Detecting, modeling and rendering complex configurations of curvilinear features

by

Evgueni Parilov

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

January 2009

Approved: _____

Denis Zorin

SOME ARE HERE TO GET TO THEIR JOBS, SOME ARE TRYING TO ENTERTAIN THEMSELVES OR THE OTHERS AROUND, SOME SLEEP, SOME LOUGH... I DO NOT BELONG TO ANY OF THEM, I WRITE MY THESIS WHEN I TAKE A SUBWAY FROM BROOKLYN TO MANHATTAN. BUT SOON ENOUGH, I WILL BECOME AGAIN NORMAL, LIKE EVERYBODY ELSE.

To Sveta and Seva

Acknowledgments

I would like to acknowledge my advisor Denis Zorin for believing in me and encouraging me all these years. Special thanks to him for letting me finish what have I started. From him I learned scientific approach, how can a small finding turn into a big discovery.

I must acknowledge Demetri Terzopoulos. His multidisciplinary research always fascinated me and inspired me to always attack a problem from several different angles. His valuable input during my proposal defense helped me a lot to accomplish my goals in the final thesis.

Ken Perlin, for his invaluable comments and unconventional ideas. He inspires everybody in MRL by his unlimited energy and broad knowledge base. He always has an advice to everybody on any subject.

I am very grateful to Davi Geiger, Yan LeCun, and Chee Yap, who served in my dissertation committee, for their valuable time. I enjoyed so much seeing in their faces how much they appreciate my work, I am thankful and humbled.

Mary Potasek for her subtle guidance and supervision on topics that while did not get to the thesis nevertheless influenced my vision and helped me forming my scientific opinion.

Thanks to various researchers around the globe who helped me in learning basics of geometric random processes to be able to explore more advanced topic of random

fibre processes. Especially, many thanks are to the very valuable input from Marie-Colette van Lieshout, Dietrich Stoyan, and Ilya Molchanov.

I cannot even estimate how much inspiration I have got from talking to my friends and colleagues from Computer Science department and MRL. Especially, Henning Biermann, Lexing Ying, Jeff Han, Harper Langston, George Biros, Emre Mengi, Vikram Sharma, Wei Shao, M. Alex O. Vasilescu, Ilya Rosenberg, Marc'Aurelio Ranzato, Vladimir Savchenko, Marina Spivak, Denis Kovacs, Sung-Hee Lee, Tatyana Kichkaylo, and I thank them all.

Rosemary Amico – the best departmental coordinator ever. I always knew that she never gives up helping students, and can come up with a nontrivial alternative solution to any organizational problem. CS department is lucky to have her on board. Special thanks to Meghan Hartley whose professional and courteous assistance saved enormous amount of time to the students and researchers in the lab.

Special thanks to Bio-Optics Lab at Unilever Research, Edgewater NJ, for sharing knowledge and company during all the summer internships.

My parents, for believing in me all these years since I first expressed my interest in pursuing a higher degree. You made me trust that dreams always come true when you sincerely believe in them.

My wife and son, for their unconditional support all those years on my road to fulfilling my dream to becoming a scientist.

Abstract

Curvilinear features act as a basis in description and representation of a variety of real world patterns spanning from simple regular patterns like honeycomb tiling or text glyphs to very complicated random patterns like networks of furrows on the surface of the human skin, webs of cracks and fissure patterns on dry soil, clay, or old paintings, networks of blood microvessels, and planar maps of linear fault zones. In this work we have developed a set of methods and new data representations for solving key problems related to curvilinear features, which include (1) robust detection of intricate networks of curvilinear features from digital images, (2) GPU-based sharp rendering of fields with curvilinear features, and (3) a parametric synthesis approach to generate systems of curvilinear features with desirable local configurations and global control.

Existing edge-detection and image segmentation techniques for detecting features from digital images may underperform in the presence of inevitable noise, usually do not link the detected edge points into chains, often fail on complex structures and weakly presented curves, heavily depend on initial guess, or/and assume significant manual phase. We have developed a technique based on active contours, or snakes, which avoids laborious manual initial positioning of the snakes, does not require user interaction during optimization, and can detect large networks of curves with complex junctions.

The standard bilinear interpolation of piecewise continuous fields defined on regular grids results in unwanted smoothing along the curvilinear discontinuities, which is common during rendering of surfaces with small detail features like creases, wrinkles, and dents. In many cases, spatially varying features are best represented as a function of the distance to the discontinuity curves and its gradient, like the normal field near the creases along curves. We have presented a real-time, GPU-based method for unsigned distance function field and its gradient field interpolation which preserves discontinuity feature curves. The discontinuities are represented by a set of quadratic Bezier curves, with minimal restriction on their topology.

Detail features are very important visual clues which make computer-generated imagery look less artificial. Sample-based synthesis has become ubiquitous technique in generating arbitrary size textures of a high quality which locally resemble a given small reference texture. However, it shows inferior performance on textures with well structured patterns producing gaps in features or breaking feature coherency. Control on the pattern specification is very limited and often requires exhaustive ad-hoc search for the right parameters, yet not intuitive. We have explored an alternative approach of generating features using random fibre process. Existing mathematical models of random fibres — stochastic processes of networks of curves and lines — model only completely random, stationary, and isotropic curve arrangements in the plane. We have developed a Gibbs-type random process of linear fibres based on local fibre interactions. It allows generating non-stationary curvilinear networks with some degree of regularity, and provides an intuitive set of parameters which directly defines fibre local configurations and global pattern of fibres.

For random systems of linear fibres which approximately form two dominant orientation fields, mutually orthogonal at each point, we have adapted a streamline placement algorithm which converts such systems into overlapping random sets of

coherent smooth curves.

Contents

Dedication **v**

Acknowledgments **vi**

Abstract **viii**

List of Figures **xv**

List of Tables **xxii**

List of Algorithms **xxiii**

Introduction **1**

Motivation 1

Thesis and methodology 5

Contributions 9

Thesis organization 12

1 Curves detection by autonomous snakes **13**

1.1 Introduction 13

1.2 Snakes optimization framework 19

1.3 Interactive active contours 22
1.3.1 Merge-split operations on snakes 25
1.3.2 Three-step optimization algorithm 30
1.3.3 Discussion 31
1.4 Summary 36

2 Real-time rendering of feature curves **37**
2.1 Introduction 37
2.2 Related work 39
2.3 Background: GPU, shaders, framebuffers 45
2.4 Overview of algorithm 49
2.5 Distance field to features 54
2.5.1 Distance functions 55
2.5.2 Calculating distance field to feature curves 57
2.6 Feature discretization: discontinuity configuration and signature . . 60
2.6.1 Curvilinear features: optimizing invalid signatures 66
2.6.2 Rasterization of distance field and its gradient 70
2.7 Rendering of feature maps 71
2.7.1 Side test 74
2.7.2 Interpolation in a curvilinear triangle 80
2.7.3 Code outline for the normal interpolation shader 83
2.8 Interactive rendering of linear features 85
2.9 Implementation and Results 88
2.9.1 Feature curves 90
2.9.2 Linear features 96
2.10 Summary 102

3 Parametric synthesis of patterns with curvilinear features 104

3.1 Introduction 104

3.2 Related work 115

3.3 Background: random arrangements of objects in the plane 121

3.3.1 Framework of Poisson point processes 122

3.3.2 Gibbs point process: inter-point interactions 124

3.3.3 Fibre processes: general models 131

3.3.4 Metropolis-Hastings algorithm for generating random point arrangements 136

3.4 New model of interacting random fibres 143

3.4.1 Adaptation of Gibbs point process models to random fibres; energy conservation requirement 143

3.4.2 Interaction pair-potential between pair of fibres 150

3.4.3 Complete model for random processes of interacting fibres . 154

3.5 Random systems of line segments — linear fibres 157

3.5.1 Interaction model for linear fibres 158

3.5.2 Zero- and first-order potentials 160

3.5.3 Interaction pair-potential for linear fibres 163

3.6 Parametric synthesis of linear fibres systems 168

3.6.1 Detailed balance equation for linear fibres 169

3.6.2 Synthesis algorithm 173

3.6.3 Examples of generated linear fibres systems 180

3.7 User control on synthesis 187

3.7.1 User control models through distance based constraints and weighted models 189

3.7.2 Total length constraints 194

3.7.3 Connectivity constraint 200
3.7.4 Hard-constraints: weighted models to enforce aligning with orientation vector field 206
3.7.5 Soft-constraints 215
3.8 Summary 221

4 A method of generating networks of curves aligned with random systems of linear fibres 226
4.1 Multi-orientation vector fields 228
4.2 Streamlines aligned with cross-orientation vector fields 237
4.3 Streamlines based on context-dependent vector fields 240
4.4 Summary 252

5 Results: synthesis and GPU rendering of a feature curve network 253
5.1 Synthesis of a random network of curves 254
5.2 Real-time rendering of network of curves 258

Conclusion 265
Future work 268

Basic Notation 271

Bibliography 274

List of Figures

1.1 Skin images with visible fine-scale structures. 14
1.2 Images of skin negative replicas 15
1.3 Autonomous snakes after several iterations. 23
1.4 Forming the bundles of snakes. 24
1.5 Static thinning of snake bundles is problematic. 24
1.6 Operation 1. Merging partially aligned snakes. 25
1.7 Operation 2. Splitting partially aligned snakes. 27
1.8 Stage 1: iterations without applying snake merging/splitting. 31
1.9 Stage 2: iterations with snake merging only. 31
1.10 Stage 3: iterations with applying merging/splitting. 32
1.11 Close snapshot: result of thinning of snake bundles. 32
1.12 Resulting furrow network recovered from a skin image. 33
1.13 Image domain cell decomposition. 35

2.1 Typical normal map interpolation artifacts versus our artifact free normal maps. 38
2.2 Our normal map is a blend of (conventional) continuous and (procedural) discontinuous normals. 40
2.3 Comparing a quality of distance function calculations. 43

2.4 Left: graphics pipeline. Right: Texture interpolation under the presence of discontinuity. 47
2.5 Outline of our method to interpolate normal field $\mathbf{N}(u,v)$. 50
2.6 Interpolation between the global smooth normal map $\mathbf{N}(u,v)$ and the discontinuous normal map $\mathbf{n}^h$ defined by local profile $h(d)$. 53
2.7 Left: Distance function singularities within a rectangle. Right: Distance function level lines rendered by using our techniques. 56
2.8 Interpolation domains form a partition of a texel. 57
2.9 Calculation of unsigned distance to the feature lines and curves. . . 58
2.10 Texel valid discontinuity configurations. 62
2.11 Texel prohibited discontinuity configurations. 62
2.12 Rasterization of curvy features. Possible artifacts. 63
2.13 Discontinuity descriptor defining curvy feature. 64
2.14 Discontinuity descriptors for linear features. 65
2.15 Examples of invalid discontinuity signatures. 66
2.16 An initial configuration for spline energy optimization. 67
2.17 Spline energy optimization constraints. 69
2.18 An example of optimal Bezier splines locations. 70
2.19 Localizing a subdomain from the texel partition covering a sample. . 73
2.20 Ambiguous side assignment with respect to an implicit curve. 75
2.21 A sufficient condition for unambiguous side assignment 76
2.22 Resolving the side test ambiguity for the case $x_0 \neq 0$. 77
2.23 Resolving the side test ambiguity for the case $x_0 = 0$. 79
2.24 Interpolation within a regular triangle and a trianble with one curve edge. 81
2.25 Interpolation within a triangle with two curvy edges 82

2.26 Three-step algorithm to interpolate normal $\mathbf{N}_t$. 86
2.27 Interactive linear fibres. 87
2.28 Comparing different interpolation methods. 90
2.29 Applying different user-defined profiles along curvy features. 91
2.30 A coke can with added fingerprint features. 92
2.31 A plate with an Escher pattern of curvy features. 95
2.32 Glass with fractures. 98
2.33 A snake head model with curvilinear features approximated by a set of linear segments. 99
2.34 An example of applying different profiles to a 3D geometry. 100
2.35 Resolving complex topology of several curves meeting at one point. 100
2.36 Sharp rendering of glyphs. 102

3.1 Examples of natural patterns formed by curvilinear features. 106
3.2 Two examples of fibre configurations produced by given interaction functions $h_{\boldsymbol{\theta}}(d,w)$. 112
3.3 Example of a linear fibre system and a phase space $\mathcal{X} = \mathcal{X}_P \times \mathcal{X}_L \times \mathcal{X}_W$. 113
3.4 Our unsuccessful attempt to generate a new target texture with network of curves by applying Heeger and Bergen steerable pyramid texture synthesis. 116
3.5 Our attempt to generate random network of curves by applying feature matching algorithm of Wu and Yu. 119
3.6 Examples of homogeneous Poisson point processes. 123

3.7 Examples of constructing new simple random processes from homogenous Poisson process. Clustering, Neyman-Scott process,hard-core process. 125
3.8 Examples of two Strauss point processes 132
3.9 A simple fibre system $\Phi = \varphi_0 \circ \varphi_1 \circ \ldots \circ \varphi_{10}$. 145
3.10 Alternative representations of a fibre system Φ. 147
3.11 Change of interaction profile between two fibres after partitioning one of them. 148
3.12 Graphical representation of the fibre system interaction integral. . . 150
3.13 Interaction between infinitesimal pieces of two fibres. 150
3.14 Representation of a small fragment of a curvilinear fibre by a 3D point. 152
3.15 Energy conservation for the first-order interaction term $\tilde{h}_1$ and for the second-order interaction term $\tilde{h}_2$. 156
3.16 Schematic proof of conservation of interaction integral $(\varphi)_1 + (\psi)_1 + (\varphi,\psi)_2$ calculated between two fibres φ and ψ. Here: $(\varphi)_1 \equiv \tilde{h}_1(\varphi)$, $(\varphi,\psi)_2 \equiv \tilde{h}_2(\varphi,\psi)$ 157
3.17 Representation of fibre within its phase space. 159
3.18 Two interacting linear fibres φ and ψ. 164
3.19 An example of adding a new fibre ψ to a fibre system $\Phi = \varphi_1 \circ \cdots \circ \varphi_4$. 175
3.20 Neighborhood search query $Nb(\psi;\Phi,R)$ optimized by using a domain partition. 176
3.21 Resolving undesirable configurations of features. 178
3.22 Relaxation scheme for a newly added fibre. 179
3.23 Different types of Poisson fibre processes. 181
3.24 Poisson linear fibre processes with large proximity radii R_Δ. 182
3.25 Homogeneous Poisson fibre process versus inhomogeneous. 184

3.26 Poisson homogeneous linear fibre process versus linear fibre process with observable regularity. 186
3.27 Polar plots of different point interaction functions $h_{\boldsymbol{\theta}}(d,w)$. 188
3.28 Snapshots of fibre systems corresponding to different SA-temperature values $t = 0.075$ and $t = 0.025$. 196
3.29 Different linear fibre systems generated under the same total length constraint, $l_T = 350$. "Cross-pattern" fibre model. 197
3.30 Possible pattern deterioration. 200
3.31 Fibre process with the local connectivity constraint. 203
3.32 Examples of candidates for fibre relaxation. 204
3.33 $point \times point$ and $fibre \times fibre$ relaxation schemes. 205
3.34 An extended $point \times fibre$ relaxation scheme. 207
3.35 Fibre systems generated by applying relaxation schemes. 208
3.36 Original "cross-pattern" fibre process with different alignment weighted models: am2-02, with W1 alignment weight function, am1, W1, amL(3/4), W1. Vector field VF-up was applied. 210
3.37 Different alignment weight functions applied to the weighted model am2-2: W1,W2,W3, and W4. 213
3.38 Examples of fibre systems. 214
3.39 Simulation domain wrapping. 216
3.40 Vector fields: VF-up, VF-diag, VF-rght, VF-trn, VF-rad, VF-circ. . 217
3.41 Applying hard and soft constraints to the (original) unconstrained models. 220
3.42 Examples, model am2-2: different alignment weight functions. . . . 222
3.43 Applying different soft constraint models with the "diagonal" feature: asm2-2, W1, asm2-2, W2, asmL(3/4), W1, and asm1, W1. . . 223

3.44 Applying soft constraint models to a feature which follows a letter “C”.224

4.1 Examples of “fibre flow” singularities. 229
4.2 Examples of MOVFs, consistent (on the left) and inconsistent (on the right) with a given vector field. 230
4.3 Linear fibre systems $\Phi_{(a)}$ and $\Phi_{(b)}$ used for creating fibre-induced tangent vector fields, $\mathsf{V}^{\Phi_{(a)}}$ and $\mathsf{V}^{\Phi_{(b)}}$. 234
4.4 Fragments of optimal COVF (right column) against original fibre-induced tangent vector field $\mathsf{V}^{\Phi_{(a)}}$. 235
4.5 Examples of COVF optimization within two fragments of the fibre-induced tangent vector field $\mathsf{V}^{\Phi_{(b)}}$. 236
4.6 A fragment of a COVF with discontinuities. 238
4.7 A result of running a simple adaptation of a streamline algorithm [86].239
4.8 Context-dependent direction sampling from discrete COVF $\hat{V}^{+}$. . . 242
4.9 Example of curvilinear network generated by our CD-algorithm with context-dependent direction sampling from a COVF. 244
4.10 Weighted context-dependent direction sampling from a discrete COVF.246
4.11 Example of curvilinear network of two coherent sets of curves generated by our streamline placement CD-wCD-Algorithm applied to a COVF. 249
4.12 Example of curvilinear network of two coherent sets of curves generated from the linear fibre system shown in Figure 3.36-c. 250
4.13 Example of curvilinear network of two coherent sets of curves generated from the linear fibre system shown in Figure 3.30-a. 251

5.1 An example of a linear fibre process with “cross-pattern” interaction model. 254

5.2 Network of smooth coherent curves generated from linear fibre system illustrated in Figure 5.1. 256
5.3 Result of post-processing the network of curves illustrated in Figure 5.2. 257
5.4 Rendering the network of features on a plane. 259
5.5 Rendering the network of features on a can with a different embossing profile. 260
5.6 Wiremesh-like appearance of a plate (originally illustrated in Figure 2.31). 261
5.7 Glass wiremesh-like appearance of a plate. 262
5.8 Interactive wire thinning by decreasing the visibility distance d_{SHOW}. 263

List of Tables

2.1 Feature curves performance table. 97

2.2 Feature lines performance table. 101

2.3 Comparison in performance measured for two different generations of NVidia GPUs. 103

3.1 Examples of point process density functions. 129

3.2 Table of Gibbs fibre process notations. 146

3.3 One step of Metropolis-Hastings "Birth-and-Death" algorithm (MH-BnD). 169

3.4 Interaction rates table which defines a "cross-pattern" model. 187

3.5 Interaction rates table of a model favoring $\{30^\circ, 60^\circ\}$ interaction angles. 198

3.6 Interaction rates table of a model favoring 45° interaction angles. . . 199

3.7 Interaction rates table of a "cross-pattern" model with better connectivity. 212

3.8 Different weight functions for the vector field alignment models. . . 214

3.9 Different weight functions for the feature-based soft constraints. . . 219

List of Algorithms

1 Sharp normal interpolation shader 84

2 One iteration of Metropolis-Hastings “Birth-and-Death” algorithm for generating fibre systems. 170

3 Fibre neighborhood search $Nb(\psi;\Phi,R)$. 177

Introduction

Motivation

Segmentation. For several decades the edge detection algorithms have been dominant segmentation techniques in computer vision and image processing [160]. A typical vision problem is to search for all the pixels in a target still image or a video sequence which belong to the contour or the edge boundary of one object or a group of overlapping objects for a subsequent object(s) segmentation. A standard edge detection method looks for the image areas where a signal undergoes a sharp jump from a locally constant lower level of intensities to a higher level. These methods have proved their effectiveness in many applications in 3D shape reconstruction and recognition, image compression, motion tracking, image enhancement and noise removal, in which the objects have non-degenerate interiors and are of the size of more than just a few pixels.

But there is a class of natural images which requires special attention and implies developing different detection techniques. It includes the real-world images which are formed by a mixture of thin curvilinear features — *curvilinear structures* — whose image widths are on the level of just a few pixels. For example, curvilinear structures are dominant features in the images of fingerprints, the microscope images of blood vessels, the satellite images of roads and rivers, the images of hu-

man skin replicas with meso-scale furrows and others. Many industries would benefit from using robust detecting methods tuned for curvilinear structures, including but not restricted to cosmetics industry, dermatology, biomedicine for investigating and detecting healthy patterns of skin furrows [121, 126, 29], [148, Part II, Ch. 3], geographic information system industry (GIS) for detecting linear structures (e.g., like roads) from the satellite imagery [46, 51, 48], robot vision systems [77] or automatic classification of protein crystallization [17] for curve tracking. The main problem in detecting curvy structures is that the information about trace points is unknown [100]: once the pixels belonging to a curve are found they have to be properly connected to form a nice curve. The tracing process becomes even more complicated when the curvy structures happen to be broken due to a relatively weak signal in some fragments of the image. Many real-world textures contain intricate arrangements of objects needed to be detected with a high degree of certainty. Available detection techniques usually involve a great deal of user guidance at the beginning to allocate potential feature locations and/or during detecting process to successfully segment such objects. To avoid this potentially very laborious process, it is desirable to have a small-input algorithm which runs automatically and may require only episodic help from a user. And finally, it is advantageous to reconstruct topological information about detected curves and lines from the images with complex feature arrangements: knowing adjacency structure of connected curvy features helps in performing statistical analysis on the features to find correlations between nearby feature curves.

GPU Rendering. The curvilinear features also play very important role in computer graphics providing ways of adding small detail to virtual objects — necessary to increase rendering photorealism. Such features can be attributed to an object boundary or follow an object shadow, and may also define fine-scale geometry, for

example, creases, cracks, or furrows. The quality of rendering depends on how well sharp texture features are captured. Texture mapping has become a primary model to map features onto objects to be rendered. A standard bitmap texture is usually represented by a raster image which contains a discrete values of the feature map, and provides a fast random access to its values — texels — which are linearly interpolated during rendering to get feature samples at arbitrary location. The quality of (originally continuous) features can be lost already during their rasterization to bitmap texture. But what brings more trouble is that close views of textured objects inevitably result in blurring which cannot be resolved by standard interpolation techniques, including trilinear filtering by mipmapping. A resolution independent feature-based interpolation, sharp at feature curves, is desirable.

All the existing feature-based methods explicitly represent the discontinuity lines and store them either directly in the texels or in a spatial hierarchical structure. Real-time rendering of feature-based textures is mapped to dedicated graphics processors (GPU) by executing C-like *pixel shader* programs. Majority of such feature-based techniques accept line segments only as discontinuity lines, which cannot render true feature curves as the corners at segments joints will be observable at some resolution. Some techniques support curvy features but do not allow them to intersect or require them to follow the boundaries of the objects, while its adaptation to independent open feature curves would require an additional effort.

In this work we focus on normal maps — 3D textures which replace the original normals of the shape to produce appearance of small-scale geometry on the surface without increasing the size of the underlying mesh — containing visible sharp features, common for surfaces with creases, wrinkles, and dents. Such sharp features are defined by some profile as a function of distance and its gradient to the discontinuity curve. As a consequence, the quality of rendering of normal maps along discontinuity

curves depends on the quality of interpolation of the distance function and the distance gradient field. Some existing techniques are able to approximate well distance function calculation for points sufficiently close to discontinuity curves; however, it would be impossible to apply these methods for features with relatively large widths and/or with high curvature values occurring along their discontinuity curves. All distance-based existing techniques consider only signed distance fields which are not friendly with representing open feature curves.

Synthesis. The task of generating realistic textures with rich feature content could be as difficult as developing texture rendering techniques which preserve the features. Nonparametric sample-based texture synthesis and statistical texture modeling have been the most successful techniques for generating a new high quality arbitrary size texture from a given small reference texture. Both techniques have demonstrated their best performance on textures which can be considered as outcomes of local and stationary random fields of 3D color values. The former technique makes use of the locality property to gradually grow an output texture by copying the fragments of the reference texture whose small neighborhood best matches the current neighborhood in the output. The latter technique decomposes the reference and the output image into a hierarchy of subbands by applying linear filters and progressively modifies the output subbands by matching certain statistical functions (e.g., marginal histograms as in [61]) calculated on the related reference and output subbands. However, the assumptions are too strong for a variety of natural textures with the observable regularity; in particular, both techniques tend to fail on textures with well structured patterns producing unwanted gaps in features or breaking coherency. The only feature-based technique by Wu and Yu [151] preserves well the curvilinear features, but could introduce observable periodicity, and, what is more important, lacks of any control for

desirable pattern modifications. For example, it would be hard to modulate feature locations and orientations in a desirable way, say, make them closer to or farther from each other at some locations, or change the average distribution of angles between the nearby features. Moreover, what is shared with all sample-based texture synthesis techniques is that the existing control is not intuitive and selecting optimal parameters is routinely done in ad-hoc manner, which most of the time require significant experience and time. Also, the parameters which work perfectly on one set of reference images could completely fail on other sets.

Generating features directly can preserve desirable feature correlation patterns while enabling an intuitive control to safely modulate feature correlations. The closest, and probably the only techniques of this nature, includes spatial random processes of geometric objects which are a popular subject in stochastic geometry. Spatial point processes based on models of local interactions between the individual points have been used in describing and modeling a majority of real-world point-based randomly distributed collections which manifest a fair amount of regularity. In particular, Gibbsian point distribution models are easy to interpret and convenient to work with. The resulting patterns of points are aperiodic by construction. Unfortunately, to the best of our knowledge, an analogue of Gibbsian distributions for curves and lines does not exist. Random arrangements of curves and lines have been only studied in the context of completely random Poisson special processes. A random process with interaction-based distributions which simulates random sets of curves or lines, with intuitive local and global control is desirable.

Thesis and methodology

This dissertation addresses three key problems in computer graphics, computer vision and stochastic geometry related to curvilinear features: (1) detecting curvilinear structures from the real-world images, (2) real-time resolution-free sharp rendering of textures with curvilinear features, and (3) stochastic modeling of near-regular patterns with networks of curves distributed randomly in the plane.

Curvilinear Structure Detection. We want to develop a robust automatic technique to recognize curvilinear structures from still images for the case when such structures form complicated networks and can be weakly presented. A user involvement should be reduced to a minimum. The output should form a plane graph (a planar graph embedded in the plane) with its edges covering all the presented curvilinear structures in the input image and its nodes representing all the intersections of the image structures.

Feature curves sharp rendering. We want to build a procedural texture which stores a normal map containing discontinuity features and smoothly interpolates the map in real time preserving the features along the discontinuity at any resolution. A crucial requirement on the geometry of the discontinuities is to consider features following quadratic curves and to allow complex intersections between the feature curves.

Generating random patterns with networks of curves. For the problem of texture synthesis, we want to develop a parametric texture synthesis algorithm capable of generating random near-regular patterns of curvilinear features distributed according to some probabilistic model. The set of parameters of the model should be intuitive and should define explicitly the local correlation pattern between the nearby curves. User also should have a global control to be able to synthesize non-stationary tex-

tures.

To detect curvilinear structures from still images, we optimize a set of adaptable snakes by applying merge-split operations. To avoid a laborious process of finding plausible initial locations for the snakes, a target image is populated with a large set of randomly placed short snakes. We apply reconfigurations of evolving snakes dynamically: during optimization we change topology of the snakes, if necessary, by merging the entirely aligned nearby snakes or by splitting the snakes which are aligned strictly along their interiors. At each split operation, the involved snakes are linked to each other by introducing a linkage node which stays till the end of optimization unless only one snake is left linked to it later during optimization. The locations of the nodes are not fixed and can propagate within the image together with its adjacent snakes. The optimization is stopped when the spline energy of the entire snake system stabilizes. The outcome of the algorithm is a plane graph covering all the curvilinear structures in the image.

The main challenges are (1) developing an efficient data structure for search queries of nearby aligned snakes, (2) and implementing safe links which would not bring the system of connected snakes unstable during optimization.

To build a sharp resolution-free procedural normal field, we blend at rendering time two parts, continuous and discontinuous. The continuous part is obtained by the standard texture interpolation of the normal map. The discontinuous normal field is computed from the feature curve network only and is defined as a function of the unsigned distance to the curves and its gradient — such function can represent the feature cross-section profile and may be provided as a 1D texture. Thus, the original problem can be reformulated in terms of sharp interpolation of distance function and its gradient. Our solution is to encode the discontinuity curves in the texel and to prohibit interpolation of distance and gradient samples located on the different sides

of the discontinuity. This allows to keep normals sharp at any resolution while using fixed size textures. We use unsigned distance field, which allows for more natural and flexible representation of curvy features.

Our technique has two main components: the first runs as a pixel shader and computes the discontinuous normal by interpolating the distance and its gradient and by applying the feature profile afterwards. Computing a suitable distance function is the essential part of our algorithm. To ensure that feature curves are artifact-free at any resolution, the distance function is interpolated as close as possible to the actual Euclidean distance. At the same time, it should be possible to evaluate it efficiently, so that it can be done in a shader. It should be exactly zero on the feature curves for them to be resolution-independent. We also require that its gradient to be perpendicular to the curve. To achieve high performance, we compute distance and its gradient by interpolating samples on curvilinear triangles formed by partitioning the texels. The samples from the texel corners are propagated to the triangle corners while respecting the discontinuity curves. The second component is a preprocessing algorithm that encodes discontinuities into a discontinuity descriptor texture. This texture contains two components — discontinuity signature and discontinuity configuration — which are used in a shader to fully reconstruct the features within a texel. At this stage we convert all complex feature intersections into valid configurations by running a constrained thin-plate spline optimization within texels.

Our synthesis algorithm is conceptually different from the existing texture synthesis techniques found in graphics and vision. The latter are based on multi-scale matching fragments of output texture with reference sample texture, and which are generally not capable of synthesizing curvilinear features, like lines and curves, without introducing visible artifacts. We synthesize the feature arrangements themselves in the texture domain by representing such arrangements as outcomes of some ran-

dom process of curves and lines — *random fibres*. In contrast to the existing models of fibre processes, our model of random fibres is local and is able to produce near-regular fibre configurations. We adapt the Gibbs distributions of spatial point processes to our fibre model as they allow to describe explicitly the local dependencies between the objects (fibres, in this case). Such dependencies are expressed as interaction potentials which, roughly speaking, describe the level of repulsion or attraction between infinitesimal parts of nearby fibres. In particular, the parameters of such potentials define favorite and prohibited angles between tangents of two fibres. This allows to specify local pattern directly. In this work we focus on linear fibres which are represented by the line segments. We apply a variant of the Monte Carlo Metropolis-Hastings algorithm to generate samples of random systems of linear fibres. We also provide the means for a global control by adding several types of constraints to the Monte Carlo simulation, including control on the fibre density, connectivity, and on aligning the fibre system with a given vector field. To convert resulting systems of random line segments, usually not C^1 connected, into smooth curves, we developed a streamline generation algorithm which generates overlapping sets of coherent curves. The curves closely follow cross-orientation vector field (COVF) which is optimized from a tangent field of a given systems of linear fibres, which are preliminary generated by our Gibbs-type simulation algorithm.

Contributions

We developed a set of computer graphics and vision algorithms which explore different aspects of working with curvilinear features and structures. These algorithms can potentially form the following production pipeline: (1) a network of curvilinear structures can be segmented from a target image by using our detection technique to

produce a plane graph covering all the structures, (2) the graph is analyzed to capture local correlation pattern between nearby graph's edges — the results of this analysis can be used to generate a plausible set of parameters to our fibre simulation model which can in turn generate a network of curvilinear structures whose local pattern closely matches that of the original image and which can be modulated globally, (3) the resulting network of curvilinear structures can be used as a basis in creating a very complex texture with sharp (possibly wide) features closely following the network — one can use our GPU rendering technique to render objects wrapped by the texture which will prevent sharp details at any resolution. Images with intricate networks of human skin furrows is a good example of the input to such a pipeline.
Here is the detailed list of our contributions which are covered by this thesis:

Detection of curvilinear structures.

- A new robust automatic algorithm for detecting networks of thin curvilinear features from still images based on optimizing adaptable snakes. The algorithm does not make any assumption about the topology of curvilinear structures;

- Snakes optimization algorithm runs in three phases: (1) with no interaction of snakes, pulling snakes towards the image curvilinear structures, (2) with merging the aligned snakes down to thin bundles of snakes already arrived at their local optima, and (3) with merging/splitting the snakes to link aligned and intersecting snakes into a graph-like structure;

- The output is a plane graph whose edges are splines which closely follow all curvilinear structures, possibly weakly presented in the image, and whose nodes are located at all the intersections of the structures.

Real-time rendering of textures with feature curves.

- A new real-time GPU-based algorithm to render normal maps containing sharp visible features, which are preserved at any resolution along the discontinuity curves and are smooth away from the discontinuities;
- Our discontinuity features go along either line segments or quadratic Bezier curves. Discontinuity curves are explicitly stored in the textures of discontinuity descriptors;
- We use unsigned distance field to represent the features. The field is interpolated as close as possible to Euclidean distance during rendering;
- Open and close features are allowed;
- A preprocessing algorithm converts complex feature intersections to a format suitable for storing in discontinuity descriptors. Up to two curves can be stored in the texel;
- We implemented embossing which enables a user to define the normals near the feature curves by providing an arbitrary cross-section profile;
- Originally developed for normal fields, our technique can be applied to any type of discontinuous textures whose discontinuity features can be expressed as a function of distance and/or its gradient.

Parametric synthesis of random fibre systems.

- A new parametric model for representing and generating networks of random fibres based on Gibbs-type distributions of local interdependencies which are described by an intuitive set of parameters;

- Detailed description of random linear fibres — a subclass of random fibres operating with line segments only — with providing a parametric global control;
- A new algorithm of converting systems of linear fibres into overlapping sets of coherent smooth curves.

Thesis organization

This dissertation is divided into three relatively independent parts: (1) our method for detecting the curvilinear structures in still images is covered in Chapter 1, (2) our real-time rendering technique for textures with curvilinear features is described in Chapter 2, and (3) our model for generating textures with random networks of curvilinear features is presented in Chapters 3, 4 and 5. Each part contains a separate discussion on motivation and previous work regarding the topic of the part and describes our methodology to address the particular problem related to the part. The chapters can be read in an arbitrary order.

The third part of dissertation consists of three chapters. Chapter 3 contains a detailed overview of stochastic point processes in general and Gibbs point processes in particular. It outlines our model of general fibre system and describes in detail our Gibbs type model of linear fibres and the Metropolis type algorithms to generate random samples of linear fibre systems distributed according to a given interaction model. Our method of adding the local and global constraints are also described in Chapter 3. We formed a separate Chapter 4 for describing our variant of streamline generation algorithm. Results for generating and rendering networks of curves are presented in Chapter 5.

Chapter 1

Curves detection by autonomous snakes

1.1 Introduction

The problem of edge detection has been an intense topic of research for already several decades, resulting in developing a variety of powerful edge detection techniques based on gradient filtering, parametric fitting, and the optimal enhancement paradigm (see, for instance, overview of such methods in [160]). Unfortunately, while capable of finding edge points, these methods do not provide a way to link the points into chains to generate networks of curves. Besides, most of them are tuned to find object boundaries and usually fail to detect one dimensional features like lines and curves. The following techniques were specifically developed for a problem of curve detection.

The close-up images of human skin are an important example of images on which the conventional edge detectors show inferior performance. The main fine-scale geometry features on the surface of the skin are formed by intricate inhomogeneous

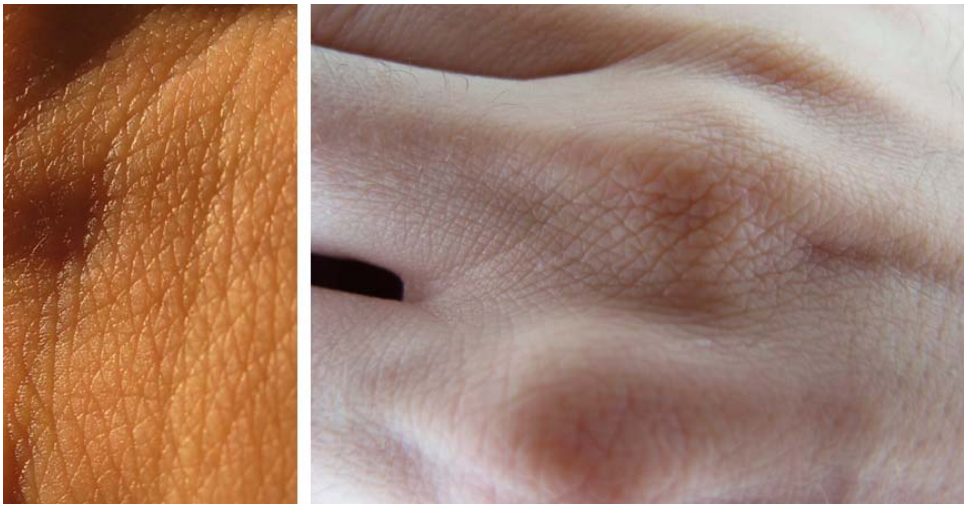


Figure 1.1: Skin images with visible fine-scale structures.

networks of the skin furrows and enlarged pores. The importance of robust segmentation of skin furrows has been widely recognized in many fields, including cosmetics, dermatology, biomedicine [121, 126, 29], [148, Part II, Ch. 3], for instance, to detect healthy (cancer-free) patterns of skin surface or to find effective ways of hiding unwanted visible skin features. The skin can be captured by either taking high resolution pictures of skin fragments illuminated by fluorescent light source (a ring-shaped lamp with a cover on the camera's side can be placed between the camera and the skin sample), or by photographing fragments of the highly-detailed skin negative replicas. The skin images taken by using the former method are illustrated in Figure 1.1, while the replica pictures taken from different areas of the human body are shown in Figure 1.2.

Maintaining evenness of skin color in the images and furrow sharpness along the wide area of skin is challenging. A network-type topology of the skin furrows makes their detection a very challenging problem, especially when the image signal is noisy or when depicted furrows are weak and broken. A number of methods have

Figure 1.2: Images of skin negative replicas taken from different ares of human body. Left to right: back, cheek, and forehead.

been specifically developed for detecting the curves from the images. For most, the performance could significantly degrade in detecting networks of curves similar to that of formed by human skin furrows. Many could require a substantial user involvement during detection, as for the case of skin images.

The simplest technique, which was extensively utilized in early detection systems, is the thinning procedure [74] . The standard thinning algorithm iteratively cuts pixels located on the boundaries of a target object until only one-pixel skeleton remains. Two requirements should be satisfied during thinning: a pixel is cleared only if this does not change connectivity of the object and if the background topology is preserved. One can apply in advance such technique to an one-bit image representing detected edge points. The points located on the resulting skeletons are then consecutively linked together to form curves. Another way is to apply the thinning algorithm directly to a gray scale image using one of the following one-bit algorithm extensions [39, 154, 1]. The main drawback of this approach is that it does not handle well images with complex structure and can not detect weak curves which are either blurred or fragmented due to the noise.

Haralik [59] presented a method for classification of ridges and valleys by preliminarily transforming noisy original image into its smooth approximation. For a continuous image, he classifies a pixel being on a ridge (valley) if the directional derivative at this pixel is zero in the direction of greatest magnitude of the second derivative. To make the original image continuous, a neighborhood of each image's pixel is fitted to a bicubic surface, so that all derivative calculations are done for the resulting smooth surface. Ridges and valleys, defined in this method, closely correspond to the thin curvilinear features that we want to detect from the images. However, the method suffers from detecting a large number of false ridge pixels in the places with radially symmetric surface profiles, and it incorrectly fits the areas with narrow ridges and valleys.

Among the early methods of linking detected edge points into curves is Hough transform (HT) [66]. HT is applicable for those curve patterns which can be described by simple parametric shapes. Duda and Hart [37] applied HT for detecting straight lines, while Ballard [11] studied possibility of extending HT to detect general shapes, and applied HT for classifying circular and elliptical shapes. In addition to the image space, the space of parameters of chosen shapes is created during HT. In such parametric space, one accumulates the statistics on how many detected points in the image space a curve with given parameters intersects. A set of parameters with the accumulated values greater than some threshold will define a corresponding set of curves we are looking for. The method is robust against image noise and possible gaps in the source curves. However, it is restricted to simple shapes with small number of parameters (less then 3), so that it can not be effectively applied to detect curves with complex intersection patterns. Besides, because of inevitable introduction of accuracy error during discretization of the parametric space, the quality of curve detection may be poor.

Searching for curves following the boundaries of the analyzed objects is one of the goals of the image segmentation techniques. Dynamic programming graph search is one class of these techniques, where the problem of boundary detection is formulated as a search for an optimal path in a certain directed graph. According to the Udupa's formulation [140], such graph has the image pixels as its nodes; the 8-neighboring pixels are connected by edges; and, every edge is assigned a cost, which is low if the edge is aligned with a close boundary. Udupa used the dynamic programming to find an optimal path from a seed node to all other nodes by reducing the cumulative path cost. Morse et al. [88] used Udupa's formulation to search for a piecewise optimal path. They required user to specify control points along the target object's boundary and ran a search iteratively with no user interruption. A drawback of this approach is that in the case of unsatisfactory segmentation a user has to repeat iterations with a new set of control points. To overcome this problem, Mortensen [89] developed a 'Live-wire' technique, which searches for an optimal path from a specified start node to a goal node with a possibility of immediate feedback from a user. Based on the live-wire method, Mortensen et al. [90, 91] presented a set of image composition tools, called 'Intelligent scissors', which incorporate on-the-fly training, an efficient implementation of the Dijkstra's search for an optimal path, and tools for optimal selection of goal nodes. In this method a user defines a search direction; therefore, an automatic search is not available.

In contrast to the searching techniques described so far, Geiger et al. [50] proposed a non-iterative method for detecting deformable contours using the dynamic programming. They defined sufficiently large windows around a user-specified contour and ran a version of the dynamic programming algorithm inside these windows. Their method was guaranteed to always produce a global optimum. They also presented a faster version of the search algorithm by applying a multi-scale approach.

Though global optimality was not guaranteed in this case, a speed up might approach of a factor of 20. This method was effectively used for detecting, tracking, and matching of a small number of objects. However, the manual phase may be significantly increased in the case of images with a large number of features.

Witkin et al. [136] built a segmentation technique based on active contours, or snakes. Represented by deformable splines, snakes reach their minimum energy when located along the image features with high gradient magnitude, like object's boundaries and individual curves. Initially placed around a target boundary, snakes quickly evolve to the closest optimal position by minimizing their energy. However, the method can not guarantee convergence to a global minimum and strongly depends on a choice of snake parameters and its initial position. These limitations were partially resolved by a method of McInerney and Terzopoulos [85], where they presented geometric and topologically adaptable snakes. In their approach, snakes grow, shrink, split and merge with each other with a goal of finding all presented features in the image. This makes snakes to be more autonomous, but a user still needs to provide information about possible shapes in the analyzed image.

The segmentation techniques based on graph search, dynamic programming, or active contours require a user to interact with a system either by specifying original locations of deformable shapes, or by directly guiding the optimization time to time. However, there exist many examples of natural images which are comprised of hundreds of distinct feature curves to be detected from. It just would not be feasible to find all seed locations manually. To address this type of textures we look for a more automatic detection approach, which assumes a minimal manual phase. We also need to know how the curves interact with each other. This analysis was not covered in the methods presented so far. To summarize, we aim to develop a method of detecting thin curvilinear features from noisy images reducing a manual step of initial locations

to a minimum and is capable of handling complex curve intersections.

1.2 Snakes optimization framework

Our goal is to develop a method that produces a network of curves whose intersections are explicitly represented by the nodes of the network, and which is robust with respect to possible breaks occurring along the curvy features in the image.

The need for an explicit curve representation makes the active contour models — snakes — a natural choice for a base model. The results presented in [136] show that the curvilinear objects can be extracted from an image. In addition, snakes have a beneficial property of being at least C^1 smooth, which is crucial for many real world textures containing continuous curves, e.g., meso-scale human skin furrows form overlapping sets of piecewise continuous curves. By tuning the parameters of the snake energy function, it is possible to ensure that snakes align with thin curvy features and are able to detect relatively weak features or features with gaps. Point-like detectors on the other hand may go through a complex adaptive search for a plausible threshold to find robustly all features, which are potentially vary significantly in strength. Finally, snakes can be easily connected into networks.

A snake — an energy minimization spline — is pulled toward the near feature where its energy has a local minimum. Three terms are included [136] in the snake's energy functional, $E(\mathbf{s};u)$: standard thin-plate energy of the spline $E_{\text{spline}}(\mathbf{s};u)$, a user-provided external energy $E_{\text{image}}(\mathbf{s};u)$, and the energy imposed by a set of constraints $E_{\text{cnstr}}(\mathbf{s};u)$

$$E(\mathbf{s};u) = E_{\text{spline}}(\mathbf{s};u) + E_{\text{image}}(\mathbf{s};u) + E_{\text{cnstr}}(\mathbf{s};u), \tag{1.1}$$

where $\mathbf{s}(u) : [0,1] \rightarrow \mathbb{R}^2$ is some parametrization of the spline. $E(\mathbf{s};u)$ describes the

energy of the spline $\mathbf{s}$ at a point $\mathbf{s}(u)$. The spline optimization problem is formulated as a minimization of the following energy functional

$$E(\mathbf{s}) = \int_0^1 E(\mathbf{s};u)\,du \to \min\,. \tag{1.2}$$

E_{spline} is a sum of membrane energy (elastic string) and thin-plate energy, given by $E_{\text{spline}}(\mathbf{s};u) = \alpha|\frac{d\mathbf{s}}{du}|^2 + \beta|\frac{d^2\mathbf{s}}{du^2}|^2$. The external energy E_{image} gets its largest values in the areas of the high contrast and depends only on the images signal: $E_{\text{image}}(\mathbf{s};u) = -E^2_{\nabla I}(\mathbf{s};u)$, where $E_{\nabla I}(\mathbf{s};u) = |\nabla I(\mathbf{s}(u))|$. To derive a force balance equation one can use variational principles which, in particular, postulate that a minimizer of the following functional $J(\mathbf{s}) = \int_{u_0}^{u_1} g(\mathbf{s},\mathbf{s}_u,\mathbf{s}_{uu})\,du$ should solve the following Euler-Lagrange equation $\frac{\partial g}{\partial \mathbf{s}} - \frac{d}{du}\frac{\partial g}{\partial \mathbf{s}_u} + \frac{d^2}{du^2}\frac{\partial g}{\partial \mathbf{s}_{uu}} = 0$. So, by considering $g(\mathbf{s},\mathbf{s}_u,\mathbf{s}_{uu}) = \alpha|\mathbf{s}_u|^2 + \beta|\mathbf{s}_{uu}|^2$, one can derive the following *force balance equation*

$$\alpha\mathbf{s}_{uu}(u) + \beta\mathbf{s}_{uuuu}(u) + \nabla E_{\nabla I}(\mathbf{s};u) + \nabla E_{cnstr}(\mathbf{s};u) = 0. \tag{1.3}$$

which defines an optimal position of the snake. The first term describes the membrane behavior (it tries minimizing the distance between the control points), and defines the bending force. The second term characterizes the thin-plate behavior (it flattens the high curvature areas) and defines the elastic force.

The balance equation is solved numerically by discretizing a given set of snakes $\mathbf{s}_1,\mathbf{s}_2,\ldots,\mathbf{s}_M$ into an $N\times 2$ "vector" of 2D points (the *snake nodes*)

$$\begin{aligned} X = (&\mathbf{s}_1(u_1^1),\mathbf{s}_1(u_2^1),\ldots,\mathbf{s}_1(u_{N_1}^1),\\ &\mathbf{s}_2(u_1^2),\mathbf{s}_2(u_2^2),\ldots,\mathbf{s}_2(u_{N_2}^2),\ldots,\mathbf{s}_M(u_1^M),\mathbf{s}_M(u_2^M),\ldots,\mathbf{s}_M(u_{N_M}^M))^{\mathrm{T}}\,, \end{aligned} \tag{1.4}$$

where $N = N_1 + N_2 + \cdots + N_n$ is the total number of snakes nodes.

A discrete version of the balance equation 1.3 can be written by the following

system of equations

$$\mathbf{A}X + \mathbf{F}(X) = \mathbf{0}, \tag{1.5}$$

where $\mathbf{A}$ is an $N \times N$ pentadiagonal banded matrix corresponding to a finite difference approximation of the spline force operator (given by the first two terms in equation 1.3), and an n-th row of the $N \times 2$ matrix $\mathbf{F} = \left[\mathbf{f}^1 \,|\, \mathbf{f}^2\right]$ represents an external force vector (f_n^1, f_n^2) (given by the rest of the terms in equation 1.3) applied to the corresponding n-th snake node. Nonlinearity of the resulting equation makes it impossible to find an exact solution, therefore an iterative method should be used instead. The Newton algorithm is the fastest among other algorithm near a solution, and is based on the numerical integration of the following system

$$\frac{dX}{dt} + \mathbf{A}X + \mathbf{F}(X) = \mathbf{0}.$$

The locations of snake nodes $X(t)$ are considered as functions of time t. They evolve (when the derivative dX/dt is not zero) until the balance equation 1.5 is satisfied. The following are the explicit and implicit Euler schemes to solve the ODE above:

$$\gamma(X_{t+1} - X_t) + \mathbf{A}X_t + \mathbf{F}(X_t) = \mathbf{0}, \tag{1.6}$$

$$\gamma(X_{t+1} - X_t) + \mathbf{A}X_{t+1} + \mathbf{F}(X_t) = \mathbf{0}. \tag{1.7}$$

The authors showed [136] that inside some range for spline parameters and for the iteration steps, the Euler implicit iteration scheme

$$(\mathbf{A} + \gamma\mathbf{I})X_{t+1} = \gamma X_t - \mathbf{F}(X_t) \tag{1.8}$$

rapidly converges to a local solution. The matrix $(\mathbf{A} + \gamma\mathbf{I})$ is pentadiagonal, so that its inverse can be efficiently calculated by using LU decomposition in $O(N^2)$ steps.

The main difficulty with obtaining a reasonable solution is in determining the best initial locations of the snakes and in choosing the eight values for snakes lengths.

In the previous work, these locations were chosen manually and snakes are guided during optimization to avoid getting in spurious local minima. Our method optimizes a large number of snakes and filter out the snakes which are stuck at fake optimal locations. In this case we can afford to choose the initial locations for the snakes at random.

1.3 Interactive active contours

Our approach is to populate a given image with a large number of short random snakes. With a sufficiently large number of initial snakes, all curvilinear features will have a number of sufficiently close snakes, which can be aligned with it; at the same time, snakes that are not close to any feature and end up converging to spurious local minima need to be deleted. As shown in Figure 1.3, a large fraction of snakes were pulled towards the features. However, the snakes remaining at random positions are not easy to remove. An obvious approach to solve this problem is to threshold the spurious snakes, getting rid of those snakes that are either not aligned with the dominant local orientation or/and are not located in the area of high snake density. Besides the fact that the thresholding may significantly slow down the detection process, one should always find a balance between removing spurious snakes and preserving aligned snakes. A threshold may be proper to cut not aligned snakes, but at the same time using this threshold, we may remove important snakes, which accurately localize the features.

In addition to the problem of thresholding of specious snakes, we found that the bundles of valuable snakes, pushed to the furrows, can be wide. A close view Figure 1.4 reveals forming two types of bundles: relatively narrow and wide bundles. Any simple process of merging the immediate neighbors in a wide snake bundle may

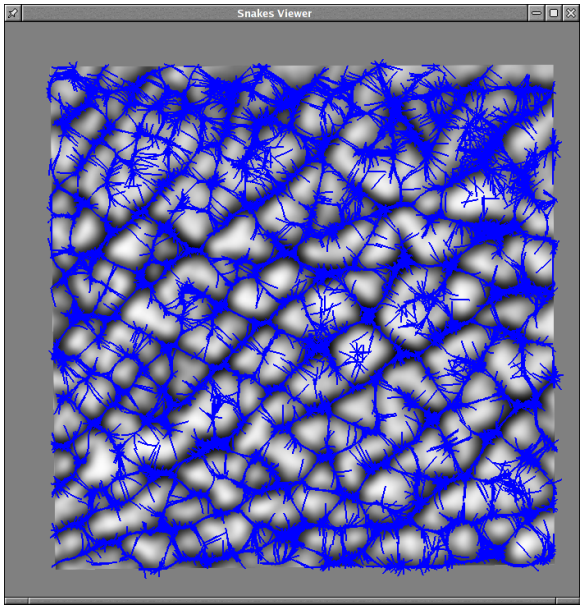


Figure 1.3: Snakes snapshot after several iterations of energy minimization of randomly placed short snakes.

produce several bundles sufficiently far from each other not to be merged further, see Figure 1.5. Then the following questions should be raised: how to detect the entire bundles of valuable snakes, and how to choose a representative snake from a significantly wide bundle. A solution for both questions, as well as for the problem of thresholding, comes from thinning the bundles dynamically instead of running filtering as a post-process.

We impose an additional requirement that only one-snake width curves are allowed to be presented in the image at the end of snake iterations. Specifically, no pair of snakes with a parallel segment should be close to each other. If the collection of snakes is reduced using this rule, the remaining snakes unambiguously and compactly represent the curvilinear features in the image. We have considered two strategies to impose this requirement: to enforce the requirement once the snakes'

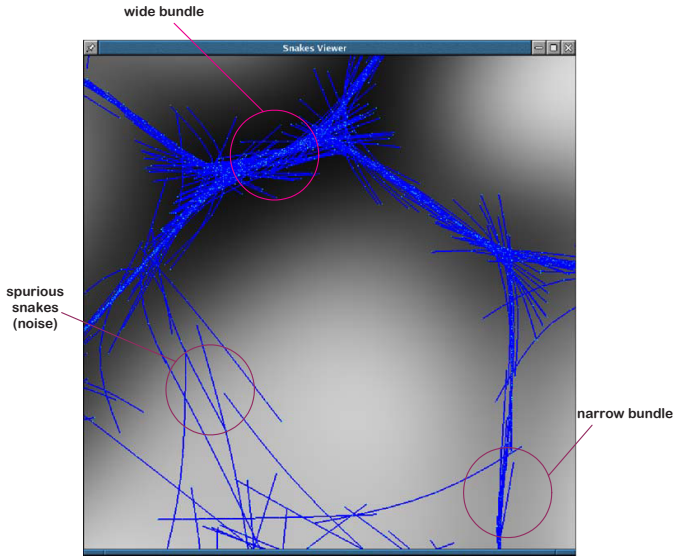


Figure 1.4: A close view at the resulting bundles of snakes. Some snakes were detected falsely.

motion is terminated, and to merge snakes during snake optimization. As shown in Figure 1.3, static merging does not handle well bundles, which could be significantly wide. On the other hand, dynamic merge can guarantee that the current bundle representatives will be close enough to the features for merging down to one snake. We apply the dynamic merge of the aligned snakes in our approach, so we need to have an efficient procedure for finding a new positions of just merged snakes and their energy.

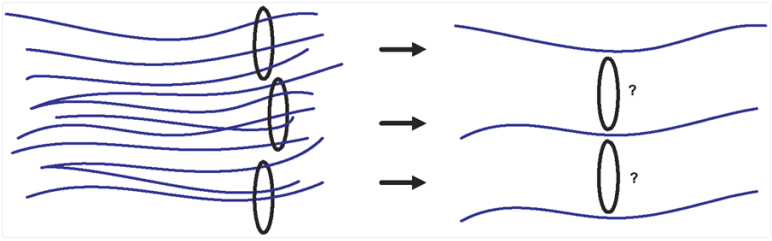


Figure 1.5: Thinning relatively large snake bundles may be problematic.

1.3.1 Merge-split operations on snakes

Now we will describe our merge and split operators which are applied to snakes during optimization. We define a pair of snakes, s_1 and s_2, to be in the chain configuration if each snake can be split into two or three parts s_{1i} and s_{2i}, $i = 1,2,3$, so that one snake is a continuation of the other through the aligned parts parts s_{1i} and s_{2j}. The *merge operation* produces a new snake by averaging the aligned pieces to a new piece s_{mid} and by concatenating s_{mid} with the other misaligned pieces. Figure 1.6 shows all the possible allowed chain configurations and the results of applying the merge operation on them: 1) for the case of the alignment of snakes tails, s_{mid} results from averaging s_{12} and s_{21}, so that a new snake is formed by $s_{new} = [s_{11}, s_{mid}, s_{22}]$, and 2) for the case of the alignment of entire body of one snake with a part of the other snake, s_{mid} is an average of s_{12} and s_2, so that $s_{new} = [s_{11}, s_{mid}, s_{13}]$.

