University of Wisconsin Milwaukee
UWM Digital Commons

Theses and Dissertations

December 2016

Radial Basis Functions: Biomedical Applications and Parallelization

Ke Liu
University of Wisconsin-Milwaukee

Follow this and additional works at: https://dc.uwm.edu/etd

Part of the Computer Sciences Commons

Recommended Citation

Liu, Ke, "Radial Basis Functions: Biomedical Applications and Parallelization" (2016). *Theses and Dissertations*. 1382.
https://dc.uwm.edu/etd/1382

This Dissertation is brought to you for free and open access by UWM Digital Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UWM Digital Commons. For more information, please contact open-access@uwm.edu.

RADIAL BASIS FUNCTIONS: BIOMEDICAL APPLICATIONS AND PARALLELIZATION

by

Ke Liu

A Dissertation Submitted in
Partial Fulfilment of the
Requirements for the Degree of

Doctor of Philosophy
in Engineering

at

The University of Wisconsin-Milwaukee

December 2016

ABSTRACT

RADIAL BASIS FUNCTIONS: BIOMEDICAL APPLICATIONS AND PARALLELIZATION

by

Ke Liu

The University of Wisconsin-Milwaukee, 2016
Under the Supervision of Professor Zeyun Yu

Radial basis function (RBF) is a real-valued function whose values depend only on the distances between an interpolation point and a set of user-specified points called centers. RBF interpolation is one of the primary methods to reconstruct functions from multi-dimensional scattered data. Its abilities to generalize arbitrary space dimensions and to provide spectral accuracy have made it particularly popular in different application areas, including but not limited to: finding numerical solutions of partial differential equations (PDEs), image processing, computer vision and graphics, deep learning and neural networks, etc.

The present thesis discusses three applications of RBF interpolation in biomedical engineering areas: (1) Calcium dynamics modeling, in which we numerically solve a set of PDEs by using meshless numerical methods and RBF-based interpolation techniques; (2) Image restoration and transformation, where an image is restored from its triangular mesh representation or transformed under translation, rotation, and scaling, etc. from its original form; (3) Porous structure design, in which the RBF interpolation used to reconstruct a 3D volume containing porous structures from a set of regularly or randomly placed points inside a user-provided surface shape. All these three applications have been investigated and their effectiveness has been supported with numerous experimental results. In particular, we innovatively utilize anisotropic distance metrics to define the distance in RBF interpolation and apply them to the aforementioned second and third applications, which show significant improvement in preserving image features or capturing connected porous structures over the isotropic distance-based RBF method.

Beside the algorithm designs and their applications in biomedical areas, we also explore several common parallelization techniques (including OpenMP and CUDA-based GPU programming) to accelerate the performance of the present algorithms. In particular, we analyze how parallel programming can help RBF interpolation to speed up the meshless PDE solver as well as image processing. While RBF has been widely used in various science and engineering fields, the current thesis is expected to trigger some more interest from computational scientists or students into this fast-growing area and specifically apply these techniques to biomedical problems such as the ones investigated in the present work.

© Copyright by Ke Liu, 2016
All Rights Reserved

TABLE OF CONTENTS

Chapter 1 Introduction 1

1.1 Radial Basis Function (RBF) 1

1.1.1 Definition 1

1.1.2 Applications 3

1.1.3 Pros and Cons 9

1.2 Parallel Computing 10

1.2.1 Architecture Perspective 10

1.2.2 Memory Perspective 11

1.2.3 Current Trends 14

1.3 Thesis Objectives 17

Chapter 2 Modeling of Calcium Dynamics 18

2.1 Introduction 18

2.2 Mathematical Models and Meshless Numerical Methods 19

2.2.1 Governing Equations 19

2.2.2 Geometric Model Considered 20

2.2.3 Space Discretization (LRBFCM) 21

2.3 Results and Discussion 23

Chapter 3 Anisotropic Image Restoration and Transformation 25

3.1 Introduction 25

3.2 Anisotropic Image Restoration 31

3.2.1 Adaptive Mesh Generation from Images 32

3.2.2 Radial Basis Function (RBF) Interpolation 34

3.2.3 Anisotropic Radial Basis Function (ARBF) Interpolation 36

3.2.4 Algorithms 37

3.3 Anisotropic Image Transformations 39

3.3.1 Isotropic Image Transformations 39

3.3.2 Anisotropic Image Scaling 41

3.4 Results and Discussion 43

3.5 Conclusions 46

Chapter 4 Porous Structure Design in Tissue Engineering 53

4.1 Introduction 53

4.2 Prior Works 55

4.3 Methods 58

4.3.1 Radial Basis Function (RBF) Based Construction 58

4.3.2 Anisotropic Radial Basis Function (ARBF) Interpolation 59

4.3.3 Algorithms 61

4.4 Results and Discussion 62

4.5 Conclusions 66

Chapter 5 Parallelization 68

5.1 Introduction to GPGPU 68

5.2 Parallelization of Modeling of Calcium Dynamics 68

5.2.1 Parallelization with OpenMP 69

5.2.2 Parallelization with GPU 69

5.2.3 Experiment Environment 73

5.2.4 Results and Discussion 73

5.3 Parallelization of Image Restoration with GPU 76

5.3.1 Methods 76

5.3.2 Results and Discussion 76

Chapter 6 Conclusions and Future Works 79

6.1 Thesis Summary 79

6.2 Future Directions 79

6.2.1 Applications of Radial Basis Function Networks 79

6.2.2 Porous Structure 79

BIBLIOGRAPHY 81

APPENDEX: PUBLICATIONS 104

CURRICULUM VITAE 105

LIST OF FIGURES

1.1 Figure 1 Gaussian basis function. (a) Shape of Gaussian basis function in 1D. c is the center. (b) Shape of Gaussian basis function in 2D..2

1.1 Figure 2 RBF support domains. (a) Compact support. (b) Non-compact support................3

1.1 Figure 3 Discretization using meshless method: nodes, domains of influence (in circle shape). Blue line illustrates problem domain. Red dots illustrate nodes. Grey circle illustrates the influence of local domain Ω_I.(Courtesy of [38])...4

1.1 Figure 4 Surface reconstruction from point cloud. Left is the point cloud as input. Right is the reconstructed surface. (Courtesy of [45])..7

1.1 Figure 5 The traditional radial basis function network. $\boldsymbol{x} = \{x_i\}_{i=1}^{n}$ is the input. $\boldsymbol{h} = \{h_j\}_{j=1}^{m}$ are basis functions. The output $f(\boldsymbol{x})$ is the linear combination of weights $\boldsymbol{w} = \{w_j\}_{j=1}^{m}$. (Courtesy of [46])..8

1.2 Figure 6 Illustration of instruction-level parallelism. (a) 5-stage pipeline RISC processor. (b) 5-stage pipelined superscalar processor. (Courtesy of [58])..11

1.2 Figure 7 Different types of memory models of parallel computing. (a) Execution sequence of shared memory model. Different colors illustrates different threads. (b) Shared memory model. (c) Distributed memory model. (d) Execution sequence of one of heterogeneous memory models. (Courtesy of [59] [60])..13

2.2 Figure 8 (a) The model considered in current study, containing the t-tubule (blue), surrounding half sarcomere (red), and external cell membrane (green). (unit: μm). (b) Flow chart of the meshless algorithm for modeling of calcium dynamics.................................22

2.3 Figure 9 Results of calcium dynamics. Vertical coordinate illustrates concentrations in μM. (a) Concentration of Ca over time. (b) Concentration of Fluo over time. (c) Concentration of ATP over time. (d) Concentration of Cal over time. (e) Concentration of TN over time.24

3.2 Figure 10 Example of interpolation. (a) Interpolation by vertices. Green dots are vertices defined on feature. Blue dots are vertices defined on feature edge. (b) Interpolation by faces. Green dots are face centers. Blue dots are face centers used for interpolation of the intensities of pixels enclosed by the blue triangle..34

3.2 Figure 11 Interpolation schemes. (a) Isotropic RBF interpolation. (b) Anisotropic RBF interpolation. (c) Eigenvectors on an edge pixel. e_1 shows the normal direction. e_2 shows the tangent direction..35

3.3 Figure 12 Resampling of original image ($M \times N$) to target image ($X \times Y$). As the gray triangle shows, every triangle in target image keeps the shape unchanged but has larger sampling density..41

3.4 Figure 13 Summary of restoration of Lena. (a) Original Lena image. (b) Result of piecewise interpolation. (c) Result of vertex-based iso-RBF interpolation. (d) Result of triangle-based iso-RBF interpolation. (e) Result of triangle-based ARBF interpolation using MQ basis. (f) Result of triangle-based ARBF interpolation using IMQ basis..44

3.4 Figure 14 Details of Lena. (a) Original Lena image. (b) Generated mesh of (a). (c) Result of triangle-based ARBF interpolation using MQ basis. (d) - (f) are zoomed-in views of (a) - (c), respectively......47

3.4 Figure 15 Details of brain MRI. (a) Original brain MRI. (b) Generated mesh of (a). (c) Result of triangle-based ARBF interpolation using MQ basis. (d) - (f) are zoomed-in views of (a) - (c), respectively......48

3.4 Figure 16 Details of breast MRI. (a) Original breast MRI. (b) Generated mesh of (a). (c) Result of triangle-based ARBF interpolation using MQ basis. (d) - (f) are zoomed-in views of (a) - (c), respectively......49

3.4 Figure 17 Details of CT-scanned image of heart. (a) Original heart image. (b) Generated mesh of (a). (c) Result of triangle-based ARBF interpolation using MQ basis. (d) - (f) are zoomed-in views of (a) - (c), respectively......50

3.4 Figure 18 Isotropic image translation and rotation. After transformation, image dimensions are unchanged. (a) Image translated to another coordinates. Three red circles indicates image features before and after translation. (b) Left: original image. Right: image rotated 45°......51

3.4 Figure 19 Super-resolution results of biomedical images. Results are rescaled for display. Red boxes show the zoom-in areas. (a) is the original brain MRI. (b) is the zoom-in part of (a). (c) is ARBF-5X SR result and (d) is bicubic-5X SR result. (e) is the original CT-scanned heart artery image. (f) is the zoom-in part of (e). (g) is ARBF-5X SR result. (h) is bicubic-5X SR result. (i) is the original breast MRI. (j) is zoom-in part of (i). (k) is ARBF-5X SR result. (l) is bicubic-5X SR result......52

4.3 Figure 20 Values assigned to mesh nodes. Red dots represent value of 1. Blue dots represent value of -1. In 3D meshes, interior dots are represented by lighter colors. (a) Sample 2D triangle mesh. Note both edge and triangle centers are given values -1. (b) Sample 3D tetrahedron mesh. Small triangle represent tetrahedron centers. (c) Sample 3D hexahedron mesh. Small triangle represent hexahedron centers.......................................58

4.3 Figure 21 2D interpolation schemes. x is the pixel to be interpolated. Dashed circle (for RBF) or ellipses (for anisotropic RBF) are support domains of underlying basis functions. (a) RBF interpolation. (b) Anisotropic RBF interpolation……………………………..59

4.3 Figure 22 Cases to calculate anisotropic distance. (a) Point x is on line segment (a,b). (b) Point x and line segment (a,b) form an acute triangle then distance is defined as the length of $\|xx'\|$. (c) Point x and line segment (a,b) form an obtuse triangle. (d) Distance between line segment (a,b) and (c,d)..60

4.4 Figure 23 Results based on tetrahedron meshes. (a) Scaffold based on single tetrahedron. (b) Input icosahedron mesh (formed by 20 tetrahedrons). (c) Scaffold using (b) as input...63

4.4 Figure 24 Results based on hexahedron meshes. (a) Scaffold based on single hexahedron. (b) Scaffold based on 8 hexahedrons arranged to form a large cube. (c) Scaffold based on 4 hexahedrons arranged to form a rod shape..63

4.4 Figure 25 Results taken by different iso-values. (a) - (d) are results taken by increasing iso-values...64

4.4 Figure 26 Results obtained by using different basis functions in ARBF interpolation. Shape parameter is 0.1 for all results. (a) Result interpolated by multiquadrics (MQ) basis. b)

Result interpolated by inverse multiquadrics (IMQ) basis. (c) Result interpolated by Gaussian basis. (d) Result interpolated by thin plate spline (TPS) basis............................65

4.4 Figure 27 Experimental results. (a) Result interpolated by isotropic RBF and based on a 2D triangle mesh. (b-d) Results based on hexahedron meshes with disturbance.................66

4.4 Figure 28 Porous structure obtained by TPMS method. (a) Structure obtained by P-type function. (b) Structure obtained by D-type function. (c) Structure obtained by G-type function. (d) Structure obtained by IWP-type function..67

5.2 Figure 29 Column 1 shows the average concentrations of Ca^{2+}, mobile and stationary buffers for point set 2 (30,807 points) in three implementations, namely, serial, OpenMP and CUDA. Note that the concentrations of the three versions are almost identical. Column 2 shows the average relative errors of point set 2 (30,807 points), as compared to the serial execution. Column 3 shows the average relative errors of point set 3 (234,921 points), as compared to the serial execution...75

LIST OF TABLES

3.3 Table 1 Summary of the Lena image (Figure 13)……………………………………………46

3.3 Table 2 Summary of the three medical images (Figure 15, Figure 16, Figure 17)………...46

4.1 Table 3 Category of methods to design porous scaffolds in tissue engineering..................54

4.3 Table 4 Type of distances used in porous structure construction.......................................61

5.2 Table 5 Running time of Serial, OpenMP, CUDA implementations on 3 point sets (unit: seconds)……………………………………………………………………………..76

5.3 Table 6 Execution time of image restoration. Results are the almost identical for block size of 128, 256 and 512...77

ACKNOWLEDGEMENTS

First, I would sincerely express my gratitude to my advisor, Professor Zeyun Yu, to the continuous support and guide to my Ph.D. study and research, for his patience, motivation, and vast knowledge. His insightful thoughts and advice helped me a lot during the time of research and writing of this thesis.

Besides my advisor, I would very much like to thank all my thesis committee members: Professor Guangwu Xu, Professor Tian Zhao, Professor Lei Wang, and Professor Roshan D'Souza, for their valuable comments and encouragement.

My sincere thank also goes to Jason Bacon, who provided me access to the research cluster at UW-Milwaukee, and gave me instructions to use the cluster. Without his precious support, it would not be possible to conduct this research.

Last but not least, I would like to thank my family for their spiritual support throughout writing my PhD thesis and my life in general.

Chapter 1 Introduction

1.1 Radial Basis Function (RBF)

1.1.1 Definition

Radial Basis Function (RBF) is a real-valued function whose value depends only on the distance from two points in multi-dimensional space. One of these points is called the center, which could be the origin or alternatively some other point in this space. RBF interpolation is one of the primary methods to analyze multi-dimensional scattered data. Its abilities to generalize arbitrary space dimensions and to provide spectral accuracy have made it particular popular in different types of applications. Some of the applications include function approximation, numerical solutions of partial differential equations, computer vision and neural networks, etc.

RBF is formally defined as $\phi(r) = \phi(\|r\|)$. Any function ϕ that satisfies the property $\phi(r) = \phi(\|r\|)$ is a radial function. Norm usually is defined as the Euclidean norm but other distance functions like taxicab metric or Łukaszyk–Karmowski metric are also used to define the norm in some applications. Let $\mathrm{r} = \|\boldsymbol{x} - \boldsymbol{x}_i\|$ ($\boldsymbol{x}_i$ is center), c be a constant called shape parameter (or free parameter in some literatures), commonly used types of radial basis functions include

- Gaussian: $\phi(r) = e^{-(cr)^2}$
- Multiquadric (MQ): $\phi(r) = \sqrt{r^2 + c^2}$
- Inverse multiquadric (IMQ): $\phi(r) = \frac{1}{r^2+c^2}$
- Thin plate spline (TPS): $\phi(r) = r^2 \ln(r)$

The shape parameter plays an important role for the accuracy but how to choose the value is still an open research topic. Most researchers choose the value by trial and error or some other ad hoc

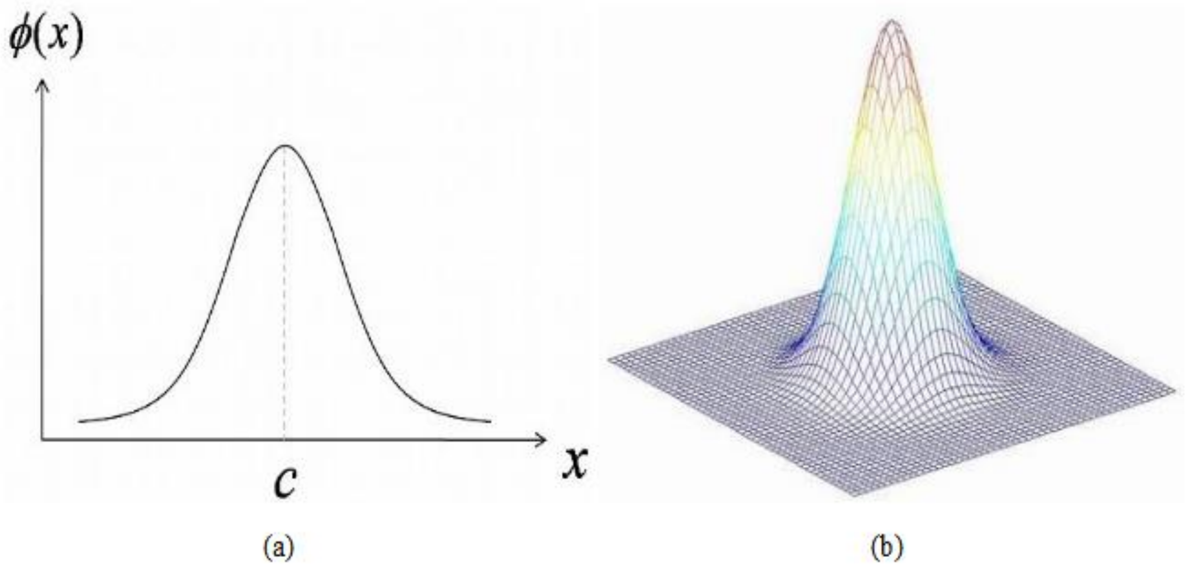


Figure 1 Gaussian basis function. (a) Shape of Gaussian basis function in 1D. c is the center. (b) Shape of Gaussian basis function in 2D.

means. Literatures [1] [2] [3] [4] [5] [6] [7]provides more details about choosing the value for shape parameter. Figure 1 shows the shape of Gaussian basis function. Every radial basis function has a support range, which is the footprint of the basis function. There are two types of support. Compact support or finite support: function value is zero outside of certain interval. Non-compact support or infinite support: there is no interval to limit the function values. Function value goes to infinite as the range goes to infinite. Figure 2 shows the two types of support domains.

RBFs are typically used to construct function approximations defined on scattered multi-dimensional data of the form

$$\mathrm{u}(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) \qquad (1)$$

where $\mathrm{u}(\boldsymbol{x})$ is the approximated function that represented as a weighted sum of N radial basis functions. Each basis function is associated with a different center $\boldsymbol{x}_{\mathrm{i}}$ and weight w_{i}. The weights

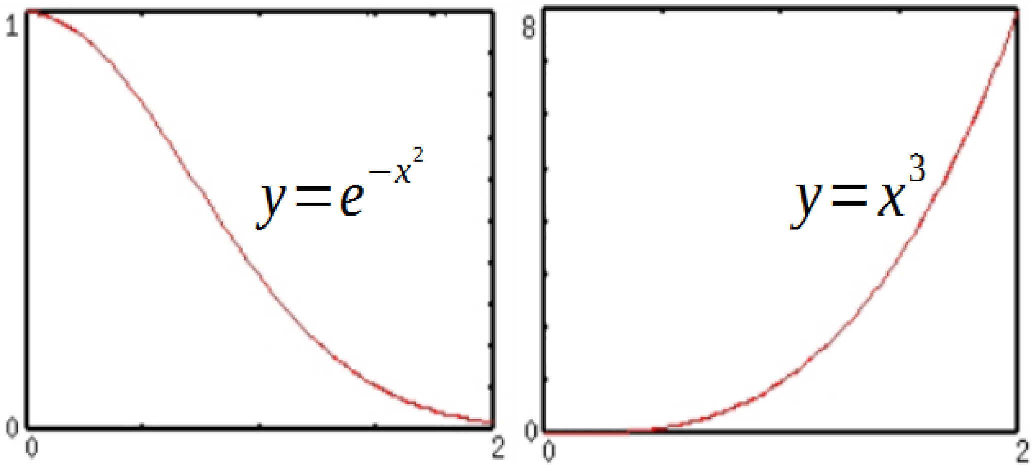


Figure 2 RBF support domains. (a) Compact support. (b) Non-compact support.

can be determined by solving linear equations. Let $u_i = u(x_i)$, by (1), the weights w_i can be solved by

$$\begin{bmatrix} \phi_{11} & \cdots & \phi_{1N} \\ \vdots & \ddots & \vdots \\ \phi_{N1} & \cdots & \phi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix} \qquad (2)$$

where $\phi_{ji} = \phi(\boldsymbol{x}_j - \boldsymbol{x}_i)$. Once the unknown weights w_i are solved, function values can be evaluated by (1). Besides by solving the linear equations in (2), the unknown weights w_i can also be solved by other matrix methods like linear least squares.

This approximation scheme is particularly useful in time series prediction, control of nonlinear systems having sufficiently simple chaotic behavior, and 3D reconstruction in computer graphics.

1.1.2 Applications

1.1.2.1 Numeric Simulation

Recent development of computer technology has made it possible to simulate a number of complex natural phenomena in experiments. In these experiments, partial differential equations (PDEs) with initial and boundary conditions are important tools to describe many mathematical models. For

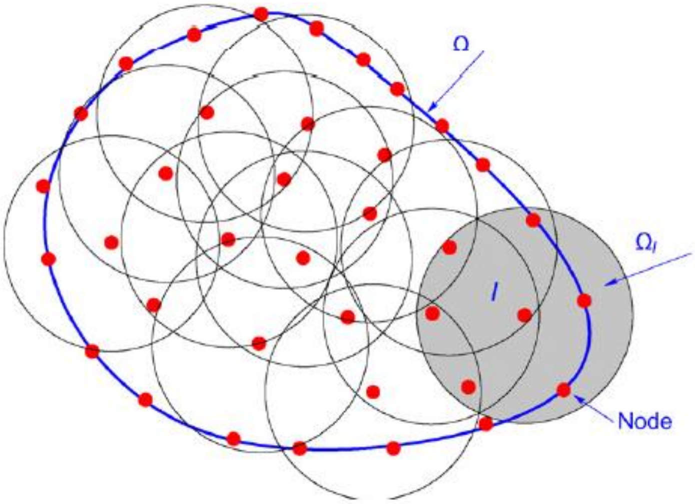


Figure 3 Discretization using meshless method: nodes, domains of influence (in circle shape). Blue line illustrates problem domain. Red dots illustrate nodes. Grey circle illustrates the influence of local domain Ω_I.(Courtesy of [38]).

complex PDEs, analytic solutions are usually too complex, even impossible to obtain. Therefore, numerical solutions as an approximation of analytical solutions are instead the goal to obtain in practice. Many numerical methods have been developed to solve PDEs. Some classical methods that solve PDEs numerically based on polynomial interpolation are finite difference method (FDM), finite element method (FEM), finite volume method (FVM) and pseudo-spectral methods. These methods solve a set of linear equations which are constructed after the analysis of the entire problem domain is analyzed and divided into elements or meshes.

Although these polynomial based methods are very effective to solve certain types of PDEs in many fields, they have limitations too, mostly because of the mesh-based interpolation. Distorted or low quality meshes lead to higher errors and necessitate remeshing, which is a task consuming both time and human labor and often is not guaranteed to be feasible in timely manner in complex 3D geometries. Moreover, the underlying mesh-based structure makes them not well suited to solve problems with discontinuity boundaries. A way to deal with discontinuities is remeshing or discontinuous enrichment. Alternatively, the extended finite element method (XFEM) [8] [9] [10]

[11] enriches the approximation space so that both strong and weak discontinuities can be captured. However, the involved difficulties of mesh-based interpolation are not only remeshing but also transiting problem states from old mesh to new mesh. Impact/penetration problem, explosion/fragmentation problem, flow pass obstacles problem, fluid-structure interaction problem and some biomedical simulations like the simulation and analysis of particular particles in cardiomyocytes during excitation and contraction are extremely difficult using the traditional methods introduced above.

RBF based methods, however, do not suffer from the adaptive remeshing procedures and the approximation is built from nodes only. Thus they belong to a category of methods called *meshless methods* or *meshfree methods*. Meshless methods are good at achieving exponential convergence rates on problems where traditional methods have difficulties or fail to solve. By constructing a univariate function with Euclidean norm, meshless methods turn a multi-dimensional problem into a one dimensional problem. One of the earliest meshless methods is the smooth particle hydrodynamics (SPH) method proposed by Lucy [12] and Gingold and Monaghan [13]. It was proposed to solve problems in astrophysics. Libersky et al. [14] were the first to apply SPH in solid mechanics. Other improved SPH methods are proposed as well [15] [16] [17] [18] [19]. In 1990s other weak form based methods were developed while SPH and their improved versions were based on strong form. One of the earliest meshless methods based on global weak form is the element-free Galerkin (EFG) method [20]. One year later, reproducing kernel particle method (RKPM) was developed in wavelets [21]. In contrast to EFG and RKPM methods which use intrinsic basis, other methods are developed to use extrinsic basis and the concept of partition of unity (PU). The extrinsic basis was used to increase the approximation order. Melenk and Babuska [22] proposed partition of unity finite element method (PUFEM) based on the similarity of meshless method and FEM. This method is very similar to hp-cloud method which employs elements of variable size (h) and polynomial degree (p).

All meshless methods introduced above are based on global weak form of PDEs. Another type of meshless methods are based on local weak forms. The most popular method of this type is the meshless local Petrov-Galerkin (MLPG) method [23]. The main difference between MLPG and global weak form based methods such EFG and RKPM is that local weak form is generated on overlapping local subdomains, on which the integration is carried out. Another well-known method which is called moving point method is mainly applied in fluid mechanics [24] [25] [26]. Because there is no mesh, tool like k-dimensional tree (KD-Tree) is usually used to divide the space and find neighboring nodes during the construction of linear systems. As a result, instead of solving a large linear system, many smaller linear systems are solved. By solving problems in collocation fashion, which is more flexible than global methods, local RBF methods can construct more stable linear systems and are easier to implement. Literatures [1] [27] [23] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] illustrate the efficacy and popularity of meshless methods in numerical areas. Giordano et al. proposes an RBF optimization method in hydrolysis area [39]. Figure 3 (courtesy of [38]) illustrates how meshless methods discretize problem domain. Red dots illustrate nodes whose values are used to approximate. Blue line illustrates problem boundary. Grey circle illustrates the influence of a subdomain.

