scaling in N. That is, the execution time increases at a decreasing rate as N increases. This is confirmed in the second plot, that shows the log-log plot of the same data. Interestingly, the polynomial growth coefficient decreasing as the number of Particle Cells in $\mathcal{L}$ increases. With the smaller number of Particle Cell benchmark being almost linear, scaling at $\sim N^{0.975}$, and the largest number of Particle Cells at $\sim N^{0.667}$.

I do not have a concrete explanation for this scaling behavior. However, it also coincides with slightly sub-linear scaling that can be observed in fixed CR benchmark data. One point of insight comes from the fact that a fixed $\#\mathcal{L}$ does not imply a fixed $\#\mathcal{V}$. In fact, as N increases so do the number of particle cells in $\mathcal{V}$. However, from the fixed N benchmark we find that the dominant component of the computational cost comes from the number of seed Particle Cells in $\mathcal{V}$. However, the number of seed Particle Cells is constant across N.

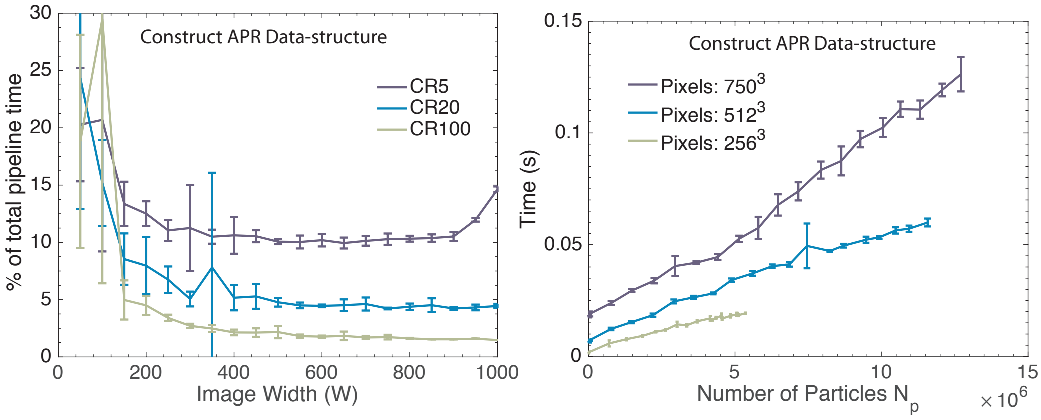


Figure 6.26: The first plot shows the percentage of the total execution time taken to form the SARA data structure from $\mathcal{V}_n$ and sample the particles intensity for the CR benchmark data plotted against the image width W. The second plot shows the execution time in seconds of the same step, but for a fixed image size and increasing number of particles.

Therefore, as N increases only the 'cheaper' steps of adding boundary and filler Particle Cells are increased. However, this is not satisfactory, but we leave further investigations to future research.

Pulling scheme summary In summary, we have confirmed that the Pulling Scheme has worst case linear scaling in N. Further, the computational cost for fixed N is proportional to the level of information content through the number of seed Particle Cells. However, we have no exact form for this scaling behavior, but it is sub linear. Further, for fixed size of $\mathcal{L}$ and increased N we find sub linear behavior, with a scaling rate that is inversely proportional to the number of Particle Cells in $\mathcal{L}$.

APR data structure

In the last step of the pipeline, the SARA data structure is formed from $\mathcal{V}_n$; this includes the particle sampling step. The first plot of Figure 6.26 shows the proportion of the total execution time for this step. Forming the data structure and sampling the particles, took on average 2, 5 and 13 percent of the total execution time for the CR100, CR20, and CR5 datasets. Therefore, showing a strong dependence on the total number of particles. This is also reflected in the second plot, showing the execution time for a fixed image size, and increasing number of particles. The dependence on N likely arises through both the maximum resolution and therefore l_{max}, and also the slow down in cache efficiency when sampling the particles from a larger image.

Execution time summary

Hence we have outlined the computational cost of forming the APR, showing an overall linear scaling in N for a fixed CR, with the Pulling Scheme showing sub linear scaling. I believe the above results show that the APR can be formed efficiently, as the largest computational cost has been reduced to the estimation of the image gradient. With the computational cost of the Pulling Scheme being relatively low.

The above results have been on a single CPU machine, utilizing OpenMP for parallelization. Preliminary work for scaling on multi-core machines and GPU implementations indicate that the pipeline is amenable to scaling through further parallelization. Especially, acceleration of the pixel steps. However, here we have preferred a shared memory implementation due to ease of development, and the typically considerably larger amount of RAM available then GPU memory. We leave these investigations to future work.

A particular task being classified as 'fast' is always relative. However, the implementation here is within the real-time values that have been given in the literature [110, 5]. However, real-time applications will entail temporal datasets. Therefore the transform would also have to include the time step, which involves additional steps, but also optimizations. If we here limit ourselves to the processing of stand alone large datasets, the above results appear to be 'fast' as they are $\mathcal{O}(1)$ with the calculation of the gradient magnitude, a simple computational task.

Therefore, for the results here, I claim that the APR can be rapidly formed with a linear scaling in the number of pixels, and hence **RC**3 is satisfied.

6.5.5 Memory Cost

The total amount of memory required to run the pipeline is approximately 3.25 copies of the original image stored as a float. This is a result of having to store the original image, and the temporary variables for calculation of the Local Resolution Estimate $L(\mathbf{y})$, and then finally sample the intensities from the original image. I am uncertain of how to reduce this further, beyond change of datatype, while processing only a single time point.

6.5.6 Storing the APR

Last, in this section, we assess the efficiency of lossless compression of the APR for file-storage. We store the APR using the HDF5 file format [124] and BLOSC HDF5 plugin [7] for lossless compression. This is done by storing $\mathcal{V}_n$ per level as a sparse matrix using an unsigned 8bit integer array, with nonzero

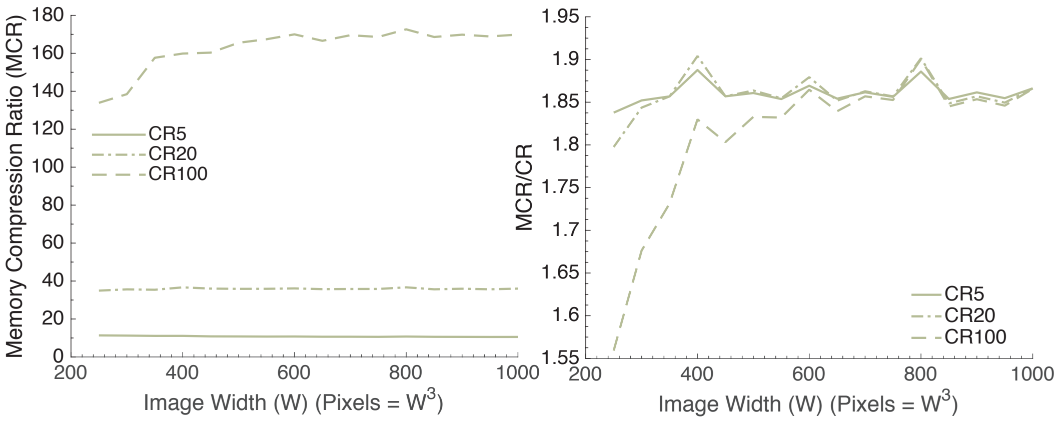


Figure 6.27: The first plot shows the Memory Compression Ratio (MCR), the ratio of the APR file-size to the original uncompressed image file size for the CR benchmark data against image width W. The second plot shows the same data, but instead plotting the ratio of the Memory Compression Ratio (MCR) and the Computational Ratio (CR).

entries set to the Particle Cell type. The particle intensities are then stored again per level unsigned 16bit integer. Therefore, in addition to storing the Particle Cell set $\mathcal{V}$ and particle intensities $\mathcal{P}^*$, we also store the Particle Cell *type*. Stored in this way the Particle Cell information is highly compressed. The high compression is reflected in that on average 92 % of the bytes are used storing the particle intensities. Further, in the limiting case where the number of particles is equal to the number of input pixels, the particle intensities account for 99.9 % of the storage cost. A.6 gives more technical details. Given the fact that the size of the APR depends linearly on image content, raw compression ratios are not necessarily informative for the storage cost of the APR without reference to its CR. Hence, we define the Memory Compression Ratio (MCR) as

$$MCR = \frac{\text{Size of the input image in bytes}}{\text{Size of the compressed APR in bytes}}. \tag{6.12}$$

We are interested in the ratio $\frac{MCR}{CR}$, that is, how well for a given CR the data set is compressed. In the first plot Figure 6.27 shows the MCR for the CR benchmark data. We can see that the MCR for images larger than 400^3 becomes constant across CR. In the second plot, we show the ratio $\frac{MCR}{CR}$ also for larger images the ratio becomes constant, with a value of approximately 1.8 for the three CRs. This result tells us that for a data set with a given CR, we can expect the file storage to be 1.8 times smaller than the CR ratio when compared to the original data size. Despite, the requirement of the additional storage of the particle cell locations, level, and type, that are not required in

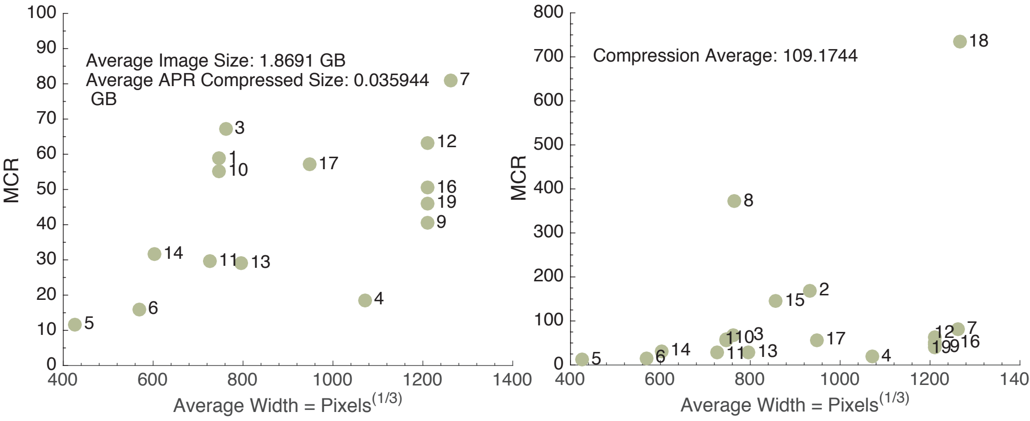


Figure 6.28: The Memory Compression Ratio (MCR) for the exemplar benchmark datasets against the geometric average width ($N^{1/3}$), where the numbers coincide with those given in Table A.1. The two plots show the same data, with the first showing a zoomed subset omitting two outlier data points.

the original image.

Figure 6.28 shows the MCR for the exemplar benchmark data. The same data is plotted twice with different scales as the variation is quite large across the data sets. The median MCR of the exemplars is 55.2 and mean 109. Regarding actual file sizes, the average size of the uncompressed input images is 1.87 GB, and the compressed APR, 36 MB.

We note that more sophisticated compression schemes should be able to be used to compress the size of the intensity data down further. In particular extensions of the Easy Path Wavelet Transform, would seem like a good fit for the APR [85]. However, we leave this to future work.

Therefore, the APR can be efficiently compressed with a filesize proportional to the image content a requirement of **RC**1.

6.6 Summary and main points

In this chapter, we presented the main empirical results of this thesis regarding the properties, and computational cost of forming, the APR.

We began by showing results for a 1D analytically defined function. There, we showed that the Reconstruction Condition held across a range of reconstruction functions, and also showed the scaling behavior of the gradient computed from the APR. We also explored the impact of discontinuities in the solution.

In the second portion of this chapter, we provided results for the APR for 3D LSFM data. First, we introduced the 3D pipeline including the choice of Local Intensity Scale σ and also the method of synthetic image generation used for benchmarking. Then we assessed the properties of the APR, both regarding reconstruction error, noise, image content, image size, and image sampling. In the last section evaluated the performance of transforming an image into the APR, regarding memory and computational costs. For this analysis, we used two datasets, called the Computation Ratio (CR) and exemplar benchmark data. Next, we discuss limitations of these results and then reflect on the representation criteria. Following this summary of the main results of the chapter is given in bullet form.

6.6.1 Limitations of the benchmark results

Here we discuss limitations of the above benchmark results, and the ability to make inferences from them on the performance on 'real' LSFM data.

Local Intensity Scale

A key limitation for the results arises from the Local Intensity Scale used here. First, we have heuristically defined an estimate of the local range of the function with a length scale. We have shown that the APR can effectively adapt taking this scale into account. We have not addressed whether or not this scale is in any way relevant or appropriate. Specifically, in the benchmarks above E^* incorporates the observed σ. Ideally, we should define a ground truth adaptation, and σ, and compare our results to these. This ignores deficiencies in the Local Intensity Scale that could lead to loss of information content. Indeed, this does occur through a lowering of resolution on the interface between bright and dim objects.

Further, the Local Intensity Scale given here requires empirical re-scaling A.4.1 and the setting of minimum thresholds in the presence of noise. These actions seem unsatisfactory, and although similar concepts may be required, ideally they would have a sound theoretical basis.

Despite these limitations, the results, both across benchmarks and real data are encouraging. Showing that the Local Intensity Scale does indeed 'achieve' its intended purpose, and I think they serve as a proof of principle. However, given the limitations above, and the lack of theoretical guarantee of the Reconstruction Condition, further development of the Local Intensity Scale seems warranted.

Synthetic data

Limitations also arise regarding the simplifications that were made for the generation of synthetic data. Although the synthetic data showed varying spatial and local intensity scales, it did not include other of LSFM data discussed in 2.2.4. This includes a lack of spatially varying PSF, more sophisticated noise models, and extraneous background signal. Qualitative evidence from real benchmark data indicates that the APR can still 'work well ' in the presence of these additional features. However, without ground truth images, it is unclear how to accurately assess the APR in these situations. A practical route would be the incorporation of these additional features into more realistic synthetic data for testing. Indeed, concurrent unpublished research has been successfully undertaken by colleagues at the MPI-CBG to generate such data sets. Hence, evaluation using this resource would seem invaluable to addressing these issues and also those of the Local Intensity Scale.

Parameters

Another limitation of the above results includes the use of knowledge of the synthetic benchmark to set parameters. For example, for the B-Spline smoothing parameter λ, using the background intensity level for a noise estimate. Further, for the Local Intensity Scale, the minimum object brightness was used for σ_{th}. The use of this information would seem to bias the above results, as such information is not 'readily' available from real data. However, we have found that the results above are relatively insensitive to the exact setting of these values. With the exception, of σ_{th} being set below the smooth noise level, or greatly above the brightness of the objects in the image.

Ideally, these parameters would all be estimated from the input image. This is an issue that is linked to both the generation of more realistic synthetic data, and the form of the Local Intensity Scale, and could be addressed jointly in future research.

Conclusions

From these results above, we can see that additional development is still required for any confident inference of the results regarding the properties of the APR from the benchmarks to real datasets. However, I do believe they provide a promising proof of principle that could form the basis of a robust solution.

6.6.2 Reflection on Representation Criteria

The results for both the properties and performance of the APR shown above were designed to test the first three representation criteria **RC**1-3. Here, we focus on the benchmark results, largely ignoring the limitations discussed above.

Below we assess whether the APR appears to have the correct properties for our synthetic benchmark data to fulfill the representation criteria.

Representation Criteria 1

We showed in 6.4.3, that the size of the APR, i.e. $\#\mathcal{V}$ scales linearly with an increase in objects in a fixed size image. Where we defined the information content of an image to be a function of the number of objects it contains. We then showed that as the image size is increased for fixed image content, the size of the APR (6.4.4) becomes constant. These results indicating that the adaptation becomes independent of the original image size. Lastly, in Section 6.4.5, we showed that as the sampling resolution for a fixed scene is increased, the size of the APR also becomes constant.

Hence, for our synthetic data, we conclude that the size of the APR reflects the information content of the image, and not the image size or sampling and **RC**1 is satisfied.

Representation Criteria 2

In 6.4.1, we showed that the Reconstruction Condition holds for noise-free images, for a range of reconstruction methods while using a non-constant Local Intensity Scale σ. For noisy images, in 6.4.2, we showed that the APR is still effectively adaptive to the image, but, the Reconstruction Condition does not hold. However, we showed that for an optimal range of relative error E, the errors made in the approximation of the APR are within the level of the noise. Further, the reconstructed APR results in an increase in PSNR compared to the original image.

Therefore, we again conclude that for the synthetic images presented here the APR satisfies **RC**2.

Representation Criteria 3

In 6.5.4, we analyzed the execution time of transforming a pixel image into the APR. The results showed the whole pipeline is linear in the number of original pixels N. With the computational cost being dominated by simple filtering operations on the full image. The pulling scheme, the basis of the adaptation, represents a small proportion of the cost, representing less than 2.5% of the total execution time. Further, the pulling scheme is worst-case linear performance, and actual performance proportional to the information content in the image. Given the computational time is within time ranges given within the literature for 'real-time' for LSFM datasets [5, 110], we conclude the transform step is 'fast'.

Therefore, we conclude that the described implementation of the APR on our test system satisfies **RC**3.

Summary

The above chapter has presented evidence that the APR satisfies the first three representation criteria for our benchmark data. In the next chapter, we address the fourth representation criteria regarding using the APR for processing tasks.

Summary of the chapter

- Provided a 1D exemplar, showing the adaptation of the APR, and the impact of the relative error E.
- Confirmed that the Reconstruction Condition holds in 1D for the benchmark example, and showed a reconstruction of the gradient of the function from the APR, and the impact of discontinuities.
- Introduced the APR pipeline for 3D LSFM data
- Described synthetic data generation process used to benchmark the APR
- Showed that for noise-free images the Reconstruction Condition holds and that for noisy images, there exists an optimal range of E that reduce the PSNR of the original image.
- Showed that the size of the APR scales linearly with information content while maintaining image reconstruction quality.
- Showed that the size of the APR becomes independent of both the original image size N, and the sampling resolution used, as both are increased.
- Introduced the Computational Ratio benchmark and examplar benchmark data sets that are used to assess the performance of the APR.
- Introduced and described the data structures that are used for the APR

- Showed that the worst-case execution time for transforming an image to the APR is *linear* in N, and that it is linear in N for a fixed ratio of $\frac{\#\mathcal{L}}{N}$. For a fixed size N the cost scales sub-linearly with the number of seed Particle Cells in $\mathcal{V}$. For fixed for fixed size of $\#\mathcal{L}$ the computational cost is sub-linear in N at a rate inversely proportional to the number of Particle Cells.
- Showed that the APR can be efficiently losslessly stored at a rate larger than the Computational Ratio (CR) of the given APR.
- Discussed limitations of the Local Intensity Scale, synthetic data, and setting of parameters
- Reflected on representation criteria (**RC**1-3) and the evidence of the APR fulfilling them for the synthetic benchmark data.

7 APR Processing

Contents

7.1 Interpretations of the APR for processing 158
  7.1.1 Collocation points and spatial partition 158
  7.1.2 Particle graph 158
  7.1.3 Continuous representation 160
  7.1.4 Tree structure 160
7.2 Evaluating performance 160
7.3 Performance benchmarks 162
  7.3.1 Limitations 162
  7.3.2 Neighbor access 164
  7.3.3 Separable pixel filtering 166
  7.3.4 Segmentation 171
7.4 Visualization 175
  7.4.1 By slice 175
  7.4.2 By racasting 177
  7.4.3 By particle rendering 177
7.5 Novel algorithms 180
  7.5.1 Adaptive APR filters 180
  7.5.2 Segmentation and visualization 183
7.6 Summary and main points 185
  7.6.1 Reflection on Representation Criteria 185

In this chapter, we present results for a selection of basic image processing tasks using the APR. Here we directly investigate **RC**4, and whether the adaptation of the APR can be used to reduce memory and computational costs, and possibly complexity, of a range of image processing tasks. Ideally, we wish to show that if the input image has been transformed into an APR, the input image is no longer needed, and all processing, storage, and visualization can be done directly using the APR.

This chapter is structured as follows. First, we discuss the different interpretations of the APR, and how this relates to different processing tasks for pixels. Next, we discuss how to evaluate the performance of processing tasks on the APR, and propose evaluation metrics. Then, we present four performance benchmarks. Namely, linear neighbor access, random neighbor access, separable pixel filtering, and segmentation. For all these examples we evaluate the performance with respect to comparable pixel algorithms and use both the CR (6.5.1) and exemplar (6.5.2) benchmark data defined in the previous chapter. Following that, we show various methods for visualization of the APR. Lastly, we discuss briefly how the APR can be used to create novel algorithms that utilize its adaptation, giving an Adaptive APR filter and segmentation as examples.

7.1 Interpretations of the APR for processing

Traditional processing tasks have been developed using, and rely on, a range of interpretations of images. By this, we mean the same pixel image could be interpreted in many ways. For example as a graph, collocation points of a continuous function, spatial partitioning of square pixels, or the highest resolution of a tree structure.

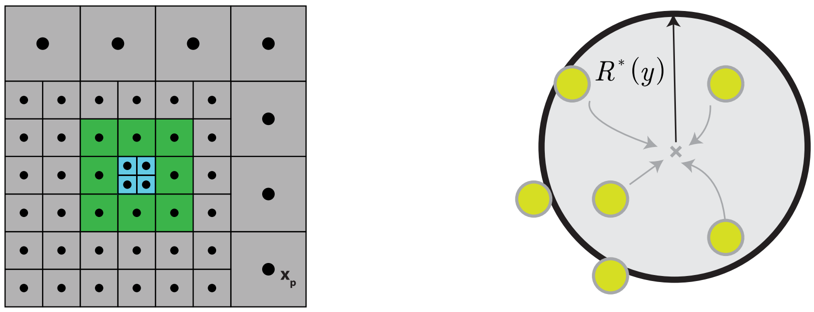
Just like pixels, we can also, interpret and use, the APR in different ways depending on the particular processing task. These interpretations align with many of those commonly used in pixel-based processing. Figure 7.1, shows a schematic representation of the four main APR interpretations we discuss below.

7.1.1 Collocation points and spatial partition

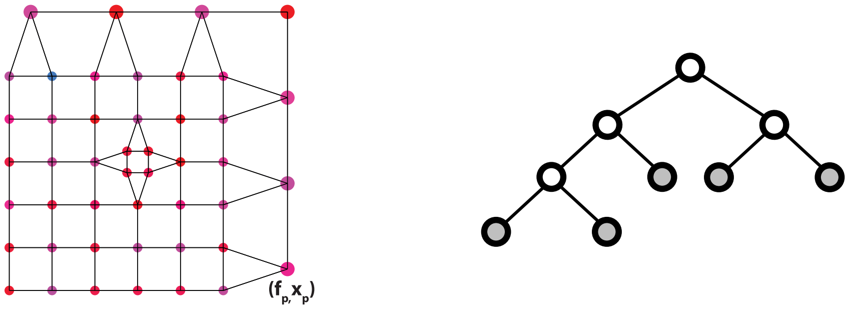
The *top right* schematic shows the APR formulated as an adaptive spatial partition, from the Particle Cells $\mathcal{V}$, and point estimates of the intensities, from the particles. Represented in this way, we could view the APR as adaptively sized pixels, or restricted superpixels [2]. Similarly, a lower resolution partition can be used, utilizing $\mathcal{V}_n$, combined with particle cell type. This interpretation is utilized the APR raycasting example below.

7.1.2 Particle graph

The *bottom right* shows the APR formulated as a particle graph (as described and defined in 5.6.2), where particles are nodes, and neighbors and edges are defined using the Resolution Function. This provides an analog to face-connected graphs on pixels often used for graphical models [21]. We use this



Collocation points and volumes Continuous Reconstruction



Graph Tree Structure

Figure 7.1: In the *top right* we show the APR represented as a spatial partition from the Particle Cells in $\mathcal{V}$ and a set of particles, which are collocation points. In the *bottom right* schematic we show the APR represented as a graph, where the particles are nodes, and edges link particles as discussed in 5.6.2. In the schematic in the *top left*, we show how the APR can be used to provide a continuous reconstruction of an image everywhere across the domain. Lastly, in *bottom left* we show a schematic of the APR as a pruned binary tree (quadtree in 2D, or octree in 3D)), where the links, are between parent and child Particle Cells.

interpretation in the APR segmentation and APR adaptive filter examples below.

7.1.3 Continuous representation

The *top left* schematic shows how the APR interpreted as a continuous function represented adaptively at collocation points. The continuous function can then be reconstructed at any point using 5.1. This reconstruction can be done locally, such that a full reconstruction of a high-resolution pixel image is not necessary. We use this interpretation in the separable pixel filters below.

7.1.4 Tree structure

Lastly, the *bottom left* shows the APR represented as a pruned tree structure. The APR has a natural tree representation due to the parent child relationships of Particle Cells (as seen in Figure 4.8). If a tree is constructed using the Particle Cells of an APR, and including all parents of Particle Cells in $\mathcal{V}$ then, the resulting tree structure, can be interpreted as an adaptively pruned tree. This tree structure could be useful for using the APR with wavelets, or multi-resolution methods such as pyramid methods [4]. We utilize such a tree structure in the calculation of an energy term in the segmentation below.

7.2 Evaluating performance

The APR can reduce the cost of existing algorithms in two ways. First, by decreasing the total processing time by a reduction in the number of operations that have to be executed, and second by reducing the amount of memory required to run the algorithm. The relative importance of the two and the degree of reductions depend on the nature of the algorithm and its implementation. Below we introduce quantitative evaluation metrics to evaluate the improvements in different algorithms and input images.

The first evaluation metrics relate to the computational performance, or speed up, of the algorithm. For a given algorithm and implementation we define the SpeedUp (SU) as,

$$\mathrm{SU} = \frac{\text{Processing time of the algorithm on pixels}}{\text{Processing time of the algorithm on particles}}. \tag{7.1}$$

Which is useful to relate to the Computational Ratio (CR, see 6.11), by

$$\mathrm{SU} = \mathrm{CR} * \mathrm{PP}, \tag{7.2}$$

where PP is the Pixel-Particle Speed Ratio and defined as

$$\mathrm{PP} = \frac{\text{Time to compute the operation on a pixel}}{\text{Time to compute the operation on a particle}}. \tag{7.3}$$

The PP reflects the relative cost of computing an operation on a particle from an APR vs. a pixel from the input image for the equivalent task (value less than one indicates pixels are faster). The value of PP depends on many factors, including memory access patterns, data structures, hardware and absolute size of the required data in memory. Consequently, even for a set algorithm running on fixed hardware, the PP will be a function of both the input image size N, and Computational Ratio CR. Therefore, for tasks where $PP < 1$, as in most low-level tasks, there will be a minimum value of CR for which the algorithm is faster on the APR than pixels. Therefore, understanding the speed up for algorithms requires knowledge of PP, CR, and N.

The second set of evaluation metrics relates to the reduction in memory. We define the Memory Reduction Ratio (MRR) as

$$\mathrm{MRR} = \frac{\text{Memory Cost (MC) for pixel algorithm}}{\text{Memory Cost (MC) for particle algorithm}} \tag{7.4}$$

Expressed using the CR gives

$$\mathrm{MRR} = \mathrm{CR} * \mathrm{MPP} \tag{7.5}$$

where

$$\mathrm{MPP} = \frac{\text{Memory required per pixel}}{\text{Memory required per particle}}, \tag{7.6}$$

and reflects the relative memory cost of a single pixel vs. particle. For an algorithm on a pixel image, the Memory Cost (MC) in Bytes usually scales directly with the number of pixels N and variables, as

$$\mathrm{MC}_{pixels} = (\text{Number of variables}) * (\text{Data type in Bytes}) * N. \tag{7.7}$$

In comparison, the APR additionally requires storage of additional information for location and neighbor access. This overhead depends on the specific data structure used. We provide a detailed analysis of the MC for the two data structures used here in A.5. In summary, for the APR in Bytes,

$$\begin{aligned}\mathrm{MC}_{APR} = {} & (\text{Number of variables}) * (\text{Data type in Bytes}) * N_p \\ & + (\text{Cost of data structure per particle}) * N_p,\end{aligned} \tag{7.8}$$

where N_p is the number of particles, and the cost of the data structure per particle depends on N. Therefore, as the number of variables increases, the additional overhead of the APR data-structures is amortized so that MPP will approach 1 and the MRR the CR.

7.3 Performance benchmarks

To demonstrate potential range of performance, we analyze three low-level and one higher-level processing task below. The three low-level tasks are linear and random neighbor access and separable pixel filtering, and the higher-level task is segmentation. These low-level tasks, in addition to being core to many processing algorithms, also represent a lower-bound for the performance benefits of the APR due to their simple operations and access patterns that are well suited to processing on pixels. In contrast, for the segmentation task, we use an external min-cut library with more irregular computation and access patterns provides an example of an algorithm more suited to the APR. A.10.3 describes the implementation and details of the benchmarks, and in the main text, we focus on discussion of the results.

For the four performance benchmarks, we provide results for the computational and memory evaluation metrics and total execution time for the CR benchmark data for input images from $N = 200^3$ up to $N = 1000^3$ and summarize the performance on the exemplar data sets. [1] We also discuss differences between the APR and pixel algorithms, and where appropriate provide additional results.

7.3.1 Limitations

We note that all these results are highly implementation dependent, for both the APR and pixel results. Ideally, exactly equivalent algorithms could be constructed and compared. Unfortunately, due to differences between the data structure used for an image, a contiguous array, and the more complicated data structure of the APR, this is not realizable. We have endeavored here only to provide equivalent optimization across both APR and pixel algorithms. Further, this is in part the motivation for the simple nature of the benchmarks used below. We only show results here using the SA, and SARA APR data structures. It is likely that alternative data structures or improvements could provide improved results. In this way, the results below can be thought of as an indicative lower bound on performance. The same argument can be applied to the pixel implementation and data structures. However, it is likely that the pixel algorithms are closer to their 'optimal' form.

---

[1] Note, that smaller sized original images have been omitted from the results as the CR could not be accurately adapted to reflect the CR of each benchmark dataset and hence confused the results.

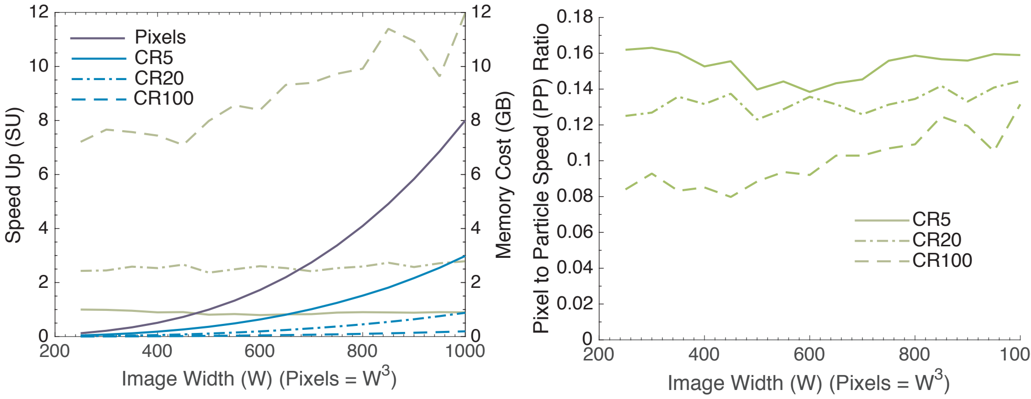


Figure 7.2: The first plot on the *left axis* shows the mean Speed Up (SU) for the CR benchmark data against image width for **linear neighbor access** in green (The dashed lines follow the same convention for both datasets). The *right axis* shows the Memory Cost (MC) of the algorithm in GB plotted in blue for the APR and pixel implementations. The second plot for the same data shows the Pixel to Particle Speed (PP) ratio against image width

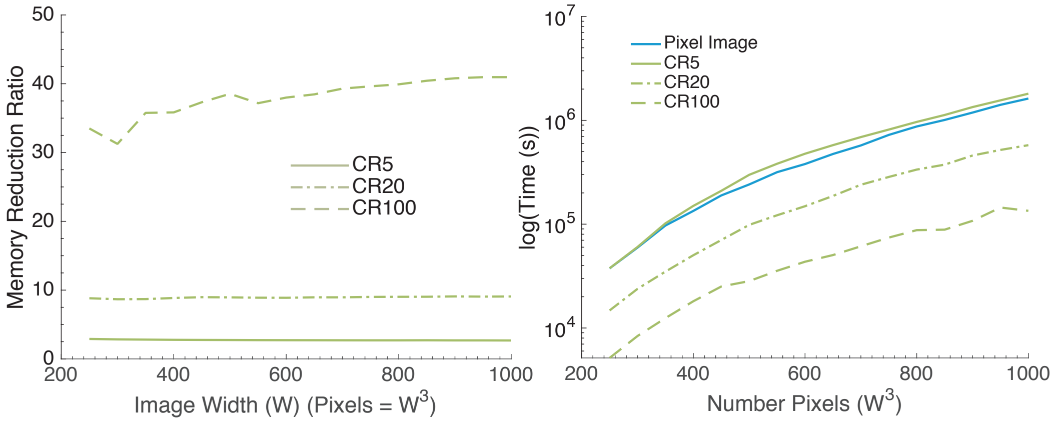


Figure 7.3: The first plot shows the Memory Reduction Ratio (MRR) for the CR benchmark data against the image width for **linear neighbor access**. The second plot shows the natural logarithm of the average execution time for the same data.

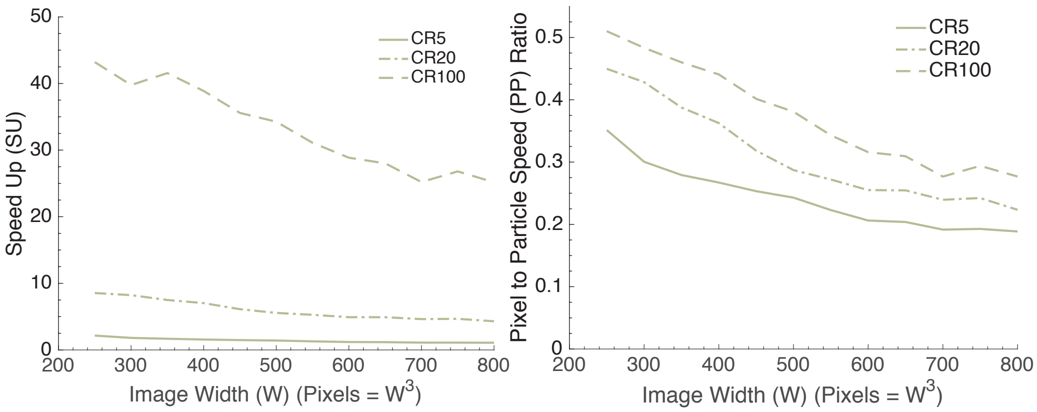


Figure 7.4: The first plot on the *left axis* shows the mean Speed Up (SU) for the CR benchmark data against image width for **random neighbor access** in green (The dashed lines follow the same convention for both datasets). The *right axis* shows the Memory Cost (MC) of the algorithm in GB plotted in blue for the APR and pixel implementations. The second plot for the same data shows the Pixel to Particle Speed (PP) ratio against image width.

7.3.2 Neighbor access

The first task we benchmark is that of neighbor access. Neighbor access is at the core of many image processing algorithms. For each pixel and particle, the task involved accessing, and then summing, the intensity of all face-connected neighbors and storing the result. The APR neighbors were defined using the 3D the APR particle graph as described in 5.6.2. The particle graph is simply the face-connected neighbors of the Particle Cells. We benchmarked two forms of neighbor access. The first we call *linear access* requires iterating over all pixels, or particles, in memory order and accessing all neighbors. The second, we call *random access*, involved randomly iterating over pixels, or particles, and accessing the neighbors. These two benchmarks allow us to assess the two 'extremes' of neighbor access patterns. See A.10.4 for an description of the implementations.

Computational cost

The left axis of the first plot of Figure 7.2 shows the SU for the linear access benchmark for the CR benchmark data. The average SU for CR5 is 0.897, CR20 is 2.386, and CR20 is 7.77. These low SUs and slow down for CR5 reflect the relative efficiency of performing neighbor access on pixel images. The PP values also reflect this, with average values showing it is between six and ten times faster to perform neighbor access on pixels (Figure 7.2 second

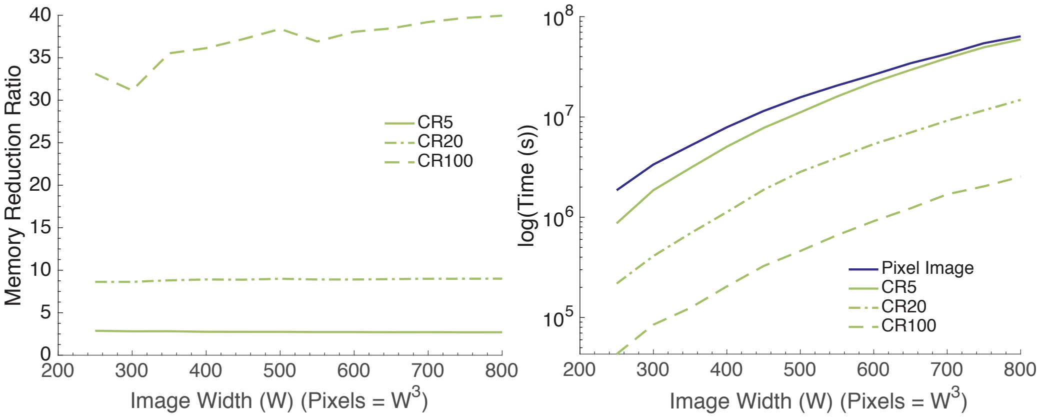


Figure 7.5: The first plot shows the Memory Reduction Ratio (MRR) for the CR benchmark data against the image width for **random neighbor access**. The second plot shows the natural logarithm of the average execution time for the same data.

plot). Interestingly, the higher the CR, the higher the PP. In the second plot of Figure 7.2, we show the natural logarithm of the execution times. For the CR20 and CR100, we see the positive SU values reflected in faster execution times for the APR. However, for CR5 for larger images, as reflected in the SU, the linear access is slower (although comparable magnitude) on the APR than on pixels. The first plot in Figure 7.6 gives the SU values for the exemplar data sets. One dataset, the smallest, shows a slowdown (SU$<$ 1), with an SU value of 0.7075. However, all other exemplars show speedups with a mean SU of 4.28 and median 2.76. In contrast, although significantly slower for both pixels and particles in total execution time, the random neighbor access benchmark showed larger SUs for the APR. The left axis of the first plot in Figure 7.4 shows the SU across CR and image size. The average SU for CR5 is 1.4, CR20 is 8.5, and CR100 is 28. We can understand the increase in SU compared to linear access further from the PP values shown in the second plot. The increase relative to linear access reflects the change in memory access patterns. With the pixel image no-longer having a cache advantage due to the layout of pixels in memory. Further, the smaller data set size for the APR likely improves cache efficiency. However, we see that the SU and PP values reduce as the original image size increases. This likely reflects the decrease in cache benefits from the small size of the APR. In the second plot of Figure 7.5 we show the natural logarithm of the execution times. In contrast, the APR total time is less than the pixels across image size and CR. For the exemplar SU, shown in the second plot in Figure 7.4 also reflects the increased relative performance with a mean SU value of 18.8 and median of 12.25.

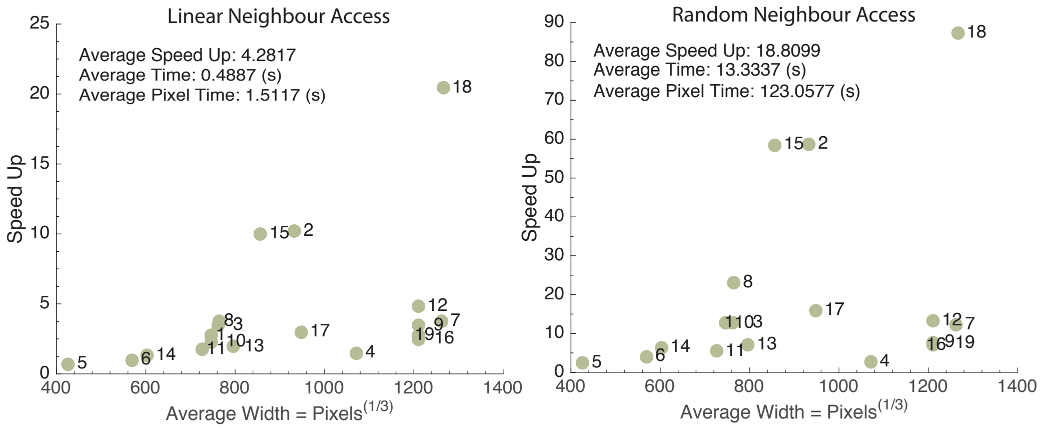


Figure 7.6: The first plot shows the Speed Up (SU) for the exemplar benchmark data plotting against image width for the **linear neighbor access** performance benchmark. The second plot shows the same data but for the **random neighbor access** performance benchmark.

Memory Cost

We present the absolute values in GB of the memory cost on the right axis of the first plots in Figure 7.2 and Figure 7.5 for the linear and random neighbor access benchmarks respectively. The memory cost for both benchmarks is identical, as they both required the original data and storage of the result. The impact of the CR on the MC can be seen, with the CR20 and CR100 datasets requiring roughly an order of magnitude less memory. The first plots in Figure 7.3 and Figure 7.5 show the Memory Reduction Ratios (MRR). Across image sizes and CRs, consistent reductions in memory were observed. The average MRR of 2.751 for CR5, 8.376 for CR20, and 40.96 for CR100. The MRR values for the exemplar datasets, not plotted here, also show an improvement in memory cost with a mean MRR value of 24.4 and median of 12.2.

7.3.3 Separable pixel filtering

In the third performance benchmark, we assess the task of separable image filtering, using a Gaussian blur. The Gaussian blur kernel is taken to be defined in 3D and over pixels. The filtering is separable in that we perform the task using three consecutive filtering steps applying a 1D filter in each direction. Doing this sequence of filter steps gives the same result as using the full 3D kernel once, and is commonly used as the computational cost scales as $3L$ compared to L^3 when L is the filter length in one dimension (although,

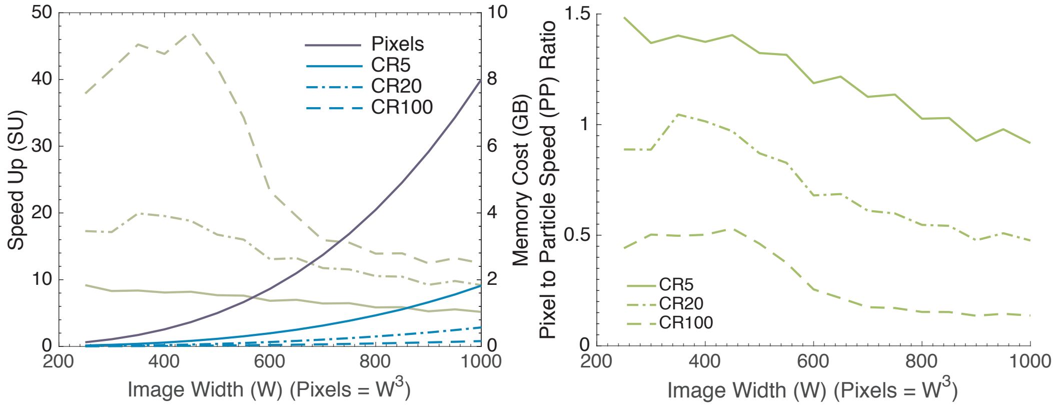


Figure 7.7: The first plot on the *left axis* shows the mean Speed Up (SU) for the CR benchmark data against image width for **separable pixel filtering** in green (The dashed lines follow the same convention for both datasets). The *right axis* shows the Memory Cost (MC) of the algorithm in GB plotted in blue for the APR and pixel implementations. The second plot for the same data shows the Pixel to Particle Speed (PP) ratio against image width.

not all 3D filters are separable). Since the APR does not have particles at each pixel location, for these sites, we use a reconstructed intensity. However, instead of interpolating the full image, only the appropriate slice needs to be reconstructed at any time. We then compute the filter on the reconstructed slice only at particle locations. We only present results for a fixed filter length L, however, the comparative results for different L are similar. For simplicity and performance, we have used the piecewise constant interpolation. A more detailed description is given in A.10.5.

Computational cost

The left axis of the first plot of Figure 7.7 shows the SUs for the separable pixel filtering task for the CR benchmark data. We find a mean SU value of 6.923 for CR5, 13.16 for CR20, and 24.93 for CR100. The results all show a peak SU for mid range image widths $200 - 400$ then converging to lower values as W increases. These dynamics are likely the result of the slice reconstruction. This step has a fixed cost with respect to N, not CR, and the cost becomes proportionally more expensive with an increase in image size. The dynamics with W are more clearly illustrated with the PP, showing a peak and then decrease of performance as shown in the second plot of Figure 7.7. Again, we see that the PP ratio is higher for higher CR, reflecting cache efficiency benefits of larger high-resolution regions, and the lower per particle cost of the reconstruction step. In the second plot of Figure 7.8 we show the natural log of

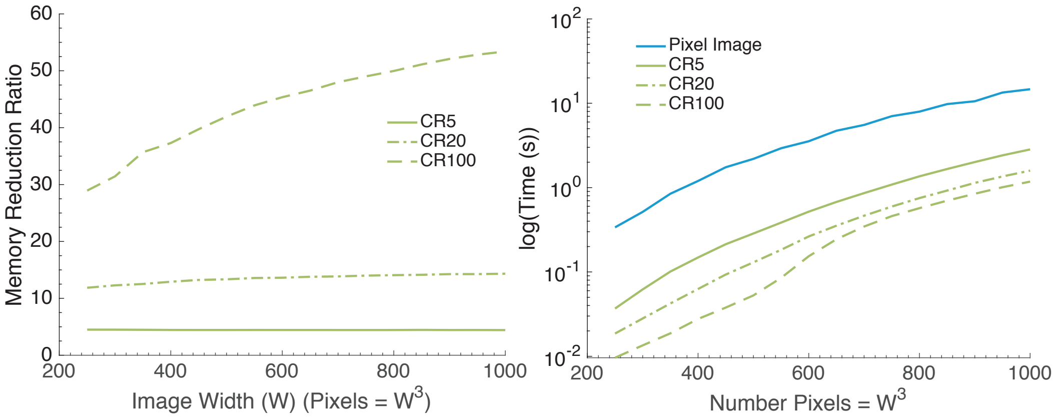


Figure 7.8: The first plot shows the Memory Reduction Ratio (MRR) for the CR benchmark data against the image width for **separable pixel filtering**. The second plot shows the natural logarithm of the average execution time for the same data.

the execution times for the APR and pixel algorithms, showing the consistent SUs across W. The SU for the exemplars benchmark data is shown in the first plot of Figure 7.9 with a mean SU of 12.27, and mean time of 1.23 and 14.13 seconds for the pixel and APR algorithms respectively.