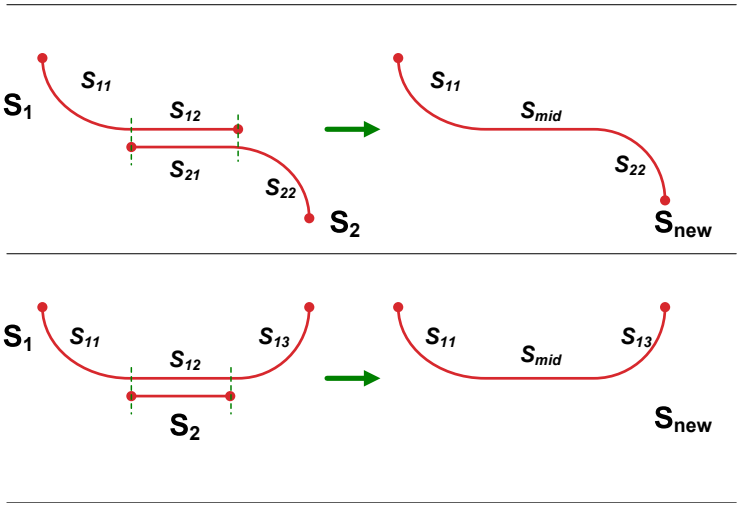


Figure 1.6: Merging of two partially aligned snakes. Top: snakes' tails are aligned. Bottom: one snake is entirely aligned with a part of the other snake.

As we intend to merge snakes bound to a feature, the merge operations are applied only after the snake simulation moves them close enough to such features. One

of the open questions is achoice of the time when merging should begin. This choice is really a tradeoff between not letting to merge spurious snakes together at the first iterations and not missing important merges of the snakes that cover the weakly presented features, as they usually move apart on the early iterations.

The set of possible chain configurations for which the merge operation can be performed is very limited. It misses one important configuration when the snakes may have an aligned area in the middle. This often occurs at the intersections of features, which are particularly important to be detected to capture the topology of curvilinear structures in the image. This limitation makes the merge operation insufficient, so that an additional procedure is needed, which merges two nearby snakes with aligned interiors.

This is achieved by a *split operation*. Similarly to the merge operation, we merge the aligned parts to one of the two snakes. From what left of the other snake, we create two new separated snakes. All allowed configurations and the split operation in action are illustrated in Figure 1.7.

The same considerations, which we described before ruling in favor of choosing a dynamic merging (i.e., simultaneously with the iterations) versus post-process static merging, can be applied for the split operation — we will be maintaining a dynamic splitting. It requires to "link" snakes at the places where one of the snakes one of the snakes was cut, so that the snakes do not separate during iterations. Thus, the snakes optimization algorithm has to be adjusted to allow linking the snakes.

Two different ways can be tried to implement links between the snakes. The simplest way is to connect two nodes of the snakes by a spring [136]. It does not modify the original snake model as spring forces can be included in the snake energy equation. Unfortunately, what we found from experience, the resulting system becomes stiff that may lead to chaotic oscillations of snakes, which most likely results

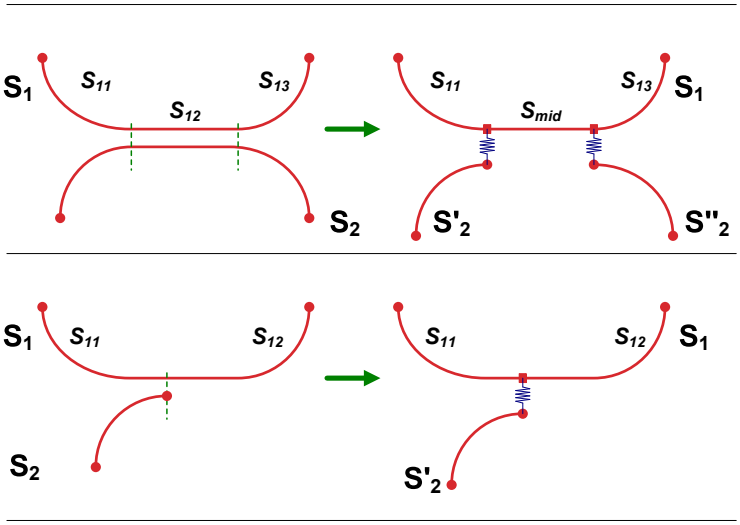


Figure 1.7: Splitting of two partially aligned snakes. Top: Snakes share their aligned interiors. Bottom: Just link the tail node of one snake to an interior node of the other.

in diverging the system optimization.

We apply a different approach: instead of introducing a spring force between snakes nodes, which could contribute to a system instability, we impose equality constraints on the snakes nodes. The corresponding isoperimetric constraints should be added to the original variational problem. We use Lagrange multipliers framework to incorporate isoperimetric equality constraints into the original mathematical model of snakes 1.5.

The following integral form represents the equality constraint which guarantees for two nodes i and j at different snakes to coincide

$$\int \left[D(u_i) - D(u_j)\right] \mathbf{s}(u)\, du = 0\,. \tag{1.9}$$

Its discrete form is given by

$$\mathbf{w}^{\mathrm{T}} X = 0, \tag{1.10}$$

for a vector $\mathbf{w}$ with $N-2$ zero components and $w_i = -w_j = 1$, if nodes i and j are constrained to be at the same location, i.e., $\mathbf{s}(u_i) = \mathbf{s}(u_j)$, $1 \le i, j \le N$, and the snakes locations X defined by equation 1.4 with N as the total number of nodes of the snakes under optimization.

To reduce an optimization problem 1.2 with $K > 0$ constraints of the form 1.9 to an extended system of equations of snake force balance we introduce $2K$ Lagrange multipliers $\{\lambda_k\}_{k=1}^K$ and $\{\mu_k\}_{k=1}^K$ as an additional set of variables (λ-s are for x-coordinates and μ-s for y-coordinates of snake nodes). The multipliers are arranged in the following $K \times 2$ matrix

$$\Lambda = \begin{pmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_K \\ \mu_1 & \mu_2 & \dots & \mu_K \end{pmatrix}^{\mathrm{T}}.$$

A k-th row of the maxtrix Λ corresponds to the k-th equality constraint of type 1.10: $\mathbf{s}(u_{i_k}) = \mathbf{s}(u_{j_k})$, $k = 1, 2, \dots, K$. We form an $N \times K$ matrix $\mathbf{W}$ with columns made of vectors $\mathbf{w}^k$

$$\mathbf{W} = \left[\mathbf{w}^1 | \mathbf{w}^2 | \dots | \mathbf{w}^K\right],$$

where $\mathbf{w}^k$ represents a k-th constraint and is a sparse vector with the following non-trivial elements $w^k_{i_k} = -w^k_{j_k} = 1$ for each $k = 1, 2, \dots, K$. The following system of Euler-Lagrange equations corresponds to a discrete version of the resulting extended system of snake force balance with constraints

$$\begin{aligned} \mathbf{A}X + \mathbf{F}(X) + \mathbf{W}\Lambda &= \mathbf{0}, \\ \mathbf{W}^{\mathrm{T}}X &= \mathbf{0}. \end{aligned}$$

The resulting system can be rewritten in a simpler form

$$\tilde{\mathbf{A}}Y + \tilde{\mathbf{F}}(Y) = \mathbf{0}, \tag{1.11}$$

by introducing the following $(N+K)\times(N+K)$ matrix $\tilde{\mathbf{A}}$, an unknown matrix-variable Y and a new matrix of external force values $\tilde{\mathbf{F}}$ with $(N+K)\times 2$ components

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} X \\ \Lambda \end{bmatrix}, \quad \tilde{\mathbf{F}}(Y) = \begin{bmatrix} \mathbf{F}(Y|_X) \\ \mathbf{0} \end{bmatrix}.$$

The system 1.11 can be solved by an implicit Euler in the same way as was done for Equation 1.5

$$(\tilde{\mathbf{A}} + \gamma\tilde{\mathbf{I}})Y_{t+1} = \gamma Y_t - \tilde{\mathbf{F}}(Y_t). \tag{1.12}$$

In the original variables it has the following form

$$\begin{bmatrix} \mathbf{A}+\gamma\mathbf{I} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \gamma\mathbf{I}^K \end{bmatrix} \begin{bmatrix} X_{t+1} \\ \Lambda_{t+1} \end{bmatrix} = \gamma \begin{bmatrix} X_t \\ \Lambda_t \end{bmatrix} - \begin{bmatrix} \mathbf{F}(X_t) \\ \mathbf{0} \end{bmatrix}, \tag{1.13}$$

or

$$\begin{aligned} (\mathbf{A}+\gamma\mathbf{I})X_{t+1} + \mathbf{W}\Lambda_{t+1} &= \gamma X_t - \mathbf{F}(X_t), \\ \mathbf{W}^{\mathrm{T}}X_{t+1} + \gamma\Lambda_{t+1} &= \gamma\Lambda_t. \end{aligned}$$

To be able to iterate, we replace Λ_{t+1} with Λ_t in the first equation, so that the iteration scheme becomes

$$(\mathbf{A}+\gamma\mathbf{I})X_{t+1} + \mathbf{W}\Lambda_t = \gamma X_t - \mathbf{F}(X_t), \tag{1.14}$$

$$\mathbf{W}^{\mathrm{T}}X_{t+1} + \gamma\Lambda_{t+1} = \gamma\Lambda_t. \tag{1.15}$$

We iterate the resulting equations in the system above in turn to be able to iterate by X_t and Λ_t separately. First, we find the next snakes locations X_{t+1} by inverting the Equation 1.14

$$X_{t+1} = -(\mathbf{A}+\gamma\mathbf{I})^{-1}\mathbf{W}\Lambda_t + (\mathbf{A}+\gamma\mathbf{I})^{-1}(\gamma X_t - \mathbf{F}(X_t)),$$

and, second, we update the Lagrange multiplier Λ_{t+1} based on the previous value Λ_t and on just updated snakes locations X_{t+1} by using Equation 1.15

$$\Lambda_{t+1} = \Lambda_t - \mathbf{W}^{\mathrm{T}} X_{t+1}/\gamma.$$

Experiments showed that the equality constraints could be progressively added or deleted from the system without harming its convergence. There is no limit on the number of links; and possible clusters of links do not dangerously oscillate.

1.3.2 Three-step optimization algorithm

Our process of feature detection has the following three phases:

- initial random positioning and optimization without merging and splitting;
- optimization with simultaneous merging;
- optimization with merging and splitting.

We run snake optimization without performing merge and split operations first, because we want to apply these operations mostly to the snakes, which are already aligned with features. To make the split operation robust, we iterate during the second phase only with simultaneous merging. This creates independent bundles of snakes aligned to the features, so that in the last phase the split operation links only already optimized bundles in a network rather than spurious individual snakes. If, on the other hand, one performs the splitting with merging during the second phase, then the resulting network may have many specious connections. Figures 1.8, 1.9, 1.10, illustrates effectiveness of our three-phase approach. The final image justifies that the approach is capable of satisfying the requirements on the detection algorithm we defined earlier in this work. Indeed, the detection of curvilinear structures is close to

precise, the intersections of snakes occur mostly on the corresponding intersections of features, and the snakes are linked into a net of snakes, based on which a plane graph covering all the features can be built in a straightforward way.

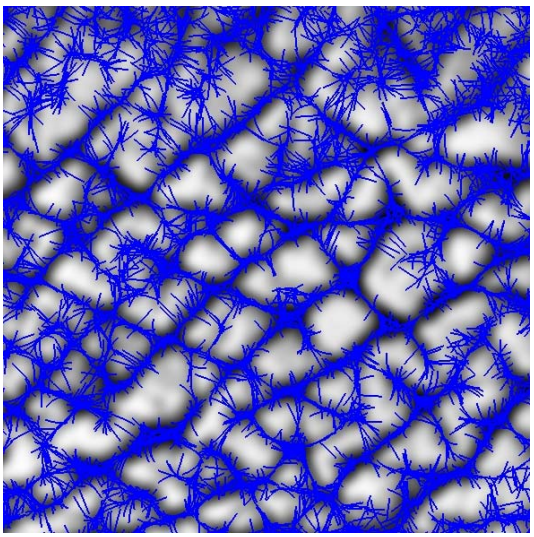


Figure 1.8: Snapshot 1: original randomly placed snakes after several iterations without merging/splitting.

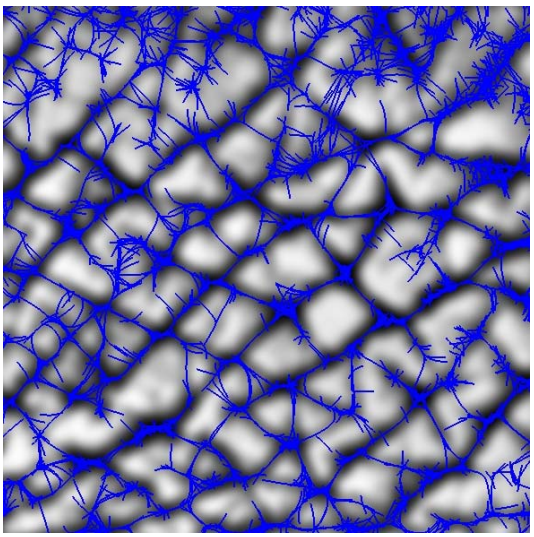


Figure 1.9: Snapshot 2: after several iterations with merging aligned snakes.

1.3.3 Discussion

Figure 1.12 shows that there is still a couple of minor problems which need to be addressed for the skin images in particular. First, the algorithm may miss detecting some enlarged furrow, especially at the furrows intersection points. At those places, a redundant number of close net nodes may occur. They can be merged during post-processing. Another minor problem is that some snakes may have unlinked tails. As the real furrows do not stop in the middle of a ridge, those unlinked snakes have to be extended to the nearest transversal snake.

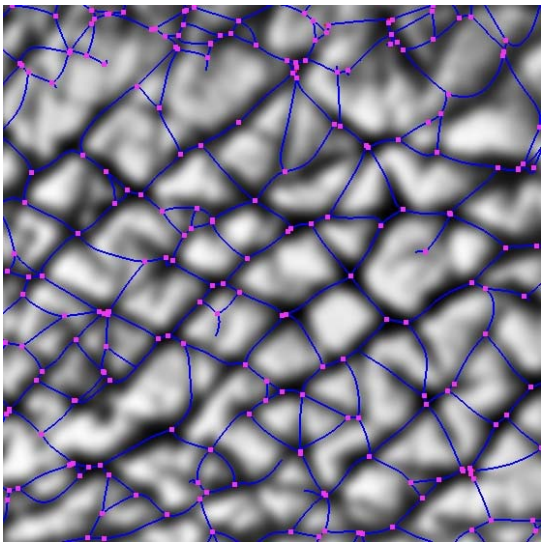


Figure 1.10: Snapshot 3: after several iterations with merging/splitting aligned snakes.

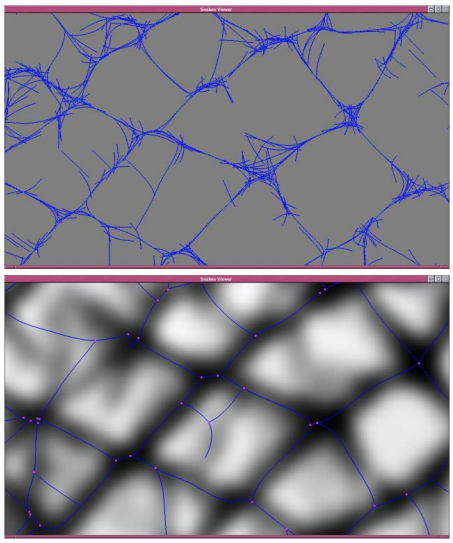


Figure 1.11: Close snapshot: result of thinning of snake bundles.

Performance of our approach still depends on how well the parameters for the snake optimization are chosen. The main difference between the original snakes and our approach is that our set of parameters does not need to be manually changed during optimization. We roughly divide all the parameters into two groups which affect the performance of the method at different optimization stages. The first group consists of the initial parameters of the snakes inherited from the original model [136]. We consider the following three most important parameters: (1) a distribution for the initial lengths of the snakes, (2) the starting number of the snakes, and (3) the snakes initial positions. The second group of parameters control optimization phases.

Preliminary analysis of the curvilinear structures in the target image (or on similar images) can help to select optimal values for parameters from the first group. For example in the case of furrows image, before running snake-based optimization one

can apply a standard edge detection technique to find candidate locations of furrows in the images — *primary snakes* — as a first approximation. The primarily detected furrows let us approximate the average distance between the neighboring valleys. Such a distance is a good estimate for the initial length of the snakes. The starting number of snakes should be proportional to the ratio of the total average length of primary snakes over the average initial snakes length — in this way each furrow gets a sufficient number of snakes to cover it and to link with the surrounding aligned snakes. We use locations of the primary snakes for initial locations of the snakes' middle points. The plateaux on the skin surface are most likely to be detected as areas free of features. We do not place the initial snakes in these areas.

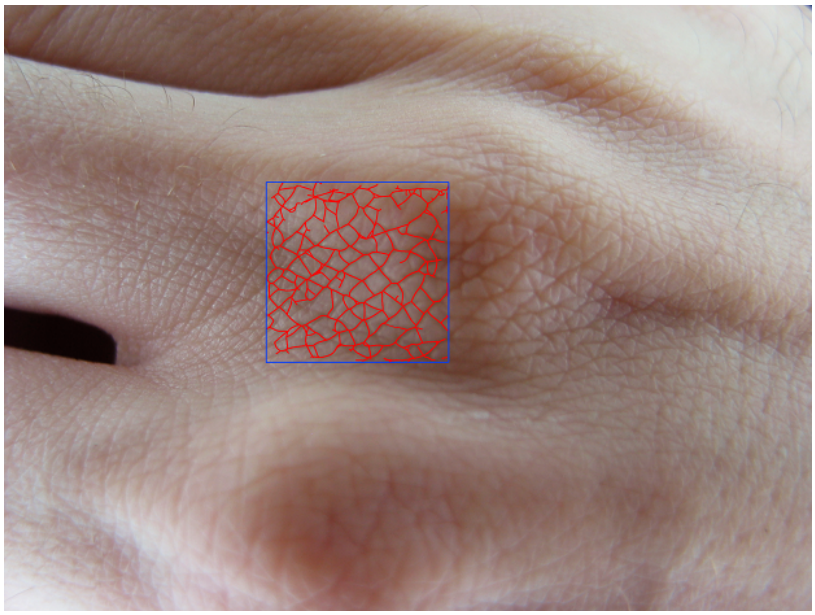


Figure 1.12: Resulting furrow network recovered from a skin image.

Multi-scale snake optimization could significantly improve snake optimization and partially solve the problem of initial snakes lengths. The idea is to optimize several sets of random snakes in turn by gradually increasing initial lengths starting

from the one with a reasonable value. Finally, based on the performances of different scales, one chooses the scale with the best detection results. In addition, the detected features from the previous scale could be used as primary snakes for the next scale.

The second group of parameters control merge/split operations. Some of the such parameters influence the way to select and determine a pair of snakes to be in one of the chain configurations. The rest determine switch times between the consecutive phases of our three-phase algorithm

The straightforward way of testing each pair of snakes to obey one of the chain configuration is extremely expensive. Also, deterministic method of choosing a snake to be absorbed by the other snake from the chain configuration should be developed. It should be based how well a new, averaged, snake configuration is tight with the close feature. We adapt the idea of decomposing the texture domain into a collection of uniform cells, so that some snakes can be accessible (referenced) from the cells and can attribute their tangential information with the cells.

We require each cell to have only one reference to a snake. The snakes which overlap a cell are tested to be aligned with the snake referenced in the cell by comparing their tangents. Those snakes which pass such test undergo merging or splitting with the cell snake. A fragment of the image decomposition with an example of referencing the snakes from the cells is illustrated in Figure 1.13.

To decide which snake should occupy a cell, we introduce the *importance levels* associated with the nodes of all snakes. The idea is to assign larger importance values to those nodes, which are statistically more significant during the optimization. In our implementation, the significance level of a snake node is measured by the number of snakes merged to the snake at the node so far. The following argument supports such a choice of the node importance measure. The more snakes a given snake absorbs at a given node, the higher is probability that it is aligned with a feature.

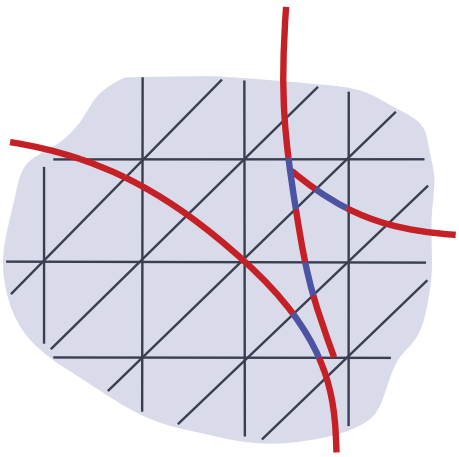


Figure 1.13: Referencing snakes from the cells. In this picture, a snake is referenced by a cell if the its part which overlaps with the cell is colored in red.

When two snakes (or their interiors) are merged, the importance level values of the survived snake are increased by the corresponding values of the nodes of the absorbed snake. Then, the importance level values of all the nodes of the survived snake are compared with that of the corresponding nodes currently referenced from the cells which are overlapped by the survived snake. The cells with smaller importance level values are updated to reference to the survived snake.

In addition, we can improve the performance of the optimization during the snake selection step by removing snakes with insignificantly small average importance level values.

From our experience, our selection method of the pairs of snakes does not affect much the quality of feature detection as compared to a detection run with exhaustive testing of all possible node pairs. Our choice of the importance level assignment may not be the optimal — one can explore different approaches in assigning significance values to the snakes nodes.

Currently, the timing for each stage of the algorithm is user-specified. An heuristic is needed to choose the switching times automatically. Such heuristic measures the readiness of the snakes to be iterated on a higher level. For example, one can measure the change of the average speed of the snakes, so that once this value passes some threshold it triggers the algorithm to proceed to the next stage. Before switching to the third stage, on the other hand, one should guarantee that the current snake bundles already cover all the features.

It is also worth mentioning that the original image should be blurred before applying snake-based algorithm to remove speckle noise and to smooth the feature gradient values.

1.4 Summary

We have described a three-stage algorithm which robustly detects curvilinear structures from the still images by optimizing a set of topology-adaptive snakes. This is a bottom-up detection algorithm which starts with optimizing a large number of randomly placed short snakes and progressively merges nearby snakes aligned to the image features into larger snakes. Such strategy allows avoiding laborious initial positioning of the snakes and frequent guiding those snakes which were stuck in spurious local minima. At the locations where more than two curvilinear structures meet at a point, misaligned snakes are linked to each other. This creates a graph structure with snakes as its edges and the linking locations as its nodes. Such a plane graph is very useful for a local feature correlation analysis.

Chapter 2

Real-time rendering of feature curves

2.1 Introduction

Texture mapping is a powerful tool for adding high-resolution detail to the material properties and geometry of models. However, the regular sampling pattern used in textures leads to a range of aliasing problems, particularly for curved one-dimensional features. Such features are common in textures and include fine-scale geometry, such as gratings, creases, scratches, cracks and seams, shadow maps and vector glyphs. For example, two main types of artifacts may be observed for normal maps: staircasing artifacts due to misalignment of linear features with the texture sample grid, and lighting artifacts due to use of bilinear interpolation in areas of discontinuity (Figure 2.1).

Recent feature-based texture representations [116, 138, 124, 133] improve texture appearance by explicitly representing discontinuities. This approach eliminates or reduces many types of artifacts, and focuses on a unique way of representing and real-time rendering of curvilinear features.

In this paper we describe algorithms for textures with *feature curves*, extending

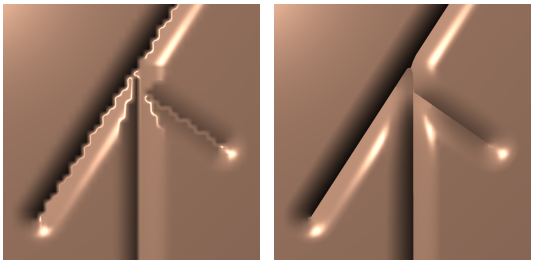


Figure 2.1: Left: Typical bump and normal map artifacts. Note the bright lines at the bottom of the crease: these lines are due to the interpolation between normals pointing in opposite directions. Right: the enhanced map using our technique. The resolution of the map in both cases is the same; the visible part is 40x40 pixels.

previously proposed feature-based texturing techniques. The distinctive elements of our approach to representing features are: a) features not only represent the object boundaries but also can be arbitrary functions of the distance and direction to curvilinear discontinuity, and b) curved features are represented by quadratic Bezier segments, rather then approximated with line segments, which is essential for resolution independence.

Another central element of our construction is an approximation of the *unsigned* distance function to the feature curve set and its gradient: the distance function vanishes *precisely* along the feature curves, is smooth away from feature curves, and is efficiently represented by a pair of auxiliary textures. Using unsigned distance functions is crucial for representing open feature curves. We present a GPU-based algorithm for reconstructing the distance map and its gradient from these textures in real time.

To avoid input restrictions, we developed a robust, incremental preprocessing algorithm, which converts complex configurations with multiple connected features

into a texture representation that can be handled by our rendering algorithm simplifying complex joints as necessary. As a result, our system imposes few restrictions on the input set of feature curves, although certain configurations may cause artifacts, as discussed in Section 2.9. The preprocessing algorithm is interactive if linear rather than quadratic curves are used.

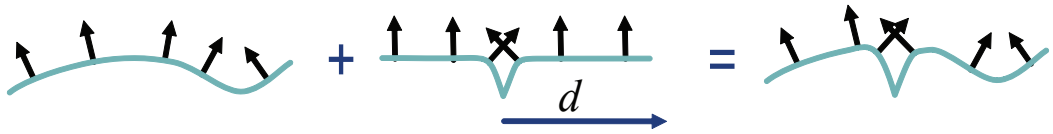
Our approach combines elements of texture-based and procedural geometry representation: the user can specify a profile for the immediate neighborhood of a sharp feature as a function of the distance to the feature. Through using user-controlled profiles, we can produce a variety of behaviors without dramatic increase in texture size.

In this paper, we focus on normal maps, for which these extensions are most relevant. However, our technique can be useful for any piecewise-smooth texture, for which it is desirable to introduce different types of feature profiles as a function of the distance to the curve (e.g., color variation as the distance to the veins on tree leaves or crease profile variation in displacement maps).

We build our normal fields at rendering time by blending two parts, continuous and discontinuous (which is illustrated in Figure 2.2). The continuous part is obtained by the standard texture interpolation of normal map. The discontinuous normal field is computed from the curve network and a given feature profile *only*:the original normal map is not used. Our focus is on construction of the discontinuous part of the normal field from feature curves.

2.2 Related work

Images and textures with resolution-independent features. The idea of explicit representation of discontinuities for data sampled on regular or unstructured grids has



$$\mathbf{n} = b(d)\mathbf{N}_{cont} + (1 - b(d))\mathbf{n}_{discont}(d)$$

Figure 2.2: Our model for sharp normal maps: continuous normal map (on the left) is blended with a procedurally defined discontinuous normals (in the middle).

appeared in computer graphics in many different contexts (e.g., discontinuity meshing for radiosity [60] to nonphotorealistic rendering). We focus only on the most closely related work. [122] presented an illustration reproduction approach, which keeps discontinuous regions sharp at any scale. The reconstruction algorithm starts from an arbitrary interpolation kernel and re-weights its intensity values according to the closeness to the sample. As the algorithm is based on finding shortest paths, it is difficult to adapt it for interactive applications.

Our method is based on feature-based textures work [116], bixels [138] and silhouette maps [124]. [10, 116] have developed a general framework for representing sharp features in textures, called feature-based textures. Feature boundaries are represented by Bezier segments, and texels may store complex intersections of boundaries. Only values in the same continuous region are used for texture interpolation. Our method can be regarded as an extension of this approach with distance functions and an adaptation of it to hardware implementation by conversion of the input curve networks to simplified form.

[138] encode discontinuities using *bixels*, i.e., pixels with additional annotation, which define texture discontinuity segments for every pixel, allowing only a restricted set of combinations of segments. Our feature maps are similar in spirit, but they

encode the distance field and are adapted to interactive rendering.

Sen's silhouette maps [124], extending [125], are similar to feature-based textures and bixels, but the interpolation approach used in this work is further simplified such that it maps well to graphics hardware. As in [138], a finite number of discontinuity boundary configurations are used for each texel. The discontinuity representation is restricted only to straight segments.

A crucial element of our technique is a distance function representation which vanishes exactly on the feature curves. Adaptively sampled distance fields (ADF) [49] is the most closely related technique, converting 2D (or 3D) object boundaries into a *signed* distance field and storing distance samples in a quadtree or octree. The boundaries of objects are accurately represented by using a polynomial distance field approximation inside cells containing object boundaries [45, 106]. Using signed distance fields makes it possible to avoid the problem of representing nonsmooth functions as the signed distance is smooth across boundaries. However, signed distance functions are only possible for objects with well-defined interiors; i.e., no open curves can be used. Using level sets of a smooth function for approximating feature curves also makes it more difficult to represent joins of several curves (e.g. Figure 2.1, right). Unlike ADF, we use parametric curves to represent features inside cells, making it possible to include curves with endpoints and curve intersections in a direct way. In addition, this considerably simplifies ensuring C^1 continuity across cell boundaries and conversion from commonly used vector graphics formats.

Adaptive refinement is a crucial feature of ADF, which makes it possible to represent 3D objects efficiently. Unfortunately, adaptivity is difficult to implement efficiently on GPUs, so we focus on uniformly sampled representations. Thus, rather than using adaptive refinement to resolve the topology of the curves as is done in ADF constructions, we use preprocessing to simplify the curve network so that it can

be represented by a fixed number of curves per cell of a texture.

Real-time curvilinear discontinuities. Textures with curvilinear features look more appealing than their analogues represented by the networks of linear segments. Direct representation of curvilinear features is essential when close-up rendering of textures is needed. Among the already mentioned works, the sharp strokes [122], feature-based textures [10, 116], and bixels [138] accept curves as their discontinuity features. However, GPU adaptation of the latter algorithms is relatively difficult.

Tarini and Cignoni introduced *the pinchmap* [133], with which smooth textures with curvilinear discontinuities are interpolated in real-time without artifacts. The algorithm accepts arbitrary curves and is very efficient, but the feature curves cannot have intersections.

In the GPU algorithm [80], objects with boundaries in the form of quadratic and cubic spline curves are sharply rendered with infinite resolution. In our approach we also use quadratic splines for the discontinuous features. However, their feature representation is not suitable for interpolating distance function and gradient field around the features, essential for making discontinuity customizable as is possible for our feature curves.

Our feature curves are comparable to the concurrent work of Qin *et al.* [115] on rendering font glyphs anti-aliased at arbitrary magnification. Similar to our technique, their approach is based on exact computation of a distance field to glyph boundaries approximated by line segments. The crucial difference is that we rasterize the distance field and its gradient during preprocessing (local approach), while [115] looks up the closest feature and calculates the distance directly (global approach). This approach is difficult to generalize to resolution-independent curvy features and to handle multiple features meeting at a point.

More recently, closely related problems were addressed in papers by Nehab and

Hoppe [94], and Qin and others [114]. Nehab and Hoppe describe an approximate distance function calculation for points sufficiently close to a quadratic curve. The discontinuities with quadratic outlines are shared between the variable-length cells and may be of arbitrary complexity. Their approach is ideal for rendering vector graphics defined by filled or stroked shapes. Another calculation is described in Qin et al who approximate the distance near the curve numerically by binary search. We believe our method is somewhat more efficient and most importantly, it yields a continuous distance function and gradient for arbitrary distances with no limitations on curvature of the curves. See comparison of distance fields calculated by our technique versus the Nehab's and Qin's techniques on Figure 2.3.

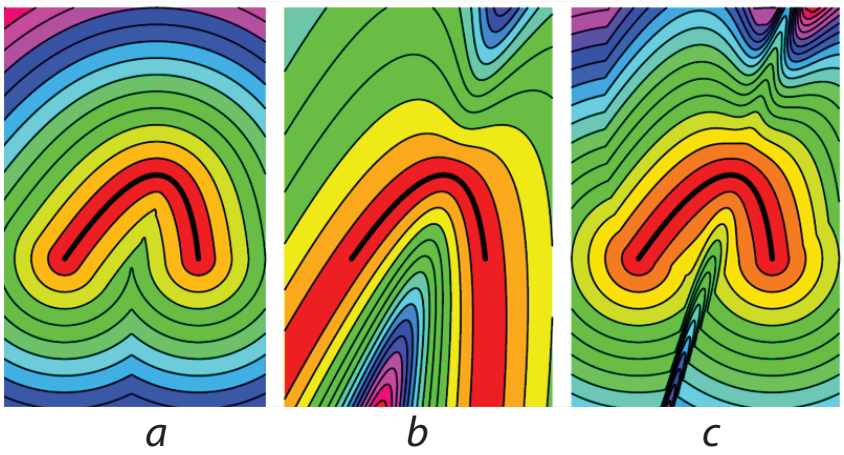


Figure 2.3: Approximation of distance function by our technique (a), Qin et al. technique (b), and Hoppe and Nehab technique (c). Figures are taken from [94].

Textures representing small-scale geometry. A variety of techniques were developed for representing fine-scale geometric detail with textures. Bump maps were invented by J. Blinn [18]. Many techniques for interactive rendering of bump maps were proposed,(e.g., [134] and references). With the appearance of programmable

graphics hardware, use of basic bump mapping became commonplace. Bump map appearance can be improved by horizon maps, a technique for self-shadowing of bump maps, introduced in [82, 83]. [129] described how horizon maps can be accelerated using hardware. Parallax mapping and steep parallax mapping [84] aim to reduce another artifact of bump maps: incorrect behavior when a surface is tilted. A more consistent way to address this problem is by using relief maps [99]. In many methods for high-quality rendering of normal maps, one needs to make a transition between different rendering modes as in [15]. Enhancements to lighting of bump maps necessary for such transitions are described in [62]. Displacement maps, introduced in [26], and relief maps [99] are a more advanced form of representing fine-scale geometry for flat surfaces (extensions to arbitrary surfaces were presented in [107]). Distance maps [35] significantly improve rendering quality of small details by making use of volumetric distance field to the closest surface features while maintaining real-time performance of displacement maps.

While techniques for interactive rendering of displacement maps were recently proposed [143], [144], these require large precomputed data sets. Interactive rendering of relief maps requires less data. While we describe our technique in the context of normal maps, it can be applied to rendering of relief maps representing the same type of geometry.

Texture and detail synthesis. As the application of our technique results in a combination of sampled and procedurally defined normals, our work is related to work in texture synthesis. Procedural bump maps first appeared in [104]. There were a large number of papers on non-parametric synthesis from examples; these techniques often can be applied to produce bump maps, e.g., [156]. Our technique addresses a specific case of the problem of adding high-resolution detail to an image, focusing on sharp curvilinear linear features. A general approach to this problem can

be found in [67].

To summarize, we believe our method to be the only real-time technique combining the following features: a) C^1 sharp feature curves at arbitrary resolution, b) open and intersecting feature curves, and c) smooth (away from feature curves) distance function and distance function gradient, enabling user-defined distance-based feature profiles. For linear features, our method is fully interactive.

2.3 Background: GPU, shaders, framebuffers

Modern GPUs (graphics processing units - highly parallel devices optimized for rich and fast graphical output) underwent a rebirth after they had evolved from a highly tuned fixed-function logic to become programmable. Fixed-function graphics pipeline supports only a fixed set of operations (slightly varying among different manufacturers): a user cannot change the order of operations but can only change the parameters of the operations and/or enable/disable some of them. A diagram of a typical rendering pipeline with a fixed-function logic is illustrated in Figure 2.4 (left) with boxes filled in blue. A user passes a set of geometric primitives which are (optionally) transformed within the world space coordinate system. At this stage camera position and orientation as well as scene lighting is also assigned. During a rasterization step, all the primitives are projected to the camera visible frustum and mapped into the pixel locations. At this stage all the parameters of the primitives are interpolated down to the pixels and those primitives with smaller depth values are shaded and stored in the pixel framebuffer.

In modern GPUs portions of graphics pipeline are programmable which gives a user additional control during the rendering stages (some portions thought are better to keep fixed for performance considerations; currently, they include rasterization,

texture filtering, post-shader α-blending and depth testing). At the current state, three programmable components are incorporated into the graphics pipeline (depicted by yellow boxes in Figure 2.4, left). *Vertex shader* [78] allows manipulating the locations of the primitive vertices and calculating lighting at the vertices before passing the vertex data farther down the graphics pipeline. Vertex shaders usually consume a minor part of the total rendering time which makes them ideal means for simple real-time animation, for simulating one-reflection surface lighting effects, for fine-scale geometry manipulation including bump-mapping, displacement mapping etc. *Geometry shader*, unlike the vertex shaders which operate on vertices alone, have an access to the entire primitive and adjacent information which allows efficient transforming such primitives before rasterization stage. *Pixel shaders* execute on each pixel and store the color components and depth information into designated frame-buffers. A user can implement fairly sophisticated shading algorithms based on the pixel shader's input which usually includes but not restricted to (u,v)-coordinates of the pixel within a set of (also provided) textures, the texture samplers, tangent basis at the geometry point corresponding to the pixel sample, lighting descriptors, interpolated per-vertex values.

Our method of sharp rendering of feature curves is implemented as a pixel shader. At each pixel (blue square in Figure 2.4, right) we reconstruct discontinuity curve (black curve in Figure 2.4, right), and fetch the values of distance and distance gradient encoded in the four texels which surround the pixel (green squares in Figure 2.4 (right) labeled T_{ij}, T_{ij+1}, etc). Based on this information and the pixel coordinates within the textures, we interpolate the values of the distance and distance gradient addressing the discontinuity and calculate the discontinuous part of the normal map which will be associated with the pixel. Continuous part of the normal map is calculated by a standard bi-linear interpolation within a pixel shader. The continuous

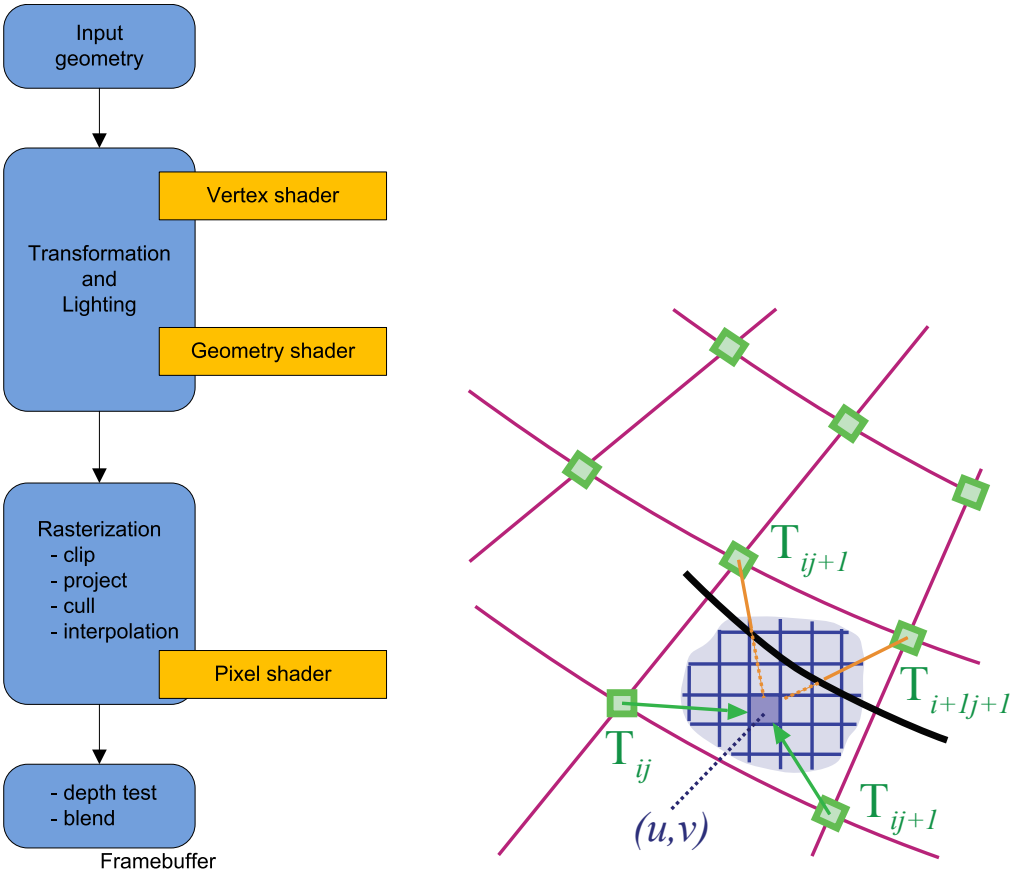


Figure 2.4: Left: Standard fixed-function logic graphics pipeline (blue boxes) with embedded programmable parts (yellow boxes). Right: Texture samples (green squares) interpolation within a pixel (a blue filled square) addressing the presence of discontinuity curve (black) defined in texture space.

and discontinuous normal components are blended to produce a final normal which will be used during a shading algorithm at the final portion of the pixel shader. We should mention that in this work we only concentrate on interpolation aspects limited to the interior of one texel. Possible under-sampling artifacts happening when pixels cover more than couple of texels and anti-aliasing along the sharp boundaries are not

addressed in this work.

Pixels shaders follow standards - shader models - which define set of operations and features a particular GPU should support. Shader models have evolved significantly since its introduction in 2001 (after launch of GeForce 3 series, DirectX 8.0). With earlier shader models (< 3.0) an implementation of the method described in this work would not be possible as it requires support for dynamic branching and takes more than 96 shader instructions. Our first implementation of a discontinuous normal map pixel shader was under shader model 3.0 which supports both half and single-precision floating numbers. To speed up the shader performance we tuned our program to half-precision which triggered some robustness issues of original geometric routines and required to make some sacrifices to the quality. The latest shader model replaces half-s by single-precision floating numbers by default, so the quality of rendering of normal maps does not suffer anymore due to lack of precision.

We take advantage of framebuffer objects (available since introducing shader model 2.0) by storing discontinuous normal map, produced by our pixel shader, into a designated texture attached to the shader. We render a geometry with normal map texture by running three passes. Pixel shaders of the first two passes calculate continuous and discontinuous parts of normal maps and store them to a framebuffer object. On the third pass, the calculated normals are provided to its pixel shader as a regular texture after being de-attached from the framebuffer object. This separates sharp normal calculations from the final shading. Calculation of the discontinuous part of the normal map is the most time consuming operation. To avoid invoking the corresponding pixel shader during the second pass, we modify the depth values for the pixels not overlapped with discontinuity (which account for majority of all rendered pixels) with the value zero while other pixels are assigned with the largest depth value.

2.4 Overview of algorithm

The input to our algorithm consists of a set of feature curves representing desired sharp features or discontinuities in a target texture, a continuous texture map to which feature curves are added, and, optionally, a one-dimensional profile defining how the texture is modified near the feature curves. Discontinuity representation overhead, overall shader performance, and possibility of enabling interactivity to our algorithm depends on what type of features is selected. We distinguish linear features, the ones which follow line segments, from curvilinear features, which may also follow curves with nontrivial curvatures, and provide separate interpolation algorithms, memory representation and individual perofrmance analyses for each type.

Definition 1 (Feature)**.** *A feature is a distinctive object in the texture which domain is concentrated along a chain of connected curves and line segments and defined by its profile around the chain. Depending on the context, we also distinguish linear features from curvy features to emphasize the differences in approaches applied to them.*

Definition 2 (Feature curve)**.** *A Curve which defines the position of texture discontinuity within a feature is called feature curve. Feature curves include line segments.*

Feature curves are represented by either line segments $L[\boldsymbol{b}_0, \boldsymbol{b}_2]$, with parametrization $\gamma_L(t)$ given by equation 2.1, or by nondegenerate (i.e., no two control points coincide) quadratic Bezier segments $B[\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2]$, parameterized by $\gamma_B(t)$ as in equation 2.2. No further restrictions are imposed on the input.

$$\gamma_L(t) = (1-t)\,\boldsymbol{b}_0 + t\,\boldsymbol{b}_2\,, \tag{2.1}$$

$$\gamma_B(t) = (1-t)^2\,\boldsymbol{b}_0 + 2(1-t)t\,\boldsymbol{b}_1 + t^2\,\boldsymbol{b}_2\,, \tag{2.2}$$

$$t \in [0,1]\,.$$

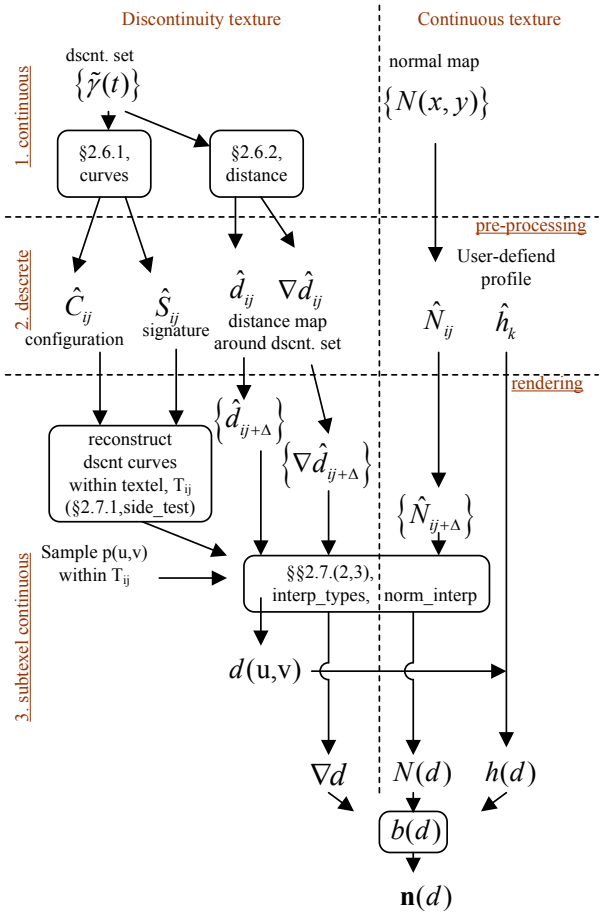


Figure 2.5: Block-diagram of our method applied to normal map $N(u,v)$: (1) continuous input, (2)pre-processing step, (3) rendering step.

In the rest of the algorithm description, we focus on normal maps, although the algorithm with minor modifications can be applied to other types of textures with features depending on a distance map $d(u,v)$ and, if necessary, on distance gradient field $\nabla d(u,v)$.

Our technique has two principal and relatively independent components. The first runs as a pixel shader which computes the normal using an approximation of the dis-

tance to the feature curve $d(u,v)$ and its gradient $\nabla d(u,v)$, and a feature cross-section profile $h(d)$ (this part is illustrate at the bottom of the block-diagram in Figure 2.5). The result is blended with the original normal map.

The second component is a preprocessing algorithm that produces textures $\hat{C}_{ij}$, $\hat{S}_{ij}$, $\hat{d}_{ij}$ and $\nabla\hat{d}_{ij}$ which encode information about feature curves. Preprocessing enables us to use a curve network as input, while using only a restricted type in the shader (this is illustrated in the top and middle part of Figure 2.5). The central element of our approach is computing a continuous unsigned distance function and its gradient from the feature curve information.

At the *preprocessing* stage, we convert the curves and line segments to a texture-based representation (Section 2.6), and create additional textures representing the distance field, $\hat{d}_{ij}$, and its gradient $\nabla\hat{d}_{ij}$ (Sections 2.5.1 and 2.6). The feature curves are split into texel-sized discontinuity segments, defined by *discontinuity descriptors*. The descriptors are stored in two textures of discontinuity configurations $\hat{C}_{ij}$ and discontinuity signatures $\hat{S}_{ij}$.

At the *rendering* stage the original texture, profile, and additional textures generated by preprocessing step are used to compute the feature texture values at pixel locations.

The combination of preprocessing and real-time algorithms aims to approximate, as closely as possible, a desired piecewise-smooth function (color, normal direction, or any other quantity encoded in the target texture) defined by the combination of the feature curves, profiles, and a smooth map interpolating texture values (details are shown in Figure 2.5).

We start with a precise definition of the normal field we aim to approximate. Suppose we are given a continuous normal map, encoded in a texture $\mathbf{N}(u,v)$: $[0..U_{max}, 0..V_{max}] \rightarrow R^3$. We split the map $\mathbf{N} = (N_x, N_y, N_z)$ into two components

$\mathbf{N} = [\mathbf{N}_{xy}, N_z]$, where $\mathbf{N}_{xy}(u,v)$ is the 2D projection of the normal to the object's tangential plane at a point $p(u,v)$. Only $\mathbf{N}_{xy}(u,v)$ needs to be represented explicitly.

Let the distance to the closest feature curve from $p(u,v)$ be $d(u,v)$ and let the gradient of $d(u,v)$ be $\nabla d(u,v)$. Let $h(d)$ be the user-specified profile defining displacement of the fine-scale surface from the coarse geometry to which the texture is applied. In the simplest case, it depends only on the distance to the feature set d but may also depend on the closest point on the feature curve and other parameters.

We use a blending function $b(d)$ to merge given normals with the normals derived from the crease profile $h(d)$. The process of obtaining a new normal, given by $\boldsymbol{n}(d) = (1-b(d))\,\mathbf{n}^h(d) + b(d)\mathbf{N}(d)$, is illustrated in Figure 2.6, where $\mathbf{n}^h$ is a normal of the crease profile pointing toward the feature curve. In the formulas, we omit the dependence of d on (u,v) where it is clear. We choose $b(d)$ to satisfy $b(d) = 0$ for $d \leq w_0$ and $b(d) = 1$ for $d > w$, where w_0 is the half-width of a band around the feature curve, for which the original normal map has no effect (w_0 can also be zero), and w is the feature curve width.

Assuming that the projection of profile normal $\mathbf{n}^h$ is aligned with the distance gradient $\nabla d(u,v)$, the (non-normalized) desired normal map $\mathbf{n}(u,v)$ continuous everywhere but along $d(u,v) = 0$ can be calculated as follows

$$\begin{aligned} \mathbf{n}_{xy}(u,v) &= -(1-b(d))h'(d)\nabla d + \tilde{b}(d)\mathbf{N}_{xy}(u,v) \\ n_z(u,v) &= (1-b(d)) + \tilde{b}(d)N_z(u,v) \end{aligned} \tag{2.3}$$

where $\tilde{b}(d) = b(d)\sqrt{h'(d)^2 \cdot \nabla^2 d + 1}$

Our algorithms are designed to approximate the function defined by (2.3), maintaining precise behavior at the feature curves. For the rest of the paper we assume working in the texture domain $[0..U_{max}] \times [0..V_{max}]$, unless a different domain is specified. The domain is split into texels $T_{ij} = \Delta U\,[j, j+1) \times \Delta V\,[i, i+1)$ whose bottom-

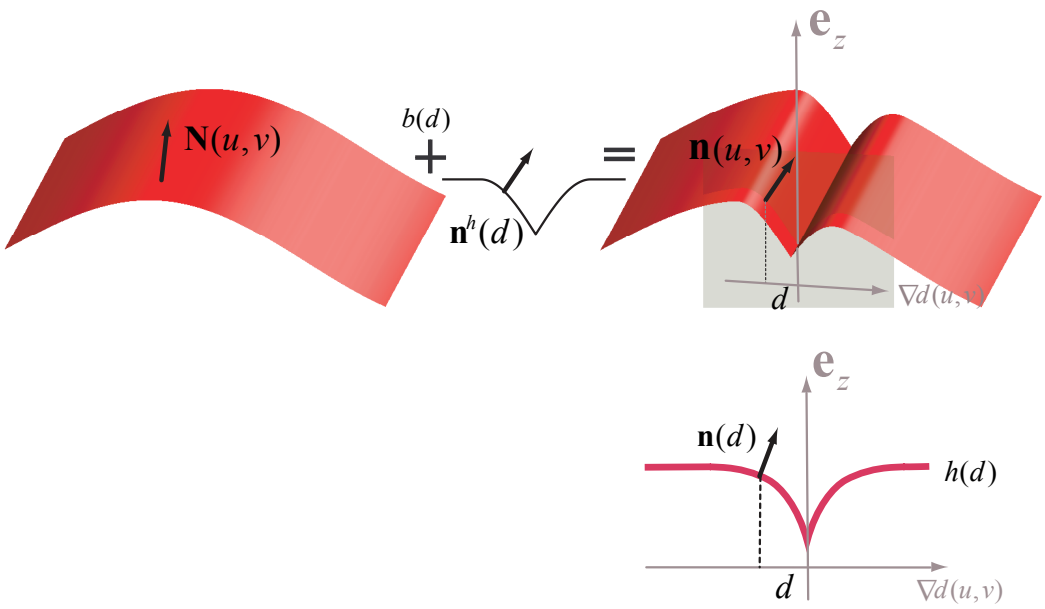


Figure 2.6: Interpolation between the global smooth normal map $\mathbf{N}(u,v)$ and the discontinuous normal map $\mathbf{n}^h$ defined by local profile $h(d)$. We show updating height maps for clarity, but we interpolate normals, not the underlying mesh.

left corners define a grid $\Omega = \{(u_j, v_i), i = 1,2,\ldots,M_v, j = 1,2,\ldots,M_u\}$. We use $\mathbf{p}_t = (u,v)$ to denote a sample in the texture domain. Texels form a partition of a given texture domain if we exclude the north and east boundaries from each texel, except the texels from the most right column and top row where the east and north texel boundaries are kept, respectively.

We also assume that underlying mesh has a nice parametrization, so that texture mapping does not distort noticeably the feature curves.

2.5 Distance field to features

An *unsigned distance map* $d_S(\boldsymbol{p})$ around a nonempty closed set S is defined as the distance from a given point $\boldsymbol{p}$ to a closest point on S

$$d_S(\boldsymbol{p}) = \min_{\boldsymbol{s}\in S} \|\boldsymbol{p}-\boldsymbol{s}\| . \tag{2.4}$$

In general, there may be more than one point from S equidistant to a given point. All such points form a medial axis M_S to a set S

$$M_S = \left\{ \boldsymbol{p} \notin S \mid \exists \boldsymbol{s}_1, \boldsymbol{s}_2 \in S \ , \|\boldsymbol{p}-\boldsymbol{s}_1\| = \|\boldsymbol{p}-\boldsymbol{s}_2\| = d_S(\boldsymbol{p}) \right\} .$$

A *gradient of the distance function* is a unit vector[1] $\nabla d_S(p)$ which is directed toward the closest point on S

$$1)\ \|\nabla d_S(p)\| = 1 ,$$
$$2)\ \nabla d_S(p) \cdot (\boldsymbol{p}-\boldsymbol{s}_p) = d_S(p) .$$

In this work we focus on texture features which are defined as a function of the distance to the corresponding feature curves. A set of feature curves Γ consists of quadratic Bezier curves and line segments, $\Gamma = \{\gamma_k(t), k = 1,2,\ldots,K\}$, given by equations 2.2 and 2.1, correspondingly. Thus, the unsigned distance field to the feature curves Γ is a scalar function defined in the texture space, $d(u,v)$: $[0..U_{max}, 0..V_{max}] \rightarrow [0,\infty)$, and is given by

$$d(u,v) = \min_{\gamma_k\in\Gamma} d_{\gamma_k}(\boldsymbol{p} = (u,v)) . \tag{2.5}$$

[1] More rigorous definition of a distance gradient can be done by introducing the generalized gradient whose construction addresses the situations when the distance function is not differentiable [23], e.g., at the points along the medial axis of a given set. The generalized gradient is a Clarke subdifferential which consists a set of vectors. Details can be found elsewhere [38].