1.1.2.2 Surface Reconstruction

RBFs find their usage in computer graphics fields as well such as surface or object reconstruction from point cloud, mesh repair, image registration, and field visualization in 2D or 3D, etc. Interpolating meshes with holes and reconstructing surfaces from point cloud are ubiquitous problems in computer graphics and computer aided design (CAD) areas. Because RBFs are

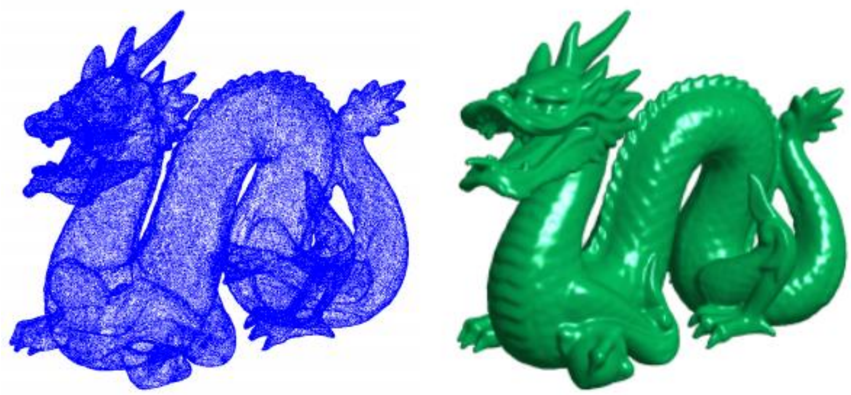


Figure 4 Surface reconstruction from point cloud. Left is the point cloud as input. Right is the reconstructed surface. (Courtesy of [45]).

polyharmonic and fast to be fitted and evaluated, smooth surface or object can be reconstructed from point cloud in large amount (in millions). The smooth blending of surfaces ensures a manifold can be constructed and thus manufacturable, which is related to many problems in CAD. Smoothing and remeshing existing noisy surfaces are also important problems in both computer graphics and CAD. These problems are considered independent problems in most cases and receive much attentions [40]. RBFs have been used to reconstruct surfaces by Carr et al. [41], Savchenko [42], Turk and O'Brien [43] [44]. These works are limited to small problems by their $O(N^2)$ storage and $O(N^3)$ arithmetic operations. By reducing the number of RBF centers used, a fast fitting and evaluation method is proposed by Carr et al. [45]. Figure 4 (courtesy of [45]) illustrates the problem of surface reconstruction from point cloud.

1.1.2.3 Artificial Neural Networks

RBFs are very useful in artificial neural networks as embodied in current versions of feedforward neural networks as well. A number of RBFs are organized together to form a network, called radial

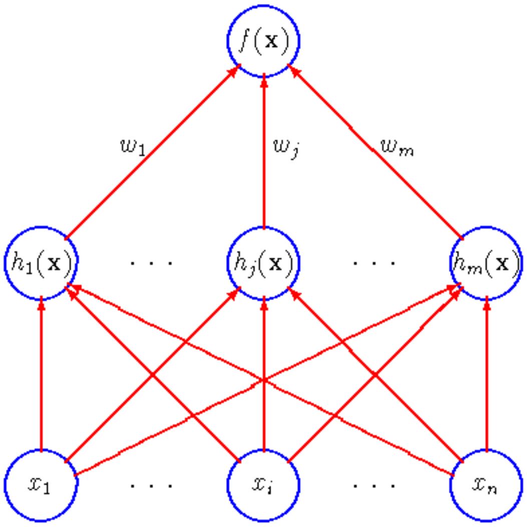


Figure 5 The traditional radial basis function network. $\boldsymbol{x} = \{x_i\}_{i=1}^{n}$ is the input. $\boldsymbol{h} = \{h_j\}_{j=1}^{m}$ are basis functions. The output $f(\boldsymbol{x})$ is the linear combination of weights $\boldsymbol{w} = \{w_j\}_{j=1}^{m}$. (Courtesy of [46]).

basis function network (RBFN). The traditional RBFN is illustrated in Figure 5 (courtesy of [46]). RBFN is mostly used for data forecasting, data mining and data classification in artificial intelligence area. RBFs in neural networks are explained in depth [47]. In 1980s, Broomhead and Lowe [48] are one of the earlies using networks to interpolate quantitatively. RBFs become popular in neural networks in 1990s. Girosi extend RBFs [49] and use them in artificial intelligence (AI) field. Varvark used elliptical RBFs to enhance neural networks [50]. Fernández-Navarro proposed a method using new basis funtion (q-Gaussian basis function) in binary classification [51]. Raitoharju et al proposed a method to train RBFN for classification [52]. Maglogiannis et al. used RBFN in classification and recognition in microscopic images [53]. Keramitsoglou et al. used RBFN to classify very high spatial resolution satellite image [54]. Sermpinis et al proposed a RBF-based method to forecast and optimize foreign exchange rates [55]. Sideratos and Hatziargyriou used RBFN in wind power forecasting [56]. Guo et al. proposed a forecasting model combing RBFN and 2D principal component analysis (PCA) in stock market [57]. Recently developed

RBFN algorithms incorporating compact-support RBF greatly increase the performance in training process.

1.1.3 Pros and Cons

The major advantages using RBF to solve PDEs (meshless methods) are 1) meshless methods have similar h-adaptivity compared to mesh-based methods, 2) it is much easier to handle problems with moving discontinuities such as crack propagation, shear bands and phase transformation, 3) more robust solution of problems with large deformation domains can be obtained, 4) higher-order basis function can be used to obtain smoother solutions, 5) solution can be interpolated globally or locally, which provides flexibility and 6) no cost of remeshing or mesh alignment. The chief drawbacks to the use of meshless methods primarily is related to the choice of shape parameters, which remains as an open research topic. Improper choice of shape parameter may produce unstable solution with large errors.

RBFs are particularly suited to fitting surfaces to non-uniformly distributed point clouds and partial meshed with irregular holes because of their independently polyharmonic smooth interpolation characteristics. Moreover, by using RBFs a fast fitting and evaluation approach can be applied to represent complicated objects of arbitrary topology with large data sets such as medical imaging and geophysical data. However, like any global models, RBFs have the drawbacks as well when manipulation of part of that model is required. In this case, the decomposition of a global RBF representation into a piecewise mesh of implicit surface patches is required.

Because RBFN has one hidden layer, which differs from multi-layer perception (MLP), RBFN is more robust in data prediction. RBFN have the advantages of easy design, good generalization, strong fault tolerance to noisy input data, and self-learning ability. These properties make RBFN an ideal tool in data forecasting, data classification, and machine learning. RBFN is able to produce smoother surface, more stable and provide better generalization comparing to traditional neural networks.

1.2 Parallel Computing

As RBF applications develop rapidly, more and more data is involved, which poses great challenge on computation. Traditional serial algorithms are no longer satisfactory for the high cost of computational time. A natural solution is using multiple processors, operating on the principle that large problems can be divided into smaller ones, which are then solved concurrently. Parallel computing can be classified from different views. The following sections describe the categories of parallel computing techniques in different perspectives.

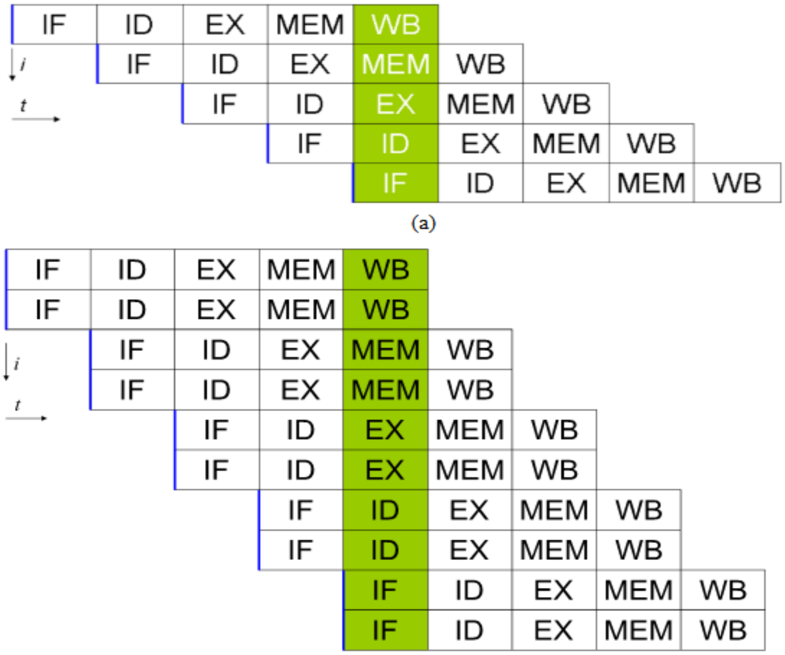
1.2.1 Architecture Perspective

In computer architecture perspective, from lower level to higher level, there are bit-level parallelism, instruction-level parallelism and task-level parallelism [58]. Bit-level parallelism relates to the computer word size. Increasing the word size (the length of bits a processor can fetch and process in single processor cycle) can reduce the number of processor cycles. The word size is increasing from 4-bit to 8-bit, 16-bit, 32-bit and now 64-bit. Instruction-level parallelism relates to reordering and combining instructions stages into groups. Modern processors uses multi-stage instruction pipelining to process multiple instructions at single stage and superscalar to issue multiple instructions at a time. A typical processor of this type is the RISC processor. Figure 6 shows a five-stage pipelining RISC processor and pipelined superscalar processor. Task-level parallelism decompose a task into sub-tasks and dispatches each sub-task to a processor for execution. Thus these sub-tasks are executed simultaneously. Some task-level parallel jobs need to exchange information among these sub-tasks to get final result. A commonly used task-level parallelism scheme is *map-reduce*. In this scheme, the original problem is first decomposed into many smaller sub-problems (the “map” step). Then solve them simultaneously. At last, all partial results will be combined to get the final result (the “reduce” step).

1.2.2 Memory Perspective

Another popular classification of parallel computing is known as *Flynn’s taxonomy*, created by Michael J. Flynn. This classification focuses on instruction and data, thus deriving four types. Single-instruction-single-data (SISD) programs are the same as sequential programs. Single-instruction-multiple-data (SIMD) programs perform the same operation on large dataset. Multiple-instruction-single-data (MISD) programs are uncommon and are mainly designed in redundant systems. Multiple-instruction-multiple-data (MIMD) programs can perform multiple operations on multiple datasets and are the most commonly used.

From the memory usage perspective, there are three popular parallel models. Shared-memory model uses single memory address space and usually is implemented by multi-threading



(a)

(b)

Figure 6 Illustration of instruction-level parallelism. (a) 5-stage pipeline RISC processor. (b) 5-stage pipelined superscalar processor. (Courtesy of [58]).

programming. Exchanging data is very easy since the memory is shared among processors. Distributed-memory model uses memory on multiple machines. Each machine has its own memory address space. Data exchange is achieved by sending and receiving messages. Heterogeneous memory model uses additional accelerator(s) called device(s) besides the CPU. The first accelerator is floating-point co-processor, which is designed for high-speed floating-point calculation. The most popular accelerator nowadays is the GPU. In this model, programs have to transfer data to the device(s), then start the device(s) for calculation, and finally get result from the device(s). Each model has its advantages and drawbacks. Shared-memory model is easier to implement and is more efficient comparing to distributed-memory model when the same number of processors are used. The main drawback of shared-memory is lack of scalability. Because the memory space is shared via system bus, all processors have to exist on a single machine. However, it is extremely difficult to continue shrinking the sizes of transistors and solve the heating problem while the chip is working to put more and more cores on a single chip, the number of cores on a chip is limited. In recent years, the number of cores on a single chip does not exceed 32. The distributed-memory model solves this problem by utilizing multiple machines. But, as the number of cores utilized is increasing, the message exchanging becomes more and more complicated so eventually, the cost of communication will dominate the overall computation time. Heterogeneous memory model takes advantage of dedicated, problem-specific device thus it has high computational efficiency. But the device causes additional program complication and incurs the cost of data transfer between the processors and device.

Figure 7 (courtesy of [59] [60]) illustrates these three memory models introduced above. Figure 7 (a) illustrates typical execution of shared memory model. Execution of program starts from master

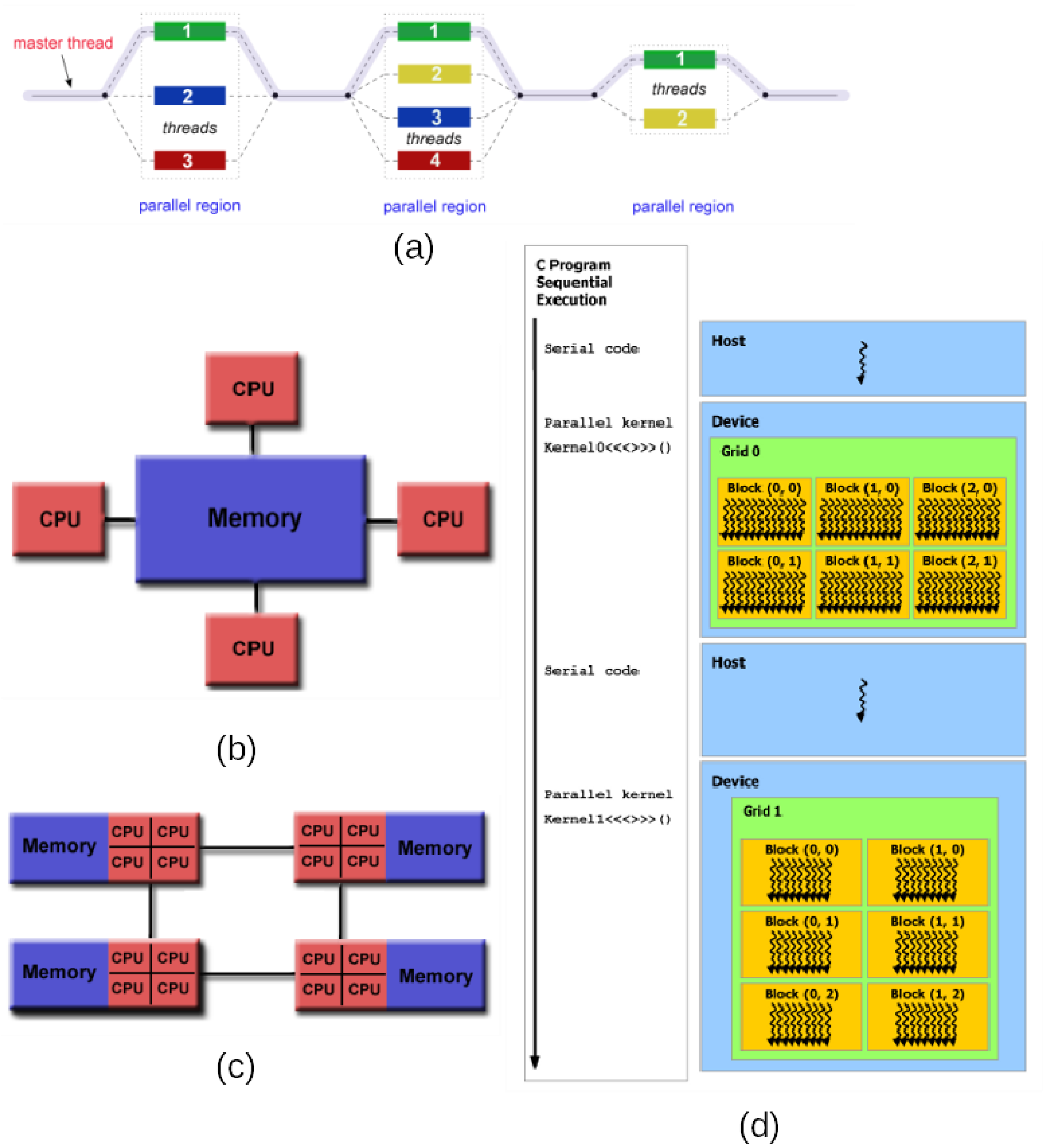


Figure 7 Different types of memory models of parallel computing. (a) Execution sequence of shared memory model. Different colors illustrates different threads. (b) Shared memory model. (c) Distributed memory model. (d) Execution sequence of one of heterogeneous memory models. (Courtesy of [59] [60]).

thread. Before the master thread enters a parallel region, the master thread spawns other threads. Inside parallel region, multiple threads execute in parallel. After parallel region, spawned threads have to be synchronized with master thread and then destroyed. Then the master thread is the only thread that is in execution until next parallel region is encountered. Figure 7 (b) illustrates a shared memory model with 4 processors. As introduced above, the memory address space is shared among the 4 processors. Figure 7 (c) illustrates a distributed memory model with 4 hosts inter-connected as a circular topology. Each host has an individual memory and 4 processors. Inside the host, the memory is shared with 4 processors. However, memory is inaccessible among hosts. If a host wants to access information in another host's memory, it has to send a message to the destination host. Figure 7 (d) illustrates the execution of one type of heterogeneous memory models, namely, compute unified device architecture (CUDA). Left of figure shows the execution. Parallel code is encapsulated in kernel functions which will be called when parallel execution points are reached. Right of figure illustrates how threads are grouped into blocks and grid. Details of CUDA will be introduced in Section 5.1.

1.2.3 Current Trends

Computing architectures evolved significantly in the last decade. There are many improvements happened across the whole spectrum of architectures ranging from individual processors to geographically distributed systems.

In the last decade, word size of a single processor has been increased from 16-bit to 64-bit in general processor. Some processors designed for special purposes such as gaming, video editing, encryption/decryption, the word size is even larger, usually ranges from 128-bit to 512-bit.

In 2002, Intel introduced the Hyper-threading (HT) technology [61], which enables a processor to store two architecture states at the same time in a single execution unit. When an execution instruction of a program is suspended for some reason, execution instruction of the other program can be executed. This technology makes a single physical execution unit appear to be two "logical" execution units. The Operating System can therefore see two virtual processors and schedule two independent threads at the same time. The execution speed is increased due to the more efficient use of the shared execution unit.

The improvements of lithography not only reduces the heating of processor, but also is able to miniaturize the chip size, which make it possible to put multiple execution units (called “cores”) on the same processor die. The multi-core processors can actually execute multiple instructions at the same time. In recent years, multi-core processors gradually become ubiquitous, being found on devices ranging from high-end servers to tablets and smartphones. CPUs with tens or hundreds of processors are already available [62]. Current desktop processors are often multi-core designed and combined with HT technology. These multi-core processors are homogeneous shared-memory architectures that all cores are identical to each other. Such system is also called symmetric multi-processor (SMP) system.

A natural thought of improving computational power is connecting multiple SMPs by low-latency networks. Processors of one SMP cannot access memories on other SMPs. In other word, memories are distributed across these SMPs. SMPs communicate with each other through sending and receiving messages. The distributed system is highly extensible.

Not all computing systems are symmetric. Asymmetric or heterogeneous multi-core systems also exist. An example is the Cell Broadband Engine [63], which contains two PowerPC cores and additional vector units called Synergistic Processing Elements (SPE). Each SPE has a programmable vector co-processor with a separate instruction set and local memory. Other widely used heterogeneous multi-core systems include general purpose graphic processing unit (GPGPU) [64]. The GPGPU is a massively parallel device that contains thousands of simple execution units connected to a shared memory. For this reason, the GPGPU is often called many-core processors. Because the RAM chips do not provide enough bandwidth to all cores at the maximum speed, the memory is organized in a complex hierarchy. GPGPU is usually manufactured as a separate GPU cards that can be inserted into a host computer and work under the control of host CPU. During execution, after host CPU launches and initializes the GPU, data is sent to the device memory on the GPU through buses. After computation, results are fetched back to host memory and the GPU is shutdown.

The hardware of modern multi-core processors are highly complex. This complexity cannot be ignored and instead it has to be carefully treated and exploited while programming to fully take advantages of new hardware features. Writing efficient parallel applications for multi-core and many-core processors requires detailed knowledge of the processor internals and proper coordination of communications and computations across available cores.

With software designed for distributed processing including distributed Operating System, task scheduling software, data persistence software, distributed file systems provided, geographically distributed systems can form a larger computing systems called Cloud Computing. Cloud computing is a model that enables on-demand remote access to a shared pool of resources. Different cloud service models are identified by the type of resources provided. In software as a service (SaaS) clouds, users access application services running in the cloud infrastructure. "Google Apps" is an example of SaaS cloud. A platform as a service (PaaS) clouds provide tools such programming languages and libraries to develop programs and a hosting environment for applications developed by cloud users. AppEngine provided by Google, Azure provided by Microsoft, and Elastic Beanstalk provided by Amazon are examples of PaaS cloud. Infrastructure as a service (IaaS) clouds provide low-level computing services such as processing, storage, and networks that users can run any applications including Operating Systems. Amazon Web Service is an example of IaaS. Details of distributed and cloud computing services and their limitations can be found in [65]

1.3 Thesis Objectives

This project has two major objectives:

- Discuss three applications of RBFs. In this work, the following applications are discussed and implemented:
    - Modeling of calcium dynamics in in cardiac myocytes. This application needs to solve a system of nonlinear partial differential equations (PDEs) over a time series. This is accomplished by RBF interpolation.
    - Image restoration and transformation. In this application, a variation of RBF interpolation called anisotropic RBF interpolation is used to preserve image features while restoring image from triangular mesh. Isotropic image translation, rotation and anisotropic image upscaling (also called image super-resolution) are researched and implemented.
    - Porous structure construction. In this application, a porous structure is constructed from volumetric meshes (tetrahedron or hexahedron mesh). Firstly, anisotropic RBF is used to interpolate internal voxels of structure. Secondly, an iso-surface is taken to obtain the final structure.
- Investigate parallelization of calcium dynamics application using a shared memory model (OpenMP), and a heterogeneous memory model (CUDA). Explore different parallel optimization techniques to reach the peak performance as close as possible. Investigate CUDA parallelization of image restoration application. In both applications, factors that affect performance as well as pros and cons are discussed.

Chapter 2 Modeling of Calcium Dynamics

2.1 Introduction

Heart failure has been one of the leading causes of human deaths in many countries including the United States. The prevalence of this disease is largely due to lack of accurate understanding of excitation-contraction (E-C) coupling in cardiomyocytes [66] [67] [68]. For its central role in E-C coupling, modeling Ca^{2+} release and concentration change has been an active research area. This chapter investigates spatial-temporal variations of intra-cellular calcium concentration at cellular and sub-cellular levels. At these scales, deterministic methods utilizing partial differential equations (PDEs) are more appropriate than stochastic methods [69] [70]. The local radial basis function collocation method (LRBFCM) developed by Sarler and Vertnik [5] has been applied to solving the PDEs in earlier work [1]. This meshless method eliminates the generation of meshes, as commonly required in finite element methods. However, the computational costs of such simulations are very high, especially when realistic geometries are considered. To this end, the main contribution of the present work is to reduce the computational time by employing modern parallel computing techniques and make comparisons between the different approaches on the specific simulation problem for numerical simulations of calcium dynamics in cardiac myocytes.

Traditional techniques on parallel computing are MPI (Message Passing Interface) and OpenMP. MPI is a distributed-memory architecture that communicates between different machines by sending and receiving messages. OpenMP, on the other hand, is a shared-memory architecture and works on a single machine with multiple processors. Computation on graphics processing units (GPUs) is a new parallel methodology, which has become increasingly popular in recent years. It is a heterogeneous-memory architecture and uses graphic cards as co-processors. Modern GPUs have thousands of cores, which makes it well suited for large-scale data parallelism. There are three programming models on GPUs, namely, Open Computing Language (OpenCL), Compute Unified Device Architecture (CUDA), and DirectCompute. Among these models, CUDA is the most user-friendly and widely used, thus we decided to use CUDA in present work. In this experiments, the computational performance and simulation accuracy of the serial version of the algorithm are compared to both OpenMP (with 4 cores) and GPU-CUDA implementations.

2.2 Mathematical Models and Meshless Numerical Methods

2.2.1 Governing Equations

To model calcium dynamics in cardiac myocytes, the following nonlinear reaction-diffusion equations, modified from [71], are considered:

$$\frac{\partial [Ca^{2+}]_i}{\partial t} = D_{Ca} \nabla^2 [Ca^{2+}]_i - \sum_{m=1}^{3} R_{B_m} - R_{B_S}, in\ \Omega \qquad (3)$$

$$\frac{\partial [CaB_m]}{\partial t} = D_{CaB_m} \nabla^2 [CaB_m] + R_{B_m}, in\ \Omega, m = 1, 2, 3 \qquad (4)$$

$$\frac{\partial [CaB_s]}{\partial t} = R_{B_s}, in\ \Omega \qquad (5)$$

$$\frac{\partial [Ca^{2+}]_i}{\partial t} = J_{Caflux}, on\ \partial\Omega \qquad (6)$$

where Ω is the interior of cell and $\partial\Omega$ is the cell surface and t-tubule membrane. In [71], the calcium flux term J_{Caflux} is defined in the entire domain, although it always takes a zero value at internal nodes. In this work, however, this term is explicitly defined only on the boundary $\partial\Omega$. Therefore, instead of merging the calcium flux term in the first equation, we have an additional equation (6).

The initial conditions (resting states) used are as follows: $[Ca^{2+}]_i = 0.10\mu M$, $[CaB_1] = 11.92\mu M$, $[CaB_2] = 0.97\mu M$, $[CaB_3] = 0.13\mu M$, $[CaB_s] = 6.36\mu M$. Note that this model and methods examine a portion of the cell, in which reflective boundary conditions are applied during numerical simulation on the part of $\partial\Omega$ where it is not the cell surface or t-tubule membrane. The reactions between Ca^{2+} and buffers are given by

$$R_{B_m} = k_+^m([B_m] - [CaB_m]) \cdot [Ca^{2+}]_i - k_-^m[CaB_m], m = 1, 2, 3 \quad (7)$$

$$R_{B_s} = k_+^s([B_s] - [CaB_s]) \cdot [Ca^{2+}]_i - k_-^s[CaB_s] \quad (8)$$

In our model, three types of mobile Ca^{2+} buffers (Fluo-3, ATP, and calmodulin, denoted by $B_m, m = 1, 2, 3$) and one type of stationary Ca^{2+} buffer (troponin, denoted by B_s) are considered. Their concentrations are denoted by $[CaB_m], m = 1, 2, 3$, $[CaB_s]$ respectively. At the resting (initial) state, all buffers were distributed uniformly throughout the cytosol but not on the cell membrane. The resting concentrations of mobile and stationary buffers satisfy equilibrium conditions (i.e. $R_{B_m} = R_{B_s} = 0$) [66]. The initial concentrations of buffers are calculated in equilibrium with the resting Ca^{2+} concentration, $0.1\mu M$. The total Ca^{2+} flux, J_{Caflux}, on the surface membrane is defined in [66] where Ca^{2+} influx / efflux through L-type calcium channels (LCCs) , sodium-calcium exchangers (NCXs), Ca^{2+} pumps and background leaks are included. J_{Caflux} throughout the cell surface membrane and the surface of t-tubules is defined as follows: $J_{Caflux} = J_{Ca} + J_{NCX} - J_{pCa} + J_{Cab}$, where J_{Ca} is the total LCC Ca^{2+} influx; J_{NCX} is the total NCX Ca^{2+} influx; J_{pCa} is the total Ca^{2+} pump efflux; and J_{Cab} is the total background Ca^{2+} leak influx.

