Characterizing Self-assembled Nanostructures via Shapelet Functions

by

Matthew Peres Tino

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Chemical Engineering

Waterloo, Ontario, Canada, 2024

© Matthew Peres Tino 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Matthew Peres Tino was the sole author for Chapters 1, 2, 3, 6, and 7 which were written under the supervision and guidance of Dr. Nasser Mohieddin Abukhdeir and were not written for publication.

Exceptions to sole authorship of material in this thesis, with reference to the CRediT (Contributor Role Taxonomy) system, are outlined below.

Chapter 4: Sections 4.1–4.3 form the basis of recently published work: M.P. Tino, R. Suderman, and N.M. Abukhdeir, "Shapelet-based orientation and defect identification method for nanostructured surface imaging," Nanotechnology, 35(16), 2024. Specific author contributions are:

- R. Suderman – Conceptualization
- N.M. Abukhdeir – Conceptualization, Resources, Writing - Original Draft, Writing - Review & Editing, Supervision

Chapter 5: This chapter forms the basis of recently published work: M.P. Tino, A.Y. Abdulaziz, R. Suderman, T. Akdeniz, and N.M. Abukhdeir, "Shapelets: A Python package implementing shapelet functions and their applications," Journal of Open Source Software, 9(95), 2024. Specific author contributions are:

- A.Y. Abdulaziz – Software, Writing - Original Draft
- R. Suderman – Software
- T. Akdeniz – Software
- N.M. Abukhdeir – Conceptualization, Methodology, Software, Resources, Writing - Review & Editing, Supervision

Abstract

Pattern formation is a natural phenomena that occurs at various length-scales. Lattice patterns are a particular type composed of spatially-repeating features with stripe, square, or hexagonal symmetries. They are of particular interest to nanotechnology researchers due to their frequent appearance in self-assembly and lithography processes. Self-assembled nanostructures provide many technological applications but are difficult to characterize due to deformations in local structure (defects, disorder). While image-based characterization techniques for nanostructures are well-known (i.e., scanning electron microscopy), appropriate computational techniques to characterize their structure are seldom developed and are typically without readily available open-source implementations.

Characterization of self-assembled nanostructures is important to develop structure-property relationships with potential to advance defect engineering research. Defect engineering corresponds to the regulation of specific defects within nanostructure to manipulate material properties (physical, chemical, magnetic) and improve material functionality.

Existing techniques to characterize self-assembled nanostructures, including Voronoi diagrams/entropy, bond-orientational order theory, and Fourier space filtering are well-known but contain inherent limitations. A more recent and promising approach uses a set of localized basis functions called *shapelets*, originally designed for the compression and reconstruction of images of galaxies. This approach uses *polar* shapelets, providing unique rotational/radial symmetry properties beneficial for analysis on non-Euclidean geometries. This response distance method is a supervised learning technique that quantifies local deformations in structure (defects, disorder) apart from regions displaying uniform pattern order.

This work presents extensions to the existing response distance method, including a decrease in computational runtime, along with the inclusion of higher-order shapelet functions to improve order quantification in areas with topological defects. New shapelet-based methods are also presented, such as quantification of local pattern orientation and a technique to directly identify topological defects and defect structures. These methods are validated against both simulated and experimental surface images of self-assembled nanostructures containing stripe, square, and hexagonal patterns and demonstrates their effectiveness in the presence of measurement noise. Furthermore, they are made available to the community as part of an open-source Python library, along with reference implementation of other shapelet functions and applications to promote collaboration and transparency in shapelet research. The mathematical framework for a higher-order shapelet class with radial symmetry is also provided, laying the foundation for future pattern analysis.

Acknowledgements

First and foremost, I want to thank my supervisor, Professor Nasser Abukhdeir, for his unwavering and consistent support throughout my Masters study and in numerous other opportunites presented to me over the past two and one half years. Any venture contains inherent risks and difficulties, but you (Nasser) continuously believed in my abilities. For this, along with the mentorship you have given me about life, family, and education, I am extremely grateful.

I would also like to thank past (Sajeda) and current (Alex, Nicholas, Arshia, Ittisak, Rithu) members of the Computational Multiphysics Research Group for the many conversations, both technical and jovial, that allowed me to enjoy my Masters journey. Special thanks to undergraduate Nanotechnology Engineering students Abbas and Cameron for their efforts toward shapelet-based research and the open-source shapelets repository.

Last, I want to acknowledge and thank the Department of Chemical Engineering for awarding me the Engineering Excellence Master's Fellowship (EEMF), which not only provided me with financial support, but granted me the ability to select my Masters supervisor and project.

Dedication

To my amazing family - mom, dad, Stephen (Ana) and my beautiful girlfriend Laura. None of the work in this thesis would be complete if not for your unconditional love and support throughout this experience.

To the late Mr. Frank Heijmans, who pushed me to consider a career in engineering. I am forever grateful for your mentorship in my youth; rest in peace.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract iv

Acknowledgements v

Dedication vi

List of Figures x

List of Tables xiii

1 Introduction 1

1.1 Research and Motivation 1

1.2 Objectives 4

1.3 Thesis Structure 5

2 Background 7

2.1 Self-assembled Nanostructures 7

2.1.1 Atomistic Growth Processes 7

2.1.2 Solution-based Methods 9

2.2 Topological Defects and Defect Structures 13
2.2.1 Disclinations and Dislocations 15
2.2.2 Grain Boundaries 18
2.3 Shapelet Functions 19
2.3.1 Polar Shapelets 20
2.3.2 Pattern Analysis 21

3 Literature **30**
3.1 Bond-Orientational Order Theory 30
3.2 Voronoi Diagrams and Entropy 31
3.3 Fourier Space Filtering 33

4 Shapelet-based orientation and defect identification method for nanostructured surface imaging **38**
4.1 Higher-order Shapelets for Pattern Analysis 38
4.2 Orientation 42
4.3 Defect Identification Method 44
4.4 Computational Improvements 46
4.4.1 C++ Implementation 46
4.4.2 Comparing Python and C++ Implementation Runtimes 48

5 Shapelets: A Python package implementing shapelet functions and their applications **53**
5.1 Summary 53
5.2 Statement of Need 54
5.3 Features 55
5.4 Usage and Examples 56

6 Higher-Order (n = 1) Shapelets **58**
6.1 Motivation . . . 58
6.2 Demonstrating Orthogonality . . . 59
6.2.1 The Comparison Test . . . 60
6.2.2 Solving Normalization Constant . . . 63
6.3 Optimal Shapelet Length-Scales . . . 64
6.3.1 Brute-Force Numerical Approach . . . 64
6.3.2 Difficulties with Increasing **m** . . . 67

7 Conclusions **69**
7.1 Conclusions . . . 69
7.2 Recommendations . . . 70

References **72**

Appendices **78**

A Shapelet-based orientation and defect identification method for nanostructured surface imaging **78**

B Mathematical Details of Higher-order (n = 1) Shapelets **87**
B.1 Definition . . . 87
B.1.1 Laguerre Polynomials . . . 87
B.2 Orthonormalization . . . 88
B.2.1 Solving the First Integral . . . 89
B.2.2 Solving the Second Integral . . . 90
B.2.3 Solving the Third Integral . . . 91

List of Figures

1.1 Nanostructured surface images from mesoscopic simulations of heteroepitaxial self-assembly . . . 2

1.2 Surface images of self-assembled nanostructures from various experimental image-based characterization techniques . . . 6

2.1 Aluminum self-assembled nanostructures resulting from both low- and high-vacuum physical vapour deposition . . . 8

2.2 Lead and strontium titanate self-assembled superlattice grown on strontium titanate substrates via molecular beam epitaxy . . . 10

2.3 Mesoporous titanium dioxide and ruthenium dioxide composite with semi-hexagonal order from evaporated-induced self-assembly . . . 11

2.4 Vertical arrays of gold nanorods with hexagonal order from evaporation-induced self-assembly . . . 12

2.5 Hexagonal arrays of thiolate-encapsulated gold nanoparticles after heat treatment using the Brust-Schiffrin technique . . . 13

2.6 Square/hexagonal ordered gold nanocrystal/silica supperlatice thin films via sol-gel self-assembly . . . 14

2.7 Template-based gold nanoparticle synthesis into rod- (near-stripe), triangular- (hexagonal), and square-ordered arrays via noncovalent binding to a pseudorotaxane . . . 15

2.8 Wedge disclinations of varying topological charge in a hypothetical smectic phase liquid crystal . . . 16

2.9 Volterra process to create a $-\pi$ wedge disclination defect . . . 17

2.10 Wedge disclinations found in a hypothetical hexatic phase liquid crystal . . . 18

2.11 Demonstrating edge dislocation defects in a hypothetical smectic phase liquid crystal . . . 19
2.12 Demonstrating a dislocation defect composed of oppositely charged disclinations in a hexagonal array . . . 20
2.13 Demonstrating a low-angle edge grain boundary defect structure in a hypothetical smectic phase liquid crystal . . . 21
2.14 Polar shapelets and their symmetry parameters n and m . . . 22
2.15 A 6-fold ($S_{nm} \rightarrow S_{06}$) cartesian *steerable* shapelet filter rotated at its optimal orientation and superimposed on a hexagonally-ordered domain. . . . 27
2.16 The result of applying the computationally-enhanced response distance method to a simulated hexagonal self-assembled nanostructure . . . 29

3.1 An example of bond-orientational order analysis applied to a hexagonal domain 35
3.2 Example of Voronoi diagram construction on a hypothetical plane of distributed points . . . 36
3.3 Disclination and dislocation defects identified via construction of a Voronoi diagram for self-assembled diblock copolymer micelles . . . 36
3.4 Application of the modified Fourier space filtering method applied to hexagonal self-assembled block copolymers . . . 37

4.1 The cumulative shapelet response for order $m \leq 30$ shapelets . . . 40
4.2 The normalized difference in the response distance when using lower-order ($m \leq 6$) and incorporating higher-order ($m \geq 7$) shapelets superimposed onto the original image . . . 41
4.3 Masking and dilation operations applied to simulated patterns . . . 49
4.4 The result of applying the blending operation via median filter [55] and superimposing this result onto the original image . . . 50
4.5 Decomposition of shapelet response vectors into k centroid vectors via k-means clustering . . . 51
4.6 The defect response distance superimposed onto the original pattern image after using a median filter [55] with kernel size $\frac{\lambda}{2}$ for smoothing . . . 52

5.1 Sample directory hierarchy . . . 56

6.1 Midline (or center-line) profile for an $n = 1$ shapelet function 65
6.2 Higher-order ($n = 1$) shapelets with optimal length-scale, β_{opt}, superimposed on an ideal hexagonal domain 67
6.3 Collapse of $n = 1$ shapelet kernel as m increases and displaying this behaviour for $m = 10$ 68

A.1 The response distance scalar field using lower-order ($m \leq 6$) shapelets . . . 79
A.2 The response distance scalar field including higher-order ($m \geq 7$) shapelets 80
A.3 The orientation mask 81
A.4 The result of applying the morphological greyscale dilation operation . . . 82
A.5 The result of applying the blending operation via median filter 83
A.6 Visualizing the response cluster associated with each pixel (response vector) 84
A.7 Radar chart representations of the centroid response vector weights for order $m \leq 7$ shapelets 85
A.8 The defect response distance scalar field (after applying blending operation) 86

List of Tables

4.1 Comparing mean computational runtimes to compute the response distance between Python and C++ implementations 48

5.1 A summary of the various shapelet function definitions provided by the shapelets Python package. 55

5.2 A summary of the various shapelet applications provided by the shapelets Python package. 56

Chapter 1

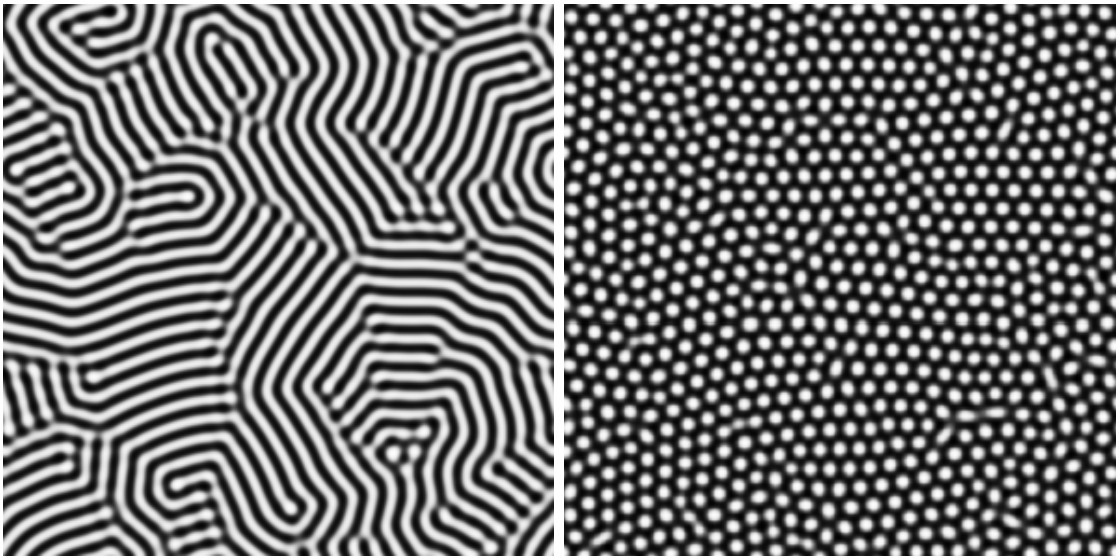
Introduction

1.1 Research and Motivation

Pattern formation is an interdisciplinary field encompassing various aspects of science, engineering, and mathematics [9]. It refers to the natural organization of repeating units (shapes) into distinct structures (patterns). Alan Turing was one of the first scientists to display an interest in pattern formation when he observed the formation of unique pattern structures during morphogenesis (i.e. cell formation) [53]. However, pattern formation can naturally occur in many other contexts and length-scales, such as *stripe* patterns; found on zebras, as ripples in desert sand, and in numerical simulations of convective flow/momentum (i.e. the Swift-Hohenberg model) [26].

The study of pattern formation remains a complex topic as its spatial and temporal evolution is governed by nonlinear dynamics [9], but this is not studied here. Rather, this work focuses on the characterization of steady-state pattern *structures*, namely from microscopy imaging, containing local/global orientational and translational order within pattern domains. In particular, the primary focus of this work is to characterize pattern structures that repeat in space and contain stripe, square, or hexagonal symmetries, so-called "lattice patterns" [26, Figs. 5.1 and 5.7].

Lattice patterns are prevalent in the world of nanotechnology and nanoscience. They arise in a variety of synthesis techniques, including self-assembly (bottom-up) and lithography (top-down) [1]. Self-assembled nanostructures are formed through chemical building blocks and spontaneously organize into energetically favourable spatially-varying pattern structures [5]. Examples of lattice pattern nanostructures from mesoscopic simulations of self-assembly are shown in Figure 1.1.



(a) (b)

Figure 1.1: Nanostructured surface images from mesoscopic simulations of heteroepitaxial self-assembly [3]: (a) stripe (or lamellar) and (b) hexagonal patterns.

Pattern structures resulting from self-assembly typically contain spatial distortions, called *defects*, that disrupt local pattern homogeneity. Manipulation of these defects within self-assembled domains to enhance material properties, so-called "defect engineering", is a relatively new field with recent research showing great promise and success already observed for inorganic and organic hard materials [28]. A practical example of defect engineering related to self-assembly is block copolymer defects for integrated circuits in microelectronics [35] (see ref. [28] for more examples). While several robust image-based characterization methods exist for nanostructures, such as scanning electron microscopy, atomic force microscopy, and transmission electron microscopy [39], subsequent computational techniques to characterize their structure are less developed [52]. Figure 1.2 shows surface images of self-assembled nanostructures from each of the aforementioned microscopy techniques.

Computational methods to quantitatively characterize surface images of nanostructure are necessary to determine material properties from structure, so-called "structure-property relationships" [2]. While this remains an important task, it can be especially challenging for those working with materials on the nanoscale [2]. This is evident for self-assembled systems, governed by thermodynamics, which lend itself to the formation of topological defects and defect structures corresponding to local metastable minimums in free energy [28]. These defects add an inherent complexity to nanomaterial structure, necessitating the

development of specific computational techniques to characterize nanostructure in order to derive structure-property relationships.

One particular solution to this problem is found in bond-orientational order theory [12]. Bond-orientational order refers to the degree of n-fold symmetry of the objects in a system where, on average, each object has n nearest neighbours [12, 48]. The degree of *order* is described by the complex bond-orientational order parameter, which can be computed for each feature in the system and averaged over the entire domain to quantify global order [48] (see Section 3.1 for more details). However, this method has inherent limitations such as pattern specificity, inapplicable to patterns with non-convex (continuous) features such as stripe patterns. As stripe (or lamellar) patterns are common in self-assembled systems (i.e. block copolymers), other quantitative methods must be used.

A more recent and promising approach uses a set of localized orthogonal basis functions called *shapelets* [44]. Shapelets were originally developed for the compression and reconstruction of images of galaxies, where large-memory images can be compressed by factors of 40-90 [44]. *Polar* shapelets [36] were developed as an analogue to the original shapelet formulation and contain unique radial and rotational symmetry properties that have proven useful for image analysis involving non-Euclidean geometries (i.e. circles, ellipses).

Suderman *et al.* [49] reformulated polar shapelets to characterize nanostructured surfaces of self-assembled domains. They developed a supervised learning technique called the *response distance method* [49] that uses convolutional responses between shapelet filters and a pattern to quantify well-ordered regions apart from those with local deformities (defects, disorder). This method was later improved by Akdeniz *et al.* [4], including a significant reduction in computational runtime and an analytical formulation to compute constant shapelet length-scales in agreement with underyling pattern feature scale.

While this response distance method [49] has demonstrated success in characterizing nanostructure for arbitrary patterns, the full potential of shapelet-based techniques in this area has not been explored. This includes identification (or classification) of topological defects and quantification of local pattern orientation in the presence of grain boundary defect structures.

The methods developed in this work intend to narrow the gap between well-known image-based characterization techniques for nanostructures and adequate computational methods to characterize their structure. These methods will be useful for materials researchers and engineers to develop structure-property relationships, providing insight into the connections between material structure, functionality, and application.

1.2 Objectives

The overall objective of this work is to extend existing and develop new shapelet-based methods to characterize surface images of self-assembled nanostructures to develop structure-property relationships for various materials. These methods are validated through application to a representative set of simulated (Figure 1.1) and experimental (Figure 1.2) self-assembled surface images with a diverse range of experimental noise and uncertainty.

Specific objectives of this work include:

1. Introduce higher-order shapelet functions into the existing response distance method to improve the accuracy in identification of uniform (i.e. well-ordered) and non-uniform (i.e. disorder, defect) regions.

2. Use shapelet-based orientational information, provided by steerable filter theory, to quantify *local* pattern orientation in the presence of grain boundary defect structures for arbitrary pattern types.

3. Develop a new method using the shapelet convolutional response, inspired by the response distance method, to directly identify topological defects and defect structures in self-assembled domains for arbitrary pattern types.

4. Develop an open-source Python package that provides reference implementation of various shapelet functions and some of their existing applications in science and engineering.

1.3 Thesis Structure

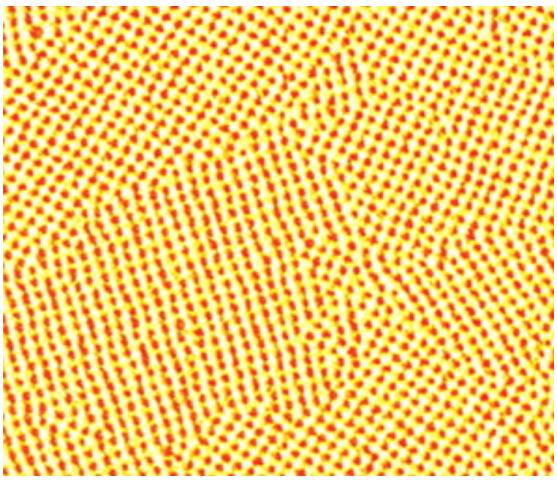
The structure of this thesis is as follows:

- Chapter 2 contains the background for this work, including numerous examples of self-assembled nanostructures, topological defects and defect structures, and a comprehensive overview of polar shapelet functions for pattern analysis.
- Chapter 3 provides a literature survey of existing computational techniques to characterize surface images of nanostructures (i.e. from simulation or microscopy imaging).
- Chapter 4 describes three core methods that form the basis of recently published work[1], introducing extensions to the existing response distance method and new shapelet-based methods to characterize self-assembled nanostructure domains.
- Chapter 5 contains an overview of the open-source shapelet-based Python package[2], including the various shapelet functions and applications implemented, along with a custom text-based configuration file user interface.
- Chapter 6 contains the mathematical framework for a higher-order shapelet class with radial symmetry, providing the foundation for future pattern analysis.
- Chapter 7 provides conclusions and recommendations for future work.

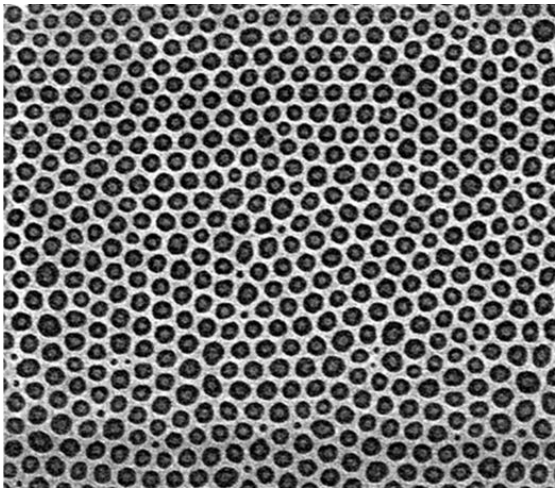
[1] See ref. [52].
[2] See ref. [51].



(a)



(b)



(c)

Figure 1.2: Surface images of self-assembled nanostructures from various experimental image-based characterization techniques: (a) stripe (or lamellar) pattern from transmission electron microscopy [41], (b) square pattern from atomic force microscopy [50], and (c) hexagonal pattern from scanning electron microscopy [42].

Chapter 2

Background

2.1 Self-assembled Nanostructures

Self-assembled nanostructures are formed from bottom-up processes [1] and are mediated by the interaction of so-called "precursor" molecules [5]. These precursor molecules, existing in vapour- or solution-phase, interact with themselves or other molecules as building blocks with underlying substrates/templates to produce unique and complex nanostructures [5]. Many examples of these self-assembled nanostructures are presented here.

2.1.1 Atomistic Growth Processes

Vapour-phase deposition processes can produce self-assembled nanostructures that are well-ordered [5], but are not necessarily highly-ordered. These processes typically involve deposition of vapour-phase atoms or molecules onto a substrate [5]. Common and well-known methods for vapour-phase deposition are atomistic growth processes, such as physical vapour deposition, chemical vapour deposition, and molecular beam epitaxy.

Physical and Chemical Vapour Deposition

Physical vapour deposition methods use a physical process to reduce a bulk material into gaseous (vapour) particles, after which they are deposited onto a substrate. They are also classified as vacuum deposition methods [43] as they require vacuum conditions for successful deposition. Such an example is thermal evaporation, where an extreme heat source is

used to vapourize some bulk material to create atomic-size particles [5]. These particles are then deposited onto a substrate where thin films are formed via condensation [5]. Another example is sputtering, where bulk material is bombarded with high-velocity ions followed by vapourized source material deposition through a condensation process [5]. However, due to the nature of physical vapour deposition, controlling resulting nanostructure is difficult, limiting the potential to produce highly-ordered nanostructures.

For example, Stagon *et al.* [47] demonstrated that oxygen can act as a surfactant to enable the growth of aluminum nanorods via physical vapour deposition. They reported that the diameter of the aluminum nanorods can be controlled via modulation of oxygen concentration, vacuum intensity (environment pressure), and substrate temperature [47]. Figure 2.1 shows microscopy images of these aluminum nanorods synthesized in both low- and high-vacuum conditions.

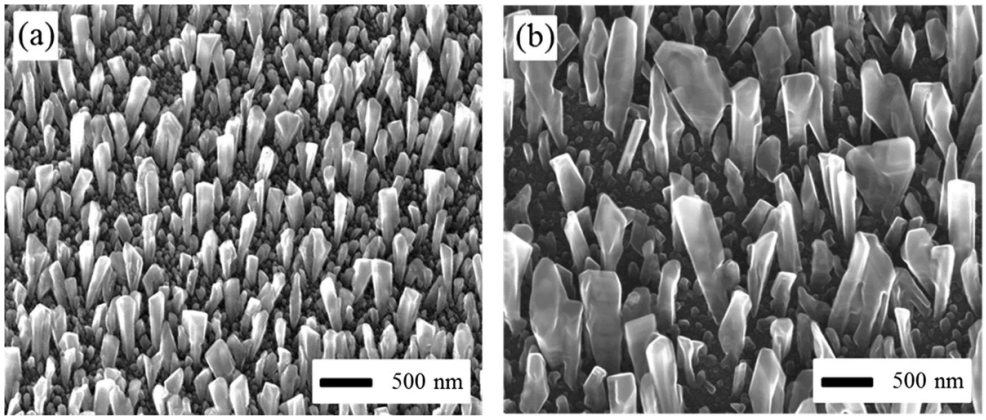


Figure 2.1: Aluminum self-assembled nanostructures resulting from both low- and high-vacuum physical vapour deposition [47, Fig. 2]: (a) growth using low vacuum and (b) growth using high vacuum environments.

An alternative to physical vapour deposition is chemical vapour deposition, which provides some practical advantages such as finer control in deposition rates due to dependence on chemical reactions as opposed to erratic, high-velocity particle behaviour observed under vacuum conditions [5]. Typical chemical vapour deposition processes involve a reactor that hosts reactant gases; these gases either deposit directly onto a substrate or participate in gas-phase reactions to form other gas molecules that will deposit onto the substrate surface [5]. Then, reactions at the gas-solid (reactant-substrate) interface produces thin films on the substrate surface [5].

Similar to physical vapour deposition, chemical vapour deposition does not typically

produce highly-ordered nanostructures, which can be attributed to the difficulty in specific control of the deposition process.

Molecular Beam Epitaxy

Molecular beam epitaxy refers to the deposition or arrangement of gaseous particles onto a crystalline surface, whereby the resulting deposited structure conforms to that of the underlying substrate [5]. Similar to physical vapour deposition methods, the source material is evaporated, and the evaporated atoms are adsorbed after bombardment onto the substrate surface and conform to the crystal lattice, while excess atoms (i.e. those not participating in adsorption) are thermally desorbed [5]. Unlike physical and chemical vapour deposition, the resulting structures from molecular beam epitaxy can be highly-ordered, such as square or hexagonal arrays, due to fine deposition control.

For example, Jiang *et al.* [29] reported the synthesis of highly-ordered square superlattice lead/strontium titanate structures grown on strontium titanate substrates. They used ozone as an oxidant to deposit strontium titanate via sequential deposition of strontium oxide and titanium dioxide layers, whereas the lead titanate structures were deposited via an adsorption-controlled growth process [29]. Transmission electron microscopy images of the superlattice, shown in Figure 2.2, demonstrate highly-ordered (square) crystalline nanostructures with no observable crystal defects. Additionally, ref. [3] demonstrates simulation of heteroepitaxial growth to produce highly-ordered nanostructures (see Figure 1.1).