A gradient of the distance function, $\nabla d(u,v) = \nabla d_\Gamma(\boldsymbol{p} = (u,v))$, in general, may not be uniquely defined, e.g., at the medial axis points $\boldsymbol{p}_m \in M_\Gamma$ at least two vectors satisfy the definition of the distance gradient $\nabla d_\Gamma(\boldsymbol{p}_m)$ (see Figure 2.7 on the right, more than one vector can be considered as a distance gradient at the points on blue line segments). At every such location we just pick randomly one vector from all the vectors meeting the distance gradient requirements.

2.5.1 Distance functions

The distance map representing the unsigned distance function is central to our algorithm: it determines the locations of the discontinuity curves and the distance to these curves. The direction of the gradient of the distance function for many feature profiles heavily influences the resulting value as can be seen from Equation (2.3). To ensure rendering quality, we would like them to satisfy the following distance function requirements (DFR):

DFR$_1$ (a) the approximate distance function should be exactly zero at feature curves so that sharp features can be created; (b) it should vary continuously; (c) it should remain close to the precise distance function near feature curves in order to be able to control feature width correctly;

DFR$_2$ the gradient of the distance function should be continuous away from feature curves;

To understand the difficulties with approximating the distance function, we observe that the Euclidean distance function has two types of singularities where the gradient is discontinuous: distance zero curves and and medial axis curves, (Figure 2.7).

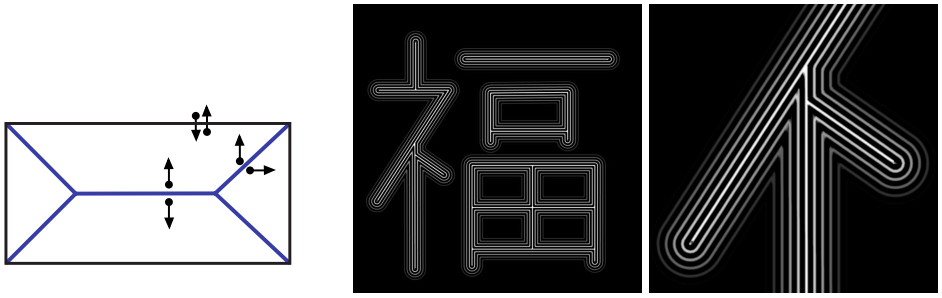


Figure 2.7: Left: Distance function singularities for a rectangle: the medial axis is depicted in blue, and vectors denote the gradient discontinuity areas. Right: Distance function level lines and close-up rendered by using our techniques.

We represent the first type of singularity explicitly. Most singularities of the second type are far from feature curves, and the distance function does not affect the result in these locations. However, at corners, the medial axis meets the feature curves, so we need to address the gradient discontinuities there.

To satisfy our requirements, we compute the approximate distance function and its gradient from texture grid samples as follows:

1. Within the texel the distance function $d(u,v)$ is linearly interpolated on curved triangular subtexel domains (Figure 2.8) aligned with feature curves to ensure that it is zero at these lines.

2. The gradient $\nabla d(u,v)$ is also interpolated linearly. Note that this is *not* equivalent to computing the gradient of the interpolated distance function, as the latter would be piecewise constant.

3. The gradient samples on the texture grid $\nabla \hat{d}_{ij} = \nabla d(u_i, v_j)$ are smoothed away from the feature lines, to eliminate discontinuities at the medial axis.

The construction of a specific representation of the distance map is discussed in detail in Section 2.6. The real-time evaluation of distances and gradients is discussed in Section 2.7.

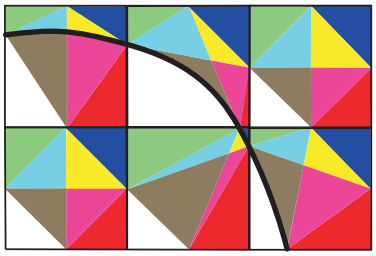


Figure 2.8: Interpolation domains (in the texture space) for the distance function near a feature curve shown in different colors.

2.5.2 Calculating distance field to feature curves

Our set of features consists of line segments and quadratic Bezier curves, given by equations 2.2 and 2.1, respectively. Computing the unsigned distance and its gradient to a line segment γ is straightforward and is decomposed into three cases, depending on a relative location of a sample point $\boldsymbol{p}$ with respect to the line segment's interior

$$d_\gamma(\boldsymbol{p}) = \begin{cases} \|\boldsymbol{p}-\boldsymbol{b}_0\|, & \boldsymbol{b}_{01}\cdot(\boldsymbol{p}-\boldsymbol{b}_0) < 0, \\ \left((\boldsymbol{p}-\boldsymbol{b}_0)^2 - |\boldsymbol{b}_{01}\cdot(\boldsymbol{p}-\boldsymbol{b}_0)|^2/\boldsymbol{b}_{01}^2\right)^{1/2}, & 0 \le \boldsymbol{b}_{01}\cdot(\boldsymbol{p}-\boldsymbol{b}_0) < l_\gamma \\ \|\boldsymbol{p}-\boldsymbol{b}_1\|, & \text{otherwise}, \end{cases}$$

where $\boldsymbol{b}_{01} = \boldsymbol{b}_1 - \boldsymbol{b}_0$ and $l_\gamma = \|\boldsymbol{b}_{01}\|$. If the (counter clock-wise) normal vector to γ is introduced, $\boldsymbol{n}_\gamma = (-b_{01,y}, b_{01,x})/\|\boldsymbol{b}_{01}\|$, then the distance gradient is given by the

following cases

$$\nabla d_\gamma(\boldsymbol{p}) = \begin{cases} \frac{1}{\|\boldsymbol{p}-\boldsymbol{b}_0\|}(\boldsymbol{p}-\boldsymbol{b}_0), & \boldsymbol{b}_{01}\cdot(\boldsymbol{p}-\boldsymbol{b}_0) < 0, \\ \left[2\chi_{\mathbb{R}_+}\{(\boldsymbol{p}-\boldsymbol{b}_0)\cdot\boldsymbol{n}_\gamma\}-1\right]\boldsymbol{n}_\gamma, & 0 \le \boldsymbol{b}_{01}\cdot(\boldsymbol{p}-\boldsymbol{b}_0) < l_\gamma \\ \frac{1}{\|\boldsymbol{p}-\boldsymbol{b}_1\|}(\boldsymbol{p}-\boldsymbol{b}_1), & \text{otherwise}, \end{cases}$$

where $\chi_{\mathbb{R}_+}(x)$ is a characteristic function of a nonnegative real numbers $\mathbb{R}_+ = [0,\infty]$. Figure 2.9-(left) illustrates three different regions (filled in yellow, blue, and green) around linear feature which correspond to the cases mentioned in the equations defining the distance and its gradient vector above. The size of the regions is defined by the width of a given feature, w_F.

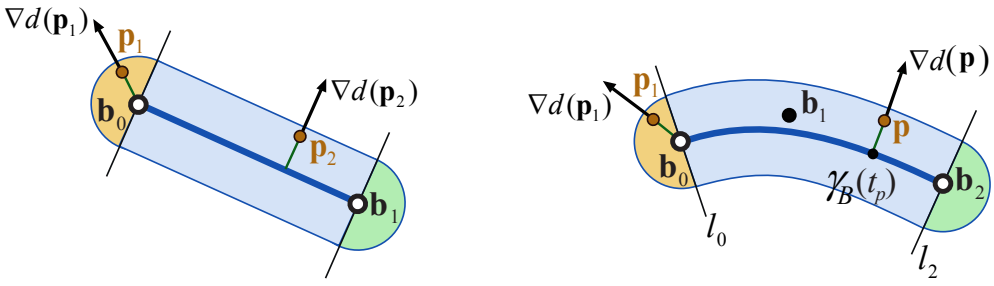


Figure 2.9: Calculation of unsigned distance to the feature line given by a line segment $L[\boldsymbol{b}_0,\boldsymbol{b}_2]$ (left) or feature curve given by a Bezier segment $B[\boldsymbol{b}_0,\boldsymbol{b}_1,\boldsymbol{b}_2]$ (right) with their gradient vectors is split into three cases depending on where a sample point is located relative to the interiors of the lines (regions corresponding to different cases are filled in different colors). Distance values are calculated only within a specified effective feature width w_F.

Computing the distance and its gradient to a quadratic Bezier segment $B[\boldsymbol{b}_0,\boldsymbol{b}_1,\boldsymbol{b}_2]$, with parametrization $\gamma_B(t) = (1-t)^2\boldsymbol{b}_0 + 2(1-t)t\boldsymbol{b}_1 + t^2\boldsymbol{b}_2$, is split into the same three cases as was done for the line segments: the samples which lie in the region on

the "left" from the line l_0 which passes the first control point $\boldsymbol{b}_0$ (filled in yellow in Figure 2.9 on the right), the region around the interior of the Bezier segment (filled in blue), and the region on the "right" from the line l_2 (filled in green) which passes the third control point $\boldsymbol{b}_2$ (lines are defined by $l_0(t) = \boldsymbol{b}_0 + \boldsymbol{n}_0 t$ and $l_2(t) = \boldsymbol{b}_2 + \boldsymbol{n}_2 t$, where $\boldsymbol{n}_k$, $k = 0, 2$, are counter clock-wise normal vectors to the Bezier tangents at $\boldsymbol{b}_k$: $\boldsymbol{n}_0 = (-b_{01,y}, b_{01,x}) / \|\boldsymbol{b}_{01}\|$ and $\boldsymbol{n}_2 = (-b_{12,y}, b_{12,x}) / \|\boldsymbol{b}_{12}\|$ where $\boldsymbol{b}_{12} = \boldsymbol{b}_2 - \boldsymbol{b}_1$). The first and the third case are of no difference from that of derived for line segments. The second case requires more effort to be solved.

So, for the second case, where sample points fallen between the lines l_0 and l_2 (region, filled in blue in Figure 2.9 on the right), we consider two possible situations depending on the magnitude of the curvature of the Bezier segment. When the curvature radius at every curve's point is sufficiently larger than the feature width w_F then the distance from a sample point $\boldsymbol{p}$ to the curve can be computed by solving the following cubic equation for a parameter t

$$\big(\boldsymbol{p} - \gamma_B(t)\big) \cdot \partial_t \gamma_B(t) = 0\,,$$

so that $d_{\gamma_B}(\boldsymbol{p}) = \big\|\boldsymbol{p} - \gamma_B(t_p)\big\|$ and $\nabla d_{\gamma_B}(\boldsymbol{p}) = (\boldsymbol{p} - \gamma_B(t_p))/d_{\gamma_B}(\boldsymbol{p})$, where t_p is a solution of the equation (such equation describes a necessary condition for a closest point $\gamma_B(t_p)$, which requires a vector spanned between a sample point $\boldsymbol{p}$ and the closest point to be orthogonal to a tangent vector at the closest point[1]). We find a root to a cubic equation by running a version of Newton-Raphson method [111, Ch.9] which converges to a solution in several iterations. Alternatively, one can use analytical methods [19] or an iterative solver described by Qin et al. [115]

[1]To prove that such condition is necessary, one can use the continuity of the normal vector to the Bezier curve at the curve's points and that for the case with small enough curvature of the Bezier there is only one line passing the sample point and being orthogonal to the interior of the Bezier.

For the case when there are points on a Bezier curve with high curvature value, we split the curve into "safe" and "unsafe" regions, $\gamma_B = \gamma_1^S \cup \cdots \cup \gamma_K^S \cup \gamma^U$, and apply a different algorithm for an unsafe part γ^U while using the same iterative algorithm described above for the safe parts, γ_k^S, $k = 1, 2, \ldots, K$. We define a portion of a Bezier to be safe if its curvature is smaller than $(1.1\, w_F)^{-1}$. For an unsafe Bezier γ^U we run a version of Newton method to find a distance to the interior of the Bezier (the region filled in blue in Figure 2.9 on the right) minimizing the following polynomial of degree four $\left(\gamma^U(t)\right)^2 - 2\gamma^U(t) \cdot \boldsymbol{p}$. For the regions filled in yellow and green in Figure 2.9 we follow the same steps as for the line feature with the only difference at points where the yellow and blue regions overlap. In such case, we calculate the distance to the first and the third control point of the unsafe Bezier and choose the minimum.

An alternative geometric approach to find distance to a parametric curve based on binary search is described in recent work of Qin et al. [114]

2.6 Feature discretization: discontinuity configuration and signature

Feature rasterization has two goals: to simplify the connectivity of the feature curves so that it can be represented using a fixed number of curvilinear segments per texel, and to ensure that in the case of curvy features the simplification process results in a network of curves with no self-intersections and sufficiently smooth curve segments. The former is achieved by a snapping process locally modifying curvilinear segments overlapping a texel. The latter requires solving a collection of local constrained optimization problems. While computationally relatively expensive, the latter step is

essential for obtaining a usable simplified curve network (see Section 2.6.1); this step is unnecessary only if line segments are considered.

For every texel overlapping a curve, we define a *discontinuity configuration* and a *signature*. The discontinuity configurations and signatures are stored as additional textures. A discontinuity configuration defines the number of discontinuity segments within the texel and which edges are crossed by the segments. The discontinuity signature stores more detailed information: exactly where the discontinuity passes through an edge and its tangential direction (in the case of curvy features).

In principle, local configurations of discontinuities can be arbitrarily complex: any number of curves can overlap a texel. Following [138] and [124], we reduce the number of allowed configurations, avoiding introduction of points in the center of texels. Valid discontinuity configurations for a single texel are defined by the following three rules:

- One boundary edge of a texel may be crossed by discontinuity curves at no more than one point.
- There are no more than two discontinuity curves inside any texel.
- A discontinuity curve ends on a texel boundary.

We call a portion of the discontinuity curve overlapped by a texel as a *discontinuity segment*. Examples of valid configurations of discontinuity segments are shown in Figure 2.10. The main difference between curvy and linear feature representations is that curves share a tangential vector at an intersection point located on the texel's edge as is shown in the top row images of Figure 2.10 (left).

Our choice of the set of valid configurations aims to achieve a good balance between generality of networks that can be represented exactly, required texture size,

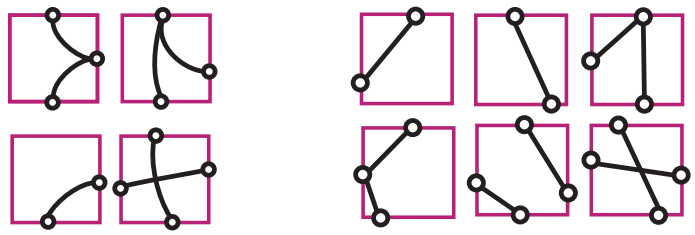


Figure 2.10: Valid discontinuity configurations for a texel. Left: curvilinear features. Right: Linear features only.

and algorithm complexity. The major limitation of our choice of valid configurations is that no more than two curves may overlap a texel, curve termination inside texels is not allowed, no more than four curves can share endpoints, and curves sharing a point on a texel boundary also have to share a tangent (see Figure 2.11).

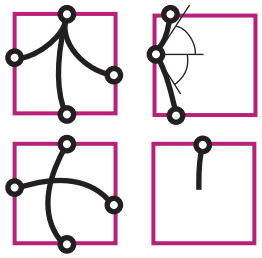


Figure 2.11: Texel prohibited discontinuity configurations.

The algorithm modifies discontinuity curves locally and transforms an arbitrary configuration into the one that satisfies our requirements.Figure 2.12 (left) illustrates the changes in the discontinuity map the algorithm performs.

Once the transformation is complete, every texel has at most two discontinuity segments, so we can store texels' configurations in the configuration map as quadruples $\hat{C}_{ij}$. Configuration C is $[0,0,0,0]$ for an empty texel, has two pairs of endpoint

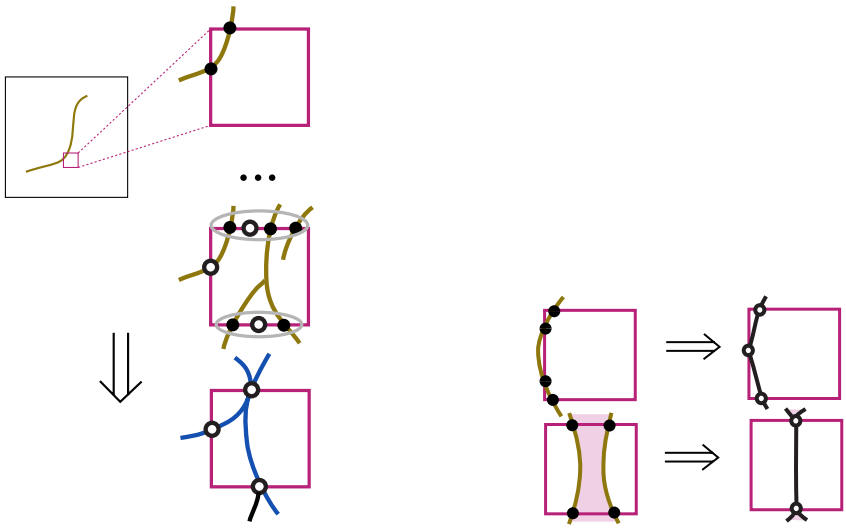


Figure 2.12: Left: Rasterization process consists of adding one by one curvy features to texels and transforming all invalid texel configurations by transforming edge points. Black points are replaced by their center of mass (white points) to satisfy the first rule, identified discontinuity segments are deleted to follow the second rule, no hanging curves inside a texel according to the third rule. Right: Transformation artifacts: (top) reduction of a curve to two linear discontinuity segments, (bottom) change of topology.

edge indices for a texel with two discontinuity segments:

$$C = [(l_1, m_1), (l_2, m_2)], \tag{2.6}$$

and two copies of endpoint edge indices of the only discontinuity segment within a texel with one discontinuity: $C = [(l_1, m_1), (l_1, m_1)]$. The edge indices l_s, m_s, $s = 1, 2$ are in the range $1 \ldots 4$ (edges are ordered clockwise starting with from the south edge).

Based on the resulting configuration map and by using the adjusted network of feature segments, we assign discontinuity signatures for all affected texels: each texel

is annotated with eight numbers representing the four distances from the left/bottom corners to discontinuity segment endpoints along the texel boundary, and four angles between tangential vectors at the endpoints and the horizon (Figure 2.13, on the right). The default distance/angle pair $(0.5,0)$ is assigned to every edge without a discontinuity point on it. For example, the default discontinuity signature for texels with no segments is $S = [(0.5,0),(0.5,0),(0.5,0),(0.5,0)]$. If there is a segment passing the south and the east edges, then $S = [(d_1,\theta_1),(d_2,\theta_2),(0.5,0),(0.5,0)]$. Finally,

$$S = [(d_{l_1},\theta_{l_1}),(d_{m_1},\theta_{m_1}),(d_{l_2},\theta_{l_2}),(d_{m_2},\theta_{m_2})]\,, \tag{2.7}$$

for texels with two discontinuities.

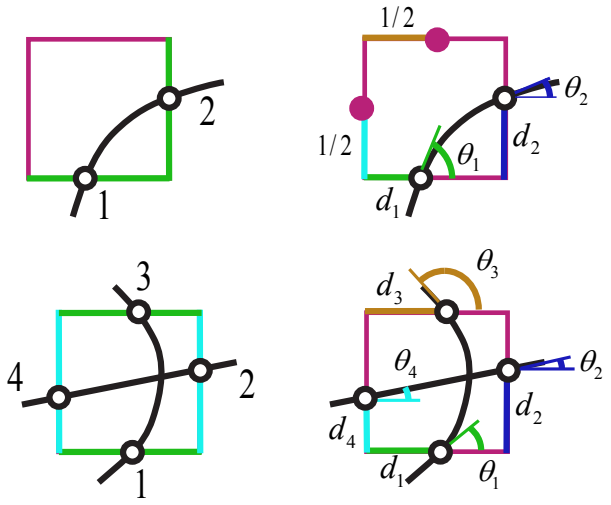


Figure 2.13: Examples of discontinuity descriptors: discontinuity configurations (left column) and discontinuity signatures (right column). First row: one curvy feature: $C = [(1,2),(1,2)]$ and $S = [(d_1,\theta_1),(d_2,\theta_2),(1/2,0),(1/2,0)]$. Second row: two curvy features: $C = [(1,3),(2,4)]$ and $S = [(d_1,\theta_1),(d_2,\theta_2),(d_3,\theta_3),(d_4,\theta_4)]$.

Encoding linear features requires storing two quadruples $C = [(l_1,m_1),(l_2,m_2)]$

and $S = (d_1, d_2, d_3, d_4)$: the signature S in this case does not include the angles (see Figure 2.14).

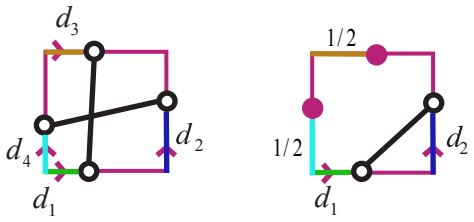


Figure 2.14: Linear features: just endpoints of discontinuity segments are stored in discontinuity signature.

The conversion process proceeds as follows. Initially, default empty configurations are assigned to all texels. Each feature curve is scan-converted to the texture to determine all texels it intersects (visited texels). For each visited texel, we check if there are one or two intersections of the curve with the texel's edges. In the former case, only the intersection point is used in the conversion process, while the part of the curve inside the texel is disregarded. In the latter case, we use a table of texel transformations to adjust the configuration; the table maps a pair (valid texel configuration, pair of edge indices for the new segment endpoints) to a transformation defining a new configuration. Any such transformation maps a valid texel configuration to another valid texel configuration and updates the texel's discontinuity signature by calculating the center of mass of all discontinuity points included so far per every edge (e.g., Figure 2.11).

The advantage of this algorithm is that it is easy to see that it will always produce a valid set of configurations for all pixels; however, it may introduce artifacts in rare situations like those depicted on Figure 2.12 (right).

2.6.1 Curvilinear features: optimizing invalid signatures

While texel configurations are guaranteed to be valid, in the case of curvy features, not every resulting discontinuity signature is valid. Some configuration-signature pairs may produce no Bezier segment at all [e.g., when their θs are equal, see Figure 2.15 (left)], produce a Bezier segment which leaves the interior of the texel [see Figure 2.15 (right)], or C^1 continuity is not maintained between connected discontinuity segments. To address these issues, we transform all invalid signatures by running a constrained optimization on the curve segments within the affected texels and their immediate edge-neighbors.

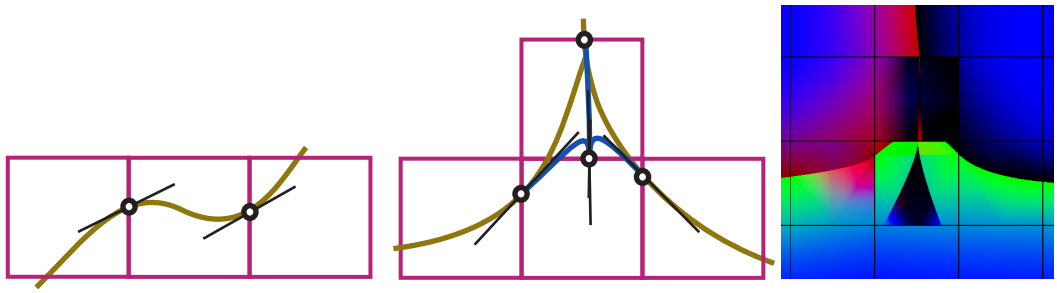


Figure 2.15: Examples of resulting invalid signatures during rasterization of feature curves. Left: parallel tangential directions at the "empty-circled" points never form any Bezier segment. Center: new Bezier segments (blue curves) leave the interior of the central texel. Right: Severe rendering artifacts resulting from an invalid signature.

We describe the details of the optimization on a texel with two discontinuity segments sharing one edge-point; other configurations are optimized in a similar way. We minimize an energy of a set of discontinuity segments including segments within the affected texel and connected segments in the adjacent texels. Figure 2.16 shows an example of an invalid configuration in the central texel with two Bezier segments δ_1 and δ_2 that are connected not C^1 continuously to α, β, and γ Bezier segments

within the east, north, and west immediate neighbors at points $\mathbf{b}_0^{\delta_2}$, $\mathbf{b}_0^{\delta_1}$, and $\mathbf{b}_2^{\gamma}$, correspondingly: e.g., to maintain C^1 continuity at point $\mathbf{b}_2^{\gamma_2}$, vectors $\overrightarrow{\mathbf{b}_2^{\gamma}\mathbf{b}_1^{\gamma}}$, $\overrightarrow{\mathbf{b}_2^{\gamma}\mathbf{b}_1^{\delta_2}}$, and $\overrightarrow{\mathbf{b}_2^{\gamma}\mathbf{b}_1^{\delta_1}}$ should be collinear.

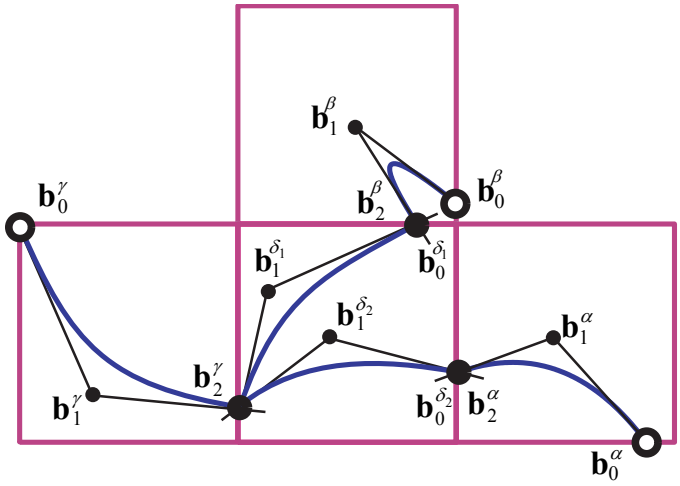


Figure 2.16: An example of the initial configuration for spline energy optimization, given by Equation 2.8, within an invalid central texel with two discontinuity segments δ_1 and δ_2. All the locations of filled circles are optimized.

To make the process local, we fix the signature angles and positions in the adjacent texels located on the edges not shared with the central texel (points $\mathbf{b}_2^{\alpha}$, $\mathbf{b}_0^{\beta}$, and $\mathbf{b}_0^{\gamma}$ depicted in in Figure 2.16 by open circles). Locations of control points (excluding fixed signature locations) of all Bezier segments are the variables in the optimization (filled circles in Figure 2.16). Linear inequality constraints keep all segments inside their original texels, and nonlinear equality constraints ensure tangent continuity with adjacent segments (δ_1 with β and γ, and δ_2 with α and γ in Figure 2.16). The complete optimization problem is

$$
\begin{cases}
E_{conf} = E_\alpha + E_\beta + E_\gamma + E_{\delta_1} + E_{\delta_2} \to \min \\
\text{S.T. linear inequalities} \\
\quad \mathbf{b}_1^\alpha \in T^o_{i,j+1}, \quad \mathbf{b}_1^\beta \in T^o_{i+1,j}, \quad \mathbf{b}_1^\gamma \in T^o_{i,j-1}, \\
\quad \mathbf{b}_2^\alpha \in \partial_e T_{i,j}, \quad \mathbf{b}_2^\beta \in \partial_n T_{i,j}, \quad \mathbf{b}_2^\gamma \in \partial_w T_{i,j}, \; \mathbf{b}_1^{\delta_1} \in T^o_{ij}, \; \mathbf{b}_1^{\delta_2} \in T^o_{ij} \\
\text{S.T. nonlinear equalities} \\
\quad P(\mathbf{b}_1^\alpha, \mathbf{b}_2^\alpha, \mathbf{b}_1^{\delta_2}) = \quad P(\mathbf{b}_1^\beta, \mathbf{b}_2^\beta, \mathbf{b}_1^{\delta_1}) = 0 \\
\quad P(\mathbf{b}_1^\gamma, \mathbf{b}_2^\gamma, \mathbf{b}_1^{\delta_1}) = \quad P(\mathbf{b}_1^\gamma, \mathbf{b}_2^\gamma, \mathbf{b}_1^{\delta_2}) = 0 \\
\text{S.T. linear equalities} \\
\quad \mathbf{b}_0^{\delta_1} = \mathbf{b}_2^\beta \\
\quad \mathbf{b}_0^{\delta_2} = \mathbf{b}_2^\alpha, \quad \mathbf{b}_2^\gamma = \mathbf{b}_2^{\delta_1} = \mathbf{b}_2^{\delta_2}.
\end{cases}
\tag{2.8}
$$

In the optimization problem formulation, T^o and $\partial_{e/n/w} T$ denote the interior and the (east/north/west) boundary of texel T, and $\mathbf{b} \in T^o_{i,j}$ is explicitly expressed by the inequalities $\left\{0 < \hat{b}_x - j < 1, 0 < \hat{b}_y - i < 1\right\}$. The points on the texel's edges, e.g. the west edge, $\mathbf{b} \in \partial_w T_{i,j}$ are identified using a function P as

$$
\left\{P(\hat{\mathbf{b}}, [j,i], [j,i+1]) = 0, 0 < \hat{b}_y - i < 1\right\},
$$

where $\hat{\mathbf{b}} = [b_x/\Delta U, b_y/\Delta V]$ and $P(\mathbf{p},\mathbf{q},\mathbf{r})$ is zero whenever its arguments lie on the same line. We use $P(\mathbf{p},\mathbf{q},\mathbf{r}) = (\mathbf{p}-\mathbf{q})\cdot(\mathbf{q}-\mathbf{r}) - |\mathbf{p}-\mathbf{q}|^2|\mathbf{q}-\mathbf{r}|^2$. Graphical representation of optimization constraints is shown in Figure 2.17.

The energy E_γ of a quadratic Bezier segment γ given in Bernstein form [43], $\mathbf{B}^\gamma(t)$, with control points $\mathbf{b}_i^\gamma$, $i = 1,2,3$, is the standard thin-plate energy:

$$
E_\gamma = \int_0^1 \left| \frac{d^2}{dt^2} \mathbf{B}^\gamma(t) \right|^2 dt = 4 \left| \mathbf{b}_0^\gamma - 2\mathbf{b}_1^\gamma + \mathbf{b}_2^\gamma \right|^2. \tag{2.9}
$$

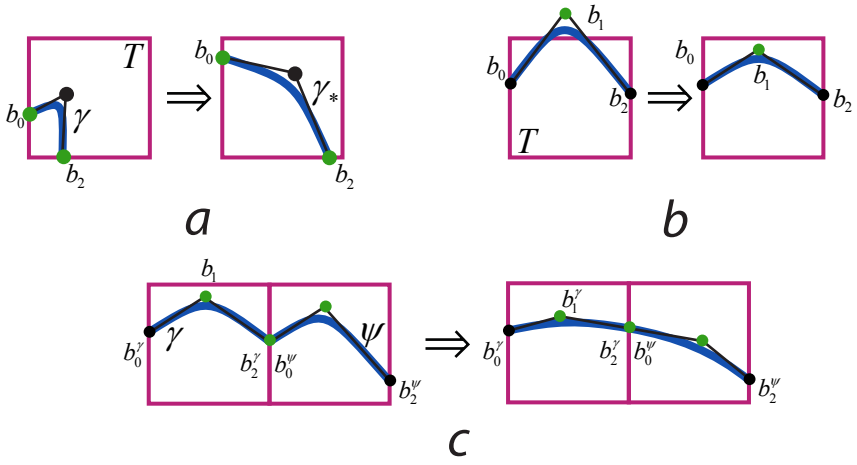


Figure 2.17: Optimization constraints. (a) End-points of the Bezier segments should move along the texel boundaries. (b) The middle Bezier control points should stay within the original texel. (c) The Bezier segments connected through the texel boundary should respect the tangent continuity. Control points under constraints are filled in green.

We use sequential quadratic programming to solve the optimization problem for each local configuration separately. An example of valid configuration optimized from an initial invalid configuration shown earlier in Figure 2.16 is depicted in Figure 2.18

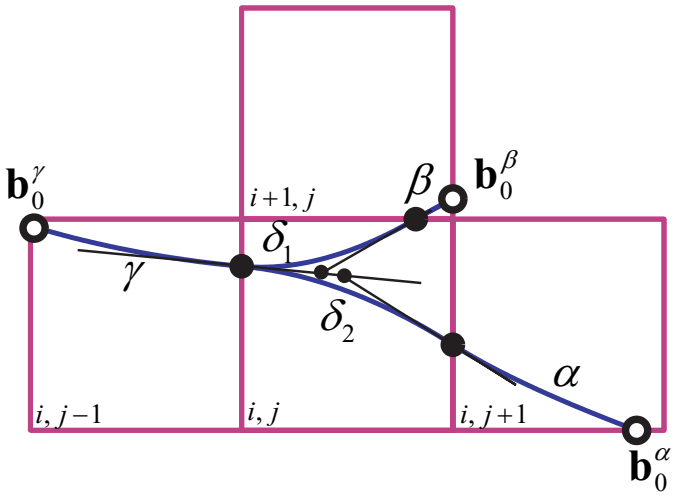


Figure 2.18: Optimal positions of the segments α,β,γ,δ_1, and δ_2. All the locations of open circles are fixed during optimization.

If only linear discontinuities are presented in the original texture, then every resulting discontinuity signature is valid. In this case, the preprocessing algorithm avoids signature optimization and is sufficiently fast for interactive feature line modification.

2.6.2 Rasterization of distance field and its gradient

The distance field values and the gradient values are calculated at the texel corners simultaneously with configuration transformations. We initialize the distance field with some "large" value and merge the local distance fields of individual features one by one by keeping the smallest distance values within the merging domains. Gradient

values corresponding to the smallest distances are stored in the gradient texture.

We smooth the resulting gradient texture $\nabla\hat{d}_{ij}$ by using constrained Laplacian smoothing (e.g., [135] for similar techniques). Specifically, each vertex moves toward the barycenter of its neighbors, excluding the neighbors on the other side of the discontinuity. We found that approximately 10 smoothing iterations is sufficient. As is well known, Laplacian smoothing quickly eliminates high frequency features, while slowly reducing low frequency features, which is the desired result in our case.

The significant improvement in appearance due to smoothing the gradient field is shown in Figure 2.28: compare two images on the right of the top row. Note that no smoothing is done on the distance function itself; we found it important to keep it unchanged, e.g., for maintaining approximately constant width of creases when desired.

2.7 Rendering of feature maps

The input to the real-time part of our algorithm include the following textures

- the distance map $\hat{d}_{ij}$[1] and the gradient field $\nabla\hat{d}_{ij}$,
- the discontinuity configurations $\hat{C}_{ij}$ and discontinuity signatures $\hat{S}_{ij}$,
- any other texture maps with features used for rendering (here we consider the normal map $\hat{\mathbf{N}}_{ij}$ as an example),
- user-defined crease profile $h(d)$. It may be stored in a one-dimensional texture $\hat{h}_k$.

[1] $\hat{d}_{ij}$ is a texture of unsigned distance field sampled at grid points $\Omega = \{(u_j, v_i), i = 1,2,\ldots,M_v, j = 1,2,\ldots,M_u\}$, $\hat{d}_{ij} = d(u_j, v_i)$. All the other textures are sampled on the same grid Ω.

Our rendering algorithm, implemented as a pixel shader, uses the normal map, per-texel discontinuity configurations and signatures and profiles to compute the normal values. Once all necessary quantities are interpolated, equation 2.3 is used to obtain the normal. The goal is to interpolate the distance field, its gradient, and normal texture in a way that respects feature curves. In particular, there should be no averaging of values across the discontinuity segments. We achieve this by using a three-stage interpolation procedure which partition the texel.

Discontinuity segments partition a cell into several domains. For valid configurations, the segments are topologically equivalent to a subset of the boundaries of an eight-triangle partition of the domain (Figure 2.8). For example, for a single discontinuity segment connecting two adjacent sides, one domain is a union of seven curvilinear triangles on one side of the segment and the other is one triangle located on the opposite side of the segment. For any configuration, we generate an *eight-triangle partition*, adding additional vertices when necessary. The vertices of the partition triangles located on the texel's edges (*edge vertices*) are either determined by the discontinuity endpoints (as for vertices q_b and q_d on Figure 2.19) or placed at the edge center when there is no discontinuity passing such edge (as for vertices q_a and q_c on Figure 2.19). The common *central vertex* r_m is an intersection of horizontal segment $\{q_b, q_d\}$ with vertical segment $\{q_a, q_c\}$. Partition triangle vertices are only created temporarily during the rendering step.

Our algorithm can be summarized as follows: we interpolate/extrapolate corner values of the texel contained in the same domain of the discontinuity partition as $\mathbf{p}_t$ (note there may be only one corner value available) to obtain the values at the vertices of the curvilinear triangle containing $\mathbf{p}_t$. Then, we obtain values at $\mathbf{p}_t$ by interpolating inside the triangle.

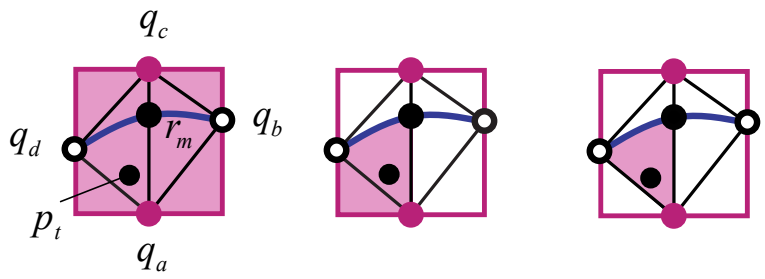


Figure 2.19: Steps in locating the partition's triangle containing $\mathbf{p}_t$; discontinuity feature passes through points q_b and q_d.

The curvilinear triangle which contains $\mathbf{p}_t$ is localized in two steps: first, by finding one of the four quadrants which contains the point, and second, by choosing the triangle containing $\mathbf{p}_t$ within the quadrant. This process is illustrated in Figure 2.19. It requires running a test to determine on which side of a discontinuity curve a point is (Section 2.7.1).

To determine the values at the edge vertices, we interpolate the samples at texel corners and then use the values at edge vertices to interpolate the value at the central vertex. The values at the edge and central are obtained using only samples from *reachable* texel corners: the corners in the same discontinuity partition domain with $\mathbf{p}_t$.

Once the samples at triangle corners are obtained, triangle's corners are interpolated to calculate the values at point $\mathbf{p}_t$: $d(u,v)$,$\nabla d(u,v)$, and $\mathbf{N}(u,v)$. Our procedure ensures the distance function is exactly zero on the discontinuity segments: if two edge vertices are on a discontinuity segment, the distance function value at these vertices is zero; the value at the central vertex is interpolated from these two values and is also zero. Our technique for interpolation on curvilinear triangles, described in Section 2.7.2, ensures all intermediate values along the discontinuity segment con-

necting an edge vertex with the central vertex are also zero.

2.7.1 Side test

Linear discontinuity. The side test determines on which side of a curve a point is located. To perform the side test, one usually finds an implicit form of the curve, $f(x,y)=0$, so that points with negative signs are located on one side of the curve and the points with positive signs are on the other side.

If a feature is a line segment $L[\mathbf{b}_0,\mathbf{b}_1]$, then its implicit form is given by $f_L(x,y)=\mathbf{v}_\gamma\cdot\mathbf{p}+c_\gamma$, where $\mathbf{v}_\gamma=(\widehat{\mathbf{b}_1-\mathbf{b}_0})_\perp$ and $c_\gamma=-\mathbf{v}_\gamma\cdot\mathbf{b}_0$.

The implicit form $f_L(x,y)=0$ describes a "larger object": a line which has a line segment as its part. As a line always divides the interior of a square-shaped texel into two parts, the side test based on the sign of line implicit form is always correct: a pair of texel points located on the same side with respect to a line segment has the same sign $f_L(x,y)$. In contrary, the curvilinear case may introduce ambiguities.

Curvilinear discontinuity segments. By using a well-known result from algebraic geometry [28], we find the implicit form of a Bezier segment, written in power basis as $\mathbf{B}(s)=[q_x(s),q_y(s)]$ with $q_i(s)=a_is^2+b_is+c_i$, by equating the resultant of the two polynomials $q_x(s)-x$ and $q_y(s)-y$ to zero:

$$\begin{aligned} f_B(x,y) &= \mathrm{Res}_{2,2}[q_x(s)-x;q_y(s)-y] \\ &= \begin{vmatrix} -a_x & -b_x & x-c_x & 0 \\ 0 & -a_x & -b_x & x-c_x \\ -a_y & -b_y & y-c_y & 0 \\ 0 & -a_y & -b_y & y-c_y \end{vmatrix} = 0. \end{aligned} \tag{2.10}$$

The two sides of the curve are defined by the sign of f_B. The side test, based on implicitization, may yield incorrect results for highly curved segments: a curved

segment may pass a unit square twice. For example, a discontinuity feature curve given by a Bezier segment $B[\mathbf{b_0}, \mathbf{b_1}, \mathbf{b_2}]$, depicted in Figure 2.20-(left), should separate points $\mathbf{p_1}$ and $\mathbf{p_2}$ from each other. However, the feature curve may have a parabola γ as its implicit form which intersects the texel a second time at the NW corner, Figure 2.20-(center). In this case the points $\mathbf{p_1}$ and $\mathbf{p_2}$ will be identified as being on the same side of the implicit curve which is unwanted in this case.

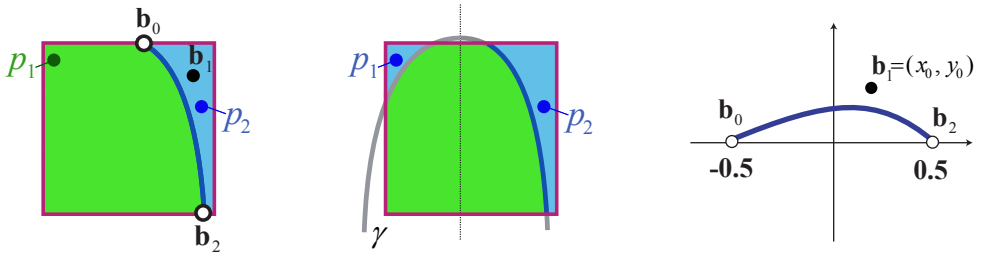


Figure 2.20: Left and Center: An example of an implicit form which assigns the same side for two points separated by a discontinuity. Right: Bezier segments.

Consider a parabola which corresponds to the Bezier segment $B'[\mathbf{b_0}, \mathbf{b_1}, \mathbf{b_2}]$ with control points: $\mathbf{b_0} = (-1/2, 0)$, $\mathbf{b_1} = (x_0, y_0)$, and $\mathbf{b_2} = (1/2, 0)$, see Figure 2.20 (right). It has the following unique form up to a constant factor:

$$4(-yx_0 + xy_0)^2 + 2yy_0 - y_0^2 = 0. \tag{2.11}$$

After applying a linear transformation

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{|\mathbf{b}_1|} \begin{bmatrix} x_0 & y_0 \\ -y_0 & x_0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.12}$$

the parabola becomes

$$\frac{2|\mathbf{b}_1|^3}{y_0^2} v^2 + \frac{x_0}{y_0} v - \frac{|\mathbf{b}_1|}{2} = -u. \tag{2.13}$$

The larger the leading coefficient is in Equation 2.13, the narrower the parabola will be, the more likely both branches of the parabola will intersect the texel.

Implementing a complete test on whether the parabola intersects the texel more than once is computationally expensive and assumes using many branching instructions. Instead, we use a simple fast test which is free from ambiguities, though, may reject some safe Bezier segments. We also describe what needs to be done to convert a Bezier segment which fails the test to its safe close variant.

The following propositions describe the methods of resolving the side test ambiguities against Bezier segments given in the form of B' for the following two possible cases: $x_0 \neq 0$ and $x_0 = 0$.

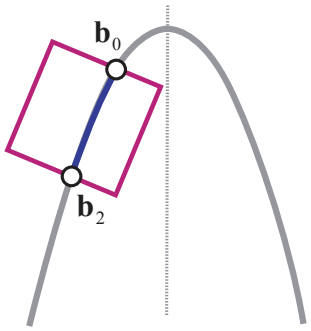


Figure 2.21: The configuration is always safe if the entire texel is located on one side w.r.t. the parabola's medial axis.

Proposition 3 (Resolving ambiguity when $x_0 \neq 0$)**.** *For a texel with the boundary edge length l_T, the side test based on the implicit form given in Equation (2.10) for the Bezier segment B' is always consistent if*

$$\min\left\{\frac{x_0 y_0}{4|\mathbf{b}_1|^3} \pm \frac{y_0}{2|\mathbf{b}_1|}\right\} > \sqrt{2}\, l_T, \tag{2.14}$$

provided that $2|\mathbf{b}_1|^2 < |x_0|$.

Proposition 4 (Resolving ambiguity when $x_0 = 0$)**.** *The case described in Proposition (3) when $x_0 \neq 0$ is changed to $x_0 = 0$ is always unambiguous if the angle between*

a tangential direction and the texel's edge, θ, is separated from zero by $\theta_{\min}$:

$$\theta_{\min} = \frac{2l_T y_0}{1+4(1+l_T)^2 y_0^2}$$

Proof. *($x_0 \neq 0$)* It is easy to see that if the entire texel lies on the same side as the control points $\{\mathbf{b}_i\}$ with respect to the parabola's axis of symmetry, then the texel will not intersect the parabola again keeping the configuration safe from ambiguity, see Figure 2.21. One way to satisfy this condition is to make sure that the distances from the control points to the medial axis are larger than the length of texel's diagonal $\sqrt{2}\,l_T$.

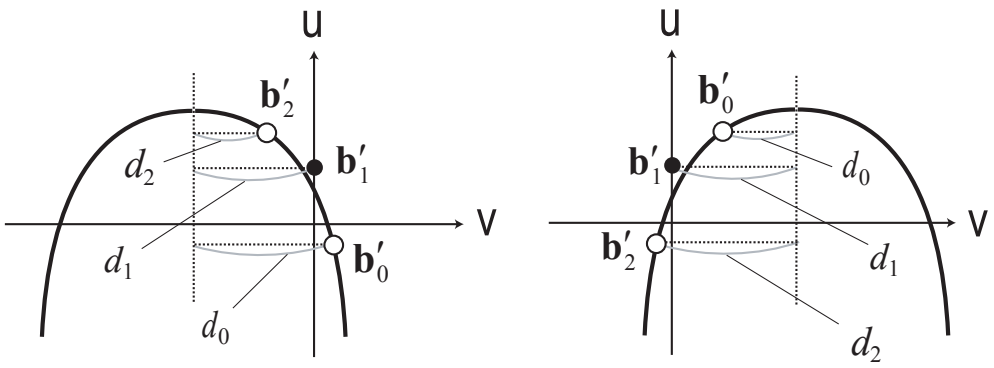


Figure 2.22: Resolving the side test ambiguity for the case $x_0 > 0$ (left), and $x_0 < 0$ (right)

Without loss of generality, we consider only the case when $y_0 > 0$. After applying a linear transformation (2.12), the control points $\{\mathbf{b}_i\}$ become

$\mathbf{b}'_\mathbf{0} = (-x_0, y_0)/2|\mathbf{b}_1|$

$\mathbf{b}'_\mathbf{1} = (1,0)/|\mathbf{b}_1|$

$\mathbf{b}'_\mathbf{2} = (x_0, -y_0)/2|\mathbf{b}_1|$.

The medial axis of the parabola given by Equation 2.13 is located at $v^* = -x_0 y_0/4|\mathbf{b}_1|^3$. Therefore, to make sure that all the points are on the right (left) from the axis when $x_0 > 0$ ($x_0 < 0$), the following condition should hold: $y_0/2|\mathbf{b}_1| < \pm x_0 y_0/4|\mathbf{b}_1|^3$ (see Figure 2.22). This may be simplified to $2|\mathbf{b}_1|^2 < |x_0|$, which is exactly what is required by the proposition.

The distance from a point $\mathbf{b}'_1$ to the parabola's medial axis, d_1, equals $|x_0|y_0/4|\mathbf{b}_1|^3$, and the corresponding distances d_0, d_2 for $\mathbf{b}'_0$ and $\mathbf{b}'_2$ are $x_0 y_0/4|\mathbf{b}_1|^3 \pm y_0/2|\mathbf{b}_1$. As $\mathbf{b}'_0$ or $\mathbf{b}'_2$ are the possible closest points to the axis (see Figure 2.22), the closest distance from the control points to the axis is $min\{d_0, d_2\}$. Therefore, the inequality $min\{d_0, d_2\} > \sqrt{2}\, l_T$ guarantees that the control points are at least the texel's diagonal length far away from the medial axis. Such inequality becomes (2.14) by substituting the values for d_0 and d_2. □

If the inequality (2.14) fails for the Bezier segment B', we adjust the position of the second control point $\mathbf{b}_1$ so that $x_0 = 0$, that will lead us to the second case. This adjustment may be brute and may introduce visible discontinuities across the texel edges. For example, one can update $\mathbf{b}_1 = (x_0, y_0)$ to be $\mathbf{b}'_1 = \big(0, 0.5\, y_0/(|x_0| + 0.5)\big)$, so that a new Bezier segment $B''[\mathbf{b_0}, \mathbf{b'_1}, \mathbf{b_2}]$ is under the case $x_0 = 0$ preserving its original tangent continuity at least at one of the texel boundary edges.

Proof. *($x_0 = 0$)* Without loss of generality, we consider only the case $y_0 > 0$ and search for necessary conditions on the angle θ for the texel's edge which passes the control point $\mathbf{b}_0$, see Figure 2.23. Suppose that the texel's edge intersects the parabola at the point $\mathbf{p}$. Then, to proof this Proposition, it is enough to derive a condition upon which the distance $d(\mathbf{b}_0, \mathbf{p})$ from $\mathbf{b}_0$ to the intersection point is larger than l_T.

A line representing the texel's edge has a parametrization $(1/2, 0) + (\cos\alpha, \sin\alpha)t$. If it were to intersect the parabola $y = -2y_0x^2 + y_0/2$ — derived from Equation (2.11)

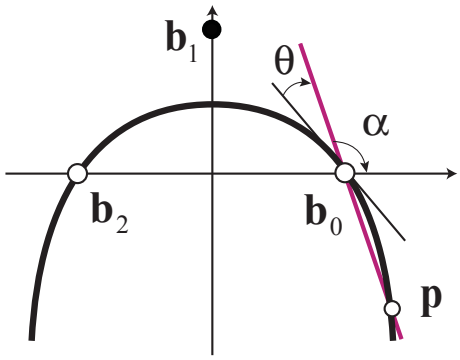


Figure 2.23: Resolving the side test ambiguity for the case $x_0 = 0$.

with $x_0 = 0$ — at point $\mathbf{p}$, the distance $d(\mathbf{b}_0, \mathbf{p})$ is

$$d(\mathbf{b}_0, \mathbf{p}) = \left(\frac{1}{2y_0}\tan\alpha + 1\right)^2 \left(1 + \tan^2\alpha\right).$$

The condition $d(\mathbf{b}_0, \mathbf{p}) > l_T$ holds if $(\tan\alpha/2y_0 + 1)^2 > l_T$. As the derivative of the parabola equation at point $\mathbf{b}_0$ equals $(-2y_0)$, the value of $\tan\alpha$ should be in the range $-\infty < \tan\alpha < -ky_0 < -2y_0$ for some $k > 2$. Therefore

$$\left(\frac{1}{2y_0}\tan\alpha + 1\right)^2 = \left(1 - \frac{k}{2}\right)^2 > l_T$$

is equivalent to $1 - k/2 < -l_T$ as $1 - k/2 < 0$. Thus, k should satisfies $k > 2(1 + l_T)$; it leads to the following condition on the $\tan\alpha$

$$\tan\alpha < -2(1 + l_T)y_0.$$

We can approximate the smallest possible θ by expanding the arctan function into Taylor series up to the first order

$$\theta_{\min} = \arctan(-2y_0) - \arctan(-2(1 + l_T)y_0) \;\approx\; \frac{2l_T y_0}{1 + 4(1 + l_T)^2 y_0^2}.$$

□

To ensure this condition holds, we always convert curvilinear discontinuity segments with small values of $\theta < \theta_{\min}$ to straight segments.

To move from a particular case of the Bezier segment B' to a general case with arbitrary control points $\{\boldsymbol{b}_k, k = 0,1,2\}$, one just needs to find the coordinates of the point $\boldsymbol{b}_1$ in the coordinate system given by the following basis vectors $\boldsymbol{e}_{B,x} = (\boldsymbol{b}_2 - \boldsymbol{b}_0)/\|\boldsymbol{b}_2 - \boldsymbol{b}_0\|$ and $\boldsymbol{e}_{B,y} = (-e_{B,x}[1], e_{B,x}[0])$, and normalize them by $l_T = \|\boldsymbol{b}_2 - \boldsymbol{b}_0\|$. Such coordinates can now be substituted into equation 2.14 to run our side ambiguity test.

2.7.2 Interpolation in a curvilinear triangle

We apply three different interpolation techniques to reconstruct the unknown values inside a triangle given three values at the triangle's corners, depending on the following types of the triangle's edge:

1. straight-line triangle: all edges are straight lines;
2. curvilinear *1C-triangle*: one curvilinear edge represented by a quadratic Bezier segment, see Figure 2.24 (right);
3. curvilinear *2C-triangle*: two quadratic Bezier edges, see Figure 2.25.

To find an unknown value $V(\mathbf{p})$ (i.e., distance, gradient, or normal) at a point $\mathbf{p}$ inside of a straight triangle $\langle \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \rangle$, given corner values $V_i \equiv V(\mathbf{p}_i)$, we apply the standard linear interpolation based on the barycentric coordinates $\{u_i\}$ of $\mathbf{p}_i$:

$$u_i = S(\mathbf{p}_{i-1}, \mathbf{p}_{i+1}, \mathbf{p})/S(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$$

$$V(\mathbf{p}) = \sum_i u_i V_i$$

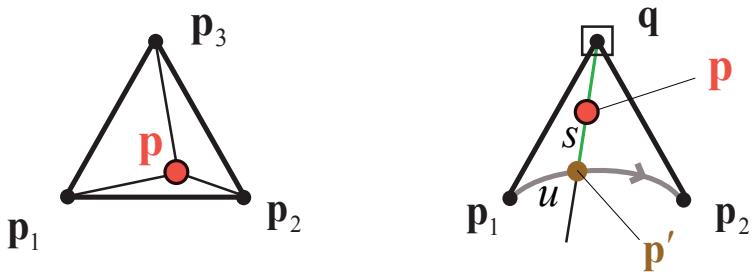


Figure 2.24: Interpolation schemes at a sample point $\mathbf{p}$ for triangles of different edge types. Left: a straight-edge triangle. Right: a 1C-triangle with one curvilinear edge.

For a 1C-triangle $\langle \mathbf{p}_1, \mathbf{p}_2, \mathbf{q} \rangle$, where $\mathbf{q}$ is adjacent to the straight edges (Figure 2.24, right), we define a map from the unit square with coordinates (s,u) to the triangle's interior, being degenerate at the edge $s = 0$. For any point in the triangle, we obtain the values of s and u by intersecting a ray $[\mathbf{q}, \mathbf{p})$ with the curvilinear edge $\{\mathbf{p}_1, \mathbf{p}_2\}$ to locate a point $\mathbf{p}'$. As the edge $\{\mathbf{p}_1, \mathbf{p}_2\}$ is parameterized by a quadratic function $\vec{\varphi}(u)$, the system of equations for s and u is

$$\begin{aligned} \mathbf{p}' &= \vec{\varphi}(u) \\ s &= \left|\mathbf{p}' - \mathbf{p}\right| / \left|\mathbf{p}' - \mathbf{q}\right| \quad . \end{aligned} \tag{2.15}$$

Solving the system reduces to solving a quadratic equation. We note that there is always exactly one intersection, which can be shown by using the variation-diminishing property of Bezier curves. Finally, we use bilinear interpolation to compute $V(\mathbf{p})$:

$$V(\mathbf{p}) = s \bullet \big(u \bullet (V_1, V_2), V_q\big), \tag{2.16}$$

where $u \bullet (V,W) \equiv (1-u)V + uW$

The procedure for locating $\mathbf{p}'$ and inverting Equation 2.15 is summarized here:

$$\mathbf{p} = (1-s)\vec{\varphi}(u) + s\mathbf{q} \tag{2.17}$$

$$s = \frac{p_x - \varphi_x(u)}{q_x - \varphi_x(u)} \tag{2.18}$$

$$\begin{vmatrix} p_x & q_x \\ p_y & q_y \end{vmatrix} = \begin{vmatrix} p_x & \varphi_x(u) \\ p_y & \varphi_y(u) \end{vmatrix} - \begin{vmatrix} q_x & \varphi_x(u) \\ q_y & \varphi_y(u) \end{vmatrix} \tag{2.19}$$

We solve the quadratic Equation (2.19) to find u, which is substituted into Equation 2.18 to find s.