2.2.2 Geometric Model Considered

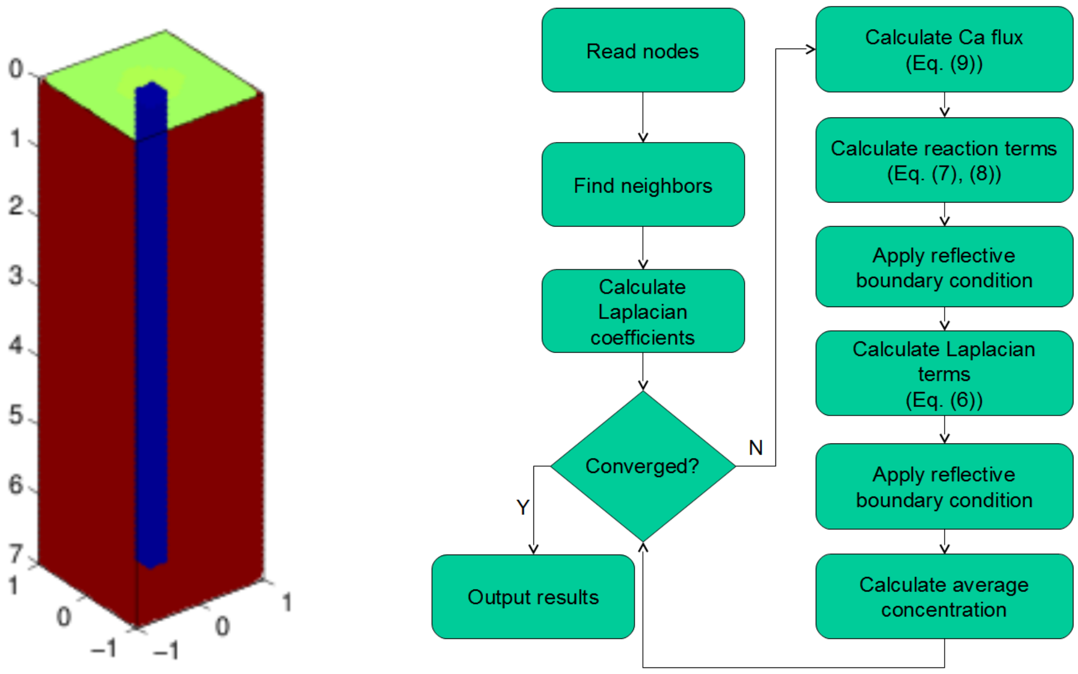
According to [72], [73], [74], a ventricular myocytes may be simplified as repeated structural units consisting of a single t-tubule and its surrounding half sarcomeres. The surrounding half sarcomeres are modeled as a cube-shaped box with dimension of $2\mu m \times 2\mu m \times 7\mu m$, enclosing a t-tubule with dimension of $0.2\mu m \times 0.2\mu m \times 6.8\mu m$. The t-tubule is assumed to be a tiny cube located vertically in the center of sarcomeres, as shown on Figure 8 (a).

2.2.3 Space Discretization (LRBFCM)

The time domain in the system of reaction-diffusion equations is discretized uniformly and explicitly. Thus, the Laplacian term needs to be approximated in each equation with certain spatial discretization. The LRBFCM is used to approximate the Laplacian term $\nabla^2 u(\boldsymbol{x}, t)$ in (3), (4). One may refer to [66] [75] [76]for details. The main idea of LRBFCM is that the collocation can be done on overlapping local domains, yielding many systems of equations with small matrices instead of a single large matrix. The size of the collocation matrices depends on the number of nodes in the local domains.

Briefly speaking, $\nabla^2 u(\boldsymbol{x}, t)$ at each node $\boldsymbol{x}_i, i = 1, 2, \cdots, N$, is approximated by its neighbors. The local domain Ω_i associated with $\boldsymbol{x}_i$ can be created using the n-nearest neighbors to $\boldsymbol{x}_i$ including itself, i.e. $\left\{\boldsymbol{x}_k^{[i]}\right\}_{k=1}^{N} \subset \Omega_i$. In this work, the number of points in each local domain is fixed at $n = 7$. To approximate $\nabla^2 u(\boldsymbol{x}_i, t)$, $u(\boldsymbol{x}_i, t)$ is interpolated on Ω_i by using RBF $\phi(r)$ as follows:

$$u\left(\boldsymbol{x}_j^{[i]}, t\right) = \sum_{k=1}^{n} w_k^{[i]} \cdot \phi\left(\left\|\boldsymbol{x}_j^{[i]} - \boldsymbol{x}_k^{[i]}\right\|\right), j = 1, 2, \cdots, n \qquad (9)$$


0
1
2
3
4
5
6
7
1
0
−1
−1
0
1
Read nodes
Find neighbors
Calculate Laplacian coefficients
Converged?
Y
N
Output results
Calculate Ca flux (Eq. (9))
Calculate reaction terms (Eq. (7), (8))
Apply reflective boundary condition
Calculate Laplacian terms (Eq. (6))
Apply reflective boundary condition
Calculate average concentration

(a) (b)

Figure 8 (a) The model considered in current study, containing the t-tubule (blue), surrounding half sarcomere (red), and external cell membrane (green). (unit: μm). (b) Flow chart of the meshless algorithm for modeling of calcium dynamics.

where the weights $w_k^{[i]}$ are unknown but can be solved by (2). In current work, the multiquadrics (MQ) basis function is used although other radial functions can be used as basis functions as well. The shape parameter $c = 300$ is used in order to achieve fast convergence.

Based on (3), (4) the calcium influx J_{Caflux}, reactions $R_{B_m}, m = 1, 2, 3, s$ and Laplacian term $\nabla^2[Ca^{2+}]_i$ need to be approximated. Because the points we used are fixed in the given geometric space over time, the Laplacian weights $w_k^{[i]}$ are constants for each point and thus can be pre-calculated. However, the J_{Caflux}, $R_{B_m}, m = 1, 2, 3, s$ and $\nabla^2[Ca^{2+}]_i$ will be updated in each time step. Figure 8 (b) shows the algorithm. In general, the numerical solution is found iteratively. At the end of each iteration, a result is tested against a pre-defined criteria. The iteration continues until a result satisfies the given criteria (converged).

2.3 Results and Discussion

The geometric model shown in Figure 8 (a) is considered to simulate calcium dynamics and compare the performances of OpenMP, GPU and serial implementations. The model is discretized in three resolutions: Set 1 contains a total of 3,969 points (including 136 T-Tubule points and 80 cell membrane points) with a node distance of $0.2\mu m$. Set 2 contains a total of 30,807 points (including 545 T-Tubule points and 360 cell membrane points) with a node distance of $0.1\mu m$. Set 3 contains a total of 234,921 points (including 2,185 T-Tubule points and 1,512 cell membrane points) with a node distance of $0.05\mu m$. The time steps for the three cases are $8e^{-3}\ ms$, $4e^{-3}\ ms$ and $1e^{-3}\ ms$ respectively to make the PDEs converge. The total time simulated is $400\ ms$.

Figure 9 shows the numerical results. Concentrations of Ca, Fluo, ATP, Cal, and TN are shown in (a) – (e) respectively. Unit for all concentrations is μM.

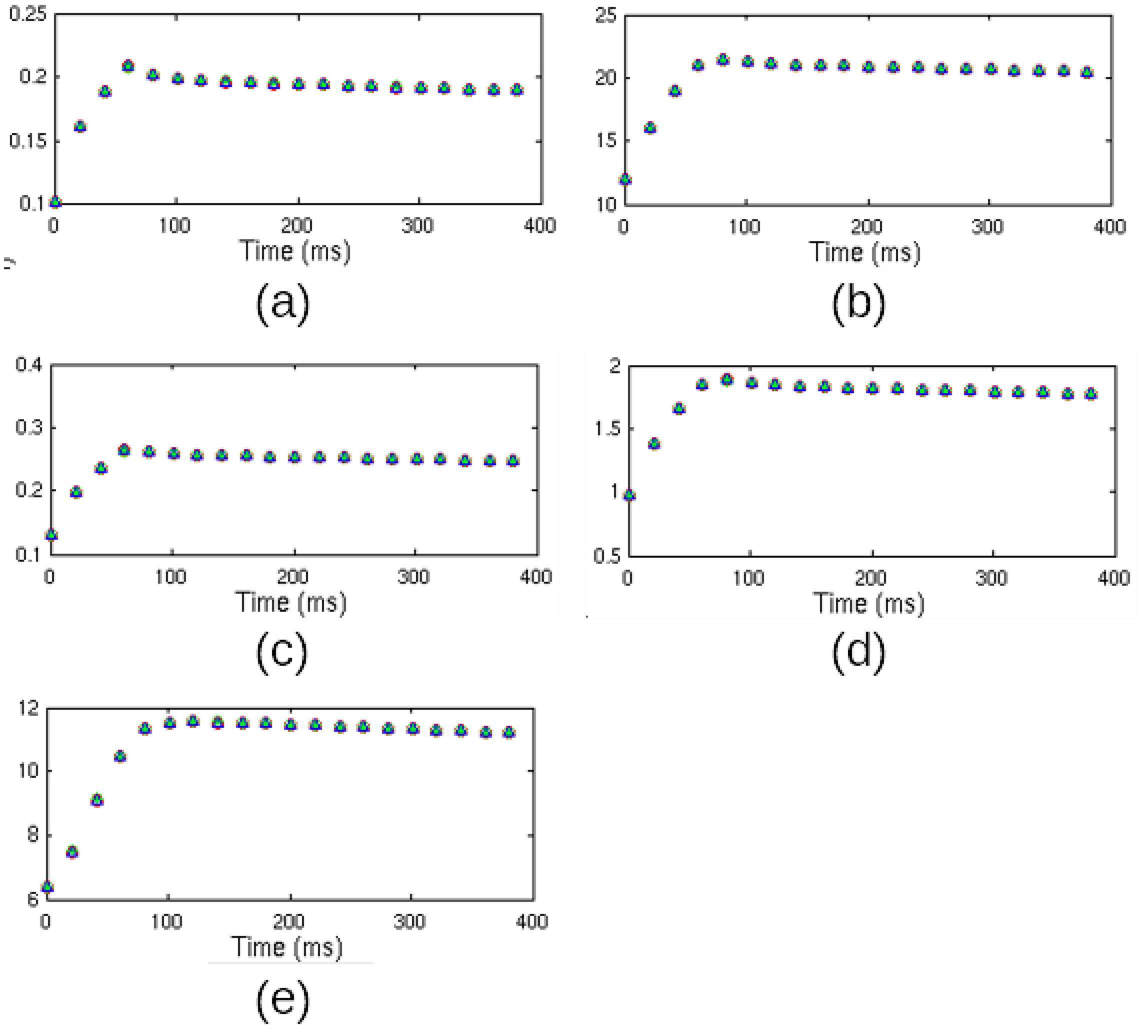


Figure 9 Results of calcium dynamics. Vertical coordinate illustrates concentrations in μM. (a) Concentration of Ca over time. (b) Concentration of Fluo over time. (c) Concentration of ATP over time. (d) Concentration of Cal over time. (e) Concentration of TN over time.

Chapter 3 Anisotropic Image Restoration and Transformation

3.1 Introduction

Image restoration is the operation of converting noisy or corrupted image to the clean original image. Corruption may appear because of motion blur, noise, and camera misfocus, etc. A similar operation is the image enhancement. But image enhancement emphasizes on features of the image to make it visually improved to viewers, but does not necessarily produce realistic data. Image restoration, however, uses a priori model to produce an image.

Modern imaging technologies often digitize an image into a uniform array of pixels (or voxels in 3D). A natural thought is using these pixels to restore an image. But with uniformly sampling, the sampling density is inevitably too high in regions where intensities change slowly and too low in regions whose intensities change rapidly. Despite the ease of use in both hardware and software developments, uniformly-digitized images often pose challenges in data storage and transmission, as well as image processing, especially in 3D medical images that have been consistently and significantly grown in size in recent years. Evolving from previously commonly-used uniform sampling, non-uniform sampling and adaptive mesh triangulation of an image has become an active research area in image processing. Image triangulation involves partitioning an image into a collection of non-overlapping small triangles called mesh elements (also called faces or triangles). This procedure often serves as an image coding method, meaning that an image in pixels is compressed by using a number of “super-pixels”. This method is a compact way to represent images for effective data storage and transmission, and also an efficient way to process and visualize images, especially for 3D images where the number of voxels can be extremely large. In addition, the resulting mesh edges are expected to be well aligned with image features (edges or corners) in order to maintain a faithful restoration of the original image. Mesh modeling of an image has many applications like porous structure design in tissue engineering [77], image compression [78] [79] [80] [81], motion tracking and compensation [82] [83] [84] [85] [86] [87] image processing by geometric manipulation [88], medical image processing [89] [90] [91] [92], feature detection [93], pattern recognition [94], computer vision [95] [96], restoration [97], tomographic reconstruction [98], interpolation [99] [100] [101], image/video coding [102] [103] [104] [105] [106] [107], video modeling [108], image retargeting [109] and registration [110] [111] [112].

A common procedure of image triangulation consists of two steps: 1) generating mesh nodes (vertices) by choosing a set of sampling points defined in the image domain, and 2) connecting these mesh nodes by Delaunay triangulation [113]. Delaunay triangulation is a geometric operator and can avoid long and thin triangles that often lead to poor approximations. The selection of sampling nodes, however, is data-dependent, where the connectivity of the triangulation depends on the data set, based on which the mesh nodes are generated. Depending on how to generate mesh nodes, there are two categories of the image triangulation. The first one places mesh nodes inside the image features but near both sides of feature edges. So the triangulated images of this category show double-layer vertices at both sides of feature edges. The second category places mesh nodes directly at the feature edges, thus there are only single-layer vertices defined right on feature edges. Yang et al. [114] employed Floyd-Steinberg error-diffusion (ED) algorithm to place mesh nodes so that their spatial density varies according to the local image content. As a result, the triangulated images fall into category I. Adams [115] employed greedy-point removal (GPR) and error-diffusion scheme together to achieve meshes of quality comparable to the original GPR scheme but at a much lower computational and memory complexities. With the conjunction of smoothing operators, this method produces image triangulation of category I. Adams also proposed a framework in [116] for mesh generation by fixing various degrees of freedom available within that framework. This method performs extremely well and produces meshes of higher quality than the GPR method, and is considered as a method of category I as well. By contrast, Li et al. [117] proposed a modified version of Rippa [118] and Garland-Heckbert (GH) [119] frameworks which can generate single-layer mesh nodes on edges, and this framework generates triangulated images of category II. Another method of this category was proposed by Tu et al. [120], based on constrained Delaunay triangulations. In this method, the approximating function is not required to be continuous everywhere but with discontinuities being permitted across constrained edges of triangles in triangulation.

Both categories of image triangulation generated by the methods mentioned above have their advantages and disadvantages. For the first category (double-layer vertices), the quality of image restoration is usually better because all vertices are well defined on images features thus the intensities of pixels during image restoration process will not be affected by edges. As a result, the edges in the recovered images are sharp and the Peak Signal-to-Noise Ratio (PSNR) is usually higher. While the restoration quality of methods in category I is high enough for subjective quality testing, the two layers must be very close to each other in order to have well-defined and sharp image edges. A consequence of this is that the resulting meshes can easily contain lots of thin and long triangles between the two layers, which could cause large approximation errors when the meshes are used for numerical analysis (like finite element analysis). Additionally, in many applications, the direct communication between different materials should be maintained, meaning that no “cushion” layer between materials should be introduced in the meshes. Moreover, using two layers of mesh vertices usually has more storage cost, resulting more memory space to be used and more time to transmit it over the network. Methods of category II avoid the small triangles and also the “cushion” layer problem, thus the mesh quality is usually better if proper steps are taken. However, the vertices are defined on feature edges, where the nodal intensities are ambiguously defined. That is, the intensity of an edge pixel will be contributed by both sides of the image edge. As will be shown in the experiments, the restored images often suffer from blurred and distorted feature edges if it is not properly addressed.

Because the image restoration method in this project is a single step in image processing, the same mesh will be used for numerical analysis in the future thus a method that lies in the second category (single-layer approaches) is more interested for this project because the obvious limitation of methods of category I. However, in order to address the blurring and distortion problems often seen in existing approaches in this category, a method based on the radial basis function (RBF) interpolation is proposed with the following improvements: 1) rather than considering only the Euclidean distances between vertices, this method also takes into consideration the image local orientations, yielding an anisotropic radial basis function (ARBF) and 2) this method does not use intensities of vertices directly, but instead utilize the intensities of triangles to eliminate the uncertainty of nodal intensities on feature edges.

This method provides a new approach to restore image from triangular mesh. Because triangular mesh representations of images have much fewer nodes (sampling rates are often as low as 5% - 6%) defined, storage, transmission, and image operations like smoothing, sharpening, etc. will be much faster at the mesh domain instead of pixel domain. Also, this proposed method can be used to visual verification for image operations done in mesh domain.

Image transformation is a function that takes an image as its input and produces an image as its output. The input and output image may appear similar but having different interpretations or entirely different, depending on the transform chosen. These transforms can be performed in spatial domain or frequency domain. Examples of image transformation includes Fourier transform, principal component analysis (PCA) and various spatial filters. Like geometric transformation in computer graphics field, image transformation in spatial domain includes translation, rotation and scaling. Translation of an image means move an image to another location in coordinates system. Its implementation is simple that only an offset is added to the location of every pixel of the image. The output of image translation produces images with the same size and intensity as input image but different coordinates. Image rotation produces images with the same size, but rotated an angle around a given position comparing to the input image. Its implementation is also not complicated that a rotation matrix is multiplied to the location of every pixel of the image. One can use this operation to rotate and flip an image.

Both translation and rotation are called rigid-body transformations (the image dimension is unchanged after transformation). Image scaling, however, changes image dimension after transformation. Image scaling relates image enlargement (upscaling) and shrinkage (downscaling). Interpolation techniques like bilinear, bi-cubic and nearest-neighbors are often used in its implementation.

High resolution images or videos are usually desired in most digital imaging applications for two principal areas: improvement of pictorial information for human perception and automatic machine interpretation. There are five types of digital image resolution: pixel resolution, spatial resolution, spectral resolution, temporal resolution and radiometric resolution. This paper concentrates on *spatial resolution*. From spatial resolution perspective, digital images are comprised of many regularly aligned small elements called pixels. Two approaches can be applied to increase spatial resolution. The first one is using better imaging sensors like charge-coupled device (CCD) or complementary metal-oxide-semiconductor (CMOS). But there is a limit on this approach because of the hardware cost and physical constraints. The other one is image super-resolution (SR).

Image super-resolution (SR) are techniques that construct high-resolution (HR) image(s) from low-resolution (LR) image(s). The basic idea is to combine multiple LR frames to generate a HR image. This paper focuses on another closely related technique - single image interpolation, which can often be used to increase the image size. Feature preserving image interpolation is an active area in the image processing. Many methods have been proposed in the past decades to tackle this problem [121], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132], [133]. Nearest neighbor and bilinear interpolation are two simple methods for image interpolation, which are of order 0 and 1, respectively [121]. Despite their simplicity and very low computational cost, these methods suffer from severe blocky artifacts as well as blurring and ringing artifacts near the edges. Although better performance can be achieved by using higher order splines, oscillatory edges and ringing artifacts still exist in higher order spline methods [134]. The main reason is that these methods are intensity-based but not feature-based. They do not consider factors other than intensity. As the final recipient of any image processing, human visual system is very feature-sensitive. These features are mostly edges and corners within the image thus the sharpness is also of high importance. Radial basis function interpolation incorporates intensity and location information so it performs better than those intensity-based methods. However, because the Euclidean distance it uses is isotropic regardless of image features, blurring artifacts are often found in RBF interpolated result. Anisotropic radial basis function (ARBF) interpolation solves this drawback by using anisotropic distance. For the interpolated point, contributions of neighbors are calculated by adaptive distance [135]. This chapter discusses a new image super-resolution method based on ARBF. Moreover, this method operates on triangular meshes instead of pixels for several advantages.

Triangulated image is comprised of many nodes (called vertices) and triangles (called faces). Traditionally, pixel-based interpolation is very popular for single-image SR problems because it is intuitive and easy to implement. Because triangular mesh of a digital image has much less vertices and faces comparing to the number of pixels, it provides a compressive representation of an image. Image operations like smoothing, denoising, etc. based on triangular mesh requires less time and memory space to compute, store and transmit. Triangulate image adaptively is desired because it generates many small triangles near the curve and corners and few large triangles on non-feature area. Thus it provides a good accuracy while keeps the number of triangles relatively small.

The remainder of this chapter is organized as the following. Section 3.2.1 briefly summarizes the mesh generation method. Section 3.2.2 introduces image restoration using traditional isotropic RBF interpolation. The more accurate image restoration is presented in Section 3.2.3. Section 3.2.4 shows the detailed algorithm of this method. The image super-resolution (SR) techniques used in this Section 3.3 are explained in Section 3.3.2. Finally, Section 3.4 shows the experimental results and discussions. Section 3.5 concludes this chapter.

3.2 Anisotropic Image Restoration

While mesh generation from images is not the main focus of this chapter, a brief summary of this step is given just for completion of the present work. Then more details of traditional (isotropic) radial basis function (RBF) interpolation is introduced, followed by the anisotropic RBF-based interpolation for image restoration from meshes. The details of the implementation algorithm is given below as well.

3.2.1 Adaptive Mesh Generation from Images

A series of algorithms are used to generate high quality, feature-sensitive, and adaptive meshes from a given image. Firstly, three kinds of the sample points (namely, Canny's points, halftoning points, and uniform points) are generated. Secondly, a triangular mesh is generated from these points by using constrained Delaunay triangulation. The Canny's edge detector is employed to guarantee that important image features are preserved in the meshes. A halftoning-based sampling strategy is adopted to provide feature-sensitive and adaptive point distributions in the image domain. Finally, a Delaunay-triangulation is used to generate initial quality triangulation of the image. These steps are briefly summarized below.

- **Canny Sample Points** Image edges are important features in an image and need to be preserved in the obtained meshes. Canny edge detector is a well-known method to deal with boundary extraction. In this chapter, Canny edge detector is used to generate the initial Canny edge points and they are strictly attached to the boundary of the features of the image. However, the initial Canny edge points are too dense to yield quality meshes if all these edges are used as mesh nodes. In this method, the curvature information of every Canny's edge point is taken into account and the Principal Component Analysis (PCA) is used to determine the sampling density. The PCA method can detect the overall attribute of the neighbors of a certain size by a statistical way. After the PCA sampling, tiny features and features with high curvature have dense sample points and big features or features with straight lines have sparse sample points.

- **Halftoning Sample Points** The edge points generated by the Canny edge detector described above can only capture pixels on or near the image edges. In order to have a decent initial mesh, one has to scatter some more points in the non-edge regions of the image. The halftoning sample points are adopted based on the approach described in [114]. This method generates the sample points using the second derivatives of an image, where most of the sample points are placed near the image features (edges).
- **Uniform Sample Points** Although the halftoning sample points can cover most non-edge regions of the image, it is possible that no point (either Canny or halftoning) is found in regions of almost constant intensities. Therefore some points need to be generated uniformly to cover the rest of the images where the first two types of sample points are not located. A point (x, y) is said to be a valid uniform sample point if no Canny's or halftoning points are found in its neighborhood in a fixed distance.
- **Mesh Generation via Constrained Delaunay Triangulation** The sample points found above are used to generate the triangular mesh for a given image by using the Delaunay triangulation. A popular open source software *Triangle* [136] is employed for Delaunay triangulation. In order to guarantee the obtained meshes being well aligned with image edge features, a set of line segments are filled in *Triangle* as additional constraints formed by connecting the Canny's sample points along the detected Canny's edges. With all the described strategies combined, high quality, feature-sensitive, and adaptive meshes are generated from a given grayscale image. Some meshing examples will be shown in the results section below.

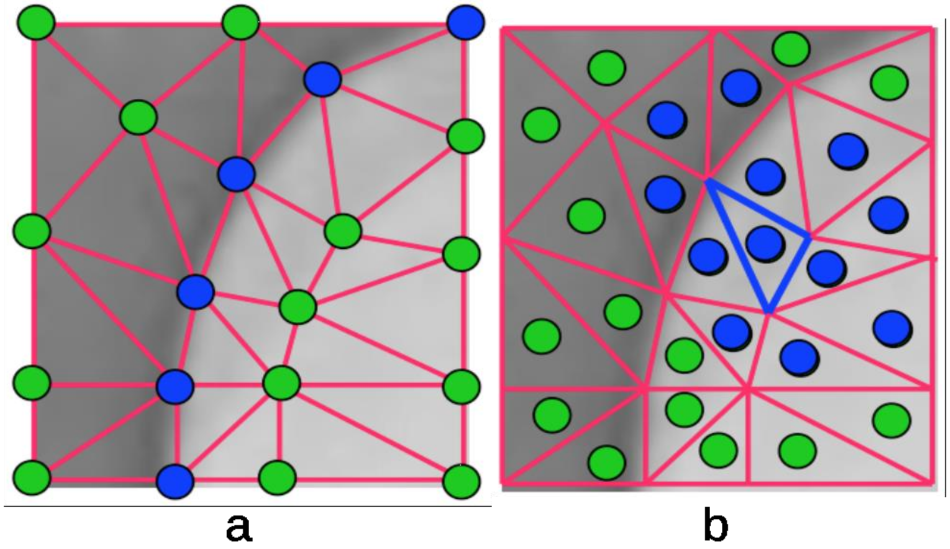


Figure 10 Example of interpolation. (a) Interpolation by vertices. Green dots are vertices defined on feature. Blue dots are vertices defined on feature edge. (b) Interpolation by faces. Green dots are face centers. Blue dots are face centers used for interpolation of the intensities of pixels enclosed by the blue triangle.

3.2.2 Radial Basis Function (RBF) Interpolation

Image restoration is all about restoration intensities of every pixel of the image. As Section 3.1 explained, in this application, triangular mesh representation of an image is used to determine the intensities of pixels of the image by interpolation.

One question about restoring image from triangular meshes is: what intensities should be used, intensities on vertices or intensities on faces? In the mesh generation approach described above, many vertices are placed on image edges. These vertices are good to capture image gradients (or orientations) but not for image intensities because there is an ambiguity in assigning intensity to a node defined on an edge, as illustrated for blue nodes in Figure 10 (a). Obviously, a very small change (or error) on the location of blues nodes would make a big interpolation difference if the mesh vertices are used as the nodal values in RBFs. A better way is to use face centers as the nodal values for RBF interpolations, which can eliminate the ambiguity and is less sensitive to mesh

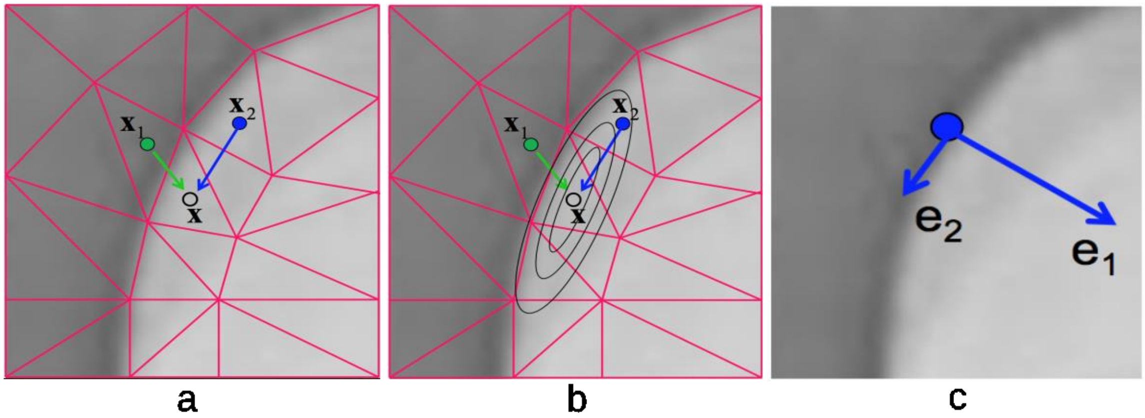


Figure 11 Interpolation schemes. (a) Isotropic RBF interpolation. (b) Anisotropic RBF interpolation. (c) Eigenvectors on an edge pixel. e_1 shows the normal direction. e_2 shows the tangent direction.

errors. Figure 10 (b) shows this idea, where the face centers are more robust to the location changes of mesh vertices. Results of vertex-based RBF interpolation and triangle-based RBF interpolation can be found in Figure 13 (c) and Figure 13 (d) in Section 3.3. Taking the intensities of triangles as input, the weights can be solved by (2).