Memory Cost

Regarding the memory cost, we find similar results as the neighbor access task. For the pixel algorithm, the memory requirements are again simply the input image and result. The APR similarly requires storage of the APR input and result but also an image slice for the reconstruction step. The MC reductions shown on the right axis of the first plot in Figure 7.7 and Figure 7.8. For the fixed CR datasets, the MRR values are 4.35 for CR5, 12.35 for CR20, and 38.02 for CR100. These results, reflect the additional overhead of the image slice having lower impact per particle for a larger CR. The exemplar data sets had a mean MRR value of 24.48 and median 19.09. In absolute values, this corresponds to an average memory cost for pixels of 7.47 GB and the APR 359 MB.

Comparison between approaches

However, the results from these two approaches are different and do not present a fair comparison. To evaluate differences we compared the result of the pixel filter, and the APR filter interpolated to an image, to the ground truth image filtered by the pixel filter for 100 realizations for a fixed image size (250^3,

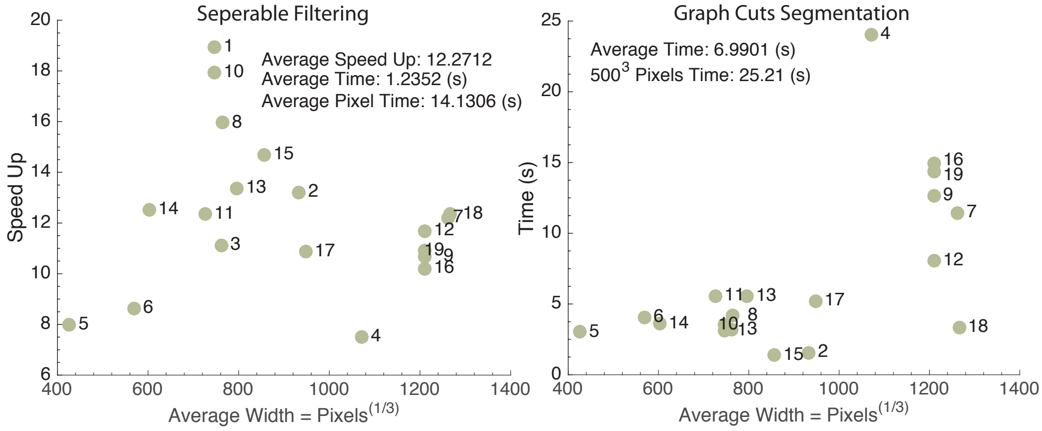


Figure 7.9: The first plot shows the Speed Up (SU) for the exemplar benchmark data plotting against image width for the **seperable pixel filtering** performance benchmark. The second plot shows the execution time for the **segmentation** performance benchmark for the exemplar benchmark data. The SU was not available, due to the pixel algorithm being unable to be run on the benchmark machine due to memory constraints. The numbers coincide with those in Table A.1.

CR5). For comparison, we also computed the result of the pixel filtering on a piecewise constant reconstruction from the APR. We did this for a small $\sigma = 0.5$ and larger $\sigma = 2$ blurred kernel.

Examples of the results are shown in Figure 7.10. For the small blur kernel, the mean APR PSNR was 30.68 with a standard deviation of 1.49, for the original image 23.38 with a standard deviation of 0.395, and for the reconstructed APR 30.71 with standard deviation 1.26. Therefore, the APR filter provided accurate results. However, in the large blur kernel, the mean APR PSNR was 31.883 with a standard deviation of 2.89, for the original image 42.93 with a standard deviation of 0.6, and for the reconstructed APR 42.56 with standard deviation 1.14. These results indicate that the APR filter no longer produces accurate results. The poor performance can also evident in the bottom right image in Figure 7.10 where distinct artifacts can be seen. This is the result of the Implied Resolution Function not being valid for the intermediate filtering results in the separable scheme. In such cases, it seems it would be necessary to result to pixel filtering on a reconstructed APR, as those results are equivalent for larger filters. This also indicates that the failing is not due to the piece-wise constant reconstruction. The poor performance is not a pure function of filter size, but depends on the spatial scales of the resulting function, with large filters designed for edge detection not showing similar issues (not shown).

Hence we conclude, that the above approach is only of specific use, and does not represent a direct replacement for pixel filtering. The example also illustrates the care that must be taken when adapting algorithms from pixels

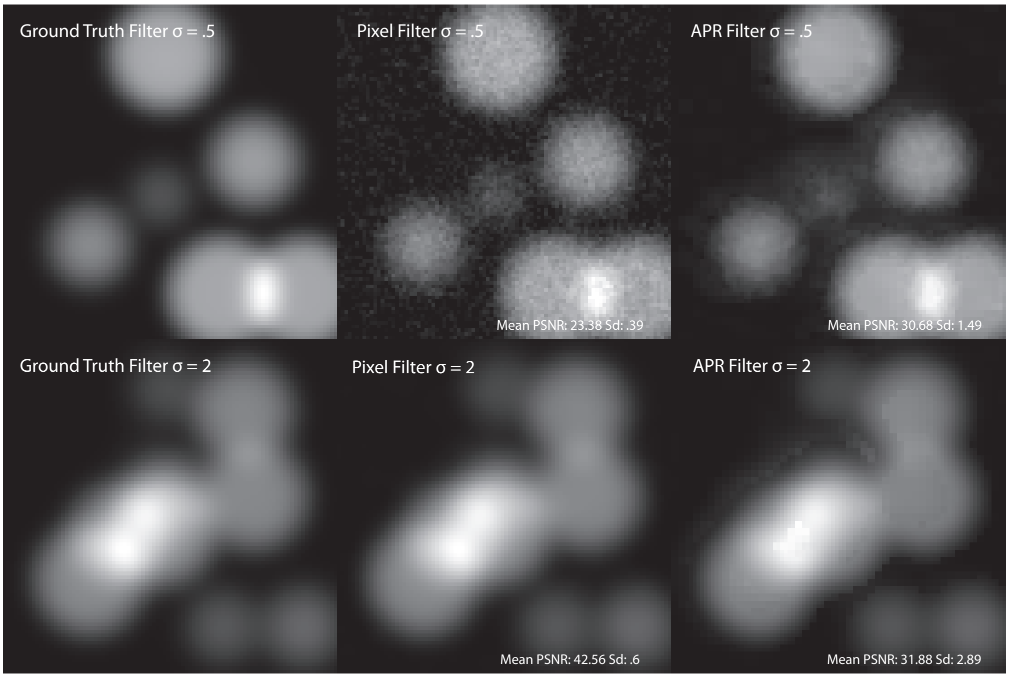


Figure 7.10: Example images of the validation for separable filtering benchmark. The APR and pixel algorithms were run using a narrow Gaussian (*top*) and broad Gaussian (*bottom*), for the pixel filter for noise-free ground truth image (*left*), the original image (*center*) and the APR filter the reconstructed to an image (*left*). The mean and standard deviation of the PSNR compared to the ground truth filter over 100 for a CR5 image of size 250^3 is provided at the bottom of the images. The results show that the APR filter works well for small blur kernels. However, as the kernel gets larger the APR filter no longer produces accurate results, reflected in the low PSNR compared to comparable the pixel filter.

to the APR.

7.3.4 Segmentation

The last performance benchmark is binary segmentation using graph cuts. The binary segmentation task involves estimation of the compactly supported region in the Object function, as described earlier in Section 2.5. This benchmark serves two purposes, showing the benefit of the APR for more sophisticated processing tasks, and also giving an example of the APR being used directly with an existing algorithm and implementation.

We compute the segmentation using graph cuts using an implementation of a min-cut maximum flow algorithm provided with Boykov and Kolmogorov [21]. The algorithm takes an input energy for a node belonging to a source and a sink, and energy between its neighbors, and using this partitions the nodes into either source or sink nodes. Where we define the source set to be those pixels or particles inside the support of our Object function.

We computed the energy terms using both the Particle Cell properties of level and type, and a local min-max range computed using an APR tree structure. The energy used is described in detail in A.10.6. To allow for direct comparison between the pixel and particle results, we use the same energy, using the piecewise constant reconstruction of quantities for the pixel algorithm. This allows us to isolate the differences between the two representations rather than energies. For both the APR and pixel image a face-connected graph was used. Because we use the same energy for both, we remove this from the analysis and only show the results for the solving of the solution through the min-cut, maximum flow step. Unfortunately, due to memory constraints on the benchmark machine, the pixel algorithm could only be run up to image sizes of 550^3, thus limiting the results that require direct comparison.

Computational cost

The left axis on the first plot in Figure 7.11 shows the SUs for the segmentation benchmark for the CR benchmark data. We find a mean SU value of almost exactly the CR, with 5.2 for CR5, 19.66 for CR20, and 92.4 for CR100. In the second plot of Figure 7.11 we see that this results in PP values very close to one. Therefore the cost between the two approaches per node is roughly the same. The equivalence is not obvious, as the APR particle graph can have up to 24 nodes, and has a different structure (However, the average number of neighbors is approximately 6.23 for the CR benchmark data). The second plot in Figure 7.12 gives the natural logarithm of the execution time. We could not compute SU values for the exemplar data sets instead, as the

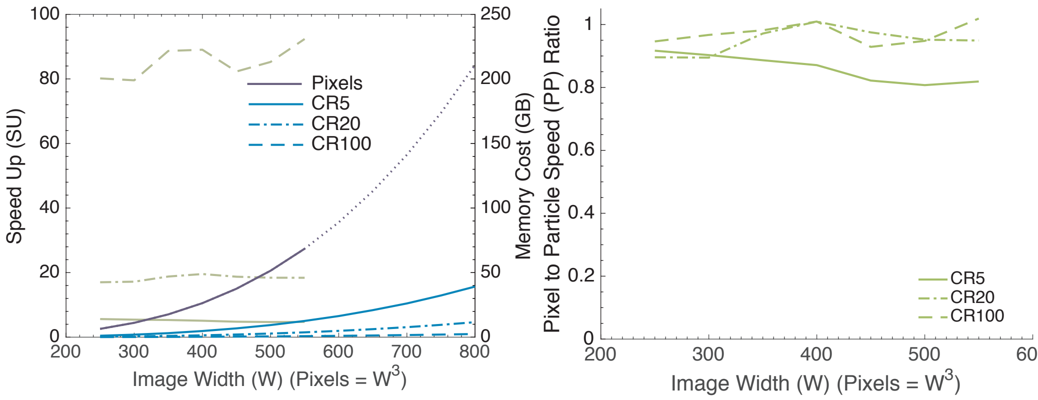


Figure 7.11: The first plot on the *left axis* shows the mean Speed Up (SU) for the CR benchmark data against image width for **segmentation** performance benchmark in green (The dashed lines follow the same convention for both datasets). The pixel algorithm could not be run due to memory constraints on the pixel data set for images larger than 550^3. However, we have extrapolated the projected memory cost. The *right axis* shows the Memory Cost (MC) of the algorithm in GB plotted in blue for the APR and pixel implementations. The second plot for the same data shows the Pixel to Particle Speed (PP) ratio against image width. Again, data is only available for images smaller than 550^3.

pixel algorithm could not be run. Instead, the second plot in Figure 7.9 gives the absolute time for the APR solution. With an average execution time for the exemplar benchmark data of 6.99 seconds. For reference, this compares with 25.21 seconds taken for the 500^3 pixel algorithm. We note that in this benchmark neither the pixel nor APR data structures are used. Instead, both using the graph data structures of the graph cut implementation. The use of the same data structure effectively accounts for the results above.

Memory Cost

As mentioned due to high memory requirements of the graph cuts solver (on our benchmark machine (64 GB RAM)) pixel results were only able to be computed for images up to 550^3. In contrast, we could compute the solution for all but two (CR5 $> 900^3$) CR and all of the exemplar benchmarks using the APR. The right axis of the first plot in Figure 7.11 shows both real and projected memory cost. The high memory cost for the pixel algorithm due to the algorithm storing edges for both the pixels and links to their neighbors. The first plot of Figure 7.12 shows the MRR scales almost linearly with CR. For the exemplar datasets, the mean MRR value was 39.7 and median 26.9. Corresponding to an average absolute memory cost of 13.5GB, compared to an estimated 384.7GB for the pixel equivalent. As as a note, these results are not given as a criticism of the max-cut memory cost. Indeed, optimized graph

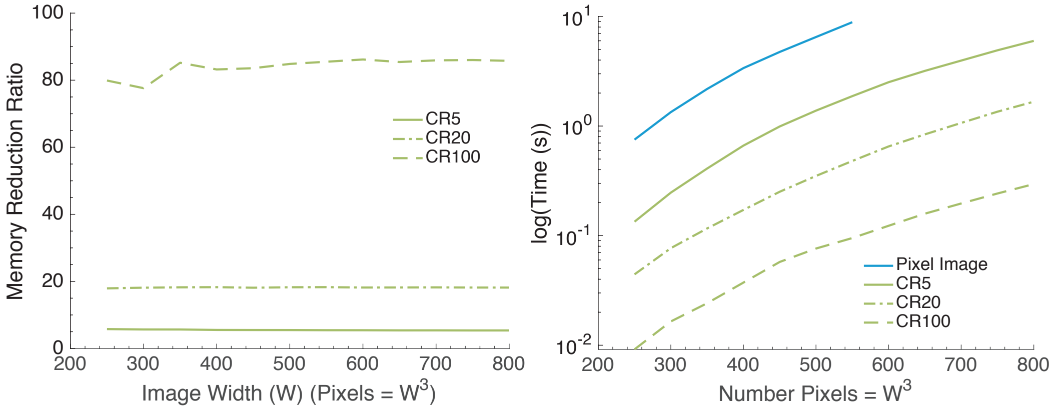


Figure 7.12: The first plot shows the Memory Reduction Ratio (MRR) for the CR benchmark data against the image width for the **segmentation** performance benchmark. Those values for $W > 550$ represent estimates from extrapolation of the pixel memory cost. The second plot shows the natural logarithm of the average execution time for the same data; we note that the CR5 benchmark data could only be run up to an image width of $W = 850$.

cut solvers do exist for reducing this memory cost. Instead, we show here that with direct placement of the pixel image with the APR for an existing algorithm, significant reductions in memory cost can be achieved, that result in a range of problems that were otherwise infeasible, feasible.

Comparison between approaches

To validate that the APR segmentation, we compared the APR and pixel image segmentations to ground-truth and calculated the Dice Similarity Coefficient (DSC) [44] for 100 repetitions of a CR5 image of width 250. We define the DSC in our case as,

$$DSC = \frac{\#(S_{gt} \cap S_p)}{\#S_{gt} + \#S_p}, \tag{7.9}$$

where S_{gt} is the set of pixels in the support of the Object function and S_p, is the estimated set of pixels from the segmentation algorithm, and $\#$ indicates the size or cardinality, of a set. To allow direct comparison with the pixel result, for calculation of the DSC a piecewise constant reconstruction from the APR segmentation was used. Figure 7.13, shows an example segmentation, with the original image, ground-truth segmentation, and segmentations results for the APR and pixel algorithm given. The calculated DSCs of both approaches are statistically identical, with the APR having a mean of 0.88 and a standard deviation of 0.069 and the pixel segmentation a mean of 0.87 and a standard deviation of 0.087. The closeness of the results can be seen in a

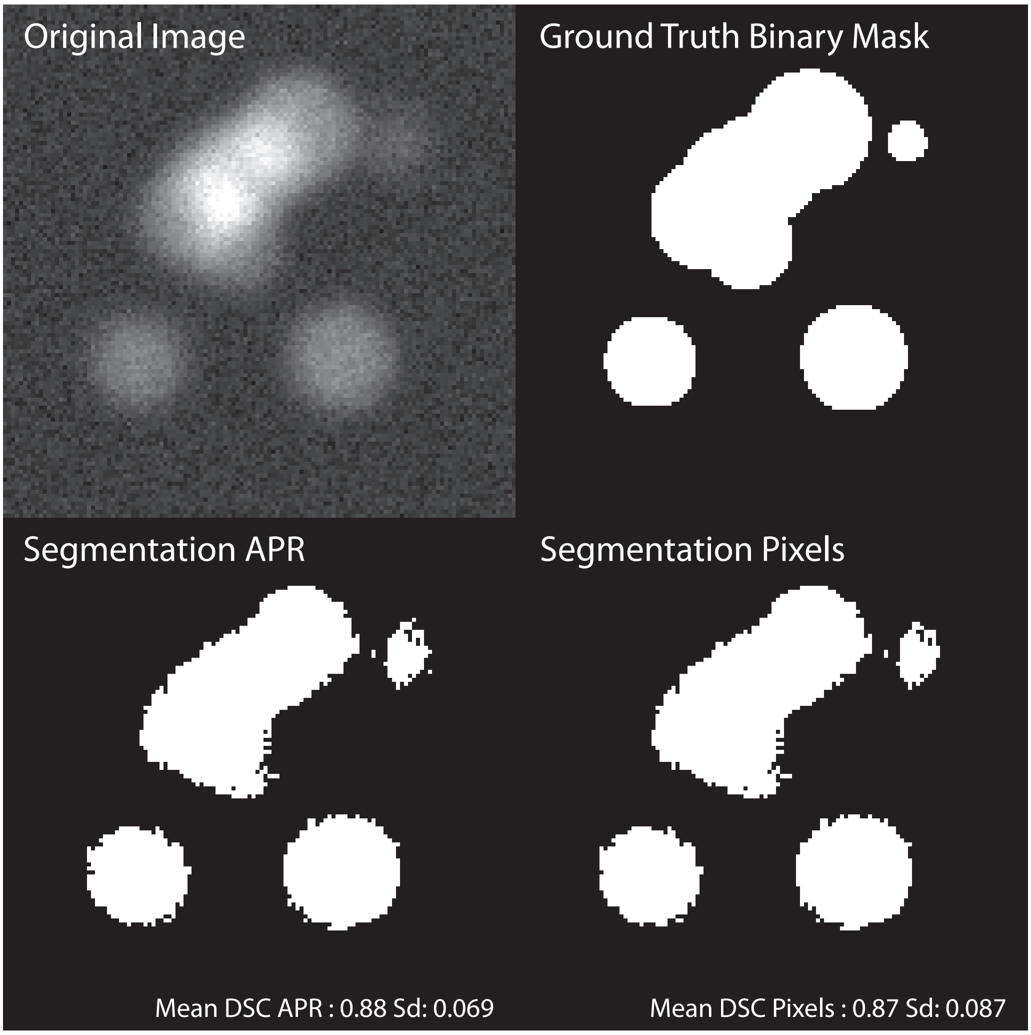


Figure 7.13: Validation for **segmentation** performance benchmark. The same energy function was used for both pixel and APR segmentation. The ground truth binary image is creating using the binarization of the Object function. The APR and pixel segmentation give near identical results as shown by the computed Dice Similarity Coefficients (DSC) mean and standard deviation over 100 repetitions for CR5 images of width 250 given in the figure.

visual comparison, with only isolated pixels being different. From above, we conclude that the results of the two approaches are comparable.

7.4 Visualization

A key processing task using LSFM data is visualization. Both for display of the original image data and any processed results. Visualization is a processing task, as the raw data can not be viewed directly, and must be processed 'in some way' to provide a visual representation. For 3D visualization, this is evident, as the image data represents an opaque 3D cube of integers. However, even for visualization of the original 2D image slices usually requires calculation, or manual setting, of a visual contrast range.

The APR, rather than restricting the visualization possibilities, extends them when compared to the original image data. Given, as that a pixel image representation can be constructed from the APR, this is not surprising. Here, we discuss three different avenues of APR visualization that can be achieved without returning to the full pixel image. We do not benchmark the relative computational or memory performance below, showing the results as proof of principle. We leave the development of efficient implementations and studies of relative performance to future work. A.11, provides additional technical information for each.

7.4.1 By slice

The first methods we discuss, is visualization by reconstruction, on a slice by slice basis. If we only wish to view one slice at a time, the reconstruction can be done on a slice by slice basis. Hence, this does not require having to reconstruct the whole image (as used in the pixel filtering above). In addition to the multiple examples throughout the thesis above, Figure 7.14 gives examples of the APR reconstruction and comparison to the input image. In practice, we find that the piecewise constant reconstruction has been sufficient for visualization purposes. This could easily be implemented in real time per slice, as on a 2013 laptop, reconstructing a 1000×1000 slice took approximately .002 seconds. Although, given correct setting of the contrast control, such piecewise constant representations do show significant 'artifacts' as shown in Figure 7.15. If these are not desired more sophisticated reconstructions could be used as described in 5.6.1.

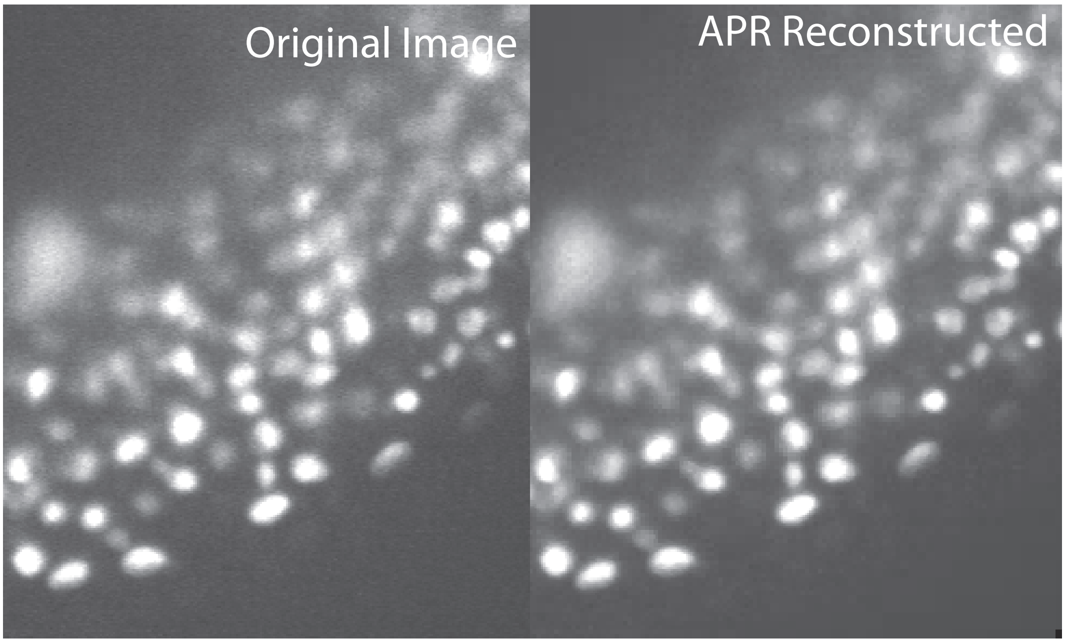


Figure 7.14: Comparison between a subset of a slice of the original image (*left*) and the APR piecewise constant reconstruction (*right*) for exemplar dataset 7 in Table A.1.

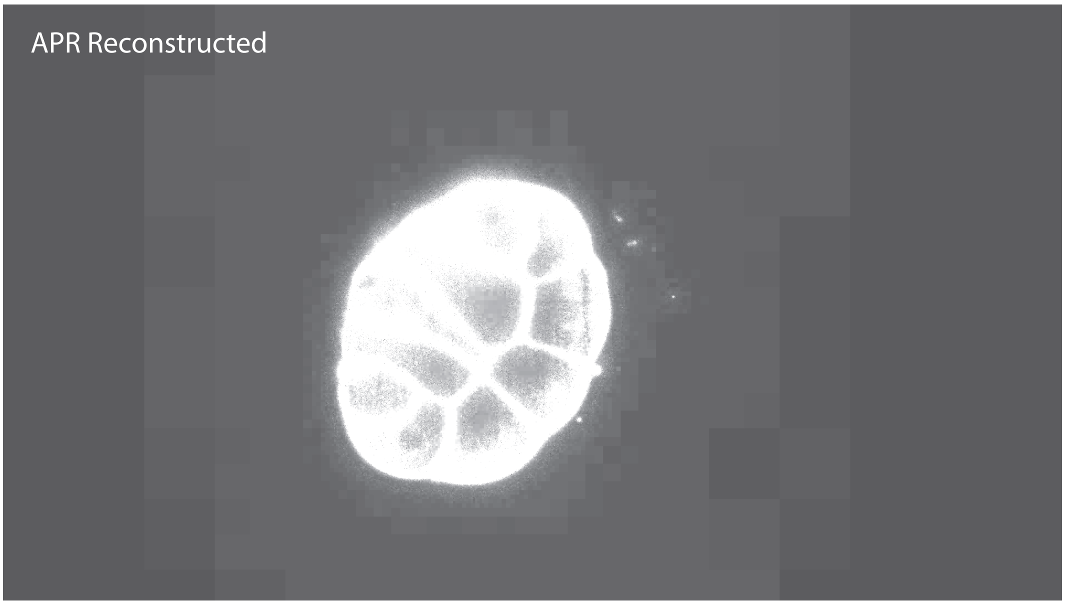


Figure 7.15: An example of piecewise constant APR reconstruction of exemplar dataset 1 in Table A.1. The contrast has been adjusted to highlight the variation over low-resolution areas. This is the same image as presented in Figure 6.10

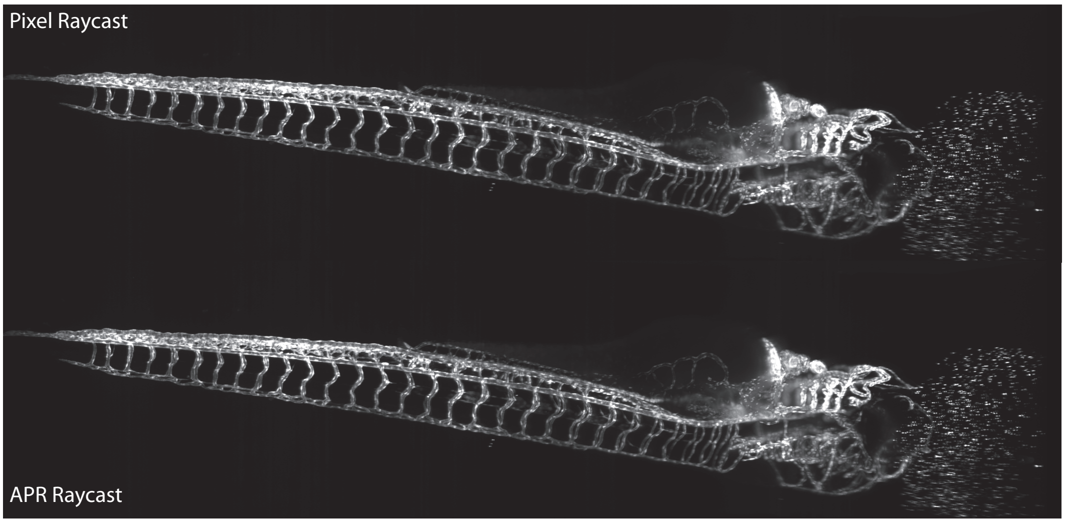


Figure 7.16: Comparison for the same view perspective of an APR (*bottom*) and Pixel (*top*) maximum intensity ray cast of exemplar dataset 17 in Table A.1.

7.4.2 By racasting

A second method allowing for 3D visualization of an APR is ray-casting. Ray casting involves generating a 2D view of a 3D scene, by tracing rays through the data to an observer. Figure 7.16, shows an example of a maximum perspective ray-cast computed on the original image, and direct on the APR for an LSFM data set. Shown at the given contrast levels the two are virtually indistinguishable. However, there are still distinct differences, with Figure 7.17 highlighting them using a different contrast range. A second example is also given in Figure 7.23, showing a visualization of a segmentation result using an extension of the algorithm. The APR ray cast is done by casting multi-resolution rays through the image, level by level, and then combining the results in a final step. The algorithm has a computational and memory complexity that is $\mathcal{O}(N_p)$, only requiring the SA data structure. Such an algorithm could form the basis of useful visualization software, as currently in for the largest images (approximately $> 1000^3$) can not be visualized at the full resolution in the current state of the art software [98] due to memory constraints. Further details of the implementation are given in A.11.2.

7.4.3 By particle rendering

The last visualization method involves direct visualization of the APR. We have given various examples of this in 1D and 2D throughout the thesis above.

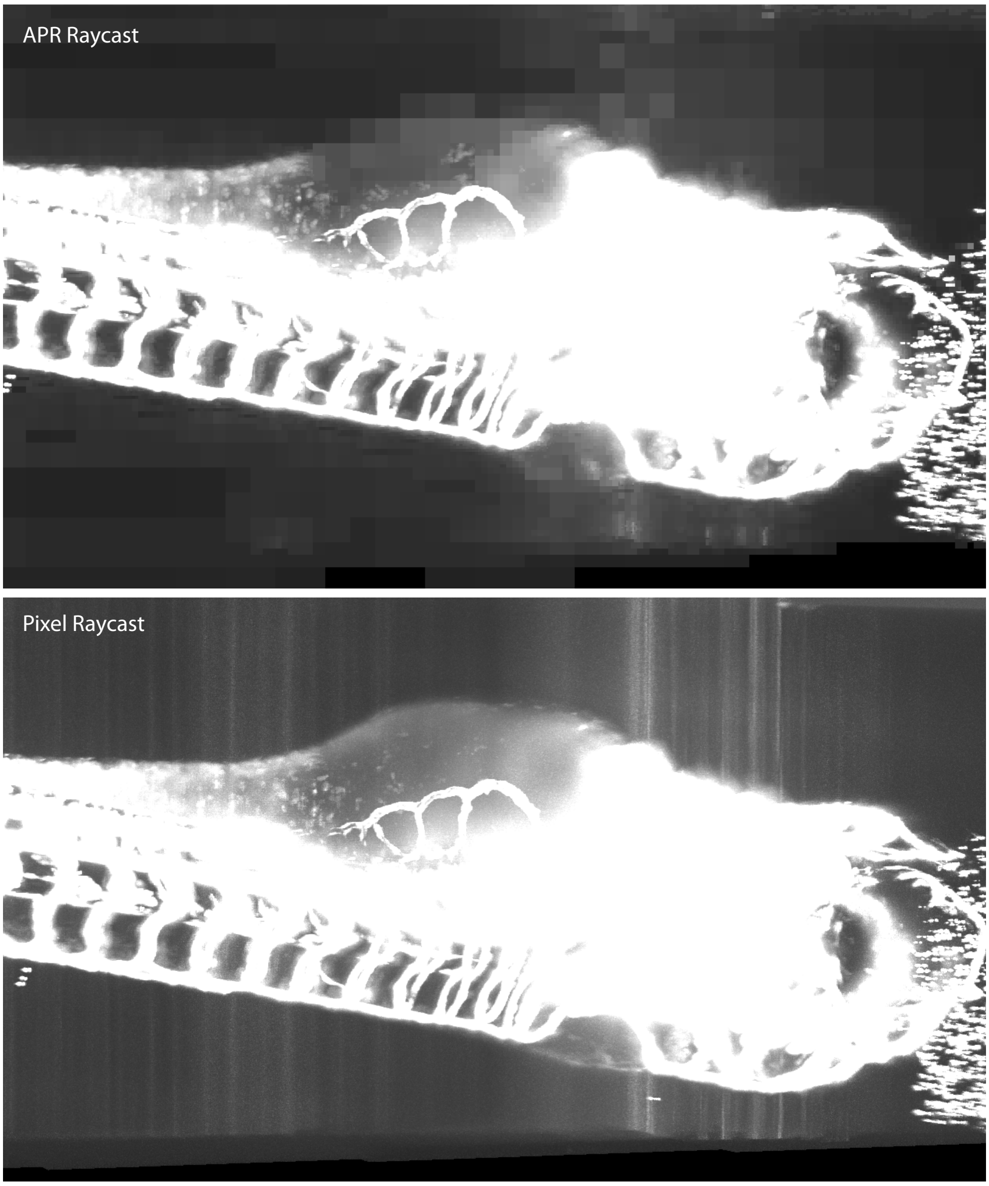


Figure 7.17: Reproduction of the pixel and APR ray-cast example for the same view shown in Figure 7.16. Contrast has been adjusted to highlight the differences and loss of information for the APR ray-cast resulting from an intensity threshold. (Dataset number: 17 in Table A.1)

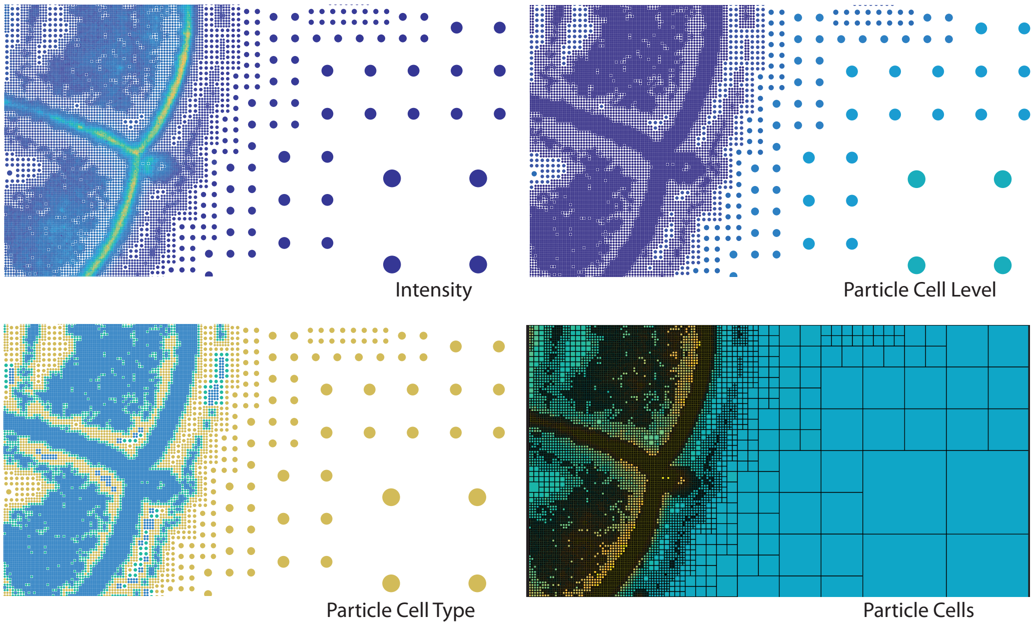


Figure 7.18: An example of the direct rendering of an individual slice of the APR exemplar data set 1 in Table A.1. Three panes show direct particle rendering with size proportional to the Particle Cell level and color reflecting the particle intensity, Particle Cell level, and Particle Cell type across the examples. The last pane shows a direct rendering of the Particle Cells colored relative to their particles intensity.

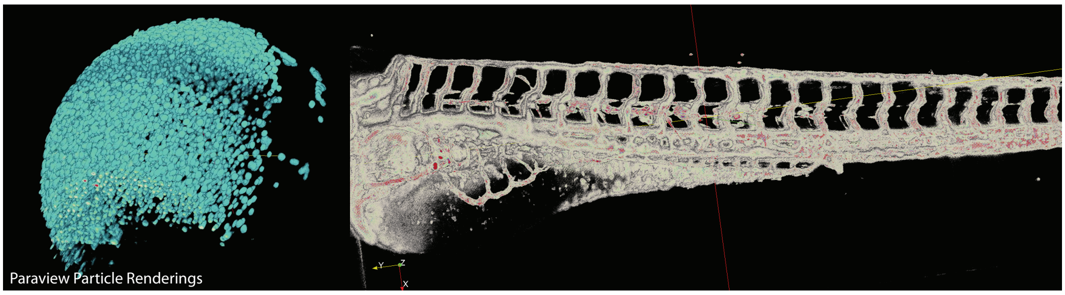


Figure 7.19: Screen captures for examples of direct particle rendering of (*left*) exemplar dataset 7 in Table A.1 and (*right*) exemplar dataset 17 in Table A.1 using Paraview [14]. The particles are coloured by intensity, and an opactiy scaling has been used to remove background particles.

Not only can the intensity be visualized, but also the particle cell level, location, and type. Figure 7.18, shows an example of this for a small portion of LSFM data in 2D. With the use of thresholding or variable opacity, particles can also be directly rendered in 3D. Figure 7.19, shows screen captures of two particle renderings of the APR of LSFM data using Paraview [14].

7.5 Novel algorithms

Some algorithms are naturally better suited for pixel images, as shown by the filter example above. However, the reverse can equally be true. The adaptation, and added information, from particle level and type, naturally provide information about the image. We can directly use this information for processing as illustrated with the segmentation example above. Further, although possible, calculating the same information directly from the image with the APR, would likely come at significant cost and complexity. In addition to providing useful information to existing algorithms, the APR can be used to create novel algorithms. We illustrate this with a simple adaptive APR filter example and segmentation using the approach above with a slightly altered energy.

7.5.1 Adaptive APR filters

Adaptive APR filters are the APR equivalent of the separable pixel filters benchmarked above. Defining the filter over neighboring particles from the particle graph, instead of equally spaced pixels. As the distance between neighboring particles varies across the image, computing an adaptive APR filter is analogous to a spatially adaptive pixel filter with filter size changing to the content of the image.

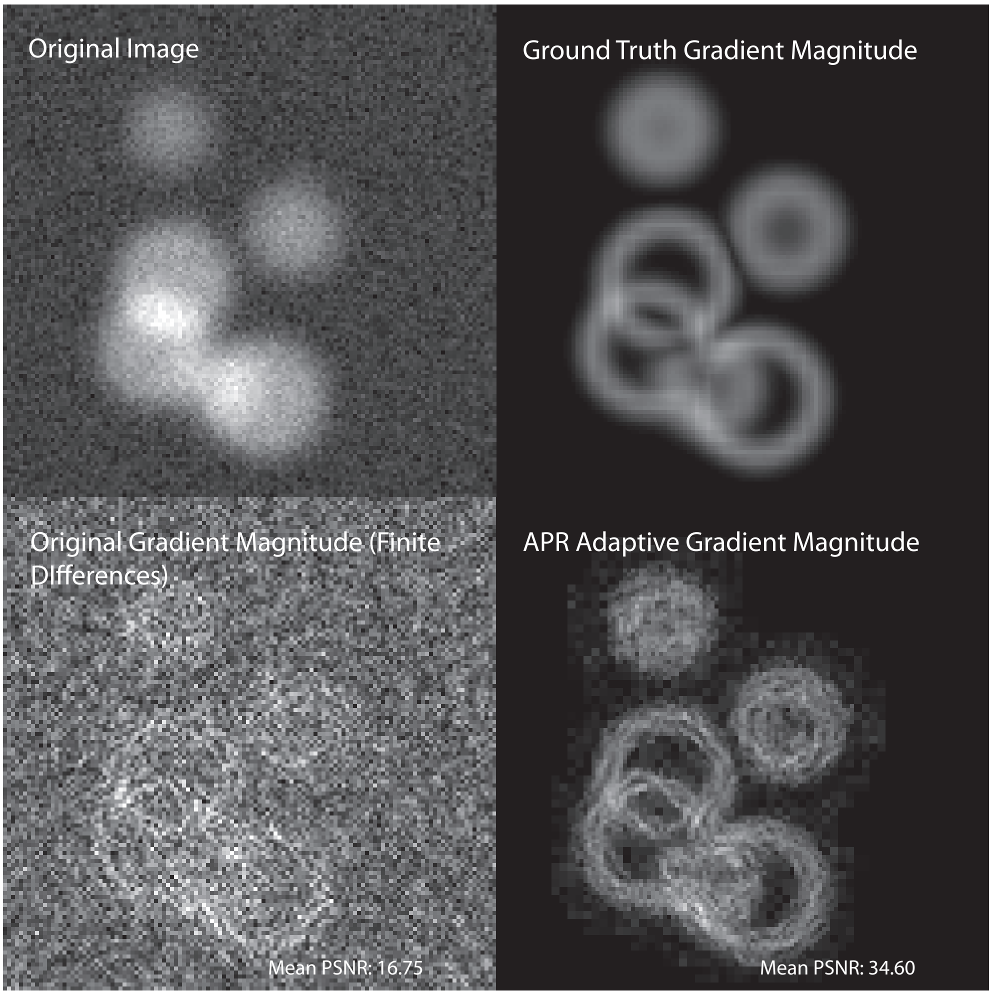


Figure 7.20: Evaluation of Adaptive APR filter used to compute the gradient magnitude with synthetic data, obtained by filtering in each direction with $\{-\frac{1}{2h_-}, \frac{1}{2h_-} - \frac{1}{2h_+}, \frac{1}{2h_+}\}$, where h_+ and h_- are the particle spacings in the positive and negative direction respectively, and multiple particles on one face being averaged. *Top left* original noisy image, *top right* ground truth gradient magnitude using finite differences, *bottom left* original image gradient magnitude using finite differences, and *bottom right* APR Adaptive gradient magnitude. The mean PSNR with respect to the ground truth gradient magnitude is given averaged over 100 repetitions, for CR5 $N = 250^3$ images.

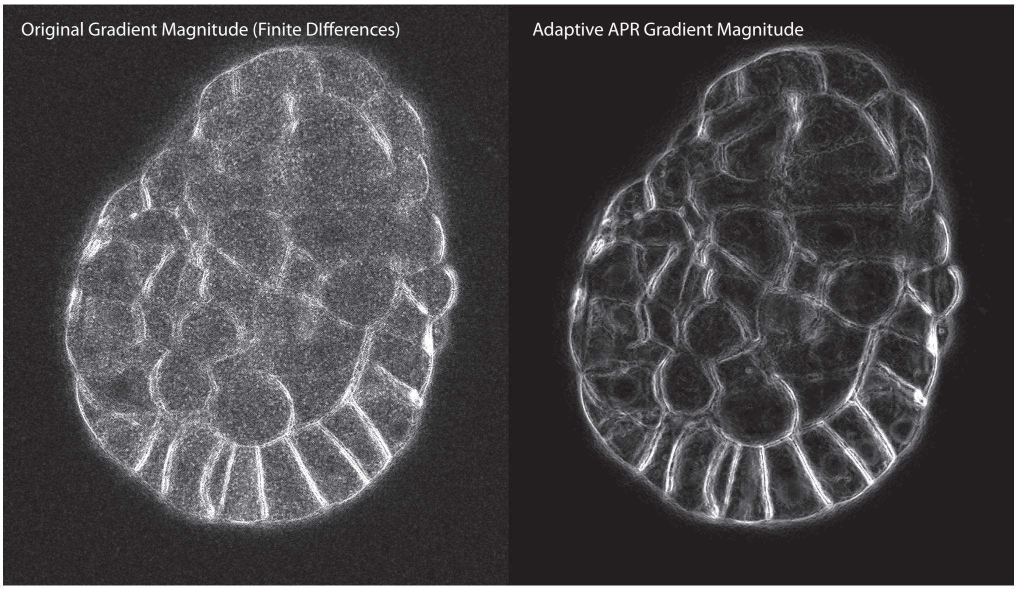

Original Gradient Magnitude (Finite DIfferences)
Adaptive APR Gradient Magnitude

Figure 7.21: Evaluation of Adaptive APR filter used to compute the gradient magnitude on an exemplar dataset 10 in Table A.1. The *right* image was generated using central finite differences to compute the gradient magnitude on the original image. The *left* image was obtained by using adaptive APR filtering in each direction with $\{-\frac{1}{2h_-}, \frac{1}{2h_-} - \frac{1}{2h_+}, \frac{1}{2h_+}\}$, where h_+ and h_- are the particle spacings in the positive and negative direction respectively, and multiple particles on one face being averaged, and then the image formed using piecewise constant reconstruction.

For the APR, implementing the adaptive APR filter involves the same process as in the linear neighbor access benchmark above. In contrast, an adaptive pixel implementation would be significantly more involved. Here we show results for a gradient magnitude calculation using adaptive APR filters. The gradient in each direction is calculated by taking the average of one-sided differences between neighboring particles in each direction (as a particle can have up to 8 neighbors in one direction, i.e. x,y, or z). Figure 7.20, shows the results for benchmark data, where we compare the result with the gradient magnitude calculated using central finite differences on the original and ground truth images for 100 CR5 images of width 250. The adaptive APR filter had an average PSNR of 34.6 and standard deviation of 1.2394 while the average PSNR for the original image approach had an average PSNR of 16.78 with a standard deviation of 0.1802. Hence, the adaptive APR filter shows significantly more robustness to noise for our benchmark data. The algorithm also provides nice denoising properties for the exemplar datasets, as shown in Figure 7.21 for one slice of an LSFM dataset. We also tested an adaptive APR smoothing filter that involves taking a weighted average over neighboring particles. Multiple passes were made for greater smoothing. Comparative results for the same data as the gradient example are shown in Figure 7.22. The adaptive APR smoothing filter showed a higher PSNR (37.8) increase than any fixed kernel Gaussian smoothing on the original image (maximum 33.42). This held for from one to seven passes. We have shown the results for four taps, as that was the best result for the adaptive APR smoothing filter. Therefore the adaptive APR smoothing filter provides an alternative for smoothing other than using a pixel filter approach discussed above.

7.5.2 Segmentation and visualization

The visualization and segmentation examples presented above can also be considered as novel algorithms, as they directly use the properties of the APR. In Figure 7.23, we show an APR ray cast volume rending of segmentation results for exemplar data set 17, using a slightly adapted energy (A.10.6) for the graph cuts approach presented above. Colour is used to indicate depth. Across the exemplar benchmark data, using this adaptive energy and segmentation approach showed promising preliminary results, with minimal change in parameters. These results were qualitatively similar in quality, and execution time, to those produced by state of the art method distributed method presented in Afshar and Sbalzarini [5] run using a high-performance cluster. Although we do not explore this further here, it provides motivation for further development of segmentation algorithms for the APR.