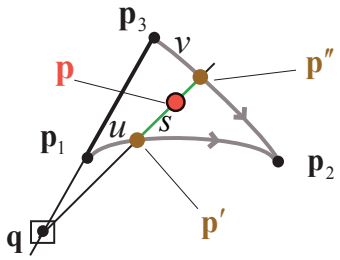


Figure 2.25: Interpolation scheme at a sample point $\mathbf{p}$ for a 2C-triangle with two curvilinear edges.

We use a different approach for 2C-triangles, as a simple radial parametrization of the type we use for 1C triangles is impossible.

For a 2C-triangle $\langle \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \rangle$ with curvilinear edges $\{\mathbf{p}_1, \mathbf{p}_2\}$ and $\{\mathbf{p}_3, \mathbf{p}_2\}$, parameterized by $\vec{\varphi}_1(u)$ and $\vec{\varphi}_2(v)$ respectively, we choose an auxiliary point $\mathbf{q}$ on the line passing the points $\mathbf{p}_1$ and $\mathbf{p}_3$ somewhere outside of the straight edge $[\mathbf{p}_1, \mathbf{p}_3]$. Consider the ray $[\mathbf{q}, \mathbf{p})$ going through point $\mathbf{p}$. Again, by the Bezier curve variation-diminishing property, one can show there will be exactly one intersection with curvilinear edges. For the sample $\mathbf{p}$, we find the intersection points $\mathbf{p}'$ and $\mathbf{p}''$ of the ray

$[\mathbf{q},\mathbf{p})$ with the curvilinear edges by applying the well known Bezier clipping algorithm described in [96]. Then, s is found the same way as it is done for a 1C-triangle by solving the following system:

$$\begin{aligned} \mathbf{p}' &= \vec{\varphi}_1(u) \\ \mathbf{p}'' &= \vec{\varphi}_2(v) \\ s &= \left|\mathbf{p}' - \mathbf{p}\right| / \left|\mathbf{p}' - \mathbf{p}''\right|, \end{aligned} \tag{2.20}$$

while the final interpolation formula is given by

$$V(\mathbf{p}) = s \bullet \left(u \bullet (V_1, V_2), v \bullet \left(V_q, V_2\right)\right). \tag{2.21}$$

2.7.3 Code outline for the normal interpolation shader

We summarize the part of our fragment shader program which estimates a desired normal $\mathbf{n}(u,v)$ at a given texel's sample $\mathbf{p}_t = (u,v)$ by reconstructing the values $\mathbf{N}_t = \mathbf{N}(u,v)$, $d_t = d(u,v)$, and $\nabla d_t = \nabla d(u,v)$, and by applying Equation 2.3 on the reconstructed values. The samples falls in the texel T_{ij}, where $i = \lfloor v/\Delta V \rfloor$ and $j = \lfloor u/\Delta U \rfloor$, so that the fragment shader has the following corner samples: $(\mathbf{n}_k)_{k=1\ldots4} = (\hat{\mathbf{N}}_{ij}, \hat{\mathbf{N}}_{ij+1}, \hat{\mathbf{N}}_{i+1j+1}, \hat{\mathbf{N}}_{i+1j})$, $(d_k)_{k=1\ldots4} = (\hat{d}_{ij}, \hat{d}_{ij+1}, \hat{d}_{i+1j+1}, \hat{d}_{i+1j})$, and $(\nabla d_k)_{k=1\ldots4} = (\nabla\hat{d}_{ij}, \nabla\hat{d}_{ij+1}, \nabla\hat{d}_{i+1j+1}, \nabla\hat{d}_{i+1j})$, See Figure 2.5 and the beginning of Section 2.7. As an example, Figure 2.26 shows normal interpolation for a texel with two discontinuity segments inside.

Algorithm 1 Normal shader

INPUT:

- the texture coordinates (u,v) of a current pixel's sample
- corners' samples $\mathbf{n}_k$, d_k, and ∇d_k, $k = 1..4$, of the texel $T_{\lfloor v/\Delta V \rfloor, \lfloor u/\Delta U \rfloor}$
- the discontinuity configuration/signature pair (C,S) for that texel.

OUTPUT: an estimated normal $\mathbf{n}(u,v)$.

1 Find locations of intermediate discontinuity points $\mathbf{q}_a$, $\mathbf{q}_b$, $\mathbf{q}_c$, $\mathbf{q}_d$ from the texel's discontinuity signature S by using offset values $S(1)$, $S(3)$, $S(5)$, and $S(7)$.

2 Fetch the texel's edge indices for end-points of its feature curves from the discontinuity configuration C: $C(1),C(2)$ for the first curve and $C(3),C(4)$ for the second curve. Reconstruct the Bernstein forms of the curves — $\mathbf{B}_1(s),\mathbf{B}_2(s)$ — by calculating their middle control points $\mathbf{b}_1^1$ and $\mathbf{b}_1^2$ as the intersection points of the rays going from the points $\mathbf{q}_*$ and into the directions given by the discontinuity signature S: angle values $S(2)$, $S(4)$, $S(6)$, and $S(8)$. Set $\mathbf{r}_m$ as an intersection of (possibly curvilinear) segments $\{\mathbf{q}_a,\mathbf{q}_c\}$ and $\{\mathbf{q}_b,\mathbf{q}_d\}$.

3 Reconstruct normals, distances, and gradients at the active discontinuity points — the points from $\mathbf{q}_a...\mathbf{q}_c$, which are on the same side of discontinuity segments as $\mathbf{p}_t$. Use the side test equations from (Section 2.7.1).

3.1 Depending on the number of corners reachable from a given active point, choose one of the following three cases to reconstruct the normal[1] $\mathbf{n}_*$:

3.1.1 two corners: interpolate the normal from the values at these corners;

3.1.2 one corner: copy the normal from the corner;

3.1.3 none (such an active point is a common end-point of the discontinuity segments with the sample point located between the segments): use the normal from the opposite end-point of either segment.

3.2 Set the resulting distance d_* to zero and assign a predefined value for the z-component of the resulting normal ∇d_* at every active point which is located on the discontinuity.

4 Determine the triangle containing point $\mathbf{p}_t$ by running the side test with respect to curvilinear edges of the triangles from the texel's partition; if needed, reconstruct the normal and distance at $\mathbf{r}_m$, interpolating between the values at the endpoints of the discontinuity with known values.

5 Depending on the type of the resulting curvilinear triangle, compute the interpolated normal $\mathbf{N}_t$, the distance d_t, and its gradient ∇d_t at $\mathbf{p}_t$ from known samples at the triangle corners, using the corresponding scheme from (Section 2.7.2); apply Equation 2.3 to the resulting values to calculate the final blended normal $\mathbf{n}(u,v)$.

2.8 Interactive rendering of linear features

Features formed by only line segments are much faster to render and require less memory than the curvy feature do. Indeed, the linear features do not include the

[1] the same rules are applied for reconstructing the distance d_* and its gradient ∇d_*; $* = \{a,b,c,d,m\}$

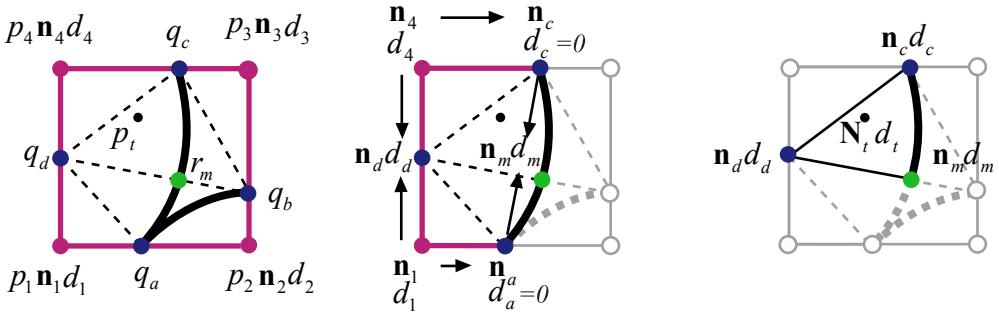


Figure 2.26: Computing the normal $\mathbf{N}_t$ and distance d_t at a texel sample $\mathbf{p}_t$. Left: $\{\mathbf{q}_a, \mathbf{q}_b\}$ and $\{\mathbf{q}_a, \mathbf{q}_c\}$ are the discontinuity segments inside the texel; locations of $\mathbf{q}_a$, $\mathbf{q}_b$, $\mathbf{q}_c$, and $\mathbf{r}_m$ are calculated from discontinuity signature; $\mathbf{q}_d$ is in the default position. Center: computing normals and distances at $\mathbf{q}_a$, $\mathbf{q}_c$, $\mathbf{q}_d$, and $\mathbf{r}_m$. Right: 1C-triangle $\langle \mathbf{q}_c, \mathbf{q}_d, \mathbf{r}_m \rangle$ covers $\mathbf{p}_t$; values at the triangle's corners are interpolated to compute $\mathbf{N}_t$ and d_t.

heavy operations on curve reconstruction and the interpolation within a straight triangle is much simpler than in a curvilinear triangle. Line segment implicitization is done in no time comparing with the Bezier curve implicitization. In fact, our implementation of linear feature shader contains only one branching which distinguish continuous and discontinuous texels [101].

A crucial difference between linear and curvy features is that a preprocessing step for linear features does not include time consuming optimization of invalid signatures as all signatures with linear features are automatically valid. We took advantage of this property and developed a program which can generate all discontinuity textures and render them simultaneously. The program starts either with a continuous texture without linear discontinuity features in it or with a texture which already has linear feature discontinuities encoded within our auxiliary textures $\hat{d}_{ij}$, $\nabla\hat{d}_{ij}$, $\hat{C}_{ij}$ and $\hat{S}_{ij}$.

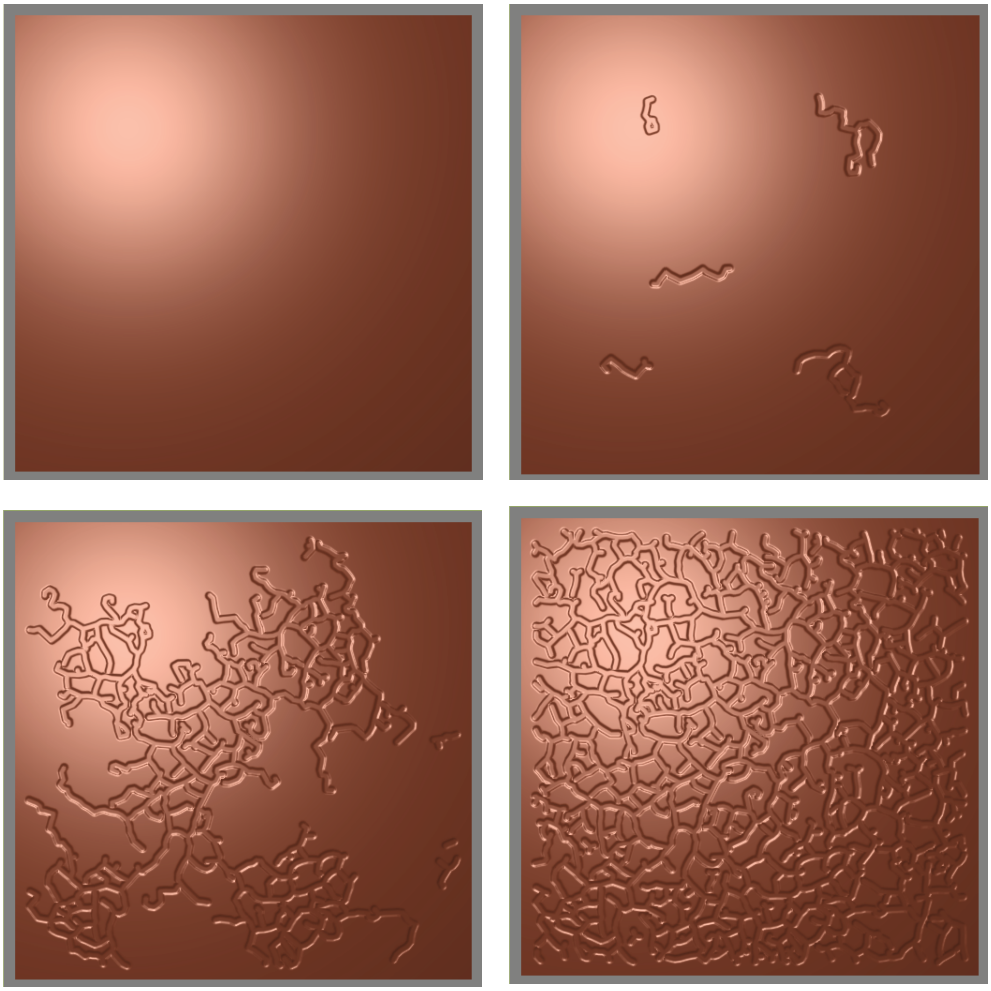


Figure 2.27: Snapshots of interactive growth of network-like embossing on a plane.

A user adds linear discontinuity features as needed either in an interactive mode or by providing a pre-computed set of features which should appear on the screen in a particular order with a given speed. For a current linear feature to be added we find a rectangular fragment in the texture domain which entirely covers the feature (taking into account the feature's width). We load corresponding fragments from the auxiliary textures into temporary textures of the size of the fragment and add a

contribution of the feature to the temporary textures by using algorithms described in Sections 2.5.1, 2.6 and 2.6.2. This approach does not require any adaptation of the latter algorithms as they are all sequential in the first place. The updated temporary textures are uploaded back to the corresponding fragments of auxiliary textures, so that a user can see a new linear feature immediately. The Figure 2.27 shows a snapshots of real-time embossing of a network-like pattern in a plane.

2.9 Implementation and Results

We have implemented the real-time interpolation algorithm as a fragment shader in Cg and tested the implementation on an NVidia GeForce 7800GS AGP 256M, running on Pentium 4, 2.8MHz. The frame rates that we have obtained for 512x512 images are in the range from 25 to 215 frames per second (fps) for interpolating curvilinear features, and from 65 to 380 fps for interpolating features approximated by linear segments. Such wide ranges is the result of sensitivity of the performance to the ratio of discontinuity pixels, configuration complexity of feature curves being rendered in the current frame, and to the size of the scene.

A normal map with feature curves is rendered in two passes. We run the interpolation shader in the first pass. It interpolates the normal texture with discontinuities represented by feature textures and stores the resulting normals in the Framebuffer object (FBO) [54] attached to the current fragment output. FBO is faster during switching than its former alternative p-buffer, and it maintains up to 32 bit float images as the fragment destination texture. On the second pass, the FBO with stored normals is detached from the fragment and is used as a general texture image that provides pixel normals for rendering.

The interpolation shader starts with reconstructing the feature curves inside a

texel containing a texture sample encoded by the texture coordinates of a point currently observed by the fragment shader. It partitions the texel interior into 8 curvilinear triangles, and locates a triangle which covers the texture sample. It then propagates known distance/gradient/normal values from the texel's corners to the corners of the triangle (see Figure 2.19). Finally, the normal at the sample is interpolated within the triangle by applying the corresponding interpolating algorithm, described earlier in (Section 2.7.2). All the calculations are performed in single precision arithmetic (floats). More detailed description of the shader can be found in the Algorithm 1.

Memory consumption. We use two extra textures: RGBA float and 16 bit grayscale. The former encodes discontinuity signatures $\hat{S}_{ij}$: its "RG" component contains the south edge pair of the signature, and its"BA" component carries the west edge pair. The east and the north edge signature pairs are located in the corresponding "RG" and "BA" components of the east and the north texel neighbors. The latter stores discontinuity configurations $\hat{C}_{ij}$. Distance values are stored in the unused "A" component of the normal map texture. Therefore, our approach requires to consume in total extra 5 floats per texel.

Figure 2.28 compares different modes of rendering normal maps with sharp features, enabling different parts of our approach one-by-one. The obvious artifacts at the bottom of the crease on the images from the first column are eliminated by using a reduced version of our method which approximates the gradient field from the distance samples. However, the resulting normal field is not smooth along the texel edges as can be clearly seen at the high intensity spots in the second column images. This occurs because the interpolated distance field is only C_0 continuous across the texel edges. The images in the third column look more smooth at the spots after separating the distance and gradient interpolation into two independent processes. Fi-

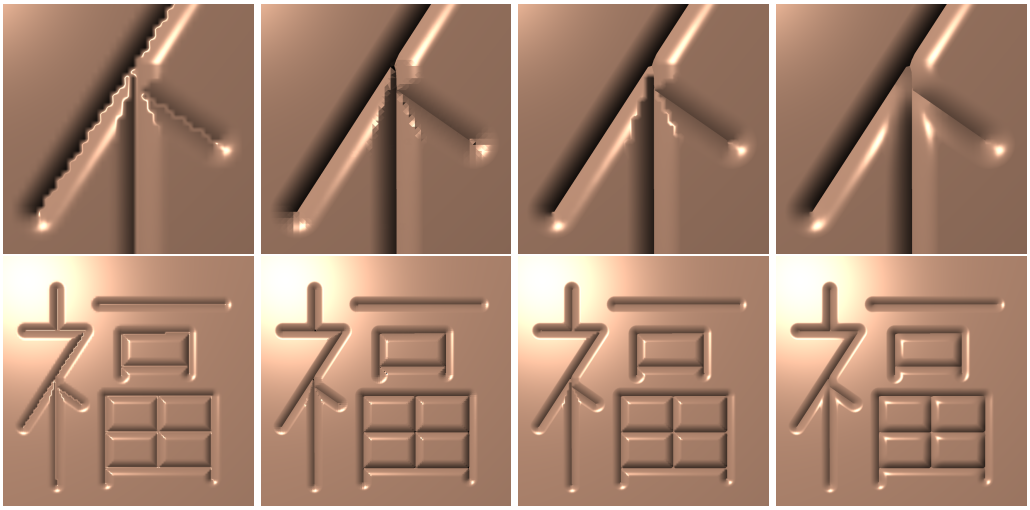


Figure 2.28: The appearance of features for different interpolation methods, from left to right: standard bilinear interpolation of samples representing the profile; piecewise linear distance function and piecewise constant gradient computed directly from the distance function; piecewise linear interpolation for gradient and distance function, without constrained smoothing of the gradient field; same with smoothing.

nally, the artifacts along the distance medial axis are smoothed away by a preliminary filtering of the gradient field at the preprocessing stage.

2.9.1 Feature curves

Figure 2.29 shows several examples of applying user-specified profiles. All discontinuity features are curvilinear (quadratic Bezier curves) and stored in the 128x128 textures (one of the examples shows texel sizes). The close-up views show that our technique results in few artifacts even in complex situations. The rendering frame rates for these images were 65 fps, with 75 fps for the close-up views and up to 215 fps when the object only partially covers the rendering fragment. These exam-

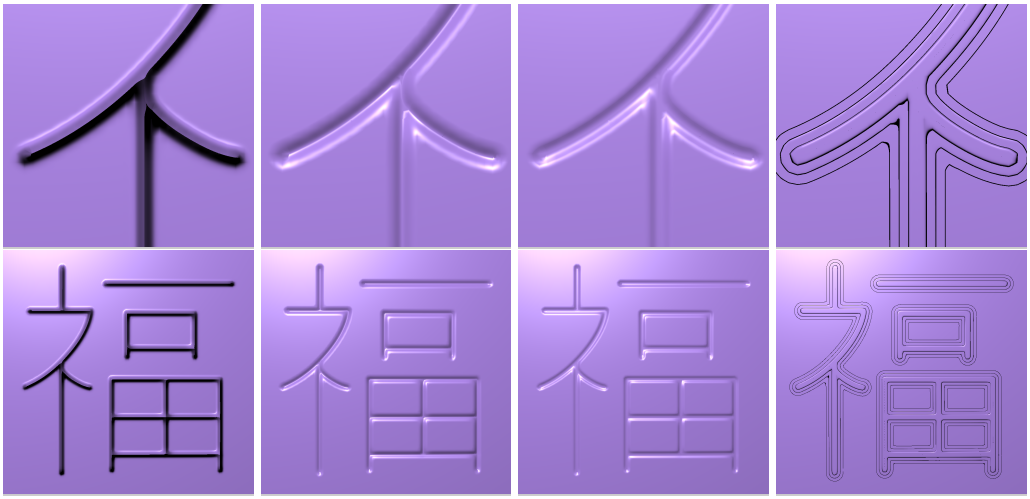


Figure 2.29: Examples of user-defined profiles. The discontinuity map is the same in all cases; note that in some cases, the discontinuity is removed entirely: a spectrum of creases of variable sharpness is possible.

ples have a simple feature pattern and the smallest mesh size, which makes them the fastest among the examples with curvilinear features that follow.

Figures 2.30– 2.34 show several images of models of feature-based normal maps rendered by our technique with different types of features and surface properties. 512x512 textures were used for rendering images on Figures 2.30, 2.34, while the snake mesh of Figure 2.33 was wrapped by a 1024x1024 texture.

The coke can with the fingerprints on its surface shown in Figure 2.30 is rendered at 46 fps (with at least 93 fps when the can is scaled to one-half of the image size). Its magnified version, shown on the right, runs as low as 26 fps. To clarify the reasons for inferior performance relative to the previous example, we considered the frame rate dependence on the level of complexity of feature curves and the size of underlying mesh.

Figure 2.30: Coke can with added fingerprint features; magnified views (without can's texture) shown at the bottom.

Definition 5 (feature coverage ratio (FCR)). *We define FCR as a ratio of pixels whose texture samples are located within discontinuity texels to the total number of pixels in the current output frame*

We measured a feature coverage ratio for the can image, and found a strong impact of the feature pattern complexity within the current rendering frame on the overall performance. Indeed, the FCR was approximately 5.5% for the full view can image, while discontinuity features covered more than 19% of the current frame for the magnified version of the can. However, the FCR only partially explains the resulting performance drop. The images of the Chinese character "luck" on the top row of Figure 2.29 also have an FCR approximately equal to 5%, but they run almost two times faster than the full view image of the can. The obvious difference between the two images is in the locality of discontinuity pixels in the current frame. Usually, the performance of GeForce 6/7 series cards suffer from frequently occurring incoherent branching in the pixel shaders. Our shader has a number of conditional statements, including the main one, which checks whether a pixel overlaps a discontinuous texel. The fragment of the magnified luck character image has many more coherent continuous pixels than the coke can image where continuous and discontinuous pixels are mixed. As a result, a combination of the FCR ratio with pixel coherence dominates overall performance.

The performance is only slightly sensitive to the size of the underlying mesh. The mesh of the coke can has 4.2K faces. However, performance improved only 6%, from 46 fps to 49 fps, for a two-triangle square with the same normal map applied.

The image of the plate with an Escher pattern Figure 2.31 runs as low as 25 fps (35 fps for the magnified image). Performance is relatively poor in this case versus other cases with comparable FCR because the pattern contains more discontinuity

texels with two feature curves in them.

Table 2.1 summarizes the performance of our feature curve shader running with textures we just discussed. Pixel coherence and the value of the FCR are the parameters which greatly influence the performance, while the size of the mesh is a less important parameter.

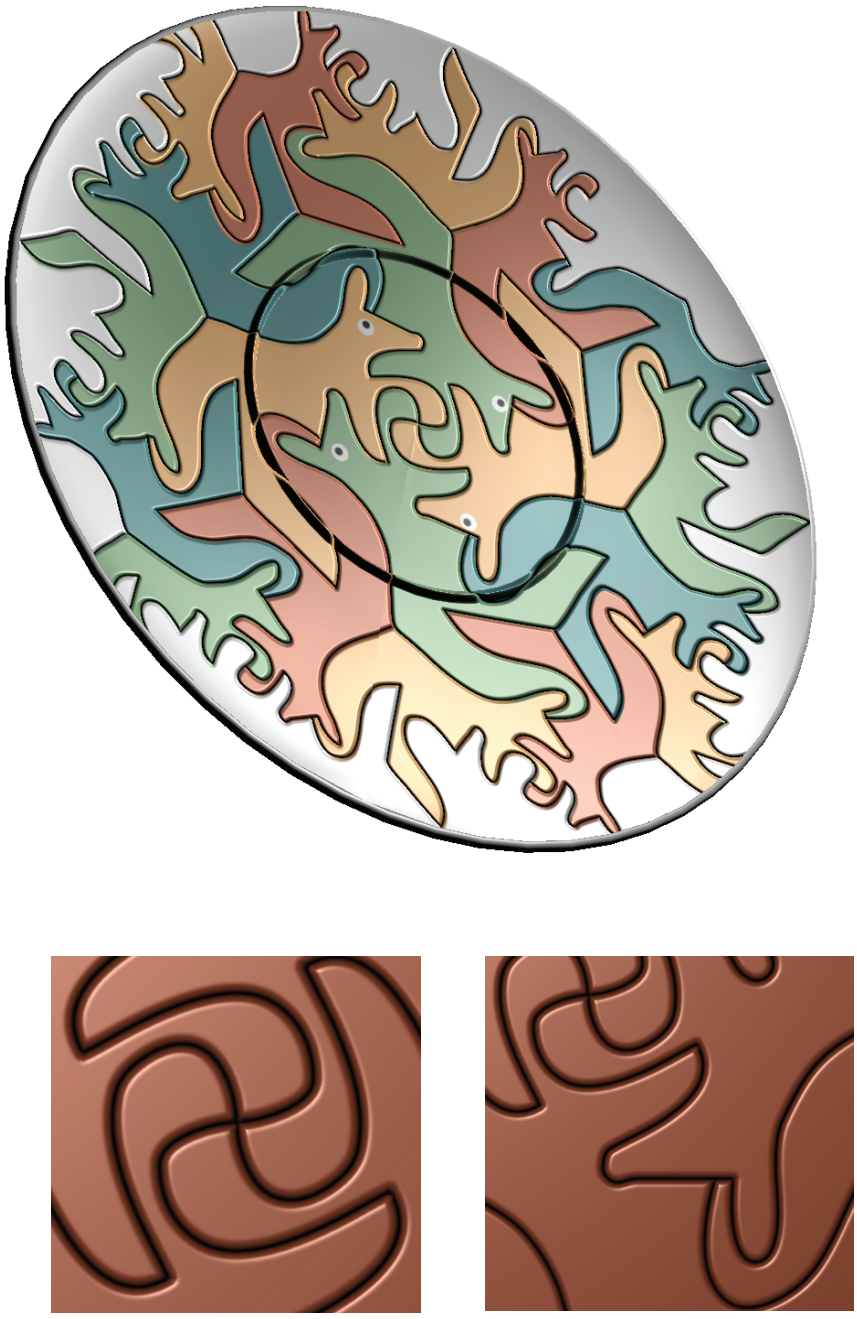


Figure 2.31: Plate with an Escher pattern; magnified views (without plate's texture) shown at the bottom.

2.9.2 Linear features

Our algorithm can be easily transformed into a shader for interpolating normal maps with linear features, which runs significantly faster at any resolution. Indeed, there is no need for time-consuming curve reconstruction in the step 2 (see Algorithm 1); the side test for the linear segments does not require computationally intensive curve implicitization in the step 3; and, an interpolation within the straight triangle does not require solving quadratic equations as in the case for curvilinear triangles in the step 5. Moreover, our version of the shader for linear features has only one branching: the one which checks the type of the pixel. Memory expense decreases to 3 floats per texel as tangential directions need not be stored.

A piece of glass depicted in Figure 2.32 has a very intricate pattern of linear features with joints having as many as 7 meeting features. The resulting feature texture has 91% discontinuous texels. The rendering rate in this case is 222 fps (114 fps for the first magnified image, and up to 380 fps for the version scaled by 75%) with FCR being close to 37% (90% and 20.2% for these two cases).

We approximated feature curves by linear segments for the snake mesh depicted in Figure 2.33. It runs at 124 fps with 10% FCR, and at the same performance with 35% FCR in the case of the magnified version (the mesh has 3.3K faces). Approximate feature curves look smooth enough at certain magnifications. However, the corners are already noticeable in the magnified version, which is not yet the closest view.

Teapots with features created by Delaunay triangulation shown on Figure 2.34 are examples of using user-defined feature profiles. They illustrate that using different profiles may change the way the teapot is perceived. Both the full view image and its magnified version run at 75 fps, though their FCR ratios are different, equaling 25.4%

	”Luck” res.=128^2, dscnt.=163K, Fig. 2.29			Coke can res.=512^2, dscnt.=2.6M, Fig. 2.30				Escher’s pattern res.=512^2, dscnt.=2.6M, Fig. 2.31	
	close-up	full view	full view zoom 1/2×	close-up	full view		full view zoom 1/2×	close-up	full view
fps	75	65	215	26	46	49	93	35	25
FCR	5	1.6	0.3	19.0	5.5	5.5	2.3	8	4
♯faces	2	2	2	4.2K	4.2K	2	4.2K	4.3K	4.3K

Table 2.1: Feature curves performance measured in frames per second (fps) and superimposed to feature coverage ratio (FCR) and the underlying mesh size. “res”=texture resolution, “dscnt”=memory used for storing discontinuities.

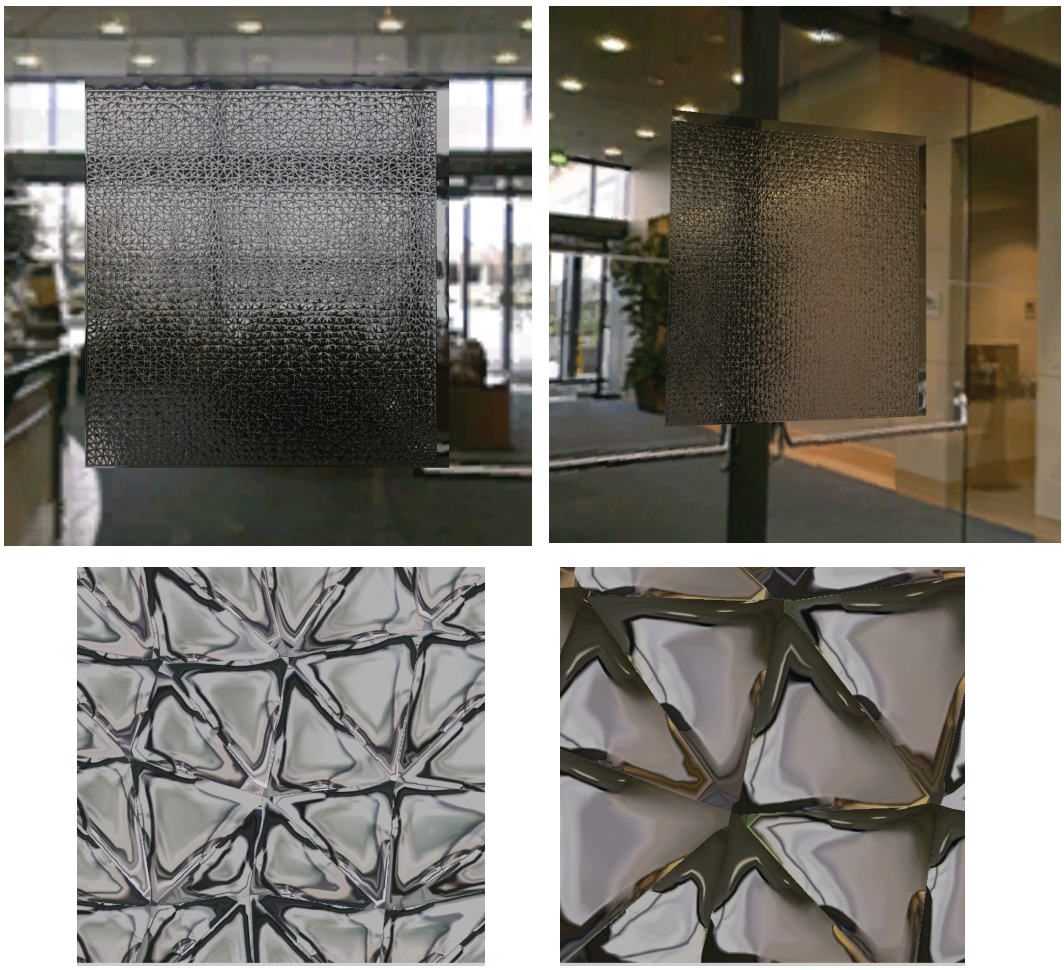


Figure 2.32: Glass with fractures; magnified views shown at the bottom.

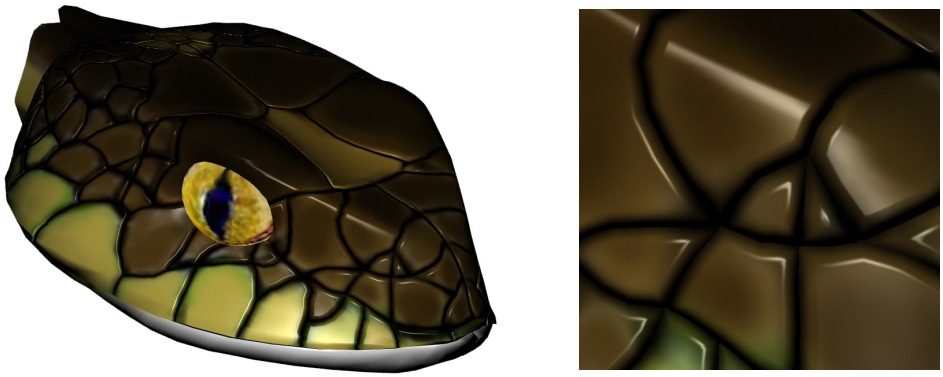


Figure 2.33: An example of a snake model with curvilinear features approximated by a set of linear segments

and 89.2%, respectively. We found a clear dependence of linear feature performance on the mesh size, in contrast to a negligible size-performance dependence for feature curves. The teapot mesh (6.3K faces) is almost two times larger than the snake mesh. The magnified versions of both examples run faster when features are mapped to the plane (keeping the same FCR values): the teapot plane fragment runs at 131 fps, and the snake plane fragment runs at 156 fps as opposed to 75 fps and 124 fps reached for the 3D meshes (More evidence on the performance dependence of the linear version of our shader on the mesh size is collected in Table 2.2).

Finally, we demonstrate how our algorithm handles complex star configurations, such as the one shown in Figure 2.35 (left), on which most of the existing GPU algorithms fail. While artifacts are inevitable by design (our discontinuity texel can not have more than two intersecting curves), we keep them minimal and in the majority of the patterns we guarantee holding C^1 continuity along the feature curves. At the preprocessing step, the central intersection point is split into several new points,

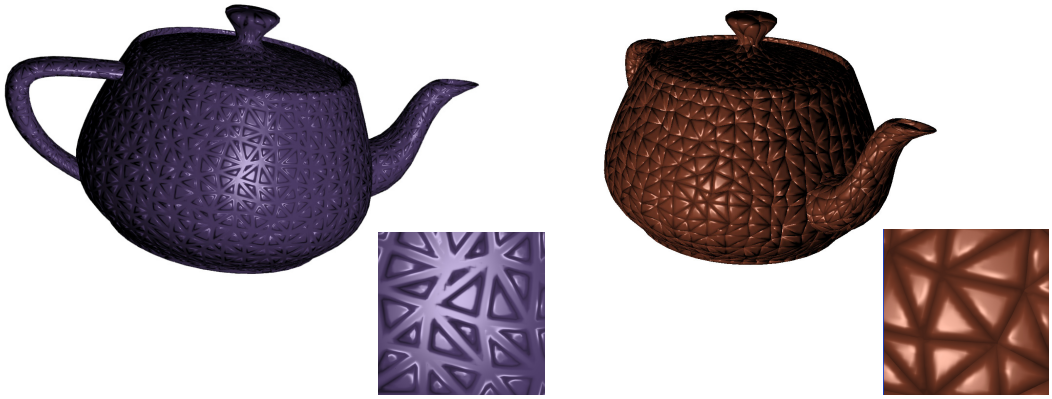


Figure 2.34: An example of different profiles; a magnified view is shown for each texture

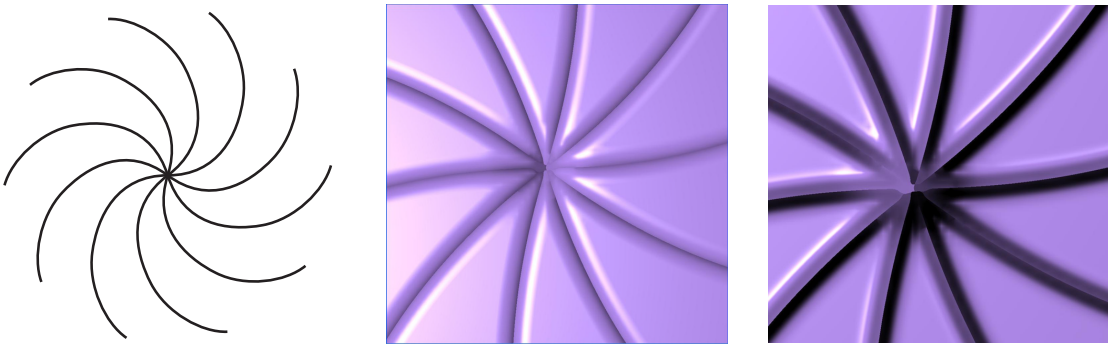


Figure 2.35: Magnified view (x15) of feature curves constructed from spiral arrangement of Bezier curves (small picture on the left). Discontinuity map resolution: 512x512.

	Glass res.=512^2, dscnt.=1.5M, Fig. 2.32 (c)			Snake head res.=1024^2, dscnt.=6.2M, Fig. 2.33			Teapot res.=512^2, dscnt.=1.5M, Fig. 2.34					
	close-up	full view	full view zoom 2/3×	close-up		full view	close-up		full view	full view reduced mesh		
fps	114	222	380	124	156	124	75	131	75	98	115	140
FCR	91	37	20.2	35	35	10	89.2	90.2	25.4	24.5	40.4	46.7
♯faces	2	2	2	3.3K	2	3.3K	6.3K	2	6.3K	4.7K	3.9K	3.1K

Table 2.2: Performance measured for linear feature shader. "res"=texture resolution, "dscnt"=memory used for storing discontinuities.

which represent simpler intersections of nearby curves in the neighboring texels: Figure 2.35.

Figure 2.36: Example of rendering sharp edges. Discontinuity map resolution: 80x80.

The shader for rendering sharp curvilinear edges of simple objects achieves the same performance as the shader for feature lines. Snapshots of rendering the letter 'R' is shown on FIgure 2.36.

2.10 Summary

We have described methods for representing and real-time rendering of surfaces with highly detailed geometry formed by sharp curvy features within a given normal map. Features are defined in the texture domain as functions of unsigned distance to the discontinuity curves, represented by quadratic Bezier segments, and its gradient. Real-time algorithm runs as a pixel shader which computes the discontinuous part of the normal map by interpolating distance and gradient fields, respecting discontinuities, and by applying a cross-section profile. The result is blended with the continuous part of the normal map and stays sharp at any resolution. Our preprocessing

algorithms converts feature curves into texture representation. It resolves intricate feature configurations and enables storing up to two curves in each texel.

Our model provides more flexibility in defining the discontinuous textures. We do not impose serious restrictions on the input network of feature curves and allow open features. The original sharp features can be modified during preprocessing and in real-time by applying an arbitrary profile provided by a user. Being able to store only two curves within each texel is a limitation of our approach; however, we demonstrated that our optimization algorithm usually resolves well complex configurations of curves meeting at one point.

We achieved a satisfactory performance for all the examples we presented in this work (see a performance summary for two generations of target GPUs in the Table 2.3). Among the main factors affecting the performance were coherence and the ratio of discontinuous texels.

	Curved features			Linear features		
	far	full view	close-up	far	full view	close-up
7800 GS	90–215	25–65	25–75	> 200	> 95	> 75
8800 GTX	> 80	> 80	> 65	> 300	> 100	> 100

Table 2.3: Comparison in performance measured for two different generations of NVidia GPUs.

Chapter 3

Parametric synthesis of patterns with curvilinear features

3.1 Introduction

Many natural and synthetic textures depend on underlying arrangements of features. Such features can follow the object boundary or shadow, or define fine-scale geometry, for example, locations and profiles of creases, cracks or furrows on a smooth surface. Usually, the basis for such features is formed by a set of one-dimensional primitives, e.g., lines, line segments, and curves. Particular configurations of such primitives are unique for certain types of natural textures. Therefore, if one tries to replicate a natural texture, the quality of the result will significantly depend on how close simulated configurations of primitives are to that of the source.

Primitive configurations can be completely random, as in the case of random locations of paper fibres illustrated in Figure 3.1-a, or be constructed by some deterministic process, as for the line segment primitives forming the boundaries of the texture with tiles shown in Figure 3.1-b. Configurations of these types are easy to replicate.

Indeed, the locations of paper fibres can be simulated by scattering a number of lines over a plane (or space) uniformly, and a tiled texture can be filled with copies of a tile in a straightforward manner. The arrangements of the former type are a part of a more general framework of random processes of geometrical objects: line processes. The methods for investigating the statistical properties and generating samples of line processes have been thoroughly developed (for example here [130, Ch.8] and [3]). The main property of line processes (and similar processes of geometric objects) is that the lines (objects) are distributed completely at random and the locations of individual lines are independent from the rest of the line configuration. However, it is clear that not all the natural arrangements follow the independence assumption – there are many real-world examples where the texture underlying networks of lines or curves form correlated structures. Such correlation needs to be analyzed and to be encoded into a mathematical model to be able to replicate the networks which match the correlation.

Typical examples of natural textures with correlated features are illustrated in Figures 3.1-(c–g). The feature configurations in these textures are more regular than that of line processes but still placed somewhat randomly. One way of replicating such textures is to investigate physical processes which cause the formation of the features. For example, a 3D surface cellular automation algorithm which models and employs soil stress distribution was successfully used in [53] to simulate dry soil crack pattern, similar to the one shown in Figure 3.1-c. Main disadvantages of the physically based techniques are their potentially extensive computational demand and a lack of user-type control over the simulation.

Another way of generating natural patterns is to develop a deterministic model with some elements of randomness which simulates key properties of the target pattern. For example, fine-scale geometry of human skin surface is defined by net-like

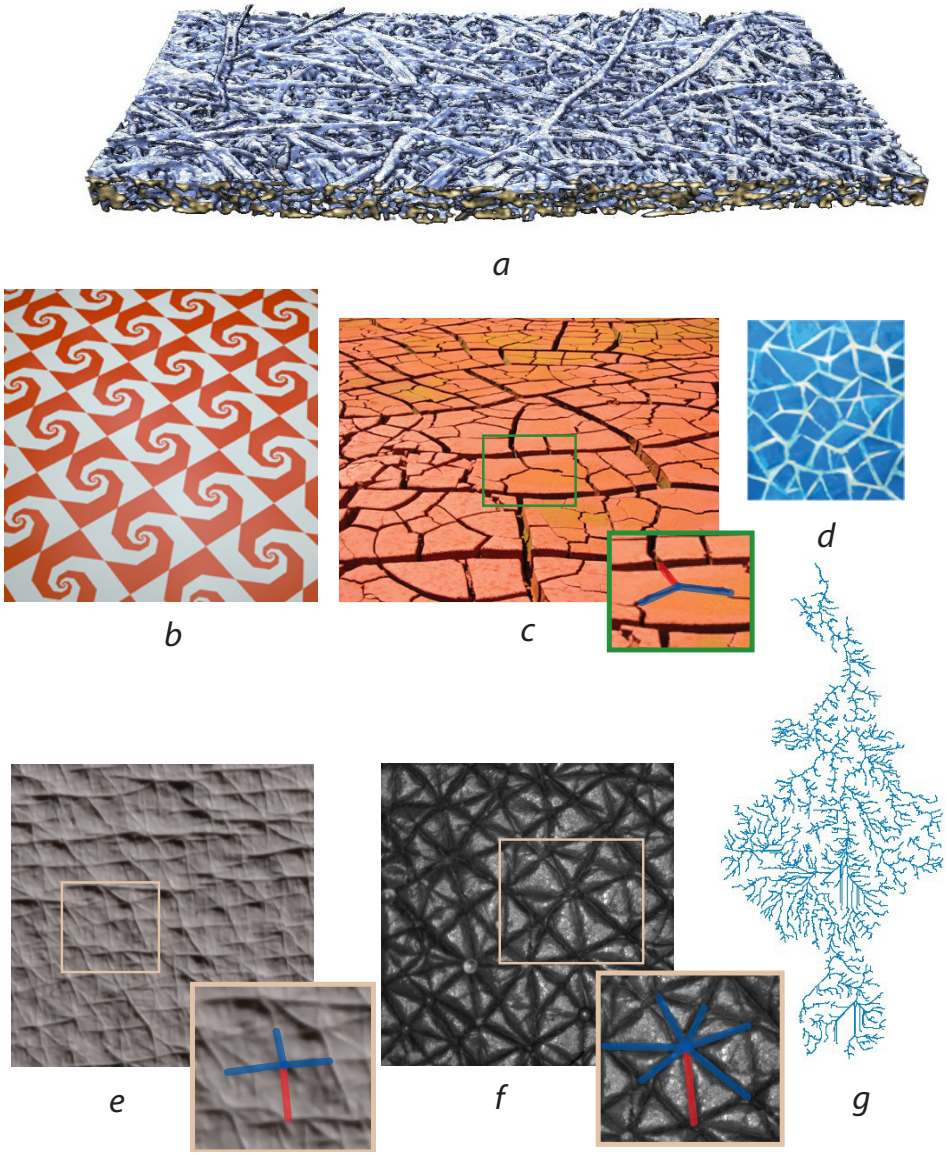


Figure 3.1: Examples of natural patterns. (a) Uncorrelated networks of paper fibres (University of Jyvaskyla, Finland), (b) tiling, (c) cracks formed in dry soil, (d) glass mosaic, (e,f) fine resolution negative replicas of fragments of human skin, (g) a fragment of the drainage network of Nile river. Inset: typical local configurations of primitives forming texture underlying networks.

micro-structures of the size about 100 microns, such as furrows and pores. Two different fragments of networks of skin furrows are illustrated in Figures 3.1-e and -f. Wu et al. [153, 152] assumed that the pattern of furrows can be decomposed into triangular cells. They applied a hierarchical Delaunay triangulation to divide texture space into a triangle mesh, so that the edges of triangles represent skin furrows. Resulting furrow networks showed close resemblance with some fragments of skin surface — generating facial skin regions would involve significant manual work to establish necessary hierarchy, and also the triangular pattern type assumption may not be valid at those regions. Bando at el. [12] made assumption that the skin furrows go locally in two directions: they grow the furrow network by placing slightly jittered streamlines which are guided by given direction fields. This method has more control on simulation and produces decent results for some skin fragments; however, it suffers the same assumption breaks as the previous method does.

Our approach is different: we want to represent the texture defining configurations of curves and lines as a random process whose samples closely reproduce the correlation pattern between primitives located nearby. Inter-primitive correlation pattern is unique for each texture and, roughly speaking, it defines likelihood of two close primitives meeting at a particular angle.[1] The random process should be represented by a simple but at the same time powerful enough probabilistic model which encodes the correlation pattern: a process described by a density function with an intuitive set of control parameters is an ideal candidate. There should be a simple and fast way to generate arrangements of curvilinear primitives distributed according to the random process.

[1] Inset Figures 3.1-c, -d, and -f show configurations of close primitives — in this case line segments — typical for their corresponding source textures. One can fix a segment (segment colored in red) and analyze correlation between the segment and other segments (colored in blue) in its vicinity.

We develop our model based on an assumption of local dependency: curvilinear primitives forming the texture are not completely independent from each other and the location and the orientation of each primitive depends only on primitives located nearby. This assumption allows us to construct a Markov type random process which is both parametric and can be sampled by running a corresponding Markov chain with a variant of the Monte Carlo Metropolis-Hastings algorithm.

Our model is based on interaction between curvilinear primitives — *fibres*. Close fibres "interact" between each other by means of interaction potentials which accumulate interactions between infinitesimal parts of the fibres and are expressed as certain interaction integrals. Infinitesimal parts of fibres are represented by 3D points ($point \times orientation$) to be able to adapt Gibbs-type 3D point distributions for describing interaction between the fibre infinitesimal parts. Gibbs point processes are powerful random models based on distance dependent potentials which are controlled by intuitive sets of parameters, are empowered by statistical inference algorithms, and can explicitly encode angular correlation of fibres. We test the interaction potential model on *linear fibres* — random fibre system consisting of line segments. The linear fibre processes are extended to include a comprehensive user control based on local and global constraints.

As in the case of Gibbs point process, the best results for the linear fibres can be achieved by using so-called inhibiting interaction potentials. The main challenge in working with these potentials is to ensure smoothness and connectivity of the resulting linear fibre systems. We developed a variant of the streamline placement algorithm which generates overlapping networks of smooth curves aligned to cross-orientation vector fields. Such fields are constructed and optimized from tangent fields formed by generated linear fibre systems.

Thus, our main contributions to the problem of parametric texture synthesis are

threefold:

- New Gibbs-type probabilistic model and algorithm for representing and generating random arrangements of linear fibres based on inter-fibre interaction potentials;
- Set of user control which significantly enhances the original linear fibre model by introducing a variety of constraints on fibres;
- New algorithm for generating smooth networks of curves aligned with given fibre systems based on the streamlines placement algorithm;

In this work we focus mainly on a concept of adaptation of the Gibbs point process framework to the case of random arrangements of curvilinear fibres. We present a substantial number of examples which support the soundness of our model of random processes of interacting fibres; however, we do not carry rigorous mathematical description, and most of the technical assumptions we introduce during the derivation of the model, including integrability of the density functions presented in this work for fibre processes, should be mathematically justified in a future work.

This chapter is organized as follows. We go over the related work and describe our early attempts to adapt the methods from the previous work for the problem of parametric synthesis of curvilinear networks in Section 3.2. None of them works well enough to satisfy our requirements for a candidate algorithm as the resulting networks of curves are not smooth in general, the synthesis models lack comprehensive and intuitive set of control, and generated networks may reveal unwanted observable repetitions. In this work we focus on a different approach: we develop a geometric stochastic process defined by a Gibbs-type probability density function

with an intuitive set of control. We have chosen Gibbs point processes as a prototype for our interaction fibre model because 1) they are intuitive[1] and can be used in a general context, 2) they simulate local interaction between the primitives and are well suited to model processes with necessary degree of regularity, 3) they are parametric and very important variants of the Gibbs random point models are enabled with statistical inference to recover their parameters, which will eventually lead to developing an algorithm for recovering parameters of fibre processes. Section 3.3 describes in detail the framework of spatial point processes: their probabilistic models of distribution functions and the Metropolis type algorithms to generate samples of point configurations. Our adapted model is based on the idea that a curve, after all, can be represented by a set of linked 2D points with attributed tangent information, so that the interaction between two curves in a vicinity of each other can be simulated by a Gibbs type interaction between 3D points which describe the curves interiors and tangents. A way of accumulating point interaction under interaction potential integrals is described in more detail in Section 3.4 for the case of general fibre processes.

We concentrate on linear fibres — a subclass of random fibres which describes a model for random systems of line segments. Linear fibre models based on interaction potentials are described in Section 3.5, while a parametric synthesis algorithm based on these models with a set of examples is presented in Section 3.6. An extensive set of user control is introduced in Section 3.7: a user can control the density of the fibres and can change the global behavior of the random system by introducing hard

[1]Gibbsian density function is based on interaction pair-potentials between closely places points, which, depending on the distance between pairs of points, favor some point configurations, prohibit other point configurations, and ignore the rest. Parameters of the Gibbsian model define a degree of interaction (see equation 3.8 and example 6.2).

and soft constraints.

The following is an outline of major steps of our synthesis algorithm:

1. *Find plausible parameters for point interaction model.* The model of inter-fibre interaction is based on integration of individual interactions between infinitesimal parts of nearby fibres (see explanation to Figures 3.13 and 3.14). Infinitesimal interactions are described by 3D-point pair-potentials — being an essential part of Gibbs random point models (Section 3.3.2) — which are represented by a piecewise constant interaction function $h_{\boldsymbol{\theta}}$ (equation 3.41). $h_{\boldsymbol{\theta}}(d,w)$ depends on the Euclidean distance between the 2D points, d, and a "distance" between orientations, w, attributed to the points. At this step, a user should find a plausible set of values for parameters of the function $h_{\boldsymbol{\theta}}$, so that it produces desired correlations of nearby fibres. Examples of interaction functions which favor fibre configurations illustrated in Figures 3.1-c and -e are shown in Figure 3.2. More rigorous examples of interaction functions based on which the actual fibre systems were generated and presented in this work are illustrated in Figure 3.27. Shaping the interaction function is a manual step — fitting the source texture feature configurations to our Gibbs linear fibre interaction model is a topic for future research;

2. *Define dimensions of linear fibre representation space.* Linear fibres are represented as points in a certain representation space, called a phase space (see Figure 3.17), which is defined by equation 3.38. At this step, the dimensions of the phase space are specified (as example of a linear fibre phase space is illustrated in Figure 3.3);

3. *Describe fibre constraints (if necessary).* A user can provide a function of distance to the constraint space, $\rho_c(\Phi)$, which describes how far a given linear fibre

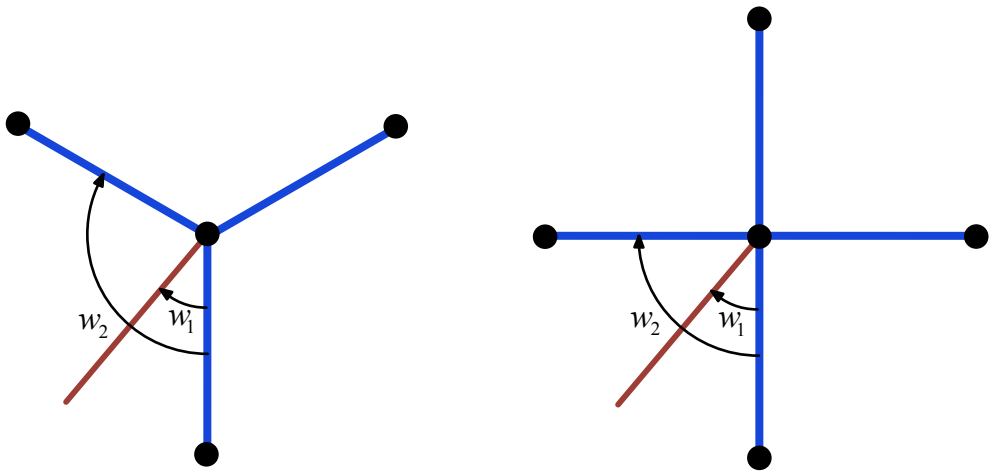


Figure 3.2: Examples of fibre configurations (blue line segments) produced by the interaction function $h_{\boldsymbol{\theta}}(d,w) = \chi_{[0,R_{\max})}(d)\left\{\chi_{\Omega}(w) + 10^{-4}\chi_{[0,\pi)\setminus\Omega}(w)\right\}$ for different sets of parameters which follow. Fibres colored in brown will be penalized by receiving a low interaction rate (10^{-4} for the angle w_1 versus 1 for the angle w_2). Left: $\Omega = [2\pi/3-\delta, 2\pi/3+\delta)$ where $\delta > 0$ is some small constant and $R_{\max}$ is some maximum interaction radius; the interaction function with such parameters favors nearby fibres which form $2\pi/3$ angles between each other, while penalizing all other fibre configurations. Right: $\Omega = [\pi/2-\delta, \pi/2+\delta)\cup[\pi-\delta,\pi)$; such function favors cross configurations of nearby fibres.

system Φ is from satisfying some constraint (equation 3.62). Alternatively, a user can use a set of constraints that we described in this work, including the total length constraints which enable the user to control the fibre density (equation 3.72), connectivity constraints which improve continuity of resulting fibres (Section 3.7.3), hard constraints which enforce aligning the fibres with a given vector field (Section 3.7.4, with examples of vector fields illustrated in Figure 3.40), and soft constraints which impose the alignment requirement

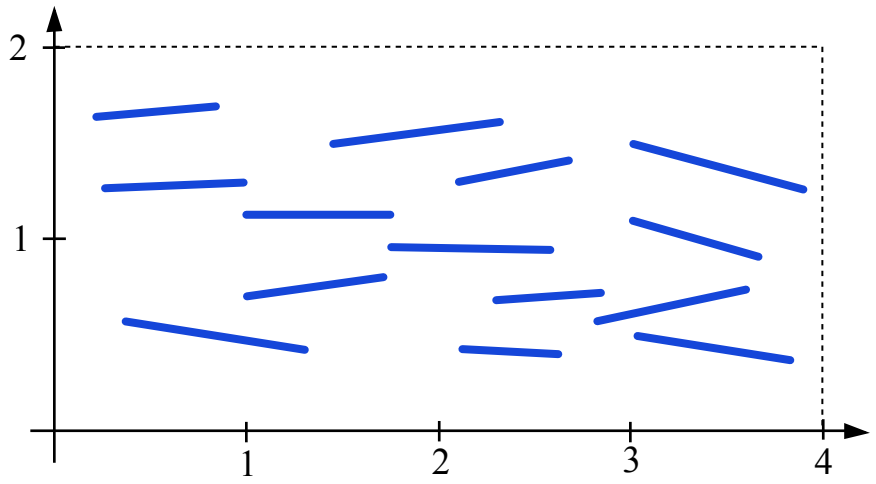


Figure 3.3: Example of a linear fibre system with fibres taken from the following phase space $\mathcal{X} = \mathcal{X}_P \times \mathcal{X}_L \times \mathcal{X}_W \equiv \{[0,4) \times [0,2)\} \times [0.5,1) \times \{[0,\pi/10) \cup [9\pi/10,\pi)\}$. $\mathcal{X}_P$ defines the domain for the fibre middle points, $\mathcal{X}_L$ defines the range of possible lengths for fibres, and $\mathcal{X}_W$ defines the range of possible angles (or, orientations) which fibres can form against the positive x-axis.

only within some distance to a given set of features (Section 3.7.5);

4. *Generate systems of random linear fibres.* We run a version of Metropolis-Hastings algorithm adapted to generate random configurations of linear fibres (original version of the algorithm for the case of point processes is thoroughly described in Section 3.3.4) distributed according some Gibbs type distribution $\pi(F)$ (equation 3.21). Gibbs $\pi(F)$ is given through its density function of the form $f(\Phi) = e^{\tilde{U}(\Phi)}/\mathcal{Z}$, where energy $\tilde{U}(\Phi)$ is a sum of fibre potentials given by equation 3.35. The algorithm simulates a discrete-time Markov chain of fibre systems and proceeds by updating current fibre system with either "birth" or "death" event, when a new fibre is added or an existing fibre is deleted, respec-

tively. Transition probabilities of the events are written in terms of conditional intensities $\lambda_c(\psi,\Phi) = f(\Phi \circ \psi)/f(\Phi)$ (equations 3.50 and 3.51) which are feasible and straightforward to calculate in the case of Gibbs type probability densities (Section 3.6.1). For optimal calculation of conditional intensities and robust addition of a new fibre to the current fibre system we developed a fibre neighborhood search query algorithm, optimized by domain cell partition, and a fibre relaxation scheme in Section 3.6.2. Linear fibre systems under constraints are generated by running a variant of simulated annealing algorithm adapted for linear fibres which is described in Section 3.7.1. Conditional intensity λ_c in this case is scaled by a factor depending on the distance to the constraint space, $\rho_c(\Phi)$ (see equations 3.65 and 3.66).