Although using face centers performs better than the vertex-based RBF interpolation, the traditional RBF is isotropic in the sense that only the geometric distance information is considered, which often causes blurring and distortion artifacts as can be seen in Figure 13 (d). To capture the anisotropicity of the image features, the direction of image edges has to be considered as well. Otherwise, nodes across feature edges may have strong influence on the pixel being interpolated. Figure 11 (a) shows the cause of the blurred edge problem. x is the pixel whose intensity needs to be calculated. The intensities on nodes x_1 and x_2 are two of the neighbors used for interpolation. The weights of them are determined only by the Euclidean distance to x based on the definition of traditional RBF. However, x_1 is on the other side of the feature edge, so it should have much less influence on x than x_2. The isotropic RBF has a hyper-spherical support domain which cannot satisfy this data-dependent requirement. Thus the intensity on x is blurred by x_1. By contrast, Figure 11 (b) shows the anisotropic RBF (ARBF) interpolation. The support domain of ARBF is a hyper-ellipsoid. By choosing proper shape parameter, the support domain could rule out the interfering node x_1, or give insignificant weight to node x_1. Thus the blurring effect will be eliminated and sharp features can be well retained. Section 3.2.3 elaborates on the detail of designing anisotropic radial basis functions for image restoration.

3.2.3 Anisotropic Radial Basis Function (ARBF) Interpolation

The main difference between the isotropic and anisotropic RBFs is the definition of distance metrics used. As in [135], the anisotropic RBF is defined as follows:

Definition 1. *Given N distinct points* $X = \{\boldsymbol{x}_j \in \mathbb{R}^d\}_{j=1,\cdots,N}$ *and a* $d \times d$ *positive definite matrix* $\boldsymbol{T}$, *the anisotropic radial basis function associated with a radial basis function* $\Phi_j(\cdot) = \phi\left(\left\|\cdot - \boldsymbol{x}_j\right\|_2\right)$ *is defined by*

$$\Phi_{\boldsymbol{T},j}(\cdot) = \phi\left(\left\|\cdot - \boldsymbol{x}_j\right\|_{\boldsymbol{T}}\right) \tag{10}$$

where $\|\boldsymbol{x}\|_{\boldsymbol{T}} = \boldsymbol{x}^T \boldsymbol{T} \boldsymbol{x}$.

The support domain of ARBF is hyper-ellipsoid instead of a hyper-sphere in traditional RBF. Its center is $\boldsymbol{x}_j$, associated with the quadratic form $\left(\boldsymbol{x} - \boldsymbol{x}_j\right)^T\left(\boldsymbol{x} - \boldsymbol{x}_j\right)$. Interested readers can refer to [137] [138] for more details of ARBF.

To construct the metric **T**, the image structure tensor is used

$$G_\sigma * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

where G_σ is the Gaussian smooth operator, and $\begin{bmatrix} I_x \\ I_y \end{bmatrix}$ is the image gradient at a pixel. Two eigenvectors e_1 and e_2 are the normal and tangent directions of the edge, respectively, as shown in Figure 11 (c). The corresponding eigenvalues are λ_1 and λ_2. The anisotropic metric is defined by

$$\mathbf{T} = [e_1 \quad e_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} e_1^T \\ e_2^T \end{bmatrix} \qquad (11)$$

Similar to the isotropic RBF but with a modified distance metric, the ARBF image interpolation problem becomes

$$\mathrm{u}(\boldsymbol{x}) = \sum_{i=1}^{N} w_i' \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|_{\boldsymbol{T}}) \qquad (12)$$

Please note that the matrix in (2) should also be updated accordingly with the new distance metric **T**. Therefore, the new set of weights w_{i}' would be different from the weights w_{i} in the isotropic RBF interpolation.

3.2.4 Algorithms

The following algorithm shows the steps of the proposed approach for image restoration from triangular meshes. The major step is the ARBF interpolation which comprises two sub-steps. First, the weight coefficients are solved by using the new distance metric **T**. As stated in section 3.2.2, this is done by taking intensities at triangle centers. Then the weights are applied to (12) to restore the intensity of each pixel. More details are explained in [139]

Algorithm: Image Reconstruction

```
ImageReconstruction()
{
     // read image mesh
     loadMesh();

     // calculate intensities for each triangle center
     calculateTriangleCenters();

     // find the neighboring vertices and triangles for
     // each node
     findNeighbors();

     // rescale eigenvalues in equation ( 11 )
     calculateEigenvalues();

     // calculate metric T in equation ( 11 )
     computeMetrics();

     // do the ARBF interpolation in equation ( 12 )
     ARBFInterpolation();

     // output result
     printResult();
}
ARBFInterpolation()
{
     for (every triangle centers)
          solveCoefficients();
     end

     for (every triangles)
          for (every pixel in current triangle)
               applyCoefficientsToInterpolation();
          end
     end
```

}

3.3 Anisotropic Image Transformations

3.3.1 Isotropic Image Transformations

The coordinates of pixels of target image basically are linear transformations of coordinates of the original image. Mathematically, image translation can be implemented by adding an offset (translation vector) to original coordinates of image pixels. Written in matrix form as $P' = P + T$, where P' is the translated image coordinates, $P = [x, y]^T$ is the original coordinates, $T = \left[t_x, t_y\right]^T$ is the offset. But translation is an affine transformation with no fixed points. Matrix multiplication always have the origin as a fixed point. So the homogeneous coordinates can be used to represent a translation of a vector space with matrix multiplication. Using homogeneous coordinate as a workaround, image translation can be written in matrix multiplication $P' = T\left(t_x, t_y\right) \cdot P$. In detail, this equation can be written as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{13}$$

To perform image rotation, a rotation angle θ and a position (rotation point or pivot point) (x_r, y_r) need to be specified. Positive θ defines a counterclockwise rotation and negative θ defines a clockwise rotation. Mathematically, assuming the origin is the pivot point, image rotation can be defined as $P' = R \cdot P$, where $R = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$. For arbitrary pivot point, the pivot point can be move to the origin along with the image, then the rotation is performed and finally the pivot point is translated back to previous coordinates along with the image. Similar to image translation, image rotation can be written in matrix multiplication form represented by homogeneous coordinates as $P' = R(\theta) \cdot P$. In detail, this equation can be written as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (14)$$

Unlike translation and rotation, image scaling is not rigid-body transformation and changes the image dimension, with respect to the origin. To perform image scaling, scaling factors $\left(s_x, s_y\right)$ and a fixed point need to be specified. Mathematically, image scaling with respect to the origin can be defined as $P' = S \cdot P$, where $S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$. For arbitrary fixed point scaling, the fixed point can be moved to the origin along with the image, then the scaling is performed and finally the fixed point is translated back to previous coordinates along with the image. There are several cases to discuss. Case 1: if $s_x < 1\ and\ s_y < 1$, image dimension is reduced (downscaling). Case 2: if $s_x = 1\ and\ s_y = 1$, image dimension is unchanged. Case 3: if $s_x > 1\ and\ s_y > 1$, image dimension is increased (upscaling). Case 4: if $s_x = s_y$, scaling is uniform. Case 5: if $s_x \neq s_y$, scaling is different. Case 6: if $s_x\ or\ s_y$ is negative, the image is not only resized, but also its coordinates are reflected about the coordinate axes. Similar to rigid-body transformation introduced above, image scaling can be written in matrix multiplication form represented by homogeneous coordinates as $P' = S\left(s_x, s_y\right) \cdot P$. In detail, this equation can be written as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (15)$$

In practice, image could be transformed according to a given transformation sequence. Instead of perform transformation at every stage, transformations can be composited to a single stage to get better performance. For example, if an image needs 3 transformations: $P' = M_3 \cdot M_2 \cdot M_1 \cdot P$. The composite transformation can be written as matrix multiplication $P' = M \cdot P$, where $M = M_3 \cdot M_2 \cdot M_1$.

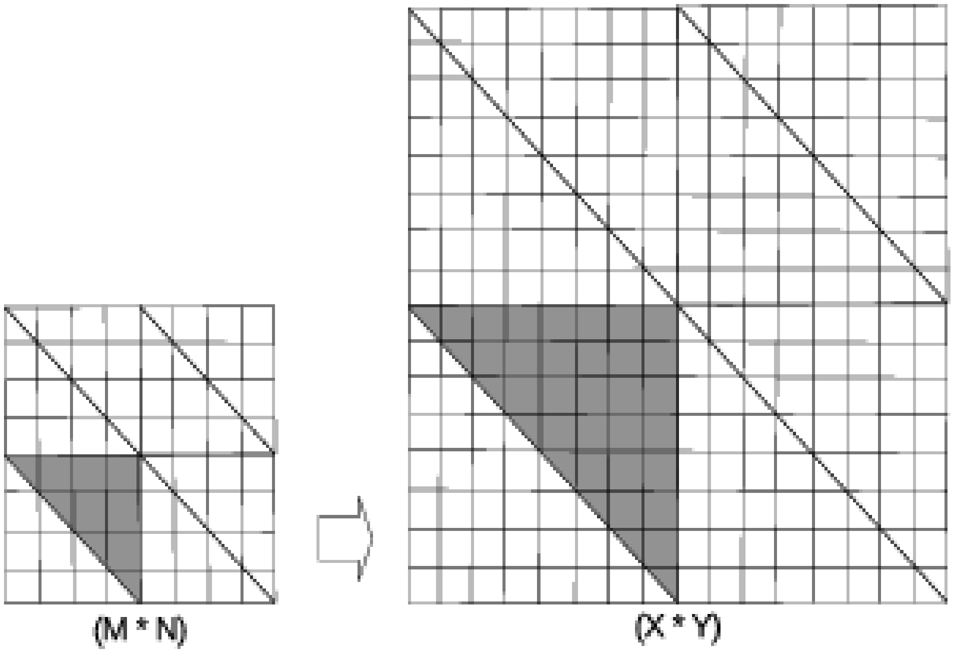


Figure 12 Resampling of original image (M × N) to target image (X × Y). As the gray triangle shows, every triangle in target image keeps the shape unchanged but has larger sampling density.

The image transformations (translation, rotation and scaling) introduced above are all isotropic. Because translation and rotation are rigid-body transformations, they do not change image features. Image scaling, however, change image dimensions after transformation, which will blur image features because isotropic scaling only considers the coordinates of pixels and thus is not feature-sensitive. To better preserve image features, anisotropic scaling is required.

3.3.2 Anisotropic Image Scaling

Based on this, the interpolated image is resampled according to the new dimension. Figure 12 shows this idea. Gray triangle which is currently considered in original image (smaller on the left) is mapped onto the target image (larger on the right). Each triangle in the enlarged image contains more pixels. Intensities of pixels in each triangle are determined by ARBF interpolation described in section 3.2.3. Unlike traditional interpolation on pixels, our proposed SR method uses triangle-based interpolation. The following algorithm shows the full process. The first step is triangulating the input image to generate the triangular mesh described in section 3.2.1 and calculating eigenvalues and corresponding eigenvectors defined on vertices. Then calculate intensity, eigenvalues and corresponding eigenvectors of each triangle (represented by triangle center) by taking average of three vertices of current triangle. The neighborhood of vertices and triangles need to be determined for calculating anisotropic metric. The eigenvalues of triangles are rescaled according to [138]. The eigenvalues and eigenvectors of triangles are then used for calculating anisotropic metric mentioned above. The interpolation is performed on triangles one by one. During the interpolation, the current triangle is first resampled to contain more pixels, then ARBF interpolation method is used when interpolate each pixel.

The formal algorithm is described below:

```
ALGORITHM: Image Super-resolution
INPUT: original image
OUTPUT: enlarged image
ImageSuperresolution() {
     // image triangulation described in section 3.2.1
     triangulate();

     // calculate intensities for each triangle center
     calculateTriangleCenters();

     // find neighboring vertices and triangles for nodes
     findNeighbors();

     // rescale eigenvalues
     calculateEigenvalues();
     // calculate anisotropic metric
     computeMetrics();
```

```
    // do the ARBF interpolation
    ARBFInterpolation();

    // output result
    printResult();
}

ARBFInterpolation() {
    for (every triangle) {
        solveCoefficients();
        resampling();
        for (every pixel in current triangle)
            interpolate();
    }
}
```

3.4 Results and Discussion

Numerous experiments have been conducted on publicly available images by using the proposed approaches and the image restoration results are all promising. Due to the space limit, only the well-known “Lena” image and three medical images of different sizes are considered. Figure 13 (a) is the original Lena image of size 256×256 pixels. Figure 13 (b) is the result of assigning a constant intensity to all pixels in a mesh triangle (so-called piecewise interpolation). Obviously this result shows heavy mosaic effect. Figure 13 (c) is the result of iso-RBF interpolation using intensities on vertices. As previously stated on section 3.2.2 the ambiguity of intensities on vertices blurred the result. Figure 13 (d) is the result of iso-RBF interpolation using intensities on triangle centers. In this case, there is no ambiguity of intensities. So the result is much better comparing to Figure 13 (c). However, the feature edges are still blurred and some distortions are clearly seen

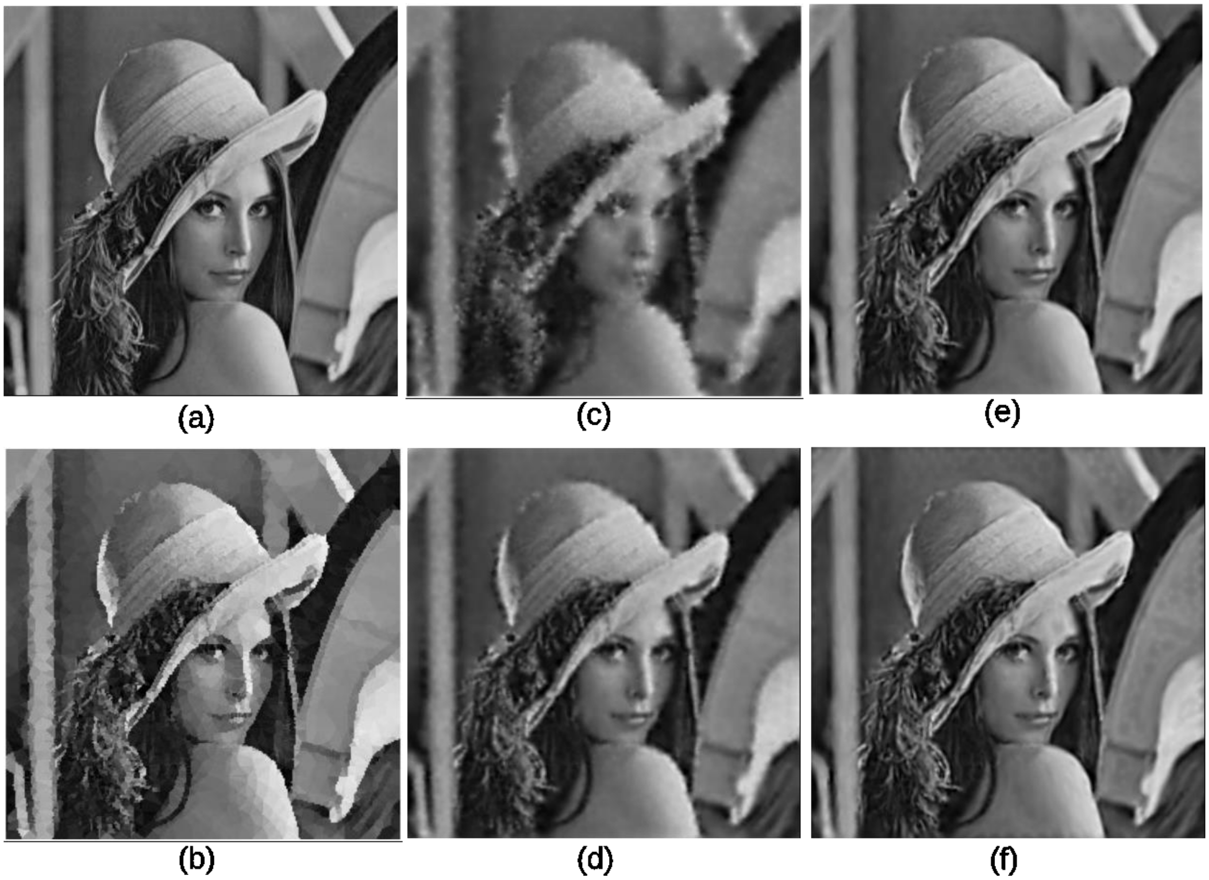


Figure 13 Summary of restoration of Lena. (a) Original Lena image. (b) Result of piecewise interpolation. (c) Result of vertex-based iso-RBF interpolation. (d) Result of triangle-based iso-RBF interpolation. (e) Result of triangle-based ARBF interpolation using MQ basis. (f) Result of triangle-based ARBF interpolation using IMQ basis.

because of the lack of directional information used in isotropic RBF. Figure 13 (e) is the result of ARBF interpolation using intensities on triangle centers with multi-quadrics (MQ) basis function. The result is much better thanks to a modified distance metric that incorporates both geometric distances and data-dependent feature orientations. Figure 13 (f) is similar to Figure 13 (e), except that the basis function is inverse multi-quadrics (IMQ). Other basis functions like Gaussian and Thin-Plate-Spline (TPS) are also tested. However, it is hard to find a proper shape parameter to get a reasonable result for Gaussian, and the TPS basis does not converge.

More details of the Lena experiment are shown in Figure 14. Figure 14 (a) is the original Lena image, the same as Figure 13 (a). Figure 14 (b) is the mesh generated by the method outlined in section 3.2.1. Figure 14 (c) is the recovered image, which is the same as Figure 13 (e). To visually see the generated mesh and compare the difference between the original and restored images, Figure 14 (d) - (f) are the zoomed-in views of Figure 14 (a) - (c), respectively. As the results show, the mesh quality is high enough for subsequent numerical analysis and the recovered image is very close to the original one. As a matter of fact, the restored image looks smoother due to the smooth radial basis functions used, and the sharp edge features are well preserved. Figure 15 shows the original brain MRI, its generated mesh, and the result of ARBF interpolation using intensities on triangle centers with the MQ basis function. The zoomed-in views show the quality of mesh and restoration as well. Figure 16 shows another MRI experiment of breast. Figure 17 shows a CT-scanning experiment. From all these examples, one can see the effectiveness of this approach for image mesh generation and feature-preserving restoration.

To give quantitative evaluation of the restored images, we use the widely-used peak signal-to-noise ratio (PSNR) as defined below:

$$PSNR = 20 * \log_{10}\left(\frac{255}{RMSE}\right)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{N}[O(i,j) - I(i,j)]^2}{M * N}}$$

where *M* and *N* are the dimensions of the image. $O(i,j)$ is the original intensity at pixel (i,j) and $I(i,j)$ is the interpolated intensity at (i,j). Table 1 gives a summary of the Lena image using different restoration approaches. The compression ratio in the table means the ratio of the number of vertices in the mesh vs. the number of pixels in the original image. The restored image with the anisotropic RBF interpolation gives the best PSNR which conforms to the visual result. Table 2 summarizes the other three data sets, where the running time of image restoration for each case is included and measured on a 2.5 GHz i5-3210m CPU and 2 GB RAM. The proposed algorithms were implemented in C language and will be released to the public.

Lena (256×256, compression ratio is 6%)	PSNR (db)	Shape Parameter
Piecewise Interpolation	22.9703	0.5
Triangle-based ISO-RBF Interpolation	26.7367	0.5
Triangle-based ARBF Interpolation (MQ)	28.2088	0.5
Triangle-based ARBF Interpolation (IMQ)	27.1836	1.8

Table 1 Summary of the Lena image (Figure 13)

Data	Size	Compression Ratio	PSNR(db)	Shape Parameter	Time (sec.)
Brain	285×341	6%	15.7058	0.5	0.41
Breast	512×512	5%	11.8763	0.5	1.10
Heart	356×396	5%	10.5208	0.5	0.63

Table 2 Summary of the three medical images (Figure 15, Figure 16, Figure 17)

3.5 Conclusions

This chapter describes a nonlinear interpolation method by using anisotropic radial basis functions and structure tensor driven metrics. Using the ARBF interpolation, an original image can be stored and processed in the mesh format with some nice advantages including less storage requirement, faster transmission speed, and more efficient image processing due to the significantly reduced number of mesh nodes as opposed to the number of pixels in the original image. The generated meshes, after some post-processing such as mesh-based segmentation, can be readily used for further numerical analysis. The present image restoration algorithm provides an effective way to

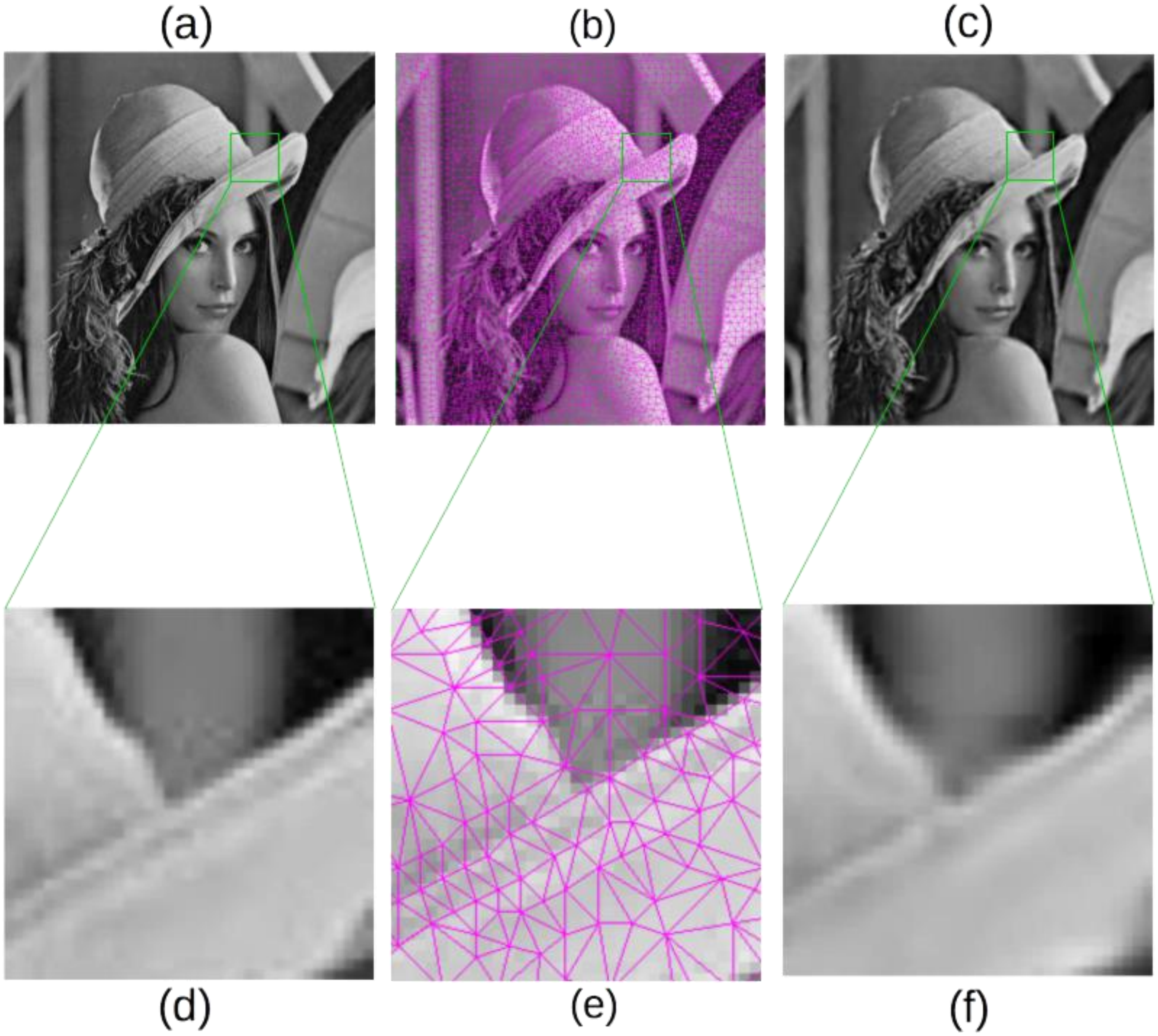


Figure 14 Details of Lena. (a) Original Lena image. (b) Generated mesh of (a). (c) Result of triangle-based ARBF interpolation using MQ basis. (d) - (f) are zoomed-in views of (a) - (c), respectively.

restore the image with an arbitrary super-resolution from a mesh representation, serving as a decoding algorithm for the mesh-based image coding technique. The anisotropic RBF algorithm can be used as a de-blurring process as well with sharp features well preserved in the images.

As the image restoration algorithm shows, the time complexity of the function `ARBFInterpolation()` is $O(m \times n)$, where m is the number of triangles and n is the number of pixels inside a triangle. In case of 3D images or very large 2D images, the running time could

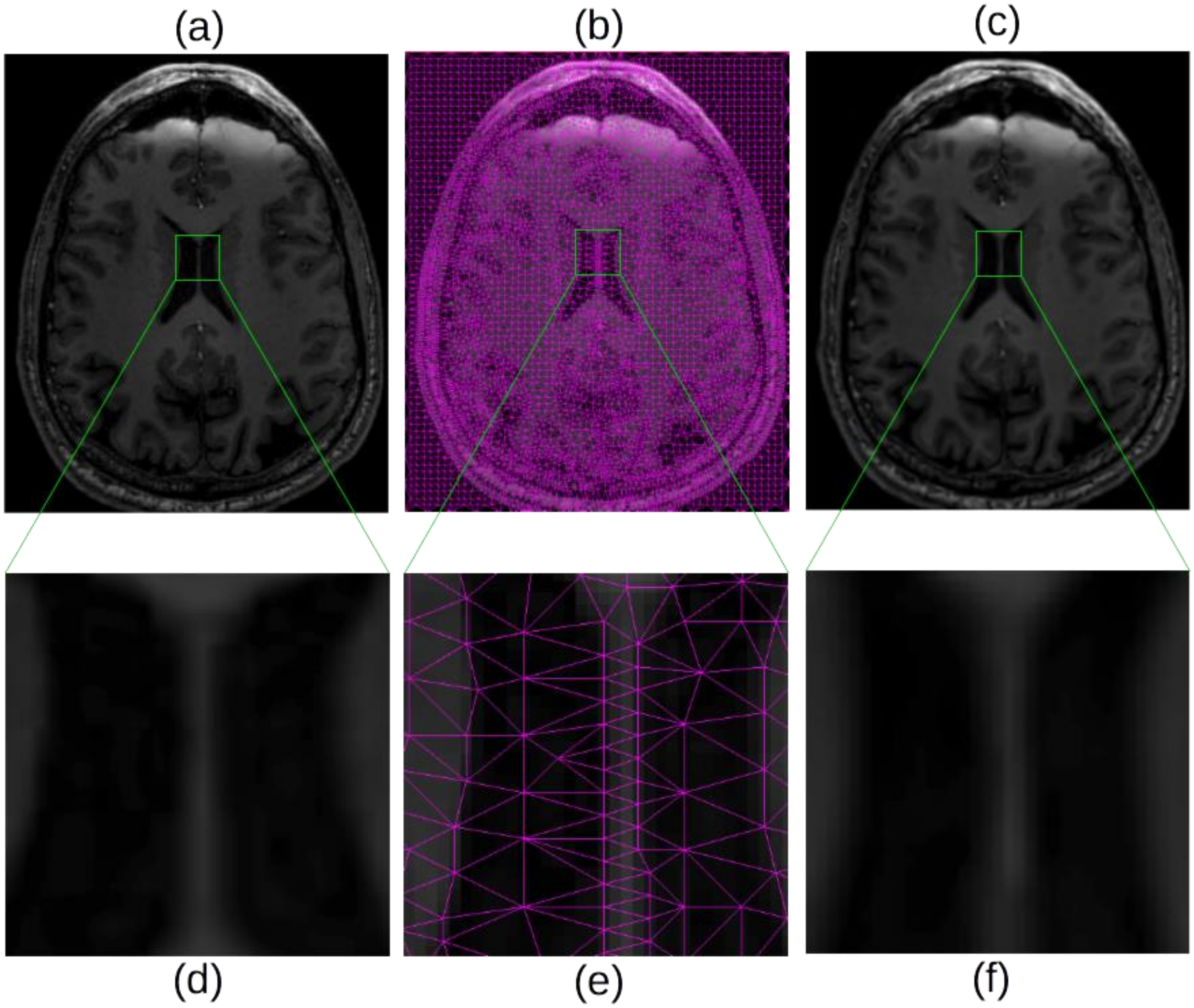


Figure 15 Details of brain MRI. (a) Original brain MRI. (b) Generated mesh of (a). (c) Result of triangle-based ARBF interpolation using MQ basis. (d) - (f) are zoomed-in views of (a) - (c), respectively.

be very expensive. One of our further investigations would be the parallel implementation of the proposed algorithm using GPU programming. Fortunately the present method is very straightforward to parallelize. Additionally we are also interested in the mesh-based image segmentation by using the adaptive meshes generated from the original images, and in applying the segmented meshes to image-based numerical analysis.