2.1.2 Solution-based Methods

An alternative to vapour-phase self-assembly processes are solution-based methods, whereby the synthesis of highly-ordered nanostructures are mediated through chemical interactions (electrostatic, van der Waals, hydrophobic) between precursor molecules [5]. Finer control in resultant nanostructure shape and geometry is provided by this class of techniques. Examples of solution-based methods include evaporation-induced self-assembly, colloidal growth and sol-gel methods, and directed self-assembly.

Evaporation-induced Self-assembly

Evaporation-induced self-assembly (EISA) is a method that produces precise nanostructures from solution [5]. It is a mixture of sol-gel and surfactant chemistry, whereby surfac-

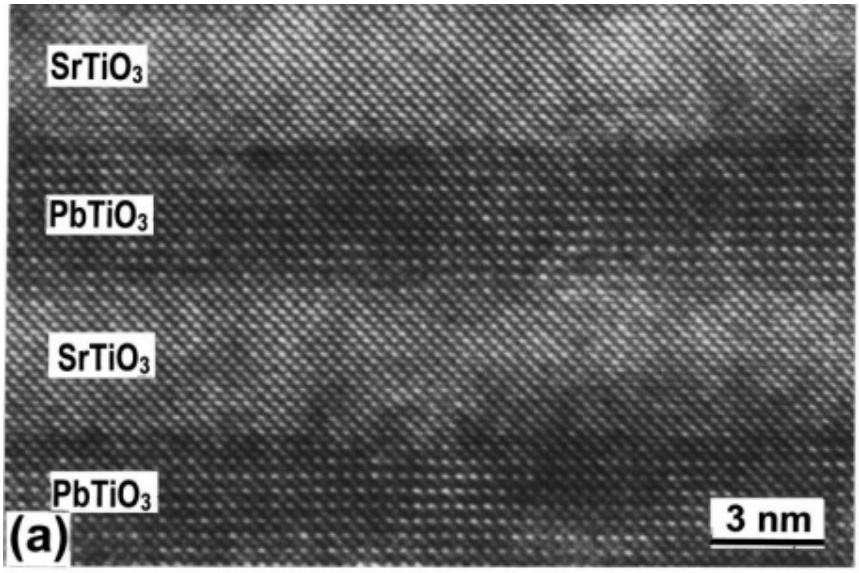


Figure 2.2: Lead and strontium titanate self-assembled superlattice grown on strontium titanate substrates via molecular beam epitaxy [29, Fig. 4a].

tants form aggregates (i.e. micelles) in aqueous solution, and hydrophobic precursors can be embedded within these aggregates to form various nanostructures.

The EISA method has been used to produce ordered thin films, such as mesoporous titanium dioxide and ruthenium dioxide composites with semi-hexagonal order [6], shown in Figure 2.3.

The EISA method has also been used to produce arrays of nanostructure, such as vertical arrays of gold nanorods with hexagonal order [21], shown in Figure 2.4. In this work, Dong *et al.* [21] reported that the morphology/nanostructure of the gold nanorod array can be well-managed by varying the "immersion time" in the target molecules' solution.

Colloidal Growth and Sol-Gel Methods

Colloidal growth and sol-gel self-assembly are another class of techniques that allow for great control of particle size distribution [5] and can form systems with hexagonal order.

For example, the Brust-Schiffrin technique is a well-known colloidal-based self-assembly synthesis method for the fabrication of thiolate-encapsulated gold nanoparticles, typically via a two-phase synthesis process [5]. It first involves a phase transfer (typically via a

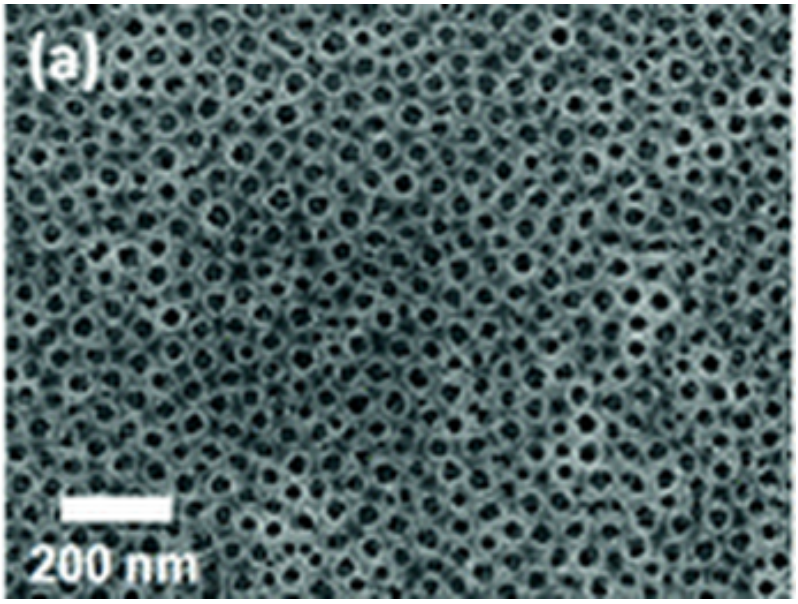


Figure 2.3: Mesoporous titanium dioxide and ruthenium dioxide composite with semi-hexagonal order from evaporated-induced self-assembly (EISA) [6, Fig. 2a].

bromide catalyst) of gold(III) from aqueous into organic phase. After separation of aqueous and organic phases, the gold is reduced to gold(I) via the addition of a specific thiol molecule. Finally, a reducing agent is added which forms the highly stable gold nanoparticles. These gold nanoparticles also have a diverse range of applications (see ref. [5]).

Maye *et al.* [37] reported the addition of a heat treatment strategy to obtain monodisperse thiolate-encapsulated gold nanoparticles using the Brust-Schiffrin technique. They found that high-temperature heating after the two-step synthesis process reduced the particle size distribution to produce ordered hexagonal arrays, as opposed to non-uniform shapes (polyhedrons, octahedrons) before heat treatment [37]. Figure 2.5 shows a transmission electron microscopy image of the post-heat treatment gold nanoparticle array.

The sol-gel method is another technique that can produce solid-state material arrays with fine control on particle distribution and morphology [5]. The central idea of the sol-gel process begins with the creation of a colloidal suspension (i.e. a "sol") followed by formation of a network structure (i.e. a "gel") [5].

In one study, Fan. *et al.* [22] reported the synthesis of three-dimensional ordered gold nanocrystal/silica superlattice thin films. They were able to obtain well-ordered square and hexagonal symmetries using their sol-gel synthesis technique. Figure 2.6 shows transmission

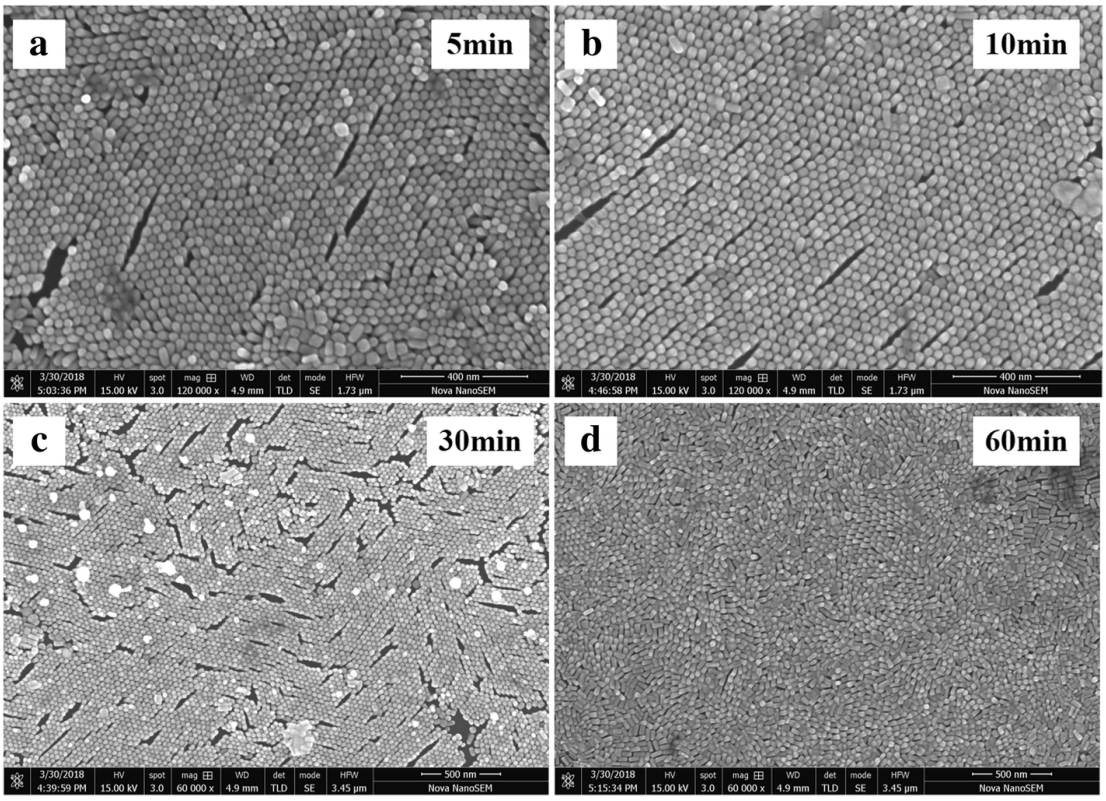


Figure 2.4: Vertical arrays of gold nanorods with hexagonal order from evaporation-induced self-assembly (EISA) [21, Fig. 4].

electron microscopy images of these superlattice structures.

Directed Self-assembly

Directed self-assembly presents another group of synthesis techniques where nanoparticles or precursor molecules can spontaneously organize into highly-ordered structures that persist over long distances [5]. This organization can either occur via internal (inter-particle) or external (i.e. electric field) forces, or through the use of "templates" [5].

Template-based self-assembly is a type of directed self-assembly method that takes advantage of n-dimensional ($n \in \{1, 2, 3\}$) substrate surfaces as a "template" [5]. Similar to molecular beam epitaxy, a template acts as a physical stage where particles can arrange themselves in agreement (or disagreement) with the underlying template structure [5].

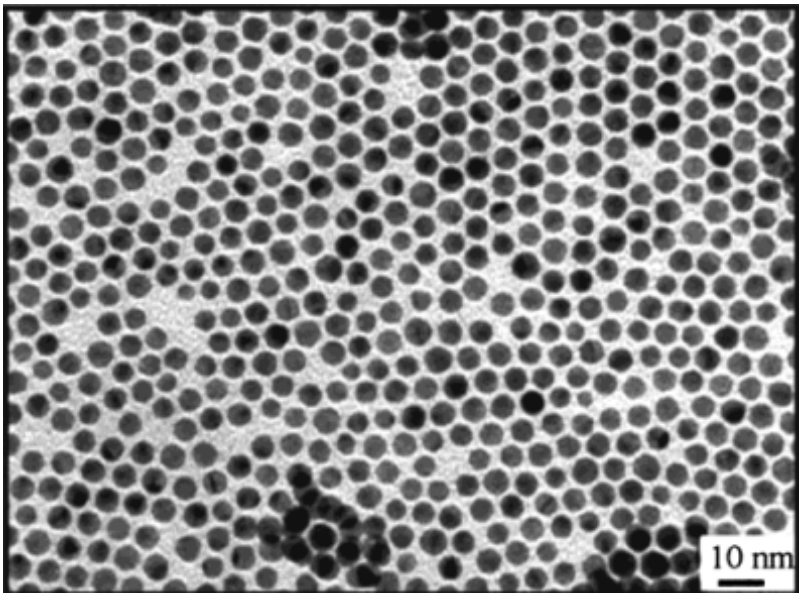


Figure 2.5: Hexagonal arrays of thiolate-encapsulated gold nanoparticles after heat treatment using the Brust-Schiffrin technique [37, Fig. 3].

For example, Olson *et al.* [40] reported reversible template-based self-assembly of gold nanoparticles that solely depends on noncovalent interactions between a supramolecular template complex and functionalized gold nanoparticles. In specific, they use a pseudorotaxane, a supramolecular compound that consists of a linear structure that is both threaded and bound within a macrocylic host via noncovalent bonding interactions [40]. This pseudorotaxane can be synthesized into dimer, trimer, and tetramer shapes, which then facilitate the binding of functionalized gold nanoparticles to produce rod- (near-stripe), triangular- (hexagonal), and square-ordered arrays of gold nanoparticles shown in Figure 2.7. Additionally, ref. [42] demonstrates a host-guest self-assembly method, analogous to template-based self-assembly, that is capable of producing ordered hexagonal domains (see Figure 1.2c).

2.2 Topological Defects and Defect Structures

Defects are ubiquitous across many classes of materials and have been a historically rich field for hard materials (i.e. crystalline phases). Materials classified as "soft matter", representing a middle state between ideally fluid and solid [28], share similar defects to

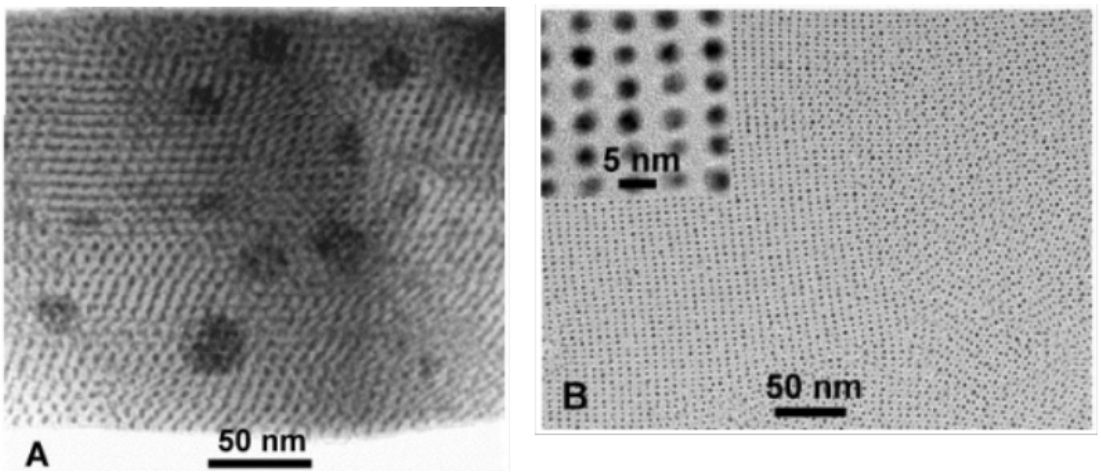


Figure 2.6: Square/hexagonal ordered gold nanocrystal/silica supperlatice thin films via sol-gel self-assembly [22, Fig. 3a-b]: (A) cross-sectional image, and (B) representative plane-view image.

crystalline phases but also experience their own unique defects and related structures. Soft matter includes materials/structures formed from self-assembly, such as block copolymers, colloids, and liquid crystals. In general, defects in soft matter can be broken into three classes: (1) doping defects, (2) connectivity defects, and (3) topological defects [28]. Doping and connectivity defects involve chemical modifications (or manipulations) on the molecular level [28] and are not discussed here.

Topological defects, in continuous systems, represent a class of defects arising from *broken* continuous symmetry [13, 30]. They are *reversible*, in that perturbations to the surrounding environment (temperature, pressure) can cause these defects to annihilate (in pairs) or move throughout the domain [28]. However, they cannot be re-arranged to form local ordered structures without creating a local defect *elsewhere* in the domain [28].

Topological defects also play a critical role in determining the properties of many classes of materials [13]. Rigorous treatment to topological defects gives rise to homotopy theory [13, 30], a rich mathematical subject concerned with quantifying *topological* structures, spaces, and their intricate relationships. Detailed information on the mathematics describing topological defects are provided elsewhere (see refs. [13, 30]). Alternatively, this section provides a cohesive overview of various types of topological defects and defect structures found in self-assembled systems from the perspective of liquid crystal phases.

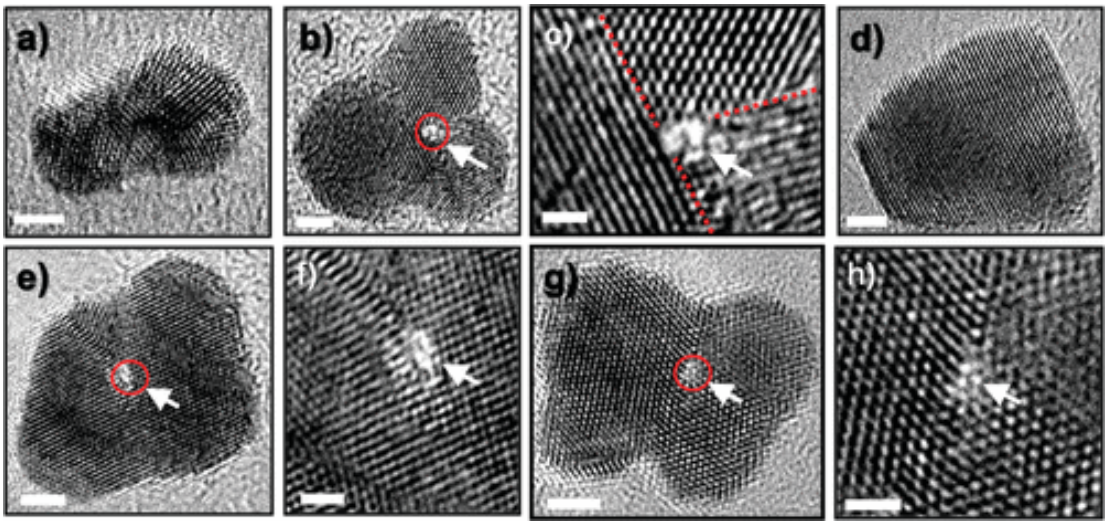


Figure 2.7: Template-based gold nanoparticle synthesis into rod- (near-stripe), triangular- (hexagonal), and square-ordered arrays via noncovalent binding to a pseudorotaxane after electron treatment [40, Fig. 5].

2.2.1 Disclinations and Dislocations

Disclinations and dislocations are two primary types of topological defects found in liquid crystal phases and are traditionally classified as line defects [30]. Dislocations disrupt translational symmetry, while disclinations disrupt rotational symmetry [30] (i.e. orientational order).

Disclinations are common to both nematic and smectic liquid crystal phases [13, 28, 30]. Particularly in 2-dimensional structures, π wedge disclinations with a topological charge (or strength) of $\pm 1/2$ are common. Figure 2.8 shows a drawing of a smectic phase liquid crystal displaying oppositely charged π wedge disclinations.

The Volterra process (or construction) describes the creation of a line defect within an ordered domain and is a common tool to characterize topological defects [13, 19, 28, 30]. In general, a cut is made along the surface plane S which is bounded by some disclination line L [13, 30]. The two lips of the cut surface are then displaced with respect to each other about some axis Ω [13, 30]. For wedge disclination defects, the two lips of the cut surface are translated along the plane S, and therefore Ω is parallel to L [19]. A *wedge* of material is then inserted or removed, depending on the angle of translation, and the material is allowed to "relax" to accompany the newly created defect [13, 19, 30]. Figure 2.9 demonstrates the Volterra process to create a $-\pi$ wedge disclination defect.

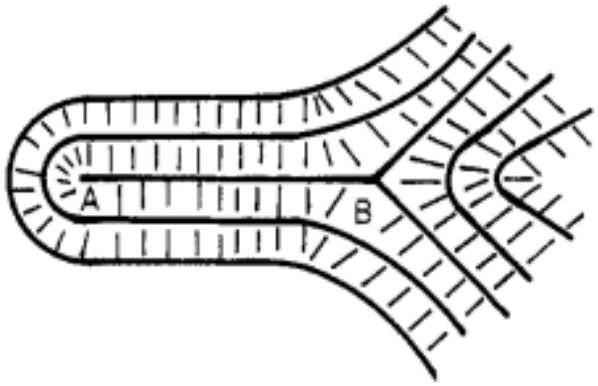


Figure 2.8: Wedge disclinations of varying topological charge in a hypothetical smectic phase liquid crystal [30, Fig. 10]: (A) represents a $+\pi$ wedge disclination with topological charge $+1/2$ while (B) represents a $-\pi$ wedge disclination with topological charge $-1/2$.

Disclinations can also appear in a hexatic liquid crystal phase, where the hexatic phase is defined as an intermediate phase between solid and isotropic liquid phases for 2-dimensional crystals [13]. Disclinations in this phase are $\pi/3$ wedge disclinations, where the core of a $+\pi/3$ wedge disclination contains 5-fold symmetry (i.e. five nearest neighbours), while a $-\pi/3$ wedge disclination core contains 7-fold symmetry (i.e. seven nearest neighbours) [13]. Figure 2.10 demonstrates $\pm\pi/3$ wedge disclinations found in hexatic phases of liquid crystals.

Dislocations are commonly found in smectic phase liquid crystals and can also be characterized by the Volterra process [13, 30]. The main types of dislocation defects are *edge* and *screw* dislocations [13, 30]. The *Burgers* vector is a quantity used to measure the distortion caused by a dislocation defect [13]. It is usually compared to the dislocation line, L, that is perpendicular to the structural plane in question. Relating these quantities, a screw dislocation line is parallel to the Burgers vector, while an edge dislocation line is everywhere perpendicular to the Burgers vector [30].

Edge dislocations are easy to quantify from 2-dimensional surface imaging of nanostructures, as the Burgers vector is parallel to the *image* plane. This is in contrast to screw dislocations, where Burgers vectors are perpendicular to the *image* plane and are difficult to quantify using a 2-dimensional image of surface structure; thus only edge dislocations are considered herein. Burgers vectors for edge dislocations can be computed as [13]:

$$\vec{b} = \oint_{\Gamma} d\vec{u} = k d e_z \tag{2.1}$$

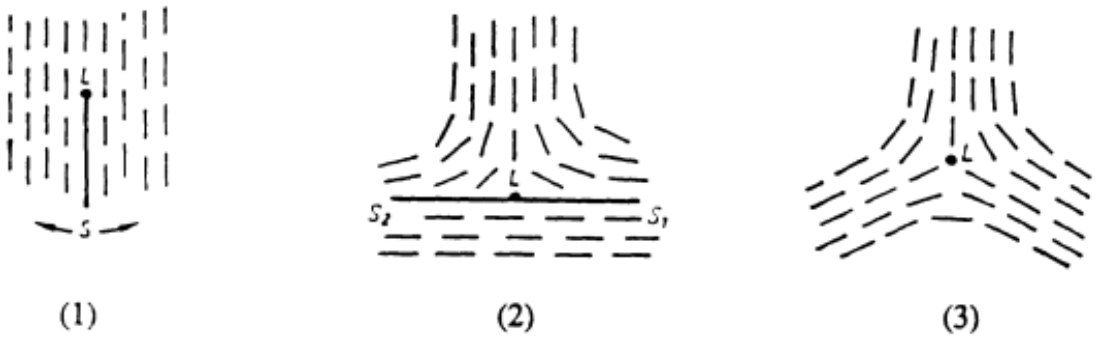


Figure 2.9: Volterra process to create a $-\pi$ wedge disclination defect [19, Fig. 1.5]: (1) A cut is made on the plane S, perpendicular to disclination line L. (2) The two lips, S_1 and S_2, are translated π radians away from each other along an axis Ω, which is parallel to the disclination line L, and ordered material is inserted in this void space. (3) The material is allowed to relax about the newly created defect.

where Γ represents the closed-loop around the edge dislocation, $\vec{u}$ is the displacement field (which for edge dislocations means that $\vec{u}$ is perpendicular to the image plane, or parallel to the dislocation line), k is an integer representing the strength of the defect, d is the repeat distance of the layer or feature, and e_z is the unit vector in the z-axis (where conventionally the z-axis is perpendicular to the image plane and parallel to the dislocation line).

A relatively straightforward method exists to determine the Burgers vector for a dislocation defect [13, Page 513] and is summarized here:

1. Draw a closed-loop *clockwise* around the dislocation and keep track of the number of steps (i.e. along nearest neighbours or lines) on each side of the loop, ensuring that opposing sides of the loop have the same number of steps.

2. If a dislocation is present, the loop will not self-close; in other words, one would need less or more steps to complete the loop and the vector completing this loop is the Burgers vector.

3. The magnitude of the Burgers vector (i.e. strength of dislocation defect) is then the integer multiple of the feature/layer repeat distance (i.e. the quantity kd in equation 2.1) needed to close the loop.

This process can be easily followed for the hypothetical smectic liquid crystal in Figure 2.11, showing edge dislocation defects with similar magnitude and opposite sign.

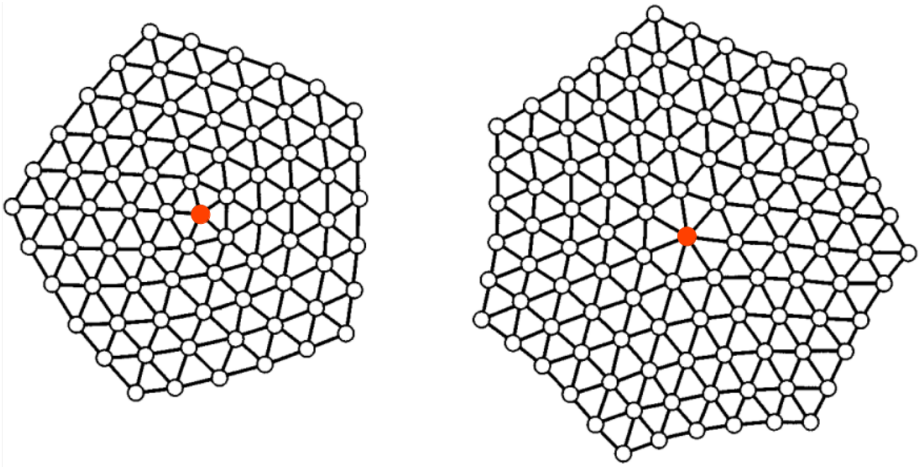


Figure 2.10: Wedge disclinations found in a hypothetical hexatic phase liquid crystal [13, Fig. 9.2.19], where the disclination core sites have been modified with a red circle for this work: (left) $+\pi/3$ wedge disclination where the disclination core displays 5-fold symmetry, and (right) $-\pi/3$ wedge disclination where the disclination core displays 7-fold symmetry.

Dislocations are also common to hexatic phase liquid crystals, where they typically appear as pairs of oppositely charged disclinations [13]. The method previously described to quantify the strength of dislocations in this phase still applies, whereby the Burgers vector is used to trace a closed-loop around the dislocation defect. Figure 2.12 demonstrates a dislocation defect in an otherwise hexagonal array, noting that this defect is composed of two oppositely charged disclination defects [13].