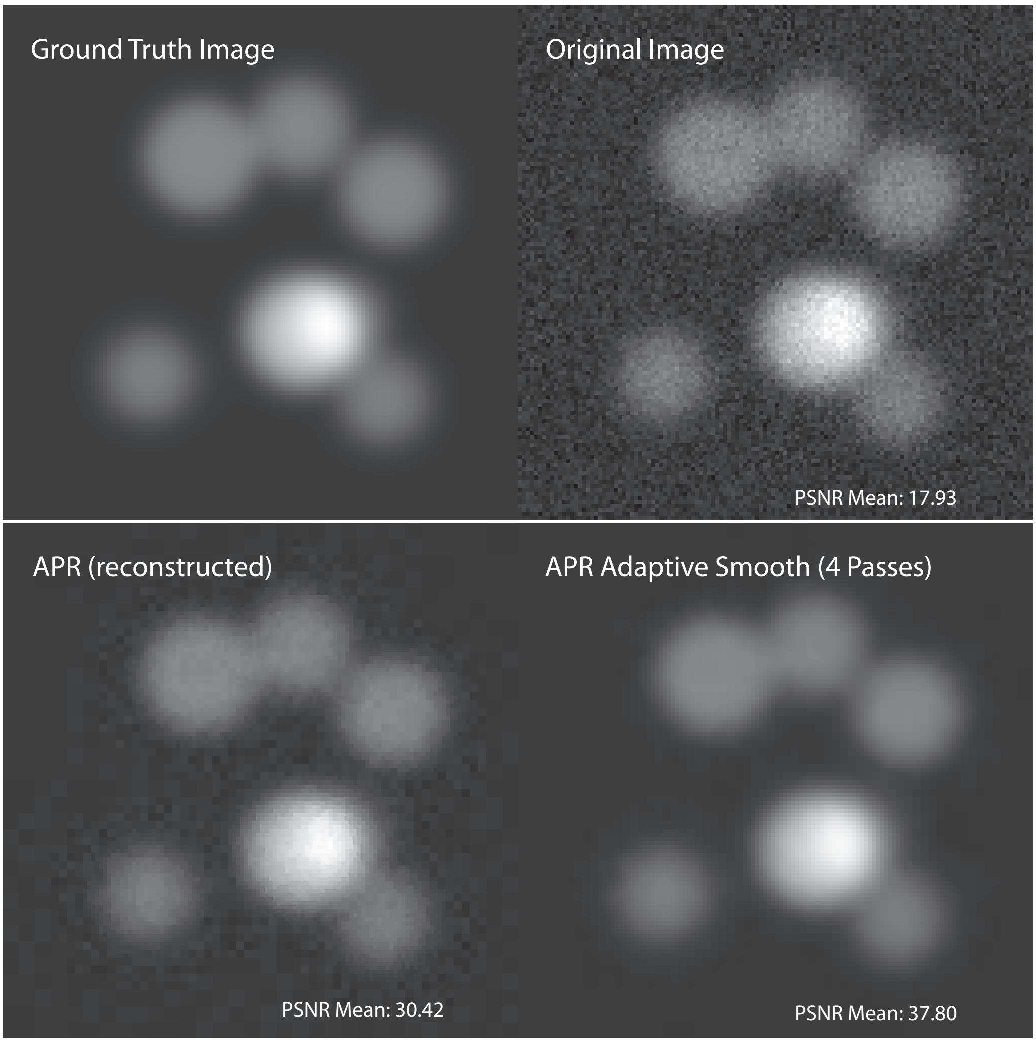


Figure 7.22: Evaluation of APR Adaptive Smooth filter with synthetic data, obtained by filtering in each direction with $\{0.1, 0.8, 0.1\}$ directly with particle neighbors generating an adaptive filter, with multiple particles on one face being averaged. *Top left*, shows the original ground truth image, *Top right* shows the original image, *bottom left* APR reconstructed image, and *bottom right* shows an example result of four passes with the filter in each direction. For each image, the mean PSNR with reference to the ground truth image is shown over 200 repetitions for CR5 images of width 250. Four passes achieved the maximum PSNR for the APR. However, any number of passes of those tested (up to 7) exceeded the PSNR of any Gaussian filter on the original or reconstructed image (maximum PSNR of 34.4).

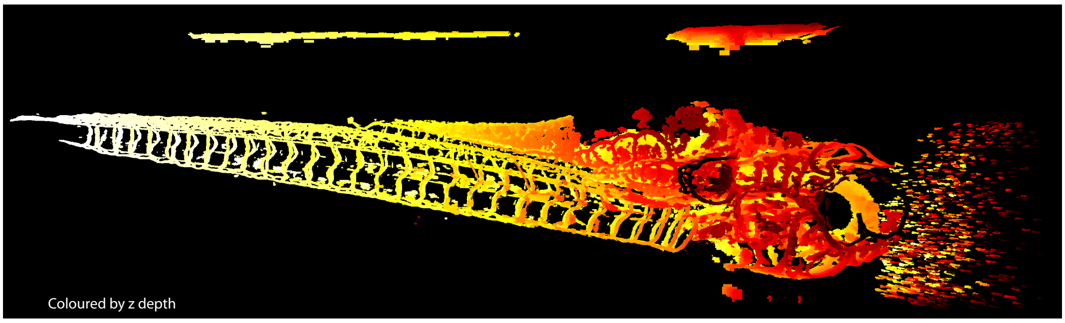


Figure 7.23: Raycast rendering of segmentation using the graph cuts segmentation approach with the energy described in A.10.6 for LSFM exemplar data set 17 in Table A.1. The image is coloured by depth, using a volume rendering approach using an extension of the ray-casting algorithm discussed in 7.4.2.

7.6 Summary and main points

In this chapter, we explored how the APR can be used for processing tasks. We did this by first discussing how the APR can be interpreted for different processing tasks, and then introduced evaluation metrics for comparison of performance with pixel algorithms. Following this, we presented a variety of processing results across three categories. These categories were performance benchmarks, visualization, and novel algorithms. The summary box below highlights the main results and findings. Lastly, we reflect on the results in light of our representation criteria.

7.6.1 Reflection on Representation Criteria

The results in this section were designed as to provide evidence regarding the APRs ability to fulfill **RC**4, which requires the adaptivity of the APR is able to be used across a wide-range of processing tasks without returning to the original image. As seen in the linear neighbor access task, there will exist tasks where the APR is less well suited than a pixel representation, resulting in lower performance. Further, we showed that in the pixel filter task, that care must be taken that the results of the adapted task still align with the original task. Further, across the benchmarks, the exact performance is highly implementation specific, with multiple alternatives for both pixels and the APR for any given task.

However, instead, let's focus on the objective of **RC**, that is the question, 'can standard processing be done effectively directly on the APR without requiring the original image?'. Further, let us consider processing tasks to be defined regarding an algorithmic objective, instead of an explicit algorithm (e.g. de-noising using blurring, as opposed to the application of a Gaussian pixel filter). Where then 'effective' encapsulates many principles and trade offs between computational and memory cost, the complexity of implementation, and its ease of adaptation to a range of existing tasks. These issues have been previously discussed in 3.4.2 and summarized in Figure 3.7. In this framework, I believe the above results for the APR are promising, for the following reasons supported by the results above:

- The simple structure of the APR allows the APR to be 'interpreted' in the same way a traditional pixel image is (See 7.1)
- The adaptation of the APR, for realistic ranges of CR, provides reductions in memory and computational cost of at least an order of magnitude across tasks (See 7.3).
- The APR provides additional information that can provide simple algorithms using the APR with high-performance.
- The APR can be directly used with some existing algorithms
- In the worst case, pseudo-pixel-APR algorithms can be developed that locally, or fully, reconstruct the pixel image (See 7.3.3).

However, there are also many limitations to processing with the APR. Firstly, there is the added complexity of implementations and data structures. This complexity is compounded by the fact that in most cases, existing algorithms will need to be redesigned and implemented, resulting in added time, and a barrier to use. Second, is that the APR is intrinsically lossy. Therefore, decisions made regarding parameters for the APR, E, and choice of Local Intensity Scale, could result in the loss of information that is vital to some processing task. In particular, certain image enhancement tasks such as deconvolution or de-noising may rely on the exact pixel distribution, therefore requiring the full original image. Although, in a particular processing pipeline, such tasks could be done before the APR formed, using the APR for later higher level tasks such as segmentation, tracking, and visualization. Alternatively, the pixel image data could be retained and used jointly with the APR.

In summary, given we cannot prove fulfillment of **RC**4, we can, however, evaluate **RC**4 in the negative. That is, is there significant evidence supporting that the APR *cannot* be used across a wide range of tasks without returning to a pixel image? If we restrict ourselves to tasks as outlined in Chapter 2, with the possible exception of image enhancement, I have yet to find such evidence. Hence, tentatively, and with further development, I conclude that the APR appears to have the correct features to satisfy **RC**4.

Summary of the chapter

- Discussed different interpretations of the APR for processing
- Introduced performance evaluation metrics for APR processing
- Evaluated performance of four processing tasks for the APR compared to pixel algorithms
- For the pixel filtering example highlighted how the APR can be limited in direct applications for certain tasks.
- Showed performance of all tasks where proportional to image content, not just original image size N
- All but the CR5 benchmark for linear access showed a speed up when compared to pixel implementations.
- All tasks showed memory reductions that scale proportional to image content.
- For the segmentation task, the memory reduction was significant, resulting in otherwise infeasible large processing tasks feasible on our benchmark machine
- Showed proof of principle of three different visualizations directly using the APR
- Showed how the APR could result in novel algorithms utilizing its spatial adaptation with the adaptive APR filters.
- Reflected on the limitations of the results and fulfillment of **RC**4

8 Comparison with previous methods and optimality

Contents

8.1 Similarity to other methods 190
8.1.1 APR form and Resolution Function 190
8.1.2 Reconstruction Condition 191
8.1.3 Local Intensity Scale 191
8.1.4 Resolution Bound 191
8.1.5 Particle Cells and Implied Resolution Function . . . 192
8.1.6 Pulling Scheme 192
8.1.7 Wavelet thresholding 193
8.1.8 Computational and Memory Cost 194
8.1.9 Summary and originality 194
8.2 Comparison to wavelet optimality 195
8.2.1 Optimal error convergence 195
8.2.2 Optimal representation of noisy signals 197
8.3 Optimal ϵ convergence of the APR 204
8.3.1 APR reconstruction 207
8.3.2 Result summary 210
8.4 Summary 211

In this chapter, we reflect on the literature introduced in Chapter 3, and discuss the similarities between the different components of the APR and existing methods in the literature. Importantly, the ideas of the APR grew out of an initialization method for the extension of Reboux et al. [93] for images, and hence many ideas are related to this approach. We structure this comparison by discussing the following elements separately: the Reconstruction Condition, the Resolution Function and APR reconstruction, the Resolution Bound, and the Pulling Scheme. We then focus on a comparison between the APR and wavelet thresholding, focusing on two optimality results regarding efficient representation and noisy reconstruction. For both, we provide an empirical comparison between the APR and wavelets reproducing the result

(Donoho and Johnstone [47] and DeVore et al. [42]). Lastly, we present some preliminary theoretical results on the optimality of APR noisy sampling.

8.1 Similarity to other methods

The Adaptive Particle Representation (APR) shares ideas and concepts across almost all of the adaptive representations discussion in Chapter 3. The APR is a member of the last group we presented, of adaptive sparse collocation methods (3.3.3). It shares the principle of adapting sampling to image content that is at the core of he mesh-based adaptive image representations, however, the APR does not rely on a mesh triangulation. Conceptually, it combines error estimates at the core of error equidistribution (3.3.3) methods with a reconstruction form similar to the 'oracle' methods reviewed by Donoho and Johnstone [47].The solution is computed using common multi-resolution structures (3.3.1) and results in a solution that is similar to those produced by sparse collocation wavelet methods (3.3.3). For brevity, for the methods we discuss here we will often refer back to the appropriate section in Chapter 3 where references and descriptions can be found.

8.1.1 APR form and Resolution Function

The APR represents a function using a Resolution Function $R(\mathbf{y})$ and set of collocation points $\mathcal{P}^*$, see 5.1. This form, is most similar to the adaptive oracle models such as Breiman et al. [22], Friedman and Silverman [52], Brockmann et al. [24] discussed in Donoho and Johnstone [47] and used for adaptive bandwidth regression tasks for unknown functions. The oracle sets the bandwidth of the regression kernel in some way across the domain to improve the predictive power of the model with the presence of noise. Another related approach is that of adaptive Lagrangian particles [93] where a local resolution is carried as a property of the particles. However, the APR differs in that the Resolution Function is defined everywhere.

The reconstruction allows the use of any weighted averaged of particles within the Resolution Function. Such a form is similar to the local reconstruction used in Partition of Unity methods [16]. This more general form differs from methods such as wavelets (3.3.1), dictionary methods (3.3.2), and adaptive mesh methods (3.3.3) , where reconstruction is restricted to a single reconstruction method often defined by a specific basis. Where as in the APR reconstruction can be achieved by a range of basis functions satisfying criteria on their particle weights and support. Further, in fixed basis methods, the local adaptation scale is implicit in the size of the local basis function, rather than explicit as for the Resolution Function in the APR.

8.1.2 Reconstruction Condition

The Reconstruction Condition 5.2, requires that the infinity norm of the reconstruction from the APR point-wise weighted by some local intensity scale is bounded a user specified constant across the domain. For a constant local intensity scale, a similar bound is required the adaptive methods described in DeVore et al. [43] and Agarwal and Suri [6]. DeVore et al. [43] provides a scheme for constructing sparse wavelet transforms that guarantee point-wise reconstruction error. Agarwal and Suri [6] discusses optimal mesh triangulations with a point-wise Reconstruction Condition and proves that finding the general optimal solution belongs to a class of NP-hard problems.

8.1.3 Local Intensity Scale

Across the review, I have not found methods that attempted to include a weighting function to control the point-wise error. This likely reflects a lack of motivation in the fields of application intended for the representations. However, existing methods could be extended to include such concepts. Further, given the lack of guarantee, or practical constraints on the Local Intensity Scale, it seems further algorithmic and theoretical work is still needed for a more flexible integration of such scales.

8.1.4 Resolution Bound

The Resolution Bound, Eq 5.3, provides a direct condition on the Resolution Function for the reconstruction that given restrictions can guarantee the Reconstruction Condition. The Resolution Bound is derived from analysis of the reconstruction error using the integral form of the remainder of a Taylor series approximation. Using such a technique has a long and rich history starting with its use for splines representations as originally presented in de Boor [37], De Boor [38], Burchard [25] in combination with the idea of error equidistribution 3.3.3. Further, the integral remainder is at the heart of a range of error equidistribution based methods.

However, the exact form of the Resolution Bound placing an inequality on the Resolution Function everywhere in the domain does not appear to be central to any of these methods. The closest appears in Reboux et al. [93] which includes a minimum step over a Local Resolution Estimate like quantity defined on particles.

Another point of difference from these methods is the formulation as a constrained maximization problem of $R(\mathbf{y})$ subject to the Resolution Bound. Further, this then allows the APR to utilize the geometry that is inherent in the Resolution Bound relating to $L(\mathbf{y})$ and $R(\mathbf{y})$, that is not a feature of either the Reconstruction Condition or the direct error analysis. This geometry arises from the width of interaction, and function value, being linked.

8.1.5 Particle Cells and Implied Resolution Function

Particle Cells are used by the APR both for adaptive sampling, and for the construction of an Implied Resolution Function that satisfies the Resolution Bound. Particle Cells partition both the spatial domain and 'resolution' domain into regular regions with sizes that are negative powers of two segments of the domain. The decomposition in the spatial domain corresponds to the popular binary, quad, and octree decompositions in 1D, 2D, and 3D. Further, such decompositions are at the heart of the Haar wavelet transform and multiresolution and pyramid based methods 3.3.1. However, the particle cells differ in that also partition the resolution domain, in a way that is similar to the scale space in scale-space decomposition [139] and adaptive particle cell-lists [13].

The Implied Resolution Function is constructed by a collection of Particle Cells which cover the domain like hypercube blocks. This special Particle Cell set $\mathcal{V}$ partitions the spatial domain without overlap. This is similar in spirit to the piecewise constant decomposition over similar elements in the Haar wavelet transform. However, in practice, it is quite different, as only one block is used for each point in space in the APR, and this is of a fixed value.

I have not come across similar work regarding the explicit formulation of the Particle Cells, and the results proved in 5.3 regarding the Implied Resolution Functions and the Resolution Bound. It is likely; they are simply restatements of existing results presented in a new formulation. I speculate that if the problem correctly formulated as a constrained maximization problem under a set of pixel constraints, this link would be made more clear. For example, the optimal solution is created by the effective polytope constructed by all the of the individual constraints of the particle cells in the Local Particle Cell set $\mathcal{L}$. Such links are left to future work.

8.1.6 Pulling Scheme

The Pulling Scheme allows for the direct construction of the OVPC $\mathcal{V}$, representing the optimal Implied Resolution Function in a worst case linear algorithm. The Pulling Scheme utilizes the range of results for Particle Cells, constructing a solution by propagation of individual solutions from highest to lowest resolution. Conceptually, this could be thought of as analogous to a Green's Function approach to solving differential equations. However, the integral is replaced by the 'minhull' operation across the sets. I note, that this minhull operation likely has a direct correspondence with an existing concept in the literature of which I am not aware.

When constructing the set, the algorithm involves steps of adding particle cells from the Local Particle Cell set $\mathcal{L}$ and layers of neighbors that increase in resolution away from local maxima in resolution. The algorithm appears to have greatest similarities with adaptive thresholding wavelet schemes presented in DeVore et al. [43], Vasilyev and Bowman [131], Rossinelli et al. [97]. All of these methods include insertion of high-resolution areas, with padding, analogous to ghost layers, added to provide support for the high-resolution particles. In DeVore et al. [43], these operations are directly motivated by the Reconstruction Condition of the methods, however in Vasilyev and Bowman [131] and Rossinelli et al. [97], they seem to be justified in a more practical heuristic manner.

8.1.7 Wavelet thresholding

Although not precise, there seems to be a conceptual link between the way wavelet thresholding 3.3.2 selects coefficients and the adaptivity of the APR. Also, the APR can be used with a wavelet based reconstruction method, using the scaling functions of the wavelet transform, selecting the largest scaling function at each point that satisfies the Resolution Bound.

We briefly describe this link for the 1D Haar wavelet transform. Let's consider the Haar wavelet transform in 1D and its detail coefficients $d_{i,j}$, where i is the spatial coordinate and j the level. These detail coefficients at any level are the difference between neighboring pixels in a mean down-sampled version of the signal at the previous level. Therefore, they represent a down sampled approximation of the local gradient. Therefore, the thresholding

$$|f(x_{j,1}) - f(x_{j,2})| \leq T \tag{8.1}$$

is related to through a one-sided finite difference operator as

$$|\frac{\partial f}{\partial x}(x_1)| \leq 2^{j-j_{max}}T \tag{8.2}$$

where it is assumed the highest resolution sampling j_{max} has spacing $h = 1$. This is similar to the assignment of level l during construction of the Local Particle Cell Set $\mathcal{L}$. If we consider a Particle Cell at resolution l^* in the APR, and a wavelet transform thresholded with some T that has a similar highest resolution element for some point x. For that to occur in the APR, would require there to be no smaller Particle Cells within the spatial domain of the Particle Cell in $\mathcal{L}$. Similarly, for the wavelet, being the highest resolution at that point would require that all detail coefficients within the support of the wavelet to be zero. Hence, both representations reflect that the gradient in that region is everywhere bounded by some $2^{j-j_{max}}T$. However, the APR having resolution l^* provides a stronger constraint on the underlying f, then the Haar wavelet, as the APR allows for more general reconstructions with a support that is not constrained to an individual particle cell element.

This concept also seems to extend to the higher order reconstruction APR, when compared to higher order wavelet reconstructions. As the details coefficients then become bounded by local integrals of the higher order derivatives and comparison to the APR can again be made this way. We leave further exploration of these ideas to future work.

8.1.8 Computational and Memory Cost

Given the pipeline for forming the APR has a worst-case linear computational and memory cost for input size N, it is, therefore, in an asymptotic sense as efficient any of the adaptive approaches above. Beyond this, providing exact comparisons requires both a reference problem and implementations. Further, based on reported execution times for natural images, it would appear that the formation of the APR can scale and be more computed efficiently when compared to results given for mesh-based image representations [104].

However, regarding memory cost, the in-place DWT approach for calculation of wavelet coefficients is superior to the full APR pipeline. As the lifting scheme can be done in-place, requiring only the original image in memory, however, the APR pipeline we have presented requires approximately 3.25 times more memory. However, if we only consider the pulling scheme, and assume that the local particle cell set is given, then the memory cost is smaller than the original image. Also, as shown above, in practice the computational performance of the pulling scheme is dependent on the image content showing sub-linear scaling in N for a fixed total level of content. Further, a wavelet equivalent of the APR would likely require additional algorithmic steps and additional storage of variables than simply the DWT.

8.1.9 Summary and originality

In summary, the APR unites many concepts and ideas across a large range of existing adaptive and multi-resolution representations. However, the focus on the Resolution Function using the Resolution Bound, and solving this using an optimal restricted solution using Particle Cells, appears to be an original approach to adaptive representations. Further, the inclusion of the Local Intensity Scale and the space-time adaptation extension (discussed in Chapter 10) appear to be novel.

8.2 Comparison to wavelet optimality

Given the wide range of algorithms and methods for adaptive representations, optimality results can be of particular use. Optimality results, usually, provide proof that a property of a given representation is 'the best' in some sense, or belongs to the class of 'optimal' results. Being able to prove a representation is optimal is useful as they provide theoretical guidance on whether alternative methods or additional development, would provide increased performance.

Wavelet thresholding (3.3.2) techniques possess many optimal properties that have contributed to their wide adoption and use. These have also inspired similar optimality results such as those proven for the greedy point removal method in Iske and Demaret [61]. We will focus on two of these results here, and compare the results of the APR using benchmark results from the original papers. First, we address the optimal scaling of reconstruction error, as introduced by DeVore et al. [42]. Second, we address the optimal representation of noisy functions as described in Donoho and Johnstone [47]. We already briefly introduced these results in 3.3.2 above. Both of these results appear to be highly relevant in assessing the APR. Inspired by these results in the following section we prove a related result for noisy sampling and the APR.

8.2.1 Optimal error convergence

As discussed previously in 3.3.2, DeVore et al. [42] proved that wavelet thresholding has an optimal rate of convergence in terms of some approximation norm as the number of coefficients in the representation $\#R$ is decreased. They do this by proving a theoretical optimal rate and showing that wavelets, given certain restrictions, satisfy this.

Explicitly they show that given a function in α Besov space (Appendix A.2) the optimal rate of convergence of the error compared to the number of coefficients $\#R \to 0$ for an adaptive representation asymptotically follows

$$||f - \hat{f}||_{L_p} \sim \mathcal{O}(\#R^{\frac{\alpha}{d}}) \tag{8.3}$$

where $||.||_{L_p}$ represents a chosen p norm with $1 \leq p \leq \infty$ and $\frac{1}{q} = \frac{1}{p} + \frac{\alpha}{2}$. They show that given the chosen wavelet can reproduce polynomials of up to degree

α, the number of non-zero co-efficients in a thresholded wavelet scheme will achieve this optimal rate.

a-priori it is not known in what α Besov class any given image or function belongs to. However, given wavelet thresholding is optimal, for any given image the optimal rate can be estimated by empirical observation of α as $\#R \to 0$ using an optimal representation.

Ideally, we would also prove whether or not the APR, or its higher order reconstructions, belong to this optimal class of methods. On first appearances using the tools of Besov spaces, this would appear to be a tractable problem, but we leave it to future research. Instead, we take the easier approach of testing this empirically. We can form the APR, and scale $E \to \infty$ such that $N_p \to 0$ and compare the scaling behavior with wavelet thresholding. If the same rate is achieved, then we have empirical evidence that they belong to the same optimal class. Practically, it is not clear to me that such a result is of immediate use, but it is an effective tool when comparing between underlying properties of methods.

Comparison with APR

We can deduce from the properties of the APR, that a comparable thresholded wavelet transform should be able to produce the same norm error using less non-zero coefficients than a given APR. This directly follows from the fact that the APR does not rely on a given basis, and allows for isotropic reconstruction at any point. These additional restrictions come at the cost of additional particles for the APR. Further, the APR representation can be further transformed using a non-thresholded wavelet representation that would have $\#R \leq N_p$ coefficients with identical reconstruction error. Hence, wavelet transforms of the APR could be a useful tool for APR compression. However, it is still of interest whether the APR adaptation of the APR belongs to the optimal class for any reconstruction norms, and if so, which. To assess this, we compare L_1 and L_2 error scaling results Haar wavelet thresholding, as was done in the original paper [42]. Using wavelet thresholding implemented in Matlab's wavelet toolbox for two test images shown in Figure 8.1. These images are the classic Lena, and MIT Cameraman images. The Lena image was used in the original paper. However, the Cameraman image was not. The results shown for these two test images were consistent with those for other images tested.

Figure 8.2 shows the L_1 and L_2 scaling results on a log-log plot against the number of non-zero coefficients in the Haar wavelet representation and the number of particles in the APR. The piecewise constant reconstruction method was used for the APR. The L_1 error is computed as the mean of the absolute value of the reconstruction error, and the L_2 the mean squared reconstruction

Figure 8.1: Classic 512×512 benchmark images, Cameraman (512×512 *left*), and Lena (*right*), used for the optimal error scaling results in Figure 8.2

error. In the *left* plots we can see that as the $\#R$ gets small the error of both the APR and wavelet asymptote to a curve of a similar slope. However, in the case of the L_2 error, the Haar wavelet result asymptotes to a steeper curve. This behavior was consistent across other test images. Hence, it seems that in terms of L_1 error the APR has similar convergence behavior, and hence the same optimal class. However, this is not the case for the L_2 error. Also, despite the same optimality property in L_1, we can see that consistent to our arguments above the Haar wavelet transform produces a lower error for the same number of coefficients (particles).

Interestingly, here, as in the original results, the linear scaling behavior only arises for small $\#R$. At such levels, the image quality is low (not shown), and hence it is unclear how useful such scaling behavior is in practice. We leave further investigations to future research.

8.2.2 Optimal representation of noisy signals

The second, maybe more important results for comparison with wavelets are those for the optimal representation of noisy signals. From our analysis above, we show that the APR seems robust and effective for noisy images, improving the PSNR from the noisy original while adaptive sampling. However, the question arises, is this being done in an optimal way? For wavelet thresholding, as already discussed above in 3.3.2, it was proven in Donoho and Johnstone

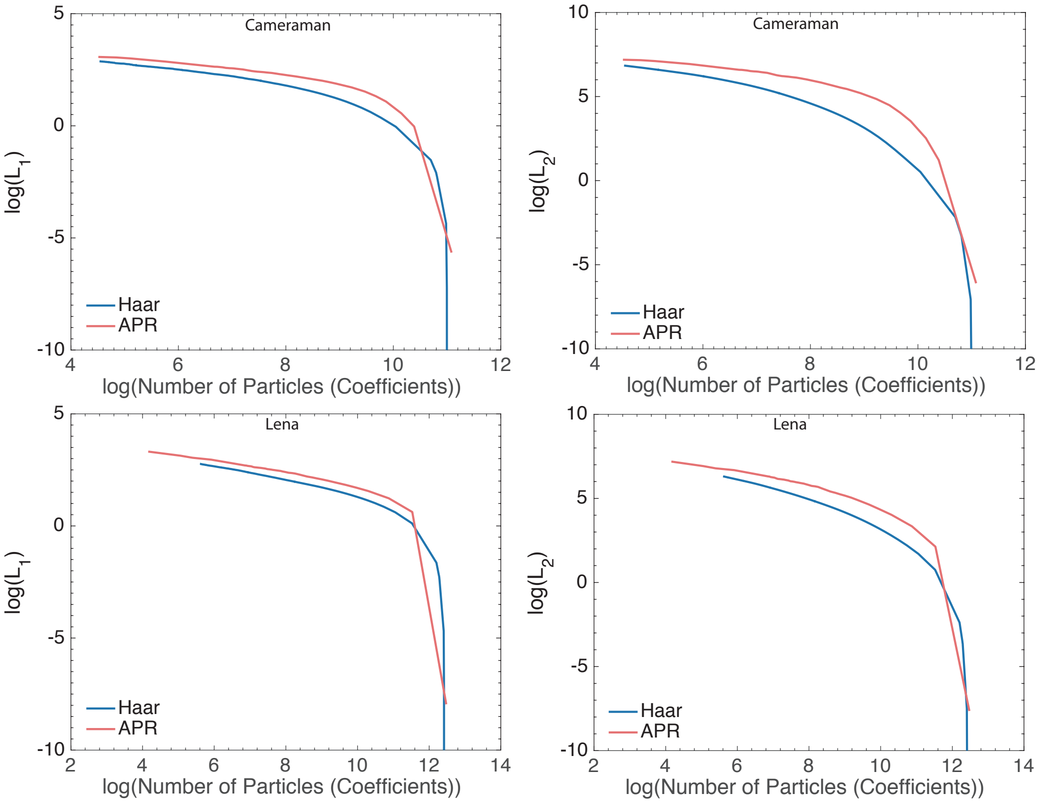


Figure 8.2: The scaling results for the L_1 (*left*) and L_2 (*right*) errors plotted as a log-log plot against the number of non-zero coefficients in the thresholded Haar wavelet transform and the number of particles in the APR. The results are shown for the Cameraman (*top*) and the Lena (*bottom*) benchmark images.

[47], that wavelet thresholding can achieve 'near' optimal convergence for the approximation of a noisy function. We shall give the exact result here, and then reproduce the results for the test functions provided in Donoho and Johnstone [47] and compare them to the equivalent results for the APR. Following this, we prove some related results for the APR.

We begin by considering a function sampled at N equally distributed points $\bar{x}$ as

$$g\{x\} = f(x) + \eta(x) \tag{8.4}$$

where $\eta(x)$ is a zero mean Gaussian noise process with standard deviation σ. Then if we consider some de-noised reconstruction of the function f, which we call $\hat{f}$, then we define the risk as

$$\mathcal{R}(\hat{f}, f, N) = \mathrm{E}[\frac{1}{N}\sum_{i=1}^{N}(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))^2] \tag{8.5}$$

that is the expected value of the MSE of the reconstruction based on N original samples. We can then consider the performance of de-noising methods by observing the convergence behavior of $\mathcal{R}(\hat{f}, f, N)$ as $N \to \infty$. The optimal rate, which Donoho and Johnstone [47] call the 'parametric rate' is $\mathcal{O}(\frac{1}{N})$. Compared to any fixed band-width (width) method that has a best case approximation rate of $\mathcal{O}(\frac{1}{N^{1/2}})$.

The result of Donoho and Johnstone [47] is that for an ideal wavelet adaptation $\hat{f}^*$ with knowledge of the true function, the risk is bounded in the following way,

$$\mathcal{R}(\hat{f}^*, f, N) \leq \frac{(C_1 + C_2 \log_2(N))\sigma^2}{N} \tag{8.6}$$

and that they provide methods that can achieve estimators $\hat{f}$ within $2\log(N)$ factor of this as

$$\mathcal{R}(\hat{f}, f, N) \leq (2\log(N) + 1)\left(\frac{(C_1 + C_2 \log_2(N))\sigma^2}{N} + \frac{\sigma^2}{N}\right) \tag{8.7}$$

where only the noisy input is used.

Comparison with APR

In Donoho and Johnstone [47] they give results for four test noise corrupted test functions called HeaviSine (Figure 8.3), Doppler (Figure 8.4) ,Bumbs (Figure 8.5) and Blocks (Figure 8.6). The function definitions are given in the

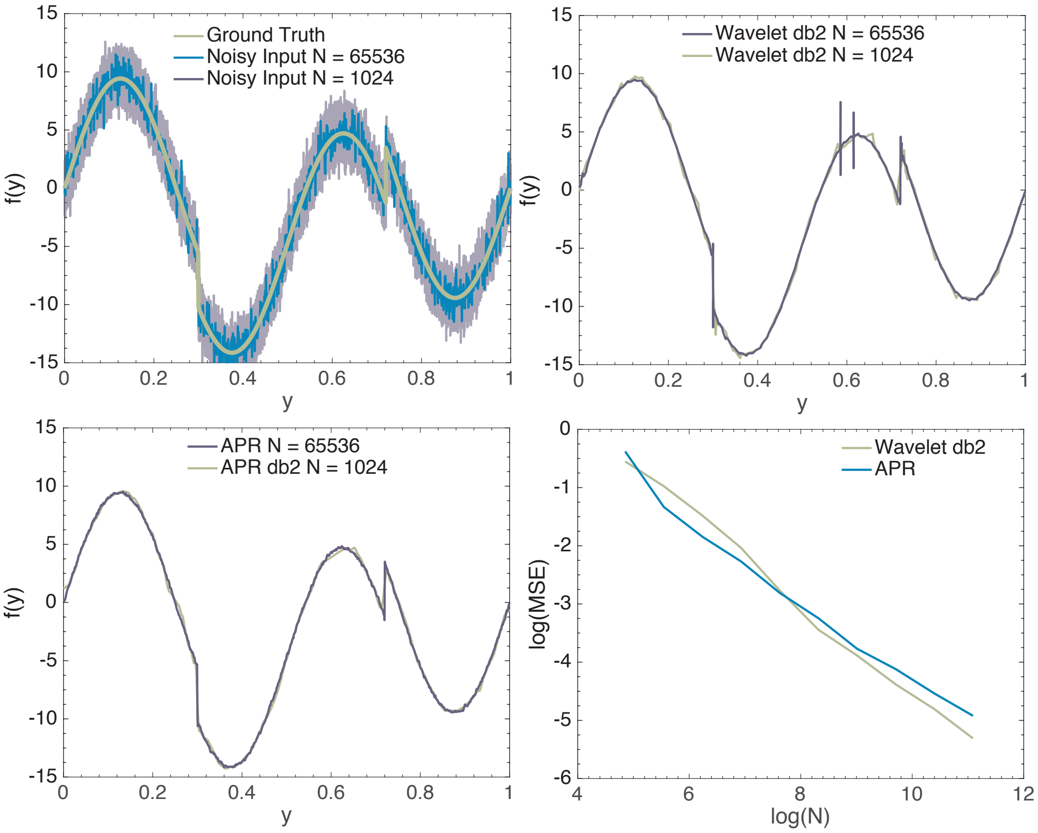


Figure 8.3: Benchmark results for the **HeaviSine** test function from Donoho and Johnstone [47]. The HeaviSine test function is a scaled version of $f(y) = 4\sin(4\pi y) - \text{sgn}(t - 0.3) - \text{sgn}(0.72 - t)$. The *top left* plot shows the noise free function (Ground Truth), and two examples of noisy input sequences. The top right plot then shows the wavelet denoising result for the two noisy input sequences, and bototm left shows the same for the APR. The bottom right shows a log-log plot of the mean MSE over 50 realizations for the APR and wavelet reconstructions for $N = 2^6, .., 2^{16}$

caption of the figures. The functions are all scaled as described in Donoho and Johnstone [47] and corrupted by Gaussian noise with standard deviation $\sigma = 1$. We reproduce the results of the paper for a thresholded wavelets and the APR.

For the wavelet results we again used Matlab wavelet functions, using the *wden* function with hard wavelet, multi-level thresholding, using the 'sqrt-log' shrinkage method allowing six levels of adaptation and using the second Daubechies wavelet [34]. These settings provided the best performance across all benchmarks of tested wavelets and settings combinations, and where 'representative', (if not the best) of the best settings for each example. This allowed for a 'fairer' and less complicated result then customizing the wavelet param-

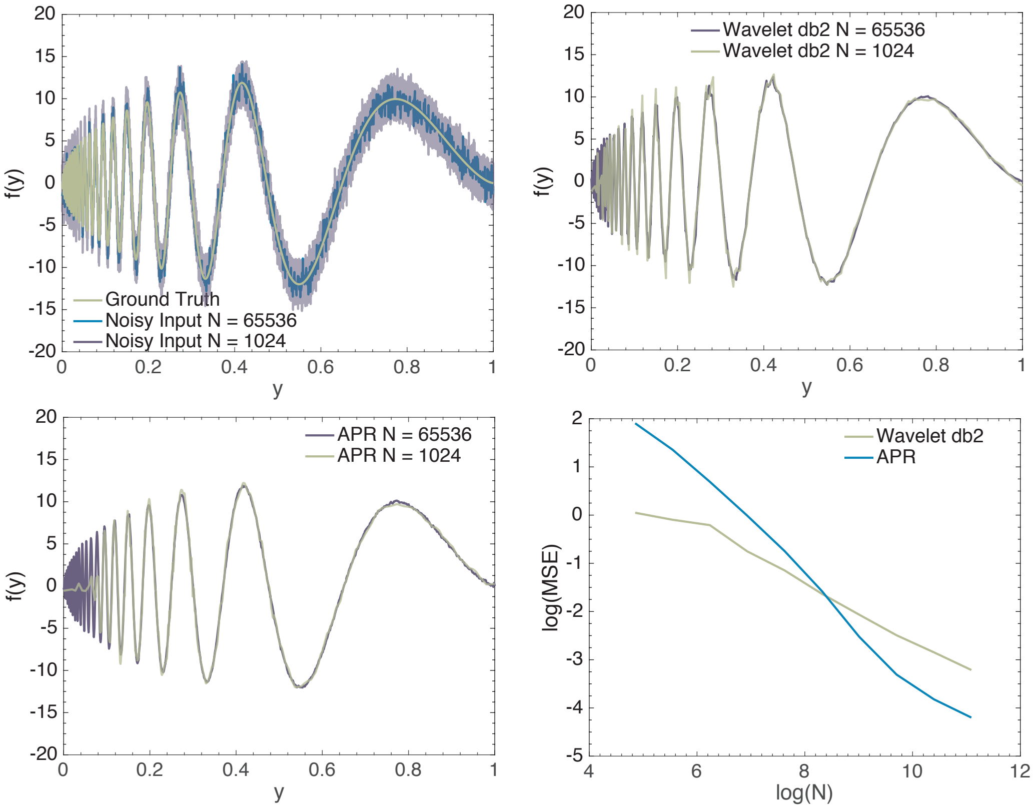


Figure 8.4: Benchmark results for the **Doppler** test function from Donoho and Johnstone [47]. The Doppler test function is a scaled version of $f(y) = (y(1-y))^{1/2}\sin\left(\frac{2\pi(1+a)}{t+a}\right)$ for $a = 0.05$. The *top left* plot shows the noise free function (Ground Truth), and two examples of noisy input sequences. The top right plot then shows the wavelet denoising result for the two noisy input sequences, and bototm left shows the same for the APR. The bottom right shows a log-log plot of the mean MSE over 50 realizations for the APR and wavelet reconstructions for $N = 2^6, .., 2^{16}$

eters for each function.

For the APR, we follow a similar scheme as that for the 3D LSFM data. We use cubic B-splines to estimate the gradient with the smoothing parameter set as $\lambda = .5N$ across the samples. The Local Intensity Scale was set to be a constant, set to the maximum of the function across the interval, and the relative error estimate set to $E = 0.08$. The particle intensities were estimated from the original image by taking the average over all points within the implied resolution function of the particle. The function was then reconstructed using piecewise linear reconstruction between the particles. Note since we are interested in risk convergence, we do not consider the number of particles used. However, consistently, the number of wavelet coefficients used was less than the number of particles used by the APR.

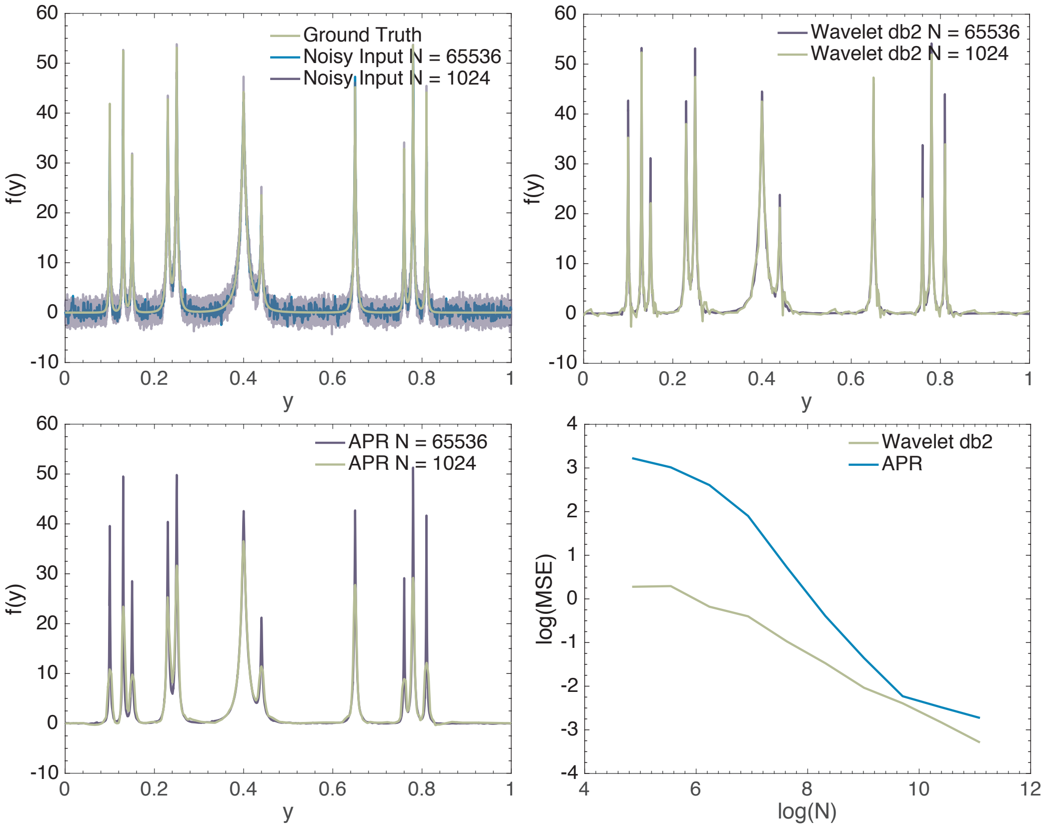


Figure 8.5: Benchmark results for the **Bumps** test function from Donoho and Johnstone [47]. The Bumps test function is a scaled version of $f(y) = \sum_j h_j K((y - y_j)/w_j)$ where $K(y) = (1 + |y|)^-4$ and $h_j = \{4, 5, 3, 4, 5, 4.2, 2.1, 4.3, 3.1, 5.1, 4.2\}$, $w_j = \{0.005, .005, .006, .01, .01, .03, .01, .01, .005, .008, .005\}$ and $y_j = \{0.1, 0.13, 0.15, 0.23, .025, 0.4, 0.44, 0.65, 0.76, 0.78, 0.81\}$. The *top left* plot shows the noise free function (Ground Truth), and two examples of noisy input sequences. The top right plot then shows the wavelet denoising result for the two noisy input sequences, and bototm left shows the same for the APR. The bottom right shows a log-log plot of the mean MSE over 50 realizations for the APR and wavelet reconstructions for $N = 2^6, .., 2^{16}$

A note, when comparing the MSE of the APR and Wavelet representations it must be remembered that the APR has an ideal non-zero MSE through the Reconstruction Condition. That is if the samples were noise-free the MSE would be a non-zero value that is less than E^2. However, the ideal wavelet representation given noise-free would converge to a zero MSE.

For each benchmark function, both the wavelet and APR method were run on the noisy function with $N = 2^6, .., 2^{16}$ samples with 50 repetitions each. For each test function we show an example of the ground truth, and two noisy function inputs for $N = 1024$ and $N = 65536$ and then in separate plots we then show the APR and wavelet solutions. Lastly, we show plot the average MSE against N on a log-log plot.

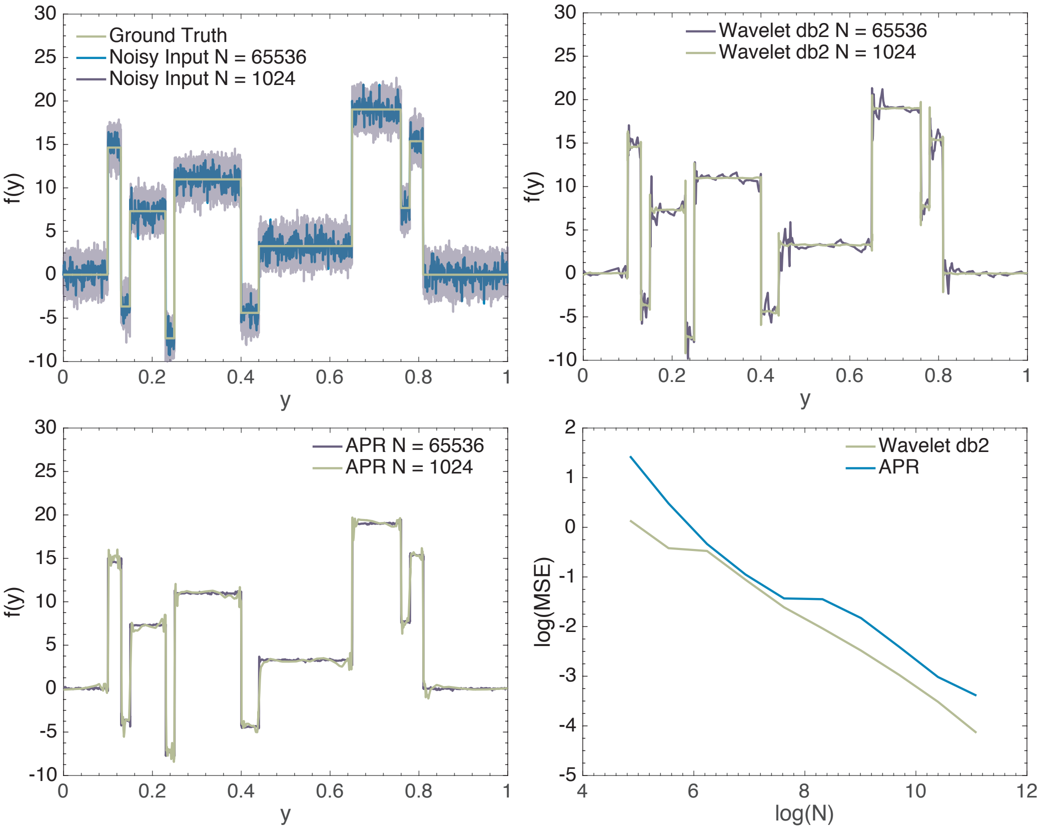


Figure 8.6: Benchmark results for the **Blocks** test function from Donoho and Johnstone [47]. The Blocks test function is a scaled version of $f(y) = \sum_j h_j K((y - y_j))$ where $K(y) = (1 + \text{sign}(y))/2$ and $h_j = \{4, -5, 3, -4, 5, -4.2, 2.1, 4.3, -3.1, 2.1, -4.2\}$ and $y_j = \{0.1, 0.13, 0.15, 0.23, .025, 0.4, 0.44, 0.65, 0.76, 0.78, 0.81\}$. The *top left* plot shows the noise free function (Ground Truth), and two examples of noisy input sequences. The top right plot then shows the wavelet denoising result for the two noisy input sequences, and bototm left shows the same for the APR. The bottom right shows a log-log plot of the mean MSE over 50 realizations for the APR and wavelet reconstructions for $N = 2^6, .., 2^{16}$

Across the benchmarks, we see that both the APR and wavelet reconstruction provide high-quality estimations of the noisy function. Further, across the benchmarks, 'by eye' the APR solutions appear to be of subjectively of equivalent or in some cases higher quality. However, we note that the behavior of the MSE is varied across the benchmarks. For the HeaviSine test function in Figure 8.3, we see very similar convergence behavior. However, the APR appears to better capture the function discontinuities. In contrast in the Bumps test function the APR provides a smoother all round reconstruction, but consistently underestimates the height of the peaks. Further, the scaling behavior of Dopper, Bumps, and Blocks test functions appears to show multiple regimes. In comparison, the wavelet results show more consistent scaling.

From the above results, it appears that the APR provides high-quality reconstructions that are comparable with those of the near optimal reconstructions with wavelets. Further, the APR appears to show multiple scaling regions, with higher and lower convergence regions. Subjectively, some of these reconstructions appear to be of better quality.

8.3 Optimal ϵ convergence of the APR

The above results provide motivation for further analysis of the approximation behavior of the APR for noisy signal inputs. However, as mentioned, if the APR allowed perfect estimation of the particle intensities, the reconstruction would still result in a non-zero MSE, reflecting the Reconstruction Condition and relative error E and Local Intensity Scale $\sigma(y)$. Hence instead it appears to be more appropriate to evaluate the noise performance regarding how, and if, the APR converges in MSE to a value within E^2, and what our expected value of the APR is given noisy input. We call this ϵ convergence and explore this below.