Figure 18 shows isotropic image translation and rotation. As introduced in Section 3.3.1, image translation and rotation are rigid-body transformations and do not change image dimensions. Three

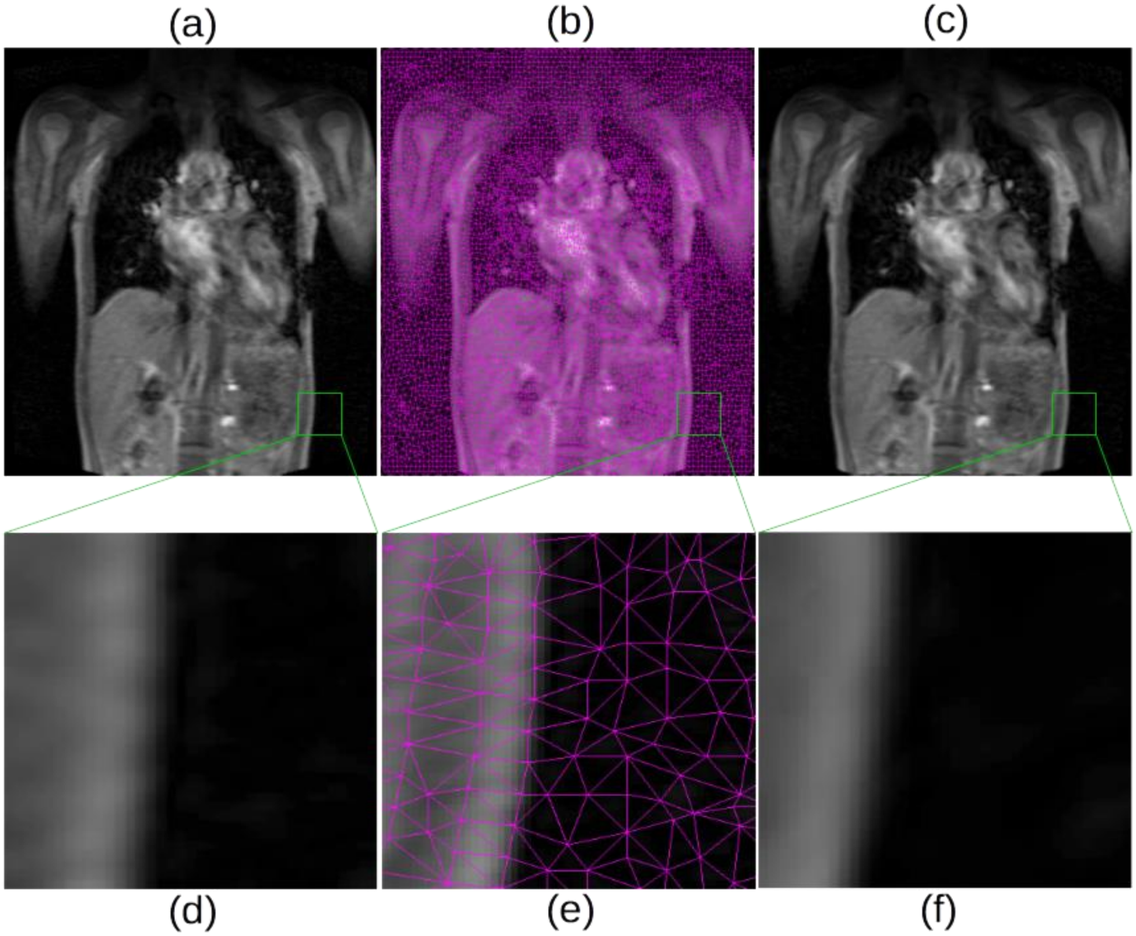


Figure 16 Details of breast MRI. (a) Original breast MRI. (b) Generated mesh of (a). (c) Result of triangle-based ARBF interpolation using MQ basis. (d) - (f) are zoomed-in views of (a) - (c), respectively.

image features are highlighted in Figure 18 (a) to show the translation clearly. Figure 18 (b) shows image rotated 45° respect to the origin.

Numerous experiments are conducted on biomedical images using this approach and the SR results are all promising. All results are enlarged by 2X, 3.5X, 5X for comparison using the shape parameter 0.5. This chapter only shows three experimental results and 5X enlargement.

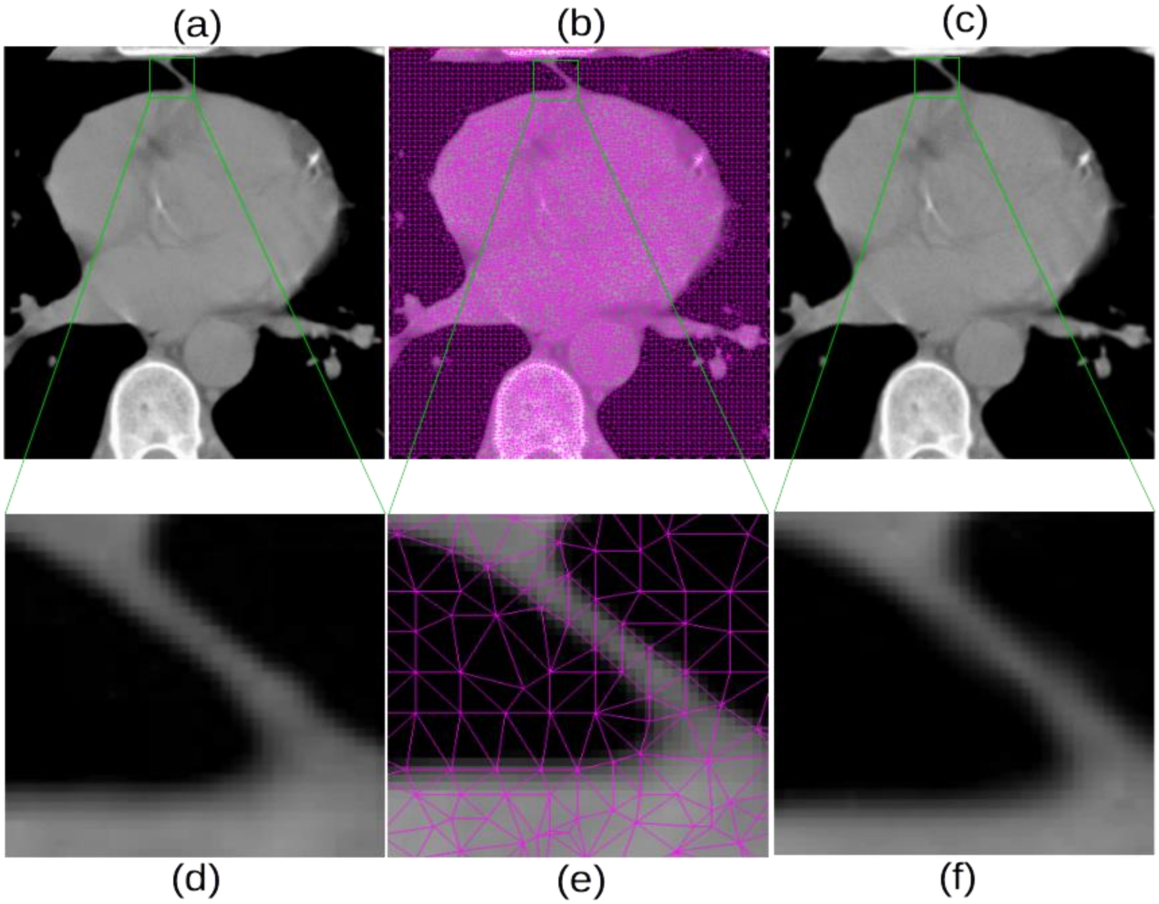


Figure 17 Details of CT-scanned image of heart. (a) Original heart image. (b) Generated mesh of (a). (c) Result of triangle-based ARBF interpolation using MQ basis. (d) - (f) are zoomed-in views of (a) - (c), respectively.

Figure 19 shows three biomedical images and the zoom-in areas of original, ARBF-5X enlarged, bicubic-5X enlarged results. The brain image is of size 285×341 and its triangular mesh has 5% sampling rate. The heart artery image is of size 356×396 and its triangular mesh has 4% sampling rate. The breast image is of size 512×512 and its triangular mesh has 4% sampling rate. As illustrated in Figure 19, while bicubic interpolation keeps the noises, the proposed method generates smoother images and sharper edges because of the following reasons: 1) during the triangulation, noisy pixels belongs to a triangle. Because this method uses intensity of triangle center, which is the average of three vertices, the noisy pixel contributes much less than other

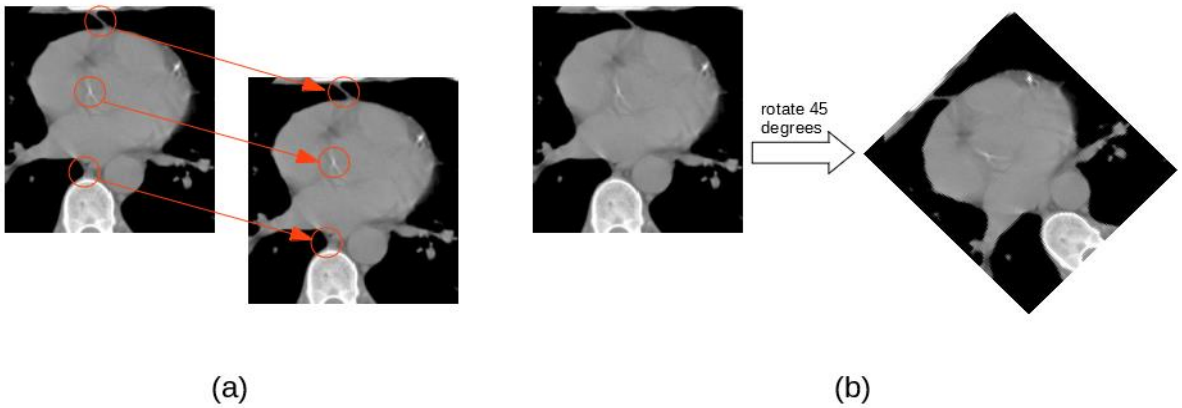


Figure 18 Isotropic image translation and rotation. After transformation, image dimensions are unchanged. (a) Image translated to another coordinates. Three red circles indicates image features before and after translation. (b) Left: original image. Right: image rotated 45°.

pixel-based interpolation methods. 2) the anisotropic distance described in Section 3.2.3 largely reduces the contribution of triangles located on other side of feature edges. These results are obtained by first performing 5X different SR interpolations, then use bicubic interpolation to return to original size. As the experiments shown, the proposed method uses mesh nodes whose amount are only 4%–5% of original pixels can produce enlarged smoothed images of visually satisfactory.

However, since the proposed method is based on triangular mesh interpolation, the SR result is largely determined by the mesh quality. Moreover, unlike multi-frame SR approach, single image interpolation approach does not have additional information provided thus the quality of the

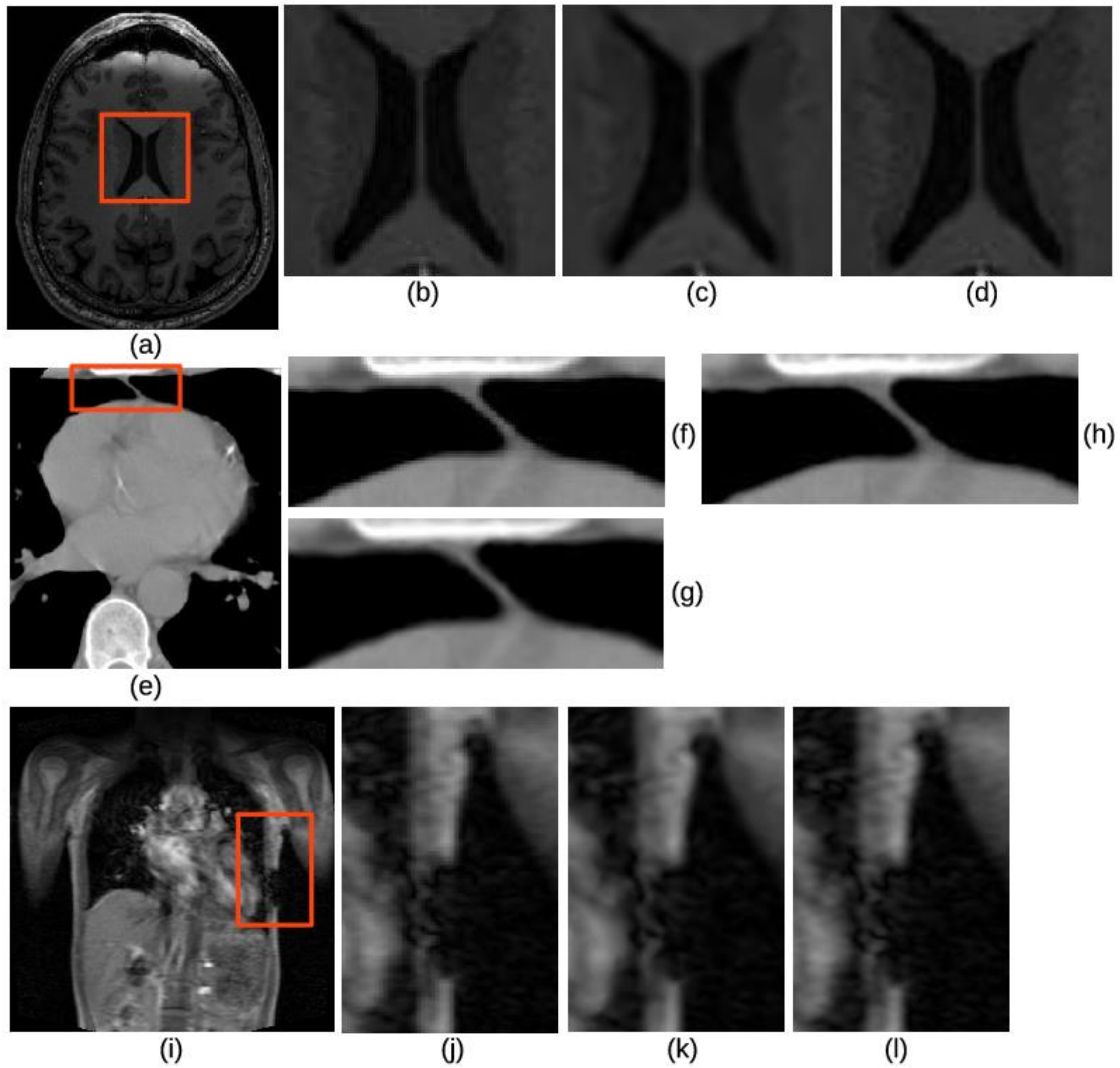


Figure 19 Super-resolution results of biomedical images. Results are rescaled for display. Red boxes show the zoom-in areas. (a) is the original brain MRI. (b) is the zoom-in part of (a). (c) is ARBF-5X SR result and (d) is bicubic-5X SR result. (e) is the original CT-scanned heart artery image. (f) is the zoom-in part of (e). (g) is ARBF-5X SR result. (h) is bicubic-5X SR result. (i) is the original breast MRI. (j) is zoom-in part of (i). (k) is ARBF-5X SR result. (l) is bicubic-5X SR result.

interpolation is very much limited by the original image and the lost frequency components cannot be recovered.

Chapter 4 Porous Structure Design in Tissue Engineering

4.1 Introduction

In order to improve biological tissues, tissue engineering (TE) which uses a scaffold to form new tissues for a medical purpose has a wide range of applications. Part of the applications in practice are repairing or replacing portion of or whole tissues and performing specific biochemical functions. Because of the inherent ability to produce customized porous scaffolds with different required architectures, the development of additive manufacturing (AM) techniques during last decade greatly improves tissue engineering. The latest ASTM standards defines AM as "a process of joining materials to make objects from three-dimensional (3D) model data, usually layer upon layer, as opposed to subtractive manufacturing methodologies" [140]. More specifically, additive manufacturing starts from a 3D computer model and builds the final product by the addition of material, usually from a layer-by-layer fashion. This is a new manufacturing techniques comparing to conventional subtractive processes which removes material from a 3D block. Commercial AM techniques to fabricate scaffolds for tissue engineering applications include selective laser sintering (SLS), stereo-lithography (SLA), fused deposition modeling (FDM), precision extrusion deposition (PED) and 3D printing (3DP). Interested readers can refer to [141] [142] [143] for details of these techniques.

Because native tissues are inherently heterogeneous and often have complex physiological architectures, literature, in practice, is primarily focused on the manufacturing of models which are simplified but functionally equivalent to the tissue to be repaired in terms of porosity and mechanical properties. Two types of porous scaffolds, namely regular porous scaffolds and irregular porous scaffolds, can be designed to achieve this goal. Numerous methodologies are proposed and categorized to fabricate these two types of scaffolds [144]. Table 3 categorizes these methods.

Type of Scaffolds	Methods
Regular porous scaffolds	CAD-based methods
	Image-based methods
	Implicit surface modeling (ISM)
	Space-filling curves
Irregular porous scaffolds	An optimization method proposed by [145]
	Stochastic methods using Voronoi models [146] [147]
	A hybrid Voronoi-spline method [148]
	Methods using volumetric meshes [149]

Table 3 Category of methods to design porous scaffolds in tissue engineering.

Periodic porous structures have a limitation that slight local modifications can affect the entire structure globally. Our proposed method provides a new implicit approach to generate porous tissue scaffold through volumetric mesh. Thus scaffold architectures can be adjusted by only modifying the mesh. Our proposed method has three major advantages over other implicit methods like TPMS-based ones. Firstly, local modifications of pore shape, size or distribution is achieved by changing local mesh accordingly, which is easy because there are only geometric changes in the mesh. Secondly, our method is flexible to simulate heterogeneities and discontinuities in natural tissue structures by using purposely-designed mesh as input. Depending on the features of tissue structure, meshes with different type (such as tetrahedron mesh and hexahedron mesh), size and density can be used to represent these characteristics. With different meshes as input, our method is able to build different tissue scaffolds with slight modifications in algorithms. Thirdly, unlike many implicit methods need post-actions like Boolean operations to get the final pieces built, the only post-action in our method is taking iso-surfaces, which is easier, faster and more flexible that user can get different scaffold architectures by taking different iso-values. In general, our method is superior in flexibility and easy to implement.

The rest of this chapter is organized as the following. Section 4.2 summarizes progresses so far of methods categorized in Table 3. Section 4.3.1 introduces porous scaffold reconstruction using conventional RBF interpolation. The proposed scaffold reconstruction is presented in Section 4.3.2. Section 4.3.3 shows the overall algorithms of our proposed method. Finally, Section 4.4 shows some experimental results and discussions. Section 4.5 concludes this chapter.

4.2 Prior Works

CAD-based methods, such as constructive solid geometry (CSG) and boundary representation (B-Rep), are used to design regular porous scaffolds. CSG-based tools combine standard solid primitives (cylinders, spheres or cubes) through Boolean operations (e.g. intersection) to design and represent complex models. B-Rep tools describe the solid cell through its boundaries by a set of vertices, edges and loops without explicitly specify relations among them. So a preliminary check is required to verify there are no gaps or overlaps among the boundaries [150]. However, as objects become large or their internal architectures become more complex, their size increases dramatically hence it is hard or impossible to visualize and manipulate them. To overcome this limitation of most CAD-based tools, different solid cells with more bio-inspired features have been introduced [151] [152].

Image-based methods combine imaging, image processing and free-form fabrication techniques to simplify scaffold design. Scaffolds can be described by 3D binary images (i.e. voxel values are Boolean and correspond to "solid" and "void"). Image-based methods produce scaffolds by taking the intersection of two 3D binary images, one representing the shape to be reproduced, and the other consisting of stacking of a binary unit cell. Empirically derived geometries are created in the unit cell with basic geometric shapes (cylinders, spheres) to represent regular pores within a scaffold. Randomly arranged pores can be obtained by the use of a random number generator to set voxel state. The topological optimization algorithms has been proved pivotal to obtain scaffolds in image-based methods [153] [154].

Implicit surface modeling is highly flexible and describes scaffold architecture by a single mathematical equation, with freedom to introduce different pore shapes, pore size gradients. Recently, a large class of periodic minimal surfaces methods such as triply periodic minimal surfaces (TPMS) have become attractive for the design of biomorphic scaffold architectures. An early attempt of using TPMS-based method to control tissue fabrication is presented by [155]. TPMSs like Schwartz's Primitive (type P), Schwartz's Diamond (type D) and Schoen's Gyroid (type G) are demonstrated their efficacy in high-precision fabrication of TPMS-based scaffolds [156] [157] [158]. However, all aforementioned works are limited to simple cubic or cylindrical outer shape. An improved method for constructing a pore network within an arbitrary complex anatomical model has been developed and successively optimized by Yoo et al. [159] [160]. In general, the TPMS methodologies, combining the advantages of both traditional CSG and image-based methods, are computationally efficient for modeling and fabrication of scaffolds.

Space-filling curves methodologies coupled to extrusion-based techniques. Such techniques consist of the extrusion of a micro-diameter polymeric filament terminating with a nozzle having an orifice diameter in the hundreds of microns range. The fabrication process involves the deposition of polymeric layers, which adhere to each other by heating temperature while retaining their shape. This process leads to regular repetition of identical pores. Thus these geometries have been named honeycomb-like patterns [161]. More complex patterns can be obtained by changing the deposition angle between adjacent layers. An alternative approach to design scaffold is using fractal space-filling curves, which can be mathematically generated by starting with a simple pattern that grows through the recursive rules.

Periodic porous structures have several advantages. They are easier to model and their structural properties are possible to predicate. Their disadvantages lie in the difficulty of controlling the pore shape, size and distribution since slight modification of the unit cell will pose global changes to the entire structure. Moreover, current CAD tools are not suitable to reproduce the complex natural structures. In scaffold with variational porous architectures, discontinuities of deposition path planning are often found at the interface of two adjacent regions [162]. To design such scaffold architecture, Khoda et al. implemented an optimization method [145]. Stochastic and Voronoi models have been used to generate random pores in scaffold design as well. Heterogeneous pores distributed according to a given porosity level are generated by stochastic methods in scaffold design [146] [147]. To overcome the limitation of only simple spheres can be used to represent pores, a hybrid Voronoi-spline representation combined with a random colloid-aggregation model is proposed [148]. The proposed method has been extended to implement graded pore sizes and pore distributions [163]. Volumetric mesh generators derived from finite element tools are used to create heterogeneous porosity within a solid model as well [149].

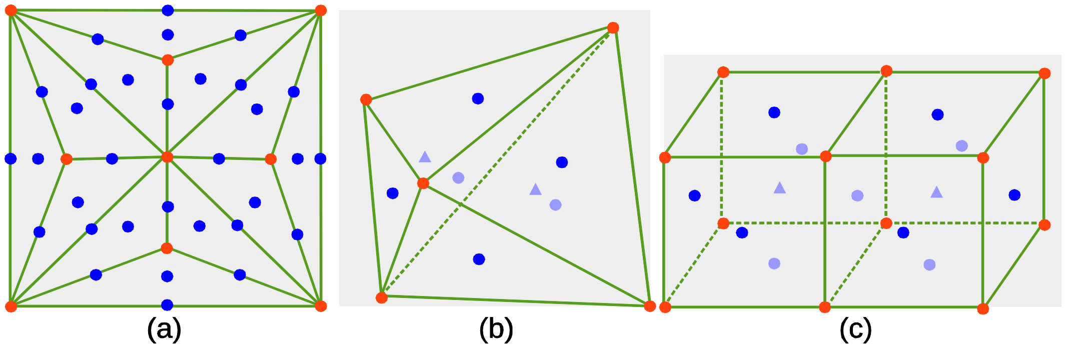


Figure 20 Values assigned to mesh nodes. Red dots represent value of 1. Blue dots represent value of -1. In 3D meshes, interior dots are represented by lighter colors. (a) Sample 2D triangle mesh. Note both edge and triangle centers are given values -1. (b) Sample 3D tetrahedron mesh. Small triangle represent tetrahedron centers. (c) Sample 3D hexahedron mesh. Small triangle represent hexahedron centers.

4.3 Methods

4.3.1 Radial Basis Function (RBF) Based Construction

In general, using the method mentioned in this chapter to construct a porous structure involves the following steps: 1) assign values to mesh nodes systematically to get ready for interpolation; 2) do RBF/ARBF interpolation; 3) iso-surface thresholding. After interpolation, iso-surfaces are taken on the interpolated piece to create porous architecture. In detail, for both 2D and 3D meshes, mesh vertices are given values 1. For 2D meshes, edge centers and tile (triangle or quadrangle) centers are given values -1. For 3D meshes, face centers and sub-volume (tetrahedron or hexahedron) centers are given values -1. Figure 20 (a-c) shows a sample 2D triangle mesh, a 3D tetrahedron mesh and a 3D hexahedron mesh with assigned values, respectively. For simplicity of drawing, 2D RBF interpolation scheme is illustrated by Figure 21 (a).

Conventional RBF seems viable to construct porous scaffolds. However, since RBF is isotropic in the sense that only geometrical distance is considered, the support domains of underlying basis

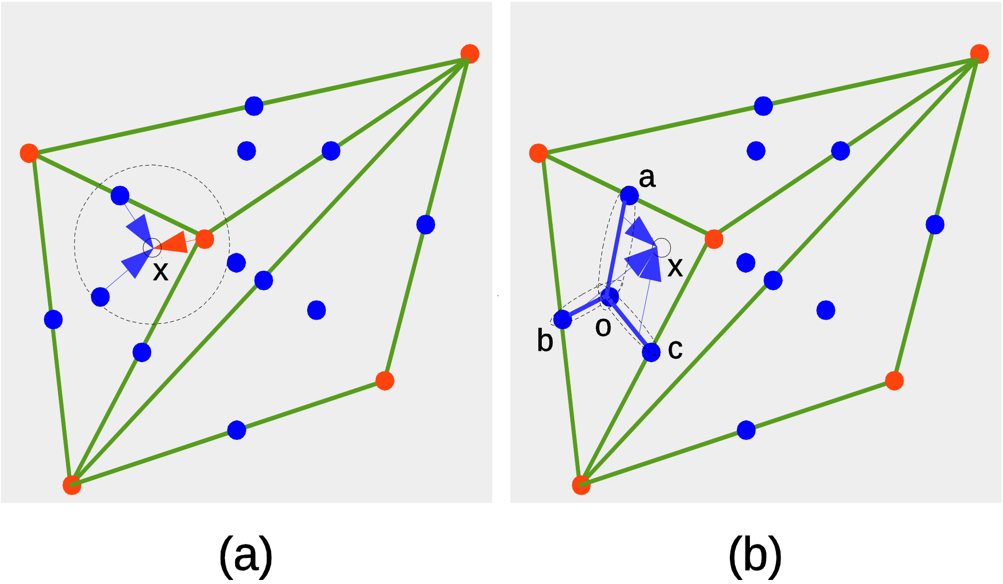


Figure 21 2D interpolation schemes. x is the pixel to be interpolated. Dashed circle (for RBF) or ellipses (for anisotropic RBF) are support domains of underlying basis functions. (a) RBF interpolation. (b) Anisotropic RBF interpolation

function always tend to be circular (in 2D) or spherical (in 3D), which sets an limitation to the customization of the internal architecture, especially for complex tissue scaffolds. Therefore, conventional RBF is only viable to generate simple architectures. Openings in desired scaffold architecture should start from sub-volume center and grows towards pores on structure surface. Given the condition that the face centers and sub-volume centers are assigned of value -1, the interpolated voxels should have values close to -1. Therefore, the internal opening directions have to be considered during interpolation and the shape of support domain should thus be anisotropic.