2.2.2 Grain Boundaries

Grain boundaries are a complex defect structure that can be interpreted as a sequence (or array) of dislocation defects [13, 30]. They represent a physical interface separating well-aligned *grains*. The angle of a grain boundary is a useful quantity, dependent on the number of dislocations per unit length in the boundary itself [13]. Suppose that each dislocation in a grain boundary is separated by a distance l_d, where d represents the usual spacing between features/lines in the domain. Geometric considerations can show that the grains on either side of the grain boundary are separated by a rotation of θ radians, computed as [13]:

$$\theta = 2sin^{-1}(d/2l_d) \approx \frac{d}{l_d} \tag{2.2}$$

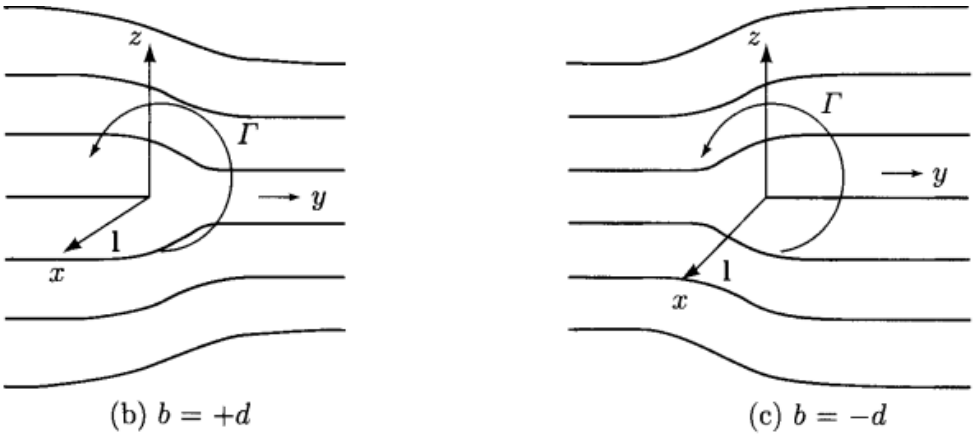


Figure 2.11: Demonstrating edge dislocation defects in a hypothetical smectic phase liquid crystal [13, Fig. 9.2.6]: (left) Burgers vector with magnitude $|\vec{b}| = +d$, and (right) Burgers vector with magnitude $|\vec{b}| = -d$. These values can be derived by applying the process described in the main text.

while noting that this approximation holds for small angles (i.e. low-angle grain boundaries).

The Burgers vector for a grain boundary is also a meaningful quantity and can be computed as follows [13]:

$$\vec{b} = \oint_{\Gamma} d\vec{u} = N_d \; d \tag{2.3}$$

where N_d is the number of unit dislocations in the enclosed loop (Γ).

Figure 2.13 shows a (low-angle) *edge* grain boundary defect structure in a hypothetical smectic phase liquid crystal, displaying the geometric relationship between aforementioned quantities θ, d, l_d, and Γ. Twist grain boundary defect structures are not discussed here, but information can be found in ref. [13, Fig. 9.2.23b].

2.3 Shapelet Functions

Shapelets are a set of localized orthogonal basis functions originally designed for the decomposition and reconstruction of telescopic galaxy images [44]. Astronomy researchers used shapelet functions to compress images of galaxies by factors of 40-90 which drastically reduced required storage capacity [44]. Additionally, shapelet functions have many

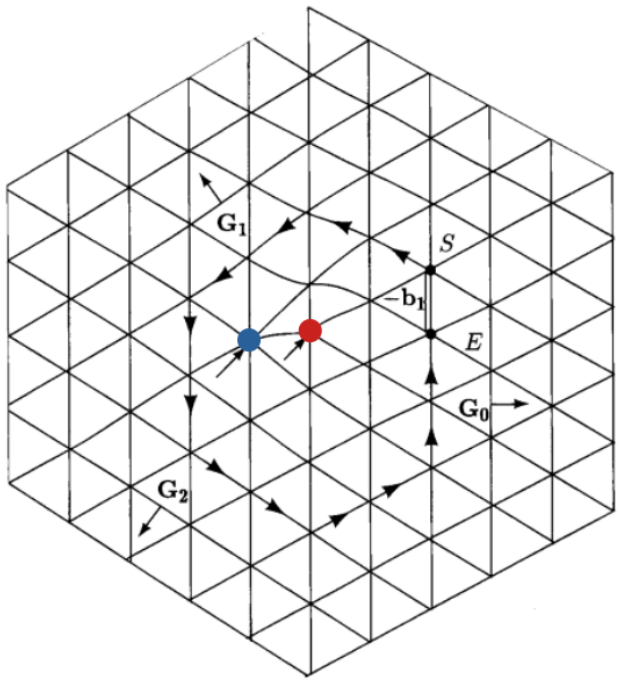


Figure 2.12: Demonstrating a dislocation defect composed of oppositely charged disclinations in a hexagonal array [13, Fig. 9.2.15], where blue and red circles have been added here for this work: tracing the closed loop (clockwise) yields a Burgers vector with magnitude $-d$, where d is the distance between cell vertices. This dislocation is also represented by a pair of oppositely charged disclinations, where the blue vertex exhibits 7-fold symmetry and the red vertex exhibits 5-fold symmetry.

applications in science and engineering (see ref. [51] for details). Shapelet functions form a *complete* mathematical set and can be used to model information from any shape or image function. However, while a subset of the (infinite) shapelet function set is used in practice, relatively small subsets can be used for image analysis with minimal relative error.

2.3.1 Polar Shapelets

Polar shapelets [36] were developed as an analog to the original cartesian formulation to enhance the quality of galaxy reconstruction. These shapelets are *complex* and have unique radial and rotational symmetry properties convenient for image analysis on non-Euclidean geometries (i.e. circles, ellipses). In particular, polar shapelets have two distinct parame-

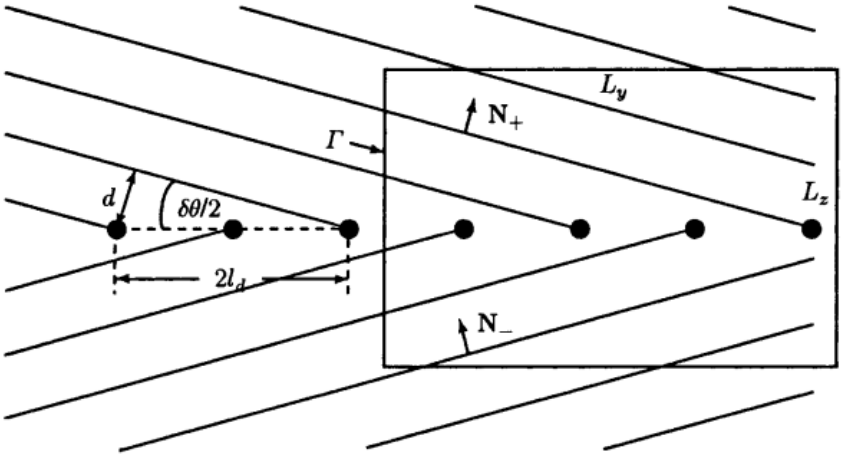
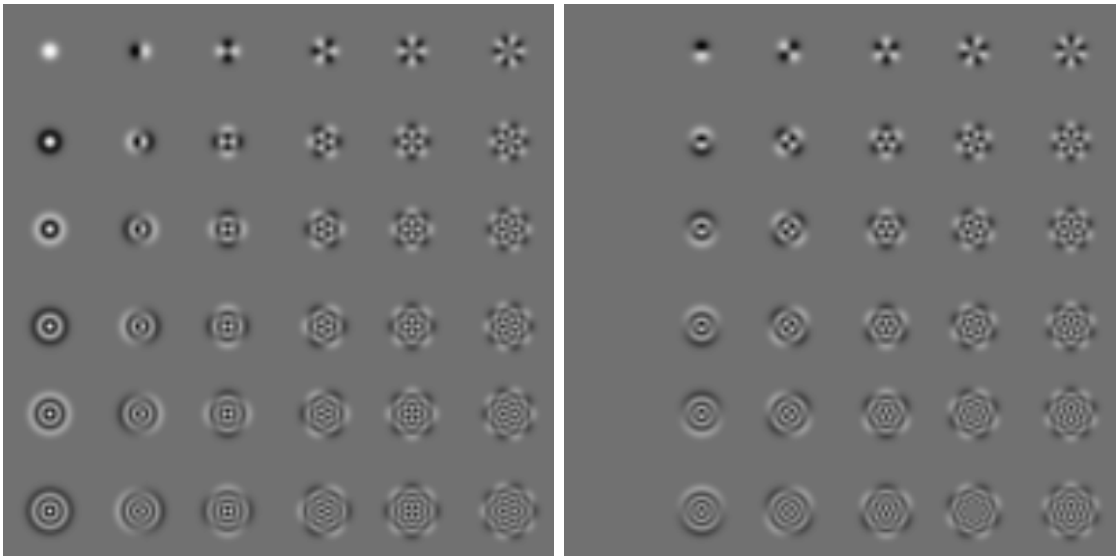


Figure 2.13: Demonstrating a low-angle edge grain boundary defect structure in a hypothetical smectic phase liquid crystal [13, Fig. 9.2.23a], where the geometric relationship between relevant quantities is visibly apparent.

ters, n and m, which control modulation in the radial and polar dimensions respectively [48]. The polar shapelets used in this work are a reformulation of the original function definition [36], proposed in ref. [48], with a further modification to shapelet length-scale proposed in ref. [4]. Figure 2.14 shows the real and complex components of these polar shapelets projected onto a discrete domain for increasing n and m parameters. It is important to note that polar shapelets are herein referred to as just "shapelets".

2.3.2 Pattern Analysis

Suderman *et al.* [49] reformulated shapelet functions to be used in characterization of self-assembled nanostructure domains. They used a specific subset of shapelets $(n, m = 0, \{1, 6\})$ to analyze stripe and hexagonal self-assembled patterns from simulation and microscopy imaging. Akdeniz *et al.* [4] also used this shapelet subset, but they reformulated shapelets to be *orthonormal* and have constant radial scale for increasing m. This reformulation was achieved through an analytical derivation of the relationship between the shapelet length-scale parameter and the characteristic length-scale of the target pattern. This analytical formula allows for the use of arbitrary m shapelets without the need to numerically compute shapelet length-scales, providing the basis for the work done in Section 4.1.



(a) (b)

Figure 2.14: Polar shapelets and their symmetry parameters n and m from ref. [48]: (a) real and (b) imaginary components. $(n, m) = (0, 0)$ begins in the top left, with n increasing downwards, and m increasing to the right. Note that $n = 0$ polar shapelets do not have an imaginary component.

The primary analysis in this work (Chapter 4) uses the orthonormal shapelet formulation with a fixed radial parameter ($S_{nm} \rightarrow S_{0m}$) from Akdeniz *et al.* [4]:

$$S_{0m}(r, \phi; \beta) = \frac{1}{\beta\sqrt{\pi m!}} \left(\frac{r}{\beta}\right)^m e^{-\frac{r^2}{2\beta^2}} e^{-im\phi} \tag{2.4}$$

where the standard polar shapelet length-scale, β, is reformulated to be proportional to the characteristic length-scale of the target pattern, λ, via the following expression [4]:

$$\beta = \frac{f\lambda}{\sqrt{m}} \tag{2.5}$$

where λ is calculated via the dominant Fourier mode of the pattern (see ref. [48]) and f is

a geometric factor that depends on m [4]:

$$f = \begin{cases} 1/4 & m = 1 \\ 1/2 & m = 2 \\ 1/\sqrt{3} & m = 3 \\ 1/\sqrt{2} & m = 4 \\ 1 & m \geq 5 \end{cases} \tag{2.6}$$

To be used in nanostructure characterization, shapelets are projected onto a discrete grid as image processing *filters* (or kernels) which then participate in convolutions with an underlying pattern. The *convolution* is a mathematical operation that takes two input *signals* and produces a third *signal* [46]. In linear systems, the convolution describes the relationship between the input signal (image function), impulse response (filter), and output signal (convolutional response) [46]. The mathematical definition of the *discrete* convolution in two dimensions is as follows [46]:

$$f(x,y) * h(x,y) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(i,j)\ h(x-i, y-j) \tag{2.7}$$

where $f(x,y)$ is an *image* function with M and N discrete points (pixels) in the x and y directions respectively, $h(x,y)$ is a filter (or kernel), and $\{*\}$ represents the convolution operation.

In practice, the discrete convolution is typically computed in Fourier space through the use of the *discrete Fourier transform* [46]. Since the convolution in the time domain corresponds to multiplication in the frequency domain, fewer computations are required [46]. Moreover, the order in which two discrete functions are convolved has no effect on the output signal because the convolution is commutative [46]:

$$f(x,y) * h(x,y) = h(x,y) * f(x,y) \tag{2.8}$$

Since the convolution can be interpreted as a similarity measure between the input signal (image function) and impulse response (filter), it is desirable to obtain the maximum convolutional response between shapelet filters and a target image. However, due to the unique rotational symmetry properties of shapelets, their filter *orientation* can greatly affect the magnitude of convolutional response. To address this problem, Suderman *et al.* [49] developed a *steerable* shapelet formulation using steerable filter theory [23].

Steerable Shapelets

In steerable filter theory [23], a filter is *steerable* if it can be represented as a linear combination of basis filters:

$$f(r, \phi; \theta) = \sum_{j=1}^{M} k_j(\theta) g_j(r, \phi) \tag{2.9}$$

where θ is the angle of rotation of the filter, $k_j(\theta)$ are weighting functions, and $g_j(r, \phi)$ are the basis filters (functions) from the M basis set.

For a *steerable* filter, the optimal filter orientation corresponding to maximum convolutional response is found by setting the first derivative (w.r.t orientation θ) to zero for the convolution between a steerable filter and image function [48]:

$$\frac{d}{d\theta}[I(x, y) * f(x, y; \theta)] = \sum_{j=1}^{M} \frac{d}{d\theta} k_j(\theta)[I(x, y) * g_j(x, y)] = 0 \tag{2.10}$$

where $I(x, y)$ is the target image (function), $\{*\}$ represents the convolution operation, and a necessary change from polar to cartesian coordinates is shown for the steerable filter ($f(r, \phi; \theta) \rightarrow f(x, y; \theta)$) and component basis filters ($g_j(r, \phi) \rightarrow g_j(x, y)\ \forall\ j$) in order to compute convolutions as defined by equation 2.7. This theory also presents a significant reduction in computational complexity to find optimal filter orientations, where the optimal orientation can be found in $\mathcal{O}(1)$ time as opposed to brute-force or searching algorithms with larger complexities [49].

Suderman *et al.* [49] used this framework to determine a *steerable* shapelet formulation with two basis filters (i.e. $M = 2$). To begin, the *steerability* of a more general class of filters that include shapelet functions is demonstrated. Consider a generic *polar* function, $F(r, \phi)$, that is separable in terms of its input variables [49]:

$$F(r, \phi) = f(r)g(\phi) = f(r)e^{-ik\phi} \tag{2.11}$$

Rotating this generic polar function (equation 2.11) by θ radians yields the following expression [49]:

$$\begin{aligned} F(r, \phi + \theta) &= f(r)g(\phi + \theta) \\ &= f(r)e^{-ik(\phi+\theta)} \\ &= f(r)e^{-ik\phi}e^{-ik\theta} \\ &= F(r, \phi)\,[cos(k\theta) - isin(k\theta)] \end{aligned} \tag{2.12}$$

where *Euler's* formula is used to expand the exponential term into trigonometric terms.

Taking the *real* part of this rotated function (equation 2.12) yields a *steerable* formulation with two basis filters [49]:

$$\begin{aligned}
\Re(F(r,\phi+\theta)) &= \Re(F(r,\phi)\,[cos(k\theta)-isin(k\theta)]) \\
&= f(r)\Re([cos(k\phi)-isin(k\phi)]\,[cos(k\theta)-isin(k\theta)]) \\
&= f(r)\,(cos(k\phi)cos(k\theta)-sin(k\phi)sin(k\theta)) \\
&= cos(k\theta)\underbrace{f(r)cos(k\phi)}_{\Re(F(r,\phi))} - sin(k\theta)\underbrace{f(r)sin(k\phi)}_{-\Im(F(r,\phi))} \\
&= cos(k\theta)\ \Re(F(r,\phi)) + sin(k\theta)\ \Im(F(r,\phi))
\end{aligned} \tag{2.13}$$

where the two basis filters are the set $M = \{\Re(F(r,\phi)), \Im(F(r,\phi))\}$ with corresponding trigonometric weighting functions.

As generic polar shapelets are similarly separable in r and ϕ (see Appendix B.1), a rotated shapelet function similar to equation 2.12 can be derived [49]:

$$S_{nm}(r,\phi;\beta) = p(r;\beta)g(\phi) = p(r;\beta)e^{-im\phi} \tag{2.14}$$

$$\Rightarrow S_{nm}(r,\phi+\theta;\beta) = S_{nm}(r,\phi;\beta)\,[cos(m\theta)-isin(m\theta)] \tag{2.15}$$

and taking the real part of this rotated shapelet function yields the *steerable* shapelet formulation with two basis filters [49]:

$$\Re(S_{nm}(r,\phi+\theta;\beta)) = cos(m\theta)\ \Re(S_{nm}(r,\phi;\beta)) + sin(m\theta)\ \Im(S_{nm}(r,\phi;\beta)) \tag{2.16}$$

noting of course that this *steerable* shapelet formulation is the *real* part of the rotated shapelet function (equation 2.15) and is therefore strictly real-valued.

Furthermore, a change of coordinates provides the cartesian *steerable* shapelet formulation that is used in practice (i.e. to participate in convolutions) and is defined as follows [49]:

$$\Re(S_{nm}(x,y;\beta,\theta)) = cos(m\theta)\ \Re(S_{nm}(x,y;\beta)) + sin(m\theta)\ \Im(S_{nm}(x,y;\beta)) \tag{2.17}$$

where θ is now interpreted as a parameter of this *steerable* formulation and the two basis filters form the set $M = \{\Re(S_{nm}(x,y;\beta)), \Im(S_{nm}(x,y;\beta))\}$ with corresponding trigonometric weighting functions.

It is now necessary to solve equation 2.10 using the cartesian *steerable* shapelet formulation to find the optimal filter orientation. For brevity, the M basis filters of the cartesian

steerable shapelet formulation in equation 2.17 are denoted as $\Re(S_{nm}(x, y; \beta)) \rightarrow M_1$ and $\Im(S_{nm}(x, y; \beta)) \rightarrow M_2$. Now,

$$\begin{aligned} 0 &= \frac{d}{d\theta}[I(x,y) * \Re(S_{nm}(x,y;\beta,\theta))] \qquad (2.18) \\ &= \frac{d(cosm\theta)}{d\theta}[I(x,y) * M_1] + \frac{d(sinm\theta)}{d\theta}[I(x,y) * M_2] \\ &= -(m)sin(m\theta)[I(x,y) * M_1] + (m)cos(m\theta)[I(x,y) * M_2] \\ &= -tan(m\theta)[I(x,y) * M_1] + [I(x,y) * M_2] \\ \rightarrow \theta \equiv \theta_{max} &= \frac{tan^{-1}\left([I(x,y) * M_2]/[I(x,y) * M_1]\right)}{m} \qquad (2.19) \end{aligned}$$

where θ_{max} represents the optimal filter orientation corresponding to the maximum (convolutional) response between the cartesian *steerable* shapelet formulation (equation 2.17) and pixel $\{x, y\}$ for arbitrary image function $I(x, y)$. The *magnitude* of this maximum (convolutional) response is subsequently given as:

$$S_{max} = \sqrt{[I(x,y) * M_1]^2 + [I(x,y) * M_2]^2} \qquad (2.20)$$

This translates to a computationally beautiful result, in that *each* standard (non-rotated) complex shapelet function, when projected onto a discrete domain yielding separate real and imaginary filter components (i.e. the *steerable* basis filters), need only be convolved *once* (at a given pixel) to produce a complex-valued response. This complex convolutional response can then be used to compute the optimal filter orientation (equation 2.19) and corresponding maximum response (equation 2.20) for the *real* part of the rotated shapelet function (i.e. the *steerable* shapelet formulation) at some pixel location $\{x, y\}$. In practice, these quantities can be computed via `numpy` [25], where `numpy.angle` and `numpy.abs` provide the optimal filter orientation and corresponding maximum response, respectively. Figure 2.15 shows the *real* part of a rotated shapelet filter (i.e. the cartesian *steerable* shapelet in equation 2.17) projected onto a discrete domain and rotated at its optimal orientation.

Response Distance Method

The response distance method [49] is a supervised learning technique that translates shapelet convolutional responses into a meaningful representation of order by quantifying uniform (i.e. well-ordered) regions apart from those containing local deformities (defects, disorder). The method begins by requiring the user to choose a reference subdomain within

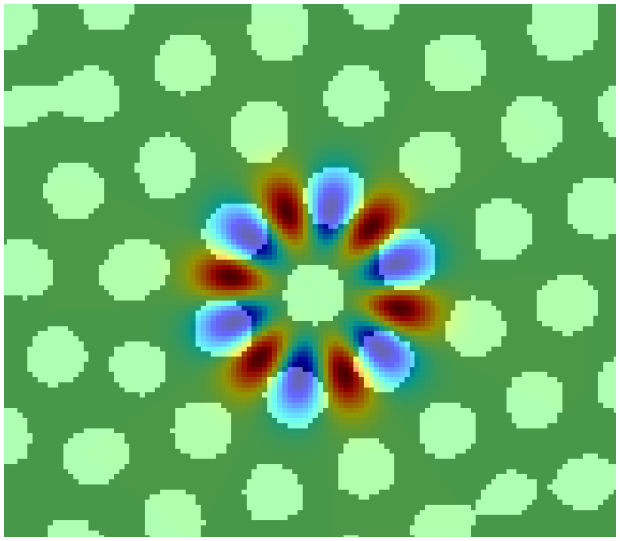


Figure 2.15: A 6-fold ($S_{nm} \rightarrow S_{06}$) cartesian *steerable* shapelet filter rotated at its optimal orientation and superimposed on a hexagonally-ordered domain.

the target image. Ideally, this is a region in the image that is well-ordered and does not contain defects (or disorder). Then, for each m shapelet filter in some pre-defined shapelet subset of length N, the convolution is computed (for each filter) at each pixel in the image domain. This gives rise to a *response vector* of length N, for each pixel, containing the relative m shapelet convolutional responses.

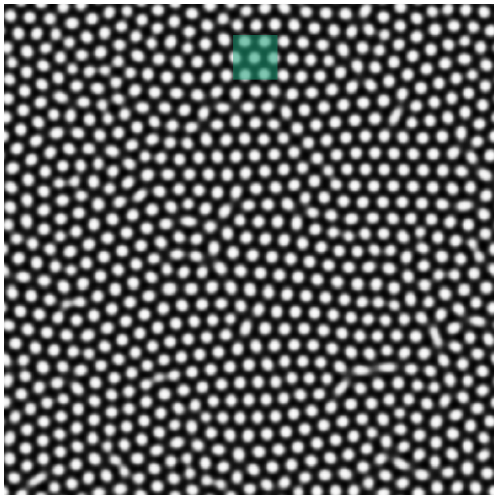
The *response distance* [49], which is computed for each pixel $\{i, j\}$, represents the minimum Euclidean distance between a given response vector and all response vectors in the reference subdomain. Formally, this is given as [49]:

$$d_{i,j} = \min_{n \in \{1,2,\ldots,N\}} \|\vec{R_n} - \vec{r_{i,j}}\|_2 \tag{2.21}$$

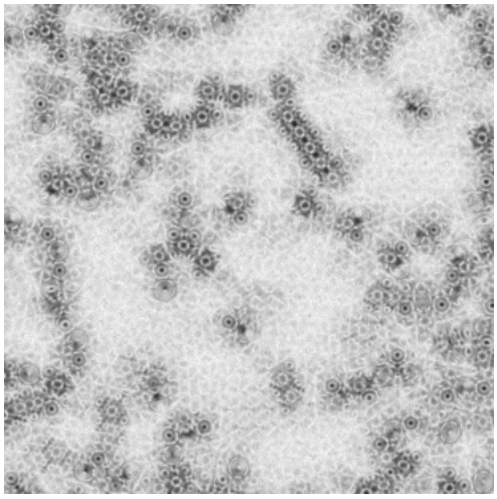
where $\vec{r_{i,j}}$ denotes the test response vector at pixel location $\{i, j\}$ and $\vec{R_n}$ is the n^{th} response vector in the set of N reference response vectors. A larger response distance indicates greater deviation in shapelet response from the reference subdomain, which, if the reference subdomain was chosen correctly (i.e. no observable defects), reveals regions in the image that contain local structural deformities (defects, disorder).

Akdeniz *et al.* [4] noticed that computing the response distance for each pixel in an image is a computationally expensive process. To combat this, they applied k-means clustering [58], an unsupervised machine learning technique, to reduce the computational

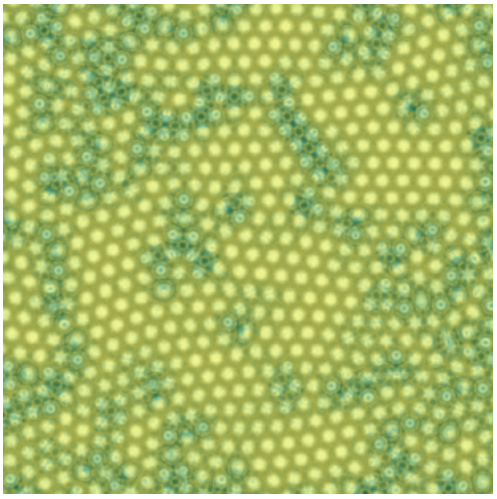
runtime of the response distance method by an order-of-magnitude [4]. In specific, they applied k-means clustering to the reference subdomain to reduce the number of response vectors from $10^2 - 10^3$ (depending on image resolution) to 20 reference response vectors [4]. This value was chosen after conducting a distortional analysis, which analyzed the effect of using k centroids (or response vectors) as a condensed representation of the reference region (for various integer k values) [4]. Figure 2.16 demonstrates application of this computationally-enhanced response distance method [4] to a simulated hexagonal self-assembled nanostructure (Figure 1.1b).



(a)



(b)



(c)

Figure 2.16: The result of applying the computationally-enhanced response distance method [4] to a simulated hexagonal self-assembled nanostructure: (a) the original image (Figure 1.1b) with the highlighted reference subdomain in green, (b) the response distance scalar field, revealing regions in the image that contain local deformities, and (c) the response distance scalar field superimposed onto the original pattern.

Chapter 3

Literature

A brief overview of relevant techniques to characterize self-assembled nanostructures is presented. Disadvantages of each technique is also discussed to assess their effectiveness in practice.

3.1 Bond-Orientational Order Theory

Bond-orientational order theory [12] is a well-known method to quantify the spatial arrangement of a collection of objects. It is primarily concerned with the number of nearest neighbours of a given object [12], similar to the *coordination* number in traditional crystalline physics. Bond-orientational order is quantitatively measured via the complex order parameter, Ψ_n, computed for a given pattern feature as [12, 48]:

$$\Psi_n = \frac{1}{N} \sum_{m=1}^{N} e^{nj\theta_m} \tag{3.1}$$

where n is the degree of order, N is the number of nearest neighbours, and θ_m represents the angle of the "bond" (i.e. line between feature centers) from the current feature to its m^{th} nearest neighbour (relative to some axis).

As this order parameter is complex, it has an inherent magnitude and phase component with different interpretations of the underlying characteristic pattern. The magnitude of Ψ_n is related to the degree of order [12], approaching unity (1) for perfect n-fold symmetry. Moreover, the phase of Ψ_n represents a "symmetry-breaking" parameter and does not

correlate to pattern order [12]. The phase of Ψ_n is computed as the argument of the complex number, which reveals local orientation [48]. This complex order parameter can also be computed for each object (feature) in the domain and then averaged to provide a *global* indication of order (magnitude) and orientation (phase) throughout the domain [48].

Computation of the bond-orientational order parameter requires division of the domain into neighbouring triangles, where *vertices* represent particle centers and *edges* represent "bonds" to nearest neighbours [48]. This can be achieved via Delaunay triangulation (see ref. [16, Ch. 9] for more details). Figure 3.1 shows successful implementation of bond-orientational order analysis on a hexagonal domain [48].

However, bond-orientational order analysis [12] contains inherent limitations, as it is not applicable to patterns with continuous features (i.e. stripe patterns) and does not quantify information on pixel-level scale, but rather feature-level [2].