While the model of linear fibres replicates the desired local inter-fibre dependencies well, it is still more effective for generating non-smooth systems of fibres. To resolve the issue for a certain type of linear fibres, we propose an algorithm for generating networks of smooth curves aligned with a certain cross-orientation vector field. We require that this field is smooth enough and it assigns two orthogonal orientations at every point where at least one orientation is locally aligned with the linear fibre system. An obvious candidate for building such networks is a variant of the streamline placement algorithm extended for the case of two-orientation vector fields. The main problem in adapting such an algorithm is making sure that generated streamlines form two sets of nicely overlapping and coherent streamlines. The algorithm for generating and optimizing the cross-orientation vector field and a two-pass version of the streamline placement algorithm adapted to cross-orientation vector fields capable of generating coherent networks of curves is explained in Chapter 4. The algorithms for representation of and streamline generation based on multi-orientation

vector fields which cover broader spectrum of linear fibre processes are left for a future work.

3.2 Related work

The problem of texture synthesis have attracted a lot of attention across different disciplines during the last three decades. Existing texture synthesis techniques, which are related to the problem of generating networks of curvilinear features, can be roughly put into the following four main categories: procedural textures, statistics driven synthesis, sample-based texture synthesis, and sampling random processes of geometrical objects. *Procedural textures* are generated by running a relatively short set of instructions which are governed by some mathematical model [40], based on either a nice mathematical abstraction [104, 93] or on a real process in nature [139, 149, 150, 47, 142]. Our approach has some similarities with procedural texture synthesis. Our algorithm of generating random fibres is obviously resolution independent, can cover arbitrary large area with no observable repetition in the pattern of lines, and can produce unlimited number of different textures by changing the parameters of the fibre interaction model and the parameters of the synthesis. The main difference is that our technique cannot be performed in real time as it involves running the time consuming Monte Carlo simulation algorithm.

The next two groups of methods generate an arbitrary-sized new texture by trying to match the pattern in a reference (input) texture, which usually has a small size. The algorithms of *statistics driven synthesis* model textures by matching certain statistics of the input and output textures at different scales. Both the input and output textures are expanded into their image pyramids by applying a low-pass filter, so that each level of each pyramid is decomposed into a set of subbands by applying another

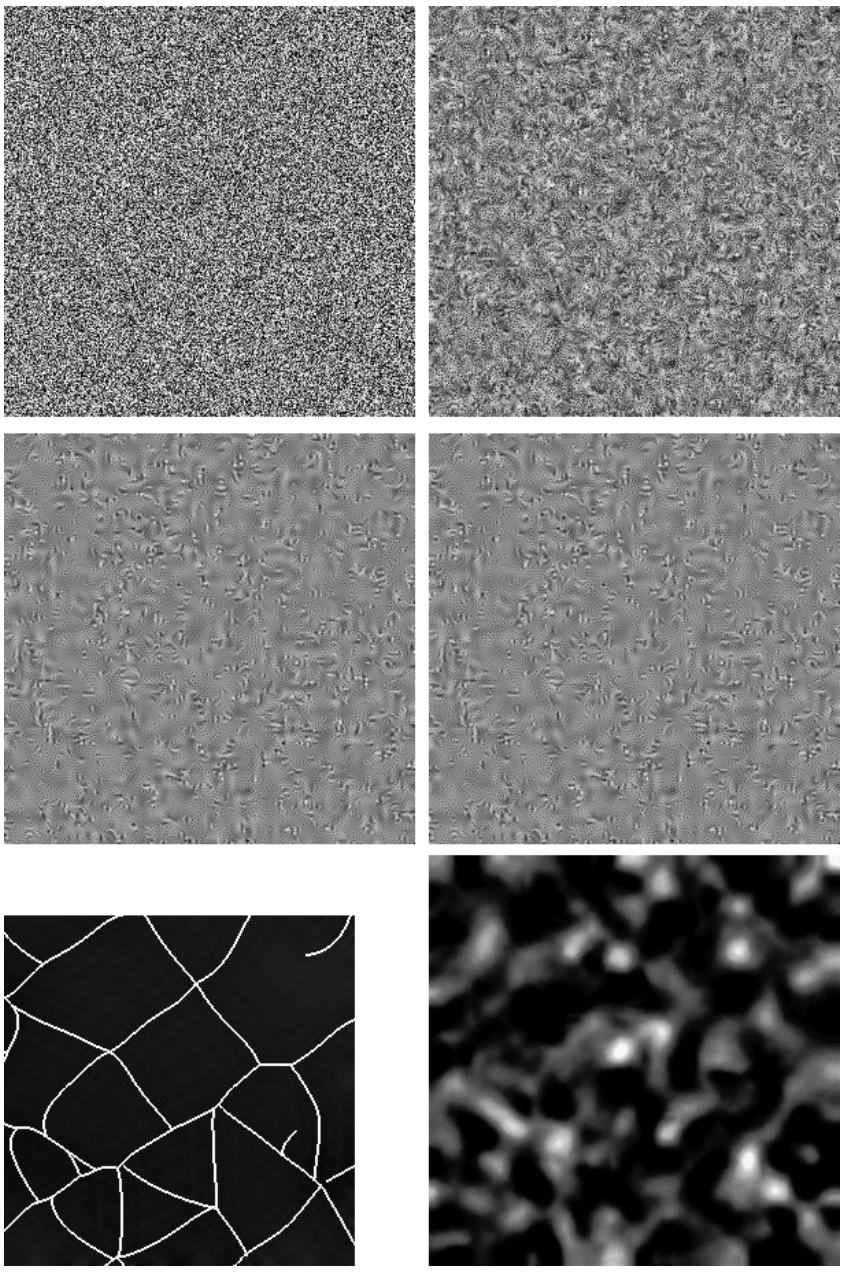


Figure 3.4: Our attempt to generate a larger texture with network of curves (bottom right image) by applying Heeger and Bergen steerable pyramid texture synthesis method [61] to a small sample texture (bottom left image). Images in the first two rows show the synthesis result at different pyramid scales.

set of filters [61]. All the subbands of the output texture are recursively updated from the coarsest to the finest level to match histograms of relative subbands of the input texture. Finally, the resulting output pyramid is collapsed to the finest scale to produce the output texture. Existing multi-scale synthesis methods are based on Laplacian or Steerable pyramids [61], on Gaussian pyramids [108], on feature-based pyramids which preserve cross-scale feature dependencies [20], on steerable pyramids [127, 128] which can also match joint statistics within each scale of the image pyramid [109], on statistical learning [13], and on some entropy-related energy minimization [158, 159]. These methods work very well on highly stochastic textures but can fail on more structured ones. As an example, we applied the steerable pyramid transform [61] to the input texture which was formed by rasterizing a fragment of the network of curves which were optimized from the skin image by our snake-based algorithm (Section 1, Figure 1.11). The method of steerable pyramid proved to be well tuned for textures which have oriented or elongated structures. However, collapsing of the finest scale in the pyramid fails to reconstruct curvilinear structures, which can be clearly seen on the bottom right image in Figure 3.4.

Sample-based texture synthesis is rooted in the assumption that the signal in a source texture is distributed according to some Markov Random Field. This implies that each pixel in such a texture depends only on the pixels within its neighborhood. Based on this property, the synthesis algorithm starts with an empty texture and fills it out, pixel by pixel, by copying at each iteration the pixel from the reference texture whose neighborhood best matches the neighborhood of a current pixel in the output [42]. Applying this strategy achieves its best results for the images with homogenous features and weakly structured patterns. Our model for random fibre processes is also local. However, we explicitly incorporate the feature correlation pattern in the mathematical model rather than just matching an output texture

to a reference texture without extracting interdependencies. The latter strategy can produce "garbage" regions in the output and finer details could be blurred. These artifacts were addressed with some success in a multi-resolution fast synthesis [147], synthesis of non-stationary textures by exploring local coherence between the pixels and by providing user control maps to control large-scale features [4], synthesis based on "k-coherent search" by building the similarity sets for each pixel in the input [137], interactive rate algorithm based on constructing jump maps during preprocessing [155]. Patch-based techniques copy entire patches from the reference texture at every iteration instead of copying pixel by pixel: new patches overwrite over existing regions [110], optimal cuts are sought through overlapping regions by applying dynamic programming [41] or the patches are stitched together along the optimal seams [73], the search for a matching patch is enhanced by matching underlying curvilinear features in the feature matching algorithm [151], parameters of a patch matching algorithm are adaptively optimized to improve synthesis efficiency [145]. Other texture synthesis techniques include global optimization which progressively refines the output texture by minimizing a mismatch energy [72, 58], synthesis of near-regular texture [79], context-aware textures [81], tile-based synthesis [25, 36], and a method of intrinsic mode functions (IMF) which decomposes the reference texture into a series of IMFs, each of which captures well directional features [157].

One of the drawbacks of all theses algorithms is that they are very sensitive to a choice of synthesis parameters which, in reality, have nothing to do with the genuine parameters of the signal, e.g., it is hard to translate a requirement for nearby curvilinear features to be at a particular angle with respect to each other into a plausible set of synthesis parameters. Another problem with the sample-based synthesis techniques is that avoiding synthesizing features explicitly will inevitably cause forming broken features in the output texture and degrading tangent continuity between connected

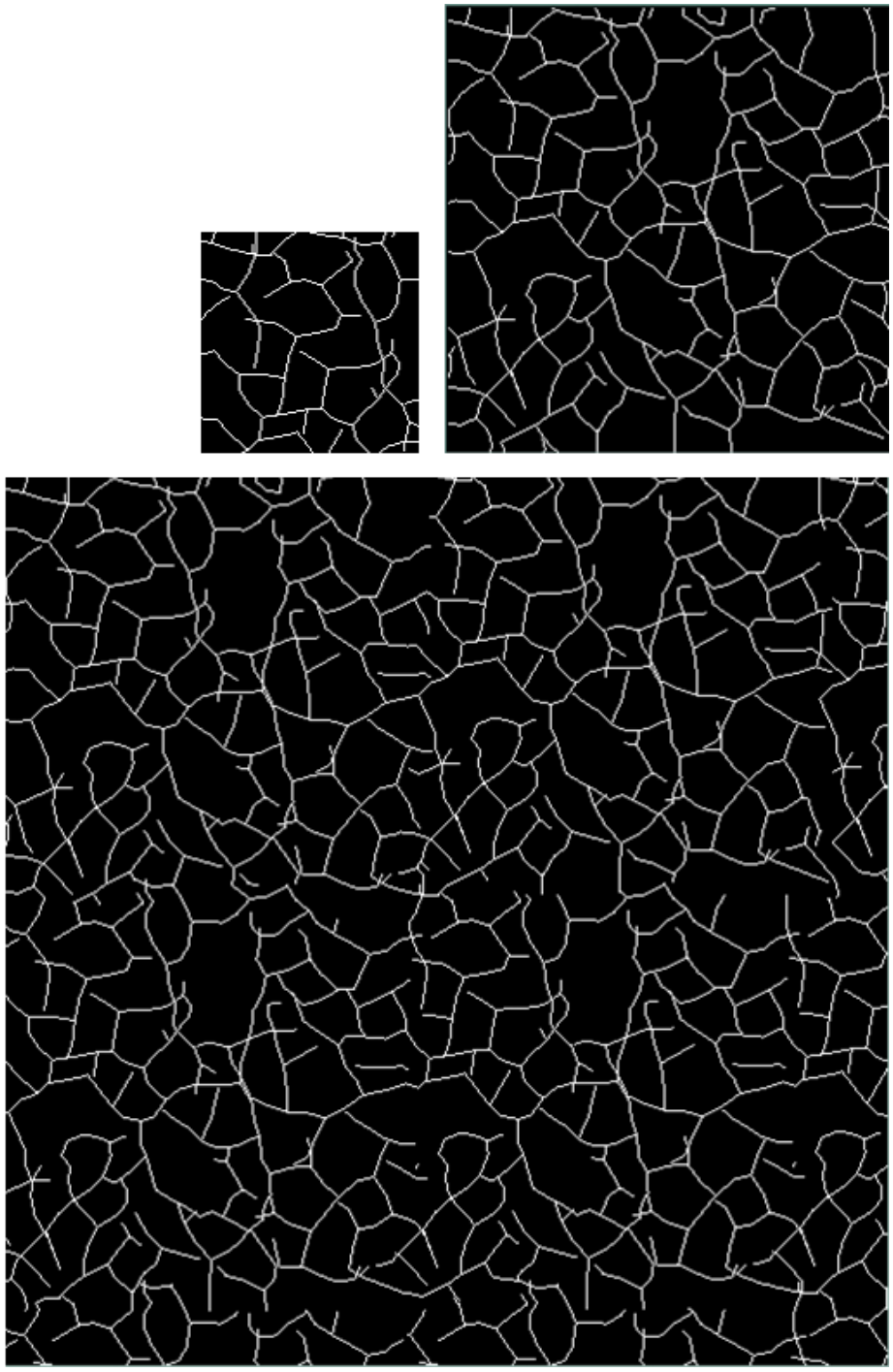


Figure 3.5: Our attempt to generate random network of curves from a small sample (top left) by applying feature matching algorithm of Wu and Yu [151].

features. To our knowledge, the only technique which addressed these artifacts is the method of feature matching and deformation by Wu and Yu [151]. According to their patch-based approach, patches are warped to ensure continuity along the curvilinear features across the patches. We applied their method to generate a larger texture (bottom image in Figure 3.5) from a small black and white reference texture with curves (top left image in Figure 3.5). The algorithm indeed created nice looking curves with a reasonable number of broken features and tangent discontinuities, which locally resembles very well the pattern of network of curves given in the reference texture. What strikes you first are the repetitions which are quite noticeable in the texture. It is not clear how to modulate the synthesis process to be able to slightly change feature configurations for producing conceptually new appearances. In our approach, we give a user more flexibility in specifying the correlation patterns between the features, so that modulation is naturally integrated in the synthesis model. We also provide a global control on synthesis.

The approach described in this work falls into the last category of texture synthesis algorithms — *random processes of geometrical objects*. Synthesized textures are outcomes of certain random processes which governs distributions of geometrical objects, like points, lines, curves, circles, triangles etc in $\mathbb{R}^d$ [14]. A typical synthesis algorithm in this category involves developing a parametric multivariate probability distribution function (pdf), which describes likelihood of occurring a set of objects at given locations, and running a randomized algorithm, like the Monte Carlo simulation, to generates random set of objects distributed according to a given pdf. The advantage of using such algorithms against the algorithms from other categories is that stochastic models are truly parametric; their parameters are easy to interpret; due to explicit randomization of the algorithm steps there is a little chance of producing textures with noticeable repetitions; and, some of the models come together

with the statistical inference algorithms which allow tune the model parameters to a given objects arrangement in the reference texture. In this work we adapt a model of Gibbs point processes (which are thoroughly described in Sections 3.3.1 and 3.3.2), which describe random arrangements of locally dependent points in $\mathbb{R}^d$, for the case of random fibres — random arrangements of line segments and curves in $\mathbb{R}^2$. In our overview of synthesis techniques for random fibres, given in Section 3.3.3, we list all existing models for generating random fibres which happened to be not powerful enough to produce fibre networks with fair amount of regularity. To the best of our knowledge, our model of random fibres is a first model which takes into account complex interdependencies between the nearby fibres and provides powerful and intuitive set of control on synthesis.

3.3 Background: random arrangements of objects in the plane

We start with describing the existing models of random arrangements of objects in the plane. Poisson point processes are key elements in defining majority of spatial point processes. We provide its definition in Section 3.3.1 and describe Gibbs point processes through a density function with respect to the Poisson process in Section 3.3.2. Fibre process is not a new concept: its simple models based on independent arrangements of lines in the plane and space has been thoroughly investigated and applied to real-world physical systems. We describe fiber processes and summarize available statistical methods for them in Section 3.3.3. Finally, we review some aspects of Markov Chain Monte Carlo (MCMC) algorithm applied for sampling spatial point processes and describe their two key representatives: MC based

on Metropolis-Hastings sampling and Birth-and-Death processes, in Section 3.3.4.

3.3.1 Framework of Poisson point processes

Poisson point process is the central element in the theory of random arrangements of objects. It is the simplest spatial point process and the majority of the existing models of point processes are based on certain underlying Poisson processes. For example, any countable random set of independent and uniformly distributed points $x = \{x_1, x_2, \ldots, x_n, \ldots\}$ in $\mathbb{R}^d$ having in average one point within a d-dimensional unit cube forms a Poisson point process with unit intensity. Without loss of generality, we consider only locally finite random arrangements of points within some bounded Borel set $X \subset \mathbb{R}^d$ (i.e., the number of points within any $B \subset X$ should be finite).

Definition 6 (Poisson point process). *Homogeneous (inhomogeneous) Poisson point process $\boldsymbol{X}$ of a constant intensity $\beta > 0$ (with varying intensity function $\beta(x) > 0$, or with some intensity measure $\lambda(B)$, $B \subseteq X$) satisfies the following two conditions:*

1. *$\boldsymbol{X}(B)$ — the number of points of $\boldsymbol{X}$ located within B — is a random variable of Poisson distribution [55] with mean value $\beta\,|B|$ ($\int \chi_B(x)\beta(x)dx$ and $\lambda(B)$, respectively).*

2. *Random variables $\boldsymbol{X}(B_1), \ldots, \boldsymbol{X}(B_K)$ are independent for disjoint sets $B_k \subseteq X$, $k = 1, 2, \ldots, K$, for any $K > 0$.*

From the definition above, one can derive that a homogeneous Poisson process $\boldsymbol{X}$, conditioned on the number of points $\boldsymbol{X}(B) = n$, generates locations of n points distributed independent and uniformly within B. Therefore, to generate samples of homogeneous Poisson process, one first chooses a number of points n which has a Pois-

son distribution with mean $\beta\,|B|$, and, second, uniformly locates the points within the set B by using, for example, the rejection method [55] with pdf $\chi_B(x)/\,|B|$.

The *rejection sampling* technique [76, 119] is used for generating random arrangements of points with inhomogeneous Poisson distribution. Instead of $\beta(x)$, which may have any level of complexity, a constant $\beta_{\max}$ is used as the intensity to generate samples of a homogeneous Poisson process provided that $\beta(x) \leq \beta_{\max}$. Every such a sample is kept with probability $\beta(x)/\beta_{\max}$, and is disregarded with probability $1-\beta(x)/\beta_{\max}$. Two examples of Poisson processes are shown on the Figure 3.7.

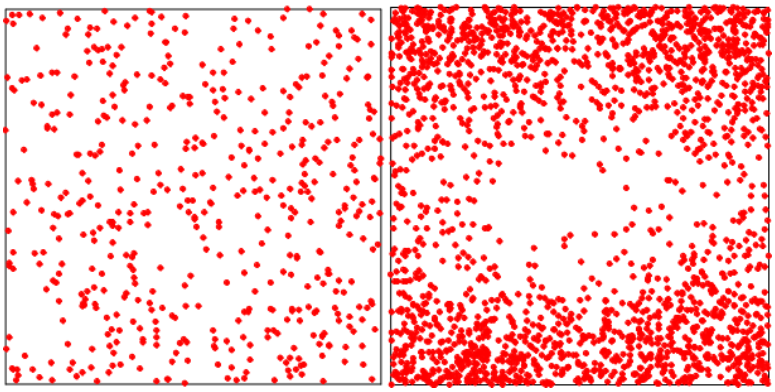


Figure 3.6: Examples of samples of homogenous Poisson process with $\beta = 5$ (on the left) versus inhomogeneous Poisson process with intensity function $\beta(x,y) = \frac{1}{2}(x-5)^2 + 2(y-5)^2$ (on the right).

The Poisson point process may be used as a basis to construct a variety of new spatial point processes by using the following three operations. *Thinning* operation deletes just generated points on the basis of some deletion probability function $0 \leq p(x) \leq 1$. *Clustering* operation consists of applying independent parent and daughter processes. The parent process generates a set of seed locations, while the copies

of the same daughter process generate random points around all seed locations with given intensity value and scattering radius. Figure 3.7 (a) shows a sample of famous Neyman-Scott clustering process [95] which was designed to represent a distribution of clusters of galaxies. A random number of daughter points, representing galaxies' locations, are uniformly scattered around the cluster locations, which in turn form a Poisson parent process of some intensity β_p. Finally, a new point process may be formed by union of two independent processes which samples do not coincide with probability one. This operation is called *superposition*. One can find more examples of derivatives of Poisson processes elsewhere [141, 130].

Not all the natural patterns can be described by totally random point processes described so far. Random locations of certain objects/particles with non-trivial area/volume which can not physically overlap form another type of random point processes (e.g., populations of biological structures like cells may be modeled by placing random points on a plane which represent the centers of the structures). Points of such processes are prohibited to be closer than some predefined distance R to each other. Because of this property, such processes are known as *hard-core point processes*. They exhibit strong regularity and, unlike Poisson processes, represent random point arrangements whose points are locally inter-dependent.

3.3.2 Gibbs point process: inter-point interactions

Poisson and hard-core point processes represent two extremes within the entire spectrum of point processes: Poisson processes show no mutual interactions between the points at all (as being realizations of independently distributed points), while hard-core processes reveal strong regularity in the resulting point arrangements because of hard-core interaction. Considering soft-core interaction "kernels" significantly in-

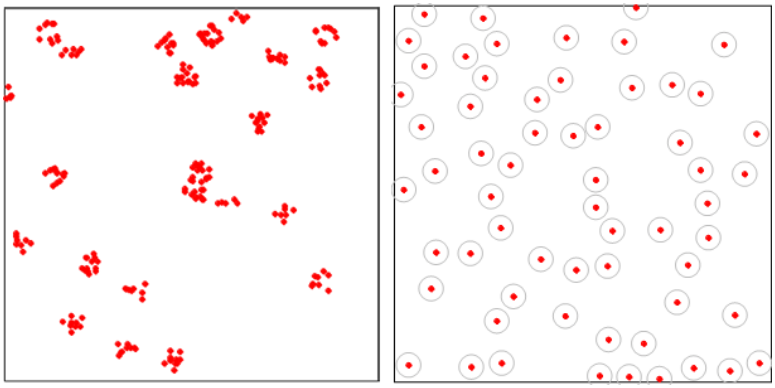


Figure 3.7: Examples of constructing new simple random processes from homogenous Poisson process. Clustering, Neyman-Scott process with $\beta = 0.25$, $R = 0.3$, and the number of daughter points 10 (on the left). Hard-core process with $\beta = 2.0$ and repulsion radius $R = 0.7$ (on the right). Circles around the points can not intersect. This process is known as Poisson-disk pattern in Computer Graphics [87, 27].

creases applicability domain of point processes [132, 69, 97, 123, 98, 32, 117, 8, 33, 9, 57, 14] allowing to model point processes with some degree of regularity, moderate clustering. Soft-core models provide an additional control for letting users to assign specific interaction profiles which describe better the underlying point processes.

In principle, a soft-core process is constructed by transforming a known distribution function of a standard point process to a new distribution being absolutely continuous with respect to the original distribution. The probability distribution of a homogenous (or inhomogeneous) Poisson point process is an obvious candidate for the original distribution because in a majority of cases it has a simple analytic form. Distribution transformations are best described within the framework of probability measures and probability densities which will be described later in this section. A prevailing group of soft-core processes are based on the probability densities of

Gibbs type which model a wide spectrum of point patterns with observable regularity by well-established simulation and statistical inference procedures.

First we define a Poisson random measure μ with unit intensity, in its general form (e.g., with domain on a general measure space). Derivative processes will be expressed through the densities defined with respect to Poisson measure μ. Formally, a Poisson point process is a random variable $\boldsymbol{X}$ which defines a measurable mapping from a probability space $(\Omega, \mathcal{A}, P)$ to an exponential measure space[22] $(X_\infty, \mathcal{F}_\infty, \mu)$.

For a compact X on $\mathbb{R}^d$, enriched by a (Lebesgue) measure $\lambda(B)$, which is defined on a σ-algebra $\mathcal{B}$ generated by bounded Borel sets over X, we consider an *exponential space* X_∞ defined by the union of all possible symmetric products of the copies of the original space X given by

$$X_\infty = \bigcup_{n=0}^{\infty} X_n, \;\; X_n = \underbrace{X \circ X \circ \ldots \circ X}_{n}. \tag{3.1}$$

Elements of the symmetric power sets X_n are *point configurations* $\bar{x}$ consisting of n different points from X:

$$\bar{x} = x_1 \circ x_2 \circ \ldots \circ x_n, \;\; N(\bar{x}) = n. \tag{3.2}$$

The order of the points within a configuration is irrelevant. We use symbols $\circ$ as separators between the points to distinguish point configurations from ordered sets of points, like vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ (to avoid any confusion with previous work notation we remind that in the original work [22] point configurations are written without separators at all, i.e., $x_1 x_2 \ldots x_n$, while the ordered sequences are denoted by $x_1 \bullet x_2 \cdots x_n$). To avoid possible complications, only finite point configurations over X are considered, i.e. $\boldsymbol{X}(B) < \infty$ for any $B \in \mathcal{B}$, and λ is a finite measure on X, i.e., $\lambda(X) < \infty$ (here, $\boldsymbol{X}(B)$ is the number of points from a random point configuration $\boldsymbol{X}$ located within $B \in \mathcal{B}$). Thus, each element $\bar{x}$ of the exponential space X_∞, $\bar{x} \in X_\infty$, is a

(unordered) collection of a finite number of distinct points $\bar{x} = x_1 \circ x_2 \circ \ldots \circ x_n,\ n < \infty$, so that X_∞ perfectly describes a simulation space for random point arrangements.

A Borel σ-algebra $\mathcal{B}$ is naturally extended to a σ-*algebra of exponential space*, $\mathcal{F}_\infty$. Elements of $\mathcal{F}_\infty$ are unions of symmetric products of a finite number of subsets from $\mathcal{B}$:

$$\begin{aligned} F &= \bigcup_{m=0}^{\infty} F_m = \bigcup_{m=0}^{\infty} F_m^1 \circ F_m^2 \circ \cdots \circ F_m^m,\ F \in \mathcal{F}_\infty, \\ \mathcal{F}_\infty &= \bigcup_{m=0}^{\infty} \mathcal{F}_m\,, \end{aligned} \tag{3.3}$$

where each $F_m^j \in \mathcal{B}$. In what follows we will use $F_m = F \cap \mathcal{F}_m$.

A Poisson measure μ , which describes a *distribution of the Poisson process* with intensity $\lambda(B)$, is defined as an infinite sum of finite measures having their supports on finite symmetric products $\mathcal{F}_m$ and is given by

$$\begin{aligned} \mu(F) &= e^{-\lambda(X)} \Big[\chi_{F\cap\mathcal{F}_0}(\emptyset) \\ &+ \sum_{n=1}^{\infty} \frac{1}{n!} \int \cdots \int \chi_{F\cap\mathcal{F}_n}(x_1 \circ x_2 \circ \ldots \circ x_n)\, \lambda(dx_1)\lambda(dx_2)\, \cdots \lambda(dx_n) \Big] \end{aligned} \tag{3.4}$$

A factor $1/n!$ is needed to take into account all possible permutations of points in collections $\{x_1, x_2, \ldots, x_n\}$. The total mass of X_n is given by $\mu_n(X_n) = \lambda_n(X_n)/n! = \lambda(X)^n/n!$, and describes a probability of finding n Poisson distributed points within a phase space X. It is easy to check that μ is a probability measure:

$$\begin{aligned} \mu(X_\infty) &= e^{-\lambda(X)} [\chi_{X_\infty}(\emptyset) \\ &\quad + \sum_{n=1}^{\infty} \tfrac{1}{n!} \int \cdots \int \chi_{X_n}(x_1 \circ x_2 \circ \ldots \circ x_n)\, \lambda(dx_1)\lambda(dx_2)\, \cdots \lambda(dx_n) \Big] \\ &= e^{-\lambda(X)} \left[1 + \sum_{n=1}^{\infty} \tfrac{\lambda_n(X_n)}{n!} \right] = e^{-\lambda(X)} \left[1 + \sum_{n=1}^{\infty} \tfrac{\lambda(X)^n}{n!} \right] = 1. \end{aligned}$$

New point process are constructed by specifying their *density functions* f with respect to the Poisson process μ. Density function defines the level of deviation of the

new process from the original Poisson process. Intuitively, $f(x_1 \circ x_2 \circ \ldots \circ x_n)\,\mu(d\bar{x})$ defines the probability of finding n points of the new point process within the increments $dx_1, dx_2, , dx_n$. Formally, for a given point process π, its density function f is a Radon-Nikodym derivative [30] of π with respect to the Poisson measure μ, written by $f = d\pi/d\mu$, provided that π is absolutely continuous with respect to μ, so that π is given by

$$\pi(F) = \int_F f(\bar{x})\mu(d\bar{x}), \quad (F \in \mathcal{F}_\infty). \tag{3.5}$$

To make sure that a transformation of Poisson distribution function μ given by Equation 3.5 results in a valid distribution function π, one should at least guarantee that the sequence of n-masses $p_n = \pi(X_n)$ sum up to 1, $\sum_{n=0}^{\infty} p_n = 1$.

The Table 3.1 lists the density functions of the most popular point process models. The following miscellaneous functions are used to describe the density functions in the table; all indices i, j run within the range $[1 \ldots N(\bar{x})]$; $x \underset{R}{\sim} y$ defines a neighborhood relation:

$$\begin{aligned} s(\bar{x}) &= \#\Big\{(i,j) \mid 1 \le i < j \le N(\bar{x}),\ \rho(x_i,x_j) \le R\Big\}, \\ t(y,\bar{x}) &= \#\Big\{i \mid 1 \le i < j \le N(\bar{x}),\ \rho(y,x_i) \le R\Big\}, \\ x \underset{R}{\sim} y &\iff \rho(x,y) \le R. \end{aligned} \tag{3.6}$$

Each density function in the Table 3.1 is associated with a set of parameters, which, in many situations, may be recovered by using likelihood inference techniques designed for parametric point processes (for example, maximum pseudolikelihood methods [5, 6, 7]).

All the point process in Table 3.1 (excluding the Poisson process) describe so-called pairwise interaction models: a pair of points from a given point configuration

Model name / Parameters	Density function
Homogeneous Poisson $\beta > 0$	$f(\bar{x}) = \beta^n \, e^{-(\beta-1)\lambda(X)}$
Inhomogeneous Poisson $\beta(x) > 0$	$f(\bar{x}) = \prod_i \beta(x_i) \, e^{-\int(\beta(x)-1)\lambda(dx)}$
Strauss (stationary) [132, 69] $\beta > 0, 0 \leq \gamma \leq 1$	$\beta > 0, \;\; f(\bar{x}) = \beta^{N(\bar{x})} \, \gamma^{s(\bar{x})} \, / \mathcal{Z}$
SoftCore (nonstationary) [98] $\beta(x) > 0, \sigma \geq 0,$ $0 \leq \kappa \leq 1$	$f(\bar{x}) = \prod_i \beta(x_i) \prod_{\substack{i<j \\ x_i \underset{R}{\sim} x_y}} \exp\left\{ -\left(\frac{\sigma}{\rho(x_i,x_j)} \right)^{2/\kappa} \right\} / \mathcal{Z}$
Lennard-Jones $\sigma > 0, \tau > 0$	$f(\bar{x}) = \prod_{\substack{i<j \\ x_i \underset{R}{\sim} x_y}} \exp\left\{ -\left(\frac{\sigma}{\rho(x_i,x_j)} \right)^{12} + \tau \left(\frac{\sigma}{\rho(x_i,x_j)} \right)^{6} \right\} / \mathcal{Z}$
Diggle Gratton [34, 32] $\delta > 0, \rho > 0,$ $\kappa > 0$	$f(\bar{x}) = \prod_{\substack{i<j \\ \delta \leq \rho(x_i,x_j) \leq \rho}} \left(\frac{\rho(x_i,x_j) - \rho}{\rho - \delta} \right)^{\kappa} / \mathcal{Z}$
Geyer [14, Ch.3] $\beta > 0, s > 0,$ $\gamma > 0$	$f(\bar{x}) = \beta^{N(\bar{x})} \prod_i \gamma^{\min\{s, t(x_i,\bar{x})\}} \, / \mathcal{Z}$

Table 3.1: Specific density functions describing point process models derived from homogenous Poisson process with unit intensity.

that are close enough to each other contribute a nontrivial pair interaction function term to a density function. *Pair interaction functions* characterizes the level of interaction (or, influence) between two points and is defined as a function of inter-point distance, $g_2(x,y) = g_2\big(\rho(x_1,x_2)\big)$. All the models in the table are instances of more general interaction model which may include the interaction terms of higher orders:

$$f(\bar{x}) = \prod_i g_1(x_i) \ \prod_{i<j} g_2(x_i,x_j) \ \cdots \ g_n(x_1,x_2,\ldots,x_n).$$

However, in this work we will only concentrate on the models of the pairwise interactions:

$$f(\bar{x}) = \prod_i g_1(x_i) \ \prod_{i<j} g_2\big(\rho(x_i,x_j)\big). \tag{3.7}$$

Gibbs point processes constitute a subclass of interaction models whose density functions belong to the exponential family. Gibbs distributions arose from the area of statistical physics and described equilibrium states of large systems of particles by specifying some model energy function U. Gibbsian density function has the following form

$$f(\bar{x}) = e^{U(\bar{x})}/\mathcal{Z}. \tag{3.8}$$

A frequently used form of the energy function is a sum of all *pair-potentials*, $h_2(x,y) = h_2\big(\rho(x,y)\big)$, which define interaction forces between the pairs of close points of a given point configuration. Again, in a more complete model of Gibbs processes, energy functional contains interaction potentials of higher order; however, we focus only on the first order (self-interaction) terms, $h_1(x)$, and on second order pair potentials, $h_2\big(\rho(x,y)\big)$:

$$U(\bar{x}) = \sum_i h_1(x_i) + \sum_{\substack{i<j \\ x_i \underset{R}{\sim} x_j}} h_2\big(\rho(x_i,x_j)\big). \tag{3.9}$$

Example 6.1 Hard core and Poisson processes

Hard code and Poisson point processes respect the Gibbs model with the following potentials

$$h_1(x) = \ln\beta,\ \ h_2(\rho) = \begin{cases} 0, & \rho > 2R \\ -\infty & \end{cases} \qquad (Hard\ core)$$

$$h_1(x) = \ln\beta,\ \ h_2(\rho) = 0 \qquad (Inhomogeneous\ Poisson)$$

Example 6.2 Inhibition and attraction processes

Two examples of the stationary Strauss model (written in Gibbs form) are inhibition and attraction processes which have the following parametric potential functions:

$$h_1(x) = \ln\beta,\ \ h_2(\rho) = \chi_{\rho<R}(\rho)\ln\gamma.$$

When $\gamma > 1$ the points tend to attract to each other when brought closer than a distance R. For $\gamma = 1$ the process becomes Poissonian with rate β. However, the model exhibits "repulsion" between the points lying at the distances smaller than R when γ is chosen to be less than 1, $1 > \gamma > 0$. And finally, $\gamma = 0$ corresponds to the hard core process. The Figure 3.8 shows examples of attracting and inhibiting Strauss point processes against a sample of the hard and a samples of Poisson process.

3.3.3 Fibre processes: general models

Fibre processes are an example of random processes of geometrical objects which simply speaking model the systems of curves randomly scattered in space. In contrast to point processes whose random primitives — points — have trivial dimensions, fibre processes generate random arrangements of objects with non-trivial measure (e.g., length of each fibre is positive) and objects can intersect each other. The former property requires a special attention.

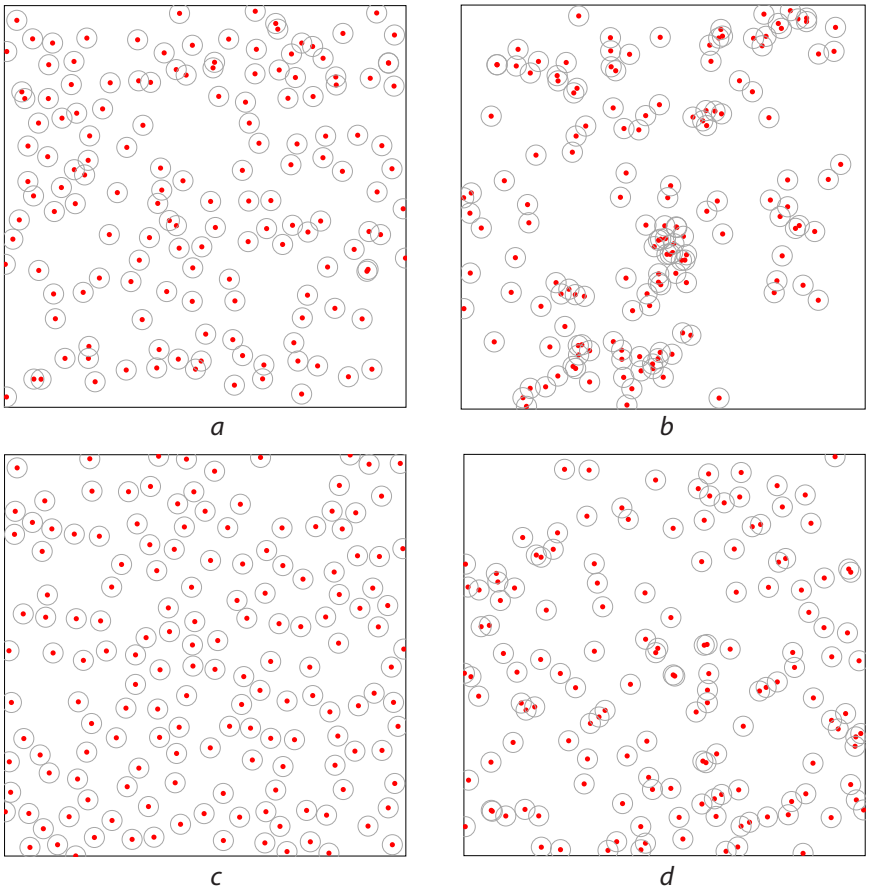


Figure 3.8: Examples of samples of two Strauss processes (a) and (b) (left on the top is inhibiting process and right on the top is attracting process), hard process (c), and Poisson process (d) with approximately the same number of points in each, 150. For all the examples, the simulation domain is $[0,10] \times [0,10]$ and interaction radius is $R = 0.5$. Parameters: (a) $\beta = 5.0$ and $\gamma = 0.2$, (b) $\beta = 0.75$ and $\gamma = 2.2$, (c) $\beta = 15.0$, (d) $\beta = 1.7$. All the points are surrounded by circles of diameter R.

Line processes — the simplest example of fibre processes which model random arrangements of lines in the plane or space — still can be described in a proper representation space. For example, a representation space for "directed" lines is a cylinder $\mathcal{X}$ [130, $\boldsymbol{C}^*$ in original notation] — a subset of $\mathbb{R}^3$ with points

$$\mathcal{X} = \left\{(\cos(\alpha), \sin(\alpha), p),\ \ \alpha \in [0, 2\pi),\quad p \in \mathbb{R}_+\right\}, \tag{3.10}$$

where α is the angle that a line makes with a chosen reference axis and p is the distance from line to the origin.

Measures on lines processes are naturally represented through the corresponding measures on the representation space $\mathcal{X}$. Infinitesimal element of a translation-invariant measure within representation space 3.10 is given by [130, Ch.8, μ in original notation]

$$\nu(dp \cdot d\alpha) = dp \cdot \varkappa(d\alpha),$$

where dp is a differential element at distance-to-origin value p, and $d\alpha$ is a differential element at angle value α, and $\varkappa$ is some measure on $(0, 2\pi]$. For measures invariant to rotations and translations, the corresponding differential measure happens to be [130, Ch.8,μ in original notation]

$$\nu(dp \cdot d\alpha) = s\, dp \cdot d\alpha,$$

for a certain constant $s > 0$. When $\nu(dp \cdot d\alpha)$ is defined, distributions of the type given by equations 3.4 and 3.5 can be constructed and be applied to describe and analyze fibre processes of randomly distributed lines.

In general, a *planar fibre process* $\boldsymbol{\mathcal{X}}$ is a random variable with values in a space of fibre systems $\mathcal{Y} = \{\Phi\}$ endowed with some σ-algebra $\tilde{\mathcal{F}}$, so that $\mathcal{Y}^{-1}(F)$, $F \in \tilde{\mathcal{F}}$, is measurable in the probability space $(\Omega, \mathcal{A}, P)$.

A *fibre system* Φ is a countable union of fibres $\{\varphi_i\}$ which could intersect one another only at their end-points

$$\forall \varphi_i, \varphi_j \in \Phi, \quad \forall p \in \varphi_i \cap \varphi_j \implies \varphi_i^{-1}(p), \varphi_j^{-1}(p) \in \{0,1\}. \tag{3.11}$$

Definition 7 (Fibre). *A fibre is a smooth simple plane curve* φ, $\varphi : [0,1] \to \mathbb{R}^2$, *without self-intersections.*

We will use the same symbol φ for fibre parametrization and for its length measure:

$$\varphi(B) = \int\limits_{\varphi^{-1}(B)} |\partial \varphi| dt.$$

A fibre system length measure $\Phi(B)$ is an equivalent to the counting measure $\boldsymbol{X}(\omega;B)$ described for point processes (see the definition 6 of Poisson process) and defines the length of that part of Φ which is located within a Borel set B

$$\Phi(B) = \sum_{\varphi_i \in \Phi} \varphi_i(B), \tag{3.12}$$

$$l_\Phi = \Phi(X) = \sum_{\varphi_i \in \Phi} l_{\varphi_i} = \sum_{\varphi_i \in \Phi} \varphi_i(X), \tag{3.13}$$

so that every fibre process $\mathfrak{X}$ is associated with a random length measure $\mathfrak{X}(B)$ with $\mathfrak{X}(\omega, B) = \Phi(B)$ for every corresponding realization $\Phi = \mathfrak{X}(\omega)$. The σ-algebra $\tilde{\mathcal{F}}$ is generated in a standard way by manipulating the sets of the forms $\{\Phi \in \mathcal{Y},\ \Phi(B) < l\}$, running through all possible Borel sets $B \subseteq X$ and the length values $l \in \mathbb{R}_+$.

A whole spectrum of statistical characteristics have been proposed so far in attempts to uniquely represent and analyze the real-world patterns with underlying random curves. They include but not limited to characteristics of the first order: intensity measures $\Lambda(B)$ and weighted random measures $\Psi(B \times L)$ which describe mean fibre length of $\mathfrak{X}$ within Borel set B having a mean value of tangential directions $w(x)$

within L [130]

$$\Lambda(B) = E_{\mathfrak{X}}\big(\mathfrak{X}(B)\big),$$
$$\Psi(B \times L) = \int_B \chi_L(w(x))\mathfrak{X}(dx),$$

and the following characteristics of the second order: (1) pair-correlation functions (PCF), $g(r)$ [131], which describe a correlation between lengths of infinitesimal fibre parts being in volume elements dV_1 and dV_2 within a distance r

$$g(r)\, dV_1 dV_2 \propto E_{\mathfrak{X}}\Big(\mathfrak{X}(dV_1) \cdot \mathfrak{X}(dV_2)\Big);$$

(2) second moment measures [130]

$$\mu^2(B_1 \times B_2) = E_{\mathfrak{X}}\Big(\mathfrak{X}(B_1)\mathfrak{X}(V_2)\Big);$$

(3) curvature measures which describe average mean curvature M and the total curvature K [3]

$$C_{\mathfrak{X},0}(B) = E_{\mathfrak{X}}\Big(K(\mathfrak{X} \cap B)\Big)$$
$$C_{\mathfrak{X},1}(B) = E_{\mathfrak{X}}\Big(M(\mathfrak{X} \cap B)\Big).$$

These characteristics were statistically estimated for a large class of Poisson line and plane fibre processes, Voronoi cell models, e.g. [131, 3], to analyze systems of dislocation lines, fibre and porous structures. The characteristics were specifically designed to be the best approximation to complicated and most of the time unknown distribution functions of fibre processes with respect to a criteria that two fibre systems with similar characteristics have similar geometric properties and similar patterns. However, all the fibre systems considered so far were assumed to be homogeneous (i.e., translation invariant) and isotropic (i.e., rotation-translation invariant) which is not the case for many variety of real-world systems with evolving patterns. The corresponding models were based on the assumption of complete randomness and independence between the fibres, and did not include any inter-fibre

interaction terms explicitly, as is done in the case of interacting point processes, equations 3.8, 3.9, and Table 3.1. What is more important is that the are no sampling algorithms which are capable of generating random fibres of systems with non-trivial inter-fibre interaction models.

In this work we develop a framework for generating random systems populated by line segments. As in the case of lines and planes, there have been a number of works on modeling the random processes of line segments and on estimating first and second order properties associated with processes [102, 44, 2, 75]; generation of samples of line segment systems were based on straightforward Boolean models of thin fibres [68]. Instead, we focus on designing a Gibbs type interaction model and the methods of generating line segment systems based on Monte Carlo Metropolis algorithm adapted to the fibre systems, the overview of which will be described in the next section.

3.3.4 Metropolis-Hastings algorithm for generating random point arrangements

Generating samples of random point processes specified by non-Poissonian density functions is generally not feasible with only rejection sampling, thinning, clustering, and superposition operation at hand. Except the trivial situations, Gibbsian density functions 3.8 contain partition functions $\mathcal{Z}$ which typically cannot be determined explicitly — this makes applying standard sampling techniques impossible. Instead, majority of spatial point processes are simulated by using models based on the conditional intensities: the calculation of the partition functions in these methods are avoided and sampling algorithms usually proceed with simple steps by updating a current sample to be distributed closely to a target distribution. Two methods

of this type were successfully applied to simulate arrangements of random points: Metropolis-Hastings (MH) [52] and birth-and-death (BnD) [112] algorithms. Both methods are based on sampling certain Markov processes which generate sequences (or, families) of dependent random variables $\{\mathsf{X}_t, t \geq 0\}$ — Markov chains — with the following Markov condition on the process history:

$$Q(\mathsf{X}_{t+\Delta t} \in F | \mathsf{X}_{t_1} = \mathsf{x}_1, \mathsf{X}_{t_2} = \mathsf{x}_2, \ \ldots, \mathsf{X}_{t_k = t} = \mathsf{x}) = Q(\mathsf{X}_{t+\Delta t} \in \ F | \mathsf{X}_t = \mathsf{x}) \,,$$

for any $F \in \mathcal{F}_\infty$ and $t_1 < t_2 < \ldots < t_k = t < t + \Delta t$, $\mathsf{x}_{k'} \in X_\infty$ for $k' = 1, 2, \ldots, k$, such that the transition probability $Q(\cdot|\cdot)$ on the RHS is not trivial.

Metropolis-Hastings (MH) algorithm is a part of a more general framework of Markov Chain Monte Carlo (MCMC) technique which generates discrete-time Markov chains $\{\mathsf{X}_k, k \geq 0\}$ by performing uniform updates of a current state, $\mathsf{X}_k \to \mathsf{X}_{k+1}$, according to a transition probability density $Q(d\mathsf{y}|\mathsf{x}) = Q(\mathsf{X}_{k+1} \in d\mathsf{y} | \mathsf{X}_k = \mathsf{x})$. The objective of MH-algorithm is to design a transition probability function, so that in the limit X_k becomes distributed as a target distribution function π

$$Q^{(k)}(F) = \int\limits_{X_\infty} Q(\mathsf{X}_k \in F | \mathsf{X}_0 = \mathsf{x}) d\mathsf{x} \ \to \pi(F), \ \ F \in \mathcal{F}_\infty \,. \tag{3.14}$$

Markov chain update consists of drawing a value $\mathsf{y} \in X_\infty$ for the next chain's state distributed according to a *proposal Markov kernel*, $P(F|\mathsf{x}) \equiv P(\mathsf{y} \in F|\mathsf{x})$, for a given current state $\mathsf{X}_k = \mathsf{x} \in X_\infty \setminus F$, and keeping this value with *acceptance probability* $A(\mathsf{y}|\mathsf{x})$. The transition probabilities in this case are given by

$$Q(F|\mathsf{x}) = \int\limits_{\mathsf{y} \in F} A(\mathsf{y}|\mathsf{x}) \, P(d\mathsf{y}|\mathsf{x}) \,,$$

$$Q(\{\mathsf{x}\}|\mathsf{x}) = P(\{\mathsf{x}\}|\mathsf{x}) + \int\limits_{\mathsf{y} \neq \mathsf{x}} [1 - A(\mathsf{y}|\mathsf{x})] \, P(d\mathsf{y}|\mathsf{x}) \,.$$

Proposal kernel $P(\cdot|\cdot)$ and acceptance probability $A(\cdot|\cdot)$ should be chosen so that the resulting Markov chain is time-reversible with respect to the target point process measure π, given by its pdf $f(\cdot)$ with respect to a Poisson measure μ

$$\int_{G_{n+1}} Q(F_n|\mathsf{y}) f(\mathsf{y}) \mu(d\mathsf{y}) = \int_{F_n} Q(G_{n+1}|\mathsf{x}) f(\mathsf{x}) \mu(d\mathsf{x}) . \tag{3.15}$$

Satisfying this condition together with requiring π-irreducibility [$\forall \mathsf{x} \in X_\infty, F \in \mathcal{F}_\infty | \pi(F) > 0$, $P^{(k)}(F|\mathsf{x}) = P(\mathsf{X}_k \in F | \mathsf{X}_0 = \mathsf{x}) > 0$ for some $k > 0$] and to be a positive Harris recurrent [$P(\mathsf{X}_k \in F \text{ for infinitely many } k | \mathsf{X}_0(\mathsf{x}) = 1]$ guarantee the convergence 3.14 from all initial distributions.