Lets consider a reconstruction of the APR, where we assume that $R(y)$ is noise free, either a continuous solution satisfying the Resolution Bound, or the implied Resolution Function $R^*(y)$ for an APR with $\sigma(y) = 1$ and relative error $E = \epsilon$ (to avoid confusion with the expectations below), where we have estimated the particle intensity values $\hat{f}(\mathbf{x}_p)$ from some noisy sampling $g\{\bar{\mathbf{x}}\}$, with N points, estimated as

$$\hat{f}(\mathbf{x}_p) = \frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{x}_p,R(\mathbf{x}_p))} g\{\mathbf{x}\} \tag{8.8}$$

that is simply the weighted sum of all points in $g\{\bar{\mathbf{x}}\}$ within $R(y)$ of $\mathbf{x}_p$. As we have done in the benchmark results above. We assume, as above, that each value of g can be decomposed as

$$g\{x\} = f(x) + \eta(x) \tag{8.9}$$

where $\eta(x) \sim \mathcal{N}(0, \sigma)$. That is, each value is normally distributed with zero mean and standard deviation σ, and the process is independent of each x. For comparison with the wavelet results in Donoho and Johnstone [47] we are interested in the statistical properties of the estimate $\hat{f}(\mathbf{x}_p)$ as $N \to \infty$.

First, let's consider the expected value of the error in the estimated particle intensity,

$$\mathrm{E}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)] = \mathrm{E}[\frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} g\{\mathbf{x}\} - f(\mathbf{x}_p)] \tag{8.10}$$

now let us consider a further decomposition of this sum in the following way

$$\mathrm{E}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)] = \mathrm{E}[\frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} (f(\mathbf{x}_p) + h(\mathbf{x}) + \eta(\mathbf{x}))] - f(\mathbf{x}_p) \tag{8.11}$$

where we use 8.9, and we define $h(\mathbf{x})$ by decomposing the noisy function component in terms of $g\{\mathbf{x}\} = f(\mathbf{x}_p) + h(\mathbf{x}) + \eta(\mathbf{x})$. Now given that the expectation is linear, we can isolate the random variable, giving us

$$\begin{aligned}\mathrm{E}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)] = {} & \frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} h(\mathbf{x}) \\ & + \frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} \mathrm{E}[\eta(\mathbf{x})]\end{aligned} \tag{8.12}$$

now given that each $\eta(x)$ is independent and identically distributed (i.i.d) with mean zero, then,

$$\mathrm{E}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)] = \frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} h(\mathbf{x}). \tag{8.13}$$

where we relate the two as $g(\mathbf{x}) = h(\mathbf{x}) + f(\mathbf{x}_p)$. Now, given that $R(y)$ satisfies the Reconstruction Condition for ϵ, and let $M = \frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}}$ (the inverse of the number of sample points used in the neighborhood), then we can then bound this as,

$$\begin{aligned}|\mathrm{E}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)]| &= |\frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} h(\mathbf{x})| \\ &\leq \epsilon \left(\frac{M-1}{M}\right)\end{aligned} \tag{8.14}$$

where the factor comes from the assumption that $h(\mathbf{x}_p) = 0$. However, given we have also that $R(y)$ satisfies the Resolution Bound in addition to the Reconstruction Condition then the maximum gradient is bounded across the interval. This allows us to get an upper bound on the growth of $h(\mathbf{x})$ by assuming the it is at the worst case the local minimum or maximum of a piece-wise linear (in 1D) sections. Using this upper bound we get the tighter bound that,

$$\begin{aligned} |\mathrm{E}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)]| &= |\frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} h(\mathbf{x})| \\ &\leq \frac{\epsilon}{2^{1/d}} \end{aligned} \tag{8.15}$$

where d is the dimension. Hence, our estimate will converge to a value within $\frac{\epsilon}{2^{1/d}}$ of the true value as $N \to \infty$. Now lets consider the variance of this estimate, that is what is the asymptotic behavior of the MSE of our estimate as again $N \to \infty$. So we have

$$\mathrm{Var}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)] = \mathrm{Var}[\frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} g\{\mathbf{x}\} - f(\mathbf{x}_p)] \tag{8.16}$$

which following the same steps as above we get,

$$\mathrm{Var}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)] = \mathrm{Var}[\frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} \eta(\mathbf{x})] \tag{8.17}$$

which is the variance of the uniformly minimum variance unbiased estimator of the normally random variable η and is therefore

$$\mathrm{Var}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)] = \frac{\sigma^2}{M}. \tag{8.18}$$

Now assuming that f has a bounded first derivative, then for sufficiently large M that $M > 0$ such that the above makes sense then

$$\mathrm{Var}[\hat{f}(\mathbf{x}_p) - f(\mathbf{x}_p)] = \frac{\sigma^2}{C_0 N}. \tag{8.19}$$

since $R(y)$ defines an isotropic region representing a constant (hyper) volume fraction of the domain, and C_0 is some point dependent constant. Therefore, each estimate $\hat{f}(\mathbf{x}_p)$, converges to within ϵ of $f(\mathbf{x}_p)$, with assymptotic rate of $\frac{1}{N}$.

Pointwise approximation using $R(y)$

Next, we consider what we asymptotically get for the expected MSE that we align with Donoho and Johnstone [47] and call the risk,

$$\mathcal{R}(\hat{f}, f, N) = \mathrm{E}[\frac{1}{N}\sum_{i=1}^{N}(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))^2] \tag{8.20}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}[(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))^2] \tag{8.21}$$

now using the fact that $\mathrm{E}[X^2] = \mathrm{Var}[X] + (\mathrm{E}[X])^2$, we get,

$$\mathcal{R}(\hat{f}, f, N) = \frac{1}{N}\sum_{i=1}^{N}\Big(\mathrm{Var}[(\hat{f}(\mathbf{x}_i) + f(\mathbf{x}_i))] + (\mathrm{E}[(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))])^2\Big) \tag{8.22}$$

and now using the same steps from 8.14 and 8.19 above,

$$\mathcal{R}(\hat{f}, f, N) = \frac{\sigma^2}{N^2}\sum_{i=1}^{N}\frac{1}{C_i} + \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))}}\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} h(\mathbf{x})\right)^2 \tag{8.23}$$

where C_i is a point dependent volume scaling constant. Now we using the fact that $R(y)$ follows the Reconstruction Condition we have

$$\mathcal{R}(\hat{f}, f, N) \leq \frac{\sigma^2}{N^2}\sum_{i=1}^{N}\frac{1}{C_i} + \frac{\epsilon^2}{2^{2/d}} \tag{8.24}$$

Now since C_i are non-zero constants, we can then bound them by $\frac{1}{C_i} \leq A$, such that we get

$$\mathcal{R}(\hat{f}, f, N) \leq \frac{A\sigma^2}{N} + \frac{\epsilon^2}{2^{2/d}} \tag{8.25}$$

therefore, we see that we have asymptotic convergence to a biased estimator that behaves as $\frac{1}{N}$, which is the optimal rate with convergence that depends on A, which is a function of $R(\mathbf{y})$.

8.3.1 APR reconstruction

In the above, we assumed that for every point in the domain we used $R(y)$ to estimate the point. What if instead we only estimate $f(\mathbf{x}_p)$, and then reconstruct the intensities at the other points, as in the APR. We now repeat the above steps. So now we have

$$\begin{aligned}\mathrm{E}[f(\mathbf{y}) - \hat{f}(\mathbf{y})] &= \mathrm{E}[f(\mathbf{y}) - \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \hat{f}\{\mathbf{x}_p\}\xi_p] \qquad (8.26)\\ &= f(\mathbf{y}) - \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \mathrm{E}[\hat{f}\{\mathbf{x}_p\}]\xi_p \qquad (8.27)\end{aligned}$$

using our result from 8.14 above, we have

$$\begin{aligned}\mathrm{E}[f(\mathbf{y}) - \hat{f}(\mathbf{y})] &= f(\mathbf{y}) - \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \left(f(\mathbf{x}_p) + \frac{1}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))}} \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))} h(\mathbf{x}) \right) \xi_p \\ &= \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \left(h_1(\mathbf{x}_p) + \frac{1}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))}} \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))} h_2(\mathbf{x}) \right) \xi_p \qquad (8.28)\end{aligned}$$

where h is defined similarly as above. Now $h_2(\mathbf{x})$ is bounded by $\frac{\epsilon}{2^{1/d}}$, for arbitrary particles and worst case the Reconstruction Condition gurantees $h_1(\mathbf{x}) \leq \epsilon$. Hence we have

$$\begin{aligned}|\mathrm{E}[f(\mathbf{y}) - \hat{f}(\mathbf{y})]| &\leq \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \left(\epsilon + \frac{1}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))}} \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))} \frac{\epsilon}{2^{1/d}} \right) \xi_p \\ &\leq (1 + \frac{1}{2^{1/d}})\epsilon. \qquad (8.29)\end{aligned}$$

Therefore, any reconstruction will have a an expected value with bias smaller than $(1 + \frac{1}{2^{1/d}})\epsilon$ at all points $\mathbf{y} \in \Omega$. What is the variance of our estimator? So we have

$$\begin{aligned}\mathrm{Var}[f(\mathbf{y}) - \hat{f}(\mathbf{y})] &= \mathrm{Var}[\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \hat{f}\{\mathbf{x}_p\}\xi_p] \qquad (8.30)\\ &= \mathrm{Var}[\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \left(\frac{1}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))}} \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))} g\{\mathbf{x}\} \right) \xi_p] \\ &= \mathrm{Var}[\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \left(\frac{1}{\sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))}} \sum_{\mathbf{x} \in \mathcal{N}(\mathbf{x}_p, R(\mathbf{x}_p))} \eta(\mathbf{x}) \right) \xi_p] \qquad (8.31)\end{aligned}$$

here we have to be careful because the $\eta(\mathbf{x})$ are no longer all independent due to the overlap of the neighborhood causing different original sample points enter the variance multiple times. Therefore we have to take care of these by also considering the covariance between the samples,

$$\begin{aligned}\mathrm{Var}[f(\mathbf{y}) - \hat{f}(\mathbf{y})] = &\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \xi_p^2 \mathrm{Var}[\frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{x}_p,R(\mathbf{x}_p))}} \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{x}_p,R(\mathbf{x}_p))} \eta(\mathbf{x})] + \\ & 2 \sum_{(\mathbf{x}_p,\mathbf{x}_q)\in\mathcal{N}(\mathbf{y},R(\mathbf{y})):\mathbf{x}_p\neq\mathbf{x}_q} \xi_p\xi_q \mathrm{Cov}[\frac{1}{\sum_{\mathbf{x}_0\in\mathcal{N}(\mathbf{x}_p,R(\mathbf{x}_p))}} \\ & \sum_{\mathbf{x}_0\in\mathcal{N}(\mathbf{x}_p,R(\mathbf{x}_p))} \eta(\mathbf{x}_0), \frac{1}{\sum_{\mathbf{x}_1\in\mathcal{N}(\mathbf{x}_q,R(\mathbf{x}_q))}} \sum_{\mathbf{x}_1\in\mathcal{N}(\mathbf{x}_q,R(\mathbf{x}_q))} \eta(\mathbf{x}_1)]\end{aligned} \tag{8.32}$$

which we evaluate, as the only terms that will be non-zero in the covariance will be for the cases with $\mathbf{x}_0 = \mathbf{x}_1$. If we let $\gamma_{p,q}$ be the number of shared points in the original noisy sampling for the support particles p and q then we get,

$$\mathrm{Var}[f(\mathbf{y}) - \hat{f}(\mathbf{y})] = \sigma^2 \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \frac{1}{\sum_{\mathbf{x}\in\mathcal{N}(\mathbf{x}_p,R(\mathbf{x}_p))}} \xi_p^2 + \tag{8.33}$$

$$2 \sum_{(\mathbf{x}_p,\mathbf{x}_q)\in\mathcal{N}(\mathbf{y},R(\mathbf{y})):\mathbf{x}_p\neq\mathbf{x}_q} \xi_p\xi_q \frac{\gamma_{p,q}\sigma^2}{(\sum_{\mathbf{x}_1\in\mathcal{N}(\mathbf{x}_q,R(\mathbf{x}_q))})(\sum_{\mathbf{x}_0\in\mathcal{N}(\mathbf{x}_p,R(\mathbf{x}_p))})} \tag{8.34}$$

Next, we introduce the constants C_p such that $NC_p = \sum_{\mathbf{x}\in\mathcal{N}(\mathbf{x}_p,R(\mathbf{x}_p))}$, (ignoring complications due to the discrete nature of N) . We then note that $\gamma_{p,q}$ is bounded by the smaller of the number of points in p or q, we shall choose p to be the larger, then we have,

$$\mathrm{Var}[f(\mathbf{y}) - \hat{f}(\mathbf{y})] \leq \frac{\sigma^2}{N} \left(\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \frac{\xi_p^2}{C_p} + 2 \sum_{(\mathbf{x}_p,\mathbf{x}_q)\in\mathcal{N}(\mathbf{y},R(\mathbf{y})):\mathbf{x}_p\neq\mathbf{x}_q} \frac{\xi_p\xi_q}{C_p} \right) \tag{8.35}$$

hence, our estimator converges to a biased estimate at the optimal rate of $\frac{1}{N}$ for all $\mathbf{y} \in \Omega$.

So then lastly, we consider the Risk for our reconstruction, that is the expected asymptotic behavior of the MSE, following the steps again as above,

$$\mathcal{R}(\hat{f}, f, N) = \mathrm{E}[\frac{1}{N}\sum_{i=1}^{N}(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))^2] \tag{8.36}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\mathrm{Var}[\hat{f}(\mathbf{x}_i)] + \frac{1}{N}\sum_{i=1}^{N}\Big(\mathrm{E}[\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i)]\Big)^2 \tag{8.37}$$

$$\leq \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\sigma^2}{N}\left(\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))}\frac{\xi_p^2}{C_p} + 2\sum_{(\mathbf{x}_p, \mathbf{x}_q) \in \mathcal{N}(\mathbf{y}, R(\mathbf{y})) : \mathbf{x}_p \neq \mathbf{x}_q}\frac{\xi_p \xi_q}{C_p}\right)\right)$$

$$+ ((1 + \frac{1}{2^{1/d}})\epsilon)^2 \tag{8.38}$$

again we can see that if we bound our constants that are independent of N by some constant A_1 then we have

$$\mathcal{R}(\hat{f}, f, N) \leq \frac{A_1 \sigma^2}{N} + ((1 + \frac{1}{2^{1/d}})\epsilon)^2 \tag{8.39}$$

therefore, we again have that the MSE will converge with optimal rate $\frac{1}{N}$, to a value with bias $((1 + \frac{1}{2^{1/d}})\epsilon)^2 > \epsilon^2$, where ϵ is a user set parameter.

8.3.2 Result summary

Therefore, assuming $R^*(y)$ satisfies the Reconstruction Condition for parameter $E = \epsilon$ and particles values are estimated by the original Gaussian distributed noisy samples in $R(\mathbf{y})$ of every particle, then the APR will have the following properties as the total sampling N increases:

1. Reconstruction at particle locations from the image follows, $|\mathrm{E}[\hat{f}\{\mathbf{x}_p\} - f(\mathbf{x}_p)]| < \frac{1}{2^{1/d}}$ and $\mathrm{Var}[\hat{f}\{\mathbf{x}_p\} - f(\mathbf{x}_p)] \leq \frac{\sigma}{NC_p}$
2. Reconstruction using the noisy particles at arbitrary locations follows $|\mathrm{E}[\hat{f}(\mathbf{y}) - f(\mathbf{y})]| < (1 + \frac{1}{2^{1/d}})\epsilon$ with $\mathrm{Var}[\hat{f}\{\mathbf{x}_p\} - f(\mathbf{x}_p)] \leq \frac{A_0\sigma^2}{N}$, i.e. an introduced error factor of $\frac{1}{2^{1/d}}\epsilon$
3. The expected MSE of the reconstruction follows $\mathcal{R}(\hat{f}, f, N) \leq \frac{A_1\sigma^2}{N} + ((1 + \frac{1}{2^{1/d}})\epsilon)^2$

where C_p is a constant that depends on the size of $R(\mathbf{x}_p)$, A_0 a constant that depends on the Resolution Function around $\mathbf{y}$ and the reconstruction method and sampling used. Lastly A_1 depends on the Resolution Function and reconstruction method across the domain. It is worth noting that the bound $(1 + \frac{1}{2^{1/d}})\epsilon$ is not tight and given assumptions on function within $R(\mathbf{y})$ of y, and better than worst-case reconstruction this could be reduced to be closer to ϵ. Effectively the bias introduced into the representation is a result of the spatial average used to estimate the particles.

Hence, we can see that assuming an $R(y)$ that satisfies the Resolution Bound, guarantees optimal convergence to a solution that is within bounded distance of our noise-free reconstruction. However, we have assumed that for the given APR that $R^*(\mathbf{y})$ satisfies the Reconstruction Condition and Resolution Bound. However, with noisy function input, this would not be guaranteed to hold. However, if knowledge of the relative error bounds of estimation of $L(y)$ are known, then the above results could be adapted using to incorporate the results on noisy adaptation in A.3.

The above analysis does agree well with the benchmark results shown for the test functions. In particular, the results of the Bumps and Doppler benchmarks appear to show two regions of the MSE likely corresponding to the relative sizes of the noisy convergent component and the bias $((1 + \frac{1}{2^{1/d}})\epsilon)^2$. Therefore, these preliminary results are encouraging. However, they would appear to warrant further testing and theoretical work.

8.4 Summary

In this chapter, we reflected on the APR and existing methods, comparing the concepts and ideas of the APR to those found in existing methods. Following this, we explored two optimality properties of the APR in reference to wavelet thresholding. The first related to the optimal convergence of the error relative to the number of particles, and the second with the reconstruction of noisy functions. Lastly, we provided preliminary theoretical work on the statistical properties of the APR. We provide a summary of the key points in the table below.

Summary of the chapter

- Reflected on the main components of the APR and how they are the same or similar to existing ideas in the literature
- Highlighted the Resolution Bound, and the direct focus on the Resolution Function using Particle Cells as the main novel contributions of the APR.
- Provided empirical evidence that the APR is of similar Besov norm optimality as the Haar wavelet in the L_1 error norm, but not the L_2 norm as $E \to \infty$.
- Found the APR produced high-quality noisy reconstructions with different MSE convergence properties when compared to wavelet reconstruction from the results presented in Donoho and Johnstone [47].
- Proved that given a Resolution Function $R(y)$ that satisfies the Resolution Bound, the APR MSE converges at the optimal statistical rate $\frac{1}{N}$ to a biased estimate within a constant factor of E of the noise-free APR reconstruction.

9 APR extensions in space

Contents

9.1 Higher-order reconstruction APR 214
  9.1.1 General derivative conditions 215
  9.1.2 Multiple resolution conditions 217
9.2 1D validation of higher-order reconstructions . . 217
  9.2.1 Adding higher order reconstructions 217
  9.2.2 Restricted higher order reconstructions 218
  9.2.3 Derivative reconstruction 220
  9.2.4 Summary 221
9.3 Potential use-cases 221
  9.3.1 Computational partial differential equations 222
  9.3.2 Using the APR without a Reconstruction Condition 224
  9.3.3 Anisotropic APR neighborhood 225
9.4 Summary and main points 226

In the above work, we focused on a specific case of an adaptive representation that bounded the reconstruction error of the function value using a broad case of reconstruction methods. In this chapter, we show how the APR can be generalized to adapt to a function while bounding the reconstruction of arbitrary derivatives of the function. Further, the representation can be restricted to require additional properties on the reconstruction method and hence reduce the number of particles. Following the derivation of these extended class of models, we then briefly explore their properties using a 1D test function. After we show an example of where these extended APR reconstruction conditions could be useful for the numerical solution of Partial Differential Equations. Lastly, we show that the APR can also be used in tasks where the adaptation is not guided by the reconstruction of a function using an example of particle generation for complex geometries.

9.1 Higher-order reconstruction APR

Let us consider the same set up as outlined in Chapter 5, where we instead consider a m times differentiable function $f : \Omega \to \mathbb{R}$. We represent the function for a given $\mathcal{P}$ and $R(\mathbf{y})$ in the following way

$$\hat{f}(\mathbf{y}) = \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} f(\mathbf{x}_p)\xi_p(\mathbf{y}, \mathbf{x}_p) \tag{9.1}$$

where $\mathcal{N}(\mathbf{y}, R(\mathbf{y})) = \{x \in \Omega : |\mathbf{x} - \mathbf{y}| \leq R(\mathbf{y})\}$, now further restrict our reconstruction function such that the coefficients now have to satisfy

$$\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \frac{(\mathbf{x}_p - \mathbf{y})^{\mathbf{k}}}{\epsilon^{\mathbf{k}}} \xi_p(\mathbf{y}, \mathbf{x}_p) = \begin{cases} 1 & |\mathbf{k}| = 0 \\ 0 & 0 < |\mathbf{k}| < m \end{cases} \tag{9.2}$$

and now assume $\#(\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))) > Q$ for all $\mathbf{y} \in \Omega$, where Q is some minimum required number of particles to be able to satisfy the above condition. These are the moment conditions from the DC-PSE operators [115]. Now if we again consider the reconstruction error,

$$\epsilon(\mathbf{y}) = f(\mathbf{y}) - \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} f_p \xi_p(\mathbf{y}, \mathbf{x}_p) \tag{9.3}$$

that we wish to satisfy the Reconstruction Condition for some E and $\sigma(\mathbf{y})$, by again applying the integral form of the remainder [103], we have

$$\begin{aligned} \epsilon(\mathbf{y}) = f(\mathbf{y}) - \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \Big(\sum_{|\mathbf{k}|=0}^{m-1} \frac{(\mathbf{y} - \mathbf{x}_p)^{\mathbf{k}}}{\mathbf{k}!} \frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{|\mathbf{k}|}} \Big|_{\mathbf{x}=\mathbf{y}} \\ + \sum_{|\mathbf{k}|=m} (\mathbf{y} - \mathbf{x}_p)^{\mathbf{k}} \frac{|\mathbf{k}|}{\mathbf{k}!} \int_0^1 (1-s)^{|\mathbf{k}|-1} \frac{\partial^{|\mathbf{k}|}}{\partial \mathbf{x}^{|\mathbf{k}|}} f(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y})) ds \Big) \xi_p(\mathbf{y}, \mathbf{x}_p). \end{aligned} \tag{9.4}$$

now using the conditions on a reconstruction function 9.2 we have

$$\epsilon(\mathbf{y}) = \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \sum_{|\mathbf{k}|=m} (\mathbf{y} - \mathbf{x}_p)^{\mathbf{k}} \frac{|\mathbf{k}|}{\mathbf{k}!} \int_0^1 (1-s)^{|\mathbf{k}|-1} \frac{\partial^{|\mathbf{k}|}}{\partial \mathbf{x}^{\mathbf{k}}} f(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y})) ds \xi_p(\mathbf{y}, \mathbf{x}_p), \tag{9.5}$$

now we can then again bound this term in the familiar way as

$$|\epsilon(\mathbf{y})| \leq \max_{|\mathbf{k}|=m} \max_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \Big| \frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{\mathbf{k}}} \Big| \frac{|\mathbf{k}|}{\mathbf{k}!} \gamma(m) R(\mathbf{y})^m |\xi^{m,p}| \tag{9.6}$$

where $|\xi^{m,p}| = \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} |\xi_p(\mathbf{y},\mathbf{x}_p)|$, since we no-longer have the constraint of positive coefficients and $\gamma(m) = \sum_{|\mathbf{k}|=m} 1$. Then if we again require the $R(\mathbf{y})$ satisfies the Reconstruction Condition we get,

$$R(\mathbf{y}) \leq \left(\frac{E\sigma(\mathbf{y})}{|\xi^{m,p}|\gamma(m) \max_{|\mathbf{k}|=m} \max_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} |\frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{\mathbf{k}}}| \frac{|\mathbf{k}|}{\mathbf{k}!}} \right)^{\frac{1}{m}}. \tag{9.7}$$

which then if we make a similar assumption on $\sigma(\mathbf{y})$ we have for all $|\mathbf{k}| = m$

$$\frac{1}{\sigma(\mathbf{y})} \max_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} |\frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{\mathbf{k}}}| \approx \max_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} |\frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{\mathbf{k}}} \frac{1}{\sigma(\mathbf{x})}| \tag{9.8}$$

which again is only guaranteed to hold for constant σ. We again get the familiar form of the Resolution Bound

$$R(\mathbf{y}) \leq \min_{\mathbf{x}\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} (L_m(\mathbf{x})) \tag{9.9}$$

where now

$$L_m(\mathbf{y}) = \left(\frac{E\sigma(\mathbf{y})}{|\xi^{m,p}|\gamma(m) \max_{|\mathbf{k}|=m} |\frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{\mathbf{k}}}| \frac{|\mathbf{k}|}{\mathbf{k}!}} \right)^{\frac{1}{m}}. \tag{9.10}$$

Hence, this is a problem of the correct form to be solved by the Pulling Scheme using Particle Cells. (Note, that this is a slightly worse bound than the one we used previously for $m = 1$ by using ∇f).

9.1.1 General derivative conditions

In the previous derivations, we have placed conditions on the reconstruction of the function value everywhere. Here we show that the same procedure allows bounds on arbitrary derivatives of f.

So now we wish to calculate some high order derivative α, with order m derivative operators. This requires that f is $m + |\alpha|$ times differentiable. Here, we replace the classic Reconstruction Condition with the (α, m)-Reconstruction Condition, defined as

$$|\sum_{\mathbf{x}_p\in\mathcal{N}(\mathbf{y},R(\mathbf{y}))} f_p \xi_{\alpha,p}(\mathbf{y},\mathbf{x}_p) - \frac{\partial^{|\alpha|} f(\mathbf{x})}{\partial \mathbf{x}^{\alpha}}| \leq \sigma_\alpha(\mathbf{y}) E_{\alpha,m} \tag{9.11}$$

where α uses multi-index notation, to represent the desired derivative and $\xi_p^\alpha(\mathbf{y})$ is the derivative reconstruction kernel with convergence order m. Following the same steps as above, if we require the following conditions

$$\sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} (\mathbf{x}_p - \mathbf{y})^{\mathbf{k}} \xi_{\alpha,p}(\mathbf{y},\mathbf{x}_p) = \begin{cases} 1 & \text{if}, \mathbf{k} = \alpha \\ 0 & \text{elseif}, |\mathbf{k}| < m + |\alpha| \\ \text{bounded} & \text{otherwise} \end{cases}$$

then we have the following,

$$\epsilon_\alpha(\mathbf{y}) = \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \sum_{|\mathbf{k}|=m+|\alpha|} (\mathbf{y} - \mathbf{x}_p)^{\mathbf{k}} \frac{|\mathbf{k}|}{\mathbf{k}!} \int_0^1 (1-t)^{|\mathbf{k}|-1} \frac{\partial^{|\mathbf{k}|}}{\partial \mathbf{x}^{\mathbf{k}}} f(\mathbf{y} + s(\mathbf{x}_p - \mathbf{y})) ds \xi_{\alpha,p}(\mathbf{y},\mathbf{x}_p) \tag{9.12}$$

which we can bound by,

$$|\epsilon_\alpha(\mathbf{y})| \leq \gamma(m + |\alpha|) \max_{|\mathbf{k}|=m+|\alpha|} \max_{\mathbf{x} \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \frac{|\mathbf{k}|}{\mathbf{k}!} \left(\left| \frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{\mathbf{k}}} \right| \right) R(\mathbf{y})^{m+|\alpha|} |\xi^{m+\alpha,p}| \tag{9.13}$$

where $|\xi^{m+\alpha,p}| = \sum_{\mathbf{x}_p \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} |\xi_{\alpha,p}(\mathbf{y},\mathbf{x}_p)|$. Now here, the coefficients, will be proportional to of $\frac{1}{R(y)^{|\alpha|}}$, hence we replace this with the following global bound of $|\xi^{m+\alpha,p}| \leq C_\alpha \frac{1}{R(y)^{|\alpha|}}$, where C_α is some constant that depends on the reconstruction function and local particle orientations, giving us now

$$|\epsilon_\alpha(\mathbf{y})| \leq \gamma(m + |\alpha|) \max_{|\mathbf{k}|=m+|\alpha|} \max_{\mathbf{x} \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} \frac{|\mathbf{k}|}{\mathbf{k}!} \left(\left| \frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{\mathbf{k}}} \right| \right) R(\mathbf{y})^m C_\alpha. \tag{9.14}$$

Next, by making this reconstruction error satisfy our (α, m)-Reconstruction Condition 9.11 we get

$$R(\mathbf{y}) \leq \left(\frac{E_{\alpha,m} \sigma_\alpha(\mathbf{y})}{C^\alpha \gamma(m + |\alpha|) \max_{|\mathbf{k}|=m+|\alpha|} \frac{|\mathbf{k}|}{\mathbf{k}!} \max_{\mathbf{x} \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} |\frac{\partial^{|\mathbf{k}|} f(\mathbf{x})}{\partial \mathbf{x}^{\mathbf{k}}}|} \right)^{1/m}. \tag{9.15}$$

which if we again apply a smoothness assumption on σ_α making the usual substitution giving again

$$R(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y},R(\mathbf{y}))} (L_{\alpha,m}(\mathbf{x})) \tag{9.16}$$

where now

$$L_{\alpha,m}(\mathbf{y}) = \left(\frac{E_{\alpha,m} \sigma_\alpha(\mathbf{y})}{C^\alpha \gamma(m + |\alpha|) \max_{|\mathbf{k}|=m+|\alpha|} \frac{|\mathbf{k}|}{\mathbf{k}!} |\frac{\partial^{|\mathbf{k}|} f(\mathbf{y})}{\partial \mathbf{x}^{\mathbf{k}}}|} \right)^{1/m} \tag{9.17}$$

which is again in the correct form for using Particle Cells and the Pulling Scheme. We can see the condition in the previous section is simply the $\alpha = 0$ case.

9.1.2 Multiple resolution conditions

Above we have shown that we can formulate the Resolution Bound for a range of (α, m)-Reconstruction Conditions. What if we want more than one Reconstruction Condition? This case is simply satisfied. If we consider we have a set of Local Resolution Estimates $\mathbf{L}_i = \{L_{\alpha_i,m_i}(\mathbf{y})\}$ from $i = 1, .., q$ associated with q different (m, α)-Reconstruction Conditions (9.11), then we get one Resolution Bound of the form,

$$R(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathcal{N}(\mathbf{y}, R(\mathbf{y}))} \left(\min_i L_{\alpha_i,m_i}(\mathbf{x}) \right) \tag{9.18}$$

using the fact that the minimum operation is associative. Hence, any combination of Reconstruction Conditions can be solved finding the Implied Resolution Function $R^*(y)$ using the Pulling Scheme using the minimum across the different $(\alpha$,$m)$-Local Resolution Estimates. Hence, all of the results from the Particle Cells presented for the $m = 1$, $\alpha = 0$ case, directly extend without extra work to the multiple general (m, α)-Reconstruction Conditions case.

9.2 1D validation of higher-order reconstructions

In this section, we briefly explore the above results in 1D using a simple test function $f : \Omega \to \mathbb{R}$

$$f(y) = e^{\frac{-(x-0.5)^2}{0.005}} - e^{\frac{-(x-0.3)^2}{5}} + e^{\frac{-(x+5)^2}{0.1}} \tag{9.19}$$

where $\Omega = [-10, 10]$. We use a constant Local Intensity Scale $\sigma_0 = 1$. We will use the same benchmark example throughout the different test cases. The implementation used for sampling and Pulling Scheme is the same as used for the 1D test cases in Chapter 6 and described in 6.1.1.

9.2.1 Adding higher order reconstructions

First, we consider what happens if we wish to guarantee that the APR can be used with reconstruction functions from order $m = 1, ..q$. This will result in Local Resolution Estimates of the form

$$L_{i,0}(y) = \left(\frac{E_{0,i}\sigma_0}{\frac{i}{i!} \left| \frac{\partial^i f}{\partial y^i} \right|} \right)^{1/i} \tag{9.20}$$

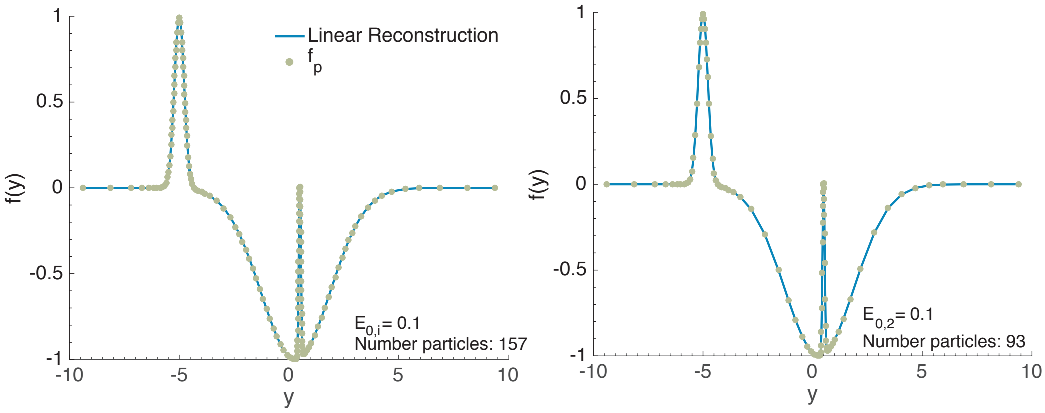


Figure 9.1: The first plot shows the APR satisfying (0,i)-Reconstruction Condition's from $i = 1, 2, 3$ for test function 9.19 and $E_{0,i} = 0.1$. The second plot shows the same function but represented by an APR only satisfying the (0,2)-Reconstruction Condition for for test function 9.19 and $E_{0,2} = 0.1$. The same sampling method (1 particle per Particle Cell) was used for both.

with the simplifying assumption that $|\xi^{m,p}| = 1$, which holds for piecewise linear interpolation used here. We then let $\mathbf{L}(q) = \{L_{0,1}, .., L_{0,q}\}$ and let $L_{min}(y) = \min_{i=1,..q}(L_{i,0}(y))$ where we set $E_{0,i} = E$. That is we wish all the higher order reconstructions to also satisfy the $(0,i)$-Reconstruction Conditions. The APR is then formed as usual simply using $L_{min}(y)$ in place of $L(y)$. The first plot in Figure 9.1 shows the APR for $q = 3$. For this test function, and others tried, we find that the APR from adding higher q only slightly alters the solution in low-resolution areas. The APR for only $q = 1$ requires 151 particles. The difference is better reflected in the first plot of Figure 9.2, showing the Local Resolution Estimates and the Implied Resolution Function. The higher order q Local Resolution Estimates slightly broaden L_{min} at higher function values. This behavior is consistent across test functions tried, with many having no change in the APR with the increase in q. However, as q increases so do the required number of particles within the support of $R^*(y)$. Hence, alternative samplings than those we have used previously would be required ($q = 2$ and linear sampling is still satisfied by the classic sampling).

9.2.2 Restricted higher order reconstructions

Next, we consider the case where we forego the lower order construction and require that our reconstruction is of order m. That is, the APR then satisfies only a $(0,m)$-Reconstruction Condition. First, we show results for $m = 2$ case, with piecewise linear reconstruction used. The APR for $E_{0,2} = 0.1$ is shown in the second plot in Figure 9.1. We can see compared to the $m = 1$ restricted

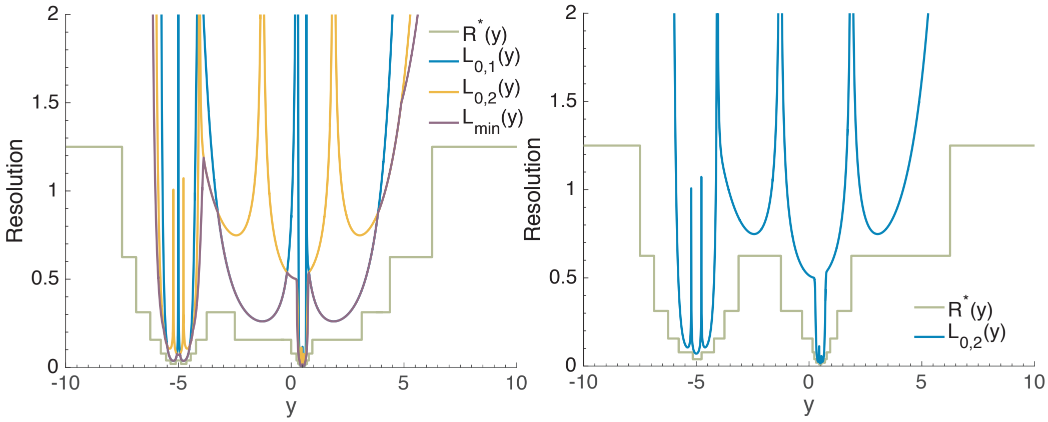


Figure 9.2: The first plot shows the different Local Resolution Estimates $L_{0,i}$ for the (0,i)-Reconstruction Conditions, the minimum across them L_{min} and the Implied Resolution Function R^* for $i = \{1, 2\}$. The second plot shows the Local Resolution Estimate for $L_{0,2}$ case and the Implied Resolution Function $R^*(y)$ for only the (0,2)-Reconstruction Condition.

case the number of particles is reduced from 151 to 93. The change in sampling is also reflected in the second plot of Figure 9.2 for the Local Resolution Estimate and Implied Resolution Function. To verify that this new APR does satisfy the (0,2)-Reconstruction Condition, in the first plot of Figure 9.3 we show the observed reconstruction error E^* for changing $E_{0,2}$. For this, we use linear piecewise reconstruction that satisfies the $m = 2$ reconstruction criteria. Indeed, we find that for all values of relative error the (0,2)-Reconstruction Condition is satisfied. However, now the (0,1)-Reconstruction Condition no longer holds, as the piecewise constant and worst-case reconstruction methods are no longer below the dashed line.

Next, we focus on how the Implied Resolution Function changes as the reconstruction constraint is increased. That is, how does $\#\mathcal{V}$ change as we only restrict the APR to the (0,m)-Reconstruction Condition for increasing m. We focus here on $\#\mathcal{V}$ to avoid issues of increases in the number of particles N_p since for higher order models $\#\mathcal{V} \neq N_p$. We leave considerations of this trade off to future work.

The second plot of Figure 9.3 shows a log-log plot of $\#\mathcal{V}$ against $E_{0,m}$ for the APR formed satisfying the (0,m)-Reconstruction Condition for $m = 1, 2, 3, 4, 5, 6$. The red arrow indicates the direction of increasing m. We note that for small E, the higher order models results in orders of magnitudes reductions in the number of particle cells in $\#\mathcal{V}$. However, at values of $E \approx 0.1$ ($\log(0.1) \approx -2.3$), the estimates are all within an order of magnitude. Hence, for highly accurate reconstructions high order reconstructions appear to be very efficient, with an affect that reduces as E increases.

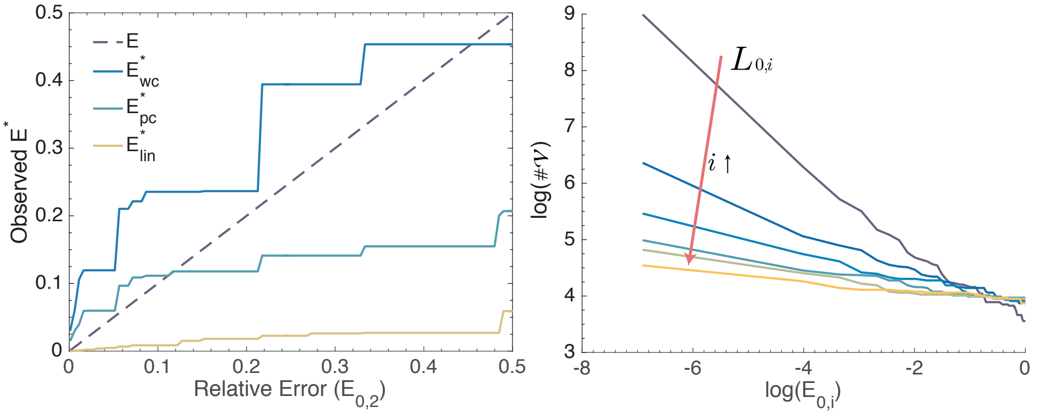


Figure 9.3: The first plot shows the observed reconstruction error E^* for the worst-case, piecewise constant, and piecewise linear reconstruction methods against relative error $E_{0,2}$ for 100 values between 0.001 and 0.5. The second plot shows the size of the OVPC set $\mathcal{V}$ for the APR formed with $L_{0,i}$ for $i = 1, 2, 3, 4, 5, 6$ in a log-log plot against the set relative error $E_{0,i}$. The arrow indicates the direction of increasing recontruction method i.

9.2.3 Derivative reconstruction

Lastly, we explore the bounding of a functions derivative for first order kernels. That is, we wish the APR to now only satisfy the (1,1)-Reconstruction Condition. We recall that in the 1D benchmarks for the classic APR in 6.1.6, that for the benchmark function tested the first order kernel did not satisfy a (1,1)-Reconstruction Condition when the APR was formed using only a (0,1)-Reconstruction Condition. Now, in the examples above the same Local Intensity Scale σ is used. However, here to make the relative error for $E_{1,1}$ have a similar meaning we set $\sigma_1 = 12.162$, that is the maximum of the absolute value of the gradient across the domain. For the Local Resolution Estimate, we use

$$L_{1,1}(y) = \left(\frac{E_{1,1}\sigma_1}{2|\frac{\partial^2 f}{\partial y^2}|} \right) \tag{9.21}$$

where for the gradient computation we used the DC-PSE [115] kernel and Particle Cell sampling that gives $C_\alpha = 2$. In the first plot of Figure 9.4 we show the linear reconstruction and particle sampling for the (1,1)-Reconstruction Condition APR for $E_{1,1} = 0.1$. We find that the number of particles required is greater than the (0,1)-Reconstruction Condition. Further, the particle distribution is different from the previous examples. To assess if the (1,1)-Reconstruction Condition holds we again test the observed reconstruction error E^* of the first order gradient against increasing relative error $E_{1,1}$. The second plot of Figure 9.4 shows the that it does indeed hold for the given test function.

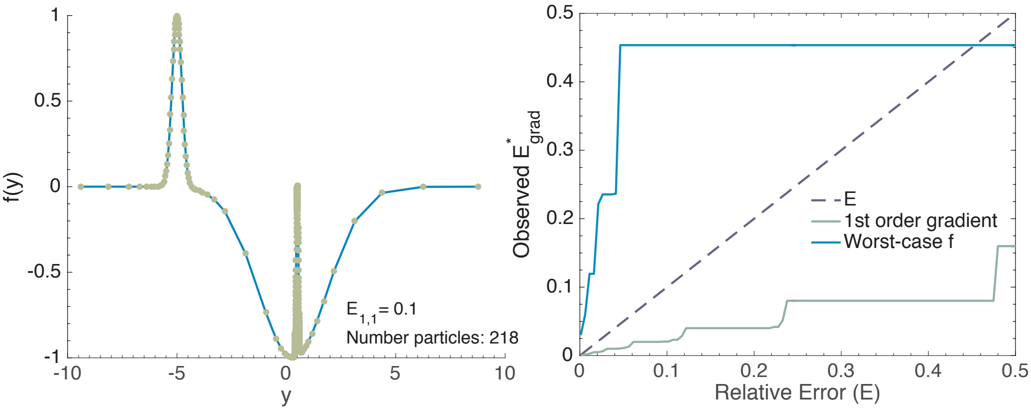


Figure 9.4: The first plot shows the particle sampling and linear reconstruction for test function 9.19 for the APR satisfying the (1,1)-Reconstruction Condition for $E_{1,1} = 0.1$. The second plot shows the observed reconstruction error E^* for the first-order gradient and worst-case function reconstruction for an APR satisfying the (1,1)-Reconstruction Condition for increasing E from 0.001 to 0.5.

In addition to the first order gradient reconstruction error, we also plotted the worst-case reconstruction of f. We see that the (0,1)-Reconstruction Condition does not hold for any $E_{1,1}$.

9.2.4 Summary

In the above section, we briefly explored some examples of extensions of the APR. We note that although we only showed specific examples and combinations, the work above provides an effective 'toolbox' such that any combination of (α, m)-Reconstruction Condition can be used in the above way to adapt to a function in a specific way. However, with the caveat, that one must find appropriate bounding constants for the higher order models, and appropriate sampling strategies. Further, in higher dimensions, the number of derivatives required as m or α increases can become large. We leave such models to future research.

9.3 Potential use-cases

Detailed explorations of further use-cases of the APR goes beyond the scope of this thesis. However, here we briefly detail how the APR could be used for simulations on particles. First, we briefly show results of how the APR could be used for the numerical solution of partial differential equations. Second, we show a simple example of how the Local Resolution Estimate, $L(\mathbf{y})$ need not originate from some Reconstruction Condition for generation of particles for a complex geometry.

9.3.1 Computational partial differential equations

In this section, we provide an example of how the 'toolbox' of APR representations could be useful in practice. We do this by using the classic test case example of numerically solving the 1D viscous Burgers equation. Here we numerically solve for a particular initial condition of

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial y}\left(\frac{1}{2}f^2\right) = \epsilon\frac{\partial^2 f}{\partial y^2} \tag{9.22}$$

with $\epsilon = 0.001$ and $\Omega = [0, 1]$, $t = [0, 0.0625]$ with Dirichlet boundary conditions $f(0, t) = 1$ and $f(1, t) = 0.1$. Where we use the $t = 0$ of a known exact solution for the initial condition. The solution is defined as

$$f(y,t) = \frac{0.1r_1 + 0.5r_2 + r_3}{r_1 + r_2 + r_3} \tag{9.23}$$

$$\begin{aligned} r_1 &= e^{\frac{-y+0.5-4.95t}{20\epsilon}} \\ r_2 &= e^{\frac{-y+0.5-75t}{4\epsilon}} \\ r_3 &= e^{\frac{-y+0.375}{2\epsilon}}. \end{aligned} \tag{9.24}$$

Here we use explicit first order Euler time stepping and differential operators computed using non-uniform finite differences computed as DC-PSE operators [115] with no exponential window function (for simplicity). The solution is computed by evolving the solution over APR particles, reforming the APR every time step using the previous time step solution to compute the new adaptation and then evolving the solution forward a time step. Cubic interpolation was used for increases and decreases in resolution from the APR adaptation. Two different forms of the APR were tested. One satisfying the (0,1)-Reconstruction Condition, and the second adapting to multiple conditions $\{(0, 1), (1, 1), (2, 1)\}$. That is, the first used the classic APR, with an adaptation of only the function error, the second also adapted to the first and second derivatives. For both cases $E_{i,i} = 0.1$ was used and l_{max} was set by $L_{(0,1)}$.