3.2 Voronoi Diagrams and Entropy

A Voronoi diagram (or Voronoi tesselation) is a mathematical construct that partitions a system of distributed nuclei (i.e. on a 2-dimensional plane) into Voronoi cells, such that for each nucleus, its cell consists of all points in the plane closer to that nucleus than to any other [10]. The mathematical process of constructing a Voronoi diagram is well-described elsewhere (see ref. [16, Ch. 7]). Construction of a Voronoi diagram yields a set of *edges* and *vertices*, as shown in Figure 3.2.

Voronoi diagrams are mathematically related to the Delaunay triangulation (see ref. [16, Sec. 9.2] for more details). They are also directly related to characterization of ordered domains, as the number of edges in a Voronoi cell can be interpreted as the coordination number of its nucleus (i.e. number of nearest neighbours) [10]. This interpretation can lead to identification of defects, such as disclinations and dislocations as demonstrated in refs. [61, Figs. 1 and 3] and [59, Figs. 2 and 3]. Figure 3.3 shows identification of disclinations and dislocations for hexagonally-ordered self-assembled block copolymer micelles via construction of a Voronoi diagram.

Additionally, after constructing the Voronoi diagram, the Voronoi *entropy* can be used as a quantitative measure of (global) order for the underlying system [10]. The Voronoi entropy is computed as a sum of terms for each n-fold symmetry observed in the domain

(beginning with $n = 3$) and is computed as [10]:

$$S_{vor} = -\sum_{n=3}^{M} P_n \; ln(P_n) \tag{3.2}$$

where M represents the highest coordination number (Voronoi cell edges) observed in the domain and P_n is the fraction of Voronoi cells with coordination number n [10].

Suppose the Voronoi entropy (equation 3.2) is to be computed for a domain with *ideal* m-fold symmetry. This implies that $P_m = 1$ and $P_n = 0 \; \forall \; n \neq m$. The Voronoi entropy is then computed as

$$\begin{aligned}
S_{vor} &= -P_m \; ln(P_m) - \sum_{n \neq m} P_n \; ln(P_n) \\
&= -1 \; ln(1) - \sum_{n \neq m} \lim_{x \to 0^+} \; x \; ln(x) \\
&= -\sum_{n \neq m} \lim_{x \to 0^+} \; \frac{ln(x)}{1/x} \\
&= -\sum_{n \neq m} \lim_{x \to 0^+} \; \frac{1/x}{-1/x^2} \\
&= \sum_{n \neq m} \lim_{x \to 0^+} \; x^2/x \\
&= \sum_{n \neq m} \lim_{x \to 0^+} \; x \\
&= 0
\end{aligned} \tag{3.3}$$

where L'Hôpital's rule has been applied to allow for the simplifications shown above. Therefore the Voronoi entropy for a system with ideal n-fold symmetry is zero [10]. Furthermore, a non-zero Voronoi entropy is indicative of *disorder* (or defects) present in the domain, deviating further from ideal n-fold order with increasing Voronoi entropy. It is also important to note that the Voronoi entropy is an *intensive* property (i.e. it is independent of the number of nuclei in the domain) [10].

The Voronoi entropy can be a useful quantity in many scenarios, such as comparing the degree of order between different domains (i.e. between simulations or synthesis procedures) or quantifying order for a domain that evolves with respect to time [10]. Further examples can be found in refs. [31, 32], where Voronoi entropy is computed to compare monolayer ordering in nanostructured films synthesized via electrophoretic deposition.

3.3 Fourier Space Filtering

Yokojima and Shiwa [60] developed a modified Fourier space filtering [14] method that enables characterization of ordered domains, particularly local and global orientation. In this method, a single-mode Fourier series is generated and then used to compute a set of Fourier space filters [2]. These Fourier modes are generated from well-known *uniform* periodic pattern expressions [24]:

$$\psi(\mathbf{x}) = a_1 e^{i\mathbf{k_1}\cdot\mathbf{x}} + a_2 e^{i\mathbf{k_2}\cdot\mathbf{x}} + a_3 e^{i\mathbf{k_3}\cdot\mathbf{x}} \tag{3.4}$$

where $\mathbf{k_n}$ are the basis vectors of the target pattern and a_n are coefficients related to basis vector strength (corresponding to pattern modulation).

For uniform stripe and hexagonal patterns, the basis vectors are [24]:

$$k_1 = \alpha\ \mathbf{j} \tag{3.5}$$

$$k_2 = \alpha\ \left(\frac{\sqrt{3}}{2}\mathbf{i} - \frac{1}{2}\mathbf{j}\right) \tag{3.6}$$

$$k_3 = \alpha\ \left(\frac{-\sqrt{3}}{2}\mathbf{i} - \frac{1}{2}\mathbf{j}\right) \tag{3.7}$$

where $\mathbf{i} = [1, 0]$ and $\mathbf{j} = [0, 1]$ are the standard 2-dimensional cartesian basis vectors, and $\alpha = 2\pi/\lambda$ where λ is the characteristic length scale of the pattern. For stripe patterns, $a_1 \neq 0$ and $a_2 = a_3 = 0$, while for hexagonal patterns $a_1 = a_2 = a_3 \neq 0$ [24].

This logic can also be extended for square patterns, as demonstrated here:

$$k_1 = \alpha\ \mathbf{i} \tag{3.8}$$

$$k_2 = \alpha\ \mathbf{j} \tag{3.9}$$

$$k_3 = \alpha\ \mathbf{0} \tag{3.10}$$

where $\mathbf{0}$ is the zero vector and the corresponding modulation coefficients are $a_1 = a_2 \neq 0$ and $a_3 = 0$.

With Fourier modes defined for ideal patterns, the Fourier space filters can be computed by performing the Fourier transform of the Fourier modes, followed by the introduction of a filtering operator [60]. This leads to the calculation of the orientation *field* [60], revealing local/global orientation of pattern features throughout the domain (see ref. [60, Appendix

A] for more details). Figure 3.4 demonstrates application of the modified Fourier space filtering method [60] to a hexagonal domain of self-assembled block copolymers.

Shortcomings of this method include (1) prerequisite knowledge of the specific pattern order (stripe, square, hexagonal) is required to generate the single-mode Fourier series representation (thus losing generalization qualities), and (2) high computational costs [2].

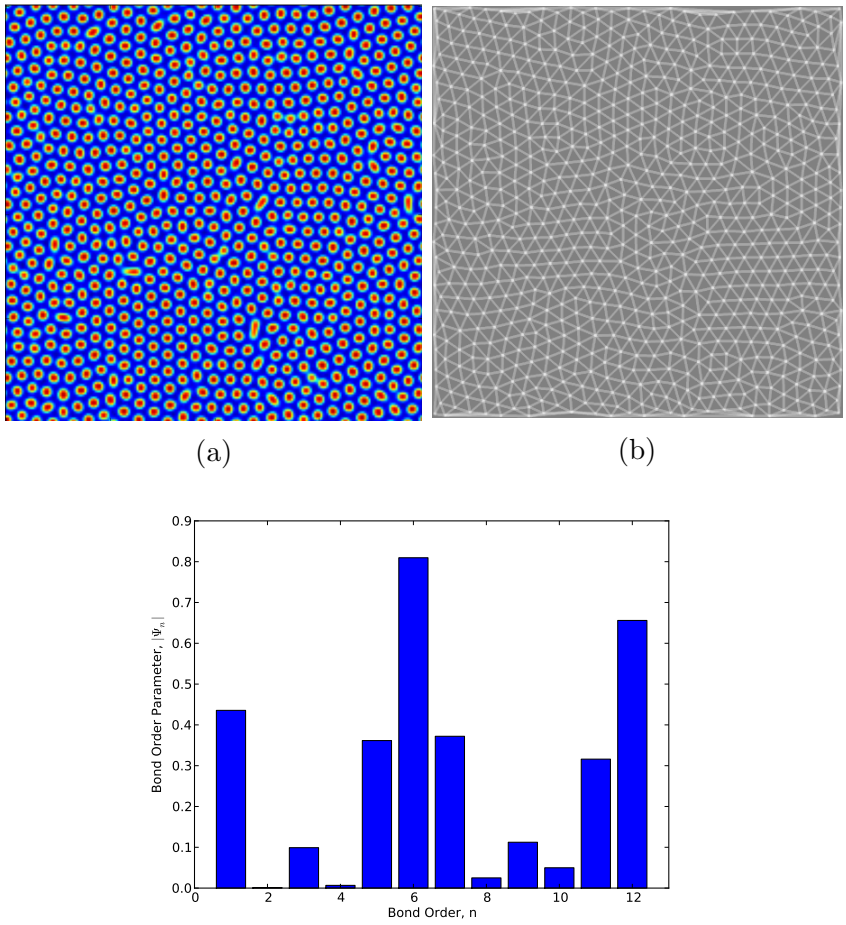


Figure 3.1: An example of bond-orientational order analysis applied to a hexagonal domain [48, Fig. 2.3 and 2.4]: (a) 2-dimensional hexagonal domain, (b) Delaunay triangulation of the domain, and (c) the distribution of bond-orientational order parameter magnitudes over the entire domain.

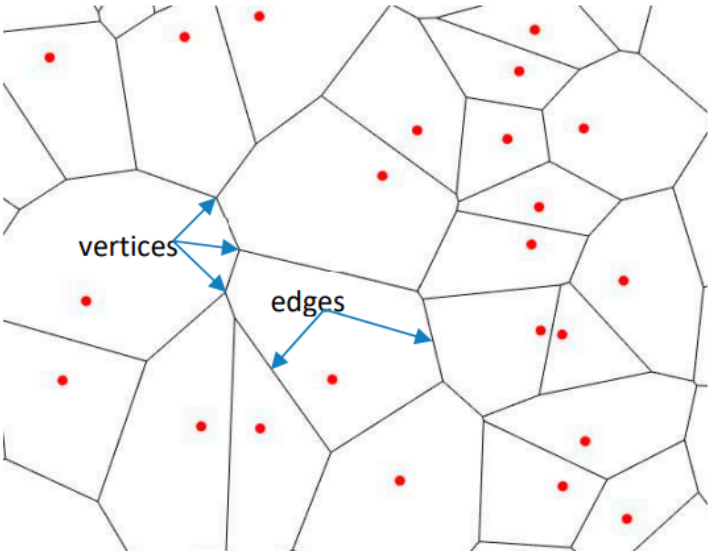


Figure 3.2: Example of Voronoi diagram construction on a hypothetical plane of distributed points [10, Fig. 3], where edges represent the bisector between neighbouring nuclei, and vertices represent points equidistance between three (3) or more nuclei.

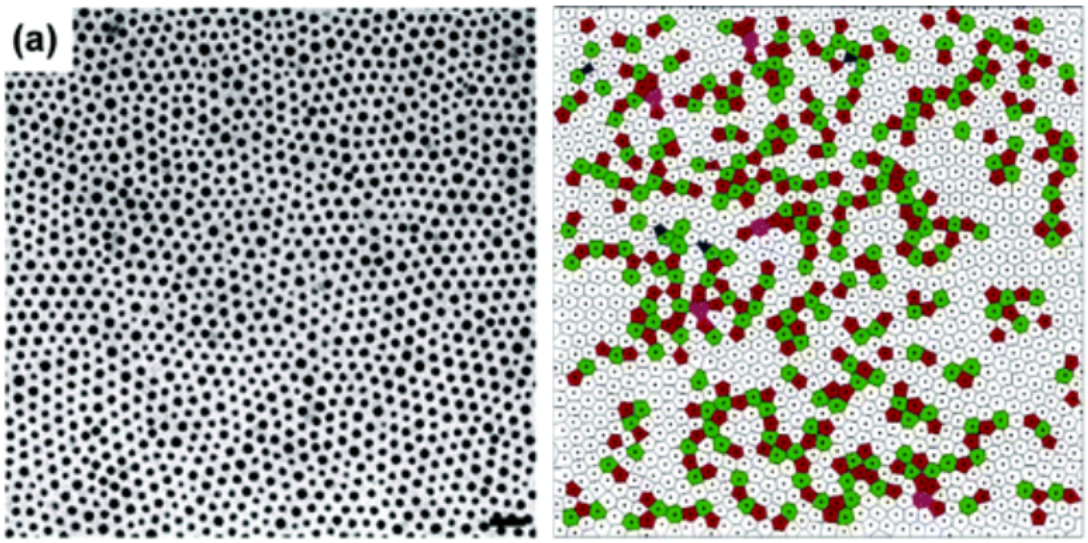


Figure 3.3: Disclination and dislocation defects identified via construction of a Voronoi diagram for self-assembled diblock copolymer micelles [61, Fig. 1], where block copolymer cores with colours blue, red, green, and pink correspond to coordination numbers 4, 5, 7, and 8 respectively.

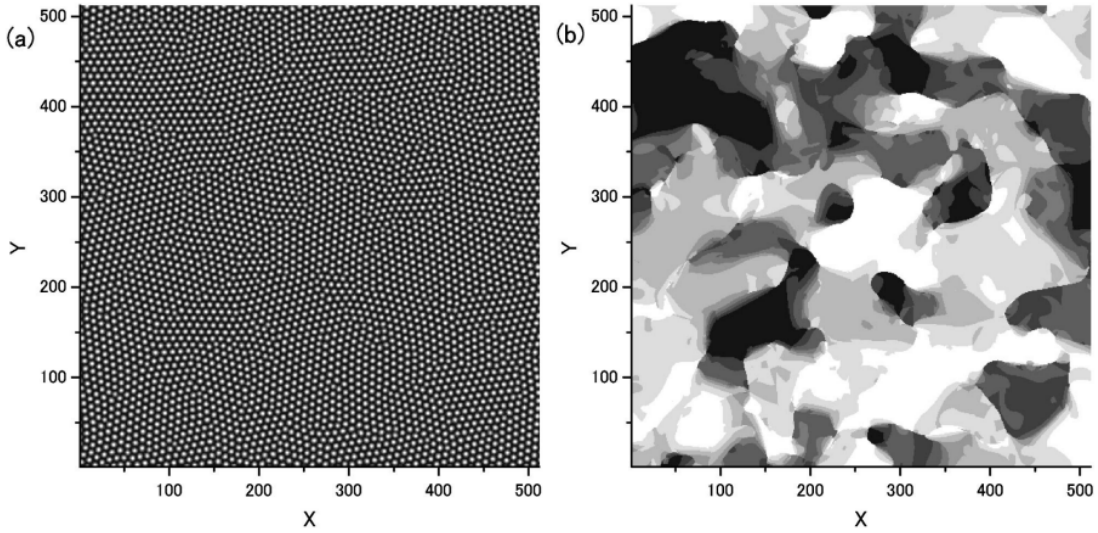


Figure 3.4: Application of the modified Fourier space filtering method applied to hexagonal self-assembled block copolymers [60, Fig. 8]: (left) simulated block copolymer domain with hexagonal order, and (right) the orientation field after applying the modified Fourier space filtering method.

Chapter 4

Shapelet-based orientation and defect identification method for nanostructured surface imaging[1]

4.1 Higher-order Shapelets for Pattern Analysis

Previously developed shapelet response distance methods [4, 49] used lower-order ($m \leq 6$) shapelets based upon two considerations:

- from a qualitative perspective, dominant symmetries observed in surface self-assembly imaging range from 2-fold (stripe) to 6-fold (hexagonal),
- the magnitude of the response to shapelets of higher-order ($m \geq 7$) decreases rapidly with increasing order (m).

However, given the significantly reduced computational complexity involved in the shapelet-based response distance method presented in ref. [4], quantitative analysis of higher-order shapelet response can be considered. This is further motivated by the work of Massey *et al.* [36], which involved the use of higher-order shapelet expansions ($m \leq 20$)

[1] Sections 4.1–4.3 of this chapter form the basis of recently published work: M.P. Tino, R. Suderman, and N.M. Abukhdeir, "Shapelet-based orientation and defect identification method for nanostructured surface imaging," *Nanotechnology*, 35(16), 2024.

to optimize the tradeoff between galactic image reconstruction quality and computational complexity.

In order to quantitatively assess the overall contribution of higher-order shapelets, shapelet response vectors were computed (see Section 2.3) for the reference self-assembly images in Figures 1.1 and 1.2. The increase in computational complexity of the response distance method associated with the incorporation of higher-order shapelets was found to be negligible, $\approx 1 - 10\%$ of the lower-order shapelet computation time, depending on image size. For each image, the sum of shapelet responses over the entire domain (herein the cumulative response) was compared for each individual shapelet component, as shown in Figure 4.1. In agreement with past work, for all surface self-assembly images, lower-order shapelet responses are dominant in magnitude, however, the response for shapelets with $m = [7, 10]$ decays but is still significant. Additionally, for all experimental images, the rate of decay is much lower, with higher-order shapelet responses being $\geq 10\%$ of the maximum response up to $m = 29$. This is especially pronounced for experimental images of hexagonal surface self-assembly (Figure 1.2c).

These results show that higher-order shapelets do capture some information from these self-assembly patterns, especially those with significant noise. This response behavior is similar to what is observed when applying the discrete Fourier transform (DFT) to images with considerable noise and/or spatial discontinuities, where coefficients for the DFT for higher frequency modes have increasing magnitude/moduli [11, Sec. 7.2].

To determine the local pattern features which are not captured using lower-order shapelets with the response distance method [4], analysis that included convolutional responses with higher-order shapelets was performed and the response distance results were compared. Maximum shapelet order (m') was determined for each image by iteratively computing the cumulative response for increasing shapelet order, until the cumulative response was found to be below a normalized threshold value ($\epsilon = 0.1$) with respect to the maximum cumulative response. Referring back to Figure 4.1, the computed truncation index m' is highlighted in the caption for each of the reference self-assembly images in Figures 1.1 and 1.2.

The response distance results for both lower-order and (incorporation of) higher-order shapelets are qualitatively (visually) similar and, consequently, these results are not shown here (see Appendix A for more details). After taking the difference in response distance using lower-order versus (inclusion of) higher-order shapelet response and superimposing this on the original images (Figure 4.2), the relation between higher-order shapelet response and local pattern features is more clear. Focusing on this result for simulated self-assembly images of stripe (Figure 4.2a) and hexagonal (Figure 4.2b) patterns, higher-order shapelet-

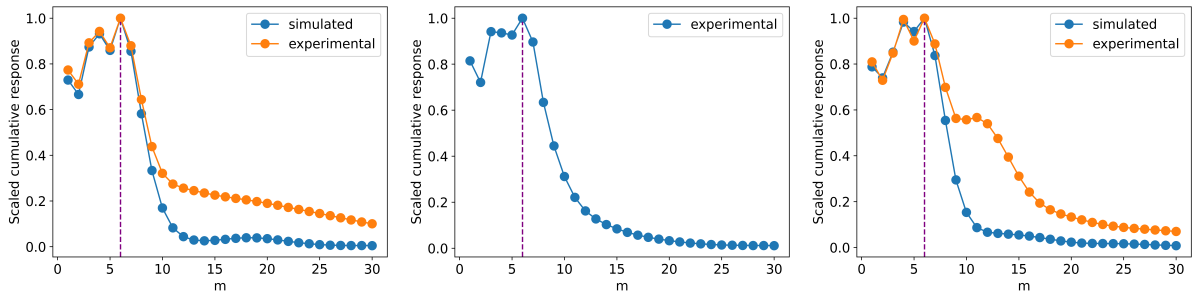


Figure 4.1: The cumulative shapelet response for order $m \leq 30$ shapelets for the reference self-assembly images in Figures 1.1 and 1.2: (a) stripe, (b) square, and (c) hexagonal pattern types. The cumulative response is scaled as a proportion of the *maximum* cumulative response (which happens to occur at $m = 6$ for all pattern types). The dotted purple line at $m = 6$ highlights the cutoff between lower-order ($m \leq 6$) and higher-order ($m \geq 7$) shapelets. The computed maximum shapelet order (m') is as follows: 10 for both simulated stripe (Figure 1.1a) and hexagonal (Figure 1.1b) reference images, and 29, 14, and 23 for the experimental stripe (Figure 1.2a), square (Figure 1.2b), and hexagonal (Figure 1.2c) reference images respectively.

based response is observed to be significantly greater in areas with (i) pattern distortion (curvature or dilation) and (ii) topological defects. This is beneficial compared to the lower-order shapelet result in that, while regions with pattern distortion and defects have lower order than uniform regions, they still have order and are topologically different from regions *without* order (disorder). This is consistent with order parameter representations of local alignment in liquid crystalline phases [17], where order parameters are composed of two components, (order) magnitude and phase. Higher-order contributions to local order have also been shown to be important for quantification of alignment of nanostructures, such as nanorods and nanowires [20]. Topological defects, such as dislocations and disclinations, correspond to regions with non-zero order magnitude and *degenerate* phase [13]. This is most evident for the hexagonal simulation image (Figure 4.2b), where higher-order shapelet-based response directly corresponds to both disclinations (both plus and minus [13, Fig. 9.2.19]) and dislocations, which are composed of adjacent pairs of oppositely charged disclinations for a hexagonal pattern.

Focusing on this result for experimental self-assembly images of stripe (Figure 4.2c), square (Figure 4.2d), and hexagonal (Figure 4.2e) patterns, higher-order shapelet-based response is observed to be significantly greater throughout the image. The difference in re-

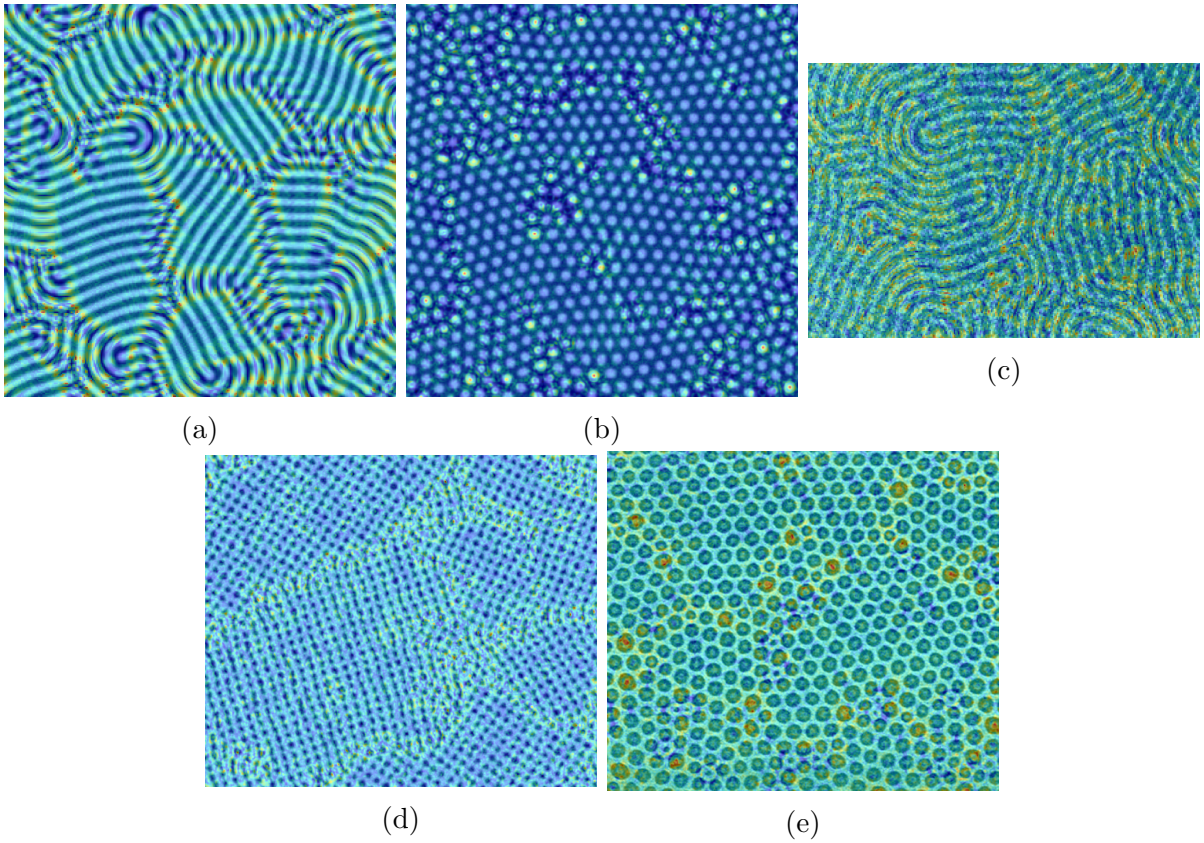


Figure 4.2: The normalized difference in the response distance when using lower-order ($m \leq 6$) and incorporating higher-order ($m \geq 7$) shapelets superimposed onto the original image for the reference self-assembly images in Figures 1.1 and 1.2. Brighter regions correspond to larger differences in response distance.

sponse when using lower-order and including higher-order shapelets can now be attributed to both (i) pattern distortion/defects and (ii) local uncertainty (noise). That is, in addition to significantly increased higher-order shapelet response in areas with distortion and defects, there is also difference in uniform regions where there is high-frequency noise. As with the simulated images, the difference in response distance for experimental images is most significant in areas directly corresponding to disclination defects.

4.2 Orientation

The shapelet response distance method also provides local orientation (at maximum response, see Section 2.3) for each shapelet function. This local orientation is a result of the use of *steerable* [23] shapelet formulations [48], which involves convolution with two (basis) components of each shapelet function, instead of multiple rotations at fixed intervals as with other methods [56]. Quantification of local pattern orientation and spatial variation is as important as quantifying the degree of order itself. However, this is not a straightforward task given that each shapelet of order m may have a different orientation at every pixel within an image. Furthermore, the orientation for each shapelet ranges from $[0, \frac{2\pi}{m}]$, further complicating the use of the local shapelet *orientation vector* in order to approximate local pattern orientation.

An approach to determine local pattern orientation was developed, inspired by a similar approach using bond-orientational order theory [12] but retains the inherent benefits of the shapelet-based approach where image segmentation is not required. Additionally, the shapelet method does not require different segmentation and pattern feature identification steps as with bond-orientational order theory. Instead of integrating all local orientational information provided by the shapelet orientation vector, the dominant rotational symmetry (m") of the pattern is determined and orientation from only this shapelet mode is used. For stripe, square, and hexagonal patterns, the dominant rotational symmetries (m") are 1, 4, and 6 respectively. This results in a maximum orientation value of 2π, $\pi/2$, and $\pi/3$ for stripe, square, and hexagonal patterns (respectively) due to degeneracy in rotational symmetry.

For stripe patterns, the *naive* choice for the dominant rotational symmetry (i.e. $m" = 2$) yielded poor orientation results. Looking at the 2-fold symmetric shapelet function in Figure 2.14, it is possible for this shapelet to experience maximum convolutional response, at the same optimal orientation, for both a vertical pattern (i.e. filter centered on a black feature) and horizontal pattern (i.e. filter centered on a white feature) or vice versa. However, the 1-fold symmetric shapelet does not have this issue. It is also important to point out that, for stripe patterns, the orientation *per grain* is actually the average of orientations of well-defined m"-fold response regions for both black and white pattern features. Since pattern orientation for a given grain is relative to the orientations of other grains, this behaviour is acceptable (i.e. the reference orientation may change between images).

Moreover, unlike local pattern response, local orientation is not a feature-scale attribute which results in highly varying local orientation vectors. The proposed method uses local

m"-fold shapelet orientation, normalized to $[0, 2\pi/m)$, only from regions with relatively high m"-fold shapelet response, masking-out local orientation values where pattern intensity is low. Standard image processing methods (masking, dilation, and blending) are then applied to interpolate local orientation in those regions with low pattern response. This approach, detailed here, is demonstrated for both simulation-based stripe (Figure 1.1a) and hexagonal (Figure 1.1b) self-assembly images shown in Figure 4.3:

Masking: Figures 4.3a and 4.3b show the result of masking the m"-fold shapelet orientation based on a threshold of scaled local response. This threshold, unique to each image, is found via an iterative scheme, beginning with the largest (and least strict) threshold and analyzing the resulting orientation plot after dilation and blending operations. If there exist undefined orientation regions after blending (i.e. either masked-out and/or not interpolated from blending), the masking threshold is reduced and the process is repeated until the post-blending image contains an insignificant number of undefined orientations ($< 1\%$ of all pixels). This results in masking-out regions of the pattern in which either (i) the pattern is relatively uniform with low image intensity (in between pattern features) or (ii) there are defects present (both orientational and translational).