Metropolis-Hastings algorithm for point processes $\boldsymbol{X}$ on X_∞ is based on the following steps:

1. Start an empty point configuration: $\mathsf{X}_0 = \emptyset \in X_\infty$;

2. After a point configuration for X_k has been found, $k \geq 0$, choose with probability $q(\mathsf{X}_k)$ to generate a new point $\xi \in X$, $\xi \sim b(\xi, \mathsf{X}_k)$ — some pdf with respect to $\lambda(d\xi)$, and accept the transition $\mathsf{X}_k \to \mathsf{X}_{k+1} = \mathsf{X}_k \circ \xi$ with probability $A(\mathsf{X}_{k+1}|\mathsf{X}_k) = \min\{1, r(\xi, \mathsf{X}_k)\}$;

3. Otherwise, delete an existing point $x_j \in \mathsf{X}_k$ with probability $d(x_j, \mathsf{X}_k \setminus x_j)$; accept the transition $\mathsf{X}_k \to \mathsf{X}_{k+1} = \mathsf{X}_k \setminus x_j$ with probability $A(\mathsf{X}_{k+1}|\mathsf{X}_k) = \min\{1, 1/r(x_j, \mathsf{X}_k \setminus x_j)\}$;

The transition likelihood function $r(\xi, \mathsf{X})$ is a core element of MH scheme and is given by

$$r(\xi, \mathsf{X}) = \lambda_c(\xi, \mathsf{X}) \frac{1 - q(\xi \circ \mathsf{X})}{q(\mathsf{X})} \frac{d(\xi, \mathsf{X})}{b(\xi, \mathsf{X})},$$
$$\lambda_c(\xi, \mathsf{X}) = \frac{f(\mathsf{X} \circ \xi)}{f(\mathsf{X})},$$

where $\lambda_c(\xi, \mathsf{X})$ is the *conditional intensity*. It is easy to check that with such transition likelihood function, the following *detailed balance equation* (DBE) is satisfied

$$[1 - q(\mathsf{x} \circ \xi)]\, d(\xi, \mathsf{x})\, A(\mathsf{x} | \mathsf{x} \circ \xi)\, f(\mathsf{x} \circ \xi) \\ = q(\mathsf{x})\, b(\xi, \mathsf{x})\, A(\mathsf{x} \circ \xi | \mathsf{x})\, f(\mathsf{x})\,. \tag{3.16}$$

As the detailed balance equation 3.16 is satisfied, so is the time-reversibility condition 3.15. Indeed, for just defined $A(\cdot|\cdot)$ and $P(\cdot|\cdot)$, $\mathsf{x}, \mathsf{y} \in X_\infty$ and $\mathsf{y} = y_1 \circ y_2 \circ \ldots \circ y_N$, we have

$$\int\limits_{G_{n+1}} Q(F_n|\mathsf{y})\, f(\mathsf{y})\, \mu(d\mathsf{y}) = \\ = \frac{e^{-\lambda(S)}}{(n+1)!} \int\limits_{G_{n+1}} \lambda(d\mathsf{y})\, (1 - q(\mathsf{y})) \sum_{j=1}^{n+1} \chi_{F_n}(\mathsf{y} - y_j) d(y_j, \mathsf{y} - y_j) A(\mathsf{y} - y_j | \mathsf{y}) f(\mathsf{y}) \\ = \frac{e^{-\lambda(S)}}{(n+1)!} \sum_{j=1}^{n+1} \int\limits_{G_{n+1}} \lambda(d\mathsf{y})\, (1 - q(\mathsf{y}))\ \chi_{F_n}(\mathsf{y} - y_j) d(y_j, \mathsf{y} - y_j) A(\mathsf{y} - y_j | \mathsf{y}) f(\mathsf{y})$$

As there is no principal difference between y_j-s within F_n, the integrals in the sum-

mation should be the same; therefore

$$= \frac{e^{-\lambda(S)}}{n!} \int_{G_{n+1}} \lambda(d\mathsf{y})\ (1-q(\mathsf{y}))\ \ \chi_{F_n}(\mathsf{y}-y_j)d(y_j,\mathsf{y}-y_j)A(\mathsf{y}-y_j|\mathsf{y})f(\mathsf{y})$$

$$= \frac{e^{-\lambda(S)}}{n!} \int_{G_{n+1}} \lambda(d\mathsf{x})\ \lambda(dy_j)\ (1-q(\mathsf{x}\circ y_j))\ \ \chi_{F_n}(\mathsf{x})d(y_j,\mathsf{x})A(\mathsf{x}|\mathsf{x}\circ y_j)f(\mathsf{x}\circ y_j)$$

$$= \frac{e^{-\lambda(S)}}{n!} \int \lambda(d\mathsf{x})\ \lambda(dy_j)\ (1-q(\mathsf{x}\circ y_j))\ \ d(y_j,\mathsf{x})A(\mathsf{x}|\mathsf{x}\circ y_j)f(\mathsf{x}\circ y_j)$$

$$\times \chi_{F_n\times G_{n+1}}(\mathsf{x}\times(\mathsf{x}\circ y_j)) \ = [\text{DBE } 3.16 \text{ is satisfied}]$$

$$= \frac{e^{-\lambda(S)}}{n!} \int \lambda(d\mathsf{x})\ \lambda(dy_j)\ q(\mathsf{x})\ \ b(y_j,\mathsf{x})A(\mathsf{x}\circ y_j|\mathsf{x})f(\mathsf{x})\ \chi_{F_n\times G_{n+1}}(\mathsf{x}\times(\mathsf{x}\circ y_j))$$

$$= \frac{e^{-\lambda(S)}}{n!} \int \lambda(d\mathsf{x})\ \chi_{F_n}(\mathsf{x}) \int \lambda(dy_j)\ q(\mathsf{x})\ \ b(y_j,\mathsf{x})A(\mathsf{x}\circ y_j|\mathsf{x})f(\mathsf{x})\ \chi_{G_{n+1}}(\mathsf{x}\circ y_j)$$

$$= \frac{e^{-\lambda(S)}}{n!} \int_{F_n} \lambda(d\mathsf{x}) \int_{\mathsf{x}\circ y_j\in G_{n+1}} \lambda(dy_j)\ q(\mathsf{x})\ \ b(y_j,\mathsf{x})A(\mathsf{x}\circ y_j|\mathsf{x})f(\mathsf{x})$$

We can proceed with derivation if we use $\mu(dx)|_{\mathsf{x}\in F_n} = \frac{e^{-\lambda(S)}}{n!}\lambda(dx)$

$$= \int_{F_n} \mu(d\mathsf{x}) \int_{\mathsf{x}\circ y_j\in G_{n+1}} \lambda(dy_j)\ q(\mathsf{x})\ \ b(y_j,\mathsf{x})A(\mathsf{x}\circ y_j|\mathsf{x})f(\mathsf{x})$$

$$= \int_{F_n} \mu(d\mathsf{x})\, f(\mathsf{x})\, q(\mathsf{x}) \int_{\mathsf{x}\circ y_j\in G_{n+1}} \lambda(dy_j)\, b(y_j,\mathsf{x})\, A(\mathsf{x}\circ y_j|\mathsf{x})$$

$$= \int_{F_n} Q(G_{n+1}|\mathsf{x})\, f(\mathsf{x})\, \mu(d\mathsf{x})\,.$$

This causes any Markov chain simulated by the Metropolis-Hastings algorithm described earlier in this section to converge, as in 3.14.

One can also include replacement transitions to the Markov chain simulation which displace one point within a current configuration at every iteration as follows: randomly pick a point $x_j \in \mathsf{X}_k$, replace it with a new point y uniformly distributed in X with the acceptance probability $A_r(\mathsf{X}_k \setminus x_j \circ y|\mathsf{X}_k) = \min\{1, f(\mathsf{X}_k \setminus x_j \circ y)/f(\mathsf{X}_k)\}$. This new transition defines a new transition probability function $Q_r(\cdot|\cdot)$. A combined

simulation of a Markov chain performs an MH-step with some probability q_{bd} or a replacement step with the probability $1-q_{bd}$, so that the overall transition probability is given by the following mixture of MH transition kernel Q_{bd} and replacement transition kernel Q_r

$$Q_{p_{bd}}(F|\mathrm{x}) = (1-p_{bd})Q_r(F|\mathrm{x}) + p_{bd}Q_{bd}(F|\mathrm{x})\,. \tag{3.17}$$

According to the Hammersley-Clifford theorem [118], a point process is a Markov point process if and only if its density function can be decomposed into the following product

$$f(\bar{x}) = \prod_{\text{cliques } \bar{y}\subseteq\bar{x}} g(\bar{y})\,. \tag{3.18}$$

The pairwise point interaction models 3.7 surely satisfy the Hammersley-Clifford condition 3.18, and so do Gibbs point processes 3.8 in the form of interaction potentials 3.9 (here, cliques are defined through the neighborhood relation 3.6 and for a non-clique $\bar{y}$ the equality $g(\bar{y}) = 1$ should hold). The functions in the Table which follows define an MCMC algorithm for simulating a Gibbs point process.

MH for Gibbs point processes

$p_{bd} = 1$	no replacement transitions
$q(\bar{x}) \equiv q_b$	Probability of "birth" event
$b(x,\bar{x}) = \frac{1}{\lambda(X)}$	pdf for a new point candidate
$d(x_j,\bar{x}-x_j) = \frac{1}{n+1}$	Probability to delete x_j from $\bar{x}$
$r(x,\bar{x}) = \lambda_c(x;\bar{x})\frac{1-q_b}{q_b}\frac{\lambda(X)}{n+1}$	"birth" transition likelihood
$r(x_j,\bar{x}-x_j) = \lambda_c(x_j;\bar{x}-x_j)\frac{1-q_b}{q_b}\frac{\lambda(X)}{n}$	"death" transition likelihood
$\lambda_c(y,\bar{x}) = \exp\left\{-h_1(y) - \sum_{\substack{y\neq x_j \\ y\underset{R}{\sim}x_j}} h_2\big(\rho(y,x_j)\big)\right\}$	conditional intensity

Spatial birth-and-death (BnD) process simulates continuous-time Markov chains with the same birth and death transitions as in MH but with waiting times till the next event — sojourn times, t_{k+1} — being exponentially distributed, $t_{k+1} \sim \mathrm{Exp}\big(BD(\mathsf{X}_{t_k})\big)$ and $BD(\mathsf{X}_{t_k}) \equiv B(\mathsf{X}_{t_k}) + D(\mathsf{X}_{t_k})$. All the transitions are always accepted, and the birth event is chosen with probability $B(\mathsf{X}_{t_k})/BD(\mathsf{X}_{t_k})$ according to $b(\xi, \mathsf{X}_{t_k})$ while the death event is chosen with probability $D(\mathsf{X}_{t_k})/BD(\mathsf{X}_{t_k})$ according to $d(x_j, \mathsf{X}_{t_k} \setminus x_j)$. Here, the total birth rate is given by $B(\mathsf{X}_{t_k}) = \int_X b(x, \mathsf{X}_{t_k})\, d\lambda(x)$ while the total death rate is given by $D(\mathsf{X}_{t_k}) = \sum_{x_j \in \mathsf{X}_{t_k}} d(x_j, \mathsf{X}_{t_k} \setminus x_j)$. $b(\xi, \mathsf{X})$ and $d(\xi, \mathsf{X})$ should be chosen to satisfy the following birth-and-death balance equation: $b(\xi, \mathsf{X}) f(\mathsf{X}) = d(\xi, \mathsf{X}) f(\xi \circ \mathsf{X})$. More details can be found in [112]. The main difference between MH and BnD steps is that the latter updates a current configuration X_k at every step while former may reject a transition; however, to construct an update itself can be a lot trickier for the latter than for the former process. Some comparisons between running MH and BnD algorithms for Strauss point processes were presented in [24].

It is very important to mention here that a convergence of an MCMC simulated by MH algorithm to a target distribution π is pretty much everything one can guarantee — an MH-based simulation should run infinite time to generate a true sample from π. The problem of when to stop iterations is still open for MH algorithms. On the other hand, an exact (or, perfect simulation) MCMC [113] algorithm delivers a sample attained to the target distribution in finite time, though the running time is random. Examples of perfect simulation algorithms for spatial BnD processes are described in [64, 71]. Perfect simulation algorithms for both MH and birth-and-death processes based on dominated Coupling-from-the-Past (CFTP) were developed in [70] for point processes satisfying the following *local stability condition*: $f(\bar{x} \circ y) \leq K f(\bar{x})$ for a constant $K > 0$.

3.4 New model of interacting random fibres

In this section we describe steps of adapting Gibbs distributions to random fibre systems of general form. We define probability measure for fibre processes of the form similar to equations 3.4 and 3.5, and will impose an energy conservation requirement for fibre energy function in Section 3.4.1. Sections 3.4.2 and 3.4.3 describe our pair-potential model for fibre interaction and other possible terms within fibre energy functional. A particular implementation of these ideas for line segment processes will be developed in Section 3.5.

3.4.1 Adaptation of Gibbs point process models to random fibres; energy conservation requirement

We have chosen to explore interaction models for random fibres because of a number of advantages this type of models could offer: they are easy to interpret and are relatively convenient to work with, they break the assumption of independent observations, they fit the framework of Markov models which include straightforward methods of generating samples of processes with local interaction between the process primitives, and, finally, the statistical interference algorithms for point interaction models can be potentially adapted to interaction fibre processes which will make the fibre models fully parametric. Therefore, the following features should be developed for a possible candidate to a fibre interaction model:

1. fibres should closely follow definition 7 and a proper representation space for fibre systems has to be found where the fibres are uniquely represented — different fibres correspond to different ”points” in the representation space; a fibre system equivalent of a “point configuration” should be defined;

2. to define neighborhood relation on fibres, a valid function of "distance" between two fibres should be obtained, which is necessary to work with Markov models;

3. interaction potentials should be defined for fibres located nearby.

In this section we assume that a proper representation space $\mathcal{X}$ describing fibres is already given, and we will consider fibres of a general form, $\varphi \in \mathcal{X}$, in accordance with definition 7. A fibre system Φ is a union of countably many fibres with only their end-points in common

$$\Phi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\},$$

and Φ is an element of a corresponding exponential space $\mathcal{X}_\infty$. A permutation of fibres in a fibre system clearly does not change the system. Therefore, we can use the same notation which were introduced for point configurations (see equation 3.2) to show members of a fibre system

$$\Phi = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n, \quad N(\Phi) = n, \tag{3.19}$$
$$\varphi_j \in \Phi, \ \forall j \in 1, 2, \ldots, n.$$

If the fibre representation space $\mathcal{X}$ is endowed with a proper measure ν, a measure space for fibre systems $(\mathcal{X}_\infty, \widetilde{\mathcal{F}}_\infty, \mu)$ may be defined in the same way as was done for point processes (see equation 3.4). Thus, a Poisson model with intensity $\nu(B)$ for independently distributed fibres is given by the following probability distribution μ

$$\begin{aligned} \mu(F) &= e^{-\nu(\mathcal{X})} \Big[\chi_{F \cap \widetilde{\mathcal{F}}_0}(\emptyset) \\ &+ \sum_{n=1}^{\infty} \frac{1}{n!} \int \cdots \int \chi_{F \cap \widetilde{\mathcal{F}}_n}(\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n) \ \nu(d\varphi_1)\nu(d\varphi_2) \ \cdots \nu(d\varphi_n) \Big], \end{aligned} \tag{3.20}$$

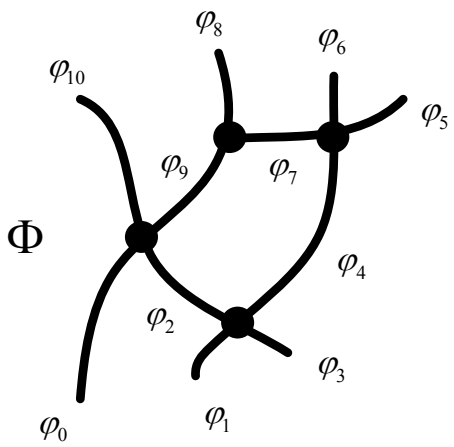


Figure 3.9: A simple fibre system $\Phi = \varphi_0 \circ \varphi_1 \circ \ldots \circ \varphi_{10}$.

for a given set of fibre systems F, $F \in \widetilde{\mathcal{F}}_\infty = \bigcup_{m=0}^{\infty} \widetilde{\mathcal{F}}_m$.

New fibre processes can be constructed from Poisson fibre process μ by means of symmetric density functions as was done for point processes in equation 3.5

$$\pi(F) = \int_F f(\Phi)\mu(d\Phi), \quad (F \in \widetilde{\mathcal{F}}_\infty). \tag{3.21}$$

Therefore, an analogue of the *Gibbs distributions for random fibres* will be defined through a probability density function of the following form

$$f(\Phi) = e^{\tilde{U}(\Phi)}/\mathcal{Z}, \tag{3.22}$$

with a proper partition function $\mathcal{Z}$.

We propose using a fibre pair-potential $\tilde{h}_2(\varphi_i, \varphi_j)$ which characterizes the level of interaction between a given pair of fibres φ_i and φ_j, and which is zero for the fibres with the distance $\tilde{\rho}(\varphi_i, \varphi_j) > R$. A sum of pair-potentials of all pairs of fibres in close proximity gives rise to one possible expression for the total energy of the fibre

system

$$\tilde{U}(\Phi) = \tilde{U}(\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n) = \sum_{\substack{i<j \\ \varphi_i \underset{R}{\sim} \varphi_j}} \tilde{h}_2(\varphi_i, \varphi_j), \tag{3.23}$$

where a neighborhood relation $\underset{R}{\sim}$ is based on a fibre distance functional $\tilde{\rho} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which should be properly defined. Table 3.2 lists new notation for Gibbs fibre processes over the corresponding notation for Gibbs point processes.

Notation	Fibre processes	Point processes
Measure space of objects	$(\mathcal{X}, \widetilde{\mathcal{B}}, \nu)$	$(X, \mathcal{B}, \lambda)$
Object configuration	$\Phi = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n$	$\bar{x} = x_1 \circ x_2 \circ \ldots \circ x_n$
Measure space of object configurations	$(\mathcal{X}_\infty, \widetilde{\mathcal{F}}_\infty, \mu)$	$(X_\infty, \mathcal{F}_\infty, \mu)$
Random process of object configurations	$\boldsymbol{\mathcal{X}}$	$\boldsymbol{X}$
Density function, f	$e^{\tilde{U}(\Phi)}/\mathcal{Z}$	$e^{U(\bar{x})}/\mathcal{Z}$
Energy of population, U	$\sum_{\substack{i<j \\ \varphi_i \underset{R}{\sim} \varphi_j}} \tilde{h}_2(\varphi_i, \varphi_j)$	$\sum_{\substack{i<j \\ x_i \underset{R}{\sim} x_j}} h_2\big(\rho(x_i, x_j)\big)$
Length / Counting measure	$\Phi(B)$, $B \in \widetilde{\mathcal{B}}$	$\bar{x}(B)$, $B \in \mathcal{B}$

Table 3.2: Corresponding notation of Gibbs fibre processes and Gibbs point processes.

A crucial difference between the point processes and the fibre processes is in that the point configurations are uniquely defined up to permutation, whereas the fibre systems may have infinitely many representations. Indeed, for a given fibre system $\Phi = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n$, one could subdivide a fibre φ_j into two parts, $\varphi_j =$

$\varphi'_j \cup \varphi''_j$, to give rise to a new fibre set $\Phi' = \varphi_1 \circ \cdots \circ \varphi_{j-1} \circ \varphi'_j \circ \varphi''_j \circ \varphi_{j+1} \circ \cdots \circ \varphi_n$ which is identical to Φ in terms of point-by-point correspondence. (To save space we introduce a new notation $\Phi_{\hat{i}} = \Phi \setminus \varphi_i$. Thus, Φ' may be alternatively defined as $\Phi' = \Phi_{\hat{i}} \circ \varphi'_j \circ \varphi''_j$.)

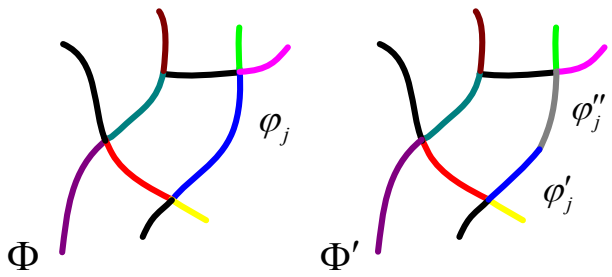


Figure 3.10: Φ' is one of uncountably many alternative representations of Φ.

Energy conservation. Such observation imposes a unique requirement on a candidate for the energy function $\tilde{U}(\Phi)$: the value of $\tilde{U}(\Phi)$ should be the same for any valid partition $\varphi'_1 \circ \varphi'_2 \circ \ldots \circ \varphi'_m$ of a fibre system Φ, i.e.

$$\Phi = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n = \varphi'_1 \circ \varphi'_2 \circ \ldots \circ \varphi'_m,$$
$$\tilde{U}(\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n) = \tilde{U}(\varphi'_1 \circ \varphi'_2 \circ \ldots \circ \varphi'_m). \tag{3.24}$$

To understand how the requirement given by equation 3.24 affects the energy function given by equation 3.23, we consider the simplest case when only one fibre φ_i, $\varphi_i \in \Phi$, is subdivided into two parts, $\varphi_i = \varphi'_i \circ \varphi''_i$. To satisfy the total energy conservation condition 3.24 in the presence of a new fibre system $\Phi_{\hat{i}} \circ \varphi'_j \circ \varphi''_j$, the following *additivity condition on the fibre pair-potential function* $\tilde{h}_2$ should hold

$$\tilde{h}_2(\varphi_i, \varphi_j) = \tilde{h}_2(\varphi'_i, \varphi_j) + \tilde{h}_2(\varphi''_i, \varphi_j),$$

for any neighbor φ_j ($\varphi_j \in \Phi,\ \varphi_i \underset{R}{\sim} \varphi_j$), and

$$\tilde{h}_2(\varphi_i', \varphi_i'') = 0\,. \tag{3.25}$$

The latter condition basically prohibits self-interaction of the fibres. In the models which allow self-interaction the additivity condition becomes

$$\tilde{h}_2(\varphi_i, \varphi_j) = \tilde{h}_2(\varphi_i', \varphi_j) + \tilde{h}_2(\varphi_i'', \varphi_j) + \tilde{h}_2(\varphi_i', \varphi_i'')\,. \tag{3.26}$$

These cases are demonstrated in Figure 3.11.

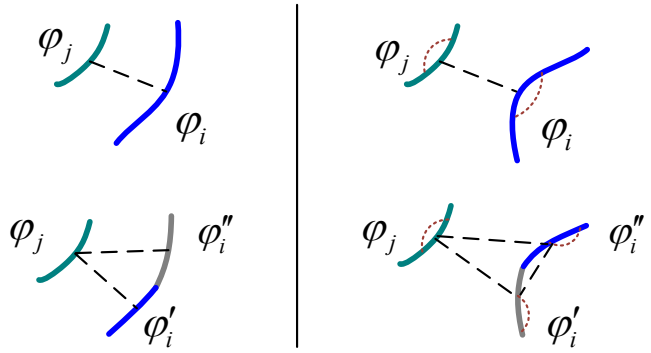


Figure 3.11: Schematic interaction links (broken line) between fibres φ_i and φ_j before (top pictures) and after (bottom pictures) partitioning φ_i. Left: no self-interaction. Right: with self-interaction (red dots).

The additivity condition restricts the choice of the pair-potential model. Similar to the Gibbs point process model which describes pair-potentials as a function of distance between the points, one can also try to develop a fibre pair-potential as a multivariate function of a vector of distances $\tilde{\boldsymbol{\rho}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^k$ (with k possible greater than one), $\tilde{h}_2(\varphi, \psi) = \tilde{h}_2\big(\tilde{\boldsymbol{\rho}}(\varphi, \psi)\big)$. However, the following argument cast doubts that such candidate can satisfy additivity condition. Indeed, by making a fibre pair-potential a homogeneous function along each of its argument,

e.g., $\tilde{h}_2(ud, w, \dots) = u\tilde{h}_2(d, w, \dots)$ for some scalar $u > 0$, and by requiring the first component of the distance function $\tilde{\boldsymbol{\rho}}$, $\tilde{\boldsymbol{\rho}} = (\tilde{\rho}_d, \tilde{\rho}_w, \dots)$, to be also homogenous $\tilde{\rho}_d(v\varphi, \psi) = v\tilde{\rho}_d(\varphi, \psi)$, one can guarantee fulfilling the additivity condition when one of the fibres is broken in two equal pieces:

$$\begin{aligned}\tilde{h}_2\big(\tilde{\boldsymbol{\rho}}(\varphi, \psi)\big) = \; & 2\tilde{h}_2\big(1/2\,\tilde{\boldsymbol{\rho}}(\varphi, \psi)\big) \; = 2\tilde{h}_2\big(\tilde{\boldsymbol{\rho}}(\varphi/2, \psi)\big) \\ & = \tilde{h}_2\big(\tilde{\boldsymbol{\rho}}(\varphi/2, \psi)\big) + \tilde{h}_2\big(\tilde{\boldsymbol{\rho}}(\varphi/2, \psi)\big)\,.\end{aligned}$$

However, to find a plausible homogenous distance functional RHO is problematic. One can consider, for instance, using the famous Hausdorff metric [92] as a candidate for the distance between fibres. Such distance is defined by $\tilde{\boldsymbol{\rho}}_H(\varphi, \psi) = \max\big\{\tilde{\boldsymbol{\rho}}_H^{\max}(\varphi, \psi), \tilde{\boldsymbol{\rho}}_H^{\max}(\psi, \varphi)\big\}$, where $\tilde{\boldsymbol{\rho}}_H^{\max}(\varphi, \psi) = \sup_{\boldsymbol{p} \in \varphi} \rho(\boldsymbol{p}, \psi)$. For two short line segments placed parallel and far from each other, the Hausdorff distance $\tilde{\boldsymbol{\rho}}_H$ does not change much if one of the segments is shortened by half. However, the additivity condition requires the distance decreasing twice.

Our model is based on "*interaction integral*". Instead of looking for distance-dependent pair-potential which would describe the interaction rate between pair of fibres as being indivisible objects, we accumulate interaction potentials between infinitesimal parts of nearby fibres. Interaction integral model clearly satisfies the additivity condition as the integration is a linear operator. It is also clear that the interaction integral model should include self-interaction terms of the form $\tilde{h}_2(\varphi, \varphi)$ (see Figure 3.12).

Fibres are not just sets of points — the points are "continuously" linked and a tangential direction is attributed to every point. To incorporate these intrinsic properties into interaction integral, we represent fibres by small enough line segments and accumulate interaction potentials between these lines segments.

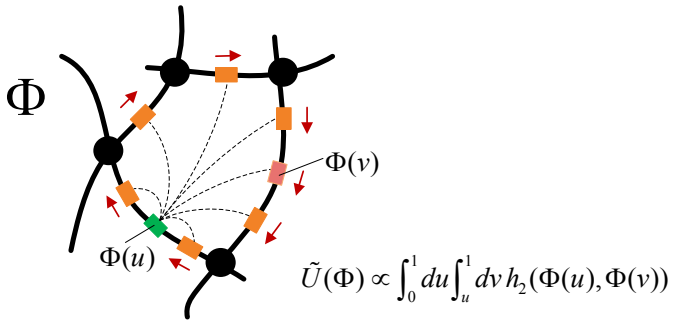


Figure 3.12: Graphical representation of the fibre interaction integral for a fibre system $\Phi : [0,1] \to \mathbb{R}^2$.

3.4.2 Interaction pair-potential between pair of fibres

We start from constructing a pair-potential function $\tilde{h}_2$ based on the idea of interaction integral. Our goal is to reduce the fibre interaction model to something already established and thoroughly investigated, and, in particular, we want to find an expression for $\tilde{h}_2$ which involves original pair-potential models of point processes.

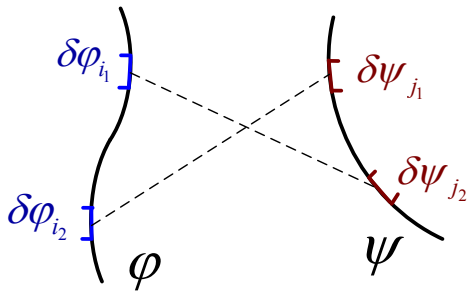


Figure 3.13: Partitioning two fibres φ and ψ into infinitesimal pieces $\{\delta\varphi_i\}_{i=1,2,\ldots,N_\varphi}$ and $\{\delta\psi_j\}_{j=1,2,\ldots,N_\psi}$, which in the limit become line segments.

We partition a given pair of fibres φ and ψ into infinitesimal pieces of the same length, $\varphi = \varphi_1 \cup \varphi_2 \cup \cdots \cup \varphi_{N_\varphi}$ and $\psi = \psi_1 \cup \psi_2 \cup \cdots \cup \psi_{N_\varphi}$, $l_{\varphi_i} \approx l_{\psi_j} \approx \Delta S$, as shown

in Figure 3.13. A fibre pair-potential between φ and ψ is a sum of pair-potentials between each piece of φ and every piece of ψ (see equation 3.27). Every infinitesimal piece of each fibre can be approximated by a short line segments, so that we need to provide a model for a pair-potential between short segments [it will be sought in the form $\bar{h}_{\Delta S}\big(\bar{\boldsymbol{\rho}}_{\Delta S}(\delta\varphi_i,\delta\psi_j)\big)$]. Finally, a line segment pair-potential is derived from pair-potential model of 3D point processes whose first two dimensions bear a "position" of the line segment and the third dimension reflects the tangential direction angle against a specified coordinate axis. The following is the main approximation we apply for pair-potential in our model (for some positive constant $c_2 > 0$)

$$\tilde{h}_2(\varphi,\psi) = c_2 \lim_{\Delta s\to 0} \sum_{i=1}^{N_\varphi}\sum_{j=1}^{N_\psi} \bar{h}_{\Delta S}\big(\bar{\boldsymbol{\rho}}_{\Delta S}(\delta\varphi_i,\delta\psi_j)\big)\,, \tag{3.27}$$
$$N_\varphi = \frac{l_\varphi}{\Delta S},\ \ N_\psi = \frac{l_\psi}{\Delta S}.$$

Proposition 8 (Fibre pair-potential)**.** *A pair-potential between a pair of close to each other fibres φ and ψ can be written as the following interaction integral*

$$\tilde{h}_2(\varphi,\psi) = c_2 \int_0^{l_\varphi} du \int_0^{l_\psi} dv\ \ h\big(\boldsymbol{\rho}\,(\boldsymbol{x}^\varphi(u),\boldsymbol{x}^\psi(v))\big), \tag{3.28}$$

where $h(d,w)$ is a multi-variate analogue of the point pair-potential depending on a vector of distances $\boldsymbol{\rho}(\boldsymbol{x},\boldsymbol{y})$ between two 3D points, and $\boldsymbol{x}^\varphi(u) = \big(\varphi(u);w_\varphi(\varphi(u))\big)$ is a 3D point of the fibre point $\varphi(u)$ and the tangential direction angle (or, orientation) $w_\varphi(u)$ at $\varphi(u)$.

Proof. In the limit, infinitesimal pieces $\delta\varphi_i$ and $\delta\psi_j$ (see Figure 3.13) may be approximated by short line segments $S(p_i,w_\varphi(p_i),l_{\delta\varphi_i})$ and $S(q_j,w_\psi(q_j),l_{\delta\psi_j})$, where p_i and q_i are the middle points of $\delta\varphi_i$ and $\delta\psi_j$, $w_\varphi(p_i)$ denotes a tangential direction to the fibre φ at its point p_i , and $l_{\delta\psi_j}$ is the length of $\delta\psi_j$ (a shorter notation

for the line segments can be used: $S_\varphi[p_i, w_i] \equiv S(p_i, w_\varphi(p_i), l_{\delta\varphi_i})$ to emphasize that the short line segments of the same length are uniquely defined by the pairs of their middle points p_i and orientations $w_i = w_\varphi(p_i)$ of directions tangent to φ at p_i, see Figure 3.14).

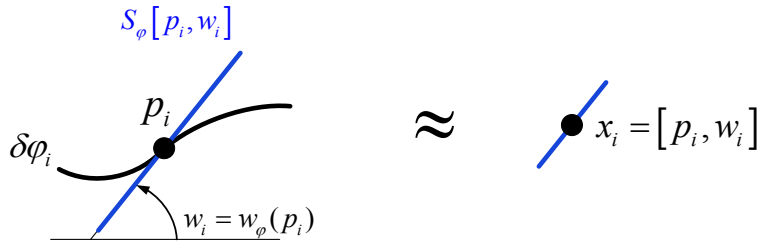


Figure 3.14: Representation of a small fragment of a curvilinear fibre by a 3D point. Left: Replacing the fragment $\delta\varphi_i$ by a short line segment $S_\varphi[p_i, w_i]$. Right. Representation of $S_\varphi[p_i, w_i]$ by just a 3D point $\boldsymbol{x}_i = (p_i; w_i)$.

We assume that an interaction rate between two short line segments is proportional to their lengths

$$\bar{h}_{\Delta S}\big(\bar{\boldsymbol{\rho}}_{\Delta S}(\delta\varphi_i, \delta\psi_j)\big) \propto \alpha\, l_{\delta\varphi_i}\, l_{\delta\psi_j} \approx \alpha\Delta s^2,$$

where a coefficient α is intended to be a function of the segments middle points p_i and q_j and the orientations at these points, $w(p_i)$ and $w(q_j)$,

$$\alpha \equiv \alpha\left(\boldsymbol{x}_i^\varphi = \begin{bmatrix} p_i \\ w(p_i) \end{bmatrix},\ \boldsymbol{x}_j^\psi = \begin{bmatrix} q_j \\ w(q_j) \end{bmatrix}\right).$$

Thus, $\boldsymbol{x}_i^\varphi$ and $\boldsymbol{x}_j^\psi$ are just 3D points, and we can now adapt the pair-potential model of point processes to a model of interacting fibres as follows

$$\alpha \equiv h\big(\boldsymbol{\rho}(\boldsymbol{x}_i^\varphi, \boldsymbol{x}_j^\psi)\big).$$

$\boldsymbol{\rho}(\boldsymbol{x},\boldsymbol{y})$ is a vector of distances between two 3D points $\boldsymbol{x} = (x_1,x_2,x_3)^T$ and $\boldsymbol{y} = (y_1,y_2,y_3)^T$. The first component of $\rho(x,y)$ can be a function of the Euclidean distance between middle points (x_1,x_2) and (y_1,y_2), while its second component can measure an "angular difference" between orientations x_3 and y_3. In this way, h is similar in spirit to the point pair-potential which depends on the inter-point distance. Therefore

$$\bar{h}_{\Delta S}\big(\bar{\boldsymbol{\rho}}_{\Delta S}\left(\delta\varphi_i,\delta\psi_j\right)\big) \approx h\Big(\boldsymbol{\rho}\left(S_\varphi\left[p_i,w_i\right],S_\psi\left[q_j,w_j\right]\right)\Big)\,\Delta s^2,$$

or, more compact,

$$\bar{h}_{\Delta S}\big(\bar{\boldsymbol{\rho}}_{\Delta S}\left(\delta\varphi_i,\delta\psi_j\right)\big) \approx h\big(\boldsymbol{\rho}(\boldsymbol{x}_i^\varphi,\boldsymbol{x}_j^\psi)\big)\Delta s^2.$$

Finally, we can rewrite equation 3.27 in an integral form by making the following substitutions

$$\begin{aligned}
\tilde{h}_2(\varphi,\psi) &= c_2 \lim_{\Delta s\to 0} \sum_{i=1}^{N_\varphi}\sum_{j=1}^{N_\psi} \bar{h}_{\Delta S}\big(\bar{\boldsymbol{\rho}}_{\Delta S}(\delta\varphi_i,\delta\psi_j)\big) \\
&= c_2 \lim_{\Delta s\to 0} \sum_{i=1}^{N_\varphi}\sum_{j=1}^{N_\psi} h\big(\boldsymbol{\rho}(\boldsymbol{x}_i^\varphi,\boldsymbol{x}_j^\psi)\big)\Delta s^2 \\
&= c_2 \int_0^{l_\varphi} du \int_0^{l_\psi} dv\; h\Big(\boldsymbol{\rho}\left(\boldsymbol{x}^\varphi(u),\boldsymbol{x}^\psi(v)\right)\Big),
\end{aligned} \tag{3.29}$$

where $\boldsymbol{x}^\varphi(u) = \big(\varphi(u), w_\varphi(\varphi(u))\big)$.

Remark: in the derivation we assumed that the fibres φ and ψ as curves have natural parameterization (i.e., $l_{\varphi(u_1:u_2)} = |u_2 - u_1|$), so that $\varphi : [0,l_\varphi] \to \mathbb{R}^2$. □

For the fibre pair-potential of the form 3.28, the fibre neighborhood relation $\underset{R}{\sim}$ can be naturally defined by calling two fibres as neighbors if they have at least two

points no farther than some distance R from each other

$$\varphi \underset{R}{\sim} \psi \iff \exists u, v \in [0,1] \;\; \rho\big(\varphi(u), \psi(v)\big) \leq R. \tag{3.30}$$

3.4.3 Complete model for random processes of interacting fibres

Similar to the point Gibbs model given by equation 3.9, we extend the fibre energy functional 3.23 by adding a *"zero-order" interaction term* $\beta(\varphi)$, which assigns some weight to a fibre and, essentially, controls the intensity of the fibres. The energy functional with zero-order term becomes

$$\tilde{U}(\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n) = \sum_i \beta(\varphi_i) + \sum_{\substack{i<j \\ \varphi_i \underset{R}{\sim} \varphi_j}} \tilde{h}_2(\varphi_i, \varphi_j). \tag{3.31}$$

A similar additivity condition on a candidate for zero-order interaction term should hold to keep the total energy conserved under arbitrary fibre system partitioning. Thus, for a partitioned fibre $\varphi = \varphi' \cup \varphi''$ the following should be satisfied

$$\beta(\varphi) = \beta(\varphi') + \beta(\varphi''). \tag{3.32}$$

The following integral form is an obvious model for zero-order interaction term which depends on some kernel function h_0, and which clearly satisfies the additivity condition 3.32

$$\beta(\varphi) = c_0 \int_0^{l_\varphi} du \;\; h_0\left(\boldsymbol{x}^\varphi(u)\right). \tag{3.33}$$

The simplest possible form for zero-order interaction term is the following linear functional proportional to the fibre length

$$\beta(\varphi) = c_0 \, l_\varphi \,,$$

which is a derivative of the model 3.33 with a translation and rotation invariant kernel function h_0.

The second part of the pair-potential additivity condition given by equation 3.25 is very strong and does not hold in general (especially when fibres are curvilinear). The additivity condition given by equation 3.26 is more realistic; therefore, an additional term to the energy functional model 3.31 is needed to address the fibre self-interaction. We propose using an integral form similar to the interaction integral for a self-interaction term. We call it as a *first-order interaction term* $\tilde{h}_1$ and, in a way, it serves as a normalization factor to keep the energy constant under partitioning

$$\tilde{h}_1(\varphi) = c_1 \int_0^{l_\varphi} du \int_u^{l_\varphi} dv \;\; h\big(\boldsymbol{\rho}\left(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\varphi(v)\right)\big)\,. \tag{3.34}$$

The total energy functional becomes

$$\tilde{U}(\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n) = \sum_i \beta(\varphi_i) + \sum_i \tilde{h}_1(\varphi_i) + \sum_{\substack{i<j \\ \varphi_i \underset{R}{\sim} \varphi_j}} \tilde{h}_2(\varphi_i, \varphi_j). \tag{3.35}$$

W.l.o.g. we can consider a fibre system Φ consisting of just two fibres, $\Phi = \varphi \circ \psi$, to prove that the energy functional 3.35 is conserved during partition. The energy functional for system Φ is given by

$$\tilde{U}(\varphi \circ \psi) = \beta(\varphi) + \beta(\psi) \; + \tilde{h}_1(\varphi) + \tilde{h}_1(\psi) \; + \tilde{h}_2(\varphi, \psi)\,. \tag{3.36}$$

If φ is subdivided into two pieces $\varphi = \varphi' \cup \varphi''$, the total energy functional becomes

$$\begin{aligned} \tilde{U}(\varphi' \circ \varphi'' \circ \psi) \;\; &= \;\; \beta(\varphi') + \beta(\varphi'') + \beta(\psi) \; + \tilde{h}_1(\varphi') + \tilde{h}_1(\varphi'') \; + \tilde{h}_1(\psi) \\ &+ \;\; \tilde{h}_2(\varphi', \psi) + \tilde{h}_2(\varphi'', \psi) + \; \tilde{h}_2(\varphi', \varphi'')\,. \end{aligned} \tag{3.37}$$

Term-by-term comparison of equations 3.36 and 3.37 results in the following set of

necessary conditions to satisfy the energy conservation $U(\varphi \circ \psi) = U(\varphi' \circ \varphi'' \circ \psi)$

$$\begin{aligned}
\beta(\varphi) &= \beta(\varphi') + \beta(\varphi''), \\
\tilde{h}_1(\varphi) &= \tilde{h}_1(\varphi') + \tilde{h}_1(\varphi'') + \tilde{h}_2(\varphi', \varphi''), \\
\tilde{h}_2(\varphi, \psi) &= \tilde{h}_2(\varphi', \psi) + \tilde{h}_2(\varphi'', \psi).
\end{aligned}$$

Figure 3.15 graphically shows the second and the third conditions from the above equations.

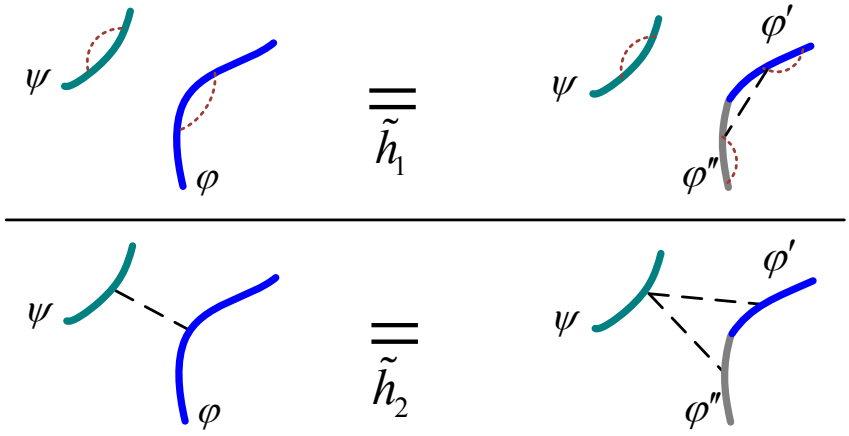


Figure 3.15: Energy conservation for the first-order term $\tilde{h}_1$ (Top) and for the second-order term $\tilde{h}_2$ (Right). Red color dots are the first-order interaction links, and black broken lines are the links of the second-order interaction.

The first and the last conditions are valid because of the way the zero-order interaction and the fibre pair-potential are defined (see equations 3.28 and 3.32). The

following is a simple proof that the second condition is satisfied if $c_1 = c_2$

$$\begin{aligned}
\tilde{h}_1(\varphi)/c_1 &= \int_0^{l_\varphi} du \int_u^{l_\varphi} dv\, h\Big(\boldsymbol{\rho}\big(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\varphi(v)\big)\Big) \\
&= \int_0^{l_{\varphi'}} du \int_u^{l_\varphi} dv\, h(.,.) + \int_{l_{\varphi'}}^{l_\varphi} du \int_u^{l_\varphi} dv\, h(.,.) \\
&= \int_0^{l_{\varphi'}} du \int_u^{l_{\varphi'}} dv\, h(.,.) + \int_0^{l_{\varphi'}} du \int_{l_{\varphi'}}^{l_\varphi} dv\, h(.,.) + \int_{l_{\varphi'}}^{l_\varphi} du \int_u^{l_\varphi} dv\, h(.,.) \\
&= \tilde{h}_1(\varphi')/c_1 + \tilde{h}_2(\varphi', \varphi'')/c_2 + \tilde{h}_1(\varphi'')/c_1 \,.
\end{aligned}$$

Figure 3.16 shows a graphic proof of additivity of the energy functional 3.35.

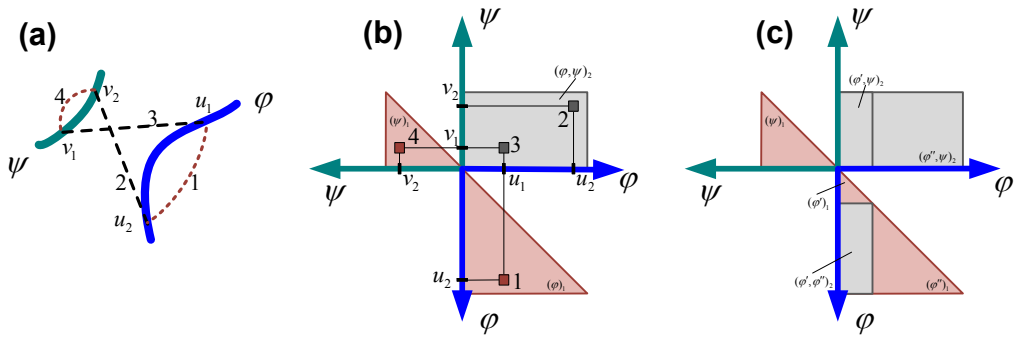


Figure 3.16: Schematic proof of conservation of interaction integral $(\varphi)_1 + (\psi)_1 + (\varphi,\psi)_2$ calculated between two fibres φ and ψ (a:) brefore (b:) and after (c:) φ partition. Here: $(\varphi)_1 \equiv \tilde{h}_1(\varphi)$, $(\varphi,\psi)_2 \equiv \tilde{h}_2(\varphi,\psi)$.

3.5 Random systems of line segments — linear fibres

In this work we focus on a particular type of random fibres — *linear fibres* — which are represented by line segments. However, more general models of fibres can be also constructed by applying the interaction integral model described in the previous sections and by following the same steps which will be described in this section. We

start by describing a representation space for linear fibres and constructing a proper measure space, defining a distribution of Poisson linear fibre process and a way to construct non-trivial linear fibre processes by means of probability density function in the following Section 3.5.1. Though the density function may be of any level of complexity and type, we mainly concentrate on interaction models with pair-potentials based on step functions. They can be used to approximate more complicated interaction models and were proved to be feasible for statistical inference of the point process models [5, 6, 7]. Zero-order potential as well as first-order potential based on a step function are derived in Section 3.5.2. In Section 3.5.3 we show that the linear fibre pair-potential based on a step function can not be resolved in elementary functions, so that an approximation algorithm is also presented.

3.5.1 Interaction model for linear fibres

To define a measure space for linear fibres we first define a representation space for line segments, called *phase space*. Any line segment φ is uniquely represented by its middle point p, its (non-trivial) length l, and its orientation w (the c.c.w. angle made by the line segment against the positive x-axis)

$$[\varphi] = (p_x, p_y, l, w)\,.$$

Therefore the space of linear fibres is in one-to-one correspondence with its representation space $\mathcal{X}$ which is a subset of $\mathbb{R}^4$ of the following form

$$\begin{aligned} [\varphi] \in \mathcal{X} &= \mathcal{X}_P \times \mathcal{X}_L \times \mathcal{X}_W \\ &\equiv X \times [0,\infty) \times [0,\pi)\,. \end{aligned} \tag{3.38}$$

The symbol $\mathcal{X}$ is usually attributed to both the actual space of linear fibres and the phase space 3.38.

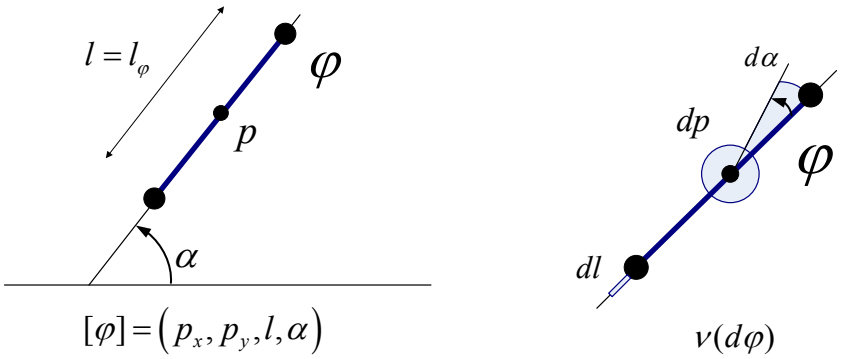


Figure 3.17: Left. A linear fibre in its representation space. Right. Infinitesimal element of the linear fibre measure.

Any Borel set B in the phase space $\mathcal{X}$ defines a set of linear fibres, which is called a *cylindrical subset* of $\mathcal{X}$. The measure on the cylindrical sets is induced by the product of the measures defined on the components of $\mathcal{X}$. An infinitesimal element of such measure in the phase space can be given by (see Figure 3.17, on the right)

$$\nu(d\varphi) = dp \cdot dl \cdot dw .$$

A σ-algebra $\widetilde{\mathcal{B}}$ of the phase space is generated from the cylindrical subsets giving rise to a measure space of linear fibres $(\mathcal{X}, \widetilde{\mathcal{B}}, \nu)$.

The space of linear fibre systems $\mathcal{X}_\infty$ consists of all possible configurations of linear fibre systems $\Phi = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n$ and is defined as an exponential space by the same construction as was described for the point processes in equation 3.1: $\mathcal{X}_\infty = \bigcup_{n=0}^{\infty} \mathcal{X}_n$. In principle, σ-algebra of the exponential space of linear fibre systems, $\widetilde{\mathcal{F}}_\infty = \bigcup_{n=0}^{\infty} \widetilde{\mathcal{F}}_n$, can be constructed in the same ways as is done for the point processes (see description to equation 3.3). However, overlaps between different σ-algebra components, $\widetilde{\mathcal{F}}_{n'}$ and $\widetilde{\mathcal{F}}_{n''}$ for $m' \neq m''$, should be addressed (such overlaps are due to lack of uniqueness in representation of fibres, e.g., a linear fibre can be split in

two parts, and though the original fibre and its split version represent the same fibre, they may belong to exponential space σ-algebra components of different orders, like $F_1 \in \widetilde{\mathcal{F}}_1$ and $F_1 \circ F_2 \in \widetilde{\mathcal{F}}_2$, respectively). Impact of this overlaps on Poisson measure μ for linear fibre processes should be investigated in future work. For now, we define such μ to be similar to that of random point processes

$$\mu(F) = e^{-\nu(\mathfrak{X})} \Bigg[\chi_{F \cap \widetilde{\mathcal{F}}_0}(\emptyset) + \sum_{n=1}^{\infty} \frac{1}{n!} \int \cdots \int \chi_{F \cap \widetilde{\mathcal{F}}_n}(\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n) \ \nu(d\varphi_1)\nu(d\varphi_2) \ \cdots \nu(d\varphi_n) \Bigg] .$$

Probability distributions of new processes are constructed through density function f as in equation 3.5. In this work we investigate new Gibbs type density functions for processes of linear fibres which were defined for general fibre systems through interaction potentials by equations 3.22 and 3.35.

The following notation will be used to represent a linear fibre in what follows: a triple $\left(\boldsymbol{p}, l_\varphi, w_\varphi\right)$ with the fibre middle point $\boldsymbol{p}$, its length l_φ, and its orientation w_φ, or a triple $(\boldsymbol{p}, l_\varphi, \hat{\boldsymbol{\varphi}})$ with the fibre unit direction $\hat{\boldsymbol{\varphi}}$ is associated with any fibre φ.

3.5.2 Zero- and first-order potentials

A general model for zero-order potential $\beta(\varphi)$ was defined by means of some kernel function h_0 in equation 3.33. The orientation of a linear fibre is fixed along its interior; therefore, the kernel function can be represented as a family of functions $h_0(w_\varphi; \cdot)$, indexed by the fibre orientations w_φ, and which depend on a position within a simulation domain

$$h_0\left(\boldsymbol{x}^{\varphi}(u)\right) = h_0\left(w_\varphi; \varphi(u)\right).$$

For a translation invariant kernel function $h_0^{TI}(w_\varphi; p) = h_0(w_\varphi) \equiv h_0(\hat{\boldsymbol{\varphi}})$, zero-order potential becomes

$$\beta^{TI}(\varphi) = c_0\, h_0(\hat{\boldsymbol{\varphi}})\, l_\varphi. \tag{3.39}$$

For a rotation invariant kernel function $h_0^{RI}(w; p) = h_0(p)$, zero-order potential is given by the following integral form

$$\beta^{RI}(\varphi) = c_0 \int_0^{l_\varphi} du\;\; h_0\left(\varphi(u)\right). \tag{3.40}$$

And, finally, for a translation and rotation invariant kernel function, zero-order potential has its simples form depending only on the length of the fibre

$$\beta^{MI}(\varphi) = c_0\, l_\varphi\,.$$

Models for higher order fibre potentials depend on chosen interaction function, which, in general, may be of arbitrary complexity. In this work we consider a piecewise constant interaction function $h_{\boldsymbol{\theta}}$ for a multi-variate point pair-potential. It is defined by the following sum of the products of characteristic functions of the first and the second components of the vector of distances $\boldsymbol{\rho}$

$$\begin{aligned} h_{\boldsymbol{\theta}}(d, w) &= \sum_{\alpha,\beta} \theta_{\alpha\beta}\;\; \chi_{[R_{\alpha-1}, R_\alpha)}(d)\;\; \chi_{[w_{\beta-1}, w_\beta)}(w) \\ &= \sum_{\alpha,\beta} \theta_{\alpha\beta}\;\; I_\alpha(d) J_\beta(w), \end{aligned} \tag{3.41}$$

with parameters $\theta_{\alpha\beta}$ characterizing the interaction rate between two 3D points whose Euclidean "distance" d falls in the half-open segment $[R_{\alpha-1}, R_\alpha)$, and whose orientation "distance" w is within the range $[w_{\beta-1}, w_\beta)$ for a given set of radii $\{R_\alpha, \alpha = 1 \ldots M\}$ and a set of orientations $\{w_\beta,\ \beta = 1 \ldots M_w\}$, and with the following short notation for the characteristic functions

$$\begin{aligned} I_\alpha(d) &\equiv \chi_{[R_{\alpha-1}, R_\alpha)}(d)\,, \quad R_0 = 0\,, \\ J_\beta(w) &\equiv \chi_{[w_{\beta-1}, w_\beta)}(w)\,, \quad w_0 = 0\,. \end{aligned}$$

Proposition 9 (First-order potential for linear fibres)**.** *Linear fibre first-order potential based on the piecewise interaction function 3.41 has the following closed form*

$$\frac{\tilde{h}_1(\varphi)}{c_2} = \begin{cases} l_\varphi \sum\limits_{\alpha=1}^{M} \theta_\alpha \Delta R_\alpha \; - \frac{1}{2} \sum\limits_{\alpha=1}^{M} \theta_\alpha \Delta R_\alpha^2, & l_\varphi \in [R_M, \infty) \\ l_\varphi \sum\limits_{\alpha=1}^{m-1} \theta_\alpha \Delta R_\alpha \; - \frac{1}{2} \sum\limits_{\alpha=1}^{m-1} \theta_\alpha \Delta R_\alpha^2 & \\ \qquad + \frac{1}{2}\theta_m (l_\varphi - R_{m-1})^2, & l_\varphi \in [R_{m-1}, R_m) \end{cases} \tag{3.42}$$

for a fibre φ with the length l_φ, when the following constants are defined:

$$\theta_\alpha \equiv \theta_{\alpha 0}, \quad \Delta R_\alpha = R_\alpha - R_{\alpha-1}, \quad \Delta R_\alpha^2 = R_\alpha^2 - R_{\alpha-1}^2.$$

Proof. For a linear fibre φ with the middle point $\boldsymbol{p}_\varphi$ and orientation $w_\varphi \equiv w(\varphi)$, the function $\boldsymbol{x}^\varphi(u) : [0, l_\varphi] \to \mathbb{R}^3$, defined for equation 3.28, is given by

$$\begin{aligned} \boldsymbol{x}^\varphi(u) &= \left(\boldsymbol{p}_\varphi^T + \left\{ u - \tfrac{l_\varphi}{2} \right\} (\cos w_\varphi, \sin w_\varphi), w_\varphi \right)^T \\ &= \left(\boldsymbol{p}_\varphi^T + \left\{ u - \tfrac{l_\varphi}{2} \right\} \hat{\boldsymbol{\varphi}}', w_\varphi \right)^T. \end{aligned}$$

Therefore, the components of the vector of distances $\boldsymbol{\rho} = (\rho_d, \rho_w)^T$ are given by

$$\begin{aligned} \rho_d(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\varphi(v)) &= \left\| \boldsymbol{p}_\varphi + (u - l_\varphi/2)\hat{\boldsymbol{\varphi}} - \left(\boldsymbol{p}_\varphi + (v - l_\varphi/2)\hat{\boldsymbol{\varphi}} \right) \right\| \\ &= |u - v| \,, \\ \rho_w(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\varphi(v)) &= \left| w_\varphi - w_\varphi \right| = 0. \end{aligned}$$

Substituting the following expression for multi-variate point pair-potential $h(\boldsymbol{\rho}(.,.))$ written for linear fibre φ

$$h_{\boldsymbol{\theta}} \left(\boldsymbol{\rho} \left(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\varphi(v) \right) \right) = h_{\boldsymbol{\theta}}(|u - v|, 0) = \sum_{\alpha=1}^{M} \theta_\alpha I_\alpha \left(|u - v| \right).$$

(with $\theta_\alpha = \theta_{\alpha 0}$) into equation 3.34 results in the following expression for the first-order potential $\tilde{h}_1(\varphi)$

$$\frac{\tilde{h}_1(\varphi)}{c_2} = \int_0^{l_\varphi} du \int_u^{l_\varphi} dv \sum_{\alpha=1}^{M} \theta_\alpha I_\alpha(|u - v|) = \int_0^{l_\varphi} du \int_0^{l_\varphi - u} dv \sum_{\alpha=1}^{M} \theta_\alpha I_\alpha(|v|) \,.$$

We consider two cases depending on the length of fibre φ: $l_\varphi \geq R_M$ and $R_{m-1} \leq l_\varphi < R_m$ for some $m \leq M$.

(*Case* $l_\varphi \geq R_M$.) By finding the level sets of the function $\sum_{\alpha=1}^{M} \theta_\alpha I_\alpha(|v|)$ within the domain $\{(u,v) \in [0,l_\varphi] \times [0,l_\varphi]\}$, the integral above can be reduced to the following analytic expression (where $\overline{R_\alpha} \equiv l_\varphi - R_\alpha$)

$$\begin{aligned}
\frac{\tilde{h}_1(\varphi)}{c_2} &= \theta_1\big(R_1\overline{R_1} + \frac{1}{2}R_1(l_\varphi - \overline{R_1})\big) + \sum_{\alpha=2}^{M} \theta_\alpha\big\{\, \Delta R_\alpha \overline{R_\alpha} + \frac{1}{2}\Delta R_\alpha(\overline{R_{\alpha-1}} - \overline{R_\alpha}) \,\big\} \\
&= \theta_1\big(l_\varphi R_1 - \frac{1}{2}R_1\big) + \sum_{\alpha=2}^{M} \theta_\alpha\big\{\, l_\varphi \Delta R_\alpha - \frac{1}{2}(R_\alpha^2 - R_{\alpha-1}^2) \,\big\} \\
&= l_\varphi \sum_{\alpha=1}^{M} \theta_\alpha \Delta R_\alpha - \frac{1}{2}\sum_{\alpha=1}^{M} \theta_\alpha \Delta R_\alpha^2.
\end{aligned}$$

(*Case* $R_{m-1} \leq l_\varphi < R_m$.) For the second case, we end up with a summation of the same form as above but with indices running up to $m-1$ for such an m that $R_{m-1} \leq l_\varphi < R_m$

$$\begin{aligned}
\tfrac{\tilde{h}_1(\varphi)}{c_2} &= \theta_1\big(R_1\overline{R_1} + \tfrac{1}{2}R_1(l_\varphi - \overline{R_1})\big) + \sum_{\alpha=2}^{m-1} \theta_\alpha\big\{\, \Delta R_\alpha \overline{R_\alpha} + \tfrac{1}{2}\Delta R_\alpha(\overline{R_{\alpha-1}} - \overline{R_\alpha}) \,\big\} \\
&\quad + \theta_m\big(\tfrac{1}{2}(l_\varphi - R_m)\overline{R_{m-1}}\big) \\
&= l_\varphi \sum_{\alpha=1}^{m-1} \theta_\alpha \Delta R_\alpha - \tfrac{1}{2}\sum_{\alpha=1}^{m-1} \theta_\alpha \Delta R_\alpha^2 + \tfrac{1}{2}\theta_m\big(l_\varphi - R_{m-1}\big)^2.
\end{aligned}$$

□

A graph of the function 3.42 is a parabolic curve within each radial bucket $[R_{\alpha-1}, R_\alpha)$ and is a ray within $[R_M, \infty)$.