The first plot of Figure 9.5 shows the solution at four different time points for the $\{(0, 1), (1, 1), (2, 1)\}$-RC APR. The solution adapts to the two steep fronts of the solution through time. To better show this adaptation in the second plot of Figure 9.5 we show this adaptation of particles by plotting

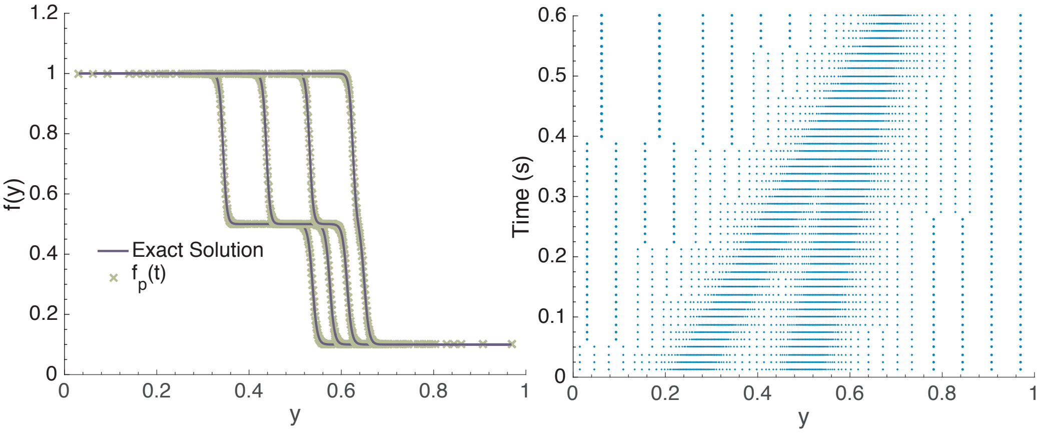


Figure 9.5: The first plot shows a comparison of the $\{(0,1),(1,1),(2,1)\}$-RC APR with the exact solution for four different time steps at intervals of 2000 time steps. The second plot shows the location of the particles as a function of time for the $\{(0,1),(1,1),(2,1)\}$-RC APR plotted every 200 time steps.

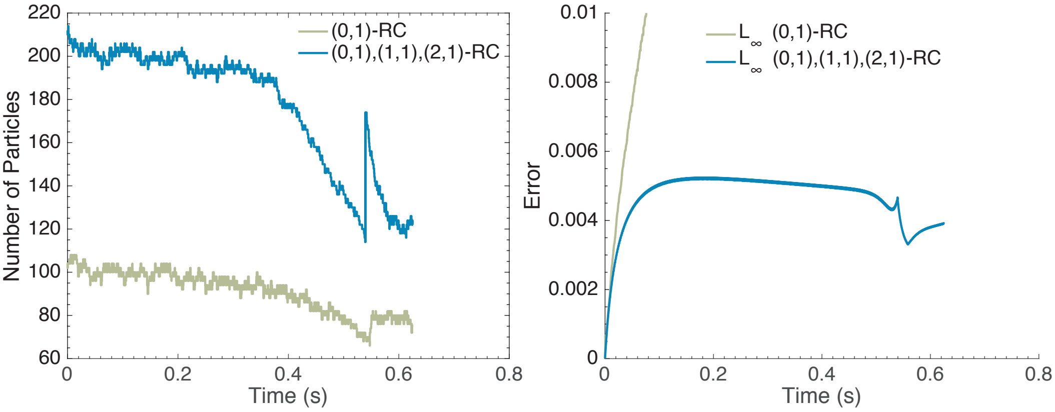


Figure 9.6: The first plot shows the number of particles used for the (0,1)-RC APR (green) and $\{(0,1),(1,1),(2,1)\}$-RC APR (blue) over time. The maximum sampling was equivalent to $N = 2048$ for a full mesh solution. The second plot shows a comparison of the L_∞ error of the two APR solutions against time. The (0,1)-RC APR (green) solution continues to grow almost linearly with time to a value of approximately 0.4 across the interval. With the solution being of similar shape, but having a slightly slower speed.

the distribution of particles through time. The number of particles required remained relatively constant through time ranging between $120-200$ particles, while the effective maximum resolution used was equivalent to $N = 2048$. This is shown for both APRs in the first plot of Figure 9.6. We see that the multiple conditions APR corresponded to almost twice the amount of particles required to satisfy the classic $\{(0,1)\}$-RC APR (also in the figure). However, we found that the solution although stable, was not consistent, with the L_∞ error of the solution diverging linearly with time. However, we found that the L_∞ of the $\{(0,1),(1,1),(2,1)\}$-RC APR stayed bounded through time as shown in the second plot of Figure 9.6. Hence, the multiple conditions seem necessary for a convergent solution for this problem. We note that care must be taken with constant time stepping and increases in resolution l_{max}, as an increase in local resolution can result in the problem no longer being stable through time. A simple solution to this would be setting an adaptive global time step set according to l_{max}.

For reference regarding the computational time taken we also computed a homogenous grid solution using the same method for $N = 2048$, representing the highest resolution required for the APR. We found that the average time per time step of the APR was 0.062 seconds and 0.25 for the full grid solution. Indicating that the APR time step had a PP-ratio of around 1/3. However, no efforts were made to optimize this solution, and the derivatives were calculated twice for the APR for the adaptation and then again for time stepping. Further, of this time calculation of the pulling scheme accounted for only .0012 seconds on average per time step (2% of the computational cost).

We note that the ability to adapt successfully to such 1D equations is not novel (For example the simple example we showed earlier from Vasilyev and Bowman [131]). However, the above illustrates the potential benefit of the extensions of the APR, over the classic case, and indicates that further research into the application of the APR for the numerical solution of differential equations is warranted.

9.3.2 Using the APR without a Reconstruction Condition

In all of the examples giving in the rest of this thesis, the APR has been adapted using a Local Resolution Estimate $L(\mathbf{y})$ that arose from error analysis from some Reconstruction Condition of a function f. However, the principles of Particle Cells and the Pulling Scheme are not dependent on the source of the Local Particle Cell Set $\mathcal{L}$. Hence, $\mathcal{L}$s can be used that arise from alternative sources. One simple example of this is using the APR for mesh generation

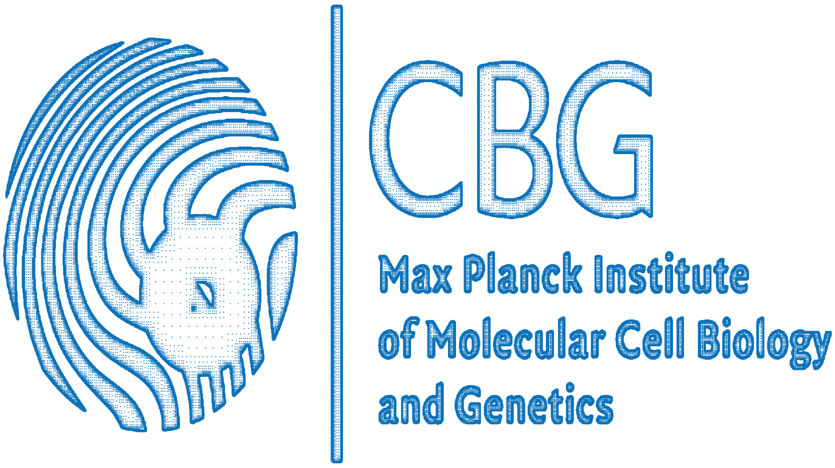


Figure 9.7: An example of the APR and the Pulling Scheme used with a Local Particle Cell set that does not originate from a Local Resolution Estimate $L(\mathbf{y})$. Instead, here the elements of $\mathcal{L}$ came from a boundary of a complex geometry generated from the Max Planck Institute of Molecular Cell Biology and Genetics official logo.

of complex geometries. In this case, the $\mathcal{L}$ can be set to boundary elements of complex geometry, and then a solution generated, with those Particle Cells inside the domain used to define a mesh. An example of such an approach is shown in Figure 9.7 for a complex geometry generated from the binarization of the logo of the Max Planck Institute of Molecular Cell Biology and Genetics (where this research was undertaken).

Whether or not such generated particle distributions are useful or not, I am not sure. However, the example provides an example of the generality of Particle Cells and the Pulling Scheme to problems beyond function approximation.

9.3.3 Anisotropic APR neighborhood

The use of the APR with an isotropic neighborhood is not necessary. Further 'non-classical' APR representations can be generated with slight alternations of the methods for Particle Cells and the Pulling Scheme. One approach that would likely lead to significant reduction in total particle number would be the extension of the Resolution Function from a scalar to a vector valued function. However, for brevity, we do not explore such extensions here and leave their exploration to future research.

9.4 Summary and main points

In this chapter, we have discussed generalizations of the APR in space. First, we introduced a wider class of adaptive representations that directly use Particle Cells and the Pulling Scheme but satisfy more general (α,m)-Reconstruction Conditions. Where α indicates the function derivative that is to be bounded and m the minimum order of the reconstruction function used. These different conditions can be easily combined, providing a 'toolbox' for generating adaptive representations using the APR. We then provided three different categories of benchmark results. First, we used a 1D test function to show the behavior of different (α,m)-Reconstruction Conditions and their resulting APR. We then used an example of solving the 1D viscous Burgers equations where the use of the generalized APR was useful. Lastly, we showed how the APR could be used for applications without an associated Reconstruction Condition using the example of particle generation for a complex geometry. We provide a further summary of the key-points of this chapter below.

Summary of the chapter

- Extended the APR to general (α,m)-Reconstruction Conditions utilizing higher order derivatives
- Showed this results in a Resolution Bound solvable by Particle Cells and the Pulling Scheme with no adaptation simply a change in $L(\mathbf{y})$
- The results provide a 'toolbox' for arbitrary combinations of (α,m)-Reconstruction Conditions APRs
- Validated the results in 1D for a range of combinations of (0,m)-Reconstruction Conditions
- Provided an example of a pure gradient adapted APR satisfying the (1,1)-Reconstruction Condition
- Showed a benchmark example of a solution of the 1D viscous Burgers equation where the use of multiple conditions allowed a consistent and stable solution through time for a benchmark example where the classic APR did not.
- Illustrated how the APR could be used for tasks where the $L(\mathbf{y})$ does not arise from a Reconstruction Condition using particle generation for a complex geometry.

10 Extensions in time ($\mathbf{APR}_t$)

Contents

10.1 $\mathbf{APR}_t$ 228

10.1.1 Theory 228

10.1.2 Algorithm 229

10.2 Benchmark 1D 230

10.2.1 Time Reconstruction Condition 231

10.2.2 Temporal adaptation 232

10.2.3 Relative size of update, add and remove 235

10.2.4 Summary 235

10.3 Extension to LSFM data 237

10.3.1 Complication of noise 237

10.3.2 Real-time optimizations 237

10.4 Summary and main points 237

In all previous work above we ignored adaptation through time. However, as discussed in Chapter 2 and Chapter 3, LSFM data also shows varying temporal scales. Hence, ideally the APR should also be able to adapt to different temporal scales, as described in the fifth representation criteria:

- **RC5**: The representation must also be able to similarly account for varying temporal scales.

Even though time could be considered as 'just another dimension,' in the context of LSFM data, it has special properties. First, the data is generated in temporal order, hence if we wish to be able to perform any real-time processing tasks, we should be able to generate the representation without complete knowledge of the function through time. Real-time processing would not be possible using the classic APR approach with time treated as an additional spatial dimension. Further, it would be ideal if the APR in space would maintain its present representation such that all the previous processing analysis could still be valid. Lastly, any Reconstructions should be 'causal', in that they only use information from the past. This criterion is justified as to keep temporal ordering of any events detected in subsequent processing steps.

10.1 APR$_t$

In this chapter, we outline one possible temporal-method we call APR$_t$ that has the above-mentioned properties. First, it can be constructed through time, with only information of the current time step, and a current state APR, which is kept updated through time. Second, the APR$_t$ adapts through time, taking into account different temporal scales, while still allowing the reconstruction of the classic APR exactly in terms of Particle Cells and within some tolerance E_t regarding the spatial reconstruction error E. Hence, all algorithms on designed for the classic APR can be directly used. Lastly, the APR$_t$ is causal, only reconstructing the function using information from the 'past'.

In the section below, we will briefly outline the extension of the theory and give then describe the additional algorithm steps to construct the APR$_t$. We present this section as a 'proof of principle', and leave more rigorous treatment, exposition, and exploration to future work.

10.1.1 Theory

We extend the previous scenario to now consider a time varying function $f : \Omega \times \Omega_t \to \mathbb{R}$ defined over a spatial domain Ω and temporal domain Ω_t and assume it is once differentiable in both space and time. Now we represent a function with a set of particles $\mathcal{P}_t$ and two time varying Resolution Functions, one in space defined in the usual way $R(\mathbf{y}, t)$ and an additional resolution function for time $T(\mathbf{y}, t)$. Hence, this is an example of an anisotropic APR formulation. We then reconstruct the function at any point in space and time in the domain as,

$$\hat{f}(\mathbf{y}) = \sum_{\mathbf{x}_{p,t} \in \mathcal{N}(\mathbf{y},t,R(\mathbf{y},t),T(\mathbf{y},t))} f(\mathbf{x}_{p,t})\xi_{p,t}(\mathbf{y}) \tag{10.1}$$

where now $\mathbf{x}_{p,t}$ defines both a spatial and time coordinate $\mathbf{x}_{p,t} = (\mathbf{y}_p, t_p)$ and we assume our reconstruction function again sums to one and is positive as in the original APR case. Now the neighbourhood is isotropic in space, and backwards in time. That is, $\mathcal{N}(\mathbf{y}, t, R(\mathbf{y}, t), T(\mathbf{y}, t)) = \{(\mathbf{x} \in \Omega, s \in \Omega_t : |\mathbf{x} - \mathbf{y}| \le R(\mathbf{y}, t), 0 \le t - s \le T(\mathbf{y}, t)\}$ which we now wish to satisfy the time Reconstruction Condition

$$|\hat{f}(\mathbf{y}, t) - f(\mathbf{y}, t)|_\infty \le E_t + E \tag{10.2}$$

where we a spatial relative error E and a temporal relative error E_t and we drop the use of Local Intensity Scale σ for simplicity and the infinity norm is across both space and time. We now formulate the error in the usual way, we shall use the $m = 1$ case again

$$|\epsilon(\mathbf{y},t)| \leq R(\mathbf{y},t) \max_{\mathbf{x}\in\mathcal{N}_s(\mathbf{y},t,R(\mathbf{y},t))}(|\nabla f(\mathbf{y},t)|)+ \\ |\sum_{x_{p,t}\in\mathcal{N}(\mathbf{y},t,R(\mathbf{y},t),T(\mathbf{y},t))}(t-t_p)\xi_{p,t}(y)\int_0^1 \frac{\partial}{\partial t}f(\mathbf{x}_p,t_p+s(t_p-t))ds| \tag{10.3}$$

where the error is decomposed into a time path and space path seperately, where $\mathcal{N}_s$ and $\mathcal{N}_t$ correspond to the appropriate restricted in space and time neighborhoods. Then we can now bound this now using $T(\mathbf{y},t)$ as

$$|\epsilon(\mathbf{y},t)| \leq R(\mathbf{y},t) \max_{\mathbf{x}\in\mathcal{N}_s(\mathbf{y},t,R(\mathbf{y},t))}(|\nabla f(\mathbf{y},t)|)+ \\ T(\mathbf{y},t) \max_{s\in\mathcal{N}_t(\mathbf{y},t,T(\mathbf{y},t))}(|\frac{\partial f(\mathbf{y},s)}{\partial t}|) \tag{10.4}$$

now we bound each seperately as,

$$T(\mathbf{y},t) \leq \frac{E_t}{\max_{s\in\mathcal{N}_t(\mathbf{y},t,R(\mathbf{y},t))}(|\frac{\partial f(\mathbf{y},s)}{\partial t}|)} \tag{10.5}$$

$$R(\mathbf{y},t) \leq \frac{E}{\max_{\mathbf{x}\in\mathcal{N}_s(\mathbf{y},t,T(\mathbf{y},t))}(|\nabla f(\mathbf{y},t)|)} \tag{10.6}$$

given both of these hold, then the time Reconstruction Condition 10.2 holds. Now we note that both of these are Resolution Bounds of the correct form for the use of particle cells and the pulling scheme. Further, the first condition is simply the classic APR condition. Now if we assume that we first integrate forward in time, then do the spatial step. We can see that we simply need the time Resolution Function to hold on all particle locations.

10.1.2 Algorithm

Now we describe the basic principles of the formation of the APR$_t$. For each time step t the classic APR is formed as usual. Following this, $\mathcal{V}(t)$ is compared to $\mathcal{V}(t-1)$, and any additions (unique to $\mathcal{V}(t)$) are added to a set called ALL$_t$, along with the particle intensity, and removals (unique to $\mathcal{V}(t-1)$) are added to a set REMOVE$_t$ (Difference in both directions between the sets). For all those particle cells that are in both $\mathcal{V}(t)$ and $\mathcal{V}(t-1)$ are only stored if they are required to ensure that 10.5 is satisfied. This is done by considering each Particle Cell in space, as also having a temporal level l_t. Because the reconstruction neighborhood is one sided, the Pulling Scheme is easily adapted, and only requires the current particle cell level as input. For brevity, we do not describe these steps in detail. However, it is $\mathcal{O}(1)$ per particle per time step.

An additional data structure APR_c is maintained that contains the previous particle information, and the type and level l_{t-1} at the previous time step. The one difference from spatial Particle Cells, to their time extension, is that we consider the particle to be sampled at the far edge of the Particle Cell (in comparison to the particle placed in the center as in the classic case). Such that, once a new Particle Cell is required by the Pulling Scheme, the particle will also be sampled. Any time a new Particle Cell is required, the location and particle intensity is stored in an additional set called UPDATE_t.

These steps are summarized in Algorithm 9. The APR_t then consists of three sets, that is APR_t=$\{\mathrm{ADD}_t, \mathrm{REMOVE}_t, \mathrm{UPDATE}_t\}$, combined with the classic APR at $t = 0$. We note that the add and update sets require storage of both the Particle Cell and particle function value, where as the remove, simply require the removed Particle Cell.

From this APR_t, a classic APR for a given t can be reconstructed, with particle intensities, which are updated such that any reconstruction from this resulting APR will follow the Reconstruction Condition with the added factor of E_t. Further, additional information is known at each particle location regarding the spatial resolution function $T(\mathbf{y}, t)$ through the Particle Cell level that is implicit in the sampling. Reconstructed in this way, the APR_t requires time order access and therefore does not allow $\mathcal{O}(1)$ reconstruction of arbitrary function values in time. However, given data-sets are usually processed and viewed in time order, I do not believe this is a prohibitive restriction.

10.2 Benchmark 1D

We briefly validate the APR_t above in 1D using the following test function of moving Gaussian pulses, defined as

$$f(y,t) = \sum_{i=1}^{G_n} e^{\frac{-(y-y_i+u_i t)^2}{s_i}} \tag{10.7}$$

defined on $\Omega = [-10, 10]$ and $\Omega_t = [0, 2]$ where for the examples here we have $G_n = 6$ and $y_i \sim U(-10, 10)$ and $s_i \sim U(0, 0.2)$ and $u_i \sim U(-u^*, u^*)$ and U is the random uniform distribution. This corresponds to Gaussian pulses of random width, being placed randomly with speeds that are distributed within a range set by u^*. In the benchmarks below we use $N_t = 1000$, and therefore dt of .002. At each time step, the APR is generated from the new function values

```
Data: Sampled function $f\{\bar{x},\bar{t}\}$
Result: APR$_t$={ADD$_t$,REMOVE$_t$,UPDATE$_t$}
Initialize APR$_c$ with APR($t_0$);
for $t_c = (t_0 + dt) : dt : t_f$ do
    1.) compute APR($t_c$) for $t_c$;
    2.) compare to APR$_c$;
    2.) Add all new Particle Cells and their particle to ADD$_t$;
    3.) Add all removed Particle Cells to REMOVE$_t$;
    4.) For all Particle Cells present in previous solution, compute $|\frac{\partial f}{\partial t}|$,
      and the local Particle Cell level $l^*_{t,i}$ using APR$_c$;
    5.) compute ($l_{t,i}$,$i_t$,APR$_c$)=causal_pulling_scheme($l^*_{t,i}$,APR$_c$);
    6.) If a new Particle Cell is required, add particle and Particle Cell
      to UPDATE$_t$;
    7.) Update APR$_c$ with current time step particle info from APR($t_c$);
end
```

Algorithm 9: Summary of the steps to calculate the APR$_t$, which is represented by three sets for every time step, ADD$_t$, REMOVE$_t$ and UPDATE$_t$. For these three sets, the APR at any time step can be recreated in temporal order, with a spatial reconstruction error bounded by $E_t + E$.

at the old particle locations, as if generated as in solution of a differential equation. All of the results here are consistent with the case where instead at each time step the new APR is produced from a uniform grid of size N. In Figure 10.1, we give examples of two time points of this test function for an early and late time point. The reconstruction from the original APR and the particles from APR$_t$ are shown. Little difference can be seen by eye. However, if one focuses on the fourth peak in the first plot one can see what looks like disordered particles in their placement.

Here we only provide a brief survey of indicative results; we again leave more rigorous benchmarking to future work.

10.2.1 Time Reconstruction Condition

First, we consider the time Reconstruction Condition 10.2. We test the reconstruction by using the fact that a spatial APR for each time step can be formed. For the particle function values, this corresponds to a 'causal' nearest neighbor in time interpolation step followed by the classic spatial reconstruction step. We test the usual piecewise constant, worst-case, and piecewise linear construction methods. In Figure 10.2, we present results for $E = 0.08$

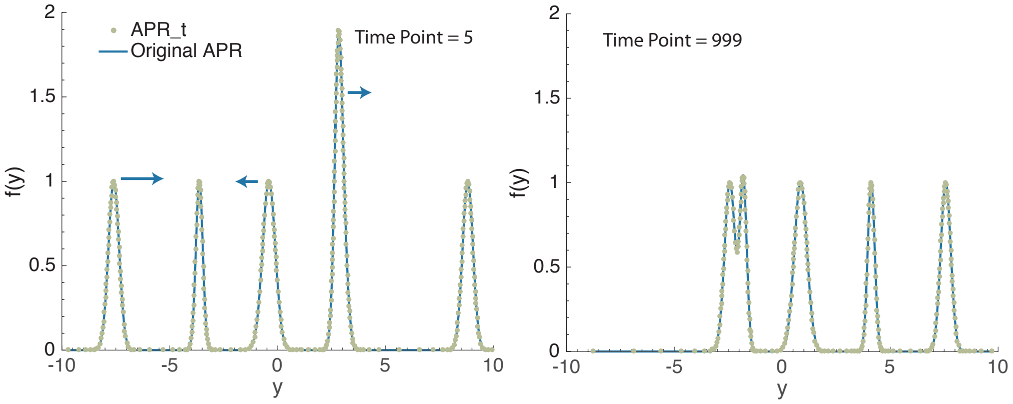


Figure 10.1: Examples from two time points of the simple test function defined in 10.7, used to validate the APR_t. The first plot shows the fifth time point, and the second plot the 999th with the linear interpolation of the original APR, and $u^* = 4$, $E = 0.08$ and $E_t = 0.03$, with $dt = 0.002$. The arrows are added to illustrate the random direction of movement of the peaks.

and $E_t = 0.01$ in the first plot and $E_t = 0.03$ in the second, where the observed reconstruction error E^* is plotted against time. For the time Reconstruction Condition to hold, all reconstructions should have an error less than $E + E_t$ for all time points. We find this is the case for the three reconstruction methods presented as they are below the red dashed line for all t. We note the oscillations reflect the adaptation through time.

Further, in both cases, the worst case reconstruction is above the blue dashed line represented the classic APR's Reconstruction Condition. Breaching this bound reflects the lossy adaptation that is now occurring through time.

10.2.2 Temporal adaptation

Now we assess whether this formulation allows the APR_t to adapt to spatial scales. First, we consider the limiting case where $u^* = 0$ such that the function is constant through time. In this case, the temporal scale (one-sided) at each point, is effectively getting larger and larger as time time passes. Now if we consider an image data set with N points in space and N_t sample points, we can then define the Computational Ratio (CR) again as

$$CR(N_t) = \frac{N N_t}{\# APR_t} \tag{10.8}$$

where $\#\ \mathrm{APR}_t = \#\mathrm{ADD}_t + \#\mathrm{REMOVE}_t + \#\mathrm{UPDATE}_t$, i.e. the sum of its parts. Then for the case where the function is not changing, the perfect case

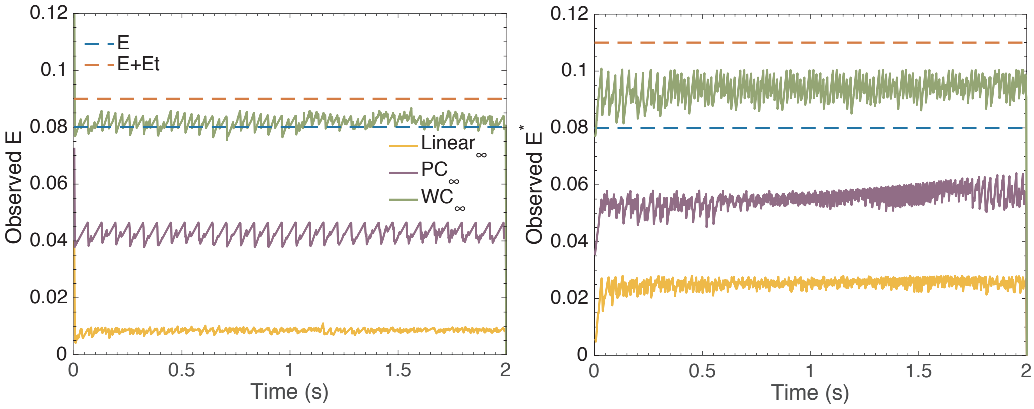


Figure 10.2: The observed reconstruction error E^* (Infinity norm at $N = 1000$ evenly spaced locations) for piecewise linear interpolation, piecewise constant interpolation, and worst-case reconstruction. The first plot is for $E = 0.08$ for the classic APR, and $E_t = 0.01$ for the time adaptation, the second plot is for $E_t = 0.03$. The dsahed red line represents the time Reconstruction Condition 10.2. The maximum speed was set to $u^* = 4$.

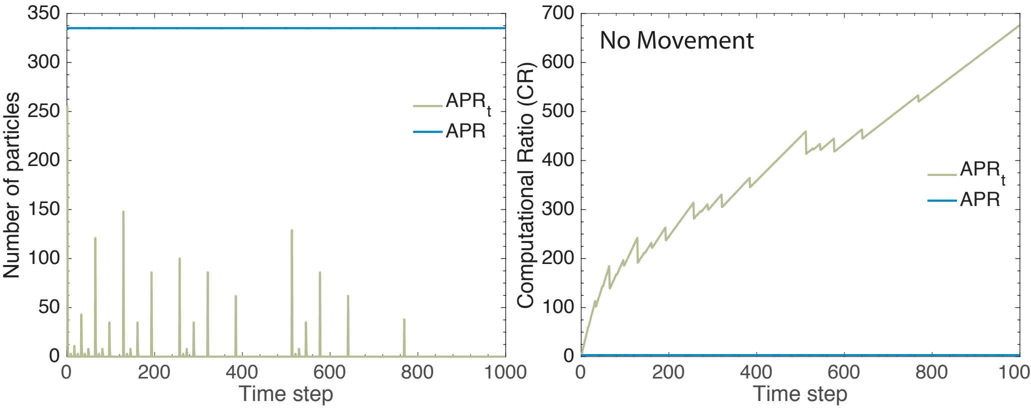


Figure 10.3: The first plot shows the number of particles in the classic APR for each time step (blue) and the equivalent size (# APR$_t$) of the APR$_t$ (green) for the time test function with **no movement** ($u^* = 0$). The second plot shows the Computational Ratio (CR) for the APR and APR$_t$ for the same test function.

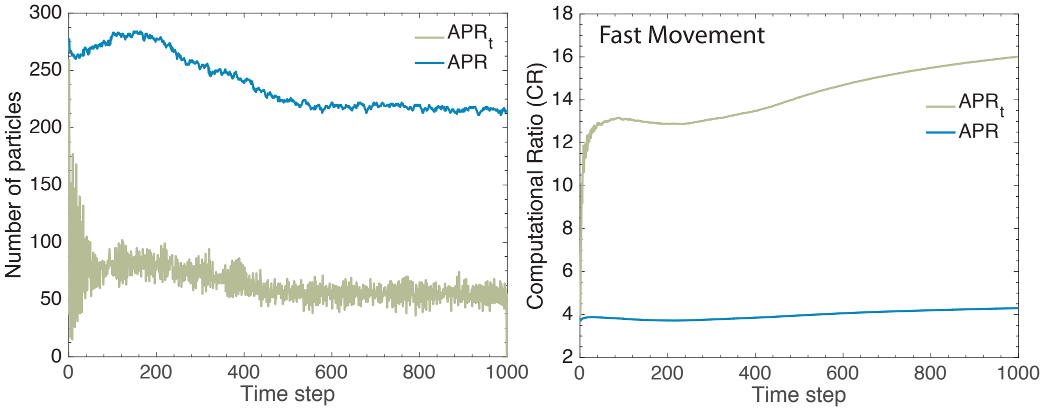


Figure 10.4: The first plot shows the number of particles in the classic APR for each time step (blue) and the equivalent size (# APR$_t$) of the APR$_t$ (green) for the time test function with **fast movement** ($u^* = 4$). The second plot shows the Computational Ratio (CR) for the APR and APR$_t$ for the same test function.

would be $CR = N_t$. That is, only the classic APR for the original time step would be required. We show the results of the one run of the no-movement benchmark in Figure 10.3. The first plot shows the size of the classic and time APR for each time step. We see that the classic APR is indeed constant, and the APR$_t$ is zero except at locations that become more sparse as time increases. These additional particles represent updates to the function intensity due to the causal Pulling Scheme. The second plot shows the CR for the classic and time APR. As expected, we find a constant CR for the APR of approximately three. In contrast, the CR for the APR$_t$ increases with the number of time steps, reaching a CR of 680 for 1000 time steps. However, it is not quite linear, or equal to N_t, due to the Particle Cells increasing in size through time. These extra particle cells in time, are analogous to the filler particle cells in space. Next, we provide some preliminary results on adaptation to varying temporal scales. We consider two cases of fast ($u^* = 4$) and slow ($u^* = 0.4$) movement. If the time APR is adapting correctly, we should expect the CR of the APR$_t$ to reflect this change. The results are shown in Figure 10.4 for the fast case, and Figure 10.5 for the slow case. We find that the APR$_t$ CR does reflect these changes in average speed, with CR being 3.72 times larger than the classic APR in the fast case and 14 in the slow case. Hence in the slow case, the reductions for the time APR are an order of magnitude greater than the classic case. Although, between the slow and fast speed example we find that an increase of approximately 3.8, less than half of the reduction of the maximum bound of ten. However, that is an upper bound of one random iteration and does not necessarily accurately reflect the results. Hence, more detailed analysis is warranted, however, we again leave this to future research.

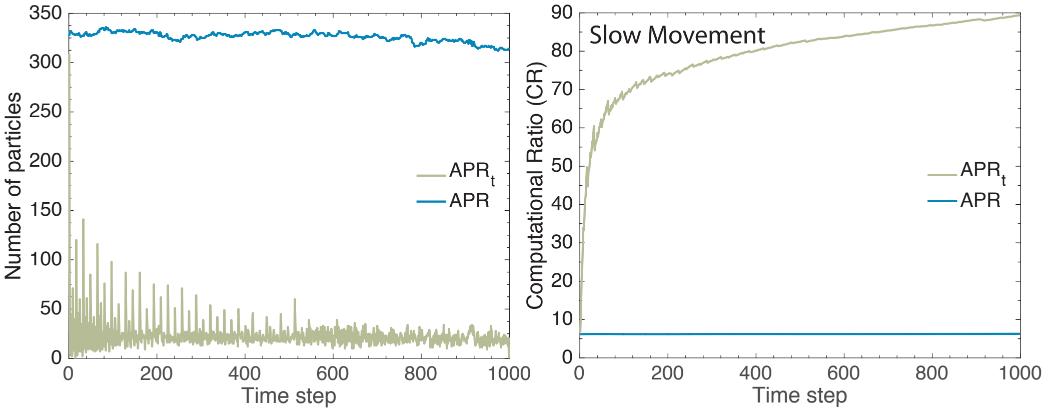


Figure 10.5: The first plot shows the number of particles in the classic APR for each time step (blue) and the equivalent size (# APR$_t$) of the APR$_t$ (green) for the time test function with **slow movement** ($u^* = 0.4$). The second plot shows the Computational Ratio (CR) for the APR and APR$_t$ for the same test function.

10.2.3 Relative size of update, add and remove

In the above, we have provided no insight into the relative proportions of the three components of the APR$_t$={ADD$_t$,REMOVE$_t$,UPDATE$_t$}. In general, across benchmark examples, the particle update set represents the majority of the size of the APR$_t$. We show this for a fast movement example in the second plot of Figure 10.7, where both the remove and add sets represent less than ten percent of the total required Particle Cells.

To provide the reader with some insight into how the add, remove and update events are distributed through space and time, we plot the spatial location of Particle Cells in the sets through time in Figure 10.6 and the first plot of Figure 10.7. We find that in all cases, the Particle Cells are located spatially around the peaks. The first plot of Figure 10.6 shows where Particle Cells are added and removed, with two trajectories in the center showing higher density that reflect two fast moving peaks. The same behavior is shown in the update set in the second plot of Figure 10.6 and first plot of Figure 10.7. Both plots show the same data, with the first plot in Figure 10.7 showing only a smaller subset of one hundred time steps to allow observation of the varying density of update particle cells for the different moving peaks through time.

10.2.4 Summary

Unfortunately, the above results, are not detailed enough to make conclusive statements. We note that a contributing factor to the sparse nature of the above analysis is the additional complexity and cost of dealing with the

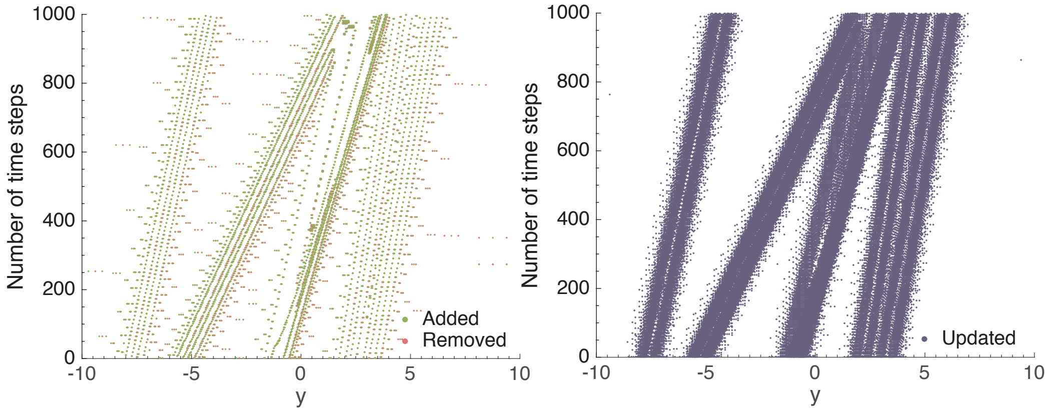


Figure 10.6: The first plot shows the location in space and time step of Particle Cells in the ADD$_t$ set (added to the APR at time point t) and REMOVE$_t$ set (removed from the APR) for a fast movement benchmark example. The second plot shows the location in space and time of Particle Cells in the UPDATE$_t$ (function values adaptively sampled to satisfy the time Reconstruction Condition). A smaller portion of 100 time-steps is shown for the update Particle Cells in the first plot of Figure 10.7

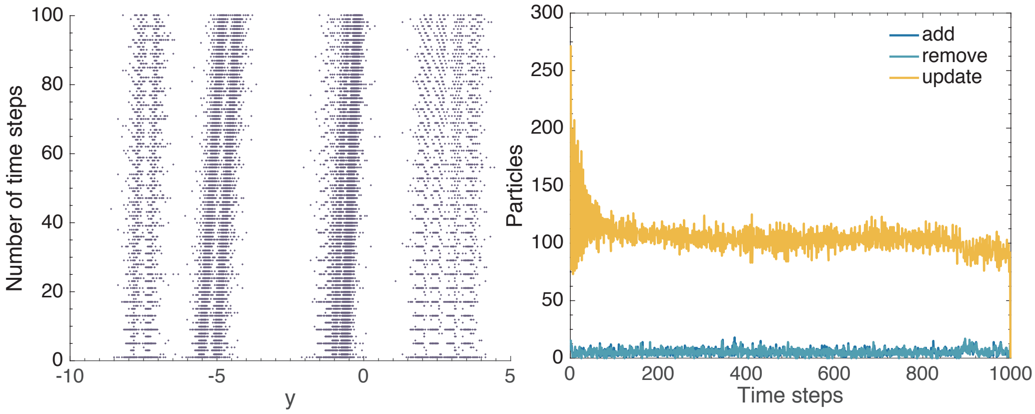


Figure 10.7: The first plot shows only 100 time-steps of the data presented in the second plot of Figure 10.6 for the location in space and time of Particle Cells in the UPDATE$_t$ set for a fast movement example. The second plot shows the relative size of the UPDATE$_t$, ADD$_t$ and REMOVE$_t$ sets for the example.

additional time variable does not allow direct application of the spatial benchmarking methods used previously. However, I believe the preliminary results above provide indications that such an APR_t scheme has the correct properties for **RC**5, allowing a means to include temporal adaptation, while not complicating the use of methods developed for the classic APR case.

10.3 Extension to LSFM data

The above scheme can be directly applied to arbitrary dimensions in space d. With the temporal adaptation steps showing little change. Preliminary work has replicated the noise-free results given above for the 3D LSFM data pipeline discussed in 6.2. However, the full extension will require the addressing of two still open issues described below.

10.3.1 Complication of noise

The first open issue is how to correctly deal with the impact of noise in the time adaptive environment. The introduction of noise results in oscillations of the classic APRs Implied Resolution Function. These oscillations are then captured by the temporal APR, hindering effective temporal adaptation. A possible solution would be the use of temporal information, or additional stability criteria, in the calculation of the spatial APR at each time step.

10.3.2 Real-time optimizations

The second open issue is related to temporal integration. In the classic APR, the filtering steps on the full image account for a significant portion of the computational cost of the classic APR per time step (See 6.5.4). Hence, it would be beneficial if the knowledge of the spatial distribution of previous time steps was utilized to speed up the estimation of the function gradient, Local Intensity Scale, and particle intensity estimation. One approach would be the calculation of the Local Resolution Estimate $L(\mathbf{y})$ only at particle locations from the previous time step, as is used in the benchmarks above and the numerical solution of partial differential equation benchmark (9.3.1). This would then serve to amortize the extra cost of temporal adaptation.

10.4 Summary and main points

In this chapter, we introduced how the APR can be extended to account for temporal scales and satisfy the last remaining representation criteria, **RC**5.

We summarize the main points of this chapter in the table below.

The treatment here is at a high level, and provides only a rough 'proof of principle'. However, I believe the presentation of these results is vital for the full evaluation of the APR and its application the LSFM data. If the above work can be successfully applied to real-time settings, the APR$_t$ would effectively allow acquiring of data that is independent of the original temporal and spatial sampling. Also, the adaptation of time could be utilized in processing steps, with great utility for performing joint segmentation and tracking tasks. Further, the anisotropic nature of the APR$_t$ allows a multiplicative factor from the time scales that are not present in the classic isotropic case. In the classic isotropic case, the local spatial resolution is effectively the minimum resolution in each spatial dimension. However, by making the temporal dimension independent, the APR$_t$ can simultaneously have high spatial resolution and low temporal resolution (and vice-versa). While still allowing for the 'simple' isotropic data structures and implementation of algorithms in space. Therefore further advancement of these ideas and extension to the LSFM case would appear to be vital to full utilization of adaptation with the APR.

Summary of the chapter

- Discussed desired properties of the extension of the APR to account for temporal scales.
- Derived the two Resolution Bound conditions for the Anisotropic formulation of the time adaptive APR$_t$ for the satisfaction of a time Reconstruction Condition, now with two error thresholds E and E_t.
- Provided a high-level description of the algorithmic steps required to form the APR$_t$.
- Showed preliminary results for the APR$_t$ in 1D using a test function of moving Gaussian peaks.
- Provided indications that the APR$_t$ successfully accounts for temporal time scales, while still allowing 'classic' APR processing per time step.
- Outlined challenges and future work required for the APR$_t$ to be extended for use in LSFM data.

11 Outlook and Discussion

Contents

11.1 Executive summary 239
  11.1.1 Chapter Result Summary 243
11.2 The APR for representation of LSFM data 246
  11.2.1 Critical evaluation of the Representation Criteria . . 247
  11.2.2 Conclusions and outlook 253
11.3 APR as a general purpose adaptive data representation 254
  11.3.1 Other image modalities 254
  11.3.2 Numerical solution of differential equations 254
  11.3.3 Statistical adaptive regression 255
  11.3.4 Surface approximation in computer graphics 255
11.4 Concluding remark 256

In this final chapter, we critically summarize, discuss, and reflect on the work presented in this thesis and its limitations, and speculate on future research directions and challenges.

The chapter is structured as follows. First, we provide an executive summary of the main conceptual and theoretical contributions of this work. Following, we provide a summary and description of the chapters of this thesis. Next, we critically assess the use of the Adaptive Particle Representation (APR) for studying spatiotemporal processes in biology (STB) using Lightsheet Fluorescence Microscopy Data (LSFM). We do this by evaluating the APR's fulfillment of the representation criteria and critically discuss limitations of the results and then highlighting directions of future work and challenges. We conclude this chapter and thesis, by discussing the potential uses of the APR as a general purpose adaptive data representation.

11.1 Executive summary

The main contribution of this thesis is the introduction of Adaptive Particle Representation (APR). The APR is an adaptive function representation that represents a function in a spatially adaptive way using a set of Particle Cells $\mathcal{V}$ and a set of function values stored at particle collocation points $\mathcal{P}^*$. The set of Particle Cells $\mathcal{V}$ fully defines both an implicit piecewise constant Implied Resolution Function $R^*(\mathbf{y})$ and the sampling locations of the particles.

The APR, as an adaptive function representation, can then be used to improve the storage, computational and memory costs of numerical processing tasks when compared to the equivalent task using homogeneously sampled data. The improvement can be achieved by reduction number of elements that must be computed over, reductions of the required memory for processing, and reduction in storage costs for both the input and result.

The APR represents a function by allowing reconstruction of function values at arbitrary locations $\mathbf{y}$ using any positive weighted combination of particles within $R^*(\mathbf{y})$ of $\mathbf{y}$. The combination of $\{\mathcal{V}, \mathcal{P}\}$ are selected such that the pointwise absolute error between the function and its reconstructed value is guaranteed to be below a user-defined fixed value $E\sigma(\mathbf{y})$ we call the Reconstruction Condition. Where $\sigma(\mathbf{y})$ is a special function known as the Local Intensity Scale, that can be used to adjust the allowed reconstruction error across the spatial domain. The sets $\{\mathcal{V}, \mathcal{P}^*\}$ are selected optimally, such that they represent the smallest number of Particle Cells $\#\mathcal{V}$, and also largest everywhere Implied Resolution Function $R^*(y)$, such that the APR satisfies the Resolution Bound. The Resolution Bound relates the derivatives of the function to an upper bound on the reconstruction error and guarantees the satisfaction of the Reconstruction Condition.

This optimal solution is found using Particle Cells, and a novel algorithmic approach we call the Pulling Scheme. The Pulling Scheme allows optimal solutions of problems defined by a Resolution Bound and requires knowledge, or computation of, function derivatives across the domain that are used to construct a Local Particle Cell set $\mathcal{L}$. This Particle Cells set $\mathcal{L}$ is then used to solve for the Optimal Valid Particle Cell set $\mathcal{V}$ used to form the APR. The Pulling Scheme has worst case linear scaling in N, where N represents the number of function values in a constant resolution sampling at the highest resolution. The Pulling Scheme utilizes the underlying geometry of problems defined by a Resolution Bound. The geometry arises from the Resolution Function $R(y)$ that is optimized being restricted by a bound over a spatial neighborhood that has its size set to the value of $R(y)$. The pulling scheme solves this optimization problem by constructing the largest function satisfying a bound out of regular blocks. Where only one block is allowed to define the function at each point in space and their widths are restricted to powers of two. The algorithm relies on many theoretical properties of solving the Resolution Bound using Particle Cells, which allow for simplification of the problem and more efficient computation of the result.

The particle sampling is chosen by placing one particle at the center of every particle cell in $\mathcal{V}$. Sampling this way is 'in some way' optimal, in the sense that it reflects the same integrated density of particles as achieved for a constant Resolution Function and homogenous sampling (as in a pixel image).

The APR is formulated and derived assuming noise-free knowledge of the function and its derivatives. However, in practice, the APR is likely to be formed from input data sampled at discrete locations that are possibly corrupted by noise. However, the Particle Cell and resolution function structures allow a few theoretical results for the impact of discrete sampling and errors in both the derivative and function values. First, regarding discrete sampling, the APR should be valid for all points in the domain, if the derivative function values at the discrete point represent an upper bound on the derivative between sampling points. Second, if the derivative input can only be estimated within a known relative error, this produces a bounded relative increase in the point wise reconstruction error to E^*. Lastly, if we assume that the Implied Resolution Function R^* satisfies the Resolution Bound, then for Gaussian corrupted noisy functions the APR has an MSE that converges at the optimal statistical rate $\frac{1}{N}$ to a biased estimate to the noise-free APR reconstruction. Where this biased estimate is within a constant factor of E, and convergence is in the sense of increasing the number of noisy input samples N. This can be compared to an optimal rate of $\frac{1}{N^{1/2}}$ for any fixed size kernel approximation method to an unbiased estimate.

The general form of the APR, and the use of the Particle Cells and the Pulling Scheme to find an optimal Implied Resolution Function, can also be used to create a wider class of representations. One extension is the forming of APRs that satisfy more general (α,m)-Reconstruction Conditions. Where α defines the function derivative to be bounded, and m, the required order of the reconstruction. These different conditions can be used together in arbitrary combinations. The isotropic restriction can also be dropped allowing the formation of APRs with anisotropic isotropic function neighborhoods. These ideas can be used for the construction of causal time (APR$_t$) adaptive APRs. Lastly, the Resolution Bound that is solved for does not have to be derived from a corresponding Reconstruction Condition and can have any source. Hence, the APR framework provides a 'toolbox' of representations that can be adapted to specific functions and computational problems.

The development, and use, of adaptive function representations to improve the efficiency of numerical processing tasks, is not novel. Similar ideas and adaptive representations are successfully used in a range of fields. Further, the main ideas and concepts used in the APR are closely related to those found in a variety of existing methods. However, the form of the APR, regarding its representation and solution, focusing on a Resolution Function and Resolution Bound, the use of Particle Cells, and the Pulling Scheme appears to be novel.

A second novel aspect of the APR is the incorporation of a Local Intensity Scale $\sigma(y)$. The Local Intensity Scale is included to allow the absolute function value of the reconstruction value to vary across the spatial domain. The feature was designed for the specific use case of the APR for LSFM data and was inspired by local gain controls in the human visual system. Critically, although we have shown above that spatial varying $\sigma(\mathbf{y})$ to be useful in practice, the Resolution Bound satisfying the Reconstruction Condition is only guaranteed to hold for $\sigma(\mathbf{y})$ that satisfy strict conditions that depend on the solution they are used to compute. Unfortunately, we have no methods to generate $\sigma(\mathbf{y})$ that guarantee this condition beyond constant $\sigma(\mathbf{y}) = \sigma_c$. However, such a constant Local Intensity Scale negates its original purpose.

Although the APR can be considered optimal some features, in others, it is sub-optimal. First, the APR is by construction sub-optimal concerning minimization of the number of coefficients to represent it $(\#\{\mathcal{V}, \mathcal{P}^*\})$ for a given function error. The sub-optimality is a result from the fact that the APR can be losslessly represented by an associated Haar wavelet representation with $\mathcal{P}_h$ where $\#\mathcal{P}_h \geq \#\mathcal{P}^*$. Second, the Implied Resolution Function is a suboptimal solution to the Resolution Bound. Sub-optimal in that there exists a continuous resolution function $R_c(\mathbf{y})$ such that $R_b(\mathbf{y}) \leq R^*(\mathbf{y})$ that also satisfies the Resolution Bound. However, the maximum ratio between R_c and R^* is bounded by $4\sqrt{d}$ where d is the dimension. However, R^* is the largest everywhere (optimal) of all potential Implied Resolution Functions. Also, there is no guaranteed tightness on the difference between the Resolution Bound and the Reconstruction Condition. Last, the reconstruction neighborhood as defined by the Implied Resolution Function is isotropic. The isotropic neighborhood restricts the adaptation and hence provides sub-optimal spatial adaptation when compared to a fully anisotropic representation.

However, as a trade-off for sub-optimality in these properties, the APR has, a predictable and regular structure locally, an explicit local spatial scale through the Resolution Function, and can be used to provide bounded reconstructions with a wide range of reconstruction methods. Further, the restrictions leading to sub-optimality play vital roles for the efficient calculation of the APR using Particle Cells and the Pulling Scheme and the construction of efficient data structures for computation and storage. Also, this formulation is key to the ability of the APR to be extended to a 'toolbox' of higher order models with little modification.

For any given numerical processing problem and data set type, there is a large variety of adaptive representations that could be used to provide an improvement in computational and memory performance through adaptation to spatial and temporal scales. Each of these adaptive representations comes with its unique benefits and trade-offs. Therefore, selection of a particular representation comes down to an evaluation of the given problems computational demands and scope and how they align with different representations. Evidence of such a trade off is reflected in the wide range of adaptive representations that are currently used for different problems and fields of research. Similarly, the APR appears to have its unique benefits and tradeoffs as discussed above and throughout this thesis. Hence, its suitability over existing methods is dependent on the task and data set. In any case, the use of the APR will depend on the development of high-quality software libraries and implementations of the ideas in algorithms in this thesis that can utilize the range of hardware acceleration techniques through GPU, shared, and distributed memory based parallelism.

In this thesis, we focused on the evaluation of the APR to be used for a specific task. The replacement for pixel images for the study of STB using LSFM data. We evaluate this shortly in a dedicated section using the representation criteria.

11.1.1 Chapter Result Summary

First, we provide a summary of the content and results on a per chapter basis. At the end of each chapter, we provided a more detailed summary and a bullet point table summarizing the key results, and we direct the reader to these for more detail. However, the below can be used for reference as to where the results discussed in the above summary were presented.

In the first Chapter 2, we began by describing how developments in LSFM hold the promise of being able to study previously unobservable processes in biology through space and time in high resolution. We then introduced the key steps involved in this process and highlighted how processing and handling of the raw data creates a significant bottleneck progress in the field. We argued that this problem comes from the data exhibiting varying and dynamic spatial, temporal and intensity scales that result in significant inefficiencies when using a homogeneously sampled data representation. We then surveyed current approaches to processing and reflected on the key concepts of the human visual system. From this, we concluded that a potential solution to the bottleneck would be the use of an adaptive data representation, as an alternative to pixel images, for all steps in pipelines, from acquisition to analysis. The adaptive data representation would alleviate the bottleneck by accounting for the dynamic spatial, temporal, and intensity scales in the data and remove the inefficiencies generated through homogeneous sampling.

Next, in Chapter 3, we discussed and presented five criteria that an adaptive data representation should be able to fulfill to meet the goal set out in the previous chapter. We called these the *represenation criteria*. Following this, we reviewed and reflected on existing adaptive data representations. From the review, we concluded that although almost all the representation criteria could be fulfilled by existing ideas, there were two main areas that still required development. First, the inclusion of a local gain control like mechanism that allowed for the adaptation to dynamic intensity scales of LSFM data. Second, the demand that the representation 'as close to pixels as possible' to allow ease of re-development and implementation of existing methods.

In Chapter 4, we then introduced the key concepts and ideas of the APR. This chapter focused on presenting the ideas in 1D and taking a didactic approach to the introduction of new concepts. General dimensional treatment and technical details and proofs were omitted and left to the following chapter. The chapter introduced the key ideas of the Reconstruction Condition, and its link to the Resolution Bound, the Implied Resolution Function and its construction from Particle Cells, the Local Intensity Scale, and the key ideas that form the Pulling Scheme that represents the 'Transform' algorithm for the APR. Last, in the chapter, we discussed practical aspects of the use of the APR including the impact of discrete sampling and noise, again using 1D for simplicity of exposition.

Complementing the previous 1D introductory chapter, in the following Chapter 5, we presented the technical details, proofs, and extensions of the ideas of the APR to the general dimension case. Given the concepts had already been introduced, in this chapter and attempt was made to present the ideas in a general way, that could allow the possible extension to more complicated models in the future. The core of the chapter is dedicated to proving the results and properties of Particle Cells that form the basis of the Pulling Scheme that is used to generate the APR. Following this, we also provide details of how to form the APR particle graph and a description of function reconstruction methods used in the following chapters.

Next, in Chapter 6, we explored the properties of the APR and validated the theoretical results using empirical benchmarks. In this chapter, we provided empirical evidence for the evaluation of the first three representation criteria. First, we validated the APR in 1D using functions of a known form with a constant Local Intensity Scale. Following this, we introduced the 3D implementation of the APR for LSFM data and a method for generation of synthetic images. This implementation included a spatially varying Local Intensity Scale based on a slowly varying estimate of the local scale in the image. Using the synthetic images, we explored the APRs properties regarding an increase in information content, sampling, and noise. Following this, we introduced two benchmark datasets, the Computational Ratio and Exemplar benchmark data. These were used to represent a different range of image size and content levels. We then introduced the APR data structures we use for processing in the next chapter. Following this, we used the two benchmark datasets to evaluate the execution time of the different steps of forming the APR, the memory cost of its transform, and the size of the APR for file storage using simple lossless compression. Lastly, we reflected on the results, discussed limitations and the fulfillment of the first three representation criteria (**RC**1-3).

In Chapter 7, we evaluated a variety of simple processing tasks on the APR. These results were presented as empirical evidence for the fulfillment of the fourth representation criteria. We began by discussing how the APR can be 'interpreted', or used for processing, and how this relates to similar concepts for processing on pixels. We then introduced performance metrics that we used to evaluate the computational and memory performance of the APR and compare it to pixel image representations. Using these metrics, we then evaluated the relative performance on the APR on four simple processing tasks, linear neighbor access, random neighbor access, separable pixel filtering, and segmentation. Comparing their performance regarding execution time and memory cost. Then we demonstrated a range of methods for visualization using the APR and showed how the APR can be used to create novel algorithms using the example of adaptive APR filters. Lastly, we reflected on these results regarding the APRs fulfillment of the fourth representation criteria (**RC**4).

Next, in Chapter 8, we discussed the similarity between the ideas and concepts of the APR and existing methods in the literature. We did this on a 'by concept' basis and discussed a conceptual link between the APR and wavelet thresholding. Next, we discussed two optimality properties of wavelet thresholding, first regarding optimal error convergence regarding the size of the representation and then for optimal representation of noisy signals regarding the MSE. For both, we compared benchmark results with those presented in the original wavelet thresholding papers. Motivated by the optimal noisy representation results, we then derived results on the convergence of the MSE of the APR for noisy signals.

In Chapter 9, we presented extensions of the classic APR in space. First, we introduced how the APR can be used satisfy a generalization of the Reconstruction Condition, to more general (α,m)-Reconstruction Conditions. Where α represents the function derivative being bounded, and m the order of the reconstruction method required. We then showed that these (α,m)-Reconstruction Conditions can be arbitrarily combined to form a single Resolution Bound that can be solved by the Pulling Scheme. Next, we briefly validated and explored some of these extended APRs. Then, we provided a potential use case of these general APRs by showing preliminary results for the solving of the 1D viscous Burgers equation. Lastly, we showed that the APR can be used without a Reconstruction Condition, showing the APR used for particle generation for a complex geometry.

In the last content Chapter 10 in this thesis, we showed preliminary results on how the APR can be extended to adapt to temporal time scales and address the fifth representation criteria (**RC**5). First, we discussed the desired properties of the temporal adaptation. Next, we introduced the time Reconstruction Condition and derived its two separate Resolution Bounds. Then we provided a high-level overview of how the time extended APR, APR_t, can be formed using a combination of the classic APR and a 1D causal Pulling Scheme. Following, we then validated the model in 1D, showing preliminary results that it satisfies the APR_t has the desired properties for noise-free functions. Lastly, we discussed the challenges that still need to be addressed for application to LSFM data and fulfillment of the fifth representation criteria.

11.2 The APR for representation of LSFM data

In this section, we critically evaluate the APRs ability to be used as a replacement for pixels in pipelines for studying spatiotemporal processes in biology using LSFM data as motivated in Chapter 2. Where, by replace, we mean that instead of the original pixel image, the APR is stored, and used for all processing tasks. By storing and then processing solely on the APR, a whole processing pipeline can utilize the computational and memory benefits of an adaptive data representation. It is proposed that if the adaptive data representation can effectively capture the dynamic spatial, temporal, and intensity scales the removal of inefficiencies from pixel sampling will alleviate current processing bottlenecks.

To aid in the evaluation of an adaptive data representation to be used for this purpose, in Chapter 3 we proposed five *representation criteria* (**RC**) an adaptive representation should fulfill. The objective of the criteria was to provide a framework to evaluate if an adaptive representation could be used as a wholesale replacement of pixels. Hence, these criteria are not designed as to evaluate finding an optimal solution, merely if a representation would be sufficient. However, we note critically that the representation criteria are heavily influenced by concepts and the design of the APR. As a result, an adaptive representation could objectively fail to fulfill these criteria, while being a valid solution to the LSFM data bottleneck. Thus, our evaluation of existing representations in necessarily 3.3 is biased towards the concepts and the ideas used by the APR. Nonetheless, I believe the representation criteria provide a valuable tool for evaluating the question "could the APR be used as a replacement for pixel images in processing pipelines?"

11.2.1 Critical evaluation of the Representation Criteria

Next, we address each representation criteria separately, providing a summary of evidence for its fulfillment, and then providing critical comment on limitations and potential future work for each. We direct the reader to Chapter 6, and 6.6.2, for more details on the results for **RC**1-3, Chapter 7 and 7.6.1 for **RC**4, and Chapter 10, for preliminary results for **RC**5.

RC 1

- The size of the APR $\#\{\mathcal{V}, \mathcal{P}^*\}$ must be proportional to the information content of an image, accounting for varying spatial scales, and not scale with the number of pixels.

In Chapter 6, we provided evidence that the size of the APR scales with information content and not the number of original pixels N. The benchmarks relied on using the number of objects in the image as a proxy for information content. Benchmarks showed that the APR would reach a maximum size for a given number of objects, and hence become insensitive to the number of pixels in the original image N, or the sampling resolution used. This adaptation was robust even though the individual objects varied randomly in their local brightness by an order of magnitude. This was done through the use of a Local Intensity Scale $\sigma(\mathbf{y})$ that provided a smooth estimate of the local range in the image.

However, these results come with many caveats. First, in the presence of noise, the adaptation becomes sensitive to parameters used. First, regarding the estimation of the gradient and setting of the B-spline smoothing parameter λ. If this is set too small, the noise will increase the resolution across the domain limiting adaptation. Too high, and the adaptation could 'miss' certain features, and provides less spatial localization. Second, the Local Intensity Scale requires the setting of a lower bound through σ_{th} to provide a minimum intensity scale. In its absence, in flat regions $\sigma(\mathbf{y}) \to 0$, and the full capturing of background noise. Further, although we heuristically motivated the Local Intensity Scale, we did not provide any rigorous justification, nor effectively benchmark its performance in isolation. Third, for real-datasets, images contain background signal b (See 2.2.4), that is not needed to be resolved. To selectively remove this information requires either addition of extra parameters, such as the intensity threshold we used, or changes in the Local Intensity Scale. Last, LSFM data also often show additional complications that were accounted for in the benchmarks. These include anisotropic spatially varying PSFs, anisotropic sampling, and spatially varying non-Poisson noise-processes.

Another criticism arises for the ability of the APR to 'account for varying spatial scales'. We provided evidence that the APR can account for spatial variation and this is explicitly represented in the Implied Resolution Function. However, this adaptation is isotropic. The isotropic restriction, only allows the APR to adapt to the local minimum spatial scale in each direction, and not fully to the scales of the data in each direction.

RC 2

- The APR must guarantee a user-controllable representation accuracy E for noise-free images, relative to a local intensity scale σ, and not reduce the signal-to-noise ratio of noisy images.

In Chapter 6, we showed that for noise free functions the APR satisfies the Reconstruction Condition for a given E for all allowed reconstruction methods. The result was validated both in 1D with a constant σ and in 3D with a spatial varying σ. Importantly, the bound still empirically held, despite no guarantee of the satisfaction of the smoothness criteria for σ (4.3.5). For noisy images, we found that satisfaction of the Reconstruction Condition was not possible for small E, as impacts of the noise and bias in the reconstruction dominated as $E \to 0$. However, regarding PSNR of the reconstruction APR image, we found a range of E for which the image quality was maximized ($\approx 0.08 - .15$). Also, in this range of E and lower, the MSE of the reconstruction was less than half the noise level in the original image. Further, the adaptation of R^* was robust to noise, as the Reconstruction Condition was again satisfied when noisy particle values were replaced by noise free values (Although, likely over-sampling).

Critically, although we have shown that a lack of guarantee on σ, did not impact the adaptation, we did not evaluate critically whether this was in any way a relevant Local Intensity Scale to normalize the error. Further, we did not assess the negative impact that the choice of Local Intensity Scale had regarding an increase in particles, beyond what would be optimally required for more optimal choice of σ.

Another strong criticism can be directed at the representation criteria itself. The form is tailored identically almost trivially to the form of the results.

First, regarding the choice of a pointwise, or infinity norm, for the Reconstruction Condition. This could have been replaced by a global norm, such as a weighted L_1 or L_2 norm. However, the infinity norm does have the comfort that in the 'eyes' of the algorithm, all data points are 'treated equal'. This is not necessarily the case for a set value E with a global norm. An immediate criticism of a set bound E, with any norm, is that in the presence of noise, in general, can not be guaranteed to be satisfied due to the loss of information. Instead, as shown by the representation criteria and the results, one falls on concepts of MSE or the equivalent PSNR of the reconstruction. This then raises the question "Why not use an MSE minimization criteria for adaptation?". This is a valid criticism, however, in 8.3 of Chapter 8 we showed that under the assumption of satisfaction of the Reconstruction Condition, the representation would converge at the optimal statistical rate to an APR reconstruction within a constant factor of that of the optimal E noise-free APR. Hence, the satisfaction of the Reconstruction Condition does have benefits in a noisy context. However, these results rely on the robustness of the Implied Resolution Function, which must itself be estimated from the noisy data.

Therefore, again we have shown promising results, however additional work is still be warranted. First, further development and investigation of the smoothness assumption on the Local Intensity Scale, and theoretical work on guarantees on how its violation impacts the satisfaction of the Resolution Bound and hence Reconstruction Condition. Second, the further development of the results regarding the performance of the APR under noisy situations, including the direct influence of a given noise level through to the Implied Resolution Function, and then its impact on the bias and optimal convergence rate of the APR.

RC 3

- It must be possible to rapidly convert a given pixel image to the APR with a computational cost at most proportional to the number of pixels.

Using the CR benchmark and exemplar data, In 6.5.4 of Chapter 6, we showed that for our given implementation, the APR could be formed with a linear computational and memory complexity in the number of original pixels N. This could be broadly broken into two parts, the calculation of the Local Resolution Estimate $L(\mathbf{y})$ and then the calculation of the APR using the Pulling Scheme. The first part is inherently linear in both memory and computational complexity, being computed using filtering operations. Further, this cost can be fixed independently of parameters by the use of recursive implementations. The Pulling Scheme also has worst-case linear scaling in N. However, the actual performance is dependent on the information content of the image. The computational and memory cost is dominated by the first part. Regarding memory cost, the simultaneous storage of the original image, the gradient magnitude, and the down-sampled Local Intensity Scale represents the largest overhead. Given the computational cost was dominated by simple filter operations, that would often be classed as 'rapid', we concluded that the overall formation of the APR is also rapid. Further, this cost is small enough to fall in time ranges that can be classified as 'real-time' in certain contexts [5, 110].

However, critically, although the algorithm is not objectively 'slow' ideally both the computational and memory cost would be further reduced. Here, we used a pipeline accelerated using shared memory processing (albeit with some slight issues). However, ideally, the pipeline would be able to make use of all available acceleration hardware. This includes both the use of GPU's and Xeon-Phi type accelerators. Though, these do place additional memory restrictions on computation. Indeed, benchmarking (not-shown) have indicated that the computational cost can be efficiently reduced for the pixel filtering steps by both these methods. Further, for expansion to larger problems, it would also be useful to have a distributed memory implementation. However, for a distributed implementation special treatment would need to be taken regarding the recursive steps in the filtering algorithms for calculation of the Local Resolution Estimate. Although, this does not seem prohibitive. Further, regarding the Pulling Scheme, this will require correct handling of data structures and ghost layers, but as a whole, the algorithm appears to be amenable to distributed extension.

Ideally, the dependence on N of both the memory and computational cost would be reduced when computing more than one APR in a time varying data set; we discuss this further below with **RC**5.

RC 4

- The APR must reduce the computational cost of image-processing tasks without resorting to the original pixel representation.

In Chapter 7, we considered, and provided benchmark results for, the evaluation of the processing performance of the APR. As discussed in 7.6.1, this appears an 'ill-posed' and difficult task to assess given the wide range of types of image processing tasks, algorithms, and implementations. In 7.6.1 we provided a more detailed discussion of those issues to which we direct the reader.

In summary, we argued that the APR has many favorable properties regarding **RC**4. First, the APR allows comparable interpretations and operations to those that form the basis of many pixel algorithms. I argue that this property should reduce complexity and implementation costs for adapting algorithms from pixels to the APR. Second, the APR, except linear neighbor access, showed improvements in computational costs compared the equivalent algorithm on pixels. However, the exact degree of this speed up depends both on the reduction in a number of particles to pixels (CR), and the given task and implementation. More consistently, the APR provided memory cost reductions across all tasks that were proportional to the CR. Interestingly, in the case of graph-cuts based segmentation, this allowed a previously infeasible problem due to memory constraints on pixels, feasible on the APR. Lastly, with examples in visualization, segmentation, and adaptive APR filters, we showed that the APR can allow for easy development of novel algorithms, that would otherwise be costly, or 'complicated', using a pixel representation.

However, the results also highlighted the trade-off that comes with the use of an adaptive data representation such as the APR. First, adaptive representations require more sophisticated data structures and usually less efficient memory access patterns for simple tasks than those allowed for a pixel data representation. This was reflected in the APR's poor relative computational performance in the linear neighbor access benchmark. Second, care must be taken on the suitability of the implementation of an algorithm on the APR. This point was illustrated with the separable filtering algorithm on the APR. This algorithm showed good computational and memory performance, however, for particular filtering tasks, such as large blur operations, qualitatively poor results. However, there may often be a more natural APR equivalent, such as adaptive APR filters, in the filtering case, that show better performance. However, even in this case, if the spatial length scales of a result or temporary results differ significantly from the spatial scale of the APR, either a pixel representation or a new APR sampling, will be required.

The one key area of processing where it is possible that the APR would be inappropriate may be image enhancement tasks. These lower level tasks, are more likely to rely on the exact distribution of the pixels, and hence, given the APR is lossy, the APR equivalents could be sub-optimal. Examples of tasks where this could be the case are super-resolution, deconvolution or de-noising. A simple solution to this would be simply to do any pixel sensitive tasks before the formation of the APR. The second would be to use the APR 'in concert' with the original image. Hence, providing, APR enhanced, versions of existing algorithms.

Another criticism for any adaptive representation including the APR is that many algorithms and implementations cannot be directly used. Requiring additional development and implementation time and costs. This cost is further exacerbated by the often more complicated data structures and implementations for the APR when compared to a pixel based algorithm. One possible means to alleviate this would be the development of an efficient library that hides the complexity of the APR, presenting an interface that is as similar to pixels as possible. Hence providing building blocks for rapid and efficient development on the APR (in the spirit of libraries such as PPM Library Sbalzarini et al. [107] for distributed parallel simulations). Further, the APR transform, and the processing implementations would also be ideally accessible through popular image processing software such as Fiji [109].

Lastly, we did not address the use of the APR for Deep Learning [54] based approaches to processing. Given the current success [54, 138], and the rise of deep learning methods, it would seem vital for the APR also to be able to utilize these approaches. One possible avenue would be the utilization of existing graph based methods such as Defferrard et al. [39]. However, ideally, efficient deep learning algorithms could be developed directly for the APR, allowing for the utilization of the additional spatial information that it provides.

RC 5

- The APR must also be able to similarly account for varying temporal scales.

Last, in Chapter 10, we provided details of the time extension of the APR, to allow for adaptation to varying temporal scales. We provided a specific extension to the classic APR we called APR_t, which introduced an additional temporal relative error E_t in addition to the spatial relative error E. The APR_t allows adaptation to temporal scales, by adaptively sampling in time, as well as space. The adaptation is done anisotropically, such that the classic spatial APR at any time step can be reconstructed with a known bounded reconstruction error. Unfortunately, we only provided preliminary results in 1D. However, these limited results were positive indicating that the APR_t can keep the benefits of the spatial APR, while also adapting to temporal time scales. Note, the lack of more detailed analysis and extension to 3D is a reflection of the infancy of this work, rather than reflecting challenges to its development.

The successful application of the ideas of the APR_t to LSFM data, would, in my opinion, present a significant increase in benefit in the use of the APR, and provide it a unique advantage over existing approaches. Although the reductions from the APR in a single time step can result in significant reductions in both memory and computational cost for individual time steps other approaches still seem feasible. However, given the anisotropic nature of the time adaptation, any reductions in time are effectively multiplicative. Hence, if factors in time, even in the conservative range of $5-10$ could be realized this then results of a total CR for the APR_t in the range of $25-1000$, rather than $5-100$ in the classic APR case. These levels of CRs could result in data sets that in their raw form being in the TB range, being reduced to less than a GB. Further, the temporal adaptive information could be utilized to provide novel tracking and segmentation algorithms.

Further, the APR_t could be utilized to reduce the memory and computational cost of forming the APR at each time step. This could be done by focusing the calculation of the Local Resolution Estimate $L(\mathbf{y})$ on high-resolution areas from the previous time step. Thus, allowing reduction memory and computational cost for calculation of the gradient magnitude and Local Intensity Scale. Further, the temporal context allows for the development of Local Intensity Scales that can utilize the estimate of the local spatial resolution from a previous time step, potentially allowing methods that could guarantee the smoothness assumption.

However, development is still needed for the application of the APR, particularly exploring how to increase the robustness of the time adaptation to noise, and development of efficient algorithms utilizing ideas discussed in the previous paragraph.

11.2.2 Conclusions and outlook

From the above, I conclude that the APR has sufficient properties to be used, in possibly limited use cases, for the replacement of pixels pipelines for the studying of STB in LSFM data. Above, I have highlighted areas of additional development and extension to the theoretical basis of the APR. However, given the existing work, the more significant next steps would appear to be implementation and use of the APR for real pipelines and problems. Such applications would still require significant development. However, I believe the work in this thesis forms an adequate basis from which successfully APR based pipelines could be designed and built.

Stepping back, it is possible that the current bottleneck in LSFM data processing will, in essence, 'solve itself'. It is possible that new technological developments could increase memory capacity and computational power sufficiently such that the bottleneck in processing no-longer exists. This could result in the lack of need for an adaptive representation such as the APR. Another approach that could render the adaptive representations irrelevant could be the advancement of deep learning Goodfellow et al. [54]. Deep learning, or other machine learning approaches, could advance to a level such that efficient representations and all processing tasks could be effectively learned. This could provide tailored representations and processing tasks that provide a complete solution to processing pipelines and remove existing bottlenecks. The author's hope would be that the APR could form a part of accelerating progress in this direction, by using the APR and possibly learned variants as the input representation, rather than pixels, to the neural networks.

11.3 APR as a general purpose adaptive data representation

The potential applications of the APR are not limited to LSFM data. Although the APR was motivated, and benchmarked, for acceleration of processing tasks in LSFM data, the concepts, ideas, and algorithms of the APR, including Particle Cells, the Implied Resolution Function, and the Pulling Scheme are more generally applicable. In this last section, we will briefly discuss other research areas where the APR could be utilized.

11.3.1 Other image modalities

An area that could allow almost direct application of the implementation of the APR used here would be its use with other image modalities. For example, it could be for adaptive representation of natural images, as briefly benchmarked in Chapter 8. However, as shown in benchmarks and discussion, regarding the reduction of coefficients the APR is inferior to wavelet approaches. Although, the APR could still be used for processing tasks or the integration of a local adaptive representation error through the Local Intensity Scale. Further, for compression exploration of anisotropic APR extensions, or use integration of ideas such as the Easy Path Wavelet Transform on the APR may be interesting areas of research. Another related application of the APR could be the exploration of the time extended APR_t to video compression or processing tasks.

Indeed, any contrast technique with 'pixel' like data could also be used with the APR. For example, a promising area could include application in medical image processing such as processing of Magnetic Resonance Images (MRI). However, it would have to be assessed for any benefits above wavelets based techniques that are already in use [128].

11.3.2 Numerical solution of differential equations

As reflected in the review of adaptive methods in 3.3.3 of Chapter 3, a range of adaptive data representations are used for the numerical solution of differential equations. Therefore, it would be of interest whether or not the unique features and trade-offs for the APR can provide a new approach to complement the existing work. Indeed, in 9.3.1, we illustrated using a simple test case of the solution of the 1D viscous Burgers equation, that the use of the generalized (α,m) APR can be used for the spatially adaptive numerical solution of a partial differential equation. We note, that the solutions in 9.3.1, appear to provide similar adaptation as in sparse wavelet collocation methods such as Schneider and Vasilyev [114, 114], Vasilyev and Bowman [131], as shown for a 1D example in Figure 3.5. This similarity raises the question of there being any additional utility in the use of the APR as an alternative. Possible benefits could be the tool-box of (α,m) conditions, simple data structures and similarity to classical collocation methods, or the memory and computational efficiency of the Pulling Scheme. We leave such evaluations to future research.

Again, for application of the APR for use in state of the art problems would require a distributed memory version of the Pulling Scheme that does not rely on the explicit storage of the Particle Cell tree. A possible avenue would be the integration and use in distributed memory software libraries such as Sbalzarini et al. [107].

Lastly, rather than providing adaptation through time, the APR could be used simply as a method of particle generation for complex geometries as shown in Figure 9.7. Although, due to the simple nature of this approach I am not clear if this is in any way novel.

11.3.3 Statistical adaptive regression

As discussed in 8.3, the 'oracle' based representations [22, 52, 24], used for adaptive regression models have a similar form to the APR. Therefore, it would interest to explore whether the APR, or extensions, could be useful in this field. It would also be interesting to compare the statistical convergence properties of the APR presented in 8.3 with these existing methods.

11.3.4 Surface approximation in computer graphics

DeVore et al. [43] and Agarwal and Suri [6] both developed adaptive representations for surface reconstruction for computer graphics satisfying infinity norm equivalent to the Reconstruction Condition. Given the similarity, it would be again interesting if the APR could provide benefit for surface approximation in computer graphics.

11.4 Concluding remark

This concludes the work in this thesis. The appendix follows after the references providing additional technical details that supplement the main text. All code and raw data are available from the author on request, and all code is expected to be made available in the future under an open source license.

Bibliography

[1] Repository of static 3d-meshes. URL http://193.48.251.101:8080/3dsegbenchmark/dataset.html.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[3] Michael D Adams. A flexible content-adaptive mesh-generation strategy for image representation. *IEEE Transactions on Image Processing*, 20 (9):2414–2427, 2011.

[4] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[5] Yaser Afshar and Ivo F Sbalzarini. A parallel distributed-memory particle method enables acquisition-rate segmentation of large fluorescence microscopy images. *PloS one*, 11(4):e0152528, 2016.

[6] Pankaj K Agarwal and Subhash Suri. Surface approximation and geometric partitions. *SIAM Journal on Computing*, 27(4):1016–1035, 1998.

[7] F Alted. Blosc, an extremely fast, multi-threaded, meta-compressor library, 2017. URL https://github.com/Blosc/hdf5-blosc.

[8] Fernando Amat and Philipp J Keller. Towards comprehensive cell lineage reconstructions in complex organisms using light-sheet microscopy. *Development, growth & differentiation*, 55(4):563–578, 2013.

[9] Fernando Amat, Eugene W Myers, and Philipp J Keller. Fast and robust optical flow for time-lapse microscopy using super-voxels. *Bioinformatics*, 29(3):373–380, 2012.

[10] Fernando Amat, William Lemon, Daniel P Mossing, Katie McDole, Yinan Wan, Kristin Branson, Eugene W Myers, and Philipp J Keller. Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data. *Nature methods*, 2014.

[11] Fernando Amat, Burkhard Höckendorf, Yinan Wan, William C Lemon, Katie McDole, and Philipp J Keller. Efficient processing and analysis of large-scale light-sheet microscopy data. *Nature protocols*, 10(11):1679–1696, 2015.

[12] ArrayFire. Arrayfire library, 2015. URL `http://arrayfire.com/`.

[13] Omar Awile, Ferit Büyükkeçeci, Sylvain Reboux, and Ivo F Sbalzarini. Fast neighbor lists for adaptive-resolution particle simulations. *Computer Physics Communications*, 183(5):1073–1081, 2012.

[14] Utkarsh Ayachit. The paraview guide: a parallel visualization application. 2015.

[15] Ivo Babuška and BQ Guo. The h, p and hp version of the finite element method; basis theory and applications. *Advances in Engineering Software*, 15(3-4):159–174, 1992.

[16] Ivo Babuska and Jens M Melenk. The partition of unity finite element method. Technical report, DTIC Document, 1995.

[17] Ivo Babuška, RB Kellogg, and J Pitkäranta. Direct and inverse error estimates for finite elements with mesh refinements. *Numerische Mathematik*, 33(4):447–471, 1979.

[18] Marsha J Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[19] S Bertoluzza and G Naldi. A wavelet collocation method for the numerical solution of partial differential equations. *Applied and Computational Harmonic Analysis*, 3(1):1–9, 1996.

[20] OpenMP Architecture Review Board, 2013. URL `http://www.openmp.org/wp-content/uploads/OpenMP4.0.0.pdf`.

[21] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.

[22] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[23] Naama Brenner, William Bialek, and Rob de Ruyter Van Steveninck. Adaptive rescaling maximizes information transmission. *Neuron*, 26(3): 695–702, 2000.

[24] Michael Brockmann, Theo Gasser, and Eva Herrmann. Locally adaptive bandwidth choice for kernel regression estimators. *Journal of the American Statistical Association*, 88(424):1302–1309, 1993.

[25] Hermann G Burchard. Splines (with optimal knots) are better. *Applicable Analysis*, 3(4):309–319, 1974.

[26] Peter Burt and Edward Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

[27] Lianyu Cao, Penghui Juan, and Yinghua Zhang. Real-time deconvolution with gpu and spark for big imaging data analysis. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 240–250. Springer, 2015.

[28] Bevan L Cheeseman, Dongcheng Zhang, Benjamin J Binder, Donald F Newgreen, and Kerry A Landman. Cell lineage tracing in the developing enteric nervous system: superstars revealed by experiment and simulation. *Journal of The Royal Society Interface*, 11(93):20130815, 2014.

[29] Bi-Chang Chen, Wesley R Legant, Kai Wang, Lin Shao, Daniel E Milkie, Michael W Davidson, Chris Janetopoulos, Xufeng S Wu, John A Hammer, Zhe Liu, et al. Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution. *Science*, 346(6208):1257998, 2014.

[30] Ole Christensen. *Functions, spaces, and expansions: mathematical tools in physics and engineering*. Springer Science & Business Media, 2010.

[31] Charilaos Christopoulos, Athanassios Skodras, and Touradj Ebrahimi. The jpeg2000 still image coding system: an overview. *IEEE transactions on consumer electronics*, 46(4):1103–1127, 2000.

[32] Roger N. Clark. Notes on the resolution and other details of the human eye, 2016. URL `http://www.clarkvision.com/articles/eye-resolution.html`.

[33] Franklin C Crow. Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics*, 18(3):207–212, 1984.

[34] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

[35] Geoff Davis, Stephane Mallat, and Marco Avellaneda. Adaptive greedy approximations. *Constructive approximation*, 13(1):57–98, 1997.

[36] James Davis, Yi-Hsuan Hsieh, and Hung-Chi Lee. Humans perceive flicker artifacts at 500 hz. *Scientific reports*, 5, 2015.

[37] Carl de Boor. Good approximation by splines with variable knots. In *Spline functions and approximation theory*, pages 57–72. Springer, 1973.

[38] Carl De Boor. Good approximation by splines with variable knots. ii. In *Conference on the numerical solution of differential equations*, pages 12–20. Springer, 1974.

[39] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[40] Laurent Demaret, Armin Iske, et al. Advances in digital image compression by adaptive thinning. *Annals of the MCFA*, 3:105–109, 2004.

[41] Laurent Demaret, Nira Dyn, and Armin Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604–1616, 2006.

[42] Ronald A DeVore, Björn Jawerth, and Bradley J Lucier. Image compression through wavelet transform coding. *IEEE Transactions on information theory*, 38(2):719–746, 1992.

[43] Ronald A DeVore, Björn Jawerth, and Bradley J Lucier. Surface compression. *Computer Aided Geometric Design*, 9(3):219–239, 1992.

[44] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[45] David L Donoho. Interpolating wavelet transforms. *Preprint, Department of Statistics, Stanford University*, 2(3), 1992.

[46] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[47] David L Donoho and Iain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, pages 425–455, 1994.

[48] Paul Escande and Pierre Weiss. Sparse wavelet representations of spatially varying blurring operators. *SIAM Journal on Imaging Sciences*, 8 (4):2976–3014, 2015.

[49] Emmanuel Faure, Thierry Savy, Barbara Rizzi, Camilo Melani, Olga Stašová, Dimitri Fabrèges, Róbert Špir, Mark Hammons, Róbert Čúnderlík, Gaëlle Recher, et al. A workflow to process 3d+ time microscopy images of developing organisms and reconstruct their cell lineage. *Nature communications*, 7, 2016.

[50] Robert W Floyd. An adaptive algorithm for spatial gray-scale. In *Proc. Soc. Inf. Disp.*, volume 17, pages 75–77, 1976.

[51] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[52] Jerome H Friedman and Bernard W Silverman. Flexible parsimonious smoothing and additive modeling. *Technometrics*, 31(1):3–21, 1989.

[53] Laurent Gole, Kok Haur Ong, Thomas Boudier, Weimiao Yu, and Sohail Ahmed. Opensegspim: a user-friendly segmentation tool for spim data. *Bioinformatics*, 32(13):2075–2077, 2016.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[55] Ami Harten. Adaptive multiresolution schemes for shock computations. *Journal of Computational Physics*, 115(2):319–338, 1994.

[56] Idse Heemskerk and Sebastian J Streichan. Tissue cartography: compressing bio-image data by dimensional reduction. *Nature methods*, 12 (12):1139–1142, 2015.

[57] John M Henderson. Human gaze control during real-world scene perception. *Trends in cognitive sciences*, 7(11):498–504, 2003.

[58] Weizhang Huang and Robert D Russell. *Adaptive moving mesh methods*. Springer Science & Business Media, 2010.

[59] Weizhang Huang, Yuhe Ren, and Robert D Russell. Moving mesh partial differential equations (mmpdes) based on the equidistribution principle. *SIAM Journal on Numerical Analysis*, 31(3):709–730, 1994.

[60] Jan Huisken, Jim Swoger, Filippo Del Bene, Joachim Wittbrodt, and Ernst HK Stelzer. Optical sectioning deep inside live embryos by selective plane illumination microscopy. *Science*, 305(5686):1007–1009, 2004.

[61] Armin Iske and Laurent Demaret. Optimally sparse image approximation by adaptive linear splines over anisotropic triangulations. In *Sampling Theory and Applications (SampTA), 2015 International Conference on*, pages 463–467. IEEE, 2015.

[62] Maarten Jansen, Hyeokho Choi, Sridhar Lavu, and Richard Baraniuk. Multiscale image processing using normal triangulated meshes. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 2, pages 229–232. IEEE, 2001.

[63] Casper Bo Jensen, Mark Lyksborg, J Hecksher-S, Anna Secher, Knut Conradsen, Anders Bjorholm Dahl, et al. Active appearance segmentation for intensity inhomogeneity in light sheet fluorescence microscopy. In *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*, pages 217–220. IEEE, 2016.

[64] Hao Jiang, Tingting Yu, Jun Nie, Chunyu Fang, Dan Zhu, and Peng Fei. Automatic light-sheet imaging plugin for rapid threedimensional visualization of embryo angiopoiesis on common wide-field microscope. In *Asia Communications and Photonics Conference*, pages AF4K–4. Optical Society of America, 2016.

[65] Martin Jinek, Krzysztof Chylinski, Ines Fonfara, Michael Hauer, Jennifer A Doudna, and Emmanuelle Charpentier. A programmable dual-rna–guided dna endonuclease in adaptive bacterial immunity. *Science*, 337(6096):816–821, 2012.

[66] Elcin Kartal Koc and Hamparsum Bozdogan. Model selection in multivariate adaptive regression splines (mars) using information complexity as the fitness function. *Machine Learning*, 101(1):35–58, 2015. ISSN 1573-0565. doi: 10.1007/s10994-014-5440-5. URL `http://dx.doi.org/10.1007/s10994-014-5440-5`.

[67] Philipp J Keller, Annette D Schmidt, Joachim Wittbrodt, and Ernst HK Stelzer. Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy. *Science*, 322(5904):1065–1069, 2008.

[68] Kristin Koch, Judith McLean, Ronen Segev, Michael A Freed, Michael J Berry, Vijay Balasubramanian, and Peter Sterling. How much the eye tells the brain. *Current Biology*, 16(14):1428–1434, 2006.

[69] Max Köhler, Anja Schindler, and Stefan Sperlich. A review and comparison of bandwidth selection methods for kernel regression. *International Statistical Review*, 82(2):243–274, 2014.

[70] Uros Krzic, Stefan Gunther, Timothy E Saunders, Sebastian J Streichan, and Lars Hufnagel. Multiview light-sheet microscope for rapid in toto imaging. *Nature methods*, 9(7):730–733, 2012.

[71] Xinghua Lou, Minjung Kang, Panagiotis Xenopoulos, Silvia Munoz-Descalzo, and Anna-Katerina Hadjantonakis. A rapid and efficient 2d/3d nuclear segmentation method for analysis of early mouse embryo and stem cell image data. *Stem cell reports*, 2(3):382–397, 2014.

[72] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[73] Stephane Mallat and Wen Liang Hwang. Singularity detection and processing with wavelets. *IEEE transactions on information theory*, 38(2): 617–643, 1992.

[74] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.

[75] B. Mathew, A. Schmitz, S. Muñoz-Descalzo, N. Ansari, F. Pampaloni, E.H.K. Stelzer, and S.C. Fischer. Robust and automated three-dimensional segmentation of densely packed cell nuclei in different biological specimens with lines-of-sight decomposition. *BMC Bioinformatics*, 16(1):187, 2015. ISSN 1471-2105. doi: 10.1186/s12859-015-0617-x. URL `http://dx.doi.org/10.1186/s12859-015-0617-x`.

[76] Jon Mathews and Robert Lee Walker. *Mathematical methods of physics*, volume 501. WA Benjamin New York, 1970.

[77] Patrick Min. binvox, 3d mesh voxelizer, 2017. URL `http://www.patrickmin.com/binvox/`.

[78] Ricardo H Nochetto and Andreas Veeser. Primer of adaptive finite element methods. In *Multiscale and adaptivity: modeling, numerics and applications*, pages 125–225. Springer, 2011.

[79] Fakir S. Nooruddin and Greg Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):191–205, 2003.

[80] Andrew C Oates, Nicole Gorfinkiel, Marcos Gonzalez-Gaitan, and Carl-Philipp Heisenberg. Quantitative approaches in developmental biology. *Nature Reviews Genetics*, 10(8):517–530, 2009.

[81] Kate Okikiolu. Characterization of subsets of rectifiable curves in rn. *Journal of the London Mathematical Society*, 2(2):336–348, 1992.

[82] Hanchuan Peng, Zongcai Ruan, Fuhui Long, Julie H Simpson, and Eugene W Myers. V3d enables real-time 3d visualization and quantitative analysis of large-scale biological image data sets. *Nature biotechnology*, 28(4):348–353, 2010.

[83] Gabriel Peyré. A review of adaptive image representations. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):896–911, 2011.

[84] Tobias Pietzsch, Stephan Saalfeld, Stephan Preibisch, and Pavel Tomancak. Bigdataviewer: visualization and processing for large image data sets. *Nature methods*, 12(6):481–483, 2015.

[85] Gerlind Plonka. The easy path wavelet transform: A new adaptive wavelet transform for sparse representation of two-dimensional data. *Multiscale Modeling & Simulation*, 7(3):1474–1496, 2009.

[86] Rory M Power and Jan Huisken. A guide to light-sheet fluorescence microscopy for multiscale imaging. *Nature Methods*, 14(4):360–373, 2017.

[87] Douglas C Prasher, Virginia K Eckenrode, William W Ward, Frank G Prendergast, and Milton J Cormier. Primary structure of the aequorea victoria green-fluorescent protein. *Gene*, 111(2):229–233, 1992.

[88] Stephan Preibisch, Torsten Rohlfing, Michael P Hasak, and Pavel Tomancak. Mosaicing of single plane illumination microscopy images using groupwise registration and fast content-based image fusion. *Medical Imaging 2008: Image Processing*, 6914(1):69140E, 2008.

[89] Stephan Preibisch, Stephan Saalfeld, Torsten Rohlfing, and Pavel Tomancak. Bead-based mosaicing of single plane illumination microscopy images using geometric local descriptor matching. In *Proc. of SPIE Vol*, volume 7259, pages 72592S–1, 2009.

[90] Stephan Preibisch, Stephan Saalfeld, Johannes Schindelin, and Pavel Tomancak. Software for bead-based registration of selective plane illumination microscopy data. *Nature methods*, 7(6):418–419, 2010.

[91] Stephan Preibisch, Fernando Amat, Evangelia Stamataki, Mihail Sarov, Robert H Singer, Eugene Myers, and Pavel Tomancak. Efficient bayesian-based multiview deconvolution. *nature methods*, 11(6):645–648, 2014.

[92] Hayder Radha, Riccardo Leonardi, Martin Vetterli, and Bruce Naylor. Binary space partitioning tree representation of images. *Journal of Visual Communication and Image Representation*, 2(3):201–221, 1991.

[93] Sylvain Reboux, Birte Schrader, and Ivo F Sbalzarini. A self-organizing lagrangian particle method for adaptive-resolution advection–diffusion simulations. *Journal of Computational Physics*, 231(9):3623–3646, 2012.

[94] JD Regele and OV Vasilyev. An adaptive wavelet-collocation method for shock computations. *International Journal of Computational Fluid Dynamics*, 23(7):503–518, 2009.

[95] Pamela Reinagel and Anthony M Zador. Natural scene statistics at the centre of gaze. *Network: Computation in Neural Systems*, 10(4):341–350, 1999.

[96] Emmanuel G Reynaud, Jan Peychl, Jan Huisken, and Pavel Tomancak. Guide to light-sheet microscopy for adventurous biologists. *Nature methods*, 12(1):30–34, 2015.

[97] Diego Rossinelli, Babak Hejazialhosseini, Wim van Rees, Mattia Gazzola, Michael Bergdorf, and Petros Koumoutsakos. Mrag-i2d: multi-resolution adapted grids for remeshed vortex methods on multicore architectures. *Journal of Computational Physics*, 288:1–18, 2015.

[98] Loic A Royer, Martin Weigert, Ulrik Günther, Nicola Maghelli, Florian Jug, Ivo F Sbalzarini, and Eugene W Myers. Clearvolume: open-source live 3d visualization for light-sheet microscopy. *Nature methods*, 12(6): 480–481, 2015.

[99] Loïc A Royer, William C Lemon, Raghav K Chhetri, Yinan Wan, Michael Coleman, Eugene W Myers, and Philipp J Keller. Adaptive light-sheet microscopy for long-term, high-resolution imaging in living organisms. *Nature biotechnology*, 34(12):1267–1278, 2016.

[100] Jose L Rubio-Guivernau, Vasily Gurchenkov, Miguel A Luengo-Oroz, Louise Duloquin, Paul Bourgine, Andres Santos, Nadine Peyrieras, and Maria J Ledesma-Carbayo. Wavelet-based image fusion in multi-view three-dimensional microscopy. *Bioinformatics*, 28(2):238–245, 2011.

[101] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2003.

[102] Daniel Sage. Local normalization. http://bigwww.epfl.ch/sage/soft/localnormalization/, 2002 (accessed December 2013).

[103] Xavier Saint Raymond. *Elementary introduction to the theory of pseudodifferential operators*, volume 3. CRC Press, 1991.

[104] Michel Sarkis and Klaus Diepold. Content adaptive mesh representation of images using binary space partitions. *IEEE Transactions on Image Processing*, 18(5):1069–1079, 2009.

[105] Ph Saucez, WE Schiesser, et al. *Adaptive method of lines.* CRC Press, 2001.

[106] Ivo F Sbalzarini. Modeling and simulation of biological systems from image data. *Bioessays*, 35(5):482–490, 2013.