Dilation: Figures 4.3c and 4.3d show the result after dilation of the masked orientation images in order to approximate the local orientation of areas with relatively low pattern response. Morphological greyscale dilation, conveniently available from the Scientific Python library's multi-dimensional image processing module (`scipy.ndimage`) [55], is used to "fill in" void space between neighbouring well-defined response regions. The dilation kernel size is chosen to be 2λ, where λ is the characteristic wavelength of the pattern and also the approximate distance between well-defined response regions. This scale was chosen to allow for adequate overlapping of neighbouring well-defined response regions which ultimately defines orientation in void space and across orientational boundaries.

Blending: Figure 4.4 shows the result of blending the dilated orientation image for both simulated stripe (Figure 1.1a) and hexagonal (Figure 1.1b) self-assembly microscopy images superimposed on the original patterns. The blending operation is performed using a median filter [55] with kernel size 4λ, so that the effective local orientation at each pixel is the median between approximately two layers of surrounding well-defined features. This allows for effective transitions in orientation between well-defined response regions and void space/orientational boundaries.

For both simulated stripe (Figure 4.3a) and hexagonal (Figure 4.3b) masks, the resulting local orientation in uniform areas is qualitatively consistent and correct, with respect to manual (human) analysis. Several uniform regions in both images have gradual (defect-free) changes in local orientation corresponding to elastic-like pattern deformation. This is quantified well, with gradual local computed orientation changes corresponding to those observed manually. For sharp changes in orientation, which correspond to the presence of grain boundaries, there are corresponding sharp changes in computed orientation which validate the specific blending method used in that these orientational interfaces should be sharp and not diffuse. It is important to note that, even though the presented method estimates local orientation values in regions of orientational defects (disclinations), local orientation is degenerate in these areas, given that local m-fold order is present.

The shapelet-based local orientation method was then applied to experimental images of stripe (Figure 1.2a), square (Figure 1.2b), and hexagonal (Figure 1.2c) self-assembly, with results shown in Figures 4.4c, 4.4d, and 4.4e respectively. The presence of noise and non-ideal pattern features is not found to have a significant effect on the accuracy of the computed local orientation using the method described here. Furthermore, as with the shapelet-based response distance method, the local orientation method is generalized for arbitrary pattern orders and is able to analyze patterns with non-convex features (i.e. stripe patterns).

4.3 Defect Identification Method

Changes in local orientation in self-assembly are mediated by the presence of orientation defects, disclinations, and defect structures such as arrays of dislocations and grain boundaries [13]. Thus identification of defects is both complementary to local orientation information and an important property for structure-property relations. While crystalline and liquid crystalline phases have well-defined topological defects [13, 30] (orientational versus translational, dimensionality, charge), in self-assembly there can be significant order deformation which is not energetically possible for order on molecular and atomic scales. Thus the use of the shapelet-based response distance method [4, 49] with user supervision for defect identification is challenging, in that the user would need to identify multiple subdomains with pattern defects of greatly varying character. This is opposite of the original approach where the user selects a subdomain containing uniform (undeformed) pattern order.

To address this challenge, a complementary method for defect identification, the defect response distance method, is proposed which uses k-means clustering [58] of shapelet

response vectors from the *whole* image instead of a pre-defined subdomain. The cluster associated with each pixel (response vector) in the image is then visualized so that the user can select *multiple* clusters associated with one or more defects and/or defect structures. Figures 4.5a, 4.5b, and 4.5c show examples of this decomposition for simulated stripe (Figure 1.1a), experimental square (Figure 1.2b), and simulated hexagonal (Figure 1.1b) self-assembly patterns respectively. Stripe patterns required $k \geq 4$, where cluster centroids associated with ordered regions have higher response for specific subsets of m-fold shapelet symmetries and those associated with defects have relatively equal response. For square and hexagonal patterns, additional clusters were needed ($k \geq 8$ and $k \geq 10$ respectively) with three or four centroids associated with defects, depending on both the amount of noise present in the image and user discretion for selecting defect clusters. Figures 4.5d, 4.5e, and 4.5f show visualizations of cluster centroid weights for shapelet order $m \leq 7$. Higher-order shapelets for each image (maximum shapelet order m' found in Figure 4.1) were included in this analysis, but their corresponding centroid weight values were omitted for brevity.

User supervision is significantly more involved than the response distance method, as previously the user was required to choose a single reference subdomain, but now both the number of clusters k and the set of defect-related cluster centroids must be chosen. Given this information, the response distance method can then be applied separately to each defect cluster by using the centroid vector as the reference, instead of a collective set of reference response vectors associated with uniform order (as in Section 4.1). This defect response distance method was performed for each of the reference self-assembly images in Figures 1.1 and 1.2 and the results are shown in Figure 4.6, with intermediate results for the other experimental images (i.e. similar to those in Figure 4.5) provided in Appendix A. Additionally, a median filter [55] with kernel size $\lambda/2$, where λ is the characteristic length-scale (wavelength) of the pattern, was used to smooth the defect response distance result before superimposing on the original image, as shown in Figure 4.6. This kernel size is significantly smaller than that used for smoothing (blending) of dilated orientation results (see Section 4.2) because of the very localized nature of (topological) defects and defect structures.

Referring to defect response distance results for both hexagonal self-assembly images (Figures 4.6b and 4.6e), the method identifies both positive and negative disclinations (local 5-fold and 7-fold symmetric regions). In Figure 4.6b, several grain boundaries and pairs of adjacent oppositely charged disclinations, corresponding to dislocations in a hexagonal pattern, are also correctly identified. In Figure 4.6e, defect identification results are similarly found to be correct, however, the presence of noise in this image (experimentally determined) is manifested in less strong response. This is particularly true for positive

disclinations, but each one is correctly identified to some extent.

Referring to defect response distance results for both stripe self-assembly images (Figures 4.6a and 4.6c), the method responds much more strongly due to the presence of different grain boundary defect structures. It is important to note that stripe patterns are not composed of convex pattern structures and, instead, they are non-convex and semi-continuous within the image. Thus instead of localized point-like defect structures as is present in the representative hexagonal patterns, defect structures in stripe patterns are found to be line-like, similar to lower dimensional orientational inversion wall defects as seen in liquid crystalline phases [30]. Both positive and negative disclinations are identified by the method, along with individual and arrays of dislocations. Finally, inversion wall-like defect lines are also correctly identified due to the discontinuity of local orientation that they represent.

Referring to defect response distance results for the experimental square self-assembly image (Figure 4.6d), topological defects (disclinations, dislocations) and grain boundary defect structures are adequately identified.

While the defect response distance method is found to correctly identify *regions* which include defects, it is not currently suitable for defect *classification* or quantification of the number of defects present in a given pattern. However, the ratio of the image area to that in which defects are identified is an important quantity that could be used for structure-property relationships.

4.4 Computational Improvements

A computational improvement to the response distance method [49] is presented here via implementation in a compiled programming language.

4.4.1 C++ Implementation

The current Python implementation to compute the response distance (equation 2.21) can potentially result in long runtimes due to high-resolution images, poor computer resources, or a combination of both. It is thus desirable to improve the computational runtime of the existing implementation, especially when high-resolution images are used for analysis.

Several third-party libraries that boost the runtime of Python code exist, but issues are easily encountered. For example, `numba` presents a just-in-time compiler that translates a *subset* of Python code into (fast) machine code [33]. However, this libary is not

applicable to the entirety of the Python language, restricting its widespread use. Alternatively, compiled languages such as C/C++ offer significant computational enhancements compared to dynamic, interpreted languages like Python. These languages are easily supported through Python's built-in `ctypes` framework that allows for runtime execution/call of C/C++ functions and classes from Python. Additionally, `ctypes` includes functionality to simplify data type conversions to allow, for example, C++ pointers to be dereferenced in Python namespaces.

A self-contained and functional implementation of the response distance (equation 2.21) in C++ is as follows:

```
double* rdistance(double* refs, double* test, int numrefs, int numtest, int
mmax) {

// Allocate array on heap for final response distance values
double* final_distances = new double[numtest];

// For each test (response) vector
for (int i = 0; i < numtest; i++) {

    // Allocate array on heap for response distance array
    double* distances = new double[numrefs];

    // For each reference vector
    for (int j = 0; j < numrefs; j++) {

        // Reset summation component of l2 norm for each reference vector
        double accum = 0;

        // For each element pair, add to accum the square of the difference
        for (int p = 0; p < mmax; p++) {
            // Dereference arrays to get specific element value
            double refval = *((refs + j * mmax) + p);
            double testval = *((test + i * mmax) + p);
            accum += pow(testval - refval, 2);
        }
        // Perform square root to get l2 norm value
        distances[j] = sqrt(accum);
    }
    // Use std::min_element to return minimum value in (unsorted) array
    double* min_distance = std::min_element(distances, distances + numrefs);
    final_distances[i] = *min_distance; // dereference and assignment

    delete[] distances; // clear heap memory!
}
return final_distances;
}
```

Table 4.1: Comparing mean computational runtimes to compute the response distance between Python and C++ implementations. On average, a 14-16x speed-up was obtained using the C++ implementation.

Image Dimensions (pixels)	**Mean Python Runtime (s)**	**Mean C++ Runtime (s)**	**Times Faster (x)**
296 × 296	4.0919	0.2869	14.26
592 × 592	16.6772	1.1586	14.39
1036 × 1036	50.6047	3.4773	14.55
1776 × 1776	149.2476	9.0883	16.42

4.4.2 Comparing Python and C++ Implementation Runtimes

The reference simulated stripe self-assembly image (Figure 1.1a) was resized multiple times and this small image set was then used to compare the computational runtime associated with the response distance (equation 2.21) between Python and C++ implementations. For each image, five consecutive runs were performed (for Python and C++) and these five runtimes were averaged to produce the mean computational runtime. To ensure stable results, all computations were performed on the same machine. Additionally, the exact same reference region (although scaled) and number of clusters for k-means clustering [58] were used for each image to allow for appropriate comparisons of runtime results.

Table 4.1 shows the runtime comparison results, demonstrating that the C++ implementation of the response distance provides a speed-up (on average) of 14-16x over the implementation in Python. This is very beneficial for all users, especially for those using low-resource machines in the presence of high-resolution microscopy images. It is also useful for users who wish to compute the response distance in an automated fashion for multiple images, which is now possible on the scale of minutes as opposed to hours using the previous Python implementation.

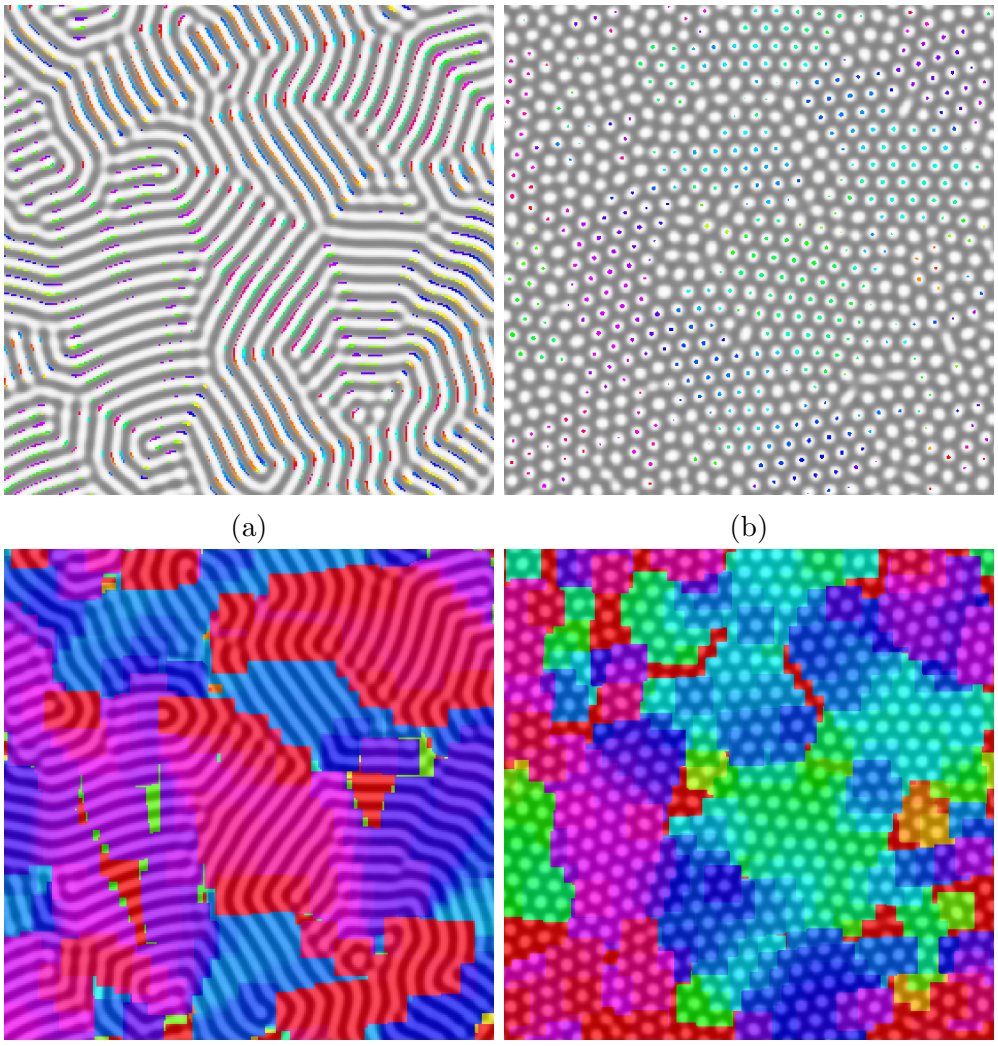


Figure 4.3: Masking and dilation operations applied to simulated patterns: (a-b) masking operation applied to simulated stripe (Figure 1.1a) and hexagonal (Figure 1.1b) self-assembly images respectively, (c-d) dilation operation applied to simulated stripe (Figure 1.1a) and hexagonal (Figure 1.1b) self-assembly images respectively using kernel size 2λ.

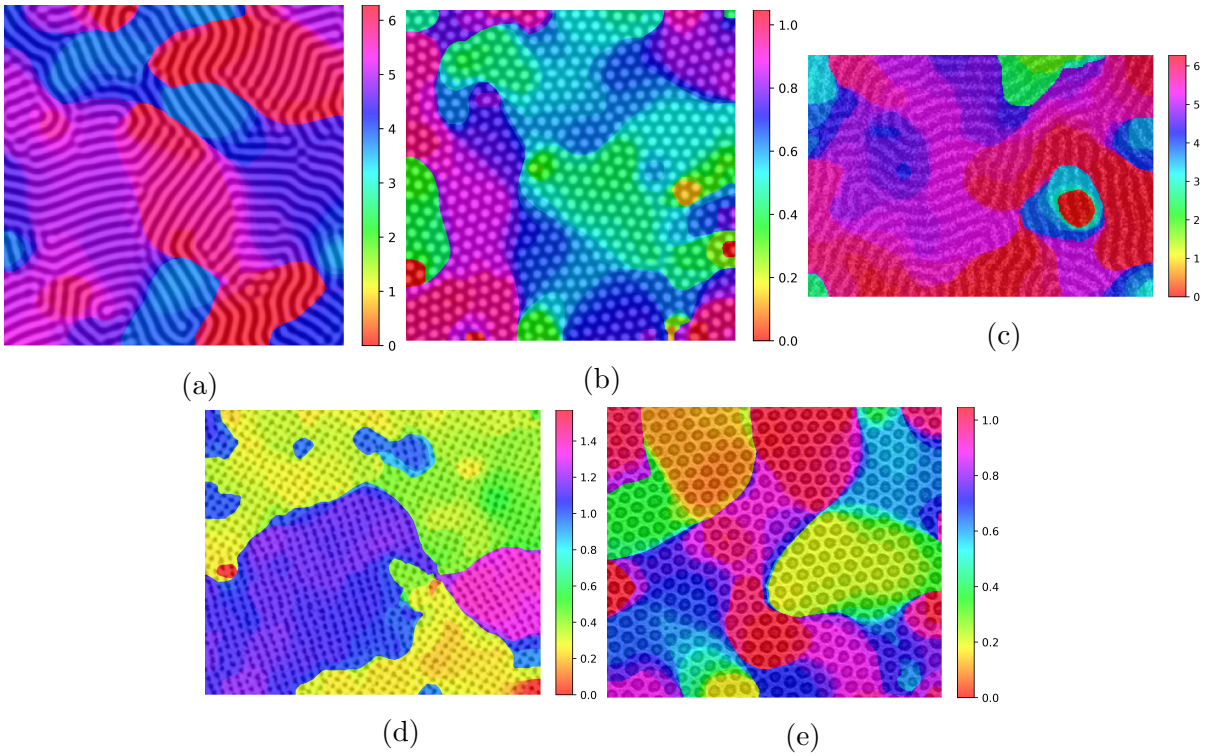


Figure 4.4: The result of applying the blending operation via median filter [55] and superimposing this result onto the original image for the reference self-assembly images in Figures 1.1 and 1.2: (a-b) simulated stripe (Figure 1.1a) and hexagonal (Figure 1.1b) images respectively, and (c-e) experimental stripe (Figure 1.2a), square (Figure 1.2b), and hexagonal (Figure 1.2c) images respectively. Cyclic colour map "hsv" from `matplotlib` [27] is used to account for degeneracy in shapelet rotational symmetry.

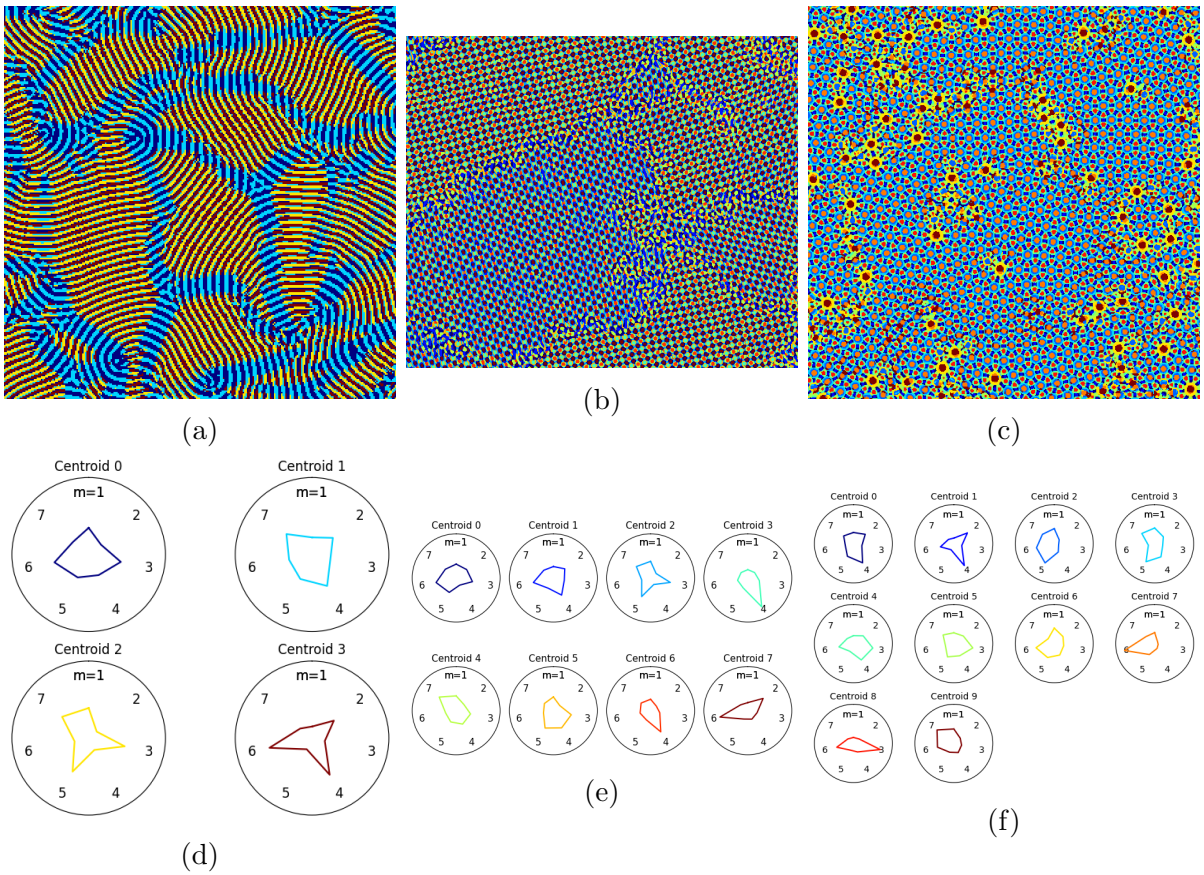
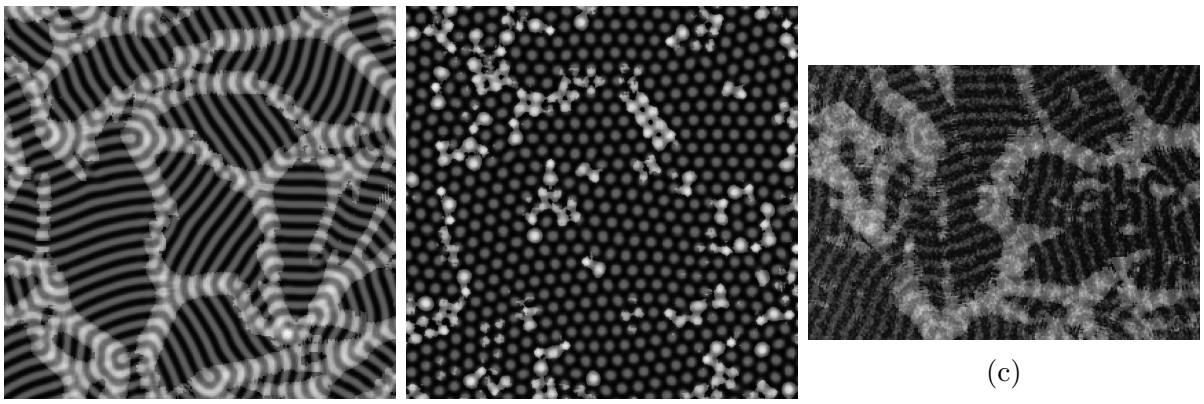
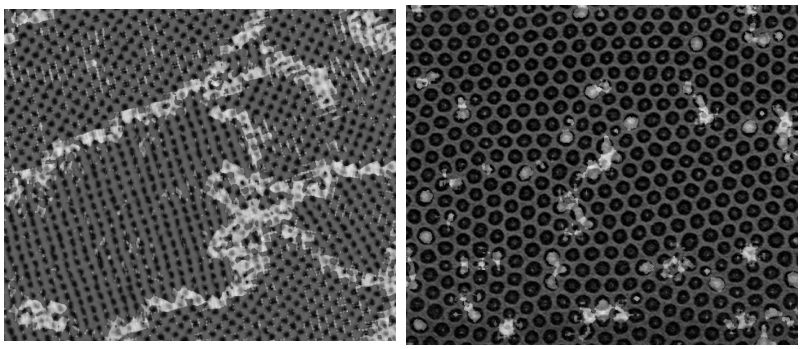


Figure 4.5: Decomposition of shapelet response vectors into k centroid vectors via k-means clustering [58], where $k = 4$, 8, and 10 for stripe, square, and hexagonal patterns respectively: (a-c) visualizing the cluster associated with each pixel (response vector) and (d-f) radar chart representations of the centroid vector weights for $m \leq 7$ shapelets. Centroids selected as defects were $\{0, 1\}$, $\{0, 1, 4\}$, and $\{2, 5, 6, 9\}$ for the (a/d) stripe (Figure 1.1a), (b/e) square (Figure 1.2b), and (c/f) hexagonal (Figure 1.1b) self-assembly images.



(a) (b) (c)



(d) (e)

Figure 4.6: The defect response distance superimposed onto the original pattern image after using a median filter [55] with kernel size $\lambda/2$ for smoothing, which identifies defects for the reference self-assembly images in Figures 1.1 and 1.2: (a) simulated stripe (Figure 1.1a), (b) simulated hexagonal (Figure 1.1b), (c) experimental stripe (Figure 1.2a), (d) experimental square (Figure 1.2b), and (e) experimental hexagonal (Figure 1.2c) images.

Chapter 5

Shapelets: A Python package implementing shapelet functions and their applications[1]

5.1 Summary

Shapelets is a Python-based software package that implements several shapelet functions [44] and some of their significant applications in science and astronomy. Shapelet functions are a complete and orthogonal set of localized basis functions with mathematical properties convenient for manipulation and analysis of images from a broad range of applications. Over the past few decades, there have been several different shapelet function formulations developed and applied in the areas of astronomy/astrophysics [7, 8, 18, 34, 36, 44], self-assembled nanostructures [4, 49, 52], computational neuroscience [45, 54], and medical imaging [57].

The shapelets software package provides reference implementation and documentation for four different shapelet functions: cartesian [44], polar [36], orthonormal polar with constant radial scale [4], and exponential [7]. Additionally, the shapelets package provides implementations of several shapelet applications, including astronomy (galactic image decomposition and reconstruction [36, 44]) and self-assembly (nanostructure characterization

[1]This chapter forms the basis of recently published work: M.P. Tino, A.Y. Abdulaziz, R. Suderman, T. Akdeniz, and N.M. Abukhdeir, "Shapelets: A Python package implementing shapelet functions and their applications," *Journal of Open Source Software*, 9(95), 2024.

[4, 49, 52]). The programming and documentation style of shapelets is based on that of `scipy.special` [55].

For ease of use, shapelets also provides a text-based configuration file interface combined with Python entry points (custom terminal commands) to improve accessibility for a broad range of potential users in science and engineering, especially those without a programming background. The text-based configuration file interface allows users to detail specific parameters in a plain-text file. The desired analysis detailed by the configuration file can then be invoked from the command line via `shapelets config`. Additionally, several unit tests were created to ensure robustness in the package software. Running the unit tests written for the shapelets package can be invoked via `shapelets-test` from the command line.

Last, the shapelets package includes a set of detailed examples which demonstrate usage of the software via text-based configuration files (and command-line arguments) or standard Python programming. These examples cover both the implemented astronomy and self-assembly applications/methods, providing users with a template to develop their own code for shapelet-related analyses.

5.2 Statement of Need

Shapelets are a class of complete and localized orthogonal basis functions with a broad range of applications in image processing and reconstruction [4, 36, 44, 49, 52]. Despite their increasingly widespread use, there is currently no single software package that is both broadly accessible (i.e. written in Python or other high-level programming language) and implements several useful applications. Currently, there exists an open-source astronomy-focused shapelet software package[2] [36], however, it is written in the Interactive Data Language (IDL) which is not widely used in the science and engineering communities. Furthermore, this package has not been updated in over a decade.

Given the increasingly broad usage of shapelets in areas outside of astronomy/astrophysics, an open-source Python-based shapelets software package would provide access to these functions and their applications for a larger community, along with facilitating open-source scientific software development through this centralized package allowing for contribution and collaboration in shapelet-related research.