3.5.3 Interaction pair-potential for linear fibres

Unlike in the case of first-order potential, a derivation of liner fibre pair-potential based on a steplike interactin function does not lead to a closed form expression. It is reflected in the following proposition.

Proposition 10 (Pair-potential for linear fibres)**.** *Linear fibre pair-potential based on the piecewise interaction function 3.41 can not be calculated analytically and requires applying a numerical integration.*

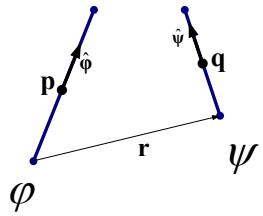


Figure 3.18: Two interacting fibres φ and ψ given by their unit directions $\hat{\boldsymbol{\varphi}}$ and $\hat{\boldsymbol{\psi}}$, respectively.

Proof. For a pair of arbitrary fibres φ and ψ uniquely defined by their middle points $\boldsymbol{p} = \varphi(l_\varphi/2)$ and $\boldsymbol{q} = \psi(l_\psi/2)$, their lengths l_φ and l_ψ, and their unit directions $\hat{\boldsymbol{\varphi}}$ and $\hat{\boldsymbol{\psi}}$, respectively (see Figure 3.18), we need to calculate the following integral

$$\begin{aligned}
\tilde{h}_2(\varphi,\psi)/c_2 &= \int_0^{l_\varphi} du \int_0^{l_\psi} dv\, h_{\boldsymbol{\theta}}\Big(\boldsymbol{\rho}\big(\boldsymbol{x}^{\varphi}(u),\boldsymbol{x}^{\psi}(v)\big)\Big) \\
&= \int_0^{l_\varphi} du \int_0^{l_\psi} dv \sum_{\alpha\beta} \theta_{\alpha\beta}\, I_\alpha\big(\|\varphi(u)-\psi(v)\|\big)\; I_\beta\big(\rho_w(\boldsymbol{x}^{\varphi}(u),\, \boldsymbol{x}^{\psi}(v))\big) \\
&= \int_0^{l_\varphi} du \int_0^{l_\psi} dv \sum_{\alpha} \theta_{\alpha\beta^*}\, I_\alpha\big(\big\|\boldsymbol{p}+(u-l_\varphi/2)\hat{\boldsymbol{\varphi}} - \boldsymbol{q} - (v-l_\psi/2)\hat{\boldsymbol{\psi}}\big\|\big) \\
&= \int_0^{l_\varphi} du \int_0^{l_\psi} dv \sum_{\alpha} \theta_{\alpha\beta^*}\, I_\alpha\big(\|\boldsymbol{r}+u\hat{\boldsymbol{\varphi}} - v\hat{\boldsymbol{\psi}}\|\big)\;,
\end{aligned}$$

where we have used that the difference between the fibres orientation $\rho_w(\boldsymbol{x}^{\varphi}(u),\boldsymbol{x}^{\psi}(v)) \equiv w_{\varphi\psi}$ is constant and $\beta^* \in \{1,\ldots,M_w\}$ is the index of the orientation bucket satisfying

$w_{\varphi\psi} \in [w_{\beta^*-1}, w^*_\beta)$, and $\boldsymbol{r} = \boldsymbol{p} - l_\varphi \hat{\boldsymbol{\varphi}}/2 - \boldsymbol{q} + l_\psi \hat{\boldsymbol{\psi}}/2$. The argument of $I_\alpha(\cdot)$ in the last integral expression can be represented as the square root of a binary quadratic form with a matrix A and a vector $\mathbf{b}$

$$\tilde{h}_2(\varphi,\psi)/c_2 = \int_0^{l_\varphi} du \int_0^{l_\psi} dv \sum_\alpha \theta_{\alpha\beta^*} I_\alpha \left(\left\{ \begin{bmatrix} u \\ v \end{bmatrix}^T A \begin{bmatrix} u \\ v \end{bmatrix} + 2\boldsymbol{b}^T \begin{bmatrix} u \\ v \end{bmatrix} + c \right\}^{1/2} \right),$$

where

$$A = \begin{pmatrix} 1 & -\mu \\ -\mu & 1 \end{pmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{r}\cdot\hat{\boldsymbol{\varphi}} \\ -\boldsymbol{r}\cdot\hat{\boldsymbol{\psi}} \end{bmatrix}, \quad c = \|\boldsymbol{r}\|^2, \quad \mu = \hat{\boldsymbol{\varphi}}\cdot\hat{\boldsymbol{\psi}},$$

and A is non-negative definite, $A \geq 0$, since $\mu \leq 1$. If we exclude the simplest case when A is singular (when fibres φ and ψ are parallel, so that $\mu = \pm 1$), then the quadratic form $< A, \boldsymbol{b}, c >$ can be transformed into its canonical form with a nonsingular diagonal matrix by an affine transformation, $(x,y)^T = T(u,v)^T + \boldsymbol{t}$. Thus, the original interaction integral can be reduced to

$$\tilde{h}_2(\varphi,\psi)/c_2 = \iint_{\Gamma_T} dx\,dy \sum_\alpha \theta_{\alpha\beta^*} I_\alpha \Big(\{\, x^2(1+\mu) + y^2(1-\mu) + c_T \,\}^{1/2} \Big), \quad (3.43)$$

after applying the transformation $< T, \boldsymbol{t} >$ given by

$$T = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad \boldsymbol{t} = \frac{1}{\sqrt{2}} \begin{bmatrix} (\boldsymbol{r}\cdot\hat{\boldsymbol{\varphi}} - \boldsymbol{r}\cdot\hat{\boldsymbol{\psi}})/(1-\mu) \\ (-\boldsymbol{r}\cdot\hat{\boldsymbol{\psi}} - \boldsymbol{r}\cdot\hat{\boldsymbol{\varphi}})/(1+\mu) \end{bmatrix},$$

where

$$c_T = \frac{(\boldsymbol{r}\cdot\hat{\boldsymbol{\varphi}})^2 + (\boldsymbol{r}\cdot\hat{\boldsymbol{\psi}})^2 - 2\mu(\boldsymbol{r}\cdot\hat{\boldsymbol{\varphi}})(\boldsymbol{r}\cdot\hat{\boldsymbol{\psi}})}{1-\mu^2},$$

and Γ_T is the rectangle $[0, l_\varphi] \times [0, l_\psi]$ rotated by 45 degrees c.w. around the origin and translated by a vector $\boldsymbol{t}$. Calculating integral 3.43 is equivalent to a problem of finding the overlapping areas between the rectangle Γ_T and the elliptical rings of the form $R^2_{\alpha-1} \leq x^2(1+\mu) + y^2(1-\mu) + c_T < R^2_\alpha$. There is no a close formula solution for this problem. □

A straightforward way to approximate the fibre pair-potential is to apply one of the standard quadrature rules to the interaction integral given by equation 3.28. The following proposition describes the simplest approximation based on a trapezoidal quadrature rule.

Proposition 11 (Approximation of fibre pair-potential)**.** *By applying a trapezoidal quadrature rule to equation 3.28, the fibre pair-potential may be approximated by the following expression*

$$\tilde{h}_2(\varphi,\psi) = c_2\,\Delta l_\varphi\,\Delta l_\psi \sum_{i=1}^{N_\varphi}\sum_{j=1}^{N_\psi} w_{ij}\, h_{\boldsymbol{\theta}}\left(\boldsymbol{\rho}\left(\boldsymbol{x}^\varphi(u_i),\boldsymbol{x}^\psi(v_j)\right)\right), \tag{3.44}$$

where the weights are taken from the following weight matrix W

$$W = (w_{ij}) = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \cdots & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \cdots & 1 & \frac{1}{2} \\ \vdots & \vdots & & \vdots & \vdots \\ \frac{1}{2} & 1 & \cdots & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \cdots & \frac{1}{2} & \frac{1}{4} \end{pmatrix} \tag{3.45}$$

and

$$u_i = \frac{\Delta l_\varphi}{2} + (i-1)\Delta l_\varphi, \qquad \Delta l_\varphi = \frac{l_\varphi}{N_\varphi},$$
$$v_j = \frac{\Delta l_\psi}{2} + (j-1)\Delta l_\psi, \qquad \Delta l_\psi = \frac{l_\psi}{N_\psi},$$

for given sampling rates N_φ and N_ψ of the fibres φ and ψ, respectively. For a degenerate case — when one of the sampling rates is one — the following weights vector is used instead of matrix W

$$w = (w_j) = \left(\frac{1}{2}, 1, \ldots, 1, \frac{1}{2}\right).$$

Proof. According to the trapezoidal quadrature rule [111, Ch.4] applied for approximating integrals of the form $\int_{u_1}^{u_N} du \int_{v_1}^{v_M} dv\, f(u,v)$, we need to calculate the weights $\int_{u_{i-1}}^{u_{i+1}} du \int_{v_{j-1}}^{v_{j+1}} dv\, \phi_{ij}(u,v)$ for the function $f(u,v)$ being interpolated by $f(u,v) = \sum_{ij} f_{ij}\phi_{ij}(u,v)$, with $\phi_{ij}(u,v) = \phi_i^u(u)\phi_j^v(v)$, $f(u_i,v_j) = f_{ij}$, and

$$\phi_i^x(x) = \frac{x - x_{i-1}}{x_i - x_{i-1}} \chi_{[x_{i-1},x_i)}(x) \; + \; \frac{x - x_{i+1}}{x_i - x_{i+1}} \chi_{[x_i,x_{i+1})}(x)\,.$$

Thus, when considering a regular grid for both dimensions $\big\{(u_i + (i-1)\Delta u, v_1 + (j-1)\Delta v)\big\}$, for a corner weight w_{11} we calculate the following integral

$$w_{11} = \iint \phi_{ij} = \int_{u_1}^{u_2} du \frac{u - u_2}{-\Delta u} \int_{v_1}^{v_2} dv \frac{v - v_2}{-\Delta u} \; = \frac{1}{2}\frac{1}{2}\Delta u\,\Delta v \; = \frac{1}{4}\Delta u\,\Delta v.$$

The same applies to the other corner weights w_{1M}, w_{N1}, and w_{NM}. For the border samples (u_1, v_j), $1 < j < M$, the corresponding weights are

$$\begin{aligned} w_{1j} &= \int_{u_1}^{u_2} du \frac{u - u_2}{-\Delta u} \left(\int_{v_{j-1}}^{v_j} dv \frac{v - v_{j-1}}{\Delta v} + \int_{v_j}^{v_{j+1}} dv \frac{v - v_{j+1}}{-\Delta v} \right) \\ &= \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2}\right)\Delta u\,\Delta v \; = \frac{1}{2}\Delta u\,\Delta v. \end{aligned}$$

Weights w_{Nj}, w_{i1}, and w_{iM} are calculated in the same way, for $1 < i < M$. And finally, the weights corresponding to the internal samples become

$$\begin{aligned} w_{ij} &= \left(\int_{u_{i-1}}^{u_j} du \frac{u - u_{i-1}}{\Delta u} + \int_{u_i}^{u_{i+1}} du \frac{u - u_{i+1}}{-\Delta u} \right) \left(\int_{v_{j-1}}^{v_j} dv \frac{v - v_{j-1}}{\Delta v} + \int_{v_j}^{v_{j+1}} dv \frac{v - v_{j+1}}{-\Delta v} \right) \\ &= \left(\frac{1}{2} + \frac{1}{2}\right)\left(\frac{1}{2} + \frac{1}{2}\right)\Delta u\,\Delta v \; = \Delta u\,\Delta v. \end{aligned}$$

□

N_φ and N_ψ are usually chosen such that $\Delta l_\varphi \approx \Delta l_\psi$ (e.g., by choosing a common, small enough sampling length Δl and by assigning $N_\varphi = \lfloor l_\varphi/\Delta l \rfloor$ and $N_\psi = \lfloor l_\psi/\Delta l \rfloor$).

A linear fibre pair-potential based on a piecewise constant interaction function is derived by a straightforward substitution of point pair-potential $h(\cdot)$in equation 3.44 by expression 3.41, and is described by the following Proposition.

Proposition 12 (Approximation of linear fibre pair-potential)**.** *A linear fibre pair-potential with interaction model 3.41 can be approximated by the following expression*

$$\tilde{h}_2(\varphi,\psi) = c_2\,\Delta l_\varphi\,\Delta l_\psi \sum_{i=1}^{N_\varphi}\sum_{j=1}^{N_\psi} w_{ij} \sum_{\alpha}\theta_{\alpha\beta^*}\, I_\alpha\big(\big\|\varphi(u_i)-\psi(v_j)\big\|\big)\,, \tag{3.46}$$

where $\beta^* \in \{1,\ldots,M_w\}$ *is such that the orientation difference between* $\hat{\boldsymbol{\varphi}}$ *and* $\hat{\boldsymbol{\psi}}$ *satisfies*

$$\rho_w(\boldsymbol{x}^\varphi(u),\boldsymbol{x}^\psi(v)) \in [w_{\beta^*-1}, w^*_\beta).$$

3.6 Parametric synthesis of linear fibres systems

Random systems of linear fibres will be simulated by applying (discrete-time) Metropolis-Hastings algorithm. The algorithm of this type clearly belongs to a group of parametric synthesis methods as the distribution of resulting arrangements of fibre systems can be controlled by the parameters of linear fibre model (e.g., one can alternate the density of fibres, the fibre orientation distribution, orientation correlations between neighbor fibres etc). We describe Metropolis-Hastings simulation algorithm applied to generate samples from the space of linear fibre systems in Section 3.6.1. In Section 3.6.2 all necessary data structures and miscellaneous algorithms to optimize simulation of linear fibre systems are developed. In Section 3.6.3 the results of simulations based on several simple piecewise constant interaction models are presented and it is noticed that more advanced simulation algorithms are required to

solve the problem of a resulted inadequate fibre density and to include constrained simulation.

3.6.1 Detailed balance equation for linear fibres

We generate samples of fibre systems by running a discrete-time Markov chain $\{\mathfrak{X}_k\}$ with Metropolis-Hastings dynamics described in Section 3.3.4. Every state of the Markov chain is a fibre system, $\mathfrak{X}_k = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n$, of a finite number of fibres. We only consider "birth" and "death" events applied to a current state of the Markov chain, following recommendations in [52], that stated that avoiding the recombination transitions ($p_{bd} = 1$ in equation 3.17) would drastically improve the convergence rate of a Markov chain of point configurations $\{\mathsf{X}_k\}$. Table 3.3 summarizes key objects of the *Metropolis-Hastings Birth-and-Death* (MH-BnD) dynamics.

Notation	"birth"	"death"
Current state $\mathfrak{X}_k$	Φ	Φ
Selection probability	$q(\Phi)$	$1-q(\Phi)$
Proposal update density w.r.t. $\nu(d\psi)$	generate ψ $b(\psi,\Phi)$	delete an item φ_i $d(\varphi_i,\Phi_{\hat{i}})$
Acceptance probability	$A(\Phi\circ\psi\|\Phi) = \min\{1, r(\psi,\Phi)\}$	$A(\Phi_{\hat{i}}\|\Phi) =$ $\min\{1, 1/r(\varphi_i,\Phi_{\hat{i}})\}$
Next state, Φ'	$\Phi\circ\psi$	$\Phi_{\hat{i}} \equiv \Phi \setminus \varphi_i$

Table 3.3: Notation for one step of Metropolis-Hastings "Birth-and-Death" algorithm (MH-BnD).

Simulation starts from an empty fibre system and proceeds by performing a series of uniform steps (transitions) until an acceptable arrangement of fibres is reached (or

by using some other empirical stopping criterion). For $\mathfrak{X}_k = \emptyset$, only "birth" transitions are allowed. A high-level description of one step of the MH-BnD is given in Algorithm 2.

Algorithm 2 One iteration of Metropolis-Hastings BnD algorithm for generating fibre systems.

INPUT:

- a current state $\mathfrak{X}_k = \Phi = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n$,
- a description of $\mathcal{X}$ and functions $q(\Phi)$, $b(\psi, \Phi)$, $d(\varphi_i, \Phi_{\hat{i}})$, $r(\psi, \Phi)$

OUTPUT:

the next state $\mathfrak{X}_{k+1} = \Phi'$, which could be $\mathfrak{X}_{k+1} = \Phi$ if a proposal for a new state is rejected.

---

1. choose a "birth" event with probability $q(\Phi)$ or a "death" event with probability $1 - q(\Phi)$.

2. "*birth*". pick a new fibre ψ distributed according to density $b(\psi, \Phi)$, and accept it with probability $\min\{1, r(\psi, \Phi)\}$, so that $\mathfrak{X}_{k+1} = \Phi \circ \psi$; otherwise, $\mathfrak{X}_{k+1} = \Phi$.

3. "*death*". pick an existing fibre φ_i from the current fibre system Φ according to probability $d(\varphi_i, \Phi_{\hat{i}})$, and accept deleting the fibre with probability $\min\{1, 1/r(\phi_i, \Phi_{\hat{i}})\}$, $\mathfrak{X}_{k+1} = \Phi_{\hat{i}}$; otherwise, $\mathfrak{X}_{k+1} = \Phi$.

A transition likelihood function $r(\psi, \Phi)$ is chosen so as to satisfy the following

detailed balance equation

$$[1-q(\Phi\circ\psi)]\,d(\psi,\Phi)\,A(\Phi|\Phi\circ\psi)\,f(\Phi\circ\psi)$$
$$=q(\Phi)\,b(\psi,\Phi)\,A(\Phi\circ\psi|\Phi)\,f(\Phi)\,. \tag{3.47}$$

For the "birth" event the likelihood function $r(\psi,\Phi)=r_b(\psi,\Phi)$ is given by

$$\begin{aligned} r_b(\psi,\Phi) &= \frac{f(\Phi\circ\psi)}{f(\Phi)}\;\frac{1-q(\Phi\circ\psi)}{q(\Phi)}\;\frac{d(\psi,\Phi)}{b(\psi,\Phi)} \\ &= \lambda_c(\psi,\Phi)\;\frac{1-q(\Phi\circ\psi)}{q(\Phi)}\;\frac{d(\psi,\Phi)}{b(\psi,\Phi)}. \end{aligned} \tag{3.48}$$

For the "death" event, $r(\varphi_i,\Phi_{\hat{i}})=r_d(\varphi_i,\Phi_{\hat{i}})$ is given by

$$\begin{aligned} r_d(\varphi_i,\Phi_{\hat{i}}) &= \frac{f(\Phi_{\hat{i}})}{f(\Phi)}\;\frac{q(\Phi_{\hat{i}})}{1-q(\Phi)}\;\frac{b(\varphi_i,\Phi_{\hat{i}})}{d(\varphi_i,\Phi_{\hat{i}})} \\ &= \frac{1}{\lambda_c(\varphi_i,\Phi_{\hat{i}})}\;\frac{1}{\frac{1-q(\Phi)}{q(\Phi_{\hat{i}})}}\;\frac{1}{\frac{d(\varphi_i,\Phi_{\hat{i}})}{b(\varphi_i,\Phi_{\hat{i}})}} = \frac{1}{r_b(\varphi_i,\Phi_{\hat{i}})}. \end{aligned} \tag{3.49}$$

For Gibbs fibre distributions $f(\Phi)=\exp\{\tilde{U}(\Phi)\}/\mathcal{Z}$ with the total energy model 3.35, the conditional intensity λ_c is given by

$$\lambda_c(\psi;\Phi) = \frac{f(\Phi\circ\psi)}{f(\Phi)} = \exp\left\{\beta(\psi)+\tilde{h}_1(\psi)+\sum_{\psi\underset{R}{\sim}\varphi_j}\tilde{h}_2(\psi,\varphi_j)\right\}, \tag{3.50}$$

if $\psi\notin\Phi=\varphi_1\circ\varphi_2\circ\ldots\circ\varphi_n$, and

$$\lambda_c(\varphi_i;\Phi_{\hat{i}}) = \frac{f(\Phi)}{f(\Phi_{\hat{i}})} = \exp\left\{\beta(\varphi_i)+\tilde{h}_1(\varphi_i)+\sum_{\substack{i\neq j\\ \varphi_i\underset{R}{\sim}\varphi_j}}\tilde{h}_2(\varphi_i,\varphi_j)\right\}, \tag{3.51}$$

otherwise. Density function $b(\psi,\Phi)$ and the probabilities $d(\varphi_i,\Phi_{\hat{i}})$ should be chosen to be as simple as possible to make transition steps easy to perform. We propose using the following *constant-birth* Metropolis-Hastings model (MH-BnD-CB) for simulation a system of linear fibres: it has a constant selection rate q_b, creates a new fibre uniformly distributed along $\mathcal{X}$ during the "birth" step, picks an arbitrary fibre for

deletion from the current fibre system at the "death" step, and accpets the resulting proposals according to the transition likelihood $r(\psi,\Phi)$. The Table entitled by "MH-BnD-CB" summarizes an iteration step of MH-BnD-CB dynamics applied for linear fibres.

MH-BnD "Constant birth"	(*MH-BnD-CB*)
$q(\Phi) \equiv q_b$	Probability of "birth" event
$b(\psi,\Phi) = \frac{1}{\nu(\mathcal{X})}$	Density for a new fibre candidate
$d(\varphi,\Phi_{\hat{i}}) = \frac{1}{n+1}$	Probability to delete φ_i from Φ
$r(\psi,\Phi) = \lambda_c(\psi;\Phi)\frac{1-q_b}{q_b}\frac{\nu(\mathcal{X})}{n+1}$	"birth" transition likelihood
$r(\varphi_i,\Phi_{\hat{i}}) = \lambda_c(\varphi_i;\Phi_{\hat{i}})\frac{1-q_b}{q_b}\frac{\nu(\mathcal{X})}{n}$	"death" transition likelihood

Drawing a sample, uniformly distributed on $\mathcal{X}$ is straightforward: first, a random location for a new fibre is uniformly sampled from a given simulation domain $\mathcal{X}_P$ (see equation 3.38); second, its length is uniformly sampled from a given length domain $\mathcal{X}_L$; and, third, its orientation is uniformly sampled from $\mathcal{X}_\mathcal{A}$. $\mathcal{X}_P$, $\mathcal{X}_L$, and $\mathcal{X}_\mathcal{A}$ are arbitrary compact subsets of $\mathbb{R}^2$, $[0,\infty)$, and $[0,\pi)$, respectively, with the only restriction for $\mathcal{X}_L$ to be a set of the form $[0,l_{\max}]$; and $\nu(\mathcal{X}) = |\mathcal{X}_P|\cdot|\mathcal{X}_L|\cdot|\mathcal{X}_\mathcal{A}|$. An accepted proposal is added to the current fibre system and, possibly, is subdivided along its interior at the locations where it intersects the located nearby fibres, so that the resulting set of fibres obeys the fibre system definition.

Neighborhood relation between the linear fibres $\underset{R}{\sim}$ is defined by specifying interaction area around linear fibre. We call it (interaction) *fibre vicinity* (or, interaction fibre neighborhood) $B(\varphi,R)$ of radius R which includes all the points near the fibre

located no farther than distance R

$$B(\varphi, R) = \left\{ x \in \mathbb{R}^2,\ \min \|x - \varphi\| \leq R \right\}.$$

Then, we call two fibres φ and ψ being neighbors if each fibre is in vicinity of the other fibre

$$\varphi \underset{R}{\sim} \psi \Longleftrightarrow B(\varphi, R) \cap \psi \neq \emptyset. \tag{3.52}$$

For example, for the linear fibre model with piecewise constant interaction function $h_{\boldsymbol{\theta}}$ given by equation 3.41, the neighborhood relation will be specified by the maximum interaction radius R_M, i.e., $\varphi \underset{R_M}{\sim} \psi$ for two fibres φ and ψ.

The algorithm intends to run until the equilibrium is reached. However, there is no a deterministic simulation termination criterion available for MH-BnD-CB type algorithm to the best of our knowledge. We suggest using empirical stopping criteria based on the average statistics of birth/death event rates measured along a substantial number of recent iterations. For example, one can stop simulation when actual birth and death rates are within 20-30% relative difference, or/and when the same proportion applies for the recent rejection rates. A successful adaptation of the exact simulation algorithm developed for point processes — an alternative Metropolis-Hastings type algorithm which provides a deterministic way for exit [64, 71, 70] — for the case of fibre systems is believed to be not feasible at this point.

3.6.2 Synthesis algorithm

The synthesis algorithm for generating arrangements of fibres in the plane requires specifying the parameters of the fibre interaction model (given by the density function f) and the parameters related to Metropolis-Hastings (MH) iterations, and choosing some stopping criterion, as was discussed in the previous Section. While most of the

steps of MH algorithm are straightforward to implement, optimality aspects of the iterations should be discussed father.

Two potentially expensive operations that are performed at every step of MH algorithm are

MH-O$_1$ calculating the total pair-potential (as a part of the conditional intensity λ_c given by equations 3.50 and 3.51) for a given fibre against all the fibres located in its proximity;

MH-O$_2$ adding a new fibre to the current fibre system during "birth" step.

Both operations make geometric search queries on localizing the neighbors for a given fibre, so that the performance of these operations depends on how optimal the neighbor search query is implemented.

For the first operation, we calculate pair-potentials only between a given fibre ψ and any, generated previously, fibre φ which is no farther from ψ than some *interaction radius* R, i.e., $\psi \underset{R}{\sim} \varphi$ and $\varphi \in \Phi$. The pair-potentials with other fibres are zero and should be disregarded in calculations. The result of the neighbors search query applied for the fibres from Φ will be denoted by $Nb(\psi;\Phi,R)$

$$Nb(\psi;\Phi,R) = \left\{\varphi \in \Phi \mid \psi \underset{R}{\sim} \phi\right\}. \tag{3.53}$$

The second operation requires finding all fibres in the current configuration which intersect a new fibre. In our language, this fibres are denoted by $Nb(\psi;\Phi,0)$. All involved fibres, $\psi \circ Nb(\psi;\Phi,R)$, should be dissected to form a new fibre subsystem with the intersection points becoming corresponding end-points of the dissected fibres. Finally, this subsystem replaces fibres from $Nb(\psi;\Phi,0)$ within Φ to form a new fibre system Φ' (an example is shown in Figure 3.19). Therefore, the complexity of

the second operation is determined by the complexity of the neighbor search query $Nb(\psi;\Phi,0)$. *Comment:* To make the operation more robust, distance 0 is usually replaced by a small distance $\Delta\rho$. Thus, the candidates for intersection test with a given fibre *psi* are taken from $Nb(\psi;\Phi,\Delta\rho)$.

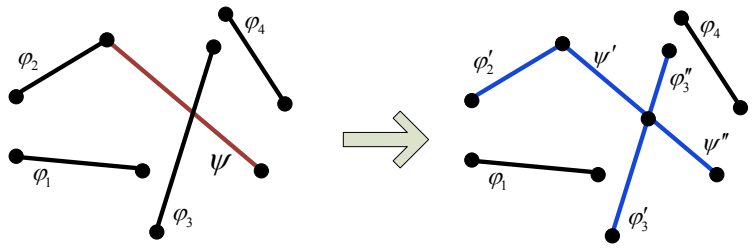


Figure 3.19: Dissection of fibres during adding a new fibre ψ to a fibre system $\Phi = \varphi_1 \circ \cdots \circ \varphi_4$. Φ will be replaced by $\Phi' = \Phi \setminus Nb(\psi;\Phi,0) \circ \Phi_\psi$, where $\Phi_\psi = \psi' \circ \psi'' \circ \varphi_2' \circ \varphi_3' \circ \varphi_3''$ and $Nb(\psi;\Phi,0)$ in this case is $\{\varphi_2,\varphi_3\}$.

The algorithm of finding the neighbors $Nb(\psi;\Phi,R)$ can be implemented by testing each of n fibres of the current configuration Φ against the neighborhood relation $\underset{R}{\sim}$ with a given fibre ψ. This brute-force strategy clearly takes $O(n)$ time, $n = N(\Phi)$. In reality, however, a fibre have no or only a few neighbors, so that the number of tests necessary to perform on the neighborhood relation could be much smaller than $O(n)$. To avoid testing the fibres that are far apart we develop an additional data structure with the size $O(n)$ which is essential in maintaining a fast lookup function that searches for neighbors only in a vicinity of a given fibre.

The simulation domain X can be enclosed in a finite size rectangle domain C, $X \subseteq C$, and the rectangle domain is partitioned by a set of cells $\{c_i\}$, $C = \cup_i c_i$. Every cell c_i has a list of references to all fibres which pass the cell:

$$F(c_i;\Phi) = \left\{ \varphi \in \Phi \,|\, c_i \cap \varphi \neq \emptyset \right\}.$$

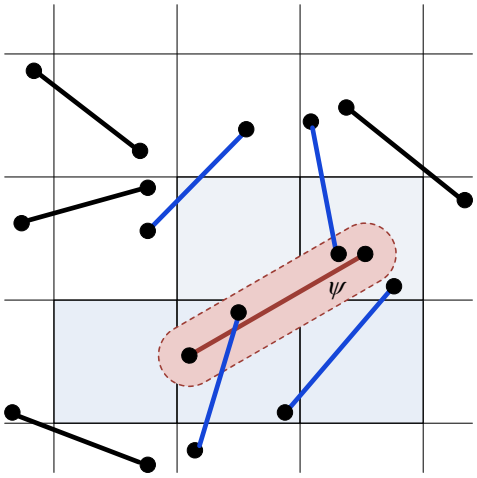


Figure 3.20: Neighborhood search query $Nb(\psi;\Phi,R)$ optimized by using a domain partition. Only fibres (blue line segments) passing the cells (filled light blue rectangles) in the vicinity of fibre ψ, denoted by $B(\psi,R)$ (rose bounded area), are tested against the neighborhood relation.

Different partitioning strategies can be applied to the domain C [31], and a choice depends on many factors including but not restricted to the estimated fibre density and/or an average fibre length of the target fibre interaction model. An easy-to-implement choice is to use a regular partitioning where all the cells are of the same rectangular shape and the same dimensions: $c_{ij} = c + i\Delta x\boldsymbol{e}_x + j\Delta y\boldsymbol{e}_x$, $C = \cup_{ij} c_{ij}$. Then the lookup function returns all the cells located in the vicinity of the fibre, so that the candidates for neighbors are picked from the found cells. This guarantees that only close fibres will be tested and the number of tests is proportional to the average fibre density multiplied by the area of an average fibre's vicinity, which is approximately equal to $\frac{\hat{N}(\mathfrak{X})}{|X|} \cdot \left|\hat{l}_\varphi + R\right|^2$. The steps are summarized in Algorithm 3 and illustrated in Figure 3.20.

Given in its original form, the synthesis algorithm 3 may produce fibre systems

Algorithm 3 Fibre neighborhood search $Nb(\psi;\Phi,R)$.

PREREQUISITES: a simulation domain partition c_i, every c_i has links $F(c_i,\Phi)$;

a global test index i_T.

INPUT:

a fibre ψ, a current fibre system Φ, and an interaction radius R.

OUTPUT:

$Nb(\psi;\Phi,R)$.

1. find cells $c(\psi)$ which are in a vicinity of ψ: $c(\psi) = \left\{ c_i \in C \,|\, c_i \cap B(\psi,R) \neq \emptyset \right\}$.
2. if $\cup_{c\in c(\psi)} F(c,\Phi) \neq \emptyset$ increase the global index i_T, otherwise, return the empty set.
3. for every $c \in c(\psi)$ and for each $\varphi \in F(c,\Phi)$

   skip neighborhood test if φ has been already labeled by i_T value;

   otherwise, label φ by i_T value and perform the neighborhood test; if $\psi \underset{R}{\sim} \varphi$ then include φ to $Nb(\psi;\Phi,R)$.

with a variety of unwanted features: narrow gaps between close aligned fibres, some fibres turn out to be extremely short, connectivity between certain fibres are unrealized (these cases are shown in Figure 3.21).

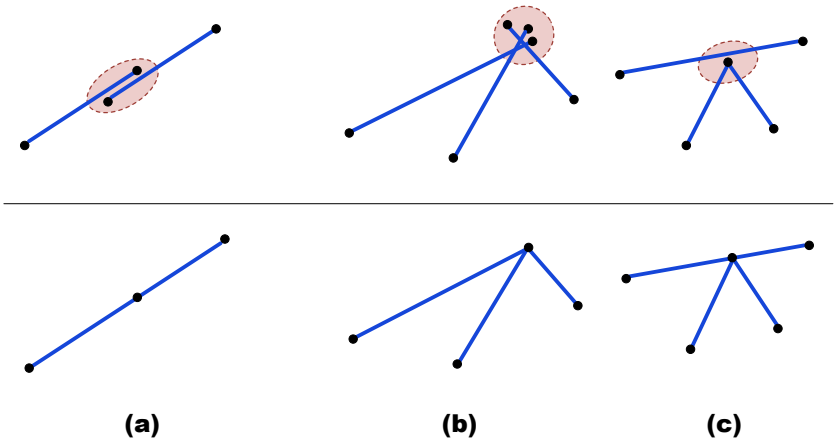


Figure 3.21: Fibre configurations with undesirable features (top row) and possible solutions (bottom row).

We developed a relaxation scheme which modifies the locations of those pairs of fibres which are in close proximity to each other. Some proximity radius R_Δ controls how close individual fibres can approach each other. By applying the relaxation scheme we want to ensure that (1) two close fibres φ and ψ with $\varphi \underset{\Delta R}{\sim} \psi$ should be connected and (2) there are no extremely short fibres which length values are smaller than R_Δ.

The relaxation scheme needs to be applied at the "birth" step of MH algorithm when a new fibre ψ of a length value greater than the proximity radius R_Δ is added to the current fibre configuration. First, we relax the end-point locations $\psi(0)$ and $\psi(1)$ if either of them is located in the proximity to the interia of the existing fibres. The end-point are projected to the corresponding interia giving rise to new end-points

$\psi'(0)$ and $\psi'(1)$, so that we replace ψ by ψ' [see Figure 3.22 (a)].

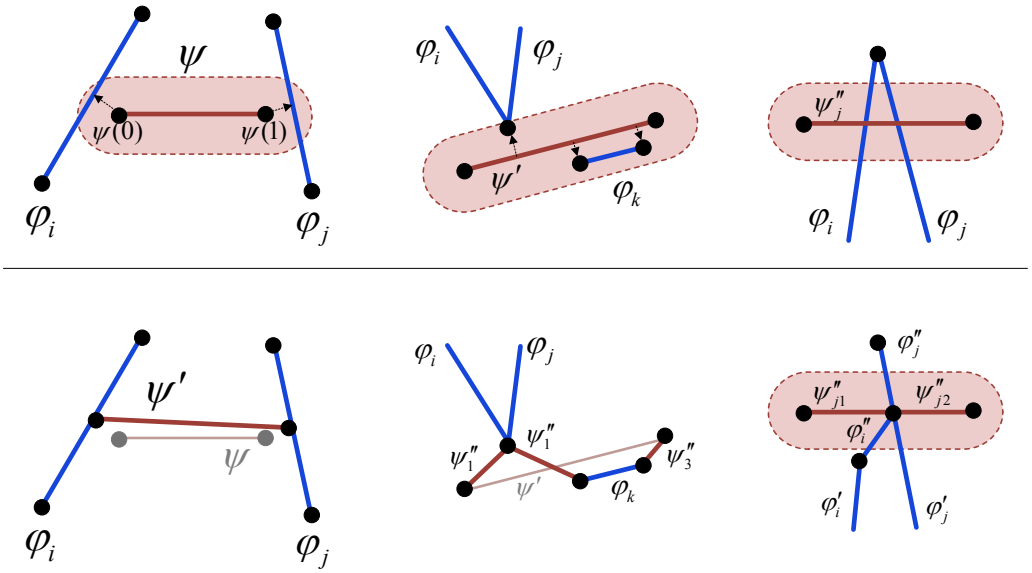


Figure 3.22: Relaxation scheme within a proximity (filled area) of a fibre ψ. (a) Translation of end-points $\psi(0)$ and $\psi(1)$ towards interia of close fibres φ_i and φ_j. (b) Pulling the interior of ψ' towards end-points of fibres φ_i, φ_j, and φ_k located within the proximity of ψ'. (c) Avoiding forming a segments shorter than the proximity radius R_Δ.

Second, we relax the interior of fibre ψ' by moving it towards the end-points of the fibres being in proximity to the interior of ψ'. Such relaxation is performed by dissecting ψ' into k sub-fibres $\psi'_1 \circ \psi'_2 \circ \ldots \circ \psi'_k$ and dislocating fibres $\{\psi'_j\}$ towards the end-points as shown in Figure 3.22 (b). Resulting set of fibres $\psi''_1 \circ \psi''_2 \circ \ldots \circ \psi''_{\tilde{k}}$ closely approximates a given fibre ψ and is passed to the original procedure of adding fibres **MH-O$_2$** described earlier in this Section. Third, while applying **MH-O$_2$** algorithm to fibres $\{\psi''_j\}$, one should avoid creating extremely short segments resulted by intersection of a fibre ψ''_j with two close enough fibres φ' and φ'' from Φ as shown

in Figure 3.22 (c). It is easy to prove that such situation is possible only when φ' and φ'' have a common end-point. As such, we can shorten one of the fibres, enough to satisfy the relaxation criteria. Such operation is safe and it does not change much the total energy functional of the original fibre system.

3.6.3 Examples of generated linear fibres systems

We start with the simplest possible linear fibre model which has a trivial probability distribution function (pdf) consisting of only a zero-order potential

$$f(\Phi) = \exp\Big\{ \sum_i \beta(\varphi_i) \Big\} / \mathcal{Z}, \tag{3.54}$$

where zero-order potential $\beta(\varphi)$ is given in its general form by equation 3.33. Such model is an analogue of the Poisson process as interaction between the fibres is not included to pdf. Therefore, we call a process, corresponding to this model, as *Poisson linear fibre process*. We use translation and rotation invariant kernel functions h_0 for zero-order potentials, so that

$$\beta(\varphi) = \beta_0 l_\varphi \,. \tag{3.55}$$

A parameter β_0 defines a concentration of the linear fibres.

Example 12.1 Poisson linear fibre system

We collected several examples of Poisson type fibre processes in Figure 3.23. In the left column, a pair of images with fibre arrangements generated by the standard line segment process with different fibre concentration levels are illustrated. Such process is formed by populating a given simulation domain with overlapping line segments of a given length range. Intersection tests are not performed during simulation of the process. The images on the right show samples of our Poisson linear fibre process

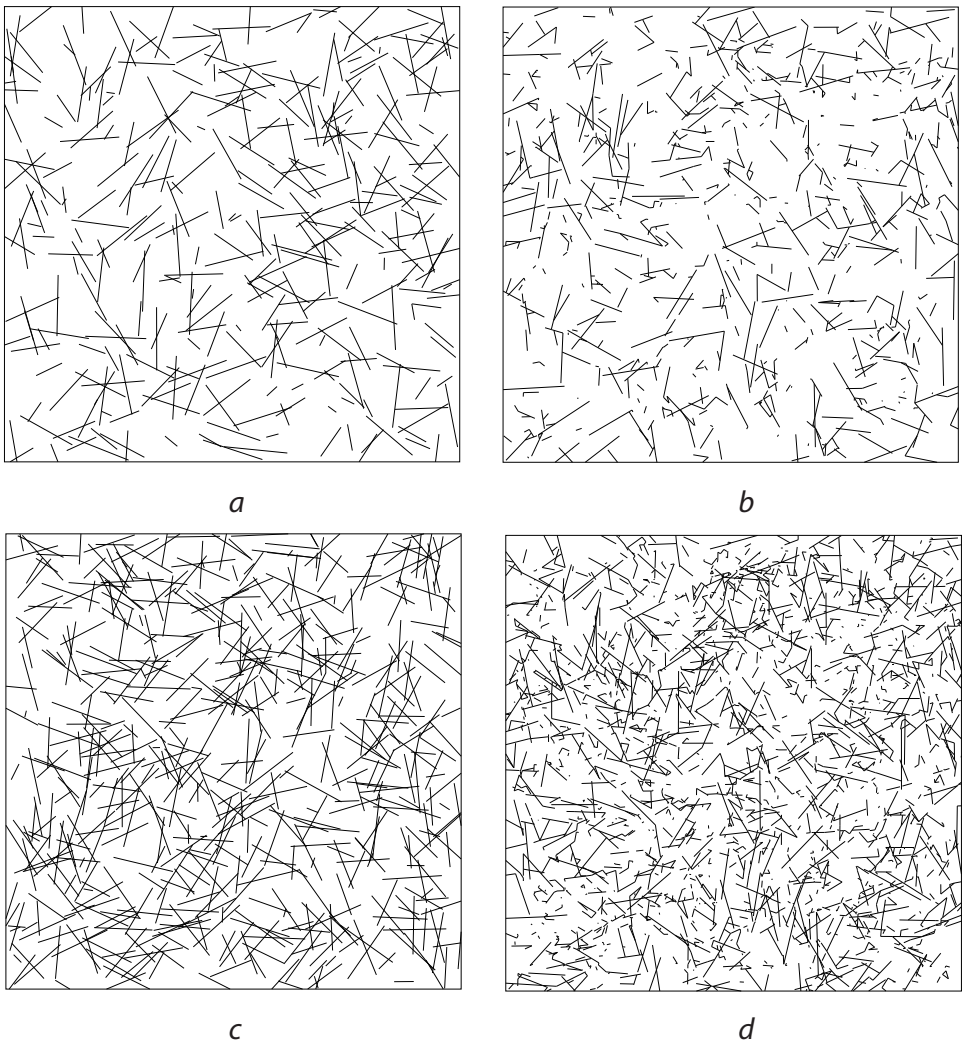


Figure 3.23: Poisson type fibre processes with different fibre concentration levels (increasing from top to bottom). Line segment process (on the left) versus our Poisson homogenous linear fibre process (on the right) with pdf given by equation 3.54. Total fibre length for the images in the first row is $l_{\Phi_{(a,b)}} = 218$ and for the images in the second row is $l_{\Phi_{(c,d)}} = 383$.

with concentration values chosen as $\beta_0 = 1.5$ and $\beta_0 = 2.5$ (from top to bottom). The phase space for both processes has the following components: $\mathcal{X}_P = [0,7]^2$, $\mathcal{X}_L = [0,1]$, and $\mathcal{X}_W = [0,2\pi)$. The proximity radius for the simulation algorithm is set to $R_\Delta = 0.01$.

The main difference between the samples of these processes is that fibre systems of Poisson linear fibre process have larger population of short segments then the line segment processes do. Indeed, the systems in each row have the same total fibre length: $l_\Phi = 218$ for the first row and $l_\Phi = 383$ for the second. However, the number of the fibres in the systems illustrated on the left column are approximately $N(\Phi_{(a)}) = 300$ and $N(\Phi_{(c)}) = 300$, while for the right column: $N(\Phi_{(b)}) = 1,300$ and $N(\Phi_{(d)}) = 3,900$.

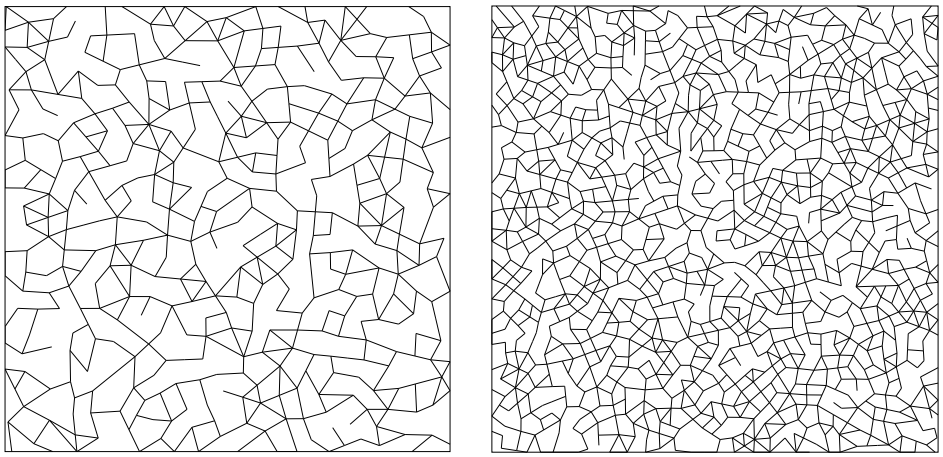


Figure 3.24: Poisson linear fibre processes with large proximity radii R_Δ.

A simple way to introduce regularity to Poisson linear fibre systems is to substantially increase the proximity radius R_Δ (refer to the previous section for its definition). Applying relaxation schemes defined in Figure 3.22 during simulation of

Poisson linear fibre process removes short fibres and merge/link relatively far fibres. For relatively large values of R_Δ such simulation produces fibre systems with profound level of regularity, though such systems are not distributed as Poisson linear fibre processes anymore and can be thought of as a projection of a Poisson process onto such a phase space where disconnected fibres can not be closer to each other than the proximity radius. Figure 3.24 illustrates two examples of fibre systems generated from Poisson processes with proximity radii $R_\Delta = 0.25$ and $R_\Delta = 0.15$ used for sub-figure (a) and sub-figure (b), respectively [which is one order of magnitude larger than the corresponding original value of the proximity radius used in generating fibre system shown in Figures 3.23].

Other variations of the Poisson linear fibre process are possible when zero-order potential model $\beta(\varphi)$ becomes depending on the location and/or orientation of the fibres. Such variations are another way to bring about some regularity to a completely random Poisson process. We demonstrate here two different models for zero-order potentials based on translation invariant kernel function h_0^{TI} and rotation invariant kernel function h_0^{RI}, and collect the resulting fibre systems in Figures 3.25-b,d and 3.25-c, respectively.

The linear fibre system shown in Figure 3.25-b is generated against a zero-order potential given by equation 3.39 based on a kernel function $h_0^{TI}(\hat{\boldsymbol{\varphi}})$ which depends on the angle between a fibre's unit direction $\hat{\boldsymbol{\varphi}}$ and a given fixed vector V ($h_0^{TI}(\hat{\boldsymbol{\varphi}}) = |\hat{\boldsymbol{\varphi}} \cdot V|^k$, $k = 1.5$, and the density parameter $c_0 = 6.0$). For the fibre system illustrated in Figure 3.25-c, the zero-order potential β^{RI} is given by equation 3.40, where $h_0^{RI}(p) = p_y/d_y(\mathcal{X}_P)$ and $d_y(\mathcal{X}_P)$ is the y-size of the simulation domain $\mathcal{X}_P$ ($c_0 = 2.5$). This is an example of inhomogeneous Poisson linear fibre system whose statistical properties vary along the simulation domain $\mathcal{X}$.

The fibre system shown in Figure 3.25-d is another example of inhomogeneous

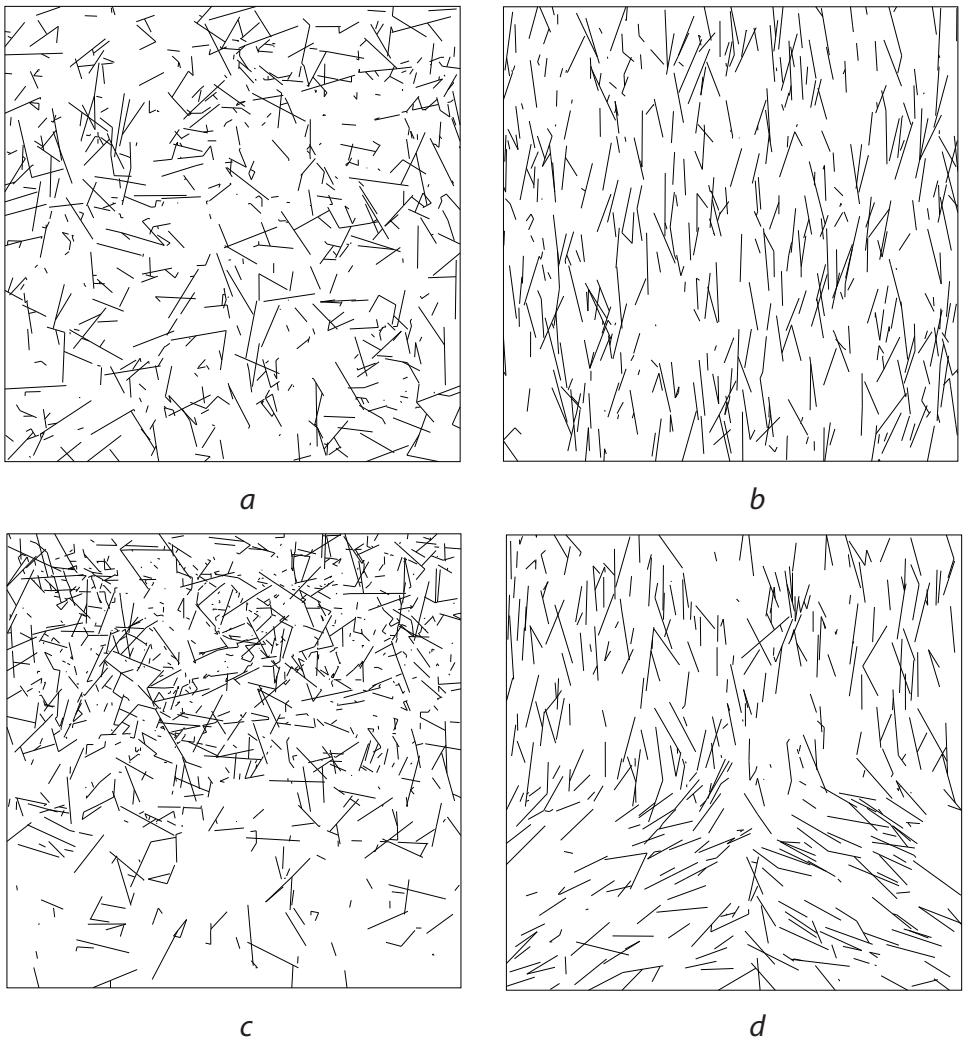


Figure 3.25: Poisson fibre processes: homogeneous (a) versus inhomogeneous (b-d).

Poisson linear fibre system with a zero-potential depending on a vector field $V(p)$ illustrated in Figure 3.40-d: $\beta(\varphi;V) = c_0 \int_0^{l_\varphi} du\, h_0(\hat{\boldsymbol{\varphi}}(u), \varphi(u); V)$. A kernel for such a β is given by $h_0(\boldsymbol{v}, p; V) = |\boldsymbol{v} \cdot V(p)|^k$. For the example we choose $k = 2.0$ and $c_0 = 6.0$. We use the same phase space for all the examples illustrated in Figure 3.25: $\mathcal{X}_P = [0,7]^2$, $\mathcal{X}_L = [0,1]$, and $\mathcal{X}_\mathcal{W} = [0, 2\pi)$; and also $R_\Delta = 0.01$.

In this work we generate samples of *Gibbsian linear fibre processes* based on the following probability density function

$$f(\Phi) = \exp\Big\{ \sum_i \beta_0 l_{\varphi_i} + \sum_i \tilde{h}_1(\varphi_i) + \sum_{\substack{i<j \\ \varphi_i \underset{R}{\sim} \varphi_j}} \tilde{h}_2(\varphi_i, \varphi_j) \Big\} / \mathcal{Z}, \tag{3.56}$$

where first-order potentials $\tilde{h}_1$ and pair-potentials $\tilde{h}_2$ are given in their general form by equations 3.34 and 3.28, respectively, and $\Phi = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n$.

We focus only on piecewise constant interaction functions $h_{\boldsymbol{\theta}}$, given by equation 3.41, which define multi-variate point pair-potentials h in the interaction integrals of the first-order potentials and pair-potentials, so that, for example, our pair-potential has the following form

$$\tilde{h}_2(\varphi, \psi) = c_2 \int_0^{l_\varphi} du \int_0^{l_\psi} dv \;\; h_{\boldsymbol{\theta}}\big(\boldsymbol{\rho}(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\Psi(v))\big), \tag{3.57}$$

The distance functional $\boldsymbol{\rho}$ is a vector function defined as follows

$$\begin{aligned} \boldsymbol{\rho}(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\Psi(v)) &= (\rho_d(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\Psi(v)),\ \rho_w(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\Psi(v))), \\ \rho_d(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\Psi(v)) &= \|\varphi(u) - \psi(v)\|, \\ \rho_w(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\Psi(v)) &= g_w(\delta_\Omega(\hat{\boldsymbol{\varphi}}(u),\ \hat{\boldsymbol{\psi}}(v))), \end{aligned} \tag{3.58}$$

where

$$\begin{aligned} \delta_\Omega(\boldsymbol{v}_1, \boldsymbol{v}_2) &= |\omega(\boldsymbol{v}_1) - \omega(\boldsymbol{v}_2)|, \\ g_w(\delta w) &= \frac{\pi}{2} - \left| \frac{\pi}{2} - \delta w \right|, \end{aligned}$$

and $\omega(\boldsymbol{v})$ is the minimum angle made from the positive x-axis to $\boldsymbol{v}$ or $-\boldsymbol{v}$ and $\hat{\boldsymbol{\varphi}}(u)$ is the tangent at $\varphi(u)$.

For easier interpretation of the interaction function $h_{\boldsymbol{\theta}}$ we convert the sample values $\theta_{\alpha\beta}$ to their logarithmic versions, so that $h_{\boldsymbol{\theta}}$ will be parameterized by point interaction rate constants $\gamma_{\alpha\beta}$ as follows

$$h_{\boldsymbol{\theta}}(d,w) \;=\; \sum_{\alpha,\beta} \ln(\gamma_{\alpha\beta})\;\; \chi_{[R_{\alpha-1},R_\alpha)}(d)\;\; \chi_{[w_{\beta-1},w_\beta)}(w)\,. \tag{3.59}$$

In this case, we call the fibre system model *inhibiting* within the corresponding ranges of distances and angles $[R_{\alpha-1},R_\alpha)$ and $[w_{\beta-1},w_\beta)$ if the interaction rates are smaller than one, $\gamma_{\alpha\beta}<1$; *non-interacting* if the rates equal to one, $\gamma_{\alpha\beta}=1$; and *attracting* if the the rates are greater than one, $\gamma_{\alpha\beta}>1$.

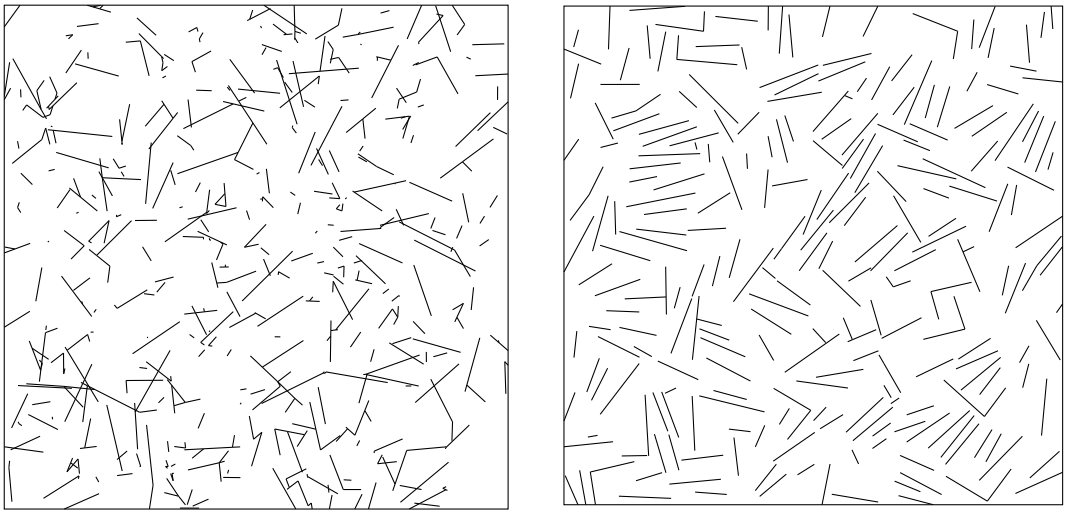


Figure 3.26: (a) Poisson homogeneous linear fibre process versus (b) linear fibre process with inhibiting model favoring cross and aligned configuration of close fibres.