[107] Ivo F Sbalzarini, Jens H Walther, Michael Bergdorf, Simone Elke Hieber, Evangelos M Kotsalis, and Petros Koumoutsakos. Ppm–a highly efficient parallel particle–mesh library for the simulation of continuum systems. *Journal of Computational Physics*, 215(2):566–588, 2006.

[108] Nico Scherf and Jan Huisken. The smart and gentle microscope. *Nature biotechnology*, 33(8):815–818, 2015.

[109] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9(7):676–682, 2012.

[110] Benjamin Schmid and Jan Huisken. Real-time multi-view deconvolution. *Bioinformatics*, 31(20):3398–3400, 2015.

[111] Benjamin Schmid, Gopi Shah, Nico Scherf, Michael Weber, Konstantin Thierbach, Citlali Pérez Campos, Ingo Roeder, Pia Aanstad, and Jan Huisken. High-speed panoramic light-sheet microscopy reveals global endodermal cell dynamics. *Nature communications*, 4, 2013.

[112] Michael Schmidt and Hod Lipson. Eureqa. *Nutonian, Somerville, Mass, USA*, 2013.

[113] Christopher Schmied, Peter Steinbach, Tobias Pietzsch, Stephan Preibisch, and Pavel Tomancak. An automated workflow for parallel processing of large multiview spim recordings. *Bioinformatics*, 32(7): 1112–1114, 2015.

[114] Kai Schneider and Oleg V Vasilyev. Wavelet methods in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 42:473–503, 2010.

[115] Birte Schrader, Sylvain Reboux, and Ivo F Sbalzarini. Discretization correction of general integral pse operators for particle methods. *Journal of Computational Physics*, 229(11):4159–4182, 2010.

[116] Stelios M Smirnakis, Michael J Berry, David K Warland, William Bialek, and Markus Meister. Adaptation of retinal processing to image contrast and spatial scale. *Nature*, 386(6620):69, 1997.

[117] Johannes Stegmaier, Fernando Amat, William C Lemon, Katie McDole, Yinan Wan, George Teodoro, Ralf Mikut, and Philipp J Keller. Real-time three-dimensional cell segmentation in large-scale microscopy data of developing embryos. *Developmental cell*, 36(2):225–240, 2016.

[118] Frederic Strobl, Alexander Schmitz, and Ernst HK Stelzer. Improving your four-dimensional image: traveling through a decade of light-sheet-based fluorescence microscopy research. *Nature Protocols*, 12(6):1103–1109, 2017.

[119] Willy Supatto, Amy McMahon, Scott E Fraser, and Angelike Stathopoulos. Quantitative imaging of collective cell migration during drosophila gastrulation: multiphoton microscopy and computational analysis. *Nature protocols*, 4(10):1397–1412, 2009.

[120] Wim Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM journal on mathematical analysis*, 29(2):511–546, 1998.

[121] Jim Swoger, Peter Verveer, Klaus Greger, Jan Huisken, and Ernst HK Stelzer. Multi-view image fusion improves resolution in three-dimensional microscopy. *Optics express*, 15(13):8029–8042, 2007.

[122] Maja Temerinac-Ott, Olaf Ronneberger, Roland Nitschke, Wolfgang Driever, and Hans Burkhardt. Spatially-variant lucy-richardson deconvolution for multiview fusion of microscopical 3d images. In *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, pages 899–904. IEEE, 2011.

[123] Maja Temerinac-Ott, Olaf Ronneberger, Peter Ochs, Wolfgang Driever, Thomas Brox, and Hans Burkhardt. Multiview deblurring for 3-d images from light-sheet-based fluorescence microscopy. *IEEE Transactions on Image Processing*, 21(4):1863–1873, 2012.

[124] The HDF Group. Hierarchical Data Format, version 5, 1997-2017. http://www.hdfgroup.org/HDF5/.

[125] Raju Tomer, Khaled Khairy, Fernando Amat, and Philipp J Keller. Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy. *Nature Methods*, 9(7):755–763, 2012.

[126] Ivana Tosic and Pascal Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28(2):27–38, 2011.

[127] Thai V Truong and Willy Supatto. Toward high-content/high-throughput imaging and analysis of embryonic morphogenesis. *genesis*, 49(7):555–569, 2011.

[128] Michael Unser and Akram Aldroubi. A review of wavelets in biomedical applications. *Proceedings of the IEEE*, 84(4):626–638, 1996.

[129] Michael Unser, Akram Aldroubi, and Murray Eden. B-spline signal processing. ii. efficiency design and applications. *IEEE transactions on signal processing*, 41(2):834–848, 1993.

[130] Lucas J Van Vliet, Frank R Boddeke, Damir Sudar, and Ian T Young. Image detectors for digital image microscopy. *Digital Image Analysis of Microbes: Imaging, Morphometry, Fluorometry, and Motility Techniques and Applications*, pages 37–63, 1998.

[131] Oleg V Vasilyev and Christopher Bowman. Second-generation wavelet collocation method for the solution of partial differential equations. *Journal of Computational Physics*, 165(2):660–693, 2000.

[132] Peter J Verveer, Jim Swoger, Francesco Pampaloni, Klaus Greger, Marco Marcello, and Ernst HK Stelzer. High-resolution three-dimensional imaging of large specimens with light sheet–based microscopy. *Nature methods*, 4(4):311–313, 2007.

[133] Giuseppe Vicidomini. Image formation in fluorescence microscopy. In *From Cells to Proteins: Imaging Nature across Dimensions*, pages 371–393. Springer, 2005.

[134] Eric Wait, Mark Winter, Chris Bjornsson, Erzsebet Kokovay, Yue Wang, Susan Goderie, Sally Temple, and Andrew R Cohen. Visualization and correction of automated segmentation, tracking and lineaging from 5-d stem cell image sequences. *BMC bioinformatics*, 15(1):328, 2014.

[135] Yao Wang and Ouseb Lee. Active mesh-a feature seeking and tracking image sequence representation scheme. *IEEE Transactions on image processing*, 3(5):610–624, 1994.

[136] Michael Weber and Jan Huisken. Light sheet microscopy for real-time developmental biology. *Current opinion in genetics & development*, 21 (5):566–572, 2011.

[137] Michael Weber and Jan Huisken. Omnidirectional microscopy. *nature methods*, 9(7):656–657, 2012.

[138] Martin Weigert, Loic Royer, Florian Jug, and Gene Myers. Isotropic reconstruction of 3d fluorescence microscopy images using convolutional neural networks. *arXiv preprint arXiv:1704.01510*, 2017.

[139] Andrew Witkin. Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 150–153. IEEE, 1984.

[140] Carsten Wolff, Jean-Yves Tinevez, Tobias Pietzsch, Evangelia Stamataki, Benjamin Harich, Stephan Preibisch, Spencer Shorte, Philipp J Keller, Pavel Tomancak, and Anastasios Pavlopoulos. Reconstruction of cell lineages and behaviors underlying arthropod limb outgrowth with multi-view light-sheet imaging and tracking. *bioRxiv*, page 112623, 2017.

[141] Yongyi Yang, Miles N Wernick, and Jovan G Brankov. A fast approach for accurate content-adaptive mesh generation. *IEEE transactions on image processing*, 12(8):866–881, 2003.

[142] Rongkai Zhao, Tao Tao, Michael Gabriel, and Geneva G Belford. Lossless compression of very large volume data with fast dynamic access. In *Proc. SPIE*, volume 4925, page 180, 2002.

Appendix

A Appendix

A.1 Thought experiment: eye as a camera

Here we present the details of the 'back of the envelope' heuristic for what the data rate would be if the human visual system functioned as a traditional fluorescence microscope. By this we mean if the eye recorded the visual scene constantly at pixels with a resolution set by the finest resolution detectable, and field of view set to the effective field of view of the visual system. [32] reviews the literature and gives an effective image size of 576 Mega Pixels. That is $576 * 10^6$ pixels, assuming a 120-degree field of view. We then assume that the eye records this image at 150 fps. This is within the maximum observed rates of over 500 fps recently shown to be detectable in Davis et al. [36]. We then assume we are storing color, hence three channels, and we assume this must be done using 64 bit unsigned as perceived light levels can vary over nine orders of magnitude in a day [116].

Combining these gives a total number of pixels per second of $576 * 1000 * 1000 * 3 * 150 = 2.592 * 10^{11}$ or 2075 GB per second. In a day this would amount to approximately $180,000$ TB per day of data.

A.2 Besov space definition

Here we reproduce the definition of a Besov space as presented in DeVore et al. [42]. Besov spaces are useful in classification of error and performance of multiresolution schemes, in particular, the analysis of wavelets. Besov spaces, are defined as $B_q^\alpha(L^p(\Omega))$, with $\alpha > 0$, $0 < p < \infty$, $0 < q \leq \infty$ and Ω is the spatial domain. Do define a Besov space we first need the following definitions. Let the forward difference with $h \in \mathbb{R}^2$ be

$$\Delta_h^k f(x) = \begin{cases} f(x) & k = 0 \\ \Delta_h^{k-1} f(x+h) - \Delta_h^{k-1} f(x) & k = 1, .., r \end{cases} \tag{A.1}$$

which then is used to define the rth modulus of smoothness in L^p as

$$\omega_r(f,t) = \sup_{|h| \leq t} \left(\int_{\Omega_{rh}} |\Delta_h^k f(x)|^p dx \right)^{1/p} \tag{A.2}$$

then a Besov space $B_q^\alpha(L^p(\Omega))$ consists of all f for with the following is bounded

$$\|f\|_{B_q^\alpha(L^p(\Omega))} = \|f\|_{L^p(\Omega)} + \left(\int_0^\infty \frac{(t^{-\alpha}\omega_r(f,t)_p)^q}{t} dt \right)^{1/q} \tag{A.3}$$

which is a norm for $p, q > 1$, and a quisi-norm otherwise.

A.2.1 Reconstruction Condition vs. Resolution Bound

First, we consider the relationship between the optimal solution to the Reconstruction Condition $R_b(\mathbf{y})$ and optimal solution to the Resolution Bound $R_c(\mathbf{y})$. Since the Resolution Bound is derived from an upper bound on the Reconstruction Condition (given the appropriate assumptions), a Resolution Function satisfying the Resolution Bound also satisfies the Reconstruction Condition. Hence the Reconstruction Condition is a tighter bound on $R(\mathbf{y})$ and $R_b(\mathbf{y}) \leq R_c(\mathbf{y})$ for all $\mathbf{y}$. The difference between R_c and R_b is the results of bounding the error by taking the uniform estimate of the maximum of the gradient across the interval 5.9, instead of the exact path integral in 5.7. Without restrictions on both E and the function f, unfortunately, I know of no upper bound on this difference. However, if we consider f that is infinitely differentiable and the limit as $E \to 0$, we observe that either: f will be constant in some interval and $R(y)$ will also reach some lower bound with a constant zero derivative, or, $R_b(y), R_c(y) \to 0$. In the first case trivially the two bounds are equal. In the second case, since f is assumed to be infinitely differentiable from its Taylor series expansion the difference between $|R_b(y) - R_c(y)| \to 0$ as $R_b(y), R_c(y) \to 0$. Hence, given assumptions in the small E limit, the solutions converge.

A.2.2 Bounds for Implied Resolution Function

Next, we consider what the relationship between $R_c(\mathbf{y})$ and the Implied Resolution Function $R^*(\mathbf{y})$ generated by the Optimal Valid Particle Cell set $\mathcal{V}$. Given that $R_b(\mathbf{y})$ and $R^*(\mathbf{y})$ both satisfy the Resolution Bound, but, $R^*(\mathbf{y})$ is restricted to be piecewise constant then necessarily $R_b(\mathbf{y}) \leq R^*(\mathbf{y})$. The difference between these two solutions represents the loss in adaptation resulting from constructing our solution of Particle Cells instead of allowing continuous adaptation. Since the solutions are both optimal (R^* over restricted solutions) for the Resolution Bound, we can bound the worst case difference between these two solutions.

Let us have some $\mathcal{V}$ that satisfies the Resolution Bound for $L(\mathbf{y})$. We then ask, in the worst case how much larger could R_b be compared to R^*? That is, what is the bound on $\frac{R_c(\mathbf{y})}{R^*(\mathbf{y})}$ given only knowledge of R^*.

We can evaluate this question by considering a bound for all particle cells $c_{\mathbf{i},l} \in \mathcal{V}$. Given that $c_{\mathbf{i},l}$ belongs to the optimal set, from Theorem 2, we know that its parent $c_{\mathbf{i}/2,l}$ must violate Theorem 1. Explicitly, that is

$$\{\mathcal{L} \cap \mathcal{ND}(c_{\mathbf{i}/2,l})\} \supseteq c^n_{\mathbf{i},l}. \tag{A.4}$$

Now, there are many combinations of $L(\mathbf{y})$ and $R_b(\mathbf{y})$ that could results in that this situation. However, the worst case, i.e. that allowing the largest $R_b(\mathbf{y})$ over the spatial domain of the particle cell is unique (ignoring equivalent configurations). The worst case occurs when $L(\mathbf{y})$ is the largest distance from the particle cell at $\mathbf{y}^*$, and occurs exactly on the interval between two particle cells i.e. $L(\mathbf{y}^*) = \frac{\Omega}{2^{l-1}}$ and for $y^* \in s(c^n_{\mathbf{i},l})$. If we assume that $L(\mathbf{y})$ is a dirac delta, where $\mathbf{y}^*$ is the only non-zero point (again worst case as it provide minimal restriction on the solution). In this way the optimal continuous solution of the Resolution Bound for this is

$$R^*_b(\mathbf{y},\mathbf{y}^*) \begin{cases} dist(\mathbf{y},\mathbf{y}^*) & dist(\mathbf{y},\mathbf{y}^*) \geq L(\mathbf{y}^*) \\ L(\mathbf{y}^*) & dist(\mathbf{y},\mathbf{y}^*) < L(\mathbf{y}^*) \end{cases} \tag{A.5}$$

where $dist(.,.)$ is the Euclidean distance between two points. This can be trivially proven by directly considering the solution for an $R^*_b(\mathbf{y},\mathbf{y}^*) + \delta$ for $\delta > 0$, and noting that the bound no longer holds. We can use this then to consider the direct upper bound, i.e. within $s(c_{\mathbf{i},l})$ how large can $R^*_b(\mathbf{y},\mathbf{y}^*)$ be? If we consider the distance of the furthest point from $\mathbf{y}^*$ that is in $s(c^n_{\mathbf{i},l})$ we get a worst case of $R^*_b(\mathbf{y},\mathbf{y}^*) \leq (4\frac{\Omega}{2^l})d^{1/2}$ and hence, we have

$$\frac{R_b(\mathbf{y})}{R^*(\mathbf{y})} \leq 4\sqrt{d} \tag{A.6}$$

where d is the dimension. Hence this corresponds to ratios bounded by 4, ≈ 5.65 and ≈ 6.93 in 1D, 2D and 3D respectively.

A.2.3 Bounds on particle sampling

Can we construct a similar bound on the Particle Sampling $\mathcal{P}$. Here we consider samplings restricted to those that satisfy

$$\#\mathcal{P} = \int_\Omega \frac{1}{R(\mathbf{y})^d} d\Omega. \tag{A.7}$$

We note that if we assume an analytical form of $R(\mathbf{y})$ then $\mathcal{P}$ that are far smaller then those obeying the above bound can be constructed. However, we

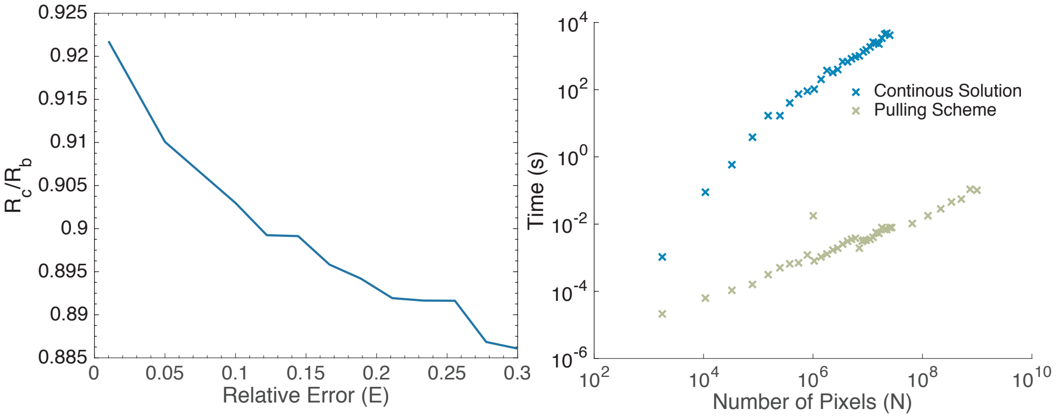


Figure A.1: The first plot shows the ratio of a numerical estimate of the ratio between the optimal continuous solutions to the Resolution Bound R_c and Resolution Condition R_b for the benchmark problem in Figure 6.1 plotted against the relative error E. We note that smaller values of E resulted in a prohibitive computational cost. The second plot shows the relative execution time in 3D of a numerical estimate to R_c and the execution time of the pulling scheme for a fixed ratio content benchmark plotted against increasing number of pixels N. We note that the continuous solution became computationally prohibitive beyond a maximum width of 128 (Beyond which the continuous solution took over 2 hours to estimate compared to less than $.01$ seconds for the pulling scheme).

only consider samplings that follow A.7. Given this assumption, we can then ask if we can also bound $\frac{\#\mathcal{P}}{\#\mathcal{P}_c}$, where $\mathcal{P}_c$ is some sampling satisfying A.7 for $R_c(\mathbf{y})$. If we consider the same worst case scenario as above we get,

$$\#\mathcal{P}_c \leq \#\mathcal{P} \int_{s(c_{\mathbf{i},l})} \frac{1}{(dist(\mathbf{y},\mathbf{y}^*))^d} d\mathbf{y} \tag{A.8}$$

where $c_{\mathbf{i},l}$ and $\mathbf{y}^*$ are as above, and we note that the argument is independent of the exact level l, and so we consider the ratio of each individual particle cell to be the same giving the multiplication factor. In one and two dimensions this has closed form, in 1D we have

$$\begin{aligned} \#\mathcal{P}_c &\leq \#\mathcal{P} \int_3^4 \frac{1}{(y)^d} dy \\ &= \log(\frac{4}{3})\#\mathcal{P} \end{aligned} \tag{A.9}$$

and in 2D

$$\begin{aligned} \#\mathcal{P}_c &\leq \#\mathcal{P} \int_3^4 \int_3^4 \frac{1}{(\sqrt{x^2+y^2})^d} dydx \\ &= \#\mathcal{P}(-2G + \frac{1}{2}\pi \log(\frac{4}{3}) + i \left(\text{PolyLog}(2, \frac{-3i}{4}) - \text{PolyLog}(2, \frac{3i}{4}) \right)) \end{aligned} \tag{A.10}$$

where G is Catalan's constant and PolyLog is the PolyLogarithm. This corresponds to $\frac{\#\mathcal{P}}{\#\mathcal{P}_c}$ ratios being bounded by 3.47 in 1D, 24.3 in 2D, and using numerical integration in 3D 221.252. However, with the exception of possibly 1D, I know of no methods to realize these ratios. Further, there bounds are also not likely tight in practice, and I am unclear as to their utility.

Observed numerical bounds

To find more likely ratios of $\frac{\#\mathcal{P}}{\#\mathcal{P}_c}$ in practice, we numerically estimated the continuous optimal Resolution Functions using an $\mathcal{O}(N^2)$ brute force approach. The brute force approach relies on testing increasing sized $R(y)$ for each location. The first plot in Figure A.1 shows the ratio of R_c over R_b for the test function in Figure 6.1 for decreasing E. We find that as suggested by the discussion above $\frac{R_c}{R_b} \to 1$ as $E \to 0$. Unfortunately, the solving for smaller values of E than 0.01 was too computationally costly.

To illustrate this high computational cost, in the second plot Figure A.1, we show a comparison in 3D between the brute force solution to the Resolution Bound and the pulling scheme for a fixed ratio benchmark. We find that the brute force solution takes between two and six orders of magnitude longer to compute. This corresponds to the brute force solution taking over 2 hours for an image of $N = 128^3$, compared to less than .01 seconds for the pulling scheme. We note that efforts that the I attempted to optimize the brute force scheme and the computed time including acceleration using OpenMP to provide a 'fairer' comparison with the pulling scheme. This high cost placed a limit on the numerical analysis that could easily be done comparing the two solutions.

Next, we explored the relationship between the implied Resolution Function and the continuous estimate of R_c. We focus here on 3D. The first plot of Figure A.2, shows the mean ratio of $\frac{R_c}{R^*}$, averaged over all pixel locations against increasing number of objects for both noisy and noise-free original images. We find the average ratio is less than 3.2 across both benchmarks, with a decrease in the mean ratio for an increase in objects for the images. Therefore, on average we find the ratio is two to three times less than the worst-case bound of ≈ 6.93.

Lastly, we use the estimate of R_c to also estimate the $\frac{\#\mathcal{P}}{\#\mathcal{P}_c}$ ratios. From the above analysis, we found a worst case bound of 225.21. In the second plot of Figure A.2 we show the estimated ratios against increasing information content for sampling based on both the isotropic and integral neighborhood sampling used in the performance benchmarks (5.5.3). We find the ratio becomes constant for increasing information content, with the isotropic sampling having a ratio of approximately 11, and 5.5 for the integral neighborhood sampling.

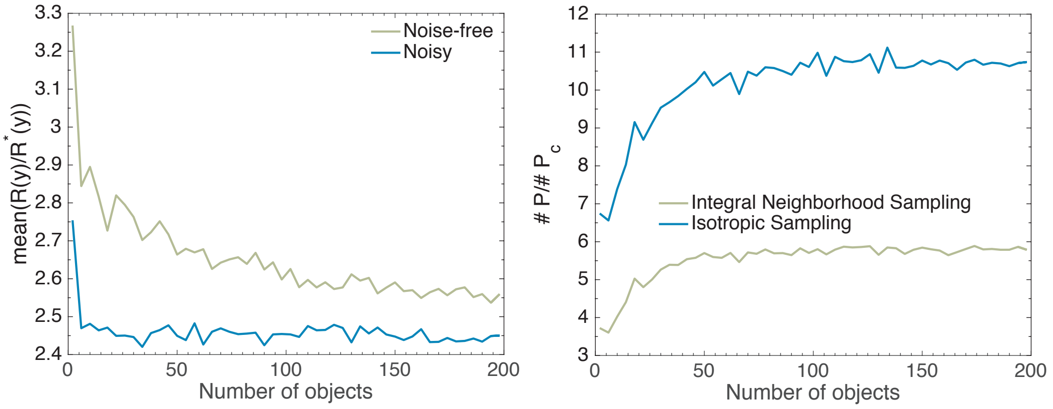


Figure A.2: The first plot shows the mean ratio of the numerical estimate of the optimal continuous Resolution Function $R_c(\mathbf{y})$ over the implied Resolution Function $R^*(\mathbf{y})$ using the integral neighborhood sampling optimization in 3D against number of objects for both noisy and noise-free images. The second plot shows the ratio of the number particles in the APR $\#\mathcal{P}$ divided by the theoretical sampling $\#\mathcal{P}_c$ based on R_c. We plot the ratios for both the isotropic and integral neighborhood sampling (5.5.3) in 3D against the number of objects.

Although this is only for one test example, we find a ratio that is much less than the worst case bound given above.

A.2.4 Optimal continuous $\mathcal{O}(N^{\frac{d-1}{d}+1})$ algorithm

Inspired by the form of the pulling scheme, and the functional form of the dirac delta like solution above, a more efficient algorithm for finding a numerical approximation to the optimal solution R_c could be considered. This involves utilizing the geometry of the Resolution Bound. It involves iterating over each sample location N, then iterating over a $d-1$ dimensional sphere centered at the sample location with radius equal to $L(\mathbf{y})$. At each location, another function, $T(\mathbf{y})$ should be set to the minimum value of either its current value or $L(\mathbf{y})$. Where $T(\mathbf{y})$ is initialized to an arbitrary large value. If this is done over each point, the resulting $T(\mathbf{y})$ should be equal to the numerical estimate of $R_c(\mathbf{y})$. However, the computational complexity per pixel is then bounded by $\mathcal{O}(N^{\frac{d-1}{d}})$, assuming a square sampling mesh. This then results in a reduced complexity of worst-case $\mathcal{O}(N^{\frac{d-1}{d}+1})$ compared to $\mathcal{O}(N^2)$ in the brute-force approach. We do not explore this further here.

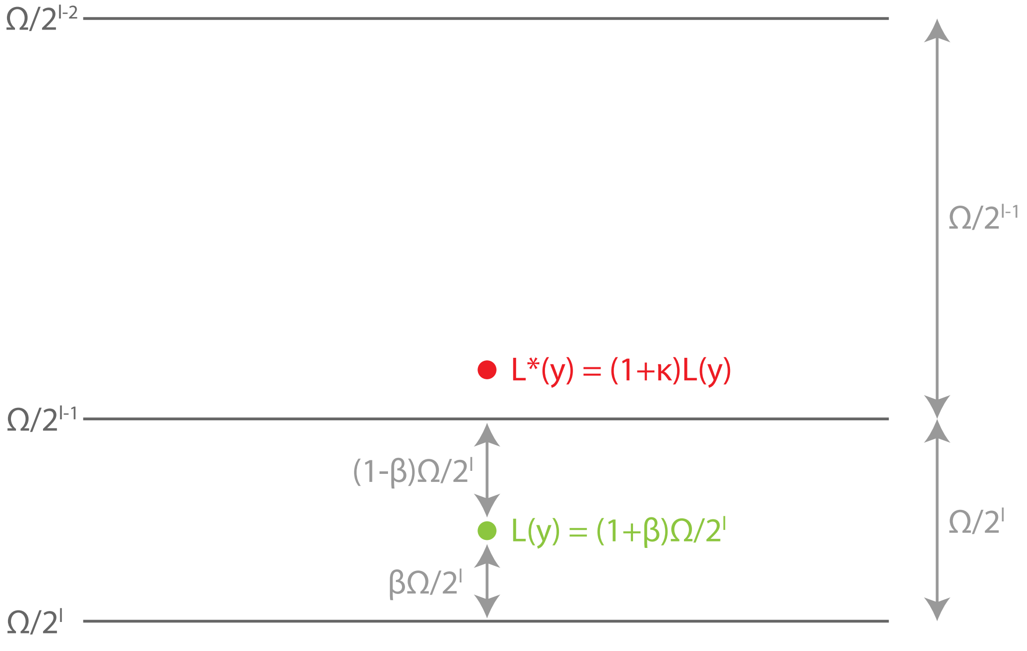


Figure A.3: Resolution Domain analysis of errors in local resolution estimate $L(\mathbf{y})$

A.3 Impact of noisy Local Resolution Estimate $L(\mathbf{y})$

Here we address the impact of error in the estimation of $L(\mathbf{y})$ on the ability of the APR to reconstruct the function within the Reconstruction Condition. Now if we suppose that we have the following scenario, as shown in Figure. A.3

$$L(\mathbf{y}) = (1+\beta)\frac{\Omega}{2^l} \tag{A.11}$$

that is subject to some error, such that the observed local resolution estimate is

$$L^*(\mathbf{y}) = L(\mathbf{y})(1+\kappa) \tag{A.12}$$

for this error to impact the solution such that it reduces the reconstruction error, the particle cell level must decrease, giving a new particle cell level $l^* = l - \phi$ which requires

$$L(\mathbf{y})\kappa > (1-\beta)\frac{\Omega}{2^l} \tag{A.13}$$

which gives us

$$\kappa > \frac{1-\beta}{1+\beta} \tag{A.14}$$

and that $\phi = \lceil \kappa \rceil$. Now we wish to consider the worse case change in the resolution that would occur, given a particular relative error κ. That is we wish to consider when the error between the true local estimate and the observed quantized value from the particle cell, given by

$$\begin{aligned} \Delta &= \frac{\Omega}{2^{l-\phi}} - L(y) \\ &= \frac{\Omega}{2^{l}}(2^{\phi} - 1 - \beta) \end{aligned} \tag{A.15}$$

is at its largest, which occurs when

$$\kappa = \frac{(2^{\phi} - 1 - \beta)}{1+\beta}. \tag{A.16}$$

Now for the APR we have $L(\mathbf{y}) = \frac{E\sigma(\mathbf{y})}{|\nabla f|}$, and then our observed local resolution estimate can be written as

$$L^{*}(y) = \frac{E\sigma(\mathbf{y})}{|\nabla f|}(1+\kappa) \tag{A.17}$$

and if we now assume that the error in the observed $L(y)$ comes from a underestimate of the gradient magnitude we can re-write this as

$$L^{*}(y) = \frac{E\sigma(\mathbf{y})}{|\nabla f|(1-\alpha)} \tag{A.18}$$

where

$$\alpha = 1 - \frac{1}{1+\kappa}. \tag{A.19}$$

Now we wish to know how a change in α would impact our observed relative error bound E^{*}, compared to our desired relative error bound E, that is

$$E^{*} - E = |\nabla f| \frac{\Omega}{2^{l}}(2^{\phi} - 1 - \beta)\frac{1}{\sigma(\mathbf{y})} \tag{A.20}$$

and therefore the relative error is

$$\frac{E^{*} - E}{E} = |\nabla f| \frac{\Omega}{2^{l}}(2^{\phi} - 1 - \beta)\frac{1}{E\sigma(\mathbf{y})} \tag{A.21}$$

where we assume that the maximum gradient magnitude occurs at $\mathbf{y}$ within its neighborhood $\mathcal{N}(\mathbf{y}, R(\mathbf{y}))$. Substituting in for the true gradient magnitude $|\nabla f| = \frac{E\sigma}{(1+\beta)\frac{\Omega}{2^l}}$ we have

$$\begin{aligned}\frac{E^* - E}{E} &= \frac{(2^{\phi} - 1 - \beta)}{1 + \beta} \\ &= \kappa \\ &= \frac{1}{1 - \alpha} - 1 \end{aligned} \tag{A.22}$$

and similarly

$$\alpha = 1 - \frac{1}{1 + \frac{(E^* - E)}{E}}. \tag{A.23}$$

This is derived for the error occurring occuring across a particle cell. How does this extend to other particle cells? Unfortunately I only have a word argument for its extension currently. The above analysis has assumed that the error occurs at its maximum value across the whole path. Any paths that cover more than one particle cell, will have contributions proportional to the length of the path in each cell. The worst case, would be that this largest relative error occurs everywhere. It is in this case, that this upper bound should hold. Further development of this bound and analysis in the future seems warranted, in addition, to the increase in cost for particles through increases in resolution.

A.4 APR 3D processing pipeline

In this paper, we have discussed and implemented one particular implementation of a pipeline for the APR optimised for large $3D$ images originally stored as tiffs. We have attempted to optimise the steps to reduce memory overhead and computation time, and to reduce the number of parameters and library dependencies. However, with the exception of requiring an estimation of $L(y)$, and an implementation of a pulling scheme, we imagine the possibility of use of vastly different algorithms, implementations and definitions of the estimation of the local scale function, gradient magnitude, and particle intensity.

A.4.1 Implementation

We have implemented the discussed pipeline in a C++ library. We have utilized shared memory parallelism across the pipeline where suited using

OpenMP [20]. We have favored the use of shared-memory parallelism over the use of GPU acceleration, due to the current larger availability of larger capacities of RAM when compared to GPU-memory allowing the processing of larger images on a shared-memory CPU implementation than one relying on fitting the whole pipeline into GPU memory. However, testing has indicated that the use of hardware acceleration, such as GPUs or Intel Xeon Phi's, can provide significant speed ups for particular steps and is left for future work.

The main dependencies of the library, besides OpenMP, are for input and output. For reading images we use LibTiff, and for output and writing the APR, we use HDF5 [124] and the plugin for the blosc compression library [7]. Also, a Java wrapper has been created using SWIG.

In the following section, we will briefly go through each of the steps in the pipeline, discussion the implementation and any parameters or requirements.

Pipeline input and memory requirements

All input images used in this report are read in from 16 bit unsigned int single channel tiff images. The loaded images are then converted to single floating precision, for processing in the pipeline. The time taken to load and convert the image to floating precision is not accounted for in the timing benchmarks in this paper. The pipeline requires approximately 3.125 floating precision copies of the original image in terms of memory storage to carry out the pipeline, given the freeing of temporary variables.

Smoothing B-splines

To be able to estimate function gradients in the presence of noise we fit cubic smoothing B-splines as introduced in [129]. This introduces a regularisation smoothing parameter λ that determines how closely the fit splines must fit the original sample points. We implement these filters using the IIR approach described in [129], using the impulse response for setting the recursive boundary conditions. A strength of this approach is that the algorithm is $\mathcal{O}(1)$ regarding parameter choice λ, and $\mathcal{O}(N)$ in pixels, therefore displaying consistent computational performance across parameter values.

The output of the B-spline coefficients are used for gradient estimation and the local scale function.

Gradient magnitude $|\nabla I|$

Using the computed smoothing B-spline coefficients, the gradient in each direction is computed using the finite difference stencil $(-1/2h_i, 0, 1/2h_i)$. Where

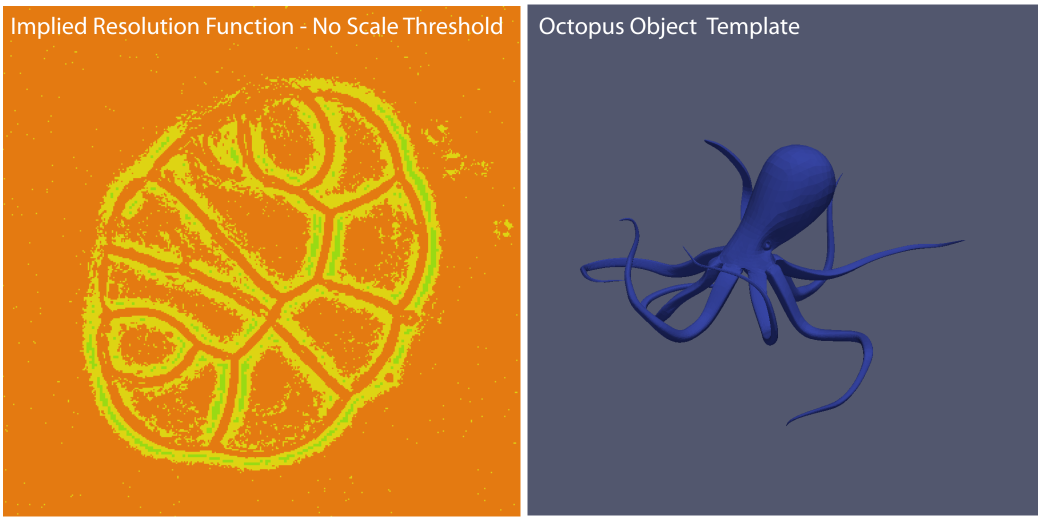


Figure A.4: The Resolution Function $R^*(\mathbf{y})$ for the same image shown in Figure 6.10, without the use of any thresholds for local scale estimation ($\sigma_{th} = 0$). The color scale is identical to that shown in Figure 6.10. In the absence of a threshold, the local scale tends to zero in the background, and therefore fully resolves the background noise. This is mitigated by using a minimum local scale threshold σ_{th}. The original data is from exemplar data set 10 in Table A.1. The second pane shows a volume rendering used for the ocpotus template used to generate the object function for synthetic images in Figure 6.14

h_i is the sampling size in that direction. These are then squared and combined to form the gradient magnitude as below

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial z}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}. \tag{A.24}$$

For noise-free benchmarks, simple finite differences on the original image are performed instead of smoothing B-Splines. Once calculated, the gradient magnitude is then down sampled by a factor of two in each direction, taking the maximum value over each patch of 8 pixels. Due to the local resolution estimate only being required at this resolution.

Local Intensity Scale $\sigma(\mathbf{y})$

For this paper the local intensity scale is intended to be a smoothly varying function that captures the local range of the input image over a certain length scale, allowing the adaptation to cope with changes in contrast across the image domain with varying sources. To set this length scale, we use the inherent length scale provided by the optical process through the point spread function. This length scale then adjusts the size of the window for the smoothing box filters used. The resulting filter is similar to a smoothed estimate of the local standard deviation of the image with filter size adjusted to the estimated standard deviation of a Gaussian approximation to the PSF, σ_{PSFi} in each direction relative to the sampling size, and was originally inspired by [102]. This parameter σ_{PSFi} must be set. Since the Local Resolution Estimate $L(y)$, is calculated at spatial points down sampled by two, and we wish the function to be slowly varying, we directly calculate it on the down-sampled smoothed b-spline image. The local scale function is first calculated as

$$\sigma(\mathbf{y})^* = A_0 \mu(|I_2^*(\mathbf{y}) - \mu(I_2(\mathbf{y}))|) \tag{A.25}$$

where A_0 is a scaling constant set by the filter window sizes, w_i, and μ represents a box, or mean, filter on the image, and $I_2^*(\mathbf{y})$, is the smooth B-spline image $I^*(\mathbf{y})$ down sampled by a factor of 2 in each direction using averaging. The box filter size in each direction w_i is set as,

$$w_i = 1 + \frac{4 * \sqrt{-2\log(0.01)\sigma_{PSFi}}}{h_i} \tag{A.26}$$

where d_i is the pixel sampling in the same units as σ_{PSF}. With the constant normalisation factor A_0, normalising the estimate to that of the local intensities, and set empirically as

$$A_0 = \frac{1}{0.02201r^3 - 0.146r^2 + 0.3521r - 0.09969} \tag{A.27}$$

where $r = \prod_i^n w_i/(1 + \frac{2*\sqrt{-\log(0.01)2*\sigma_{PSF}}}{h_i})$, where n is the dimension of the image. The scaling factors and size of arguments was set through use of synthetic benchmarking data. Then function is thresholded in the following way

$$\sigma(\mathbf{y}) = \begin{cases} \max(\sigma^*(\mathbf{y})), \sigma_{th}) & \sigma^*(\mathbf{y}) > \frac{\sigma_{th}}{2} \\ 64000 & otherwise \end{cases} \tag{A.28}$$

where σ_{th}, is set to the scale of the smallest content in the image that is wanted to be captured, e.g. the difference between the foreground and background of the dimmest object in the image. Otherwise, for noisy flat regions, as in image background, $\sigma \rightarrow 0$, resulting in the noise being captured (Figure A.4).

The box filters, are performed in a separable manner, in each direction using summed area tables [33], also known as integral images, to allow box filters computationally $\mathcal{O}(1)$ with respect to the window size.

For real exemplars, an intensity threshold was also included, to allow the exclusion of unwanted dim image content in the background of the image or image camera defects. At this step any additional information, or filters, could be used to determine which part of the image wish to be captured, this could include information from different channels in an image, or from, different time-steps, such additions are use-case specific, but could yield a significant reduction in particle numbers.

Local Resolution Estimate $L(y)$ and Local Particle Set $\mathcal{L}$

The gradient magnitude $|\nabla I|$ and local scale function $\sigma(\mathbf{y})$ are then combined to create the local resolution estimate $L(\mathbf{y}) = \frac{E\sigma(\mathbf{y})}{|\nabla I|}$ and then the local particle set $\mathcal{L}_n$.

When using the equivalence $L(\mathbf{y})$, is only required at locations that align with a down-sampled by two image (the maximum of the gradient is used over the patch at the original resolution).

First we must set the relation between the particle cells and the image domain. Given an image with N pixels, and image dimensions N_x,N_y and N_z, such that $N = N_x * N_y * N_z$, with maximum dimension N_{max}, and minimum dimension N_{min}. We can then set the minimum and maximum levels l for the APR as $l_{max} = ceil(\log_2 N_{max})$ and $l_{min} = \max(2, l_{max} - \lfloor \log_2 N_{min} \rfloor)$, and then our augmented domain length $|\Omega^*| = 2^{l_{max}} h_{min}$, set such that the maximum resolution coincides with the original pixel sampling.

Then for each down-sampled pixel value we calculate, the level l, of the particle cell $c_{\mathbf{i},l}$ it belongs to as

$$l = \max(l_{min}, \lfloor \log_2(\frac{|\Omega^*|}{L(y)}). \rfloor)) \tag{A.29}$$

The spatial co-ordinates $\mathbf{i}$ of the particle could also be calculated as $\mathbf{i} = \{\lfloor x\frac{2^l}{|\Omega^*|} \rfloor, \lfloor y\frac{2^l}{|\Omega^*|} \rfloor, \lfloor z\frac{2^l}{|\Omega^*|} \rfloor\}$. We wish to find all the unique $c_{\mathbf{i},l}$ that then form $\mathcal{L}$. However, instead of directly computing the co-ordinates, we make use of the relationship between the levels of the tree structure and parent child relationships. Each full level l of the Particle Cells corresponds to a down-sampled version of the full image a fixed number of times, forming a classic image pyramid. Therefore, the image containing l for each downsampled pixel, is simply down-sampled again using the max operation, if any of the down-sampled levels correspond to the current level l, at co-ordinates x_l,y_l and z_l, then $c_{\{x_l,y_l,z_l\},l} \in \mathcal{L}$, being then indicated as occupied in a binary image pyramid representing $\mathcal{L}_n$. Although this down-sampling results in missing $c_{\mathbf{i},l}$ that have higher level children in $\mathcal{L}_n$ however we know these particle cells can not be in $\mathcal{V}$, and therefore cannot effect the pulling scheme result (the redundancy property). This algorithm allows the construction of this reduced $\mathcal{L}_n$ in $\mathcal{O}(N)$.

Pulling Scheme $\mathcal{V}$

Given the Local Particle Cell set $\mathcal{L}$, that is stored in an image pyramid with non-zeros indicated the present Particle Cells, we then run the Pulling Scheme using the full explicit storage of the Particle Cell tree as described in 4.2.3 using Algorithm 10.

The output of the algorithm is also stored in an image pyramid structure, with non-zero values storing the Particle Cell type.

```
Data: Local Particle Cell set $\mathcal{L}$
Result: Optimal Valid Particle Cell set $\mathcal{V}$
Function pulling_scheme($\mathcal{L}$)
    $level_c = level_{max}$
    /* Loop over the resolution levels, from finest to
       coarsest (from the maximum level $l$)                  */
    while $level_c >= level_{min}$ do
        if $level_c != level_{max}$ then
            set_ascendant_neighbours($level_c$);
            set_fillers($level_c$);
        fill_boundary($level_c$);
        $level_c --$  Level done, move to next;
    end
    $\mathcal{V} \leftarrow \mathcal{L}$
    return $\mathcal{V}$
```

Algorithm 10: Generating a Optimal Valid Particle Cell set $\mathcal{V}$ from Local Particle Cell set $\mathcal{L}$

```
Function fill_boundary($level_c$)
    foreach cell in $\mathcal{L}[: level_c]$ do
        if cell.type ∈ {SEED, PROPOGATE} then
            foreach neighbour of cell do
                if neighbour.type == EMPTY then
                    neighbour.type ← BOUNDARY
            end
            cell.parent.type ← ASCENDANT
        else if cell.type == PARENT then
            cell.parent.type ← ASCENDANT
    end
```

Algorithm 11: Filling BOUNDARY and ASCENDANT cells

```
Function set_ascendant_neighbours($level_c$)
    foreach cell in $\mathcal{L}[: level_c]$ do
        if cell.type == ASCENDANT then
            foreach neighbour of cell do
                if neighbour.type == EMPTY then
                    neighbour.type ← ASCENDANT_NEIGHBOUR
                else if neighbour.type == SEED then
                    neighbour.type ← PROPOGATE
            end
    end
```

Algorithm 12: Filling neighbors of ASCENDANT Particle Cells

```
Function set_fillers($level_c$)
    foreach cell in $\mathcal{L}[: level_c]$ do
        if cell.type ∈ {ASCENDANT_NEIGHBOUR, PROPOGATE}
         then
            foreach child of cell do
                if child.type == EMPTY then
                    child.type ← FILLER
            end
    end
```

Algorithm 13: Add FILLER Particle Cells

Particles $\mathcal{P}$

Now we have the valid particle cell set, $\mathcal{V}_n$, we then choose our particle sampling. This is done, as described in 5.5.3, where particle cells of type boundary and filler, have one particle placed in the center, and particle cells of type seed, are split into 8, higher resolution particle cells with one particle again at the center of each. The highest resolution particles coincide with the original image sampling, resulting in the construction of the particle sampling $\mathcal{P}^*$.

Intensity estimation

Any method of estimation of the particle intensities $I_p = I(\mathbf{x}_p)$, could be utilized at this step. In the case of noise-free images, the closest, or interpolated pixel value would be appropriate. However, in the presence of noise, the use of information from $\mathcal{V}$, to improve the estimate of the intensity using an area of the original image would be more appropriate with $I_p = \hat{I}(\mathbf{x}_p)$, where $\hat{I}$ is some de-noised image. Here, given each particle is sampled at the center of the particle cell, we simply take the average of the intensity over all pixels within the particle cell. In the case of particles at the image resolution, this average would simply be the original pixel. For these particles we then took the result of a median filter applied separately over each image direction.

A.5 Data structures

With the above steps, we now have the valid particle cell set $\mathcal{V}$ and particles $\mathcal{P}^*$, that form the APR. The last step is to store this information in memory in a data-structure that allows its efficient use in a wide range of tasks. The optimal data-structure will be dependent on the particular use-case, or algorithms, with which the APR is being used. Given the particle cells being a sub-set of a full oct-tree decomposition of the domain, for some tasks, a tree decomposition may be optimal. However, the majority of image processing algorithms have been designed to be implemented over pixel images stored as large contiguous arrays of pixels. This format has the advantage of fast and cache efficient local neighbor access, and the implicit coding of each pixels spatial coordinates from the pixel data layout, providing performance and memory benefits. Therefore, we have opted to use data-structures that attempt to mimic these efficient properties of pixel images, namely, the implicit coding of spatial and resolution information, and fast neighbor access. In this thesis we use two different data-structures that are detailed in Figure A.6 and Figure A.5. A $2D$ example is used for illustrative purposes with the only difference for the extension to $3D$

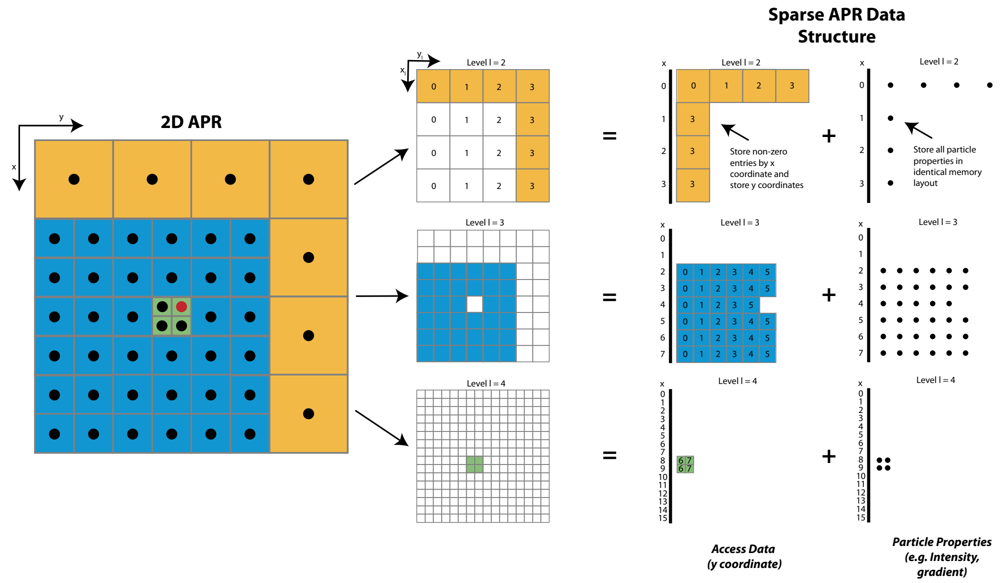

Sparse APR Data Structure
2D APR
y
x
Level l = 2
Level l = 3
Level l = 4
=
+
Store non-zero entries by x coordinate and store y coordinates
Store all particle properties in identical memory layout
Access Data (y coordinate)
Particle Properties (e.g. Intensity, gradient)

Figure A.5: Schematic showing the construction of the data structure, using a 2D APR example for illustration. The Sparse APR (SA) data structure contains an access structure, similar to Compressed Row Storage sparse matrix formats, used for determining the y spatial coordinate of the particle, the other spatial co-ordinates (x, z) and resolution level (l) are implicit in the data layout. Particle properties such as intensity are then stored using the same data layout. This structure does not allow for random access to particle neighbors. (The red colored particle is used in Figure A.6)

is the z direction is included and functions in the same way as the x direction shown (and for $1D$ only one row need be considered i.e. $x_l = 0$).

A.5.1 Sparse APR (SA) data structure

In the first data structure, called the Sparse APR (SA) data structure, each level of particles in the APR is stored separately, with the particle cell co-ordinates used for each particles corresponding particle cell $c_{\mathbf{i},l}$. The data-structure is broken into two components, each following an identical memory layout. First, the particle properties structures, stores the particle intensities for each level in rows according to their x_l and z_l coordinates (shown far left Figure A.6). The second structure allows access to the y coordinate information this property is then stored separately and labeled access data, in this work an unsigned 16bit integer is used, allowing a maximum domain length up to 64000. Any additional particle properties can be stored in the same layout as the intensity data. The global coordinates $\mathbf{x}_p$ of the particle can be directly calculated from the particle cell information. Essentially, the first structure is a multi-level implementation classic sparse matrix coding as a list of lists.

Memory Cost

This structure allows for the implicit storage of the x and z co-ordinate information and level l, with the explicit storage of y. The memory cost (MC) of the access data structure is,

$$\begin{aligned}\text{MC Access Data} &= (\text{y coordinate cost per particle})N_p \\ &+ \frac{2^{2l_{max}}-1}{3}(\text{array overhead per row}) \\ &= 2N_p + 8\frac{2^{2l_{max}}-1}{3}\text{ Bytes}\end{aligned} \tag{A.30}$$

and for each particle property,

$$\begin{aligned}\text{MC Particle Property} = (&\text{particle property cost per particle})N_p + \\ &\frac{2^{2l_{max}}-1}{3}(\text{array overhead per row}).\end{aligned} \tag{A.31}$$

Therefore, for storing only the particle intensity in memory as a float, the cost per particle is at-least 50% more, then an image with $N = N_p$ pixels. In the limiting case where the whole image is represented at pixel resolution, the data-structure overhead would be simply the 16bit y-coordinate array (50%) plus array overhead.

For an APR with a specific computational ratio (CR) = (number of original pixels)/(number of particles), the Relative Memory Cost (RMC) to the original image would be

$$\begin{aligned}RMC &= (\text{size in memory of original image})/(\text{size in memory of SA}) \\ &= CR * (\text{memory cost of particle})/(\text{memory cost of pixel}) \\ &= CR * \frac{MD}{MD + 2 + \frac{8MO}{N_p}}\end{aligned} \tag{A.32}$$

where $O = (\text{array overhead per row})\frac{2^{2l_{max}-1}}{3}$ is the array overhead, M the number of particle properties stored, and D the number of bytes per particle per property. Therefore we can see that as the number of properties increase, this amortises the cost of the data-structure, with the relative memory cost approaching the CR.

This structure is suited for tasks that have regular data access patterns and only require particle property and spatial information. Face Neighbour access can be achieved by using this structure in a linear access manner, by rastering over particles in each row with iterators for the neighbor on the same level l, and lower resolution $l-1$ and above $l+1$. However, this structure cannot provide $\mathcal{O}(1)$ random neighbor access, and iterating over all neighbors can require significant overhead.

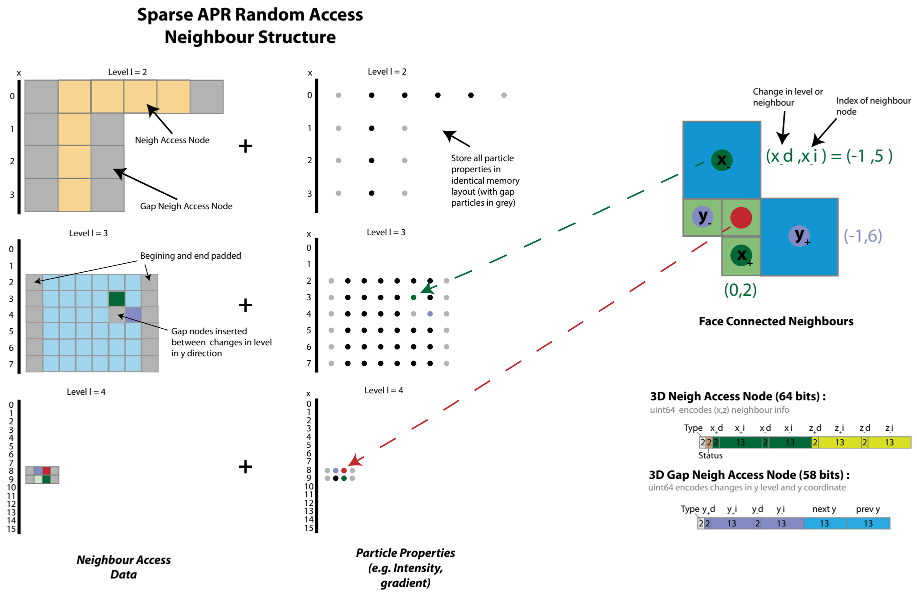

Sparse APR Random Access
Neighbour Structure
Level l = 2
Neigh Access Node
Gap Neigh Access Node
Store all particle properties in identical memory layout (with gap particles in grey)
Level l = 3
Begining and end padded
Gap nodes inserted between changes in level in y direction
Level l = 4
+
Neighbour Access Data
Particle Properties (e.g. Intensity, gradient)
Change in level or neighbour
Index of neighbour node
(x_d ,x_i) = (-1 ,5)
(-1,6)
(0,2)
Face Connected Neighbours
3D Neigh Access Node (64 bits) :
uint64 encodes (x,z) neighbour info
Type x+d x+i x-d x-i z+d z+i z-d z-i
Status
3D Gap Neigh Access Node (58 bits) :
uint64 encodes changes in y level and y coordinate
Type y+d y+i y-d y-i next y prev y

Figure A.6: Schematic showing the construction of the Sparse APR Random Access (SARA) neighbor data structure, using the 2D APR example from Figure A.5.1. As for the Sparse APR (SA) data structure, the data structure contains neighbor access data and particle properties with identical memory layout. The neighbor access data includes both neighbor access nodes, and gap neighbor access nodes, these store the changes in level, and index location of the face neighbors in each direction. An example of this for the red particle in Figure A.5 is illustrated on the left side of the figure, with the different face connected neighbors colored for illustration. The gap node, in addition to the y direction neighbor information, also encodes the jump in y coordinate. In 3D both these nodes are encoded into unsigned 64-bit integers.

A.5.2 Sparse APR Random Access data structure

For a structure optimizing random neighbor access, we use the Sparse APR Random Access Neighbour (SARA) data structure, detailed in Figure A.6. This is similar to the Sparse APR Data Structure. However, the data-layout is spaced with gap nodes, when a jump in resolution occurs in the y-direction, and the neighbor access data structure stores information required to access the face neighbors of the current particle. The addition of these gap nodes pads the previous data structure, as seen by the gray squares on the left side of Figure A.6. The particles properties are then placed in the same data layout, with zero padding values.

An example is again given in $2D$ for the neighbors of the red particle indicated in Figure A.5. Because neighbors can only be at most one level below, or above, the structure, we can decompose any neighbor into, a change in level ($\{-1, 0, 1\}$), and an index in the appropriate row. For the face neighbor, or neighbors, in the negative x direction, the change in level is x_d and index x_i as shown in the example. For the four faces in the x and z directions, the change in index and level, are stored for each particle, as shown in Figure A.5 under 3D Neigh Access Node. For increases in resolution, only one of the four neighbors is directly accessed, then the remaining three from additional neighbor steps on the new resolution level. These are stored in one unsigned 64-bit integer, in addition to the type, of the particle cell the particle belongs to. Further, the first bits of the node indicate if it is a gap or regular node. Each index is encoded in 13 bits, allowing for a maximum original image dimension of 8192 for this structure. These values can then be accessed through appropriate bit mask and shift operations. Face neighbors in the y-direction are treated differently, this is because for neighbors in the same level they are in the same row, and therefore the index will simply be an offset by ± 1 depending on the direction. Therefore, if the next or previous node in the structure is a regular node, indicated by its first 2 bits, this is the y-neighbour in that direction. Otherwise, the node is a gap node, and the gap node encodes the appropriate level change and index for the neighbor. This gap node also encodes the last, and next y, co-ordinate for each jump. Boundary conditions are encoded in the type and depth offsets, using the additional bits available. Therefore, given, any particle, the neighbors can be constructed, although, the y coordinate is not known but would need to be accessed from the nearest gap node. However, if particles are accessed in row order, the y coordinate can be updated as particles are iterated using the gap node.

Memory cost

The memory cost for the SARA data structure is greater than the SA data structure. This comes through the higher memory cost of each node, and also the padding of the gap nodes. The memory cost is,

$$MC \text{ Neighbour Access Data} = (N_p + N_g)8 + \frac{2^{2l_{max}} - 1}{3} 8 \text{ Bytes} \tag{A.33}$$

where N_g is the number of gap nodes in the structure, which is specific to each APR. For the CR benchmark datasets we found an average value of N_g of below $0.1N_p$. The particle property memory cost is then,

$$\begin{aligned} MC \text{ Particle Property} &= (\text{particle property cost per particle})(N_p + N_g) \\ &+ \frac{2^{2l_{max}} - 1}{3} 8 \text{ Bytes.} \end{aligned} \tag{A.34}$$

Therefore, relative to a comparable sized pixel image with $N = N_p$, and float data-type, the over-head would be over 200%. Therefore in the limiting case where all particles are at pixel resolution, $N_g = 0$, and the size is approximately 3 times the original image.

Again we can consider the relative memory cost (RMC) for a an APR with a certain CR. For the SARA data structure we have,

$$RMC = CR * \frac{MD}{MD + 8 + \frac{8MO}{N_p}}. \tag{A.35}$$

A.6 APR writing, output and compression

Lastly, for the pipeline, we need to be able to save and read the APR to disk. Further, we wish the APR file size to be as small as possible, i.e. highly compressed.

We use the HDF5 [124] file format for reading and writing files, using the BLOSC plugin [7] for compression that utilizes multithreading. For BLOSC, the Zstd compression algorithm is used with compression level 6 and shuffling activated.

When stored the APR is split into two parts; the particle cell information, and the particle properties (e.g. Intensity). The particle cell information is stored by re-creating the pyramid representation of the valid particle cell set $\mathcal{V}_n$ that is output from the pushing scheme. This has levels from l_{min} to $l_{max} - 1$, and is an unsigned 8bit integer, with non-zeros giving the type of the particle cell. Each level is then stored in a block, due to the low entropy of the sequence, (0,1,2,3), this data is compressed efficiently. In this way, it implicitly stores the coordinate information, type, and resolution of all the particle cells. The particle properties are then stored as a contiguous array for each level l, with each row from the data structure concatenated together, regarding increasing x_l, then z_l coordinate.

As discussed in the main text, for the benchmark data we find that the Memory Compression Ratio (MCR) is approximately 1.8*CR, when storing only intensity as the particle property. Implying that the compressed size of an individual particle is just under 9bits per particle. Although, these results are just indicative of these results will depend on the original images properties.

A.7 Pipeline parameter Summary

The required parameters for the APR pipeline presented here can be grouped into two categories, those that reflect information on the properties of the original image, and those that impact how and what the APR represents in the image.

A.7.1 Image parameters

The main parameters required for the formation of the APR are the image dimensions $\{N_x, N_y, N_z\}$, the sampling distances in each direction, $\{h_x, h_y, h_z\}$, and the standard deviation of a Gaussian approximation to the point spread function (PSF) in each direction, $\sigma_{PSFx}, \sigma_{PSFy}, \sigma_{PSFz}$. The first two, are direct image properties that should be known for a given image, the third, the PSF properties must often be estimated from the data. We have empirically found the results are relatively insensitive to the exact value, however, including some degree of anisotropy between the x, y, and z axis that exists is still important. Although formally, the PSF standard deviations are in real units, if the sampling distances are normalized to a pixel, i.e. $h_x = h_y = 1$, this value can be given regarding pixels.

A.7.2 Reconstruction parameters

These parameters impact how the APR is formed from the underlying image and includes the relative error bound E, the gradient smoothing parameter λ, and the minimum local scale threshold σ_{th}.

Relative error E

The relative error bound E, determines the allowed distance between the original image, and the APR representation. As the discussion in the main text 6.4.2 a value in the range of $0.08 - 0.15$ seems optimal for across noise levels that coincide to typical fluorescence imaging. We have found this reflected in qualitative experience with real data-sets as found in the exemplar data sets. Further, we found that for highly anisotropically sampled datasets, a value in the lower range of $0.08 - .1$ was usually appropriate, likely compensating for the resolution loss in one direction, where as for more isotropic data sets, higher values in the range of $0.1 - 0.15$ seemed optimal. However, in all cases, the results are insensitive to the exact value.

Smoothing parameter λ

The gradient smoothing parameter λ controls the how much smoothing is done in the fitting of B-splines for local resolution function estimation. With a higher value resulting in greater smoothing. This smoothing is required due to the amplification of noise properties of standard gradient operators in the presence of noise [129]. The absence of this smoothing would result in erroneous high-gradients and over-sampling. How to set this parameter depends on the signal to noise ratio of the original image. With values ranging from 0.5 to

10, seemingly optimal over standard signal to noise ranges, with lower values for higher signal to noise ratios. Again, results are not especially sensitive to this result, with a value too low, resulting in over-sampling and likely fitting of noise, and a value set too high, resulting in the APR not adapting to the fine grain structure.

Local Intensity Scale threshold σ_{th}

Lastly, the minimum Local Intensity Scale threshold σ_{th} represents the minimum local scale that will be allowed across the image domain. This is required due to the behavior of the Local Intensity Scale $\sigma(\mathbf{y})$. In flat regions, the response tends towards the average noise range. Hence, given the gradient will be non-zero in these regions it will result in the fitting of the noise by the APR, due to the normalization of the small gradients as shown in the first pane of Figure A.4. σ_{th} is introduced to curb this effect. To reduce the impact of noise, the smoothed B-spline image is used as input to the local-scale function rather than the original. This results in a reduced response of flat background areas in the Local Intensity Scale, allowing for the threshold to still function at low signal to noise ratios. Further to a minimum bound, values at half this value are then set to a maximum response. This is in effect to drive the $L(\mathbf{y})$ to large values and result in the APR ignoring these regions.

Setting this value is therefore subjective, and image dependent, as real-images often contain dim signals from sample contamination or auto-fluorescence, that may or may not wish to be captured. Corresponding to b in our image formation model (2.3). If the faintest objects that wish to be captured can be identified in the image, the value can simply be set to the local range between background and foreground for this object. Unlike the other parameters, the setting of this parameter too high can result in significant changes in the properties of the APR, due to it resulting in signal effectively being ignored. Therefore, a conservative underestimate is suggested. For the exemplar benchmarks, this was set by simple visual inspection of the original image.

Acting similarly, an image intensity threshold can also be used, where the Local Scale Estimate is set to a maximal value where the intensity is below some level I_th. This was not used in the benchmark data, but has proved useful when dealing with real data. In a similar way, any additional information can be included into the APR in a similar manner, including information from other channels, prior knowledge, or extra image processing steps.

A.8 Synthetic image generation

In this section, we provide additional technical details regarding the generation of synthetic images used in this thesis.

A.8.1 Implementation

The synthetic image generation pipeline is implemented separately in a C++ library known as SynImageGen, with the code available on request and uses the ArrayFire [12] Library for GPU acceleration for generating the images. This, therefore, limits the size of generated images to GPU memory, only allowing images up to 1000^3 to be generated. In fact, the generation of synthetic images usually accounted for the largest component of benchmarking. The pipeline is designed such that the parameters describing each synthetic image are sufficient to recreate any of the images in the pipeline. Below we will describe additional technical details of the synthetic image generation.

A.8.2 Template image

The template images used here, are piecewise constant images, with objects of various intensity, size, and location placed within a fixed $3D$ image size. The process begins with the generation of a binary object template. This template is then used for multiple instances of the same object within the image domain. Objects templates can either be generated as is the case with the used sphere template. Alternatively, templates can be generated by a binarization of 3D polygon model files (vrml, obj) using binvox [79, 77]. This is the case with the more complicated octopus benchmark shown in Figure A.4 and used for results shown in Figure 6.14. This template was downloaded from 3Ds [1]. However, as mentioned, for all other benchmarks provided here the generated sphere was used here due to its computational efficiency, and simplicity.

Objects were placed using a uniform random distribution within the volume, as not to overlap with the boundary, this is to reduce the impact of choice of boundary conditions for the pipeline on the results. Object intensities were set again with a uniform random distribution with a minimum and maximum intensity value set.

A.8.3 Ground truth image

The ground truth image is then generated by convolving the image with a blur kernel and adding a fixed background intensity. The convolution was done using separable filtering using 1D Gaussians of set sigma σ_{PSFi} in each direction. This blurred image, then served as the *ground truth*, as it represents the fluoresce distribution that we wish would observe if it was not corrupted by noise.

A.8.4 Original (noisy) image

The last step of the synthetic pipeline is the corruption of the image by noise. In fluorescence imaging, the image can be corrupted by multiple different noise sources with different properties including components that have spatial structure [130]. However, here we only consider the case of Poisson, or shot, noise arising from statistical quantum fluctuations by using a Gaussian approximation. This approximation is done due to the high computational cost of generating numbers from a Poisson distribution and its good approximation by a Gaussian for the values used here [76]. For each pixel, the following noise process is used and drawn from

$$\hat{I}(\mathbf{y}) \sim N(I(\mathbf{y}), I(\mathbf{y})) \tag{A.36}$$

where $N(I(\mathbf{y}), I(\mathbf{y}))$ is the normal distribution with mean and variance equal to the image intensity.

A.8.5 Benchmark parameter selection

See A.7 for a detailed discussion of parameters, and their use. For the synthetic datasets, all image parameters are taken as known from the image generation process including the blur sizes. For the reconstruction, parameters are set in the as described below, unless explicitly varied as a parameter for the benchmark.

Relative error bound E

Default parameter set to $E = 0.1$, unless otherwise stated.

Minimum Local Intensity Scale threshold σ_{th}

Set to the minimum bound for the random distribution of template intensities. This is a limitation of the results here and objectively should be set instead by an automated method. However, there is a large range of values over which the results are insensitive for this parameter.

Gradient smoothing parameter λ To set this parameter in an automated fashion, we utilized the fact that we knew the minimum standard deviation of the noise σ_{noise} of the benchmark image, being the $\sqrt{I_b}$, where I_b is the constant background intensity. We then ran parameter searches across different noise levels σ_{noise}, and parameter values λ, to find the minimum λ required to be still able to get three levels of resolution change with an object with brightness above the background set at the minimum local scale threshold σ_{th}. We then used the symbolic curve fitting toolbox Eureqa [112] to fit the value λ given the input variables give us

$$\lambda = (\frac{\sigma_{th}}{\sigma_{noise}}0.498763)^{\frac{-1}{0.6161}} \tag{A.37}$$

which was used in the synthetic benchmarks, providing good results in both low and high PSNR benchmarks.

A.9 Evaluation benchmarks

In this section, we give details of the synthetic benchmarks used to evaluate the properties of the APR with the results presented in 6.4. For each data point in the benchmarks, a synthetic image is generated and used as input to produce an APR that is then reconstructed using the appropriate method (usually piecewise constant reconstruction) and summary and image statistics are calculated. This analysis is then saved in an HDF5 file that is then read, analyzed and plotted using Matlab. All scripts and data for the production of the plots in this paper are available on request. Further, to aid reproducibility, each file contains the git hash for the code commit used to produce the results (for the APR library), the command line input parameters, and an exhaustive list of parameters used to generate the analysis. The parameters for the synthetic image generation are either as stated explicitly below, or as outlined in A.8.

In all cases with error bars have been given, they reflect the estimate of the standard error.

A.9.1 Noise-free Reconstruction Condition

Parameter values for results presented in 6.4.1. Images of fixed size and number of objects are generated, and the required relative error bound E is varied. The Reconstruction Condition requires that the observed reconstruction error E^* is below E for all locations.

Parameters

An image size of 128^3 was used with five sphere templates randomly placed in the domain with brightness B_i varying uniformly between 500 and 5000 with a background intensity $b = 1000$. The blur kernels used has $PSF_i = \{0.1, 0.3, 0.6\}$, corresponding to the low, medium and high blur, with isotropic sampling as $h_x = h_z = h_y = 0.1$. The minimum local scale threshold $\sigma_t h = 500$. Due to the lack of noise, finite differences were used to approximate derivatives instead of smoothing B-splines. The relative error bound E, was run in two linear sections with 40 samples, from $0.001-0.1$, then from $0.1-1.0$, with 40 repeats for each relative error bound.

A.9.2 Reconstruction error (noisy)

The same benchmark as above was repeated but with the introduction of Poisson noise and are discussed in 6.4.2.

Parameters

The image parameters and settings were set as in the no-noise case above.

Image statistics

The average observed relative error for the noisy image was constructed by taking the average of the infinity norms of the individual original images.

A.9.3 Reconstruction image quality (noisy)

Also in 6.4.2 data was presented for the image quality, measured by the Peak Signal to Noise Ratio (PSNR) and how it, varies with E, for noisy images. We do this for original images with different initial image quality (PSNR), by varying the signal to noise ratio. Further, on the right axis, a comparison between the reconstruction error from the APR, and the noise level in the original image is given as measured by the Mean Squared Error (MSE).

Parameters

Different PSNR images were created by fixing the background intensity $I_b = 1000$, and varying the brightness of the sphere templates, giving an estimated SNR of $\frac{\sigma_{noise}}{I_{obj}}$, where I_{obj}, is the intensity of the original object template. Due to the Poisson noise corruption, the effective standard deviation of the noise level will be at-least $\sigma_{noise} = \sqrt{I_b} = \sqrt{1000}$. Therefore, we run the benchmark with

3 different object intensities $I_{obj} = \sqrt{1000}, 10\sqrt{1000}, 30\sqrt{1000}$, corresponding to the low, medium and high PSNR images respectively.

An image size of 128^3, was used with 5 sphere templates randomly placed in the domain, with intensities and background set as discussed above. The medium blur kernel ($PSF_i = 0.3$) was used and isotropic sampling with $h_x = h_z = h_y = 0.1$. The minimum local scale threshold $\sigma_t h$, was set to the object intensity set for the original image. The relative error bound E, was run in two linear sections with 40 samples, from $0.001 - 0.1$, then from $0.1 - 0.4$, with 10 repeats for each relative error bound.

Image statistics

To measure image quality we use both the PSNR, calculated as

$$PSNR = 10\log_{10}(\frac{64000}{MSE}) \tag{A.38}$$

where MSE is the mean squared error and is calculated as

$$MSE = \frac{1}{N^*}\sum_{\mathbf{y}\in\hat{\Omega}}(I^*(\mathbf{y}) - \bar{I}(\mathbf{y}))^2 \tag{A.39}$$

where I^* is the ground truth image, $\bar{I}$ is the image being compared (either the original image, or reconstructed image), and $\hat{\Omega}$ is the those pixels in the domain for which the local scale function is less then 60000. This effectively excludes the calculation of statistics of background areas in the image due to the action of the minimum local scale threshold σ_{th}.

A.9.4 Increasing information content

In this benchmark, discussed in 6.4.3, we assess how well the APR is adapting to the image content. This is done by increasing the number of objects in the image and comparing both the image quality and the number of particles, with results given for the same low, medium, high levels of image quality as for the reconstruction image quality benchmark.

Parameters

The number of sphere templates randomly placed in the image was increased from $1 - 100$ in steps of 4, with 5 repetitions.

An image size of 300^3, was used, with a blur kernel between the low and medium used ($PSF_i = 0.2$) and isotropic sampling with $h_x = h_z = h_y = 0.1$. With the object intensity and background set as for the reconstruction image quality benchmark. The relative reconstruction error was set to $E = 0.1$.

Image statistics

The ratio of the PSNR for the APR reconstructed image, PSNR(APR), and the PSNR for the original image PSNR(Original) is given, showing the relative image quality of the reconstruction to the original image. Computed as described for the reconstruction image quality benchmark.

A.9.5 Increasing image size

In the last evaluation benchmark, discussed in 6.4.4, we assess the impact of the original image size, by holding the number of objects fixed, and increasing the image dimensions.

Parameters

The benchmark was run at three different levels of information content, using 10, 50 and 200 sphere objects placed randomly within the image domain. For each level of information, the image size was increased from 50^3 to 1000^3 in steps of 50, with 5 repetitions.

A blur kernel with $PSF_i = .2$ was used and isotropic sampling with $h_x = h_z = h_y = 0.1$. The template intensity and σ_{th} were set as described from the medium PSNR original image.

A.9.6 Increase sampling

In this benchmark, the sphere template object was held constant and in a fixed position in the center of the image. The sampling resolution was then increased while keeping all other variables fixed in real terms. Relative error was set to $E = 0.12$, and the $PSF_i = 1.075$. The sampling h_i ranged from a minimum of 0.027 to a maximum value of 0.3583 that corresponded to 200 different image sizes ranging from 50^3 to 650^3. The smoothing parameter $\lambda = 20\left(\frac{h_i}{50}\right)^2$ was heuristically set.

A.10 Benchmark data

In this section, we give additional technical details regarding the CR and exemplar benchmark data.

A.10.1 Computational Ratio (CR) benchmark data

To assess the performance of the APR for different levels of image content, we choose three different CRs for our synthetic benchmark data and then vary the original image size N. Given the range of CRs observed in the real exemplar data below, we choose CRs of 5, 20, and 100, representing high, medium and low image content respectively. A CR of 5 is lower than all exemplar benchmark data we tested, and 20 is a below average value, and 100 represents a very low content image, relative to its size.

Parameters

To achieve a certain CR, the number of objects must be changed with the image size N. The number of objects in the benchmarks was set as $\frac{N}{33400CR}$, that was determined empirically. The actual CR will not exactly be the ratio, but the above formula was found to provide good results for images of a width greater than 200. The images were isotropically sampled, and medium blur and medium PSNR setting were used as described in the benchmark evaluation section. The relative error bound $E = 0.1$, the gradient smoothing parameter $\lambda = 3.098$, as set by the automated scheme.

A.10.2 Exemplar datasets

To assess the performance of the APR on real datasets, 19 fluorescent microscopy datasets were also benchmarked. The datasets are across a range of image sizes, labels, specimen, and microscopes. A summary of the datasets, their properties, are given in Table A.1 and the parameters used in Table A.2 to create the APR. Parameters were set by experience and inspection of the original image. As mentioned in the discussion of parameters (A.7), the parameter that can most greatly alter the result is the setting of the minimum local scale threshold σ_{th}, and the intensity threshold I_p. In all cases, I endeavored to be conservative, including all content we could consider relevant to try and give lower bounds on the CR. In particular, the presence of auto-fluorescence influences this decision, and given the ability of the user to discriminate auto-fluorescents in many cases significantly higher computational ratios could be achieved.

A.10.3 Performance benchmarks

All examples are intended as a proof of principle and to indicate performance. Further development would be required to make these algorithms usable to

Dataset	N_x	N_y	N_z	N	**Labelled**	**Specimen**	**Microscope**	**Source**
1	1920	1080	201	4.17E+08	Membrane	Phallusia	Custom LSFM	Hufnagel Lab, EMBL
2	3169	1097	181	6.29E+08	Vasculature	Zebrafish (*Danio rerio*)	Custom SPIM	Stephan Daetwyler MPI-CBG
3	3083	970	148	4.43E+08	Vasculature	Zebrafish (*Danio rerio*)	Custom SPIM	Stephan Daetwyler MPI-CBG
4	1824	834	809	1.23E+09	Nuclei	Fly (*Drosophila melanogaster*)	Zeiss Z.1	Tomancak Lab MPICBG
5	532	1352	11	7.77E+06	Nuclei	Unknown	Unknown	Myers Lab MPI-CBG
6	1920	1200	80	1.84E+08	Nuclei	Fly (*Drosophila melanogaster*)	Zeiss Z.1	Tomancak Lab MPICBG
7	1094	1162	637	8.10E+08	Nuclei	Zebrafish (*Danio rerio*)	Custom SPIM	Gopi Shah MPI-CBG
8	2354	972	39	8.97E+07	Nuclei	Mouse	Confocal	DZNE (Christopher Schmeid)
9	1000	1820	975	1.77E+09	Nuclei	Flour Beetle (*Tribolium castaneum*)	Zeiss Z.1	Akanshka Jain MPI-CBG
10	1920	1080	335	6.95E+08	Membrane	Phallusia	Custom LSFM	Hufnagel Lab, EMBL
11	1920	1120	178	3.83E+08	Nuclei	Fly (*Drosophila melanogaster*)	Zeiss Z.1	Tomancak Lab MPI-CBG
12	1000	1820	975	1.77E+09	Membrane	Flour Beetle (*Tribolium castaneum*)	Zeiss Z.1	Akanshka Jain MPI-CBG
13	960	960	548	5.05E+08	Vasculature	Zebrafish (*Danio rerio*)	Custom SPIM	Stephan Daetwyler MPI-CBG
14	1200	1920	95	2.19E+08	Nuclei	Zebrafish (*Danio rerio*)	Zeiss Z.1	Tomancak Lab MPI-CBG
15	3169	1097	181	6.29E+08	Vasculature	Zebrafish (*Danio rerio*)	Custom SPIM	Stephan Daetwyler MP-ICBG
16	1000	1820	975	1.77E+09	Nuclei	Flour Beetle (*Tribolium castaneum*)	Zeiss Z.1	Akanshka Jain MPI-CBG
17	3935	988	219	8.51E+08	Vasculature	Zebrafish (*Danio rerio*)	Custom SPIM	Stephan Daetwyler MPI-CBG
18	3736	1432	379	2.03E+09	Nuclei	Mouse	Confocal	DZNE (Christopher Schmeid)
19	1000	1820	975	1.77E+09	Membrane	Flour Beetle (*Tribolium castaneum*)	Zeiss Z.1	Akanshka Jain MPI-CBG

Table A.1: Description and source of the nineteen Exemplar benchmark datasets used in Chapter 6 and Chapter 7. Note the MPI-CBG is the Max Planck Institute of Molecular Cell Biology and Genetics, EMBL is the The European Molecular Biology Laboratory, and DZNE the Deutsches Zentrum fr Neurodegenerative Erkrankungen e.V.

Dataset	E	λ	I_{th}	σ_{TH}	PSF_{xy}	PSF_z	APR (GB)	Image (GB)	CR	APR MCR	Image Pbzip2 MCR	Pre-processing
1	0.1	2.3	120	30	2	8	0.014	0.83	28.7	58.8	3.7	
2	0.1	3.0	998	100	2	8	0.010	1.63	110.6	168.0	3.6	
3	0.1	4.0	50	60	2	8	0.013	0.89	28.6	67.1	3.0	
4	0.15	5.0	98	200	2	8	0.132	2.46	11.4	18.6	2.6	Deconvolution
5	0.1	3.0	40	50	2	8	0.013	0.16	5.6	11.7	4.7	
6	0.085	1.3	950	200	2	10	0.023	0.37	9.7	15.9	2.1	
7	0.15	6.0	110	15	2	4	0.050	4.02	38.0	80.9	8.4	
8	0.085	0.2		5	2	9	0.002	0.90	44.2	371.4	250.2	Enhanced
9	0.15	3.0	1500	500	2	4	0.088	3.55	30.2	40.4	2.7	Deconvolution
10	0.1	3.0		25	2	8	0.015	0.83	26.0	55.2	3.8	
11	0.15	2.0	300	50	2	8	0.026	0.77	16.5	29.6	2.5	
12	0.15	3.0	800	500	2	4	0.056	3.55	49.6	63.3	2.7	Deconvolution
13	0.1	20.0	1100	500	2	8	0.035	1.01	18.8	29.0	2.3	
14	0.1	2.0	250	80	2	8	0.014	0.44	14.8	31.7	2.7	
15	0.1	3.0	998	100	2	8	0.009	1.26	97.0	145.0	3.3	
16	0.15	3.0	300	150	2	4	0.070	3.55	26.8	50.4	3.1	Deconvolution
17	0.1	2.0	1030	200	2	8	0.030	1.70	35.9	57.1	2.7	
18	0.15	0.5	2	5	2	9	0.006	4.06	180.4	734.2	306.6	Enhanced
19	0.15	3.0	400	200	2	4	0.077	3.55	27.1	46.1	3.1	Deconvolution

Table A.2: Summary of parameters used and summary statistics for the nineteen Exemplar Benchmark datasets detailed in Table A.1 and used in Chapter 6 and Chapter 7. Note that for those images where preprocessing was undertaken, this was done prior to the recieving of the image by the source.

the community, and this is left for future research. An effort was made to optimize both the APR and pixel code in the same fashion to provide reasonable comparisons. The impact of optimization is also motivation for the simple nature of the examples presented here. Shared memory parallelism was used in all steps cases where it was easily achieved using OpenMP [20]. As for the evaluation benchmarks, all performance benchmarks code and analysis data is available on request and are intended to be released open source.

A.10.4 Neighbor access

A core operation in many image-processing algorithms is access to the values of neighboring pixels. For this benchmark, we contrast the time taken to access the values of all face-connected neighbors of each pixel, or particle. In a pixel image, excluding boundaries, there are always 6 face neighbors. When the pixel image is stored as a contiguous array, accessing these pixels is simply a fixed memory offset for each pixel. However, for the APR, due to the adaptive sampling, the number of face connected neighbors can vary per particles from 6 to 24. However, in practice, the average number of neighbors per particles in the $CR = 5$ benchmark data sets was 6.23. The complication for a particle is that its neighbor can be in either the same, higher or lower resolution than the current particle. How to address this issue and access neighbors on particles is described in the discussion of APR data structure in A.5.2. In both benchmarks used here the Sparse APR Random Access Data Structure (SARA) is used.

In this benchmark, for each pixel, and particle, the face-connected neighbor's intensities are summed and stored in with another pixel image, or APR data-structure. For performance, the order in which pixels, or particles, are accessed heavily impacts performance through the impact of the various caches of the processor. Therefore, here we run two benchmarks, one where the pixels, or particles, can be accessed in order, and the second where random pixels, or particles, are accessed with a random ordering. The first case we call the Linear Neighbour Access, and the second Random Neighbour Access benchmarks. The two examples represent the two extremes of neighbor access in image processing algorithms, with linear access on a pixel image corresponding to arguably optimal memory access patterns, and random access, the worst case.

Linear neighbor access

For the pixel linear benchmark, the pixels are iterated over in memory direction. For each pixel, the neighbors are looped over, again in memory direction, checking for boundary conditions, accumulating the value in a temporary variable that is then stored in a second array.

For the particle linear benchmark, the particles are iterated over, level by level, in memory direction of the SARA data-structure. For each particle the locations of neighbors for each face are first calculated using the SARA structure, then the neighbor's intensities are looped over and the value added to a temporary variable. This temporary variable is stored as an additional particle property. For both, the results were averaged over 10 consecutive runs.

Random neighbor access

For the pixel random benchmark, instead of iterating over pixels in memory direction, a pixel is chosen randomly from the dataset, and the neighbors are summed. For the particle random benchmark, similarly, a random particle is chosen, and then the neighbors are summed. In both cases, the overhead of generating the random numbers was attempted to be removed, by calculating the time separately of generating the random numbers. In the case of particles, this is complicated by the fact that sometimes a point in the data structure was selected that was not a particle. The extra points were accounted for in the number generation compensation.

For the particle random benchmark, the number of particles drawn was equal to the number of particles in the APR, in the case of the pixel random benchmark 10000000 accesses were made. The results were corrected for the relative differences.

Memory overhead For the pixel benchmarks, the memory overhead is the original image and an array of the same size. For both, the used data-type of the images was float. Therefore the memory cost $MC = N8$ Bytes for an image with N pixels. For the particle benchmark the SARA data-structure, with float intensity plus an additional float particle property was used. The memory cost is as described in A.5.2 SARA section.

A.10.5 Separable pixel filtering

In the pixel separable filter, the 1D filter is convolved successively in each direction. This is done by iterating over the particles in memory direction, checking the boundary conditions, then looping over the neighbor offsets, multiplying by the coefficient and accumulating this in a temporary variable. The temporary variable is then assigned to the output image array.

In the particle separable filter, first, a $2D$ image slice is interpolated using piecewise constant interpolation, with the slice translated such that the filter operation could be done in the contiguous memory direction. Due to the placement of particles at intervals at powers of 2, only the highest resolution aligns with a pixel layer, with other being between the intersection of two layers. All particles that are either aligned with the slice, or intersect, the slice are iterated over, calculating the filter value through accumulating in a temporary variable as for the pixels, and then assigning this to the output particle property. In the case of when the particle intersects between two layers, the output is then the average of the two filter values.

For the benchmarks given here a large filter $1D$ constant filter stencil of length 21 was used. Relative performance results are relatively insensitive to the size of the filter. For the particle filter, each direction was repeated 10 times to get the timing values. However, for each benchmark image, the filter on pixels was only run once. This was due to the higher computational cost restricting a higher amount of repetitions. However, for each CR and N combination, at least 20 independent repetitions were performed.

Memory overhead

For the pixel benchmarks, the memory overhead is the original image and an output array of the same size. For both the used data-type of the images was float. Therefore the memory cost $MC = N8$ Bytes for an image with N pixels.

For the particle benchmarks, the computation required the SA data-structure, an additional particle property for the output, and an array for the temporary $2D$ image slice used. Therefore the memory cost is $MC = 6N_p + 16\frac{2^{2l_{max}-1}}{3} + 4N^{2/3}$ Bytes. If we ignore the over-head of the SA, and temporary array, the relative memory cost $RMC \approx 0.8 * CR$.

A.10.6 Graph cuts segmentation

For the last performance benchmark, we perform a binary segmentation with graph cuts using an external library. Here, we show how the APR can be used with existing techniques and libraries while still realizing computational and memory benefits due to the reduced number of computational points. For this, we use the maxflow-v3.04 library implementing the min cut-max flow algorithm presented in Boykov and Kolmogorov [21].

Further, we use an energy function that is defined using the information inherent in the APR, as an example of how it could be used. Because of this, then to compare with the same algorithm on the original pixel image, we first compute the energy on the particles and then interpolate them to original images to be then used for the energies for the pixel image.

To use the max-flow algorithm for segmentation, we must define two energy values for each pixel or particle, E_s, giving a likelihood of belonging to the foreground, and E_t, the likelihood of belonging to the background. Additionally, an energy is specified between neighboring pixels or particles in the graph. Here, we use again the face-connected pixel or particle neighborhoods, where we define a symmetric energy between the two neighbors p and p', as $E_{p,p'}$.

As discussed we first define the energy on particles, and then to form the graph for pixel image, we interpolate these energies to a pixel image. For the background and foreground energy, we use the following

$$E_s = 2000|I_p - I_p^{min}| \tag{A.40}$$

$$E_t = 2000|I_p - I_p^{max}| \tag{A.41}$$

where I_p^{min} and I_p^{max} are estimates of the local min and max scaled by the resolution of the particle. This is by treating the APR as a tree structure. Maximum and minimum values are propagated up the tree, taking the respective min or max of children values. The value is then averaged over the neighbors at each level in the tree. Then for each particle, the value in the tree k resolutions above is taken as the value for I^{min} or I^{max} respectively. The algorithm essentially creates an adaptive min or max. As the purpose of this benchmark is to focus on the computational and memory characteristics of the algorithm and not propose a new segmentation algorithm or energy, we do not go into further details here.

The edge energy between any two particles, or pixels, is taken as

$$E_{p,p'} = 100\frac{(s_p s_{p'})^2}{81}. \tag{A.42}$$

As discussed, once the energy has been computed over particles, those were then used to also create a pixel image with the same energies interpolated. Then the appropriate data structures for the max-flow algorithm were generated, the max-flow algorithm run, and the binary labeling of background or foreground extracted from the result.

In the benchmark performance analysis, we assess only the computational time, and memory cost, of the max-flow algorithm, and not the including the generation of the energy or setting up the graph.

Memory Cost

Due to the use of an external library, we estimated the memory cost by performing memory analysis on the code with different particle and pixel sizes and empirically evaluating the value. For the performing max-flow on the pixel graph, we found the memory cost $MC = 411.6N$ and for the particle graph $MC = 436N_p$. Providing very similar results, this is although the particles can have up to 24 neighbors, on average for the benchmark data sets there is only approximately 6.3.

Alternative energy for Exemplar datasets

For applications to exemplar data, we developed a slightly altered energy function. The background and foreground energy were altered by using one iteration of APR adaptive smoothing (See Section Below), on both the intensity and adaptive min and max. The edge energy between particles was changed to be asymmetric to the following

$$E_{p \to p'} = 100 \exp \frac{I_p - I_{p'}}{d(p,p')(I_p^{max} - I_p^{min})}, \tag{A.43}$$

where $d(p, p')$ is the distance between the two particles. We found that this energy appeared to give reasonable results across a wide range of the exemplar data-sets with no adjustment of parameters except an intensity threshold for removal of background objects. Hinting that the information gained in the APR allows for regularization of the problem that may help with designing future algorithms with stable parameters across a range of problems.

A.10.7 Adaptive APR Filters

Although smoothing and gradient operations are not well suited to the separable filtering approach shown earlier, a more natural approach for the APR is to define filters not over pixels as for traditional filtering, but over particles. With the filter coefficients acting on the particle neighbors. Using particle neighbors results in the filter adapting its neighborhood size across the domain to the resolution given in the APR.

One can then define adaptive gradient filters, similar to standard finite differences, with the coefficients adjusted for the distance between the particles. The gradient filter used here is $\{-\frac{1}{2h_-}, \frac{1}{2h_-} - \frac{1}{2h_+}, \frac{1}{2h_+}\}$, where h_+ and h_- are the distances between particles in the positive and negative directions respectively. In the case where a neighbor is of higher resolution, an average of the neighbor particles is used.

As a second example, we show benchmark results for an adaptive smoothing filter. As for the classic separable filters, each direction is filtered separately with a $1D$ filter $\{0.1, 0.8, 0.1\}$, and in succession with a 1D filter. In the case where the neighbor is of higher resolution, an average of the neighbor particles is used.

A.11 APR visualization examples

A.11.1 2D slice reconstruction

The reconstruction of 2D slices to be viewed individually can be done using interpolation of the APR to the individual slice. This is the technique that has been used for the APR reconstruction images shown in this text. The computational cost of this operation is cheap such that one could view the image as is typically done slice by slice in a stack, by reconstructing the new images in real-time as the slice is changed (on a 2013 laptop, reconstructing a 1000x1000 slice took approximately .002 seconds).

Memory overhead Memory cost for $2D$ slice reconstruction is the memory cost of the Sparse APR data structure (SA) plus the cost of storing the $2D$ slice. Viewing of a standard pixel image in software such as Fiji [109] requires the storage of the full image in memory.

A.11.2 Perspective ray-cast

A perspective ray-cast allows for the visualizing $3D$ content by constructing a $2D$ image by simulating rays that would be seen by an observer from a particular location. However, because an image volume is just intensity values, an algorithm must be specified for turning the intensities seen by each ray into an observed value. The most common algorithm is simply to take the maximum value along the ray, the basis of the maximum projection. This technique is used in current state of the art visualization software [98]. Here we have implemented a maximum intensity perspective ray-cast algorithm for the APR. For comparison, we also implemented the comparable algorithm for a pixel image.

Following we describe the principle of the pixel algorithm and then use this as a reference for the description of the APR algorithm. The pixel algorithm involved rastering over each pixel, then calculating which ray this pixel would intersect with, and then updating this ray with the value if it is greater than its current value. This is in contrast to the alternative approach where the image volume is traversed individually for each ray. Each ray corresponds then to a pixel in the final viewed image.

For the APR algorithm, the main difference is that we assign each particle to a ray corresponding to its level l, effectively creating an image view at each resolution level. Once all particles have been traversed, the maximum operation is then propagated between levels from lower resolutions to the highest resolution. Resulting in a final highest resolution image that is viewed. We find that the APR algorithm has moderate overhead, with a PP ratio of approximately 0.75, when compared to the pixel algorithm.

The algorithms compute different results, however as discussed in the main text, they produce in most cases perceptively identical results in normal contrast ranges.

Memory overhead When computing the ray-cast on a pixel image, the memory required is that of the original image and the ray-cast result. This memory cost is similar to that of the APR ray-cast. The APR ray-cast requires the Sparse APR data-structure, and the ray-cast result, plus the down-sampled by two results. For reasonably sized data-sets in both cases the memory cost is dominated by the original image, and APR data-structure respectively. Therefore, the memory reduction for a particular dataset will be approximately $CR/1.5$ (reflecting the cost of storing the y coordinates).

A.11.3 Direct particle rendering

Given that the extra information, and adaptive sampling, we can also visualise our dataset by directly rendering the particles of the APR.

Memory overhead

Ideally direct renderings memory overhead should only reflect the cost of the SA data-structure, however memory efficient algorithms for rendering have yet to be developed and depend on a GPU implementation and are thus left for future work. The current visualizations, a very memory inefficient, requiring the direct storage of the spatial coordinates as floats, in addition to the particle information.

Declaration

Erklärung entsprechend §5.5 der Promotionsordnung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Die Dissertation wurde im Zeitraum vom 22.07.2013 bis 11.08.2017 verfasst und von Prof. Dr. Ivo F. Sbalzarini, Chair of Scientific Computing for Systems Biology, Faculty of Computer Science, TU Dresden, betreut.

Meine Person betreffend erkläre ich hiermit, dass keine früheren erfolglosen Promotionsverfahren stattgefunden haben.

Ich erkenne die Promotionsordnung der Fakultät für Mathematik und Naturwissenschaften, Technische Universität Dresden an.

Datum Unterschrift