Similarly, quantification of structure-property relationships for nanomaterials is critical for continued progress in materials research [2]. This is especially true for nanomaterials

[2] https://www.astro.dur.ac.uk/ rjm/shapelets/code/index.php

Table 5.1: A summary of the various shapelet function definitions provided by the shapelets Python package.

Shapelet Functions	**Description**
Cartesian	Cartesian shapelets [44] via `shapelets.functions.cartesian1D`, `shapelets.functions.cartesian2D`
Polar	Polar shapelets [36] via `shapelets.functions.polar2D`
Orthonormal polar	Orthonormal polar shapelets with constant radial scale [4] via `shapelets.functions.orthonormalpolar2D`
Exponential	Exponential shapelets [7] via `shapelets.functions.exponential1D`, `shapelets.functions.exponential2D`

with complex spatially-varying patterns, such as self-assembled materials [2]. There are other methods to characterize nanostructure, such as bond-orientational order analysis [12], but these methods do not provide pixel-scale information and do not have readily available open-source software implementations. Methods to quantify nanostructure order, such as those implemented in the shapelets package, would significantly advance (nano)materials research and provide researchers with accessible tools to quantify order for their own material domains.

The overall aim of the shapelets package is to address these needs through (1) providing well-documented and accessible code for researchers interested in using these shapelet functions and existing applications and (2) promoting open-source collaboration for future development of shapelet-related research and potential applications.

5.3 Features

Table 5.1 summarizes the different shapelet functions implemented in the shapelets package. Additionally, table 5.2 summarizes the specific shapelet applications implemented in the package.

Table 5.2: A summary of the various shapelet applications provided by the shapelets Python package.

Shapelet Applications	**Description**
Galaxy decomposition	Galactic image decomposition and reconstruction [44] via `shapelets.astronomy.decompose_galaxies`
Response distance	Response distance method for self-assembly microscopy imaging [49] via `shapelets.self_assembly.rdistance`
Orientation	Local pattern orientation for self-assembly microscopy imaging [52] via `shapelets.self_assembly.orientation`
Defect identification	Defect identification method for self-assembly microscopy imaging [52] via `shapelets.self_assembly.defectid`

5.4 Usage and Examples

The shapelets Python package can be used in two different ways: text-based configuration files or directly via interactive or script-based Python programming. The text-based user interface is centered around configuration files and the CLI (command line interface). Each use of the shapelets package (via configuration files) should have a main directory (here called "shapelets_example") with standard subdirectories and required files as shown in Figure 5.1.

```
shapelets_example
├── config
├── images
│   ├── image_1.png
│   └── image_2.png
└── output
```

Figure 5.1: Sample directory hierarchy.

The main (`shapelets_example/`) directory contains the main configuration file (i.e. `config`) which is a plain-text file specifying parameters and methods to be used for the analysis. Acceptable parameters and options for the configuration file can be found throughout the examples documentation, depending on the shapelet-based method being applied. The `images/` subdirectory contains data for image analysis. The `output/` subdirectory is created by the shapelets software and contains output data/images based on the analysis specified in the configuration file (`config`).

Additionally, several detailed image processing examples were developed for shapelets that demonstrates usage of both the astronomy and self-assembly submodules. For each example, code and documentation is provided for users for both the text-based configuration file method or standard Python programming. For brevity, these examples are not shown here as they largely demonstrate the methods described in Chapter 4. For more information, readers are encouraged to visit the official shapelets repository[3] which also contains a link to the official website.

[3] www.github.com/uw-comphys/shapelets

Chapter 6

Higher-Order (n = 1) Shapelets

6.1 Motivation

All previous shapelet-based pattern analysis (refs. [4, 49], Chapter 4) used $n = 0$ (polar) shapelet functions corresponding to rapid radial decay. To date, higher-order n shapelets have not been used for any related pattern analysis. Higher-order n shapelets are of great interest due to their radial symmetry, providing the opportunity to conduct regional (i.e. less-local) analysis of pattern structure/order. More specifically, $n = 1$ shapelets have one degree of radial symmetry and could provide insight into two layers of nearest neighbours for a given pattern feature. The physical profile of $n = 1$ shapelets can be found in Figure 2.14.

As mentioned in Section 2.3, shapelet functions form a complete basis and are inherently orthogonal. An *orthonormal* formulism for $n = 0$ shapelets was presented in ref. [4], providing a significant computational benefit to the shapelet-based approach in that the normalization constant can be analytically computed as opposed to numerically estimated. A similar formulism for higher-order ($n = 1$) shapelets is presented here to provide the same benefit and lay the foundation for future shapelet-based pattern analysis.

The functional definition of $n = 1$ shapelets can be derived from the generic polar shapelet definition provided in ref. [4] (see Appendix B.1 for more details), but the result is shown here for brevity:

$$S_{1m}(r, \theta; \beta) = \frac{c_{1m}}{\beta} \left(\frac{r}{\beta}\right)^m \left(1 + m - \left(\frac{r}{\beta}\right)^2\right) e^{-\frac{r^2}{2\beta^2}} e^{-im\theta} \tag{6.1}$$

where β is the shapelet length-scale and m is a rotational symmetry parameter where $\{m \in \mathbb{Z} \mid m \geq 0\}$.

This higher-order shapelet function can be forced into an orthonormality relation as done previously for $n = 0$ shapelets [4]:

$$\int_0^{2\pi} \int_0^{\infty} S_{1m} S^*_{1m'} \, r dr d\theta = \delta_{mm'} \tag{6.2}$$

where $\delta_{mm'}$ is the Kronecker delta and S^* represents the complex conjugate.

Equation 6.2 can be further decomposed by recognizing that the shapelet formulation in equation 6.1 is *separable* in r and θ [4],

$$\left(\frac{c_{1m} c_{1m'}}{\beta^2}\right) G_{m,m'}(\theta) \; F_{m,m'}(r;\beta) = \delta_{mm'} \tag{6.3}$$

where c_{1m} represents the normalization constant for the S_{1m} shapelet,

$$G_{m,m'}(\theta) = \int_0^{2\pi} e^{-i(m-m')\theta} d\theta \tag{6.4}$$

and

$$F_{m,m'}(r;\beta) = \int_0^{\infty} \frac{r^{(m+m'+1)}}{\beta^{(m+m')}} \left(1 + m - \left(\frac{r}{\beta}\right)^2\right) \left(1 + m' - \left(\frac{r}{\beta}\right)^2\right) e^{-\frac{r^2}{\beta^2}} dr \tag{6.5}$$

6.2 Demonstrating Orthogonality

It is imperative to demonstrate orthogonality for higher-order ($n = 1$) shapelets before orthonormalization. Noting that the quantity $\{m - m'\}$ must be an integer, equation 6.4 is easily solved for the $m \neq m'$ case:

$$\begin{aligned} G_{m \neq m'}(\theta) &= \int_0^{2\pi} e^{-i(m-m')\theta} d\theta \\ &= \frac{i}{m - m'} \left(e^{-i(m-m')2\pi} - 1\right) \\ &= \frac{i}{m - m'} \left(cos[2\pi(m - m')] - isin[2\pi(m - m')] - 1\right) \\ &= \frac{i}{m - m'} \left(1 - 0 - 1\right) \\ &= 0 \end{aligned} \tag{6.6}$$

and for the $m = m'$ case:

$$\begin{aligned} G_{m=m'}(\theta) &= \int_0^{2\pi} e^{-i(0)\theta} d\theta \\ &= \int_0^{2\pi} d\theta \\ &= 2\pi \end{aligned} \tag{6.7}$$

Repeating this analysis for equation 6.5 requires more careful consideration. For the case of $m \neq m'$, consider the limit as $r \to \infty$ of the integrand of equation 6.5. Ignoring the β term for brevity, this becomes:

$$\lim_{r\to\infty} \frac{r^{(m+m'+1)} \left(1 + m - \left(\frac{r}{\beta}\right)^2\right) \left(1 + m' - \left(\frac{r}{\beta}\right)^2\right)}{e^{\frac{r^2}{\beta^2}}} \tag{6.8}$$

It should be apparent to the reader, without explicit demonstration, that repeated application of L'Hôpital's rule to equation 6.8 allows for the numerator to vanish while the denominator remains indefinitely. Therefore this limit is equal to zero and the integrand of equation 6.5 converges as $r \to \infty$. This analysis can be repeated for the case of $m = m'$, ultimately reaching the same conclusion that the integrand converges as $r \to \infty$; however this analysis is withheld here for brevity.

6.2.1 The Comparison Test

It is also necessary to demonstrate that the integral in equation 6.5 is *finite* in order to demonstrate orthogonality of this higher-order shapelet class. The comparison test is used here to facilitate this demonstration for the $m \neq m'$ case.

Consider the following two functions:

$$g_{m\neq m'}(r;\beta) = \left(1 + m - \left(\frac{r}{\beta}\right)^2\right) \left(1 + m' - \left(\frac{r}{\beta}\right)^2\right) \tag{6.9}$$

and

$$h_{m\neq m'}(r;\beta) = \frac{r^4}{\beta^4} + (1 + m)(1 + m') \tag{6.10}$$

Expansion of both functions demonstrates the following inequality:

$$\begin{gathered} g_{m\neq m'}(r;\beta) \leq h_{m\neq m'}(r;\beta) \\ 1+m+m'+mm'-\frac{r^2}{\beta^2}(2+m+m')+\frac{r^4}{\beta^4} \leq \frac{r^4}{\beta^4}+1+m+m'+mm' \\ -\frac{r^2}{\beta^2}(2+m+m') \leq 0 \end{gathered} \tag{6.11}$$

while noting that $\beta > 0$, $\{m, m'\} \geq 0$, and this inequality holds $\forall\ r \in (-\infty, \infty)$. Moreover, this expression is only *equal* for $r = 0$.

Substituting equation 6.10 for equation 6.9 in the integrand of equation 6.5 (for the $m \neq m'$ case) yields the complementary function:

$$E_{m\neq m'}(r;\beta) = \int_0^\infty \frac{r^{(m+m'+1)}}{\beta^{(m+m')}}\left(\frac{r^4}{\beta^4}+(1+m)(1+m')\right)e^{-\frac{r^2}{\beta^2}}dr \tag{6.12}$$

while noting that $F^{int}_{m\neq m'}(r;\beta) \leq E^{int}_{m\neq m'}(r;\beta)$ on the interval $r \in [0, \infty)$ with f^{int} representing the integrand of integral f. Thus convergence of the integral in equation 6.12 implies that the integral in equation 6.5 is *finite* (for the $m \neq m'$ case).

Expansion of the integrand in equation 6.12 yields two improper integrals:

$$\begin{aligned} E_{m\neq m'}(r;\beta) &= \int_0^\infty \frac{r^{(m+m'+1)}}{\beta^{(m+m')}}\left(\frac{r^4}{\beta^4}\right)e^{-\frac{r^2}{\beta^2}}dr \\ &+ \int_0^\infty \frac{r^{(m+m'+1)}}{\beta^{(m+m')}}(1+m)(1+m')e^{-\frac{r^2}{\beta^2}}dr \\ &= c_1 \underbrace{\int_0^\infty r^{(m+m'+1)}r^4 e^{-\frac{r^2}{\beta^2}}dr}_{E^1_{m\neq m'}(r;\beta)} + c_2 \underbrace{\int_0^\infty r^{(m+m'+1)}e^{-\frac{r^2}{\beta^2}}dr}_{E^2_{m\neq m'}(r;\beta)} \end{aligned} \tag{6.13}$$

where $c_1 = \beta^{-(4+m+m')}$ and $c_2 = \beta^{-(m+m')}(1+m)(1+m')$ are constants.

Now, convergence of improper integrals E^1 and E^2 in equation 6.13 implies convergence of the integral in equation 6.12. Beginning with integral E^1 and defining $\alpha \equiv m+m'+1 \geq 1$:

$$\begin{aligned} E^1_{m\neq m'}(r;\beta) &= \int_0^\infty r^\alpha r^4 e^{-\frac{r^2}{\beta^2}}dr \\ &= \int_0^\infty \left(r^2\right)^{\alpha/2}\left(r^2\right)^2 e^{-\frac{r^2}{\beta^2}}dr \\ &= \int_0^\infty \left(r^2\right)^{2+\alpha/2} e^{-\frac{r^2}{\beta^2}}dr \end{aligned} \tag{6.14}$$

A change of variables is now introduced, where:

$$t = \frac{r^2}{\beta^2} \rightarrow r = \beta\sqrt{t} \tag{6.15}$$

$$dt = \frac{2rdr}{\beta^2} \rightarrow dr = \frac{\beta dt}{2\sqrt{t}} \tag{6.16}$$

while noting that $\{r, t\} \geq 0$ as per the bounds of integration which allows for consideration of strictly positive square roots.

Implementing this change of variables with some minor algebraic manipulations yields:

$$\begin{aligned} E^1_{m\neq m'}(r;\beta) &= \int_0^\infty \left(t\beta^2\right)^{2+\alpha/2} e^{-t} \frac{\beta dt}{2\sqrt{t}} \\ &= \frac{\beta^{5+\alpha}}{2} \int_0^\infty t^{[2+(\alpha-1)/2]} e^{-t} dt \\ &= \frac{\beta^{5+\alpha}}{2} \int_0^\infty t^{[3+(\alpha-1)/2]-1} e^{-t} dt \\ &= \frac{\beta^{5+\alpha}}{2} \underbrace{\int_0^\infty t^{a-1} e^{-t} dt}_{\Gamma(a)} \end{aligned} \tag{6.17}$$

where $a \equiv 3 + (\alpha - 1)/2 \geq 3$ and $\Gamma(a)$ is the familiar *Gamma Function* [15].

Repeating this process for integral E^2 in equation 6.13 yields the same result:

$$\begin{aligned} E^2_{m\neq m'}(r;\beta) &= \int_0^\infty r^\alpha e^{-\frac{r^2}{\beta^2}} dr \\ &= \int_0^\infty (r^2)^{\alpha/2} e^{-\frac{r^2}{\beta^2}} dr \\ &= \int_0^\infty (t\beta^2)^{\alpha/2} e^{-t} \frac{\beta dt}{2\sqrt{t}} \\ &= \frac{\beta^{\alpha+1}}{2} \int_0^\infty t^{(\alpha-1)/2} e^{-t} dt \\ &= \frac{\beta^{\alpha+1}}{2} \int_0^\infty t^{[1+(\alpha-1)/2]-1} e^{-t} dt \\ &= \frac{\beta^{\alpha+1}}{2} \underbrace{\int_0^\infty t^{a-1} e^{-t} dt}_{\Gamma(a)} \end{aligned} \tag{6.18}$$

where $a \equiv 1 + (\alpha - 1)/2 \geq 1$ and $\Gamma(a)$ is (again) the familiar *Gamma Function* [15].

It is well-known that the *Gamma Function*, $\Gamma(a)$, converges for $\Re\{a\} > 0$ [15]. This inequality holds for both *Gamma Functions* previously derived, as $a \geq 3$ in equation 6.17 and $a \geq 1$ in equation 6.18 and therefore the integral in equation 6.12 converges absolutely on $r \in [0, \infty)$. By the comparison test, the primary integral (equation 6.5) also converges absolutely on $r \in [0, \infty)$ demonstrating that it is *finite* for the $m \neq m'$ case. An identical analysis can be performed for the integral in equation 6.5 for the $m = m'$ case, but this integral is analytically solved in the next section, implicitly demonstrating that it is *finite*.

Orthogonality of higher-order ($n = 1$) shapelet functions has now been demonstrated, as

$$\left(\frac{c_{1m}c_{1m'}}{\beta^2}\right) G_{m\neq m'}(\theta)\ F_{m\neq m'}(r;\beta) \rightarrow \left(\frac{c_{1m}c_{1m'}}{\beta^2}\right)(0)(finite) = 0 \tag{6.19}$$

and

$$\left(\frac{c_{1m}^2}{\beta^2}\right) G_{m=m'}(\theta)\ F_{m=m'}(r;\beta) \rightarrow \left(\frac{c_{1m}^2}{\beta^2}\right)(2\pi)(finite) \rightarrow finite \tag{6.20}$$

6.2.2 Solving Normalization Constant

To derive orthonormal higher-order ($n = 1$) shapelet functions (and complete demonstration of orthogonality), the integral in equation 6.5 must be solved for the $m = m'$ case:

$$F_{m=m'}(r;\beta) = \beta^{-2m} \underbrace{\int_0^\infty r^{2m+1} \left(1 + m - \left(\frac{r}{\beta}\right)^2\right)^2 e^{-\frac{r^2}{\beta^2}} dr}_{I} \tag{6.21}$$

Numerous intermediate steps to solve integral I in equation 6.21 can be found in Appendix B.2. However, the solution is presented here for brevity:

$$I = \frac{\beta^{2m+2} m!(m+1)}{2} \tag{6.22}$$

and therefore equation 6.21 becomes

$$F_{m=m'}(r;\beta) = \frac{\beta^2 m!(m+1)}{2} \tag{6.23}$$

Returning to equation 6.3 and setting $m = m'$, the normalization constant c_{1m} can be solved:

$$\begin{aligned}
\left(\frac{c_{1m}^2}{\beta^2}\right) G_{m=m'}(\theta) F_{m=m'}(r;\beta) &= \delta_{m=m'} \\
\left(\frac{c_{1m}^2}{\beta^2}\right) (2\pi) \left(\frac{\beta^2 m!(m+1)}{2}\right) &= 1 \\
c_{1m}^2(\pi m!(m+1)) &= 1 \\
\rightarrow c_{1m} &= \frac{1}{\sqrt{\pi m!(m+1)}}
\end{aligned} \tag{6.24}$$

The orthonormal definition of higher-order ($n = 1$) shapelets can now be expressed as:

$$S_{1m}(r,\theta;\beta) = \frac{1}{\beta\sqrt{\pi m!(m+1)}} \left(\frac{r}{\beta}\right)^m \left(1 + m - \left(\frac{r}{\beta}\right)^2\right) e^{-\frac{r^2}{2\beta^2}} e^{-im\theta} \tag{6.25}$$

Additionally, `sympy` [38] (the primary symbolic math library for Python) was used to solve integral I in equation 6.21 for the $m = m'$ case to confirm the analytical solution obtained here.

6.3 Optimal Shapelet Length-Scales

With an appropriate definition of orthonormal higher-order ($n = 1$) shapelets (equation 6.25), the next step is to relate the shapelet length-scale (β) to the characteristic wavelength (λ) of pattern domains.

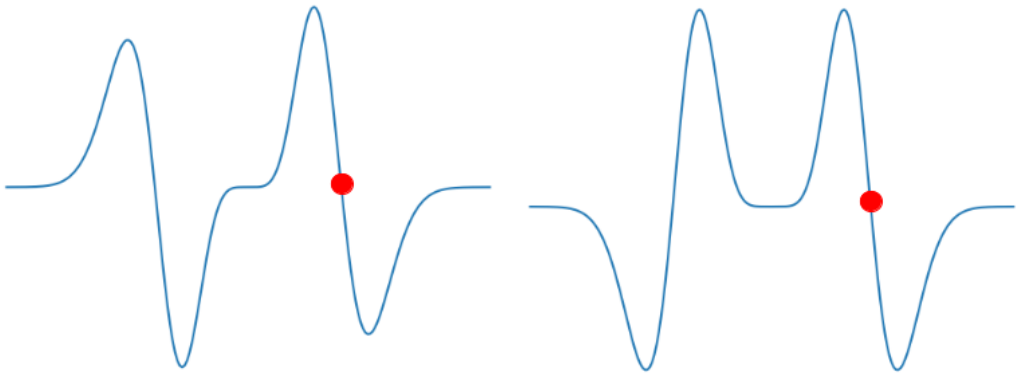
6.3.1 Brute-Force Numerical Approach

As mentioned previously, one potential use of higher-order ($n = 1$) shapelets is to conduct a less-local (or regional) analysis of pattern structure. To fulfill this potential, it would be ideal for a shapelet kernel, centered on a given feature, to physically align its i^{th} radial layer with the i^{th} layer of nearest neighbours. Let β_{opt} be defined as the shapelet length-scale that satisfies this physical constraint.

An analytical approach to determine β_{opt} (similar to that done in ref. [4] for $n = 0$ shapelets) was attempted but this approach was unsuccessful. Furthermore, other attempts

were made at automating the numerical computation of β_{opt} for arbitrary characteristic wavelength scales (λ) but the relationship between $n = 1$ shapelets and the convolution created great difficulty. Unlike past work for $n = 0$ shapelets [49], the desired β_{opt} scale for $n = 1$ shapelets does not correlate to the maximum (or minimum) of the convolution between shapelet kernels and the target image. This is believed to be caused by the adversarial nature of the inner and outer shapelet layers. For example, the maximum convolution may be achieved via the following kernel conformation: the shapelet kernel has its inner layer completely embedded within the primary pattern feature (i.e. where the kernel is centered upon) in order to achieve a net zero convolution, while the outer kernel layer is correctly aligned with the *first* layer of nearest neighbours. As a result, an optimized brute-force numerical approach was developed where a *search* for β_{opt} is performed until satisfaction of some well-defined constraints.

Recall that λ, the characteristic wavelength of a pattern [49], is roughly the distance (in pixels) between neighbouring pattern features. The brute-force approach aims to force the midpoint between the two (positive) radial extrema of the shapelet kernel (i.e. the red dots in Figure 6.1) to be located at a distance 1.5λ from the kernel center (i.e. feature center). By doing this, the two radial extrema of the shapelet will properly align with the first and second nearest neighbours of the feature it is centered upon, and β_{opt} is the shapelet length-scale that achieves this behaviour.



(a) (b)

Figure 6.1: Midline (or center-line) profile for an $n = 1$ shapelet function, which is also representative of the 1-dimensional $n = 1$ shapelet function: (a) $m = odd$ and (b) $m = even$ profiles. The red dot represents the region in between the two positive radial extrema, also known in the text as $S_{midpoint}$.

Some terms necessary to understanding the brute-force approach are as follows:

- Let S represent an $n = 1$ shapelet kernel that will be convolved with a target image.
- Let S_{center} represent the x-coordinate of the center of the shapelet kernel and allow S to be centered upon some pattern feature i.
- Let $S_{midpoint}$ represent the x-coordinate referring to the midpoint of the two positive radial extrema of S (i.e. the red dots in Figure 6.1).

The goal of the brute-force approach is to then satisfy the following relation:

$$S_{midpoint} - S_{center} = 1.5\lambda \tag{6.26}$$

while noting that in practice, this relation will be satisfied to some pixel-distance tolerance (as typical with numerical methods).

An algorithm that achieves this relation, with adaptive stepping to accelerate the rate of convergence, is as follows:

1. Compute the target distance $t_d = 1.5\lambda$ in pixels.
2. Begin with $\beta = 1$.
3. Create a $(n = 1)$ shapelet kernel using the current β length-scale.
4. Compute the (pixel) distance between the midpoint and center of the shapelet kernel, $k_d = S_{midpoint} - S_{center}$
5. Compute the (pixel) distance error, $d_{err} = k_d - t_d$, representing the pixel-gap to be minimized.
6. If $|d_{err}| \leq 1$, $\beta_{opt} = \beta$. Stop iteration.
7. If $d_{err} > 0$, go to step 8, else go to step 10.
8. If $d_{err} < 2$, set $\beta = \beta + 0.1$ then go to step 3, else go to step 9.
9. Set $\beta = \beta + 3(0.1)d_{err}$, then go to step 3.
10. If $d_{err} < 0$, then $\beta = \beta - (0.1)d_{err}$ and go to step 3.

Using the algorithm developed here, β_{opt} can be found for any $(n = 1)$ shapelet kernel. An example demonstrating β_{opt} for $n = 1$ and $m = [1, 6]$ shapelets is shown in Figure 6.2.

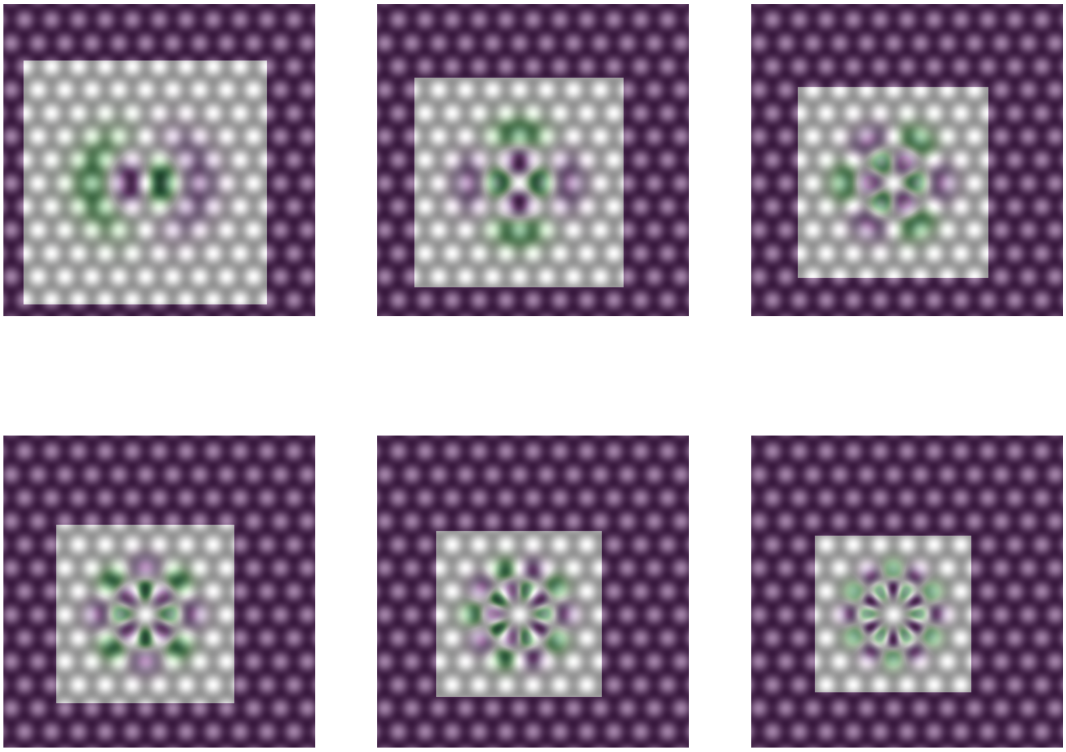


Figure 6.2: Higher-order ($n = 1$) shapelets with optimal length-scale, β_{opt}, superimposed on an ideal hexagonal domain: reading left-to-right, the top and bottom row contain $m = [1, 3]$ and $m = [4, 6]$ higher-order $n = 1$ shapelet kernels respectively.

6.3.2 Difficulties with Increasing m

As m increases, there arises some intrinsic shapelet behaviour that does not allow for the i^{th} layers of the shapelet kernel to physically align with the i^{th} nearest neighbours of the centered feature. Specifically, as m increases beyond six (6), the distance between the two positive radial extrema of the shapelet kernel decreases, causing collapse of the kernel with respect to the center feature's nearest neighbours. Figure 6.3 shows this phenomenon for the $(n, m = 1, 10)$ shapelet kernel.

With this issue, along with the complicated relationship between higher-order ($n = 1$) shapelets and the convolution operation, implementation of these shapelets for pattern analysis is difficult. This task is left for future work.