Example 12.2 Gibbsian linear fibre system with a "cross-pattern"

Let us set a phase space components as $\mathcal{X}_P=[0,7]^2$, $\mathcal{X}_L=[0,1]$, and $\mathcal{X}_W=[0,2\pi)$.

The intensity of the zero-order term is chosen as $\beta_0 = 5.5$. The proximity radius is set to $R_\Delta = 0.025$. Point interaction function $h_{\boldsymbol{\theta}}$ is defined by its constant values within corresponding ranges of distances and angles presented in Table 3.4. Figure 3.27-a shows a superimposed polar plots of the point interaction function for two radii $R_1 = 0.1$ (orange curve) and $R_2 = 0.25$ (green curve). Fibre systems based on the fibre potential models with such interaction rates favor cross configuration of close fibres.

	$[0, w_1)$	$[w_1, w_2)$	$[w_2, w_3)$	$[w_3, w_4)$	$[w_4, w_5)$
$0 \leq d < R_1 = 0.1$	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
$R_1 \leq d < R_2 = 0.25$	1	10^{-6}	10^{-6}	10^{-6}	1

Table 3.4: Interaction rates $\gamma_{\alpha\beta}$ which define the values of point interaction function $h_{\boldsymbol{\theta}}$ constant within corresponding ranges of distances $[R_{\alpha-1}, R_\alpha)$ and angles $[w_{\beta-1}, w_\beta)$. Orientation angles are $w_1 = 18°$, $w_2 = 36°$, $w_3 = 54°$, $w_4 = 72°$, and $w_5 = 90°$.

Strong regularity in a fibre system distributed with such inhibiting interaction model is clearly observed in Figure 3.26-b when compared with a sample of Poisson homogeneous liner fibre model depicted in Figure 3.26-a.

3.7 User control on synthesis

Besides changing the "internal" state of a fibre system, like the parameters of its underlying interaction model, one can also modulate the fibre system "externally" to achieve necessary behavior of the system by imposing certain constraints to the system or by making the internal parameters to depend on some external field. In this

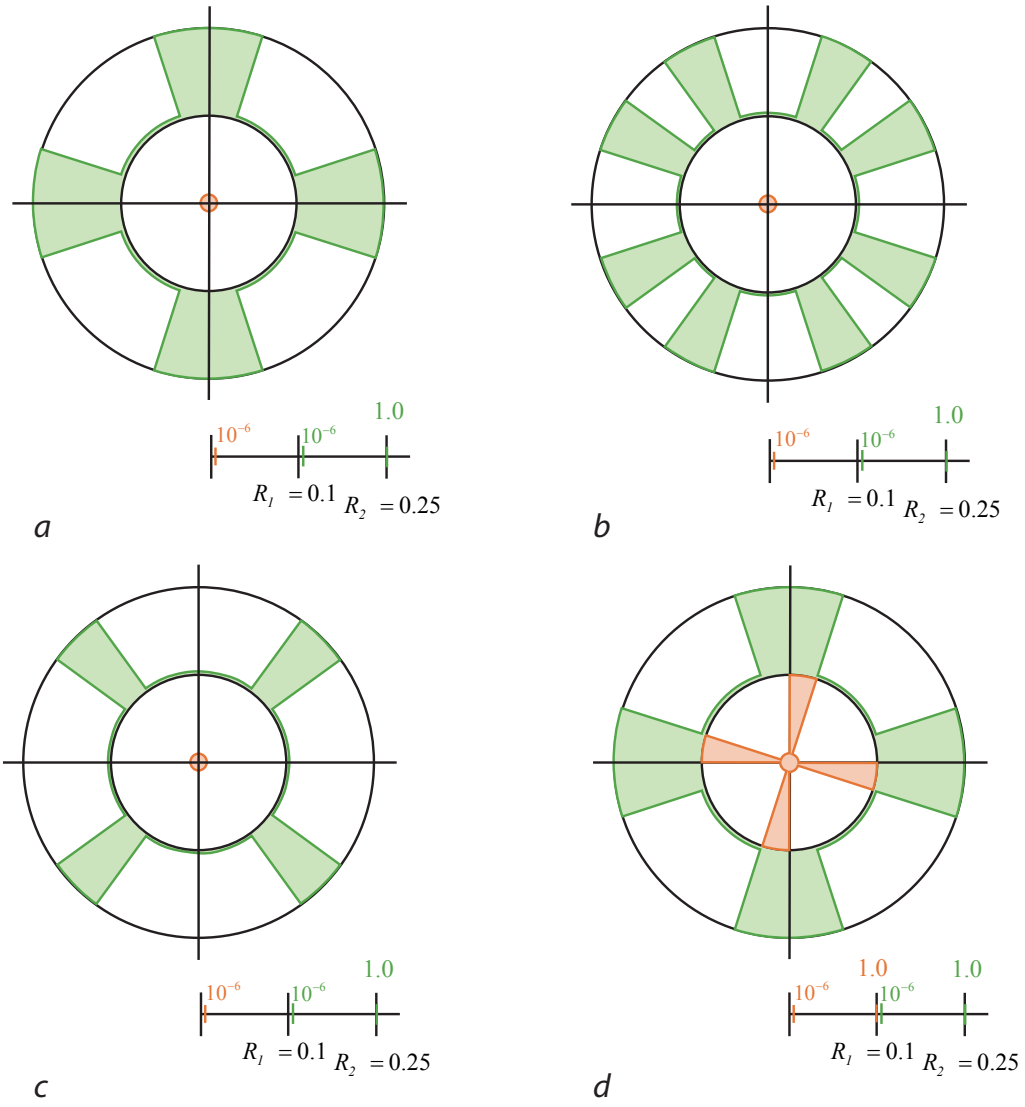


Figure 3.27: Polar plots of different point interaction functions $h_{\boldsymbol{\theta}}$ given by the following tables: (a) Table 3.4 (b) Table 3.5, (c) Table 3.6, and (d) Table 3.7. Orange curves define the values $h_{\boldsymbol{\theta}}(0,w)$, while green curves correspond to $h_{\boldsymbol{\theta}}(0.1,w)+1$.

section we introduce a set of control which enables users to generate more regular fibre systems with specified fibre density, with enforced connectivity between the fibres, and with a given degree of local or global alignment with a given target vector field or a set of curvilinear features. We give a general description of fibre models under constraints based on a simulated annealing algorithm and weight functions for user control in Section 3.7.1 with providing more detail and examples in the sections which follow. Distance based constraints with enforce attaining a given total length of the fibre system and/or a given degree of connectivity of the fibres are explained in Sections 3.7.2 and 3.7.3, respectively. Fibre models of hard- and soft-constraints support fibre system alignment along a given orientation field and are based on weight functions are described in Sections 3.7.4 and 3.7.5, respectively.

3.7.1 User control models through distance based constraints and weighted models

From many different techniques to enable an additional control on an original simulation model, in this work we mainly focus on a technique of imposing constraints to the fibre interaction model and a method of introducing varying weights within the fibre interaction kernel. The main idea of the first technique is to simulate the original fibre system within a constraint space. A possible implementation of this model is to run the original MCMC algorithm described in Section 3.3.4 and reject those fibre system candidates which leave the constraint space. The second method introduces a series of weights within the interaction model kernels which regulate the degree of alignment of the fibre system with a given vector field globally or with a given set of curvilinear features locally, near the feature medial axes. When the weight functions of the first type are used, we refer to it as applying “hard-constraints”, while the

second type is attributed to "soft-constraints".

A method of introducing constraints to the model of fibre system can be demonstrated on a simple example from the conventional probability theory of generating random point samples from a given distribution. Suppose that we want to generate samples of 2D points $\boldsymbol{x} = (x_1, x_2)$ normally distributed around the origin, $f_{\boldsymbol{x}} \sim N(\mathbf{0}, D)$, and located on a given ellipse of radius R, $\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = R^2$. This problem can be solved by applying an inverse transform sampling [120] if the cdf of just described distribution can be explicitly found and inverted. However, it would not be feasible to apply this method for the case of the random fibre systems because their probability measures contain irresolvable partition functions. Instead, we can simulate a Markov Chain by running Metropolis-Hastings algorithm with a proper transition pdf $q(\boldsymbol{y}|\boldsymbol{x})$ and an acceptance probability $A(\boldsymbol{y}|\boldsymbol{x})$. A constraint for a point to be on an ellipse can be expressed by the Euclidean distance to the ellipse $\rho_e(\boldsymbol{x}) = 0$. We denote by $f(\boldsymbol{x} \,\&\, \rho_e(\boldsymbol{x}) = 0)$ a conditional pdf of the points distributed normally and located on the ellipse. To satisfy the detailed balance equation the following expression for the acceptance probability should be used

$$
\begin{aligned}
A(\boldsymbol{y}|\boldsymbol{x}) &= \min\left\{1,\ \frac{f_e(\boldsymbol{y} \,\&\, \rho_e(\boldsymbol{y}) = 0)}{f_e(\boldsymbol{x} \,\&\, \rho_e(\boldsymbol{x}) = 0)}\ \frac{q(\boldsymbol{y}|\boldsymbol{x})}{q(\boldsymbol{x}|\boldsymbol{y})}\right\} \\
&= \min\left\{1,\ \frac{f_e(\boldsymbol{y}|\rho_e(\boldsymbol{y}) = 0)}{f_e(\boldsymbol{x}|\rho_e(\boldsymbol{x}) = 0)}\ \frac{f_c(\rho_e(\boldsymbol{y}) = 0)}{f_c(\rho_e(\boldsymbol{x}) = 0)}\ \frac{q(\boldsymbol{y}|\boldsymbol{x})}{q(\boldsymbol{x}|\boldsymbol{y})}\right\}
\end{aligned}
$$

It is highly likely that $f_c(\rho_e(\boldsymbol{y}) = 0) = 0$ most of the time which will lead to a very high rejection rate during MH steps. This results in a Markov chain which will never generate any reasonable sample at all.

Relaxing the distance constraint temporarily is an alternative successful approach which is a variant of a well-known simulated annealing algorithm, the technique which we will be using in this work for enforcing distance-based constraints on fibre

system models. The main idea of the simulated annealing algorithm is to include an additional "annealing" energy term to the original probabilistic model. Such energy term may represent how "far" is a given sample from satisfying some criterion, or constraint. In our example, such term depends on the distance of a sample to the ellipse. What makes this method unique is that a candidate for the energy term also depends on a relaxation parameter t which intuitively represents a "temperature". The simulated annealing algorithm starts its iterations with high temperature values and gradually cools the temperature down. When the temperature is high, most of the samples will be accepted according to the original unconstraint model, even though they may have a large "distance" to the constraint space. After a certain number of steps the temperature is gradually decreased, so that the samples "far" from satisfying the constraints are penalized more due to decreasing annealing energy term, and are more likely to be rejected. The following expression is a general form for the annealing energy functional

$$E(t,\boldsymbol{x}) = \frac{\exp\{-\rho_c(\boldsymbol{x})/t\}}{\widetilde{\mathcal{Z}}}, \tag{3.60}$$

where $\widetilde{\mathcal{Z}}$ is a partition function depending on a temperature parameter t. A combined pdf for sampling a distribution containing an annealing term is given by the following formula

$$f_{saa}(t,\boldsymbol{x}) = f(\boldsymbol{x})\,E(t,\boldsymbol{x}). \tag{3.61}$$

Back to our example, pdf is given by $f(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi|D|}} e^{-\boldsymbol{x}D\boldsymbol{x}}$ and $\rho_c(\boldsymbol{x}) = x_1^2/a^2 + x_2^2/b^2 - R^2$.

We define a distance based annealing energy functional for the fibre systems in the same way as we did it for point distributions above

$$E(t,\Phi) = \frac{\exp\{-\rho_c(\Phi)/t\}}{\widetilde{\mathcal{Z}}}, \tag{3.62}$$

so that the combined pdf becomes

$$f_c(t,\Phi) = f(\Phi)\,E(t,\Phi). \tag{3.63}$$

The only new additional requirement for the annealing term to be valid within the framework of the fibre systems is its conservation for different partitions of the same fibre system: the value $E(t,\Phi)$ should be the same for any valid partition $\varphi'_1 \circ \varphi'_2 \circ \ldots \circ \varphi'_m$ of a fibre system Φ and a temperature value t. This can be easily reduced to the following *distance conservation condition*

$$\begin{aligned} \Phi = \varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n = \varphi'_1 \circ \varphi'_2 \circ \ldots \circ \varphi'_m, \\ \rho_c(\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n) = \rho_c(\varphi'_1 \circ \varphi'_2 \circ \ldots \circ \varphi'_m). \end{aligned} \tag{3.64}$$

The Metropolis-Hastings algorithm for generating fibre systems, which was described earlier in Section 3.6.1, requires a simple update for the transition likelihood functions r_b and r_d, which were defined in equations 3.48 and 3.49, respectively. For example, the ' 'birth" likelihood function $r_b(\psi,\Phi)$ becomes

$$r_b(t,\psi,\Phi) = \lambda_c(\psi,\Phi)\,\lambda_E(t,\psi,\Phi)\frac{1-q(\Phi\circ\psi)}{q(\Phi)}\;\frac{d(\psi,\Phi)}{b(\psi,\Phi)}, \tag{3.65}$$

with

$$\lambda_E(t,\psi,\Phi) = \exp\left\{-\frac{\rho_c(\Phi)-\rho_c(\psi\cup\Phi)}{t}\right\}. \tag{3.66}$$

The "death" likelihood function r_d is updated in the same way.

A simulated annealing algorithm based on the updated MH (*SA-MH*) iterations may consist of the following straightforward steps:

1. Start with some large temperature value $t = t_0$;

2. Repeat the following steps until $t = t_M$ (some small temperature value);

2.1. Perform N MH iterations for a current temperature value;

2.2. Decrease the temperature by some value Δt, $t = t - \Delta t$.

In the next two sections we describe two types of distance functionals which define a fibre total length constraint and a fibre connectivity constraint.

In contrast with the external constraints, the weighted models explicitly change the original model by introducing varying weights. For our Gibbsian fibre system distributions we scale their kernel functions by some weights $W(\cdot)$ which are specifically designed to emphasize particular portions of each fibre and de-emphasize the rest based on the fibre location and orientation. In our framework of linear fibres, interaction potentials are given by the following formulas:

$$\beta(\varphi) = c_0 \int_0^{l_\varphi} du \;\; h_0\left(\boldsymbol{x}^\varphi(u)\right), \tag{3.67}$$

$$\tilde{h}_2(\varphi,\psi) = c_2 \int_0^{l_\varphi} du \int_0^{l_\psi} dv \;\; h\big(\boldsymbol{\rho}\left(\boldsymbol{x}^\varphi(u),\boldsymbol{x}^\psi(v)\right)\big). \tag{3.68}$$

We define the following corresponding weighted potential models β^{W_0} and $\tilde{h}_2^W$

$$\beta^{W_0}(\varphi) = c_0 \int_0^{l_\varphi} du \;\; h_0\left(\boldsymbol{x}^\varphi(u)\right) W_0\left(\boldsymbol{x}^\varphi(u)\right), \tag{3.69}$$

$$\tilde{h}_2^W(\varphi,\psi) = c_2 \int_0^{l_\varphi} du \int_0^{l_\psi} dv \;\; h\big(\boldsymbol{\rho}\left(\boldsymbol{x}^\varphi(u),\boldsymbol{x}^\psi(v)\right)\big) \;\; W\left(\boldsymbol{x}^\varphi(u),\boldsymbol{x}^\psi(v)\right), \tag{3.70}$$

so that the total energy functional $\tilde{U}$, defined by equation 3.35, with its potentials replaced by their weighted versions for given weight functions $W_0(\cdot)$ and $W(\cdot)$, become

$$\tilde{U}^W(\varphi_1 \circ \varphi_2 \circ \ldots \circ \varphi_n) = \sum_i \beta^{W_0}(\varphi_i) + \sum_i \tilde{h}_1^W(\varphi_i) + \sum_{\substack{i<j \\ \varphi_i \underset{R}{\sim} \varphi_j}} \tilde{h}_2^W(\varphi_i,\varphi_j). \tag{3.71}$$

Example 12.3 Weighted homogenous Poisson processes

Suppose that a zero-order interaction kernel is constant, $h_0(\boldsymbol{x}^{\varphi}(u)) \equiv b$, and a weight function is given by $W_0(\boldsymbol{x}^{\varphi}(u)) = \frac{1}{2}w(\varphi(u)) + \frac{1}{2}$ (i.e., the weight function depends only on the fibre position). The interaction kernels of higher orders are set to zero. Then the corresponding zero-order potential β becomes

$$\beta^{W_0}(\varphi) = c_0 \int_0^{l_\varphi} du \; b\left(\frac{1}{2}w(\varphi(u)) + \frac{1}{2}\right) = c_0\, b\left(\frac{1}{2}\tilde{w}(\varphi) + \frac{1}{2}\right) l_\varphi\,.$$

The weight $\tilde{w}(\varphi) = \int_0^{l_\varphi} w(\varphi(u))du/l_\varphi$ represents "an average" weight along the fibre φ and regulates the fibre density of the resulting model. When the weight value is close to one, the model turns out to be an original Poisson model with intensity $c_0\, b$, while as $\tilde{w}$ tends towards zero the resulting fibre system will be half dense comparing to the one of the original model.

An example of a fibre system distributed with the density similar to kernel $\beta^{W_0}(\varphi)$ has been already generated earlier and shown in Figure 3.25-c.

3.7.2 Total length constraints

It was observed that the point processes with hard inhibition between the closely located points usually generate more regular configuration of points than the processes with soft inhibition do. As our Gibbs fibre interaction models are based on the point potential integrals (see the equation 3.28), we expect to observe more regularity for the fibre processes with hard point inhibition. However, the samples of hard inhibition fibre processes tend to be underpopulated, so that one should maintain a certain level of fibre density to have a proper coverage of the simulation domain by fibres.

One of the methods to control the density of fibre system during synthesis is to

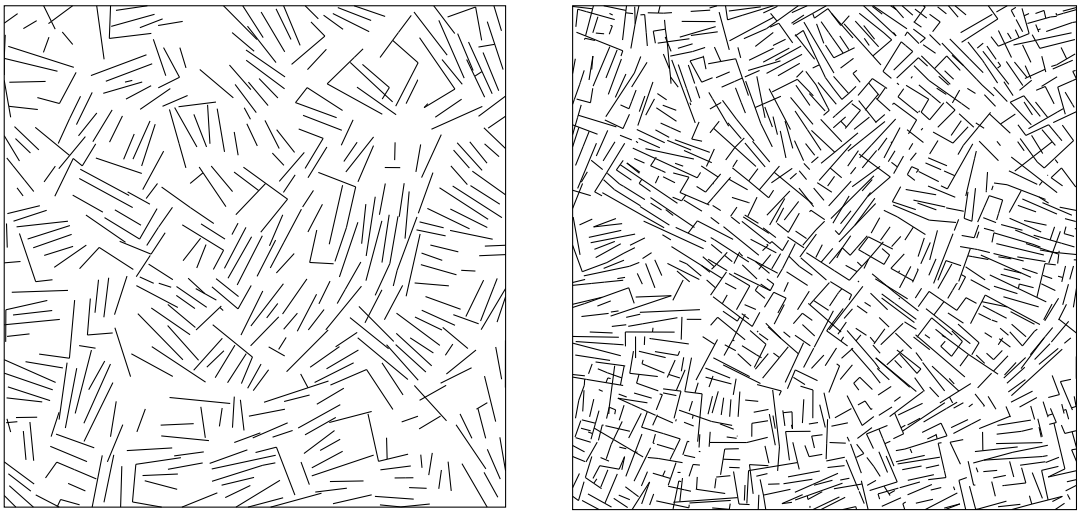
impose lower and upper bounds for the total length of the system, $l_T < l_\Phi = \sum_{\varphi_j \in \Phi} l_{\varphi_j} < l_{\max}$, and keep the length of the resulting fibre system between the specified bounds. In principle, it is possible to achieve this goal by performing trial and error analysis changing the parameters of the zero-order interaction kernel 3.33. However, it requires running many consecutive MH runs to select a right parameter. It is more efficient to run simulated annealing MH algorithm once by providing the range $[l_T, l_{\max}]$ as a constraint of simulation. To use the algorithm described in the previous section we have to define proper distance function $\rho_c(\Phi)$ which addresses the fibre system total length condition.

Let δl be the half size of the desired range of the total length values $[l_T, l_{\max}]$, i.e., $\delta l = \frac{l_{\max} - l_T}{2}$. For l_Φ — the total length of the fibre system Φ — to be within just specified range is equivalent to $|l_\Phi - l| \leq \delta l$, where $l = l_T + \delta l$. Satisfying such inequality is equivalent to keeping the value for the following distance function equal to zero:

$$\rho_{l,\delta l}(\Phi) = \big(|l_\Phi - l| - \delta l\big)_+ = 0. \tag{3.72}$$

This distance function clearly satisfies the distance conservation condition given by equation 3.64 as it depends on the total distance of the fibre system which is partition-invariant. Therefore, we can apply the simulated annealing MH algorithm (*SA-MH*) described in the previous section to generate fibre systems with defined l_T and δl. *Comment:* we found the the simulation algorithm is more stable when the parameter δl is as small as the maximum possible length of the fibre in the phase space. The resulting values l_Φ for generated fibre systems were mostly concentrated around given l_T.

Figure 3.28 shows two intermediate states of a fibre system generated by SA-MH algorithm. The fibre system has the same distribution as the one described in the ex-



a *b*

Figure 3.28: Snapshots of fibre systems corresponding to different SA-temperature values $t = 0.075$ and $t = 0.025$.

ample 12.2 in Section 3.6.3 (see a fibre system sample Figure 3.26-b). The constraints for this fibre system are chosen as $l_T = 350$ and $\delta l = 5$. The temperature parameter varies in the range $[0.025, 1.0]$. Figure 3.28 shows two snapshots corresponding to the temperature values 0.075 and 0.025, respectively. The fibre system total length values at those temperature values are $l_\Phi = 211$ and $l_\Phi = 347$ (as oppose to $l_\Phi = 140$ for the unconstrained fibre system depicted in Figure 3.26-b), respectively.

Fibre systems shown in Figure 3.28 demonstrate that with increasing fibre density a fibre system constrained on the l_Φ becomes more regular.

Figure 3.29 shows examples of different fibre system models under the same total length constraint $l_T = 350$. SA-temperature is in the range $[0.01, 1.0]$ for all the examples. The fibre system shown in Figure 3.29-a is distributed as the system from the previous example, and the fibre system shown in Figure 3.29-b has the same distribution but smaller phase space $\mathcal{X}$: the allowable fibre length was reduced

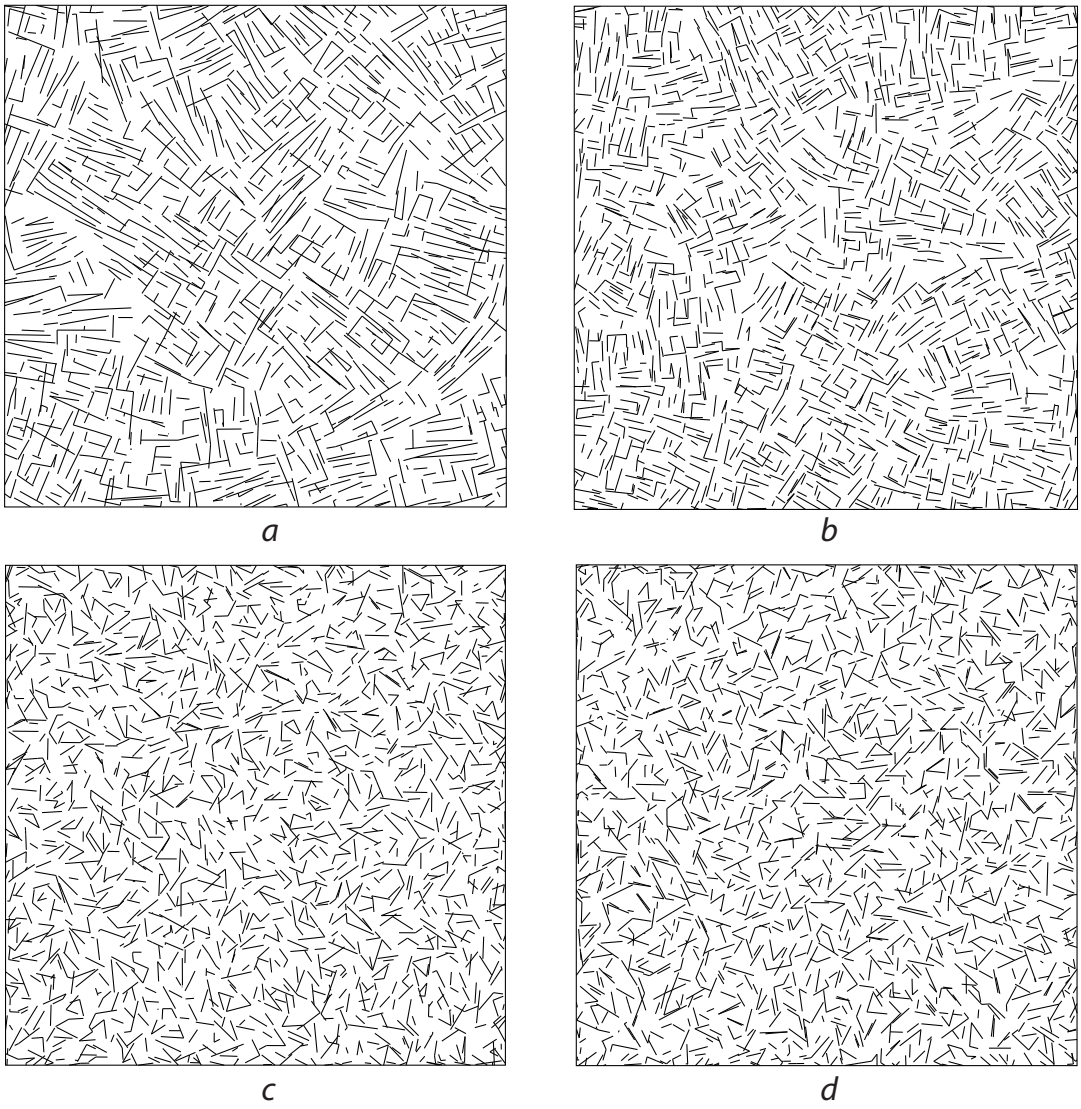


Figure 3.29: Different linear fibre systems generated under the same total length constraint, $l_T = 350$.

roughly two times, $\mathcal{X}_L = [0.1, 0.5]$. The fibre systems shown in Figure 3.29-c and Figures 3.29-d were generated from models whose pair interaction potentials favor $\{30^\circ, 60^\circ\}$ and 45° angles between the fibres, respectively. Here is more detail on point interaction model for the two last models:

Example 12.4 Linear fibre system with a model favoring $\{30^\circ, 60^\circ\}$ interaction angles (Figures 3.29-c)

A phase space of this model is given by the following components $\mathcal{X}_P = [0,7]^2$, $\mathcal{X}_L = [0, 0.5]$, and $\mathcal{X}_W = [0, 2\pi)$. The intensity of the zero-order term is chosen as $\beta_0 = 5.5$. The proximity radius is set to $R_\Delta = 0.025$. Point interaction function $h_{\boldsymbol{\theta}}$ is defined by its constant values within corresponding ranges of distances and angles presented in Table 3.5. Figure 3.27-b shows a superimposed polar plots of the point interaction function of this fibre model.

	$[0, w_1)$	$[w_1, w_2)$	$[w_2, w_3)$	$[w_3, w_4)$	$[w_4, w_5)$
$0 \leq d < R_1 = 0.1$	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
$R_1 \leq d < R_2 = 0.25$	10^{-6}	1	10^{-6}	1	10^{-6}

Table 3.5: Interaction rates $\gamma_{\alpha\beta}$ which define the values of point interaction function $h_{\boldsymbol{\theta}}$ constant within corresponding ranges of distances $[R_{\alpha-1}, R_\alpha)$ and angles $[w_{\beta-1}, w_\beta)$. Orientation angles are $w_1 = 18^\circ$, $w_2 = 36^\circ$, $w_3 = 54^\circ$, $w_4 = 72^\circ$, and $w_5 = 90^\circ$.

Example 12.5 Linear fibre system with a model favoring 45° interaction angle(Figures 3.29-d)

A phase space of this model is given by the following components $\mathcal{X}_P = [0,7]^2$, $\mathcal{X}_L = [0,0.5]$, and $\mathcal{X}_W = [0,2\pi)$. Other parameters are set to $\beta_0 = 5.5$, $R_\Delta = 0.025$. Point interaction function $h_{\boldsymbol{\theta}}$ is defined by its constant values within corresponding ranges of distances and angles presented in Table 3.6. Figure 3.27-c shows a superimposed polar plots of the point interaction function of this fibre model.

	$[0,w_1)$	$[w_1,w_2)$	$[w_2,w_3)$	$[w_3,w_4)$	$[w_4,w_5)$
$0 \le d < R_1 = 0.1$	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
$R_1 \le d < R_2 = 0.25$	10^{-6}	10^{-6}	1	10^{-6}	10^{-6}

Table 3.6: Interaction rates $\gamma_{\alpha\beta}$ which define the values of point interaction function $h_{\boldsymbol{\theta}}$ constant within corresponding ranges of distances $[R_{\alpha-1},R_\alpha)$ and angles $[w_{\beta-1},w_\beta)$. Orientation angles are $w_1 = 18^\circ$, $w_2 = 36^\circ$, $w_3 = 54^\circ$, $w_4 = 72^\circ$, and $w_5 = 90^\circ$.

One should be careful in choosing the total length lower and upper bounds: the pattern may become deteriorated when the lower bound is chosen too large. A fibre system in Figure 3.29-d can be considered as an example of the constrained simulation with overkilling value for lower bound l_T. A "clearer" sample of the same fibre interaction model but with smaller value for l_T is shown in Figure 3.30-a (in this case $l_T = 176$ and minimum temperature value was 0.025). An obvious difference between the original version, shown in Figure 3.30-b, and the clearer version of the same model is in multiple overlapping of aligned and closely located fibres in the case of the denser fibre system.

There is probably no deterministic way of selecting optimal values for minimum and maximum SA-temperature parameter. We usually choose it trial and error manner by choosing a lower bound value at which the system is not updated much during

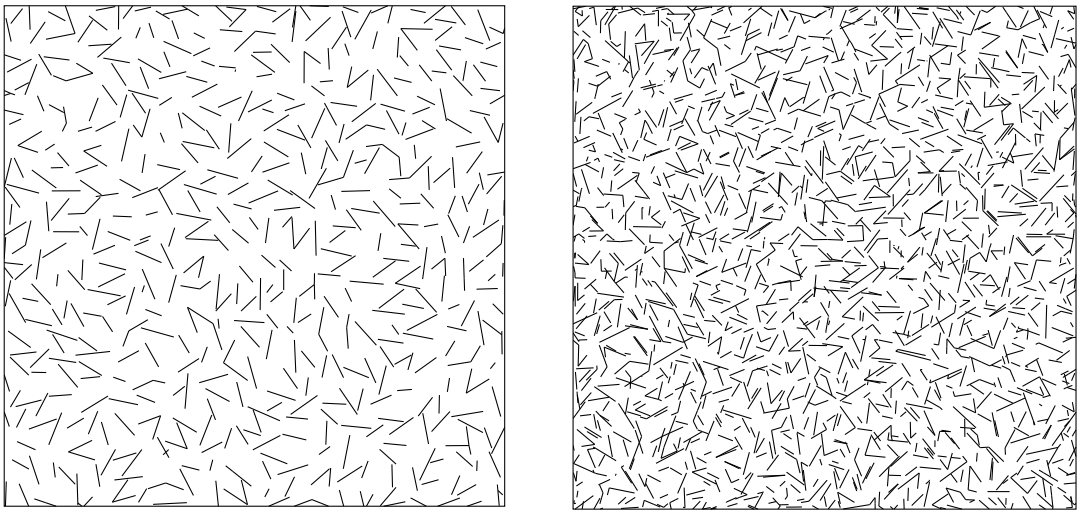


Figure 3.30: Possible pattern deterioration.

simulation for a relatively long sequence of MH steps.

3.7.3 Connectivity constraint

All examples of the fibre systems demonstrated so far are based on inhibiting interaction models - described in a nutshell, only certain configurations of fibres are favored when they are close to each other, while other configurations are penalized. Such models are more effective in producing regular patterns than the attracting and Poisson models are. However, a majority of fibres will be isolated making the resulting fibre systems scarcely inter-connected (such fibres, for example, can be seen in the fibre systems shown in Figure 3.29). To improve connectivity of the fibres we propose two strategies: one is to define a distance-based constraint to enforce the fibre connectivity, and the other one is to run a coupled process which connects close fibres based on a predefined set of rules.

A fibre system Φ can be represented as an undirected planar graph $G_\Phi = (V_\Phi, E_\Phi)$ embedded to a plane which supports the simulation domain. Fibres form the edges of the graph, E_Φ, while the fibre end-points and intersection points of the fibres constitute the graph's vertices V_Φ. In this case the connectivity of the fibres can be expressed in the language of the corresponding connectivity of the graph primitives.

The first strategy can be implemented in two ways by either imposing global or local constraints. According to the first method, one would penalize the disconnected components of a current fibre system graph by defining a constraint distance to be equal to the number of connected components minus one, $\rho_{cn}(\Phi) = \#_{\text{conn-comp}}(G_\Phi) - 1$. In this case, when all the fibres are connected, the constraint distance is zero. For optimal implementation, one can maintain online bi-connected components which allow performing quickly a test weather a new connected component is created when a fibre is deleted (this would decrease the constraint energy and more likely reject the deletion event). We leave the actual implementation for future, and concentrate in this work more on local constraints which are faster and simpler to implement.

The idea behind our model of local connectivity constraints is to penalize the fibres which have at least one hanging end-point. In the language of graphs, it means that their underlying edges have at least one vertex with degree zero. During a fibre system simulation we maintain a corresponding undirected planar graph G_Φ, updating the graph whenever a fibre is added to or deleted from the system. A penalizing term is calculated for every edge in the graph. A constraint distance $\rho_{cn}(\Phi)$ enabling local connectivity constraint is a function of the underlying planar graph G_Φ and is defined by the following simple rules:

1. $\rho_{cn}(\Phi) \equiv \rho^G_{cn}(G_\Phi) = \sum_{j=1}^{|E_\Phi|} \rho^E(G_\Phi; e_j)$ where $e_j \in E_\Phi$;

2. For a given undirected planar graph $G = (V, E)$ and a graph's edge $e \in E$,

$\rho^E(G;e)$ is defined as follows:

2.1. $\rho^E(G;e) = 0$, if $\deg\{V(e,0)\} \neq 0$ and $\deg\{V(e,1)\} \neq 0$;

2.2. For an edge $e \in E$ with one vertex $v_0 = V(e,k)$, $k = 0 \vee 1$, first, we traverse a "linear chain" of edges in the graph starting from the vertex v_0 and calculate its length $r'_{lin}(v_0)$ (a linear chain starting from a given vertex is a set of adjacent edges $\{e_{j_n} \in E, n = 1,2,\ldots,N\}$, such that e_{j_1} is adjacent to the vertex; for each $n < N$, $e_{j_n} \neq e_{j_{n+1}}$ and the edge e_{j_n} is adjacent to the edge $e_{j_{n+1}}$; the vectors supporting the edges are collinear; and, either e_{j_N} has an adjacent vertex with degree zero or it is adjacent to a non-collinear edge which is different from $e_{j_{N-1}}$); second, we find an edge e_0 which was not encountered in the traversal and is the closest to v_0 — let the distance to e_0 from v_0 is denoted by $r'_{ngb}(v_0)$; third, we reduced the resulting distances $r'_{lin}(v_0)$ and $r'_{ngb}(v_0)$ to be as large as the interaction radius R_M [see equation 3.41], i.e., $r_{lin}(v_0) = \min\{r'_{lin}(v_0), R_M\}$ and $r_{ngb}(v_0) = \min\{r'_{ngb}(v_0), R_M\}$; and finally, $\rho^E(G;e) = \min\{r_{lin}(v_0), r_{ngb}(v_0)\}$;

2.3. For an edge $e \in E$ with $\deg\{V(e,0)\} = 0$ and $\deg\{V(e,1)\} = 0$ we assign $\rho^E(G;e) = \min\{|e|, r_{ngb}(V(e,0)), r_{ngb}(V(e,1))\}$, where $r_{ngb}(\cdot)$ was defined above.

We used the following notation above: $\deg\{v\}$ is the degree of a graph's vertex $v \in V$, $V(e,0)$ and $V(e,1)$ are the "first" and "second" adjacent vertices of a given graph's edge $e \in E$.

Finally, we mix connectivity constraint with the total length constraint defined in the previous section to form a model for generating systems with connected fibres. The constraint distance for the model is a linear combination of the corresponding constraint distances of the connectivity model components with a control parameter

w_{cn}:

$$\rho_c(\Phi) = w_{cn}\rho_{cn}(\Phi) + \rho_{l,\delta l}(\Phi)\,. \tag{3.73}$$

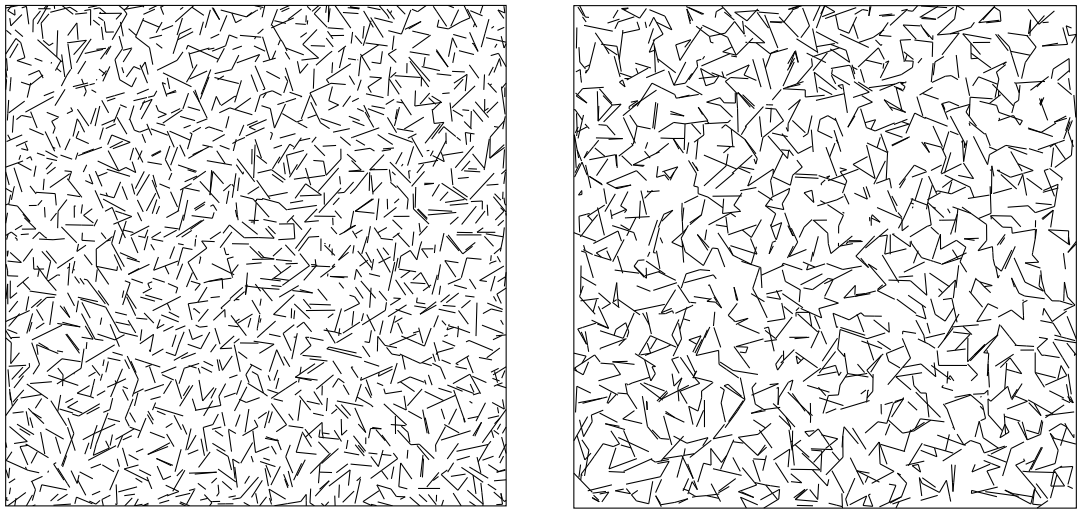


Figure 3.31: Fibre process with the local connectivity constraint.

Linear fibre system with total length constraint described in example 12.5 is placed side-by-side with a fibre system generated from the same model but with connectivity constraints in Figure 3.31 (a) and (b), respectively. R_M is set to the value of the maximum interaction radius and the connectivity weight $w_c = 1$. SA-temperature is in the range $[0.005, 1.0]$.

The second strategy to improve connectivity of the fibres is to apply a relaxation scheme to the simulation model. The fibre interaction model will be relaxed during MH-algorithm generation run to address the fibre connectivity: the closely located fibres are merged, linked, or split during simulation run. We distinguish three different scenarios for fibre relaxation depending on the type of arrangement of the closely

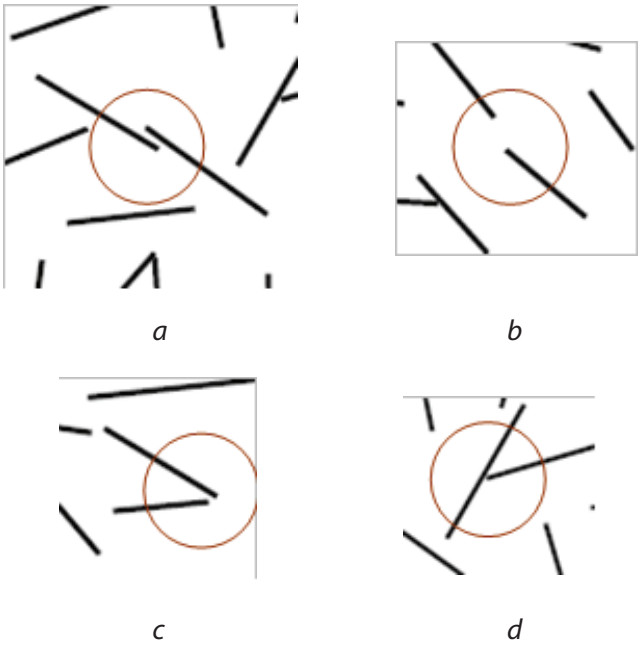


Figure 3.32: Examples of candidates for fibre relaxation. Close fibres are either connected or split at the encircled locations depending on the type of their arrangement with respect to each other. Possible types are (a) overlapping aligned fibres, (b) aligned fibres, (c) close fibres, and (d) an end-point of one fibre is close to the interior of the other fibre.

located fibres: $fibre \times fibre$, $point \times point$, and $point \times fibre$. In the first scenario, $fibre \times fibre$, shown in Figure 3.32-a, two aligned fibres located in the close proximity to each other are merged to form a new fibre. For this scenario as well the others which follow we introduce a relaxation radius, R_x, which defines the size of the proximity area. For the $point \times point$ scenario, which examples are shown in Figure 3.32-b and -c, we find two fibres whose end-points are less than R_x apart. Such fibres are connected to each other at a new point which is an average of the end-points. In the $point \times fibre$ scenario a fibre with its interior being close to an

end-point of another fibre, see Figure 3.32-d, is split into two fibres at the closest point and the resulting fibres are linked to the fibre in the proximity.

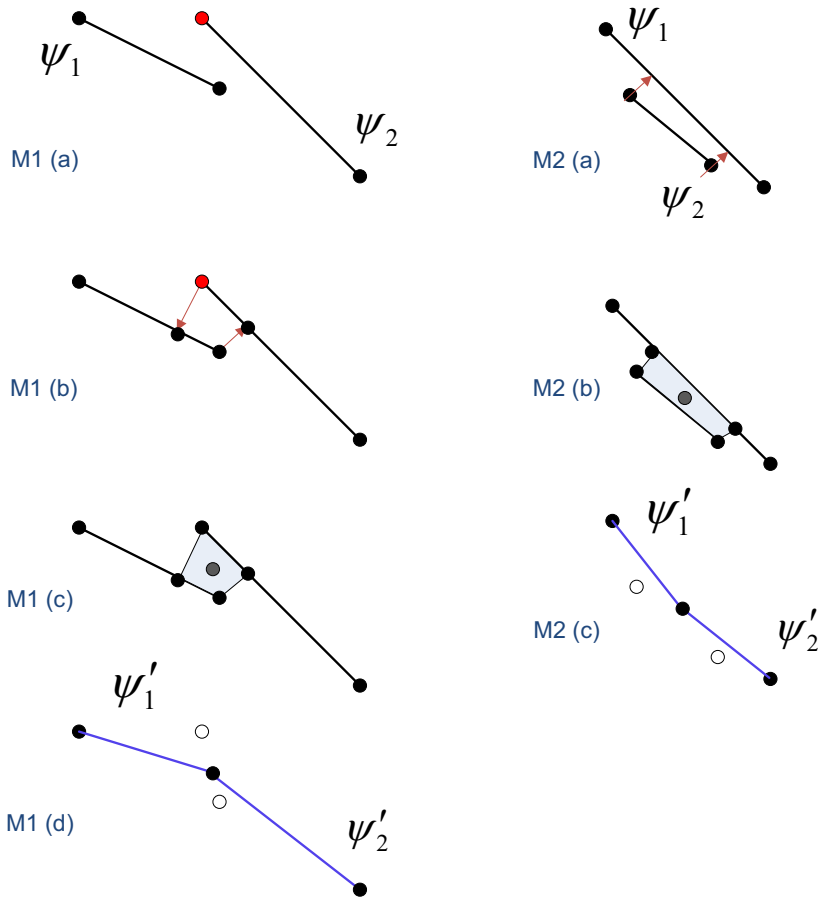


Figure 3.33: (M1) An example of *point* $\times$ *point* relaxation scheme which replaces two fibres ψ_1 and ψ_2 located nearby by two connected fibres ψ_1' and ψ_2'. (M2) Merging nearby fibres ψ_1 and ψ_2 in *fibre* $\times$ *fibre* relaxation scheme.

Examples of sequences of relaxation steps resolving all the scenarios are graphically described in Figures 3.33 and 3.34. Red points in the figures correspond to the points which are within a relaxation distance R_x from nearby fibres. Red arrows

represent a projection of corresponding points onto nearby fibres. The vertices of the polygons filled with gray are averaged to form new vertices (colored in gray) which are used as new end-point for fibres. At the end of the relaxation scheme the fibres involved in the relaxation will be connected and slightly dislocated from their original positions.

Relaxation scheme is implemented as a process coupled to our constrained Metropolis-Hastings (MH) fibre simulation algorithm. It is invoked at certain times during simulation to relax a specified number of fibres. The parameters of the combined algorithm are the relaxation radius R_x, an angle threshold w_x for an alignment test between two close fibres, number of MH steps n_{cpld} between consecutive runs of coupled relaxation scheme, and the number of relaxation steps n_{link} within each such run.

We applied the relaxation algorithm to the fibre systems shown in Figures 3.29-b and 3.29-d (see the discussion to these figures for more detail on the fibre models). Figure 3.35 shows the results of the simulations, where the following parameters were used for the relaxation component: (a) $R_x = 0.075$, $w_x = 0.3$ radians, $n_{\text{cpld}} = 250$ and $n_{\text{link}} = 50$, while SA-temperature of the simulated annealing component of the algorithm is within the range $[0.01, 0.5]$; (b) $R_x = 0.075$, $w_x = 0.3$ radians, $n_{\text{cpld}} = 500$, $n_{\text{link}} = 200$, SA-temperature is within the range $[0.005, 0.075]$.

3.7.4 Hard-constraints: weighted models to enforce aligning with orientation vector field

Using weighted models is another powerful mechanism for generating new types of random fibre systems. In this work we develop a number of weighted models which enforce the fibres to be aligned with a given vector field.

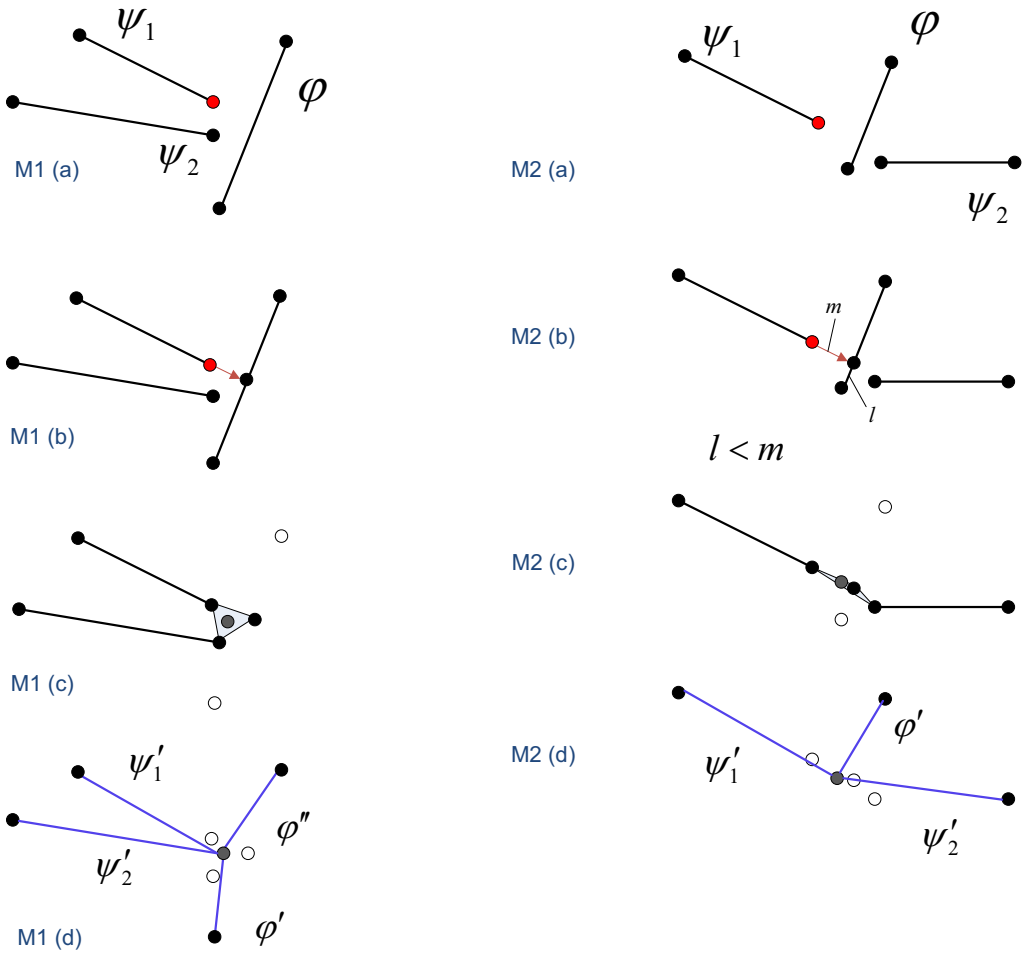
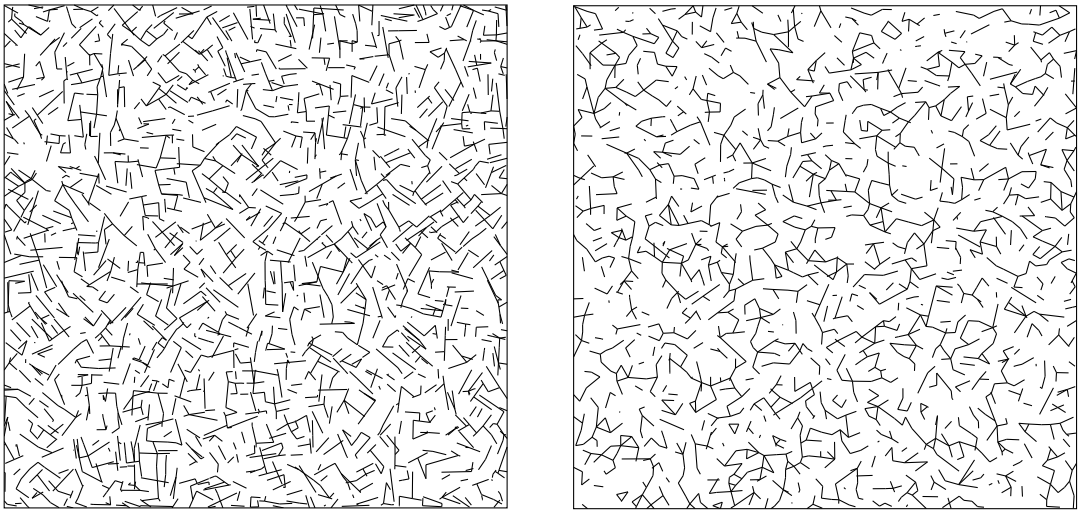


Figure 3.34: Two examples of applying fibre relaxation scheme for more than two fibres being in the relaxation proximity. (M1) an extended *point* $\times$ *fibre* relaxation scheme with replaces three fibres ψ_1, ψ_2, and ϕ by their linked versions ψ_1', ψ_2', ϕ', and ϕ''. (M2) Same scenario as in M1 but with deleting a portion of the fibre ϕ which length, l, is smaller than the relaxation proximity radius R_x.



a *b*

Figure 3.35: Fibre systems generated by applying relaxation schemes to the original models shown (a) in Figure 3.29-b and (b) in Figure 3.29-d.

Suppose that we are given a vector field $V : \mathbb{R}^2 \to \mathbb{E}$ (for the sake of brevity, in some situations we will use V_p to denote a vector $V(p)$). We need to introduce an *alignment weight function* $w(v_1, v_2)$ which defines the degree of alignment between a given pair of vectors v_1 and v_2. In this work we consider the following four different alignment weight functions which we label by W1,...,W4:

$$\text{W1: } w(v_1,v_2) = |v_1 \cdot v_2| \,, \qquad \text{W2: } w(v_1,v_2) = |v_1 \cdot v_2|^2 \,,$$

$$\text{W3: } w(v_1,v_2) = |v_1 \cdot v_2|^3 \,, \qquad \text{W4: } w(v_1,v_2) = \sqrt{|v_1 \cdot v_2|} \,.$$

Alignment weighted models β^{W_0} and $\tilde{h}_2^W$ are given by

$$\beta^{W_0}(\varphi) = c_0 \int_0^{l_\varphi} du \;\; h_0(\boldsymbol{x}^\varphi(u)) \; W_0\Big(V; \boldsymbol{x}^\varphi(u)\Big) \,, \tag{3.74}$$

$$\tilde{h}_2^W(\varphi,\psi) = c_2 \int_0^{l_\varphi} du \int_0^{l_\psi} dv \;\; h\big(\boldsymbol{\rho}(\boldsymbol{x}^\varphi(u), \boldsymbol{x}^\psi(v))\big) \;\; W\big(V; \boldsymbol{x}^\varphi(u), \boldsymbol{x}^\psi(v)\big) \,. \tag{3.75}$$

The simplest weighted model which imposes the aligning requirement is based on scaling kernel function of zero-order potential and leaving pair-potential term intact. A simple form of this model was described in Example 12.3. More general model is given by

$$\textbf{amL}(\lambda)\textbf{:} \qquad W_0\Big(V;[\varphi(u),\hat{\boldsymbol{\varphi}}]\Big) = L_\lambda\left(w(V_{\varphi(u)},\,\hat{\boldsymbol{\varphi}})\right), \tag{3.76}$$

$$W\Big(V;[\varphi(u),\hat{\boldsymbol{\varphi}}],\,[\psi(v),\hat{\boldsymbol{\psi}}]\Big) = 1\,, \tag{3.77}$$

where L_λ is a parametric linear function with values in $[0\ldots1]$, and it is given by

$$L_\lambda(w) = \lambda \bullet (1,w) \equiv\ \lambda w + (1-\lambda)\,. \tag{3.78}$$

Parameter λ is used to emphasize or to suppress the contribution of the weight function W_0 to zero-order potential. We denote this model by **amL**(λ). An example of applying **amL**($3/4$) to the cross-pattern fibre model, which was described earlier in example 12.2 constrained to the total length $l_T = 350$, to enforce alignment of the fibre system with the "VP-up" vector field (see Figure 3.40-a) is shown in Figure 3.36-d. Majority of fibres are aligned well with the vector field, while the rest of the fibres respect the cross-pattern model by being perpendicular the fibres from the majority.

The other two weighted models which we propose in this work incorporate the weight functions into both zero-order and second-order potential kernels, and are given in the form of 3.69 and 3.70. Here is the intuition behind these models. We assume that the kernel functions of zero-order potentials are always nonnegative, $h_0 \geq 0$, while kernel functions of pair-potentials are non-positive, $h \leq 0$. In this case the weights control the contribution balance between zero-order and second-order potentials. If the weights W_0 and W for a pair of close fibres turn out to be equal, then the fibres will be distributed according to the original model and respect the