4.3.2 Anisotropic Radial Basis Function (ARBF) Interpolation

Comparing to image restoration application introduced in Section Chapter 3, the main difference between anisotropic RBF in image restoration application and in this work is the definition of distance used. Given N distinct line segments $L = \{L_j\}_{j=1,\ldots,N}$ the anisotropic radial basis function is defined by

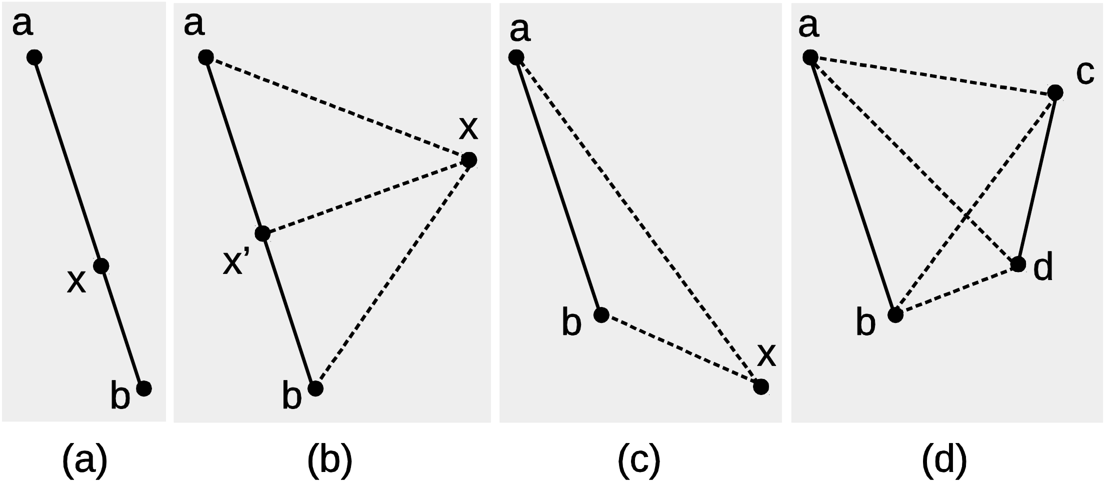


Figure 22 Cases to calculate anisotropic distance. (a) Point x is on line segment (a,b). (b) Point x and line segment (a,b) form an acute triangle then distance is defined as the length of $\|xx'\|$. (c) Point x and line segment (a,b) form an obtuse triangle. (d) Distance between line segment (a,b) and (c,d).

$$\Phi_{L,j}(\cdot) \coloneqq \phi\left(\left\|\cdot - l_j\right\|_L\right) \tag{16}$$

where $\|x\|_L$ is defined as the distance between any arbitrary point and a line segment or the distance between two line segments.

To calculate the distance between point x and line segment (a,b), there are three cases to consider. For case 1, if point x is on the line segment (a,b), distance is 0. For case 2, if point x, and (a,b) form an acute triangle, the distance is defined as the length of x's projection to (a,b). For case 3, if x, and (a,b) form an obtuse triangle, the distance is defined as $min\{\|xa\|, \|xb\|\}$. Figure 22 (a-c) illustrates the three cases.

The distance between any two arbitrary line segments (a,b) and (c,d) is defined as $min\{\|ac\|, \|ad\|, \|bc\|, \|bd\|\}$. Figure 22 (d) illustrates this case. Table 4 summaries these types of distances described above.

Types of Distance	Definition	Description
x to mesh nodes	$d = \|\cdot - x\|$	Conventional Euclidean distance
x to line segment (a, b)	$d = 0$	Case 1: x is on (a, b)
	$d = \|xx'\|$	Case 2: x and (a, b) form an acute triangle. x' is the projection of x on (a, b)
	$d = min\{\|xa\|, \|xb\|\}$	Case 3: x and (a, b) form an obtuse triangle
line segment (a, b) to line segment (c, d)	$d = min\{\|ac\|, \|ad\|, \|bc\|, \|bd\|\}$	Distance between two line segments

Table 4 Type of distances used in porous structure construction.

4.3.3 Algorithms

The following algorithm illustrates major steps to build porous structure from triangular (2D), tetrahedron and hexahedron (3D) meshes. The primary step is ARBF interpolation which consists of three sub-steps. Firstly, values are assigned to mesh nodes as explained in Section 4.3.1 to build the matrix. Secondly, the weight coefficients are solved by using the new distance definition explained in Section 4.3.2. Thirdly, the weights are used to interpolate the value at each voxel. After the piece is interpolated, an iso-surface is chosen and applied to get the final porous scaffold.

Algorithm: Porous Scaffold Construction

```
ConstructPorousScaffold()
{
     loadMesh(); // read mesh

     ARBFInterpolation(); // do the ARBF interpolation

     marchCube(); // take iso-surface

     exportResult(); // print result
}
ARBFInterpolation()
```

```
{
    assignMeshValues(); // assign mesh nodes and build matrix
   solveCoefficients(); // solve unknown weights
   applyCoefficientsToInterpolation(); // do the actual
   interpolation
}
```

4.4 Results and Discussion

This section includes results of scaffold shapes using different types of meshes and shows several experimental results. Figure 23 (a) shows a scaffold obtained from single tetrahedron. The opening grows from tetrahedron center toward the four triangle centers. Figure 23 (b) illustrates an icosahedron mesh comprised of 20 tetrahedrons. Figure 23 (c) shows the scaffold interpolated from Figure 23 (b). Figure 24 (a) shows a scaffold obtained from single hexahedron. Figure 24 (b) shows a scaffold obtained from a block-shape hexahedron mesh which is comprised of 8 smaller hexahedrons. Figure 24 (c) shows a scaffold obtained from a rod-shape hexahedron mesh which is comprised of 4 smaller hexahedrons. Input meshes for these results are regular so the final scaffolds are regular as well. The results are interpolated by ARBF interpolation introduced above using inverse multiquadrics (IMQ) as basis function and shape parameter of value 0.1. After interpolation, a proper iso-value is applied to the interpolated piece to show the final porous

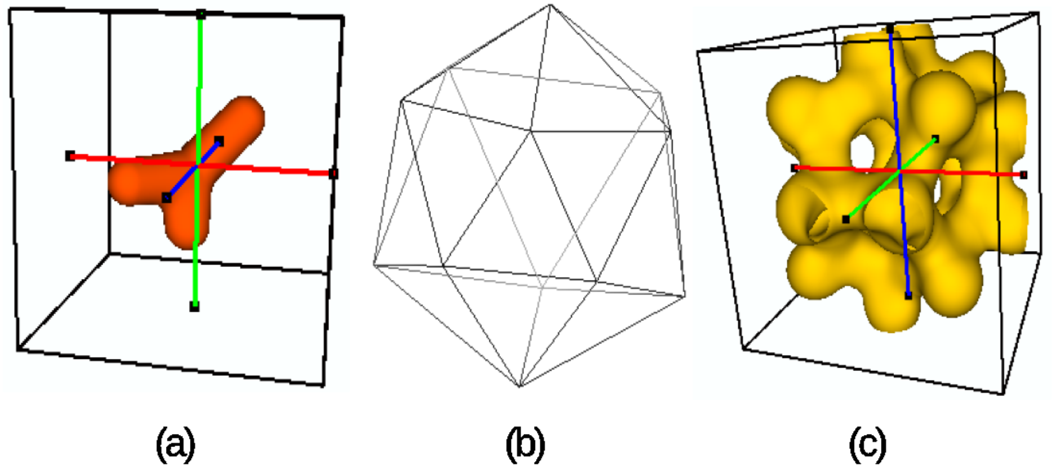


Figure 23 Results based on tetrahedron meshes. (a) Scaffold based on single tetrahedron. (b) Input icosahedron mesh (formed by 20 tetrahedrons). (c) Scaffold using (b) as input.

structure. Because the input meshes are regular, the output structures tend to be regular as well. Iso-values can be adjusted to get structure of different size of pores. Results of different iso-values are included in Figure 25. As the figure shows, size of pores increase as the iso-values increases. So after interpolation, choosing a proper iso-value is required to obtain the desired porous structure. Besides iso-values, basis function can also affect the porous architecture in terms of mostly the size and shape of pores. Because the shape of basis functions are different, when the final piece is

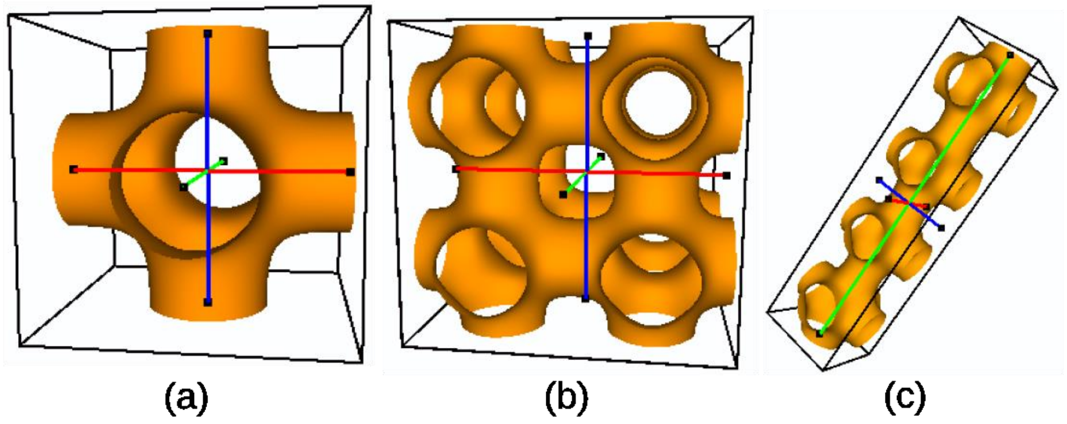


Figure 24 Results based on hexahedron meshes. (a) Scaffold based on single hexahedron. (b) Scaffold based on 8 hexahedrons arranged to form a large cube. (c) Scaffold based on 4 hexahedrons arranged to form a rod shape.

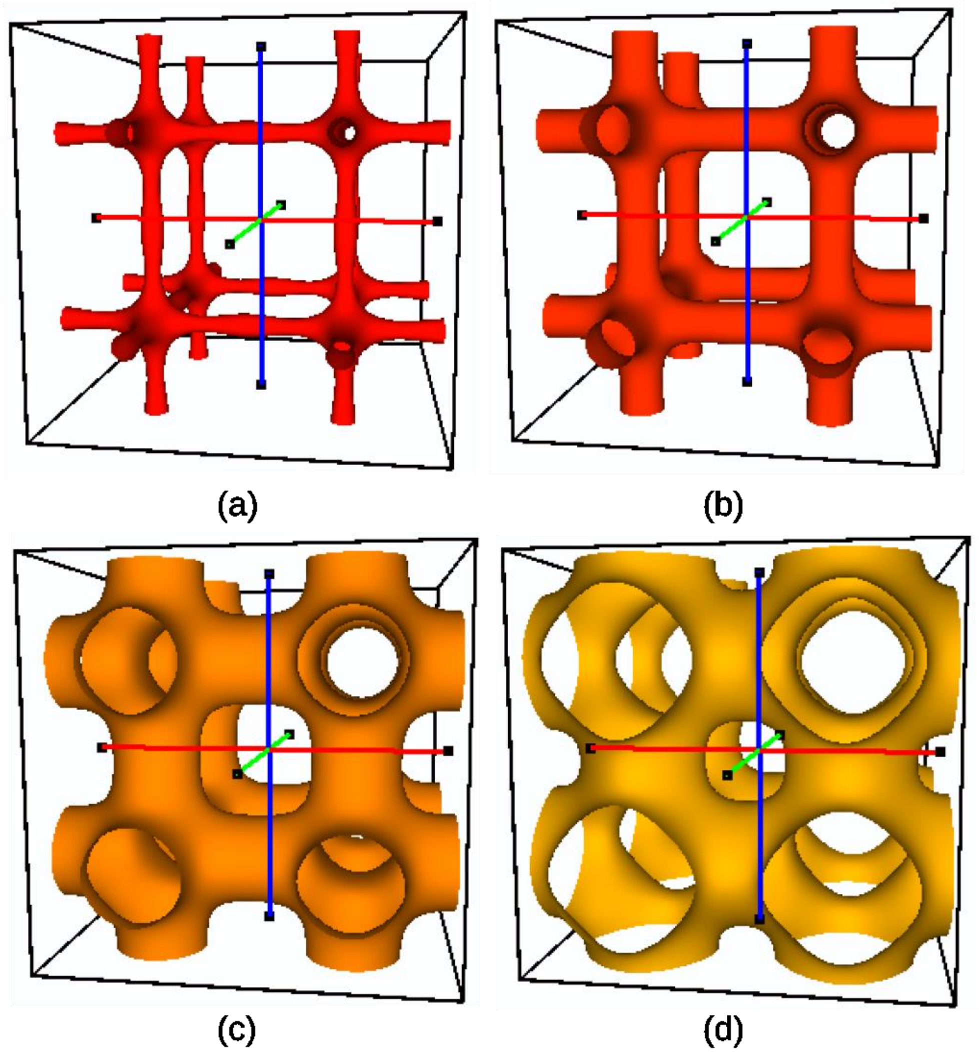

(a) (b) (c) (d)

Figure 25 Results taken by different iso-values. (a) - (d) are results taken by increasing iso-values.

interpolated, different voxels are included in their support domains. Interpolated results using different basis functions are included in Figure 26. To compare isotropic RBF interpolation and anisotropic RBF interpolation, a 2D isotropic RBF interpolated result (Figure 27 (a)) is also included. As explained before, because basis functions in isotropic RBF have circular support domains, the circle-shape artifacts can be seen in the result. Additionally, interpolation results based on disturbed meshes are also investigated and shown in Figure 27 (b-d). These disturbed

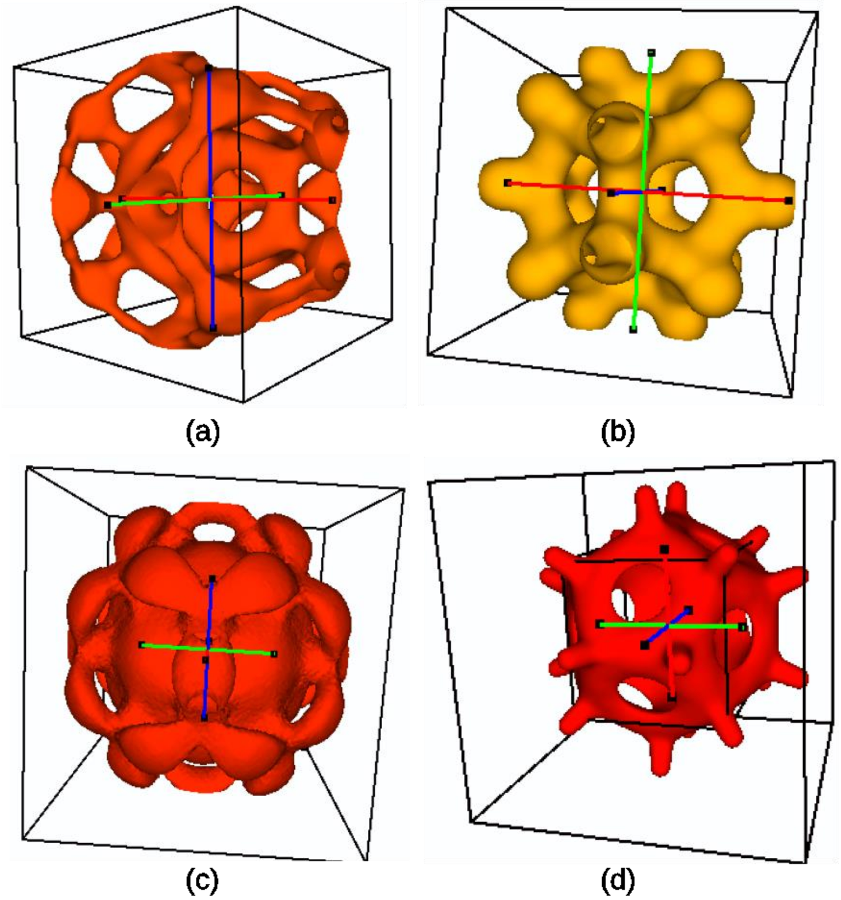


Figure 26 Results obtained by using different basis functions in ARBF interpolation. Shape parameter is 0.1 for all results. (a) Result interpolated by multiquadrics (MQ) basis. b) Result interpolated by inverse multiquadrics (IMQ) basis. (c) Result interpolated by Gaussian basis. (d) Result interpolated by thin plate spline (TPS) basis.

meshes are generated by moving some mesh vertices in random direction. As the results shown, disturbed porous structures can be obtained from disturbed meshes. Therefore, adding disturbance is a viable way to construct porous structure with randomness, which is very useful to construct tissues with complex physiological architectures. Finally, as a contrast, TPMS results are also included in Figure 28.

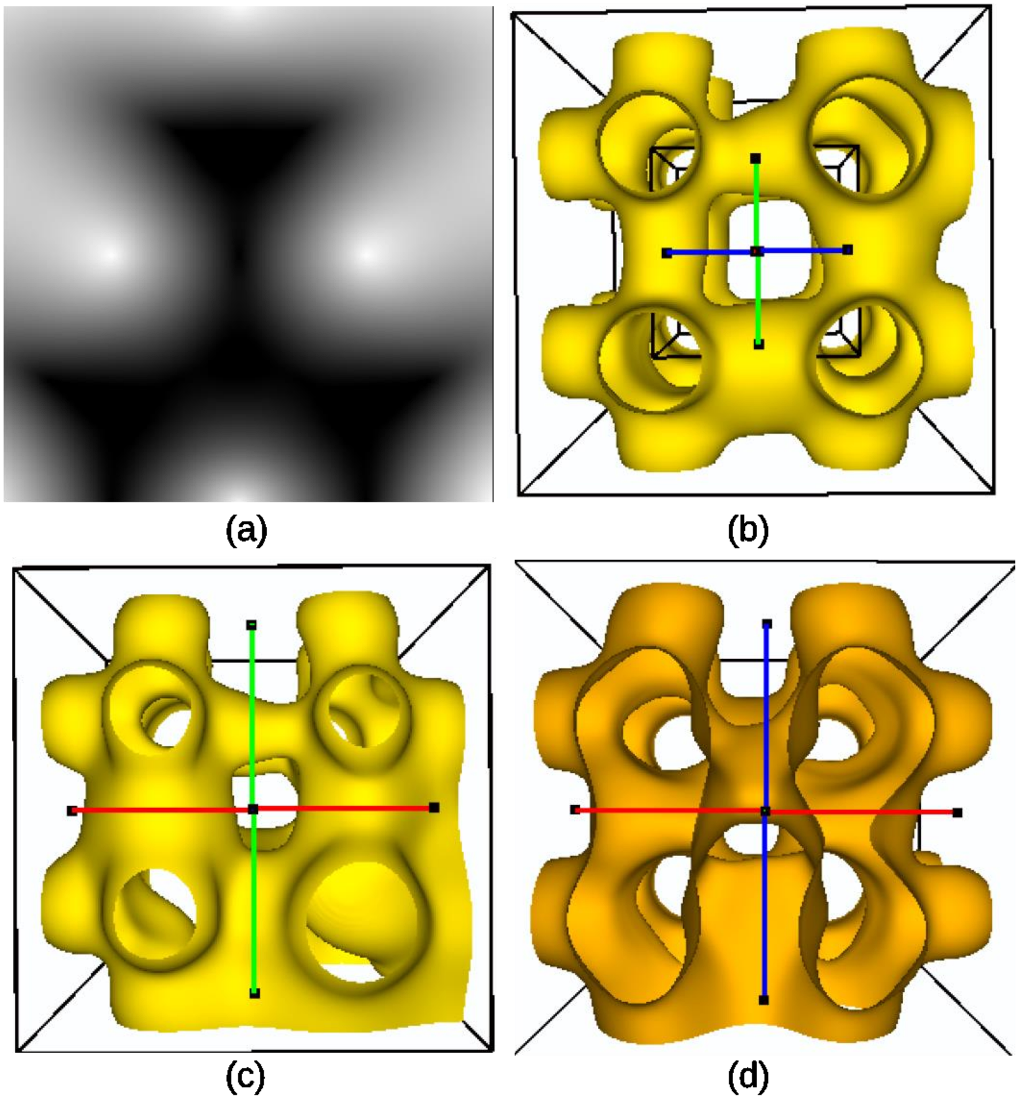


Figure 27 Experimental results. (a) Result interpolated by isotropic RBF and based on a 2D triangle mesh. (b-d) Results based on hexahedron meshes with disturbance.

4.5 Conclusions

In practical, constructed scaffold may be very complicated to simulate complex physiological tissue architecture in terms of equivalent internal connectivity and mass transportation. Periodic porous structure cannot meet this challenge satisfactorily. Our proposed method uses volumetric mesh as input which can be very complex – thanks to modern mesh modeling techniques. Thus,

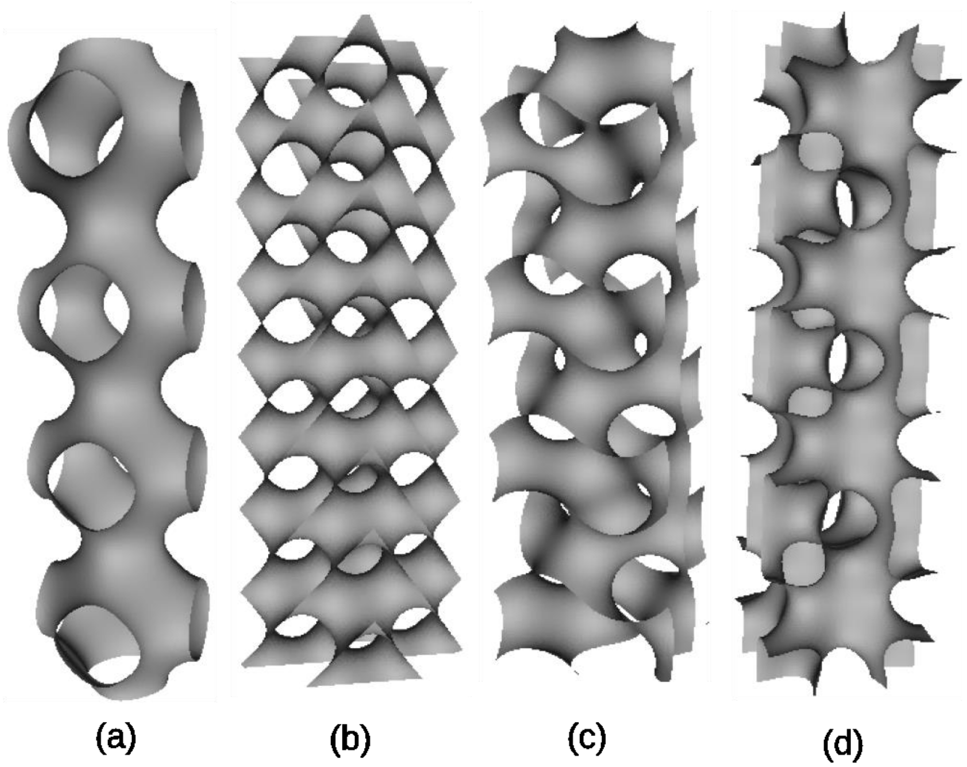


Figure 28 Porous structure obtained by TPMS method. (a) Structure obtained by P-type function. (b) Structure obtained by D-type function. (c) Structure obtained by G-type function. (d) Structure obtained by IWP-type function.

using complex mesh as input, our proposed method is able to construct complicated porous scaffold structures to meet this challenge. Moreover, modifications to the final structure can be achieved by common mesh operations or changing iso-value, which makes our method very flexible comparing to other period porous manufacturing techniques (like implicit surface methods). Finally, implementation of our proposed method is easy and computational cost is low because the core algorithm of interpolation is calculating distances and solving unknown weights. There are a lot of mature and fast linear algebra libraries available to use.

In the future, this method will be tested with adaptive meshes which represent porous architectures with heterogeneous and discontinuous structures. Moreover, a criteria to measure final scaffold will be developed to further test the efficacy and efficiency of this method.

Chapter 5 Parallelization

5.1 Introduction to GPGPU

In recent years, General-Purpose computing on Graphic Processing Unit (GPGPU) emerges as a new way to parallelize programs. There are three unique advantages of GPU, making it a very promising accelerator in computation. Firstly, GPU has large number of execution units (EUs). Usually hundreds, even thousands EUs exist on single chip, comparing to 2 or 8 EUs on CPU. Secondly, threads on GPU are very light-weighted that there are almost no cost on creating, destroying and switching threads. Therefore, it is possible to decompose the problem into hundreds of thousands, even millions sub-problems and mapping them onto the same number of threads. This is not possible because threads on CPU (including kernel threads and user threads) are heavy-weighted. There are large thread-contexts (hardware registers, kernel stacks, etc.) have to be saved when threads are switching out of execution and restored when they are switched back. Given that fact, the number of threads is usually limited by the number of EUs on CPU. Typical GPU can accelerate a program execution up to 20X - 100X faster. Thirdly, the price of GPU is cheap comparing to price of CPUs having similar computational power. A dedicated cluster is prohibitively expensive to most of researchers. Their price usually starts from 100K US dollars. Although there are some institute-owned clusters available, not every research can get access on them and the access availability is not guaranteed. A middle-level GPU however, costs only $100 - $300, making it a readily affordable device and once purchased, it is owned by the researcher.

5.2 Parallelization of Modeling of Calcium Dynamics

Section 5.2.3 listed the hardware and software environment for calcium dynamics experiment. This section discusses the implementation details.

5.2.1 Parallelization with OpenMP

Because the calculations of Laplacian terms are most time-consuming in the algorithm (see Figure 8 (b)), OpenMP is used to parallelize this step. The Laplacian terms have to be calculated for every point in the domain. OpenMP automatically divides the calculations into multiple chunks (4 chunks in this experiments because a quadcore CPU is used). Then it forks multiple threads and each thread calculates one chunk of work load. All threads execute the same algorithm but with different data. This is called data-parallelism. At the end of this step, all partial results are merged automatically and these working threads are synchronized implicitly. Because the overhead of thread-forking and thread-joining is larger than the advantages they can provide in other steps in the algorithm, OpenMP is not applied to other steps in the algorithm.

5.2.2 Parallelization with GPU

CUDA is an implementation of heterogeneous programming involving CPU (called host) and GPU (called device). The functions executed on GPU are called kernel functions. The execution starts with the serial host. When a kernel function is launched, the execution switches to the device (GPU), where a large number of threads are initiated to take advantage of abundant data parallelism. All the working threads in a kernel are organized in groups called thread blocks. When the kernel finishes executing, the control returns back to the host and the serial execution continues on the host until the next kernel launches or program terminates [164]. While the GPU is computing, CPU is available to do other tasks. In this implementation, however, CPU is simply waiting for the GPU to finish. The thread-switching on GPU has almost no overhead, so CUDA is able to create a one-to-one mapping between data points and threads. If the number of threads created is more than the number of CUDA cores available, they are scheduled to execute in batches. In execution, the number of scheduled threads is determined by several factors, including the number of CUDA cores available, the number of registers available, and the capacity of on-chip shared-memory. To achieve the best performance, a brief summary of some special techniques are considered in this implementation.

5.2.2.1 Use of Shared Memory

Because CUDA has thousands of threads running in parallel, the reads and writes to the off-chip device memory (called global memory) tend to have a memory bandwidth contention, which can significantly lower the performance. Threads in the same block, however, can share data in a small on-chip memory (called shared-memory). In this experiment, the calculation of Laplacian terms needs to read Laplacian coefficients and Ca^{2+} or $CaB_m, m = 1, 2, 3$ concentrations at neighboring points. To alleviate the global memory bandwidth contention, these variables are loaded into shared-memory once at the beginning of this step instead of multiple reads when needed. If the neighboring coefficients and concentrations are in the same block as the current point, they are fetched from shared-memory. Otherwise, they are loaded from global memory. The calculation of Laplacian terms is implemented as a kernel function. Unlike the global memory, however, the contents of shared-memory is not persistable across kernels. So at the end of this step, results have to be synchronized back to global memory for next kernel launches.

5.2.2.2 Reordering of Data Points

To increase the hit of shared-memory, we need to find a way to maximize the possibility that a point and its neighbors are all in the same block. If they are not in the same block, their coefficients and concentrations have to be loaded from global memory, which lowers the performance due to the global memory bandwidth contention. The number of thread blocks used is rounded up to $\left\lceil \frac{\text{number of points in a dataset}}{\text{number of threads per block}} \right\rceil$. Section 5.2.2.5 discusses details of block dimensions. When reading points, points in geometry are reorganized and grouped into many cells (blocks). The points in a block are grouped roughly in a cubic region such that the points and their neighbors are likely to be in the same block.

5.2.2.3 Use of Constant and Texture Memories

As stated in Section 5.1, the reads and writes of global memory have memory bandwidth contention. The constant and texture memories have built-in caches. When a program reads the same value more than once, it actually reads it from the cache, which eliminates the bandwidth problem. Constant and texture memories are read-only and constant memory is very small (only 64 KB shared by all stream-multiprocessors). There are dozens of constant arguments (refer to [1]) in constant memory. Texture memory shows a higher performance when data are localized with each other. Therefore, the neighborhood information found by the kd-tree is saved in texture memory.

5.2.2.4 Unrolling Loops

When dealing with iterations, programs on CPU use loops with specific components like programming counter, instruction decoder, etc. However, CUDA cores are simplified to have ALU (arithmetic-logic unit) only. Thus executing loops in CUDA is slower than CPU. If one can unroll a loop by storing involved variables into multiple registers instead of using one register and refresh that register at every iteration, the compiler can decode the instructions more efficiently and generate faster code. However, since the number of registers available is limited (GeForce GTX 560 Ti has 8 multiprocessors and each has 32,768 registers), the growing usage of registers could decrease the number of threads scheduled to execute and thus lowers the device occupancy. Device occupancy is determined by the equation $\frac{\text{number of threads scheduled}}{\text{warp size}}$, where the warp size is 32 in current CUDA-enabled GPUs. One may refer to [164] for more details. In this implementation, when unrolling loops increases the number of registers used by a single thread to 27, the device occupancy decreased from 50% to 33.3%. So there is a balance of increased code efficiency and decreased device occupancy. Whether to use this technique is really a case-by-case scenario. In this experiment, using this technique can improve the performance by about 10% in calculation of Laplacians.

5.2.2.5 Global Synchronization

Unlike the CPU, CUDA threads are scheduled to execute in batches. At a point requiring concentrations at neighboring points, it is likely that the threads responsible for the neighboring points are not executed yet and thus the neighboring concentrations are not updated. To cope with this situation, threads in GPU have to be synchronized. Unfortunately in the current GPU design, a kernel function can only synchronize threads in the same block. To synchronize threads in different blocks (global synchronization), a kernel function has to be split into smaller ones. In this algorithm, every step (see the 6 boxes on the right-hand side of Figure 8 (b)) is implemented as a kernel, yielding a total of 6 kernels. These kernels use different thread block dimensions in order to achieve maximum device occupancy. Because calculating Ca2+ flux and reaction terms does not require neighboring concentrations, a relatively small block size (128 threads per block) is used so that each thread can use more registers and shared-memory. Ca2+ flux is calculated on T-Tubule and cell membrane with a 58.3% device occupancy. Reaction terms are calculated on interior points with a 66.7% device occupancy. In applying reflective boundary condition, calculating Laplacian terms and calculating average concentration, shared-memory are used as high-speed cache. A large block size (512 threads per block) is used to increase the possibility of shared-memory hit. Reflective boundary condition is applied at boundary points at a 100% device occupancy. Laplacian terms and average concentrations are calculated at all points with a 33.3%

and 100% device occupancy respectively. At the end of each step, results are synchronized into global memory.

5.2.3 Experiment Environment

In this application, OpenMP and GPU are used to speed up the simulation and their performances are compared with the serial implementation. The CPU-based experiments (serial and OpenMP) are performed on a Dell Precision T7400 workstation with two quad-cores of 3.0 GHz Intel Xeon X5472 processors and 16 GB memory, a RedHat platform of kernel version 2.6.18-308.4.1.el5, and a GCC compiler of version 4.4.7. The serial experiment uses a single core and the OpenMP experiment uses four cores. The GPU-based experiments are performed on a Dell Precision T3500 workstation with a NVIDIA GeForce GTX 560 Ti, 2 GB device memory, and a CentOS 6.4 platform of kernel version 2.6.32-358.14.1.el6.x86_64. The NVIDIA graphic card driver has a version of 319.32. Details of implementation are discussed in section 5.2.

5.2.4 Results and Discussion

Column 1 in Figure 29 shows the average concentrations of Ca^{2+}, mobile and stationary buffers in three implementations, namely, serial, OpenMP and CUDA, per the legend displayed below the figure. Column 2 and Column 3 show their average relative errors compared to the serial implementation for point sets 2 and 3, respectively. Because an analytical solution does not exist for such a complicated mathematical model (see equations (3), (4), (5), (6)), we have used the numerical simulation from the serial LRBFCM approach [1] as the ground truth for comparison, which had been shown to agree with the finite element-based simulation [71] and with the available experiments as well. The errors in column 2 and column 3 show that OpenMP has exactly the same simulation results as the serial version. The CUDA result shows up to 0.5% relative error for the medium model (column 2) and 0.2% relative error for the large model (column 3). A close inspection discovers that the errors are partly caused by the fact that the GPU treats floating-point numbers differently from the CPU. The error curves also show that, as the number of points increases, the accuracy of CUDA implementation also increases.

Table 5 shows the running time of serial, OpenMP and CUDA on the three point sets. For a small model (set 1), OpenMP has the lowest performance because the overhead of multithreading is larger than the gains. CUDA has a better performance but only 3.66X faster than serial's speed. For a medium size of model (set 2), OpenMP and CUDA show 1.20X and 10.10X performance increases respectively comparing to serial implementation. For large model (set 3), OpenMP shows 1.5X performance increase and CUDA shows 19.82X performance increase. Apparently the GPU shows a great performance boost and is very promising in data parallelism because of a large number of cores available on GPUs.

However, GPU cores have simpler circuits compared to CPU cores and require special techniques like instruction-level optimization to get the best performance boost. Moreover, simpler circuits also restrict the applications of GPUs to tasks like data parallelism. The small device memory (usually 1 to 4 GB) also limits the amount of data a GPU can process simultaneously. For large input data, it is necessary to divide the data into multiple parts so that each part can be fitted into the device memory.

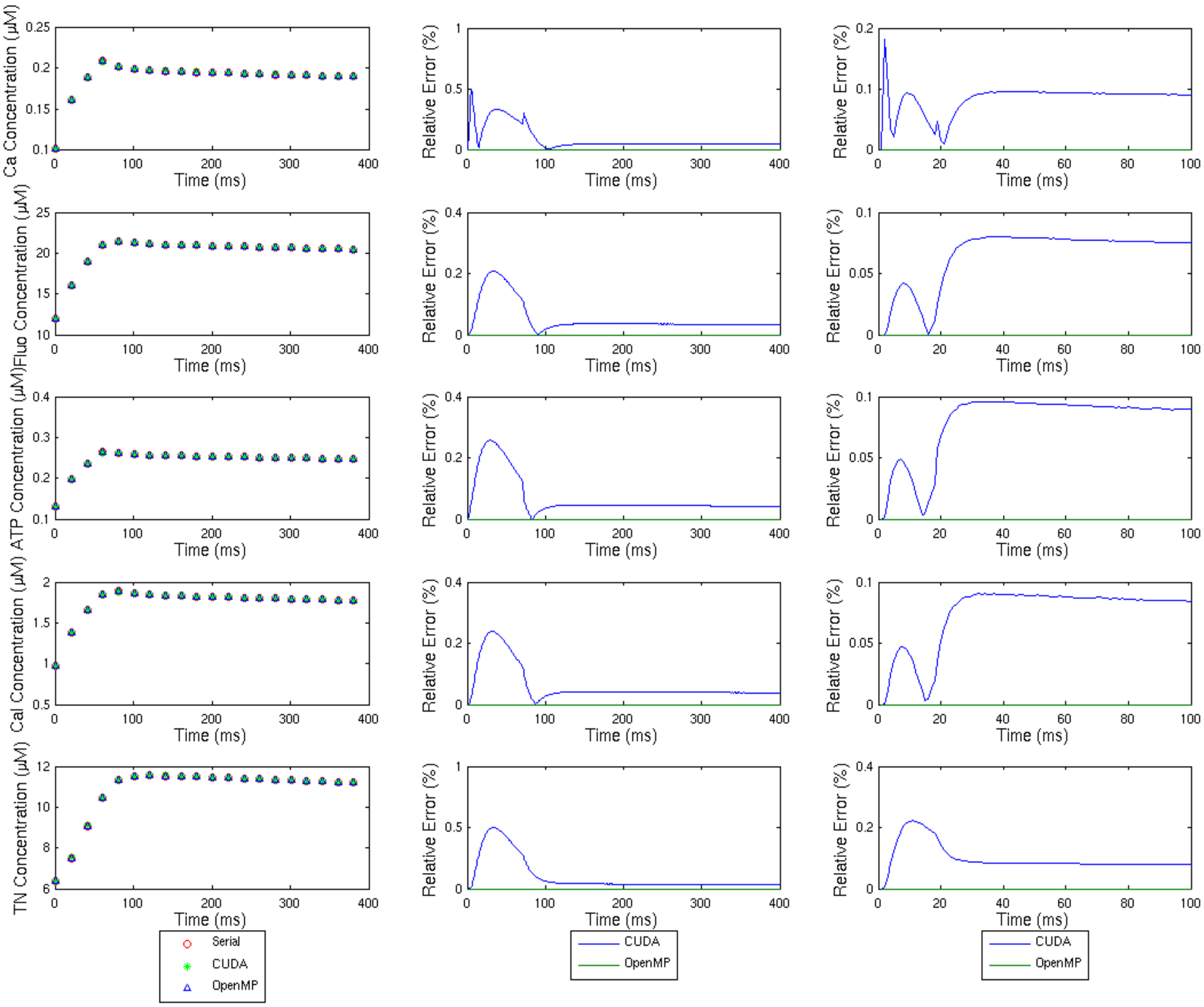


Figure 29 Column 1 shows the average concentrations of Ca^{2+}, mobile and stationary buffers for point set 2 (30,807 points) in three implementations, namely, serial, OpenMP and CUDA. Note that the concentrations of the three versions are almost identical. Column 2 shows the average relative errors of point set 2 (30,807 points), as compared to the serial execution. Column 3 shows the average relative errors of point set 3 (234,921 points), as compared to the serial execution.

Number of Points	Serial	OpenMP (4 cores)	CUDA
Point set 1 (3,969 points)	26.17	28.21	7.15
Point set 2 (30,807 points)	504.52	419.95	49.94

Point set 3 (234,921 points)	26893.21	17974.51	1356.85

Table 5 Running time of Serial, OpenMP, CUDA implementations on 3 point sets (unit: seconds)

5.3 Parallelization of Image Restoration with GPU

5.3.1 Methods

When implement the ARBF interpolation algorithm, CUDA is considered and experimented as a way to speed-up performance. The most time-consuming step of algorithm introduced in Section 3.2.4 is the ARBF interpolation. This step takes about 95.5% of total execution time of the algorithm. So naturally, this is the step needs to be parallelized.

In algorithm, the intensity of pixel in a given triangle is interpolated by the neighbors of that triangle. Blue dots in Figure 10 (b) illustrate this idea. The neighborhood size can be varied. In this work, 1-ring and 2-ring neighborhood are considered. The neighborhood of triangles is calculated before interpolation. The intensities of pixels are interpolated in a triangle-by-triangle fashion. In detail, triangles are iterated through and for each iteration, all pixels are interpolated by another nested loop. In this work, the outer loop of triangles are parallelized by CUDA, which means each triangle considered is mapped to a thread. Therefore, all triangles are able to be interpolated simultaneously.

5.3.2 Results and Discussion

One interesting thing is that the results do not depend on thread block size. After experiment with thread block size of 128, 256 and 512, the results are almost identical. Table 6 shows the experimental results.

Further profiling shows that two factors negatively impact the performance. Firstly, the cost of thread synchronization is high for every thread block. Because the size of neighborhood of each triangle is different, some thread requires more time to finish computation. On logic level, CUDA schedules execution by thread blocks. A thread block is scheduled in execution and has to be scheduled out at the same time. In detail on hardware level, threads are organized by warps, each of which has the size of 32. This means a batch of 32 threads has to be scheduled in and out of execution at the same instruction time. Different sizes of neighborhood require different execution time of threads. So threads with shorter execution time have to wait threads with longer execution time to be scheduled out of execution.

Data (image size)	Serial Time (sec.)	CUDA Time (sec.)	Maximum Device Occupancy	Speed-up
Brain (285×341)	7.56	2.85	33.3%	2.65X
Heart (356×396)	9.16	3.45	33.3%	2.66X
Breast (512×512)	17.43	6.81	33.3%	2.56X

Table 6 Execution time of image restoration. Results are the almost identical for block size of 128, 256 and 512.

Secondly, on-chip shared memory size also limits the number of threads that can be accommodated in a stream-multiprocessor (SMs). When threads are scheduled to execution, the scheduler decides how many thread blocks (and the threads in thread blocks) can be scheduled to execution based on various factors, such as compute capability of the device, size of shared memory, size of constant memory, the number of registers available on SM. The device used in this experiment can accommodate 1536 threads per SM but only have shared memory of size 48KB, which is the bottleneck. Therefore, only 33.3% of threads can be scheduled to execution for each SM. Different image data is experimented and the maximum device occupancy is the same. An attempt to solve this issue is reducing the number of threads in thread blocks. Different thread block dimensions are used, namely, 128, 256, 512. Shared memory limitation mitigates, but the number of registers available (65,535 for the device used) quickly becomes another limitation to increase the maximum device occupancy. In sum, the maximum device occupancy remains at 33.3%. Further investigation shows that the intrinsic logic of code implemented is the reason of so many registers are used.

Due to the two reasons discussed above, the GPU acceleration for image restoration is not so effective, comparing to the modeling of calcium dynamics application. In calcium dynamic modeling, the size of neighborhood for each point is a constant and hardware limitation is relatively low. Therefore the maximum device occupancy in calcium dynamic modeling is much higher (88.5%) and the acceleration is more effective (20X faster).

Chapter 6 Conclusions and Future Works

6.1 Thesis Summary

As the aforementioned applications proved, RBF is a powerful tool to reconstruct unknown function from multi-dimensional scattered data in biomedical area. Additionally, significant improvement in preserving image features and capturing connected porous structures over the original isotropic distance can be achieved by utilizing anisotropic distance metrics.

OpenMP and CUDA-based GPU programming are effective and efficient techniques to accelerate the performance of meshless PDE solver. However, as discussed in image restoration application, some intrinsic issues of algorithm such as varying sizes of neighborhood and hardware limitation like the number of registers and the size of shared memory can pose negative impacts on the efficiency of GPU parallelization.

6.2 Future Directions

In addition to the three applications and parallelization techniques discussed above, RBF can be used in wider areas and other parallel methods can be applied in current applications.

6.2.1 Applications of Radial Basis Function Networks

In the future, applications utilizing RBFN in deep learning area will be investigated. Current neural networks have multiple decision layers and complex to implement. RBFN, on the other hand, has only single layer thus is easier to implement but still effective to make decisions.

6.2.2 Porous Structure

The porous structures design application in thesis can also be extended and investigated further in several aspects.

6.2.2.1 Independent of Mesh

Current porous structures are constructed based on volumetric mesh, more specifically, tetrahedron meshes and hexahedron meshes. Although as proved above, mesh-based reconstruction algorithm is effective and has many advantages, there are disadvantages as well. One major drawback is the mesh itself. To use this algorithm, a mesh has to be generated and supplied. Mesh generation and the following optimization are processes consuming large time and human labor. Getting rid of meshes and using only a set of discrete points is a favorable improvement in the future.

6.2.2.2 Functionally Graded Porous Structure

Current porous structure design algorithm uses uniformly distributed mesh nodes, which is effective to construct structures with uniform porosities and distributions. However, natural bio-materials do not expose such architectures. Functionally graded materials (FGM) have been extensively proved to be effective in terms of transmitting matters in the past decades. In the future, a new approach to model porous structures with graded porosities and distributions will be designed.

6.2.2.3 Parallel Porous Structure

Parallel programming techniques are not applied to current porous structure design algorithm. Although the performance of current algorithm is acceptable, it may become unacceptable if more complex inputs are involved. OpenMP and CUDA will be applied to optimize the performance of current algorithm.

BIBLIOGRAPHY

[1] G. Yao and Z. Yu, "A localized meshless approach for modeling spatial-temporal calcium dynamics in ventricular myocytes," *International Journal for Numerical Methods in Biomedical Engineering,* vol. 28, no. 2, pp. 187-204, 2012.

[2] G. Kosec and B. Sarler, "Local RBF collocation method for darcy flow," *Computer Modeling in Engineering and Sciences,* vol. 25, no. 3, pp. 197-208, 2008.

[3] B. Sarler and R. Vertnik, "Meshfree explicit local radial basis function collocation method for diffusion problems," *Computers and Mathematics with Applications,* vol. 51, no. 8, pp. 1269-1282, 2006.

[4] R. Vertnik and B. Sarler, "Meshless local radial basis function collocation method for convective-diffusive solidliquid phase change problems," *International Journal of Numerical Methods for Heat and Fluid Flow,* vol. 16, no. 5, pp. 617-640, 2006.

[5] R. Vertnik and B. Sarler, "Solution of incompressible turbulent flow by a mesh-free method," *Computer Modeling in Engineering and Sciences,* vol. 44, no. 1, pp. 65-95, 2009.

[6] G. E. Fasshauer and J. G. Zhang, "On choosing “optimal” shape parameters for RBF approximation," *Numerical Algorithms,* vol. 45, no. 1-4, pp. 345-368, 2007.

[7] J. Wang and G. Liu, "On the optimal shape parameters of radial basis functions used for 2-d meshless methods," *Computer Methods in Applied Mechanics and Engineering,* vol. 191, pp. 2611-2630, 2002.

[8] T. Belytschko and T. Black, "Elastic crack growth in finite elements with minimal remeshing.," *International Journal for Numerical Methods in Engineering,* vol. 45, no. 5, pp. 503-620, 1999.

[9] S. Bordas, J. Conley, B. Moran, J. Gray and E. Nichols, "A simulation-based design paradigm for complex cast components," *Engineering with Computers,* vol. 23, no. 1, pp. 25-37, 2007.

[10] S. Bordas and B. Moran, "Enriched finite elements and level sets for damage tolerance assessment of complex structures.," *Engineering Fracture Mechanics,* vol. 73, no. 9, pp. 1176-1201, 2006.

[11] S. Bordas, V. Nguyen, C. Dunant, H. Nguyen-Dang and A. Guidoum, "An extended finite element library," *International Journal for Numerical Methods in Engineering,* vol. 71, pp. 703-732, 2008.

[12] L. Lucy, "A numerical approach to the testing of the fission hypothesis," *Astronomical Journal,* vol. 82, pp. 1013-1024, 1977.

[13] R. Gingold and J. Monaghan, "Smoothed particle hydrodynamics - Theory and application to non-spherical stars.," *Monthly Notices of the Royal Astronomical Society,* vol. 181, pp. 375-389, 1977.

[14] L. Libersky, A. Petschek, T. Carney, J. Hipp and F. Allahdadi, "High Strain Lagrangian Hydrodynamics: A Three-Dimensional SPH Code for Dynamic Material Response.," *Journal of Computational Physics,* vol. 109, no. 1, pp. 67-75, 1993.

[15] T. Belytschko, K. Y., D. Organ, M. Fleming and P. Krysl, "Meshless methods: An overview and recent developments.," *Computer Methods in Applied Mechanics and Engineering,* vol. 139, no. 1-4, pp. 3-47, 1996.

[16] J. Bonet and S. Kulasegaram, "Correction and stabilization of smooth particle hydrodynamics methods with applications in metal forming simulations," *International Journal for Numerical Methods in Engineering,* vol. 47, no. 6, pp. 1083-1240, 2000.

[17] J. Bonet and T. Lok, "Variational and momentum preservation aspects of smooth particle hydrodynamic formulations.," *Computer Methods in Applied Mechanics and Engineering,* vol. 180, no. 1-2, pp. 97-115, 1999.

[18] G. Johnson and S. Beissel, "Normalized smoothing functions for sph impact computations.," *International Journal for Numerical Methods in Engineering,* vol. 39, no. 16, pp. 2695-2864, 1996.

[19] G. Johnson, S. Beissel and R. Stryk, "A generalized particle algorithm for high velocity impact computations.," *Computational Mechanics,* vol. 25, no. 2, pp. 245-256, 2000.

[20] T. Belytschko, Y. Lu and L. Gu, "Element-free Galerkin methods.," *International Journal for Numerical Methods in Engineering,* vol. 37, no. 2, pp. 181-358, 1994.

[21] W. Liu, S. Jun and Y. Zhang, "Reproducing kernel particle methods.," *International Journal for Numerical Methods in Engineering,* vol. 20, pp. 1081-1106, 1995.

[22] J. Melenk and B. I. , "The partition of unity finite element method: Basic theory and applications.," *Computer Methods in Applied Mechanics and Engineering,* vol. 139, no. 1-4, pp. 289-314, 1996.

[23] S. Atluri and S. Shen, "The Meshless Local Petrov-Galerkin (MLPG) Method: A Simple & Less-costly Alternative to the Finite Element and Boundary Element Methods.," *Computer Modeling in Engineering and Sciences,* vol. 3, no. 1, 2002.

[24] R. Loehner, C. Sacco and E. Onate, "A finite point method for compressible flow.," *International Journal of Numerical Methods in Engineering,* vol. 53, pp. 1765-1779, 2002.

[25] E. Onate and S. Idelsohn, "A mesh-free finite point method for advective-diffusive transport and fluid flow problems.," *Computational Mechanics,* vol. 21, no. 4, pp. 283-292, 1998.

[26] E. Onate, S. Idelsohn, O. Zienkievicz and R. Taylor, "finite point method in computational mechanics: applications to convective transport and fluid flow.," *International Journal for Numerical Methods in Engineering,* vol. 39, no. 22, pp. 3761-3931, 1996.

[27] S. Atluri, The Meshless Method (MLPG) for Domain & BIE Discretizations., Tech Science Press, 2004.

[28] Y. Chen, J. Lee and A. Eskandarian, Meshless Methods in Solid Mechanics., Springer, 2006.

[29] A. Ferreira, E. Kansa, G. Fasshauer and V. Leito, Progress on Meshless Methods., Springer, 2008.

[30] M. Griebel and M. Schweitzer, Meshfree Methods for Partial Differential Equations., Springer, 2002.

[31] M. Griebel and M. Schweitzer, Meshfree Methods for Partial Differential Equations II., Springer, 2005.

[32] M. Griebel and M. Schweitzer, Meshfree Methods for Partial Differential Equations III., Springer, 2006.

[33] M. Griebel and M. Schweitzer, Meshfree Methods for Partial Differential Equations IV., Springer, 2008.

[34] V. Leito, C. Alves and C. Durate, Advances in Meshfree Techniques., Springer, 2007.

[35] G. Liu, Mesh Free Methods: Moving Beyond the Finite Element Method., CRC Press, 2002.

[36] G. Liu and Y. Gu, An Introduction to Meshfree Methods and Their Programming., Springer, 2005.

[37] J. Sladek and V. Sladek, Advances in Meshless Methods., Tech Science Press, 2006.

[38] V. Nguyena, T. Rabczuk, S. Bordas and M. Duflot, "Meshless methods: A review and computer implementation aspects.," *Mathematics and Computers in Simulation,* vol. 79, pp. 763-813, 2008.

[39] C. Giordano, J. Beccaria, C. Goicoechea and C. Olivieri, "Optimization of the hydrolysis of lignocellulosic residues by using radial basis functions modeling and particle swarm optimization," *Biochemical Engineering Journal,* vol. 80, pp. 1-9, 2013.

[40] F. Bernardini, C. Bajaj, J. Chen and D. Schikore, "Automatical reconstruction of 3D CAD models from digital scans.," *International Journal of Computational Geometry & Applications,* vol. 9, no. 4, 1999.

[41] J. Carr, W. Fright and R. Beatson, "Surface interpolation with radial basis functions for medical imaging.," *IEEE Transactions of Medical Imaging,* vol. 16, no. 1, pp. 96-107, 1997.

[42] V. Savchenko, A. Pasko, O. Okunev and T. Kunii, "Function Representation of Solids Reconstructed from Scattered Surface Points and Contours.," *Computer Graphics Forum,* vol. 14, no. 4, pp. 181-188, 1995.

[43] G. Turk and J. O’Brien, "Shape transformation using variational implicit surfaces.," in *SIGGRAPH’99*, 1999.

[44] G. Turk and J. O’Brien, "Variational implicit surfaces.," Georgia Institute of Technology, 1999.

[45] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum and T. Evans, "Reconstruction and representation of 3D objects with radial basis functions.," in *SIGGRAPH '01*, 2001.

[46] Wikipedia, "Wikipedia," [Online]. Available: http://www.anc.ed.ac.uk/rbf/intro/node8.html.

[47] M. Madan, J. Liang and H. Noriyasu, Radial Basis Function Neural Networks, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2005.

[48] D. Broomhead and D. Lowe, "M ultivar iable Functional Interpolation and Adaptive Networks," *Complex Systems,* vol. 2, pp. 321-335, 1988.

[49] F. Girosi, "Some extensions of radial basis functions and their applications in artificial intelligence.," *Computers and Mathematics with Applications,* vol. 24, no. 12, pp. 61-80, 1992.

[50] M. Varvak, "Ellipsoidal/radial basis functions neural networks enhanced with the Rvachev function method in application problems.," *Engineering Applications of Artificial Intelligence,* vol. 38, pp. 111-121, 2015.

[51] F. Fernández-Navarro, C. Hervás-Martínez, P. Gutiérrez, J. Peña-Barragán and F. López-Granados, "Parameter estimation of q-Gaussian Radial Basis Functions Neural Networks with a Hybrid Algorithm for binary classification.," *Neurocomputing,* vol. 75, no. 1, pp. 123-134, 2012.

[52] R. J., S. Kiranyaz and M. Gabbouj, "Training Radial Basis Function Neural Networks for Classification via Class-Specific Clustering.," *IEEE Transactions on Neural Networks and Learning Systems* , vol. 27, no. 12, pp. 2458-2471, 2015.

[53] I. Maglogiannis, H. Sarimveis, C. Kiranoudis, A. Chatziioannou, N. Oikonomou and V. Aidinis, "Radial Basis Function Neural Networks Classification for the Recognition of Idiopathic Pulmonary Fibrosis in Microscopic Images," *IEEE Transactions on Information Technology in Biomedicine,* vol. 12, no. 1, pp. 42-54, 2008.

[54] I. Keramitsoglou, H. Sarimveis, C. Kiranoudis and N. Sifakis, "Radial basis function neural networks classification using very high spatial resolution satellite imagery: an application to the habitat area of Lake Kerkini (Greece).," *International Journal of Remote Sensing,* vol. 26, no. 9, pp. 1861-1880, 2005.

[55] G. Sermpinis, K. Theofilatos, A. Karathanasopoulos, E. Georgopoulos and C. Dunis, "Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and Particle Swarm Optimization," *European Journal of Operational Research,* vol. 225, no. 3, pp. 528-540, 2013.

[56] G. Sideratos and N. Hatziargyriou, "Probabilistic Wind Power Forecasting Using Radial Basis Function Neural Networks.," *IEEE Transactions on Power Systems,* vol. 27, no. 4, pp. 1788-1796, 2012.

[57] Z. Guo, H. Wang, J. Yang and D. Miller, "A Stock Market Forecasting Model Combining Two-Directional Two-Dimensional Principal Component Analysis and Radial Basis Function Neural Network," *PLoS ONE,* vol. 10, no. 4, pp. 1-19, 2015.

[58] Wikipedia, "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Parallel_computing#Bit-level_parallelism.

[59] L. L. N. L. Blaise Barney, "OpenMP," [Online]. Available: https://computing.llnl.gov/tutorials/openMP/.

[60] Nvidia, "CUDA C Programming Guide," [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/.

[61] D. Marr, F. Binns, D. Hill, G. Hinton, D. Koufaty, A. Miller and M. Upton, " Hyper-threading technology architecture and microarchitecture.," *Intel Technology Journal,* vol. 06, no. 01, 2002.

[62] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, M. CC., C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney and J. Zook, "TILE64 processor: a 64-core SoC with mesh interconnect.," in *IEEE International Solid-State Circuits Conference,* 2008.

[63] M. Gschwind, "Chip multiprocessing and the cell broadband engine.," in *Proceedings of the 3rd Conference on Computing Frontiers, CF ’06, ACM,* New York, USA, 2006.

[64] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone and J. Phillips, "GPU Computing," *Proceedings of the IEEE* , vol. 96, no. 5, pp. 879-899, 2008.

[65] G. D’Angelo and M. Marzolla, "New trends in parallel and distributed simulation: From many-cores to Cloud Computing.," *Simulation Modelling Practice and Theory*, vol. 49, pp. 320-335, 2014.

[66] D. M. Bers, "Calcium cycling and signaling in cardiac myocytes," *Annual Review of Physiology*, vol. 70, pp. 23-49, 2008.

[67] D. M. Bers, "Cardiac excitation-contraction coupling," *Nature*, vol. 415, no. 6868, pp. 198-205, 2002.

[68] A. Michailova, F. DelPrincipe, M. Egger and E. Niggli, "Spatiotemporal features of Ca2+ buffering and diffusion in atrial cardiac myocytes with inhibited sarcoplasmic reticulum," *Biophysical Journal*, vol. 83, no. 6, pp. 3134-3151, 2002.

[69] X. Koh, B. Srinivasan, H. S. Ching and A. Levchenko, "A 3D monte-carlo analysis of the role of dyadic space geometry in spark generation," *Biophysical Journal*, vol. 90, no. 6, pp. 1999-2014, 2006.

[70] L. T. Izu, S. A. Means, J. N. Shadid, Y. Chen-Izu and C. W. Balke, "Interplay of ryanodine receptor distribution and calcium dynamics," *Biophysical Journal*, vol. 91, no. 1, pp. 95-112, 2006.

[71] S. Lu, A. Michailova, J. Saucerman, Y. Cheng, Z. Yu, R. Bank, T. Kaiser, W. Li, M. Holst, J. McCammon, T. Hayashi, P. Arzberger, A. McCulloch, Y. Cheng and M. Hoshijima, "Multiscale modeling in rodent ventricular myocytes," *Engineering in Medicine and Biology Magazine IEEE*, vol. 28, pp. 46-57, 2009.

[72] C. Soeller and M. B. Cannell, "Examination of the transverse tubular system in living cardiac rat myocytes by 2-photon microscopy and digital image processing techniques," *Circulation Research,* vol. 84, no. 3, pp. 266-275, 1999.

[73] M. Pasek, F. Brette, A. Nelson, C. Pearce, A. Qaiser, C. Orchard and G. Christe, "Quantification of t-tubule area and protein distribution in rat cardiac ventricular myocytes," *Progress in Biophysics and Molecular Biology,* vol. 96, no. 1-3, pp. 244-257, 2008.

[74] R. Hinch, J. L. Greenstein, A. J. Tanskanen, L. Xu and R. Winslow, "A simplified local control model of calciuminduced calcium release in cardiac ventricular myocytes," *Biophysical Journal,* vol. 87, no. 6, pp. 3723-3736, 2004.

[75] G. E. Fasshauer, Meshfree Approximation Methods with MATLAB, Singapore: World Scientific Press, 2007.

[76] E. Divo and A. Kassab, "An efficient localized RBF meshless method for fluid flow and conjugate hear transfer," *ASME Journal of Heat Transfer,* vol. 129, pp. 124-136, 2007.

[77] K. Liu, G. Ye and Z. Yu, "Porous Sturcture Design in Tissue Engineering Using Anisotropic Radial Basis Function.," in *arXiv:1612.01944 [cs.GR]*, 2016.

[78] K. Aizawa and T. Huang, "Model-based image coding: advanced video coding techniques for very low bit-rate applications," in *Proceedings of the IEEE*, 1995.

[79] H. Benoit-Cattin, P. Joachimsmann, A. Planat, S. Valette, A. Baskurt and R. Prost, "Active mesh texture coding based on warping and DCT," in *IEEE International Conference on Image Processing*, Kobe, Japan, 1999.

[80] F. Davoine, M. Antonini, J. Chassery and M. Barlaud, "Fractal image compression based on Delaunay triangulation and vector quantization," *IEEE Transactions on Image Processing,* vol. 5, pp. 338-346, 1996.

[81] L. Demaret, G. Robert, N. Laurent and A. Buisson, "Scalable image coder mixing DCT and triangular meshes," in *IEEE International Conference on Image Processing*, Vancouver, BC, Canada, 2000.

[82] Y. Altunbasak and A. Tekalp, "Closed-form connectivity-preserving solutions for motion compensation using 2-d meshes," *IEEE Transactions on Image Processing,* vol. 6, no. 533, pp. 1255-1269, 1997.

[83] P. Hsu, K. Liu and T. Chen, "A low bit-rate video codec based on two-dimensional mesh motion compensation with adaptive interpolation," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 11, pp. 111-117, 2001.

[84] G. Marquant, S. Pateux and C. Labit, "Mesh and "crack lines": application to object--based motion estimation and higher scalability," in *IEEE International Conference on Image Processing*, Vancouver, BC, Canada, 2000.

[85] A. Nosratinia, "New kernels for fast mesh-based motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 11, pp. 40-51, 2001.

[86] C. Toklu, A. Tekalp and A. Erdem, "Semi-automatic video object segmentation in the presence of occlusion," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 10, pp. 624-629, 2000.

[87] Y. Wang and O. Lee, "Active mesh - a feature seeking and tracking image sequence representation scheme," *IEEE Transactions on Image Processing,* vol. 3, pp. 610-624, 1994.

[88] M. Garcia and B. Vintimilla, "Acceleration of filtering and enhancement operations through geometric processing of gray-level images," in *IEEE International Conference on Image Processing,* Vancouver, BC, Canada, 2000.

[89] A. Singh, D. Terzopoulos and D. Goldgof, Deformable models in medical image analysis, IEEE Computer Society Press, 1998.

[90] A. Baghaie, R. D'souza and Z. Yu, "Application of Independent Component Analysis techniques in speckle noise reduction of retinal OCT images.," *Optik-International Journal for Light and Electron Optics,* vol. 127, no. 15, pp. 5783-5791, 2016.

[91] A. Baghaie, "Markov Random Field Model-Based Salt and Pepper Noise Removal.," in *arXiv preprint arXiv:1609.06341*, 2016.

[92] A. Baghaie and Z. Yu, "Curvature-based registration for slice interpolation of medical images.," in *International Symposium Computational Modeling of Objects Represented in Images,* 2014.

[93] S. Coleman, B. Scotney and M. Herron, "Image feature detection on content-based meshes," in *Proceedings of IEEE International Conference on Image Processing*, 2002.

[94] M. Petrou, R. Piroddi and A. Talebpour, "Texture recognition from sparsely and irregularly sampled data," *Computer Vision and Image Understanding*, vol. 102, pp. 95-104, 2006.

[95] M. Sarkis and K. Diepold, "A fast solution to the approximation of 3-D scattered point data from stereo images using triangular meshes," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, Pittsburgh, PA, USA, 2007.

[96] A. Baghaie, R. D'souza and Z. Yu, "Dense correspondence and optical flow estimation using gabor, schmid and steerable descriptors.," in *International Symposium on Visual Computing*, 2015.

[97] J. Brankov, Y. Yang and N. Galatsanos, "Image restoration using content-adaptive mesh modeling," in *Proceedings of IEEE International Conference on Image Processing*, 2003.

[98] J. Brankov, Y. Yang and M. Wernick, "Tomographic image reconstruction based on a content-adaptive mesh model," *IEEE Transactions on Medical Imaging*, vol. 23, pp. 202-212, 2004.

[99] D. Su and P. Willis, "Demosaicing of color images using pixel level data-dependent triangulation," in *Proceedings of Theory and Practice of Computer Graphics*, 2003.

[100] D. Su and P. Willis, "Image interpolation by pixel-level data-dependent triangulation," in *Computer Graphics Forum*, 2004.

[101] A. Baghaie and Z. Yu, "Structure tensor based image interpolation method.," *AEU-international Journal of Electronics and Communications,* vol. 69, no. 2, pp. 515-522, 2015.

[102] M. Adams, "An efficient progressive coding method for arbitrarily-sampled image," *IEEE Signal Processing Letters,* vol. 15, pp. 629-632, 2008.

[103] M. Adams, "Progressive lossy-to-lossless coding of arbitrarily-sampled image data using the modified scattered data coding method," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Taipei, Taiwan, 2009.

[104] K. Hung and C. Chang, "New irregular sampling coding method for transmitting images progressively," *IEEE Proceedings of Vision, Image and Signal Processing,* vol. 150, pp. 44-50, 2003.

[105] P. Lechat, H. Sanson and L. Labelle, "Image approximation by minimization of a geometric distance applied to a 3-D finite elements based model," in *Proceedings of IEEE International Conference on Image Processing*, 1997.

[106] G. Ramponi and S. Carrato, "An adaptive irregular sampling algorithm and its application to image coding," *Image and Vision Computing*, vol. 19, pp. 451-460, 2001.

[107] Y. Wang, O. Lee and A. Vetro, "Use of 2-D deformable mesh structures for video coding, part II - the analysis problem and a region-based coder employing an active mesh representation," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 6, pp. 647-659, 1996.

[108] J. Chen, S. Paris, J. Wang, W. Matusik, M. Cohen and F. Durand, "The video mesh: A data structure for image-based three-dimensional video editing," in *IEEE International Conference on Computational Photography (ICCP)*, Pittsburgh, PA, USA, 2011.

[109] Y. Guo, F. Liu, J. Shi, Z. Zhou and M. Gleicher, "Image retargeting using mesh parametrization," *IEEE Transactions on Multimedia*, vol. 11, pp. 856-867, 2009.

[110] A. Baghaie, R. D'souza and Z. Yu, "Sparse and low rank decomposition based batch image alignment for speckle reduction of retinal OCT images," in *IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, 2015.

[111] A. Y. Z. Baghaie and D. R., "State-of-the-art in retinal optical coherence tomography image analysis.," *Quantitative imaging in medicine and surgery*, vol. 5, no. 4, p. 603, 2015.

[112] A. Baghaie, Z. Yu and R. D'souza, "Fast mesh-based medical image registration.," in *International Symposium on Visual Computing*, 2014.

[113] B. Delaunay, "Sur la sphere vide," *Classe des Science Mathematics et Naturelle*, vol. 7, pp. 793-800, 1934.

[114] Y. Yang, N. Miles and G. Jovan, "A fast approach for accurate content-adaptive mesh generation," *IEEE Transactions on Image Processing*, vol. 12, pp. 866-881, 2003.

[115] M. Adams, "A flexible content-adaptive mesh-generation strategy for image representation," *IEEE Transactions on Image Processing*, vol. 20, pp. 2414-2427, 2011.

[116] M. Adams, "A highly-effective incremental/decremental Delaunay mesh-generation strategy for image representation," *IEEE Transactions on Signal Processing*, vol. 93, pp. 749-764, 2013.

[117] P. Li and M. Adams, "A tuned mesh-generation strategy for image representation based on data-dependent triangulation," *IEEE Transactions on Image Processing*, vol. 22, pp. 2004-2018, 2013.

[118] S. Rippa, "Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, pp. 1123-1141, 1992.

[119] M. Garland and P. Heckbert, "Fast polygonal approximation of terrains and height fields," Technical Report CMU-CS-95-181, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.

[120] X. Tu and M. Adams, "Improved mesh models of images through the explicit representation of discontinuities," *Canadian Journal of Electrical and Computer Engineering*, vol. 36, pp. 78-86, 2013.

[121] T. Blu, P. Thevenaz and M. Unser, "Linear interpolation revitalized," *IEEE Transactions on Image Processing*, vol. 13, no. 5, pp. 710-719, 2004.

[122] N. Asuni and A. Giachetti, "Accuracy improvements and artifacts removal in edge based image interpolation," in *VISAPP (1)*, 2008.

[123] A. Giachetti and N. Asuni, "Fast artifacts-free image interpolation," in *British Machine Vision Conference*, 2008.

[124] Y. Cha and S. Kim, "The error-amended sharp edge (EASE) scheme for image zooming," *IEEE Transactions on Image Processing*, vol. 16, no. 6, pp. 1496-1505, 2007.

[125] D. Zhou, W. Dong and X. Shen, "Image zooming using directional cubic convolution interpolation," *IET Image Processing*, vol. 6, no. 6, pp. 627-634, 2012.

[126] J. W. Han, J. H. Kim, S. H. Cheon, J. O. Kim and S. J. Ko, "A novel image interpolation method using the bilateral filter," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 1, pp. 175-181, 2010.

[127] J. W. Han, J. H. Kim, S. Sull and S. J. Ko, "New edge-adaptive image interpolation using anisotropic Gaussian filters," *Digit Signal Processing*, vol. 23, no. 1, pp. 110-117, 2013.

[128] X. Zhang and X. Wu, "Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation," *IEEE Transactions on Image Processing*, vol. 17, no. 6, pp. 887-896, 2008.

[129] W. Z. Shao and Z. H. Wei, "Edge-and-corner preserving regularization for image interpolation and reconstruction," *Image and Vision Computing*, vol. 26, no. 12, pp. 1591-1606, 2008.

[130] S. E. El-Khamy, M. M. Hadhoud, M. I. Dessouky, B. M. Salam and F. E. Abd El-Samie, "Efficient implementation of image interpolation as an inverse problem," *Digital Signal Processing*, vol. 15, no. 2, pp. 137-152, 2005.

[131] X. Liu, D. Zhao, R. Xiong, S. Ma, W. Gao and H. Sun, "Image interpolation via regularized local linear regression," *IEEE Transactions on Image Processing,* vol. 20, no. 12, pp. 3455-3469, 2011.

[132] J. Weickert, Anisotropic diffusion in image processing, Teubner Stuttgart, 1998.

[133] Z. Wang and A. C. Bovik, "Modern image quality assessment," *Synthesis lectures on image, video, and multimedia processing,* vol. 2, no. 1, pp. 1-156, 2006.

[134] P. Thevenaz, T. Blu and M. Unser, "Interpolation revisited," *IEEE Transactions on Medical Imaging,* vol. 19, no. 7, pp. 739-758, 2000.

[135] G. Casciola, L. Montefusco and S. Morigi, "Edge-driven image interpolation using adaptive anisotropic radial basis functions," *Journal of Mathematical Imaging and Vision,* vol. 36, pp. 125-139, 2010.

[136] J. Shewchuk, "Triangle: A two-dimensional quality mesh generator and Delaunay triangulator," 2005. [Online]. Available: http://www.cs.cmu.edu/quake/triangle.html.

[137] G. Casciola, D. Lazzaro, L. Montefusco and S. Morigi, "Shape preserving surface reconstruction using locally anisotropic RBF interpolants," *Computers & Mathematics with Applications,* vol. 51, pp. 1185-1198, 2006.

[138] G. Casciola, L. Montefusco and S. Morigi, "The regularizing properties of anisotropic radial basis functions," *Applied Mathematics and Computation*, vol. 190, pp. 1050-1062, 2007.

[139] L. Ke, X. Ming and Y. Zeyun, "Feature-Preserving Image Restoration from Adaptive Triangular Meshes," *Computer Vision - ACCV 2014 Workshops,* vol. 9009, pp. 31-46, 2014.

[140] I. ASTM, "ASTM F2792-10: standard terminology for additive manufacturing technologies," in *ASTM International*, West Conshohocken, PA, 2010.

[141] F. Melchels, M. Domingos, T. Klein, J. Malda, P. Bartolo and D. Hutmacher, "Additive manufacturing of tissues and organs.," *Progress in Polymer Science,* vol. 37, pp. 1079-1104, 2010.

[142] P. Bartolo, H. Almeida and T. Laoui, "Rapid prototyping and manufacturing for tissue engineering scaffolds," *International Journal of Computer Applications in Technology,* vol. 36, pp. 1-9, 2009.

[143] S. Peltola, F. Melchels, D. Grijpma and M. Kellomaki, "A review of rapid prototyping techniques for tissue engineering purposes," *Annals of Medicine* , vol. 40, pp. 268-280, 2008.

[144] S. Giannitelli, D. Accoto, M. Trombetta and A. Rainer, "Current trends in the design of scaffolds for computer-aided tissue engineering," *Acta Biomaterialia,* vol. 10, pp. 580-594, 2014.

[145] A. Khoda, I. Ozbolat and B. Koc, "Engineered tissue scaffolds with variational porous architecture.," *Journal of Biomechanical Engineering,* vol. 133, p. 011001, 2011.

[146] C. Schroeder, W. Regli, A. Shokoufandeh and W. Sun, "Computer-aided design of porous artifacts.," *Computer-Aided Design,* vol. 37, pp. 339-353, 2005.

[147] S. Sogutlu and B. Koc, "Stochatic modeling of tissue engineering scaffolds with varying porosity levels.," *Compute-Aided Design & Applications,* vol. 4, pp. 661-670, 2007.

[148] D. Schaefer and K. Keefer, "Structure of random porous materials: silica aerogel.," *Physical Review Letters,* vol. 56, pp. 2199-2202, 1986.

[149] F. Melchels, P. Wiggenhauser, D. Warne, M. Barry, F. Ong, W. Chong and e. al., "CAD/CAM-assisted breast reconstruction.," *Biofabrication,* vol. 3, p. 034114, 2011.

[150] W. Chiu, Y. Yeung and K. Yu, "Toolpath generation for layer manufacturing of fractal objects.," *Rapid Prototyping Journal,* vol. 12, pp. 214-221, 2006.

[151] W. Sun, B. Starly, J. Nam and A. Darling, "Bio-CAD modeling and its applications in computer-aided tissue engineeing.," *Computer-Aided Design,* vol. 37, pp. 1097-1114, 2005.

[152] B. Bucklen, W. Wettergreen, E. Yuksel and M. Liebschner, "Bone-derived CAD library for assembly of scaffolds in computer-aided tissue engineering.," *Virtual and Physical Prototyping,* vol. 3, pp. 13-23, 2008.

[153] S. Hollister, R. Maddonx and J. Taboas, "Optimal design and fabrication of scaffolds to mimic tissue properties and satisfy biological constraints.," *Biomaterials,* vol. 23, pp. 4095-4103, 2002.

[154] S. Hollister, "Porous scaffold design for tissue engineering.," *Nature Materials,* vol. 4, pp. 518-524, 2005.

[155] S. Rajagopalan and R. Robb, "Schwarz meets Schwann: design fabrication of biomorphic and durataxic tissue engineering scaffolds.," *Medical Image Analysis,* vol. 10, pp. 693-712, 2006.

[156] T. Seck, F. Melchels, J. Feijen and D. Grijpma, "Designed biodegradable hydrogel structures prepared by stereolithography using poly(ethylene glycol)/poly(D,L-lactide)-based resins.," *Journal of Controlled Release,* vol. 148, pp. 34-41, 2010.

[157] F. Melchels, J. Feijen and D. Grijpma, "A poly(d, l-lactide) resin for the preparation of tissue engineering scaffolds by stereolithography.," *Biomaterials,* vol. 30, pp. 3801-3809, 2009.

[158] L. Elomaa, S. Teixeira, R. Hakala, H. Korhonen, D. Grijpma and J. Seppala, "Preparation of poly(e-caprolactone)-based tissue engineering scaffolds by stereolithography.," *Acta Biomaterialia,* vol. 7, pp. 3850-3856, 2011.

[159] D.-J. Yoo, "Computer-aided porous scaffold design for tissue engineering using triply periodic minimal surfaces.," *International Journal of Precision Engineering and Manufacturing,* vol. 12, pp. 61-71, 2011.

[160] D.-J. Yoo, "Porous scaffold design using the distance field and triply periodic minimal surface models.," *Biomaterials,* vol. 32, pp. 7741-7754, 2011.

[161] I. Zein, D. Hutmacher, K. Tan and S. Teoh, "Fused deposition modeling of novel scaffold architectures for tissue engineering applications.," *Biomaterials*, vol. 23, pp. 1169-1185, 2002.

[162] S. Kalita, "Development of controlled porosity polymer-ceramic composite scaffolds via fused deposition modeling.," *Materials Science and Engineering*, vol. 23, pp. 611-620, 2003.

[163] X. Kou and S. Tan, "Microstructural modeling of functionally graded materials using stochastic Voronoi diagram and B-spline representations.," *International Journal of Computer Integrated Manufacturing*, vol. 25, pp. 177-188, 2012.

[164] D. B. Kirk and W. Hwu, Programming massively parallel processors: a hands-on approach, Burlington, MA, USA: Elsevier Inc., 2010.

APPENDEX: PUBLICATIONS

- K. Liu, G. Yao, Z. Yu, "Parallel Acceleration for Modeling of Calcium Dynamics in Cardiac Myocytes", *Bio-Medical Materials and Engineering*, vol. 24, no. 1, pp. 1417-1424, 2014
- K. Liu, M. Xu, Z. Yu, "Feature-Preserving Image Restoration from Adaptive Triangular Meshes", *Computer Vision - ACCV 2014 Workshops*, vol. 9009, pp. 31-46, 2014.
- [In submission] K. Liu, Y. Guo, Z. Yu, "Porous Structure Design in Tissue Engineering Using Anisotropic Radial Basis Function", *arXiv:1612.01944 [cs.GR]*, 2016

CURRICULUM VITAE

Ke Liu

Computer Science Department of University of Wisconsin Milwaukee

Advisor: Professor, Dr. Zeyun Yu

Address:

3200 N Cramer Street,

Milwaukee, WI 53211

USA

Email:

keliu@uwm.edu

Research Interests

Image/mesh processing, numerical analysis, parallel computing, scientific computing, performance optimization, computer graphics, data visualization

---

Education

- Bachelor in Computer Science, University of Electronic Science and Technology of China, 2006

  Thesis title: Instant Messaging System in Client/Server Architecture
- Coursera: Advanced Javascript Programming, 2014
- Coursera: Web Development, 2014

Professional and Work Experience

- University of Wisconsin Milwaukee Since Sep. 2015

  Graduate Teaching Assistant

  Graduate Teaching Assistant for the following courses:

  - Data Structure and Algorithms (CS 351)
  - Server-side Programming (CS 481)

- University of Wisconsin Milwaukee Summer, 2015

  School of Public Health

  Graduate Research Assistant

  1) Analyze exome DNA sample data

  2) Build analyzing software pipeline

- University of Wisconsin Milwaukee 2014 – 2015

  Academic Affairs

  Graduate Project Assistant

  1) Create Hadoop test setup and explore its applications in campus

  2) Instruct users to use cluster

- University of Wisconsin Milwaukee 2012 – 2014

  University Information Technology Services

  Graduate Project Assistant

  1) Discuss with users about performance optimization

2) Assist students in parallel computing and research computing workshops

- University of Wisconsin Milwaukee 2010 – 2012

  Biomedical Modeling and Visualization Lab

  Graduate Research Assistant

  1) Model calcium dynamics in cardiac myocytes

  2) Solve partial differential equations (PDEs) by meshless method

  3) Increase performance up to 20X by using CUDA

  4) Simulate calcium dynamics in ventricular myocytes on subcellular level

  5) Simulate particle dynamics based on Monte-Carlo method

  6) Increase performance up to 50X by using MPI

- FlysInfo Computing Services (Chengdu) 2008 – 2009

  Project Manager / Software Engineer

  1) Implement communication and document workflow

  2) Implement user authentication module

  3) Manage project development and deployment

  4) Communicate with customer to acquire and verify requirements

- Augmentum (Shanghai) 2006 – 2008

  Software Engineer

  1) Implement messaging service for Outlook 2003

2) Implement user authentication, gateway communication, database accessing and report generating modules

3) Design and implement message queue and workflow by using BusinessWorks from Tibco

4) Write development documents

Publications

- Ke Liu, Guangming Yao, Zeyun Yu, Parallel Acceleration for Modeling of Calcium Dynamics in Cardiac Myocytes. Bio-Medical Materials and Engineering, vol. 24, no. 1, pp. 1417-1424, 2014
- Ke Liu, Ming Xu, Zeyun Yu, Feature-preserving Image Restoration from Adaptive Triangular Meshes, Computer Vision – ACCV 2014 Workshop, vol. 9009, pp. 31-46, 2014
- [In submission] Ke Liu, Ye Guo, Zeyun Yu, Porous Structure Design in Tissue Engineering Using Anisotropic Radial Basis Function, arXiv:1612.01944 [cs.GR], 2016

Technical Skills

Languages: C/C++, CUDA C, Python, Java, Javascript, Matlab, bash, tcsh, C#

Tools:

- Parallel Computing: CUDA Tookit, OpenMP, MPI, Hadoop
- Graphic: OpenGL
- Web Development: Django, Amazon AWS
- Cluster: LSF, PBS, SLURM, HTCondor

- Genome Sequence Analyzing: bowtie2, bwa, picard, GATK, SeqMule, ANNORVAR

Methodologies:

- Numerical methods: radial basis function (RBF) interpolation, meshless