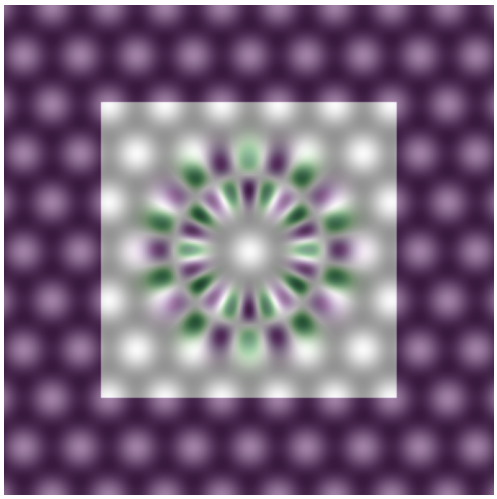


Figure 6.3: Collapse of $n = 1$ shapelet kernel as m increases and displaying this behaviour for $m = 10$. While the inner shapelet layer is well-aligned, the outer layer begins to recede (collapse) into the shapelet center, causing incorrect overall alignment between the radial layers of the shapelet kernel and the surrounding nearest neighbour layers of the center feature.

Chapter 7

Conclusions

7.1 Conclusions

In this work, significant contributions were made to shapelet-based pattern analysis, including enhancements to the existing response distance method and the development of new techniques to characterize nanostructured surfaces from microscopy imaging. These enhancements and new techniques were validated using a representative set of self-assembled nanostructure domains, including those from simulation (Figure 1.1) and experiment (Figure 1.2) containing a variety of measurement noise and uncertainty. Application of these methods to this representative image set has demonstrated their effectiveness in implementation, with specific conclusions as follows:

1. Inclusion of higher-order shapelets ($m \geq 7$) in the response distance method improved the accuracy in identifying localized spatial deformities (i.e. defects, disorder) against uniform (i.e. well-ordered) regions without affecting computational runtime.

2. Careful manipulation of existing shapelet-based orientational information, provided by steerable filter theory, via standard image processing techniques (masking, dilation, blending) allowed for quantification of local pattern orientation and identification of grain boundary defect structures.

3. Development of a new shapelet-based technique, the defect identification method (or defect response distance method), inspired by the existing response distance method, provides direct identification of relevant topological defects (disclinations,

dislocations) and defect structures (grain boundaries) encountered in self-assembled domains.

4. Computational improvements to the response distance method, achieved via implementation in C++, yielded a speed-up of 14-16x (on average) compared to the existing Python implementation. This provides a significant benefit to computational runtime, especially for low-resource machines in the presence of high-resolution microscopy images.

In general, the enhanced and newly developed methods are applicable to arbitary pattern types (stripe, square, hexagonal) representative of the symmetries found in self-assembled domains and thus support the development of structure-property relations for many classes of materials. Moreover, these methods have been made available to the research community in the form of an open-source Python library. This library provides a custom user interface, via configuration files, developed specifically for non-technical users (i.e. those without Python knowledge). This open-source framework should promote collaboration in shapelet-based research, both for existing and future shapelet-related applications.

7.2 Recommendations

1. **Defect Classification** – Existing methods (response distance method) and those developed in this work (defect identification method) are successful at *identifying* topological defects, but *classification* of these defects into specific groups/types has not been performed. Defect classification is of considerable interest to materials researchers, as specific defects and their location within ordered domains may be of interest to develop structure-property relations to advance defect engineering.

2. **Automation** – All current shapelet-based pattern analysis techniques, both existing and those developed in this work, contain an inherent level of supervision that requires some form of user input/knowledge. Given the availability and feasibility of modern machine learning techniques, efforts should be given to automate shapelet-based pattern analysis methods to yield *unsupervised* variations; increasing the usability of these methods overall.

3. **Higher-order n shapelets** – To date, all shapelet-based pattern analysis uses $n = 0$ functions. The mathematical framework for $n = 1$ shapelets was provided in Chapter

6, but the utility of this class of shapelet functions to characterize nanostructured domains is unknown. Future research should incorporate $n = 1$ shapelets, either within existing shapelet-based methods or perhaps as the basis for new techniques, to conduct less-local (i.e. regional) analysis of pattern structure.

4. **Three-dimensional (3D) shapelets** – Current and previous shapelet-based pattern analysis focused on 2-dimensional analysis of nanostructure. It is possible to extend the current formulism to three dimensions for analysis of *tomography* images of micro/nanostructure. This could provide insight into 3-dimensional material structure to derive structure-property relationships. Moreover, different types of topological defects that require 3-dimensional information, such as *screw* dislocations, could be identified with this shapelet analysis.

References

[1] Namra Abid, Aqib Muhammad Khan, Sara Shujait, Kainat Chaudhary, Muhammad Ikram, Muhammad Imran, Junaid Haider, Maaz Khan, Qasim Khan, and Muhammad Maqbool. Synthesis of nanomaterials using various top-down and bottom-up approaches, influencing factors, advantages, and disadvantages: A review. *Advances in Colloid and Interface Science*, 300:102597, 2022.

[2] Nasser Mohieddin Abukhdeir. Computational characterization of ordered nanostructured surfaces. *Materials Research Express*, 3(8):082001, 2016.

[3] Nasser Mohieddin Abukhdeir, Dionisios G Vlachos, Markos Katsoulakis, and Michael Plexousakis. Long-time integration methods for mesoscopic models of pattern-forming systems. *Journal of Computational Physics*, 230(14):5704–5715, 2011.

[4] Thomas J Akdeniz, Daniel J Lizotte, and Nasser Mohieddin Abukhdeir. A generalized shapelet-based method for analysis of nanostructured surface imaging. *Nanotechnology*, 30(7):075703, 2018.

[5] Eberechukwu Victoria Amadi, Anusha Venkataraman, and Chris Papadopoulos. Nanoscale self-assembly: concepts, applications and challenges. *Nanotechnology*, 33(13):132001, 2022.

[6] Bishnu Prasad Bastakoti, Rahul R Salunkhe, Jinhua Ye, and Yusuke Yamauchi. Direct synthesis of a mesoporous tio 2–ruo 2 composite through evaporation-induced polymeric micelle assembly. *Physical Chemistry Chemical Physics*, 16(22):10425–10428, 2014.

[7] Joel Bergé, Richard Massey, Quentin Baghi, and Pierre Touboul. Exponential shapelets: basis functions for data analysis of isolated features. *Monthly Notices of the Royal Astronomical Society*, 486(1):544–559, 2019.

[8] Simon Birrer, Adam Amara, and Alexandre Refregier. Gravitational lens modeling with basis sets. *The Astrophysical Journal*, 813(2):102, 2015.

[9] Eberhard Bodenschatz, Ronald Imbihl, and Ingo Rehberg. Focus on pattern formation. *New Journal of Physics*, 5(1):003, 2003.

[10] Edward Bormashenko, Mark Frenkel, Alla Vilk, Irina Legchenkova, Alexander A Fedorets, Nurken E Aktaev, Leonid A Dombrovsky, and Michael Nosonovsky. Characterization of self-assembled 2d patterns with voronoi entropy. *Entropy*, 20(12):956, 2018.

[11] William L Briggs and Van Emden Henson. *The DFT: an owner's manual for the discrete Fourier transform*. SIAM, 1995.

[12] Joel D. Brock. Bond-orientational order. In Katherine J. Strandburg, editor, *Bond-Orientational Order in Condensed Matter Systems*, pages 1–31. Springer-Verlag, 1992.

[13] Paul M Chaikin, Tom C Lubensky, and Thomas A Witten. *Principles of condensed matter physics*, volume 10. Cambridge university press Cambridge, 1995.

[14] MC Cross, D Meiron, and Yuhai Tu. Chaotic domains: A numerical investigation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 4(4):607–619, 1994.

[15] Philip J Davis. Leonhard euler's integral: A historical profile of the gamma function: In memoriam: Milton abramowitz. *The American Mathematical Monthly*, 66(10):849–869, 1959.

[16] Mark De Berg. *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

[17] P.G. de Gennes and J Prost. *The Physics of Liquid Crystals*. Oxford University Press, New York, second edition, 1995.

[18] G Desvignes, RN Caballero, L Lentati, JPW Verbiest, DJ Champion, BW Stappers, GH Janssen, P Lazarus, S Osłowski, S Babak, et al. High-precision timing of 42 millisecond pulsars with the european pulsar timing array. *Monthly Notices of the Royal Astronomical Society*, 458(3):3341–3380, 2016.

[19] Ding-Kuo Ding. *Structures and dynamics of disclinations and inversion walls in nematic polymers*. PhD thesis, Massachusetts Institute of Technology, 1994.

[20] Jianjin Dong, Irene A Goldthorpe, and Nasser Mohieddin Abukhdeir. Automated quantification of one-dimensional nanostructure alignment on surfaces. *Nanotechnology*, 27(23):235701, 2016.

[21] Jun Dong, Xing Zhao, Wei Gao, Qingyan Han, Jianxia Qi, Yongkai Wang, Sandong Guo, and Mengtao Sun. Nanoscale vertical arrays of gold nanorods by self-assembly: physical mechanism and application. *Nanoscale research letters*, 14:1–9, 2019.

[22] Hongyou Fan, Adam Wright, John Gabaldon, Adrian Rodriguez, C Jeffrey Brinker, and Y-B Jiang. Three-dimensionally ordered gold nanocrystal/silica superlattice thin films synthesized via sol–gel self-assembly. *Advanced Functional Materials*, 16(7):891–895, 2006.

[23] William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[24] Gemunu H Gunaratne, Qi Ouyang, and Harry L Swinney. Pattern formation in the presence of symmetries. *Physical Review E*, 50(4):2802, 1994.

[25] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[26] Rebecca B Hoyle. *Pattern formation: an introduction to methods*. Cambridge University Press, 2006.

[27] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

[28] Amir Jangizehi, Friederike Schmid, Pol Besenius, Kurt Kremer, and Sebastian Seiffert. Defects and defect engineering in soft matter. *Soft Matter*, 16(48):10809–10859, 2020.

[29] JC Jiang, XQ Pan, W Tian, CD Theis, and DG Schlom. Abrupt pbtio 3/srtio 3 superlattices grown by reactive molecular beam epitaxy. *Applied Physics Letters*, 74(19):2851–2853, 1999.

[30] Maurice Kleman. Defects in liquid crystals. *Reports on Progress in Physics*, 52(5):555–654, may 1989.

[31] Alex J Krejci, Colin GW Thomas, and James H Dickerson. Statistical assessment of order within systems of nanoparticles: Determining the efficacy of patterned substrates to facilitate ordering within nanoparticle monolayers fabricated through electrophoretic deposition. *Physical Review E*, 87(4):042307, 2013.

[32] Alex J Krejci, Colin GW Thomas, Jyotirmoy Mandal, Isabel Gonzalo-Juan, Weidong He, Ryan L Stillwell, Ju-Hyun Park, Dhiraj Prasai, Vyacheslav Volkov, Kirill I Bolotin, et al. Using voronoi tessellations to assess nanoparticle–nanoparticle interactions and ordering in monolayer films formed through electrophoretic deposition. *The Journal of Physical Chemistry B*, 117(6):1664–1669, 2013.

[33] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

[34] L Lentati, P Alexander, and MP Hobson. Generative pulsar timing analysis. *Monthly Notices of the Royal Astronomical Society*, 447(3):2159–2168, 2015.

[35] Weihua Li and Marcus Müller. Defects in the self-assembly of block copolymers and their relevance for directed self-assembly. *Annual review of chemical and biomolecular engineering*, 6:187–216, 2015.

[36] Richard Massey and Alexandre Refregier. Polar shapelets. *Monthly Notices of the Royal Astronomical Society*, 363(1):197–210, 2005.

[37] Mathew M Maye, Wenxia Zheng, Frank L Leibowitz, Nam K Ly, and Chuan-Jian Zhong. Heating-induced evolution of thiolate-encapsulated gold nanoparticles: a strategy for size and shape manipulations. *Langmuir*, 16(2):490–497, 2000.

[38] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

[39] Stefanos Mourdikoudis, Roger M Pallares, and Nguyen TK Thanh. Characterization techniques for nanoparticles: comparison and complementarity upon studying nanoparticle properties. *Nanoscale*, 10(27):12871–12934, 2018.

[40] Mark A Olson, Ali Coskun, Rafal Klajn, Lei Fang, Sanjeev K Dey, Kevin P Browne, Bartosz A Grzybowski, and J Fraser Stoddart. Assembly of polygonal nanoparticle clusters directed by reversible noncovalent bonding interactions. *Nano letters*, 9(9):3185–3190, 2009.

[41] Miri Park, Christopher Harrison, Paul M Chaikin, Richard A Register, and Douglas H Adamson. Block copolymer lithography: periodic arrays of˜ 1011 holes in 1 square centimeter. *Science*, 276(5317):1401–1404, 1997.

[42] Woon Ik Park, YongJoo Kim, Jae Won Jeong, Kyungho Kim, Jung-Keun Yoo, Yoon Hyung Hur, Jong Min Kim, Edwin L Thomas, Alfredo Alexander-Katz, and Yeon Sik Jung. Host-guest self-assembly in block copolymer blends. *Scientific Reports*, 3(1):3190, 2013.

[43] Ajay Vasudeo Rane, Krishnan Kanny, VK Abitha, and Sabu Thomas. Methods for synthesis of nanoparticles and fabrication of nanocomposites. In *Synthesis of Inorganic Nanomaterials*, pages 121–139. Elsevier, 2018.

[44] Alexandre Refregier. Shapelets—i. a method for image analysis. *Monthly Notices of the Royal Astronomical Society*, 338(1):35–47, 2003.

[45] Tatyana O Sharpee and Jonathan D Victor. Contextual modulation of v1 receptive fields depends on their spatial symmetry. *Journal of computational neuroscience*, 26:203–218, 2009.

[46] Steven W Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Pub., 2002.

[47] Stephen P Stagon and Hanchen Huang. Controllable growth of aluminum nanorods using physical vapor deposition. *Nanoscale Research Letters*, 9:1–5, 2014.

[48] Robert Suderman. Regular pattern detection and analysis using shapelets. Master's thesis, University of Waterloo, 2014.

[49] Robert Suderman, Daniel J Lizotte, and Nasser Mohieddin Abukhdeir. Theory and application of shapelets to the analysis of surface self-assembly imaging. *Physical Review E*, 91(3):033307, 2015.

[50] Chuanbing Tang, Erin M Lennon, Glenn H Fredrickson, Edward J Kramer, and Craig J Hawker. Evolution of block copolymer lithography to highly ordered square arrays. *Science*, 322(5900):429–432, 2008.

[51] Matthew Peres Tino, Abbas Yusuf Abdulaziz, Robert Suderman, Thomas Akdeniz, and Nasser Mohieddin Abukhdeir. Shapelets: A python package implementing shapelet functions and their applications. *Journal of Open Source Software*, 9(95):6058, 2024.

[52] Matthew Peres Tino, Robert Suderman, and Nasser Mohieddin Abukhdeir. Shapelet-based orientation and defect identification method for nanostructured surface imaging. *Nanotechnology*, 35(16):165705, 2024.

[53] Alan M Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.

[54] Jonathan D Victor, Ferenc Mechler, Michael A Repucci, Keith P Purpura, and Tatyana Sharpee. Responses of v1 neurons to two-dimensional hermite functions. *Journal of neurophysiology*, 95(1):379–400, 2006.

[55] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[56] Zhijun Wang, Junjie Li, Yaolin Guo, Sai Tang, and Jincheng Wang. Unique visualization of multiply oriented lattice structures using a continuous wavelet transform. *Computer Physics Communications*, 184(11):2489–2493, 2013.

[57] Jesse Weissman, Tom Hancewicz, and Peter Kaplan. Optical coherence tomography of skin for measurement of epidermal thickness by shapelet-based image analysis. *Optics express*, 12(23):5760–5769, 2004.

[58] Junjie Wu. *Advances in K-means clustering: a data mining thinking*. Springer Science & Business Media, 2012.

[59] Ji Xu, Thomas P Russell, Benjamin M Ocko, and Antonio Checco. Block copolymer self-assembly in chemically patterned squares. *Soft Matter*, 7(8):3915–3919, 2011.

[60] Y Yokojima and Y Shiwa. Hydrodynamic interactions in ordering process of two-dimensional quenched block copolymers. *Physical Review E*, 65(5):056308, 2002.

[61] Sang-Hyun Yun, Seong Il Yoo, Jin Chul Jung, Wang-Cheol Zin, and Byeong-Hyeok Sohn. Highly ordered arrays of nanoparticles in large areas from diblock copolymer micelles in hexagonal self-assembly. *Chemistry of materials*, 18(24):5646–5648, 2006.

Appendices

Appendix A

Shapelet-based orientation and defect identification method for nanostructured surface imaging[1]

[1]This Appendix contains additional images found in the Supplementary Information of recently published work: M.P. Tino, R. Suderman, and N.M. Abukhdeir, "Shapelet-based orientation and defect identification method for nanostructured surface imaging," Nanotechnology, 35(16), 2024.

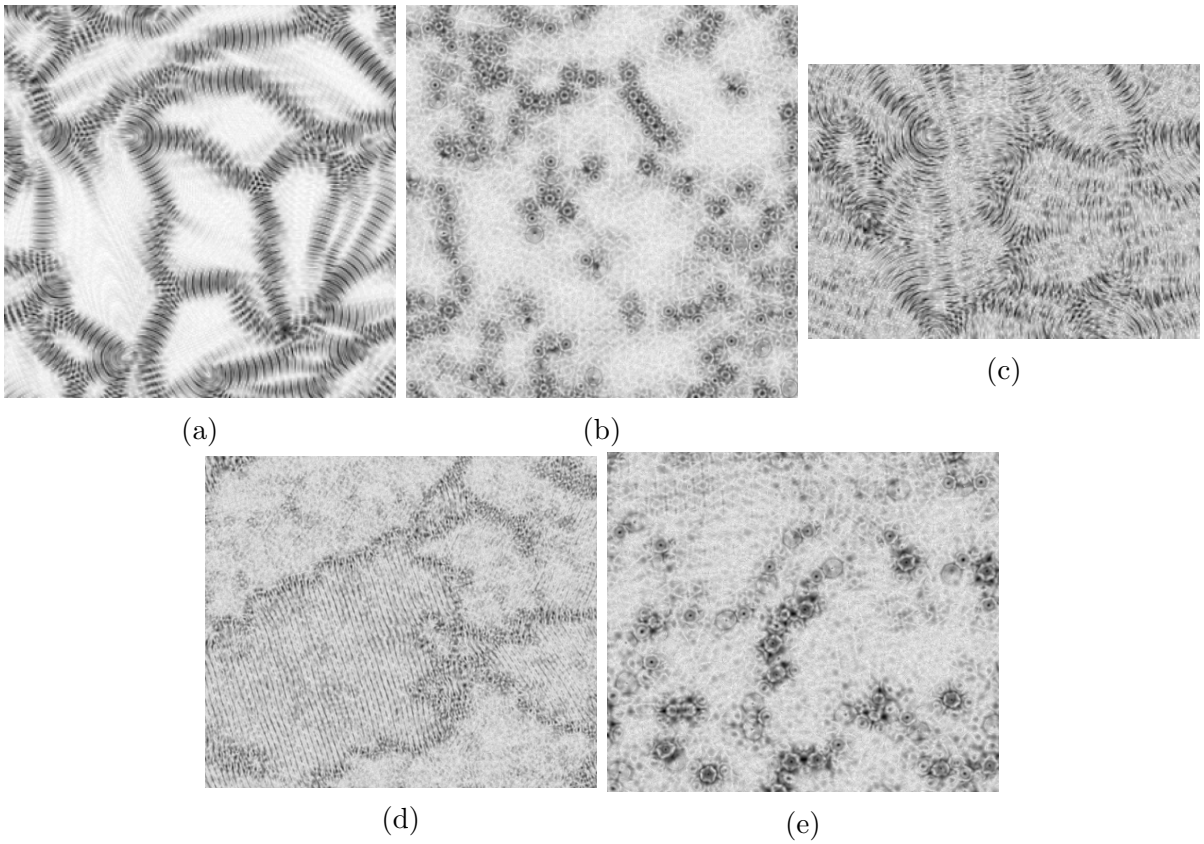


Figure A.1: The response distance scalar field using lower-order ($m \leq 6$) shapelets for the reference self-assembly images in Figures 1.1 and 1.2.

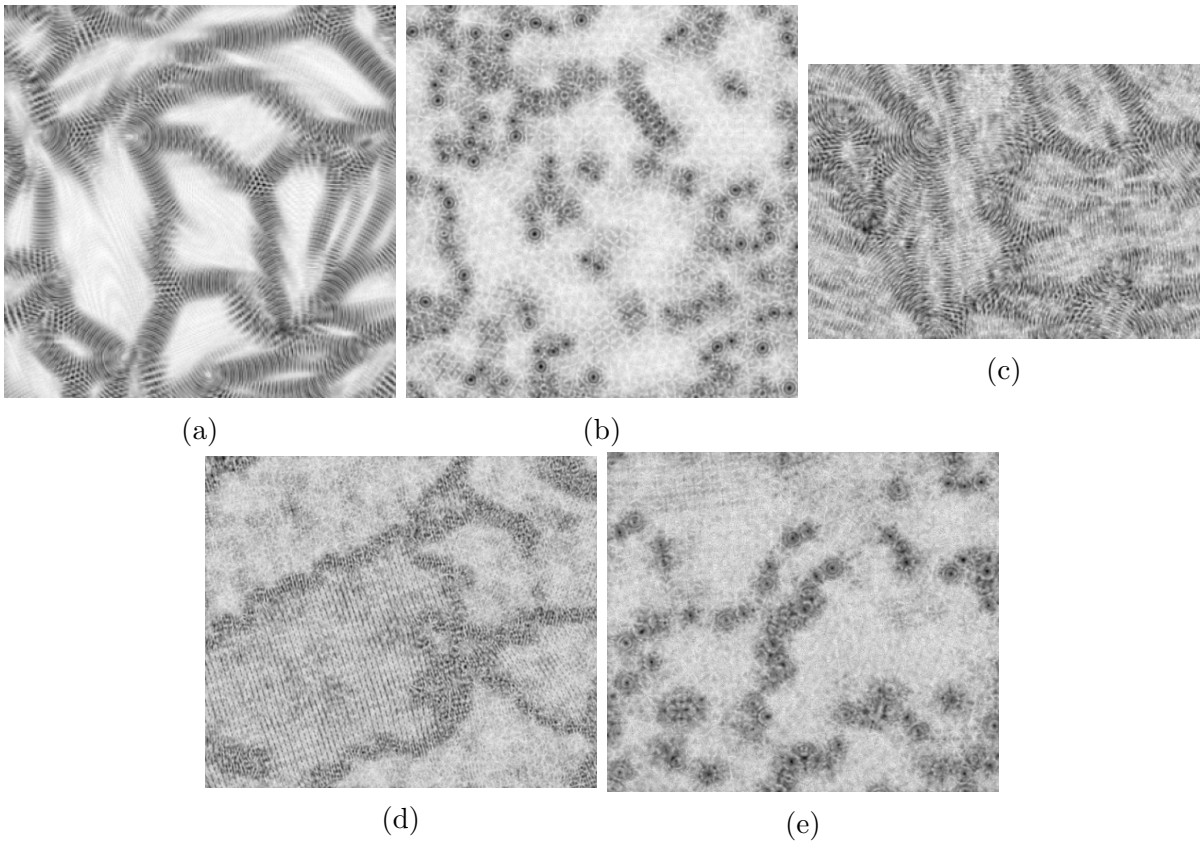


Figure A.2: The response distance scalar field including higher-order ($m \geq 7$) shapelets for the reference self-assembly images in Figures 1.1 and 1.2.

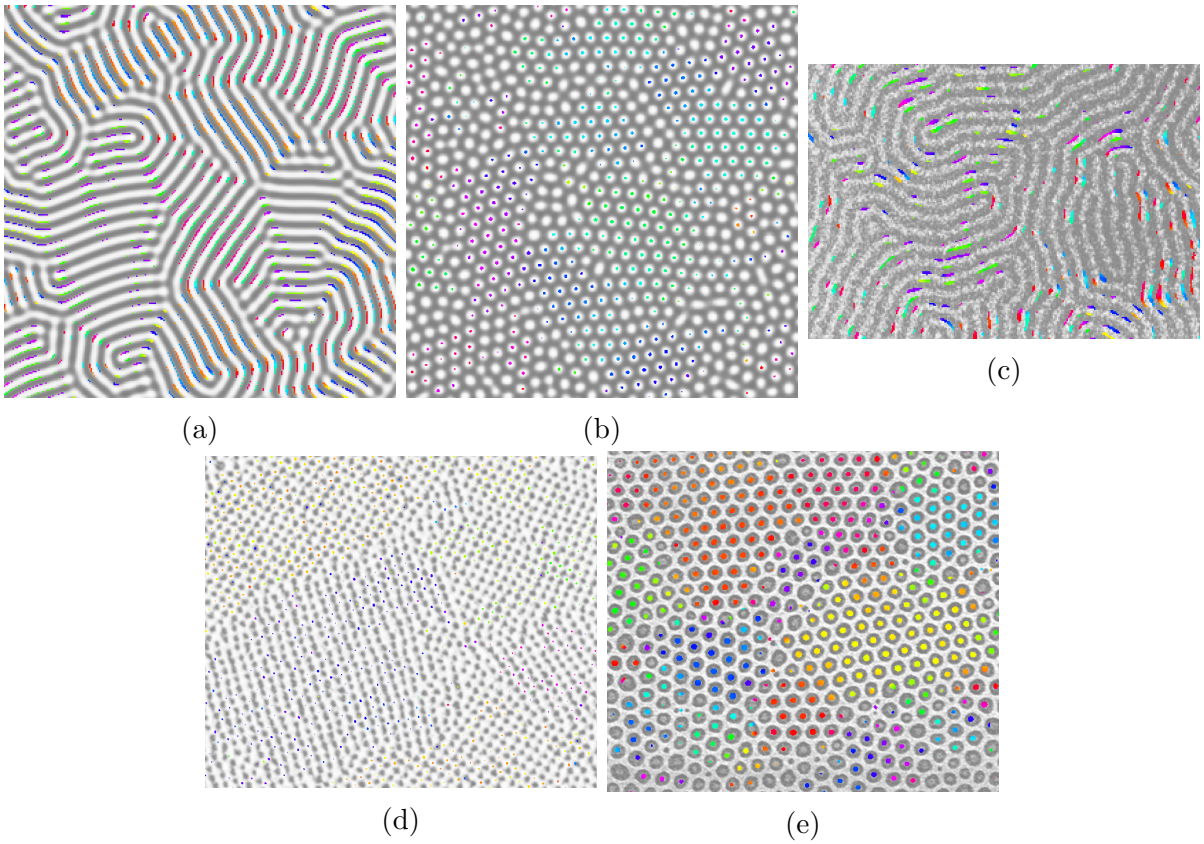


Figure A.3: The orientation masks for the reference self-assembly images in Figures 1.1 and 1.2.

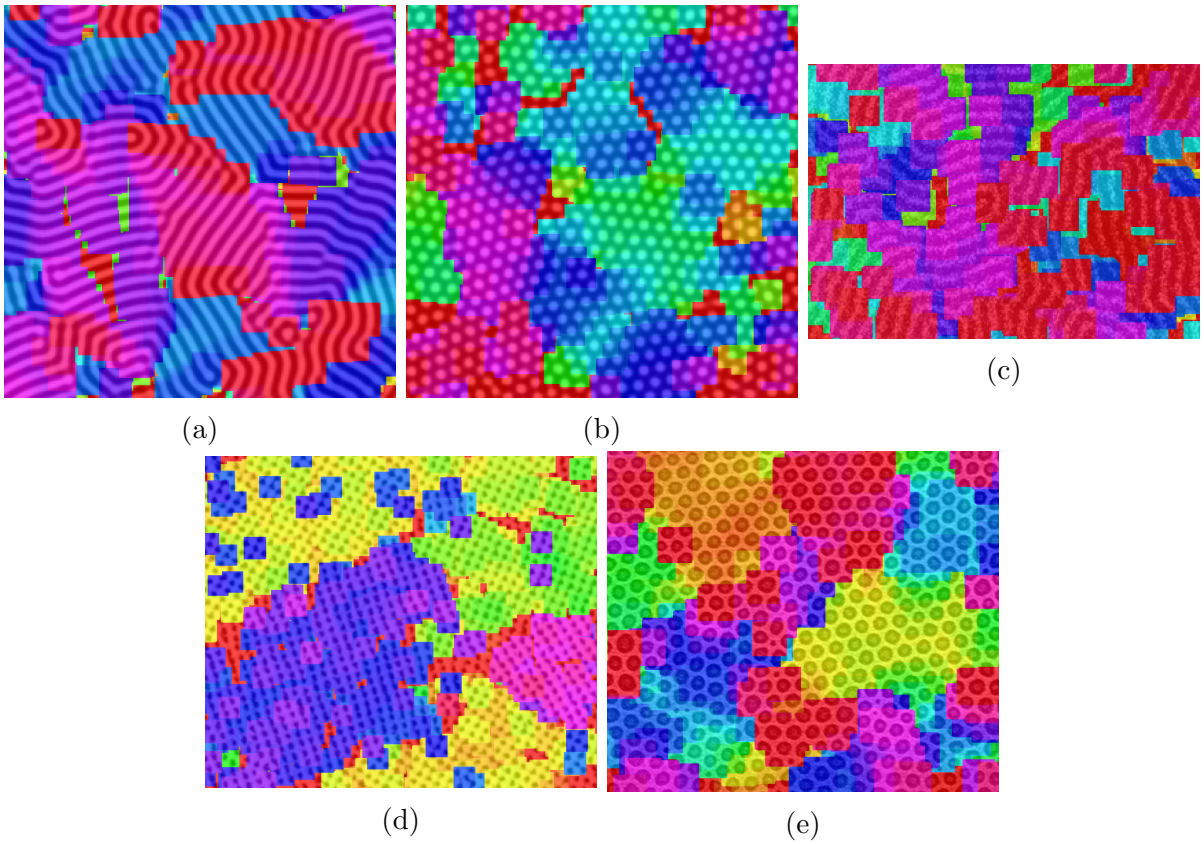


Figure A.4: The result of applying the morphological greyscale dilation operation to the images in Figure A.3.

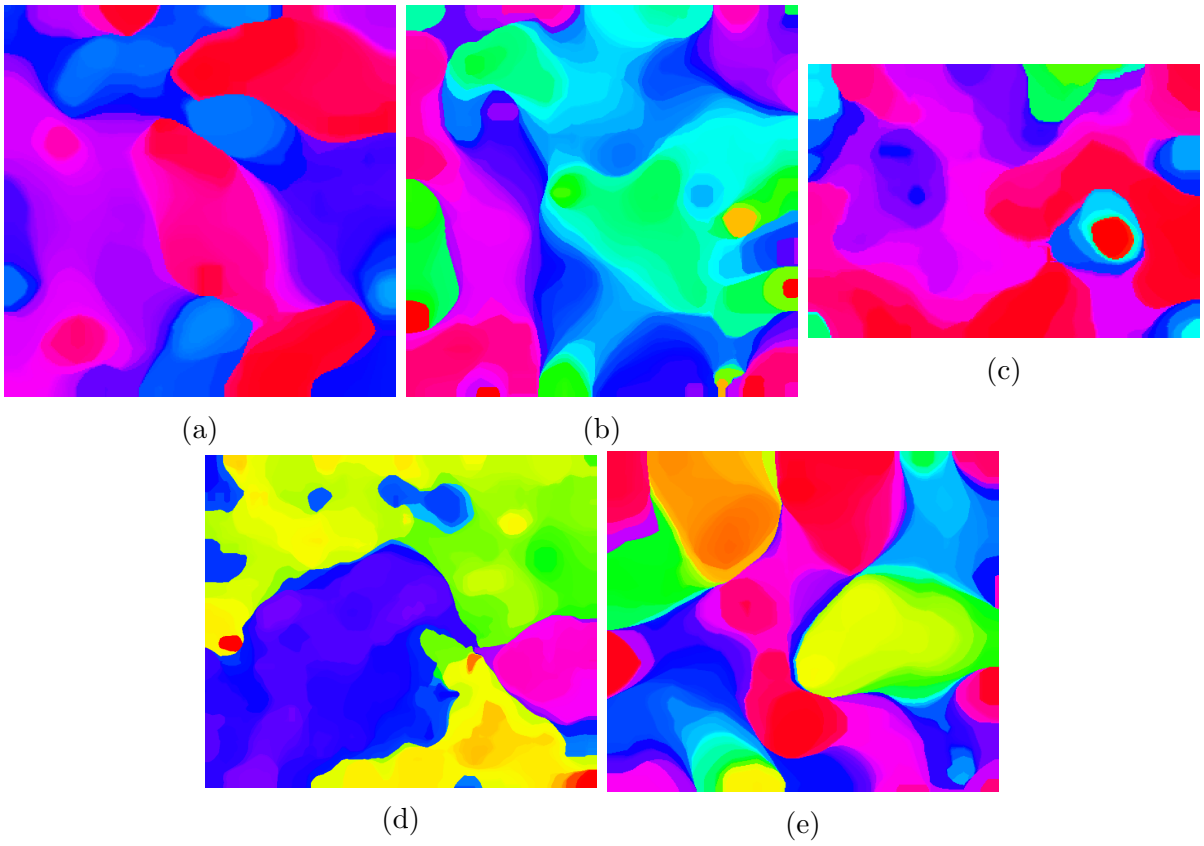


Figure A.5: The result of applying the blending operation via median filter to the images in Figure A.4. Superimposing these results onto their original images can be found in Figure 4.4.

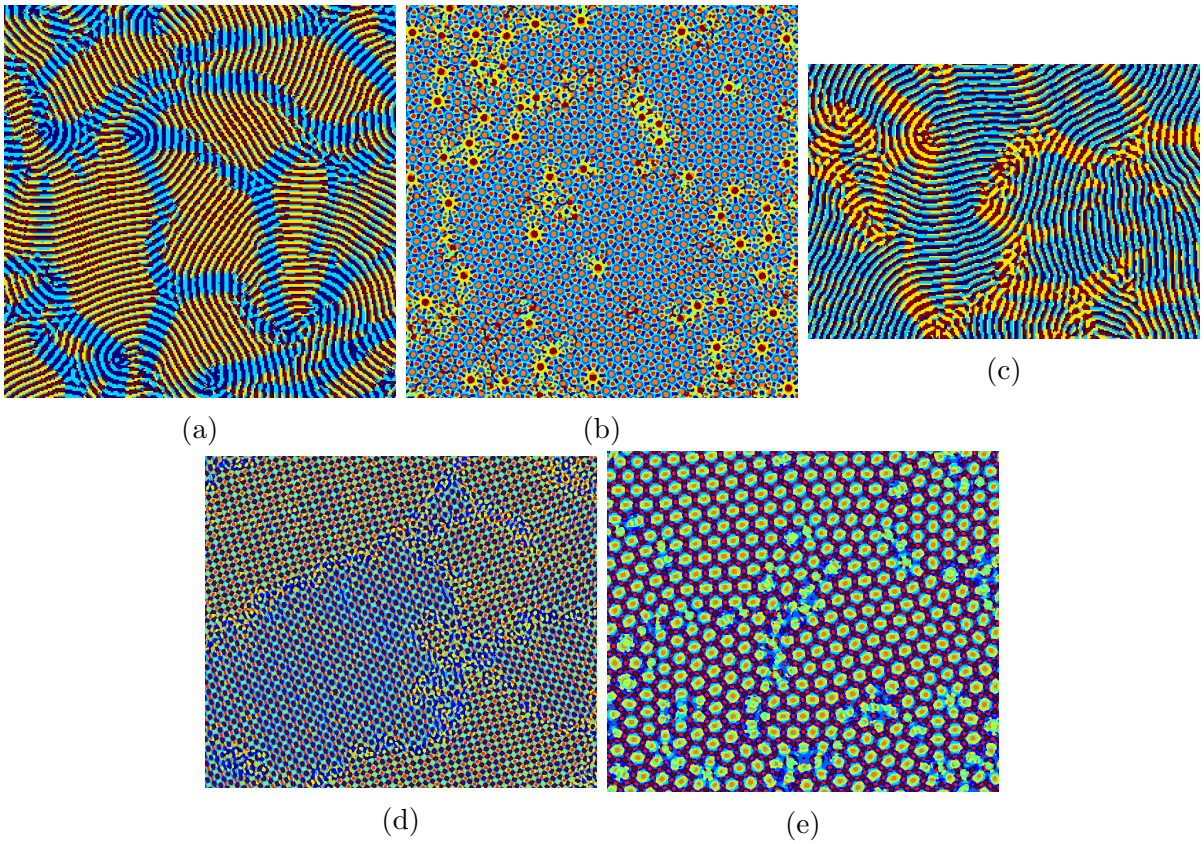


Figure A.6: Visualizing the response cluster associated with each pixel (response vector) for the reference self-assembly images in Figures 1.1 and 1.2.

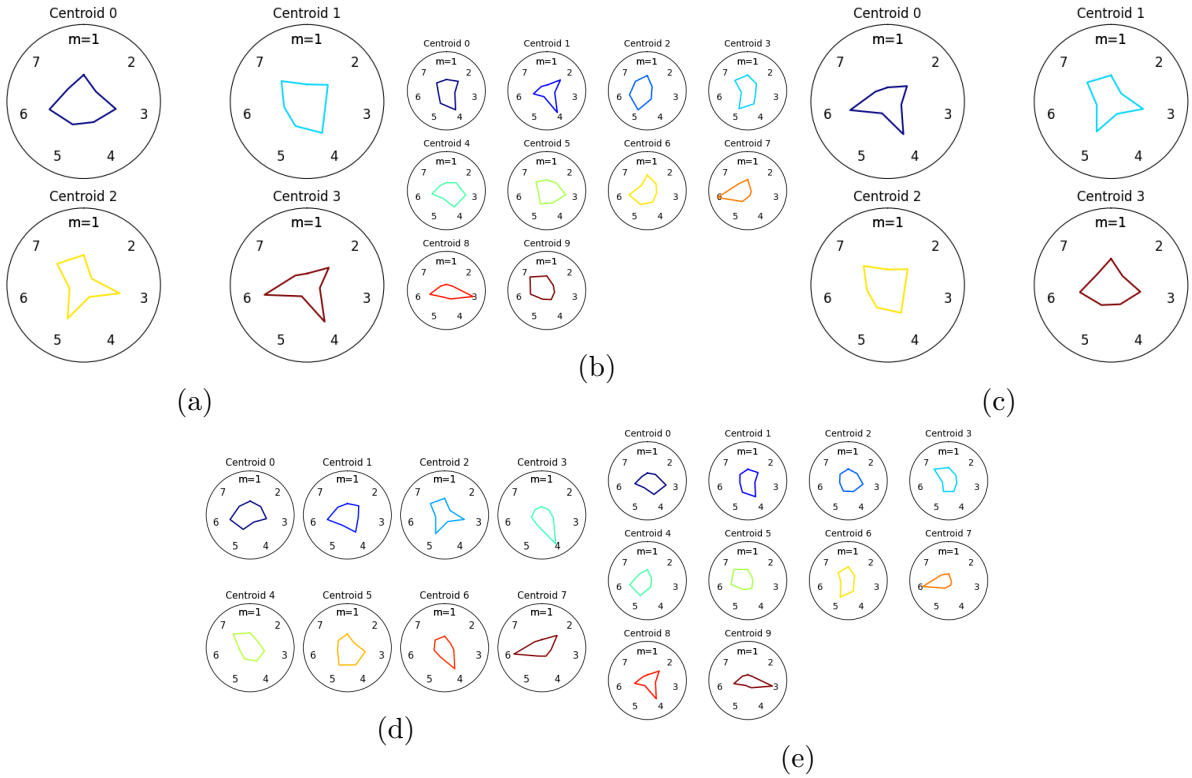


Figure A.7: Radar chart representations of the centroid response vector weights for order $m \leq 7$ shapelets for the images in Figure A.6.

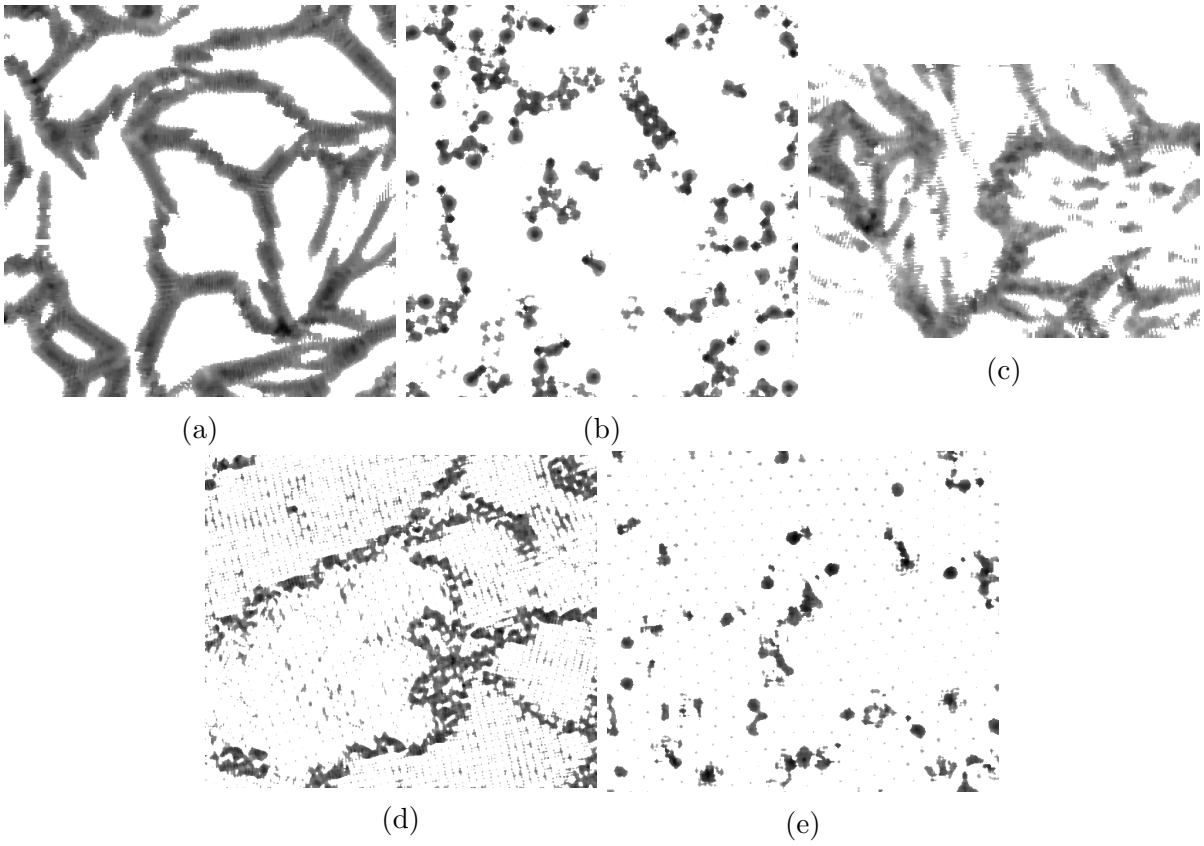


Figure A.8: The defect response distance scalar field (after applying blending operation) for the reference self-assembly images in Figures 1.1 and 1.2. Superimposing these results onto their original images can be found in Figure 4.6.

Appendix B

Mathematical Details of Higher-order (n = 1) Shapelets

B.1 Definition

Polar shapelets [36] were reformulated in ref. [48], and one definition of generic polar shapelets found in ref. [4] is

$$S_{nm}(r,\theta;\beta) = \left(\frac{c_{nm}}{\beta}\right)\phi_{nm}(r/\beta)e^{-im\theta} \tag{B.1}$$

$$\phi_{nm}(x) = x^m L_n^m(x^2)e^{-\frac{x^2}{2}} \tag{B.2}$$

where $\{n, m\}$ are natural indices that correspond to modulation in the radial and polar directions respectively, L_n^m is the generalized (or associated) Laguerre polynomial, c_{nm} is the normalization factor, and β is the shapelet length-scale.

B.1.1 Laguerre Polynomials

The general form of associated Laguerre polynomials via the Rodrigues Formula is [36]:

$$L_n^m(x) = \frac{e^x x^{-m}}{n!}\frac{d^n}{dx^n}(x^{n+m}e^{-x}) \tag{B.3}$$

Now, setting $n = 1$:

$$\begin{aligned} L_1^m(x) &= e^x x^{-m} \frac{d}{dx}(x^{m+1} e^{-x}) \\ &= e^x x^{-m}(-e^{-x} x^{m+1} + (m+1)x^m e^{-x}) \\ &= x^{-m}(-x^{m+1} + (m+1)x^m) \\ &= 1 + m - x \end{aligned} \tag{B.4}$$

With equation B.4, ϕ_{1m} can now be defined as:

$$\phi_{1m}(x) = x^m (1 + m - x^2) e^{-\frac{x^2}{2}} \tag{B.5}$$

Higher-order ($n = 1$) polar shapelets (S_{1m}) can now be formally stated as:

$$S_{1m}(r, \theta; \beta) = \frac{c_{1m}}{\beta} \left(\frac{r}{\beta}\right)^m \left(1 + m - \left(\frac{r}{\beta}\right)^2\right) e^{-\frac{r^2}{2\beta^2}} e^{-im\theta} \tag{B.6}$$

B.2 Orthonormalization

Recall that the integral I from equation 6.21 must be solved to complete the orthonormalization of $n = 1$ shapelets. It is shown here again for clarity:

$$I = \int_0^\infty r^{2m+1} \left(1 + m - \left(\frac{r}{\beta}\right)^2\right)^2 e^{-\frac{r^2}{\beta^2}} dr \tag{B.7}$$

The square term can be expanded as follows:

$$\left(1 + m - \left(\frac{r}{\beta}\right)^2\right)^2 = (1+m)^2 - 2(1+m)\left(\frac{r}{\beta}\right)^2 + \left(\frac{r}{\beta}\right)^4 \tag{B.8}$$

For brevity, let $k = m + 1$, noting that $k \geq 1$ as $m \geq 0$. Continuing to simplify I:

$$\begin{aligned} I &= \int_0^\infty r^{m+k} \left[k^2 - 2k\left(\frac{r}{\beta}\right)^2 + \left(\frac{r}{\beta}\right)^4\right] e^{-\frac{r^2}{\beta^2}} dr \\ &= \int_0^\infty \left[k^2 r^{m+k} - \frac{2k}{\beta^2} r^{m+k+2} + \frac{1}{\beta^4} r^{m+k+4}\right] e^{-\frac{r^2}{\beta^2}} dr \end{aligned} \tag{B.9}$$

Introduce $a = m + k$, noting that $a \geq 1$. Equation B.9 can now be separated into three (3) integrals as follows:

$$\begin{aligned} I &= \int_0^\infty \left[k^2 r^a - \frac{2k}{\beta^2} r^{a+2} + \frac{1}{\beta^4} r^{a+4} \right] e^{-\frac{r^2}{\beta^2}} dr \\ &= \underbrace{\int_0^\infty k^2 r^a e^{-\frac{r^2}{\beta^2}} dr}_{I_1} + \underbrace{\int_0^\infty -\frac{2k}{\beta^2} r^{a+2} e^{-\frac{r^2}{\beta^2}} dr}_{I_2} + \underbrace{\int_0^\infty \frac{1}{\beta^4} r^{a+4} e^{-\frac{r^2}{\beta^2}} dr}_{I_3} \end{aligned} \tag{B.10}$$

B.2.1 Solving the First Integral

$$I_1 = k^2 \underbrace{\int_0^\infty r^a e^{-\frac{r^2}{\beta^2}} dr}_{I_{1a}} \tag{B.11}$$

Note that $k = m + 1$ and

$$a = m + k = 2m + 1 = \{1, 3, 5, 7, ..., odd\} \tag{B.12}$$

since $\{m \in \mathbb{Z} \mid m \geq 0\}$.

The first few terms of I_{1a} from equation B.11 are solved,

$$\begin{aligned} I_{11} &= \int_0^\infty r e^{-\frac{r^2}{\beta^2}} dr = -\frac{\beta^2}{2} e^{-\frac{r^2}{\beta^2}} \Big|_0^\infty = \frac{\beta^2}{2}(1) \\ I_{13} &= \int_0^\infty r^3 e^{-\frac{r^2}{\beta^2}} dr = -\frac{\beta^2}{2} (r^2 + \beta^2) e^{-\frac{r^2}{\beta^2}} \Big|_0^\infty = \frac{\beta^2}{2}(\beta^2) \\ I_{15} &= \int_0^\infty r^5 e^{-\frac{r^2}{\beta^2}} dr = -\frac{\beta^2}{2} (r^4 + 2\beta^2 r^2 + 2\beta^4) e^{-\frac{r^2}{\beta^2}} \Big|_0^\infty = \frac{\beta^2}{2}(2\beta^4) \\ I_{17} &= \int_0^\infty r^7 e^{-\frac{r^2}{\beta^2}} dr = -\frac{\beta^2}{2} (r^6 + 3\beta^2 r^4 + 6\beta^4 r^2 + 6\beta^6) e^{-\frac{r^2}{\beta^2}} \Big|_0^\infty = \frac{\beta^2}{2}(6\beta^6) \end{aligned} \tag{B.13}$$

where an obvious pattern has begun to emerge, namely:

$$\begin{aligned} I_{1a} &= \frac{\beta^2}{2}\beta^{a-1}\left(\frac{a-1}{2}\right)! \\ &= \frac{\beta^2}{2}\beta^{(2m+1)-1}\left(\frac{(2m+1)-1}{2}\right)! \\ &= \frac{\beta^2}{2}\beta^{2m}m! \\ &= \frac{\beta^{2m+2}}{2}m! \end{aligned} \tag{B.14}$$

At last I_1 can be represented in final form:

$$\begin{aligned} I_1 &= k^2 I_{1a} \\ &= k^2\frac{\beta^{2m+2}}{2}m! \\ &= \frac{\beta^{2m+2}m!(m+1)^2}{2} \end{aligned} \tag{B.15}$$

B.2.2 Solving the Second Integral

$$I_2 = \frac{-2k}{\beta^2}\underbrace{\int_0^\infty r^{a+2}e^{-\frac{r^2}{\beta^2}}dr}_{I_{2a}} \tag{B.16}$$

Re-arranging the terms in I_{2a} from equation B.16 yields:

$$\begin{aligned} I_{2a} &= \int_0^\infty r^{a+2}e^{-\frac{r^2}{\beta^2}}dr \\ &= \int_0^\infty r^{a+1}re^{-\frac{r^2}{\beta^2}}dr \end{aligned} \tag{B.17}$$

Using integration by parts (IBP) with

$$u = r^{a+1} \tag{B.18}$$
$$du = (a+1)r^a dr \tag{B.19}$$
$$dv = re^{-\frac{r^2}{\beta^2}} dr \tag{B.20}$$
$$v = -\frac{\beta^2}{2} e^{-\frac{r^2}{\beta^2}} \tag{B.21}$$

and implementing this into equation B.17 yields:

$$\begin{aligned} I_{2a} &= -r^{a+1}\frac{\beta^2}{2}e^{-\frac{r^2}{\beta^2}}\Big|_0^\infty - \int_0^\infty -\frac{\beta^2}{2}(a+1)r^a e^{-\frac{r^2}{\beta^2}} dr \\ &= 0 + \frac{\beta^2}{2}(a+1)\underbrace{\int_0^\infty r^a e^{-\frac{r^2}{\beta^2}} dr}_{I_{1a}} \\ &= \frac{\beta^2}{2}(a+1)\frac{\beta^{2m+2}m!}{2} \\ &= \frac{\beta^{2m+4}m!}{4}(a+1) \end{aligned} \tag{B.22}$$

At last I_2 is solved:

$$\begin{aligned} I_2 &= \frac{-2k}{\beta^2} I_{2a} \\ &= \frac{-2k}{\beta^2}\frac{\beta^{2m+4}m!}{4}(a+1) \\ &= \frac{-(m+1)(2m+2)}{2}\beta^{2m+2}m! \end{aligned} \tag{B.23}$$

B.2.3 Solving the Third Integral

The final portion of the I integral is:

$$I_3 = \frac{1}{\beta^4}\underbrace{\int_0^\infty r^{a+4} e^{-\frac{r^2}{\beta^2}} dr}_{I_{3a}} \tag{B.24}$$

Before solving for I_{3a}, it is important to note that this is an identical form of equation B.17. Generalizing this form, where $i \geq 2$ and i is an even integer yields:

$$\int_0^\infty r^{a+i} e^{-\frac{r^2}{\beta^2}} dr = \left(\frac{\beta^2}{2}\right)^{i/2} \left(\prod_{j=1,3,\ldots}^{i-1} a+j\right) \underbrace{\int_0^\infty r^a e^{-\frac{r^2}{\beta^2}} dr}_{I_{1a}} \tag{B.25}$$

Using this relation, I_{3a} from equation B.24 can be solved with $i = 4$:

$$\begin{aligned} I_{3a} &= \int_0^\infty r^{a+4} e^{-\frac{r^2}{\beta^2}} dr \\ &= \left(\frac{\beta^2}{2}\right)^2 \left(\prod_{j=1,3,\ldots}^{3} a+j\right) \underbrace{\int_0^\infty r^a e^{-\frac{r^2}{\beta^2}} dr}_{I_{1a}} \\ &= \left(\frac{\beta^4}{4}\right)(a+1)(a+3)\frac{\beta^{2m+2} m!}{2} \\ &= (2m+2)(2m+4)\frac{\beta^{2m+6} m!}{8} \end{aligned} \tag{B.26}$$

And finally I_3 is solved as:

$$\begin{aligned} I_3 &= \frac{1}{\beta^4} I_{3a} \\ &= \frac{1}{\beta^4}(2m+2)(2m+4)\frac{\beta^{2m+6} m!}{8} \\ &= (2m+2)(2m+4)\frac{\beta^{2m+2} m!}{8} \end{aligned} \tag{B.27}$$

Finally I can be shown as the sum of its three integral components:

$$
\begin{aligned}
I &= I_1 + I_2 + I_3 \\
&= \frac{\beta^{2m+2} m!(m+1)^2}{2} - \frac{(m+1)(2m+2)}{2}\beta^{2m+2}m! + \frac{(2m+2)(2m+4)\beta^{2m+2}m!}{8} \\
&= \frac{\beta^{2m+2}m!}{8}\left[4(m+1)^2 - 4(m+1)(2m+2) + (2m+2)(2m+4)\right] \\
&= \frac{\beta^{2m+2}m!}{8}\left[4(m+1)^2 - 8(m+1)^2 + 2(m+1)(2m+4)\right] \\
&= \frac{\beta^{2m+2}m!(m+1)}{8}\left[4(m+1) - 8(m+1) + 2(2m+4)\right] \\
&= \frac{\beta^{2m+2}m!(m+1)}{8}\left[4m + 4 - 8m - 8 + 4m + 8\right] \\
&= \frac{\beta^{2m+2}m!(m+1)}{8}(4) \\
&= \frac{\beta^{2m+2}m!(m+1)}{2}
\end{aligned}
\tag{B.28}
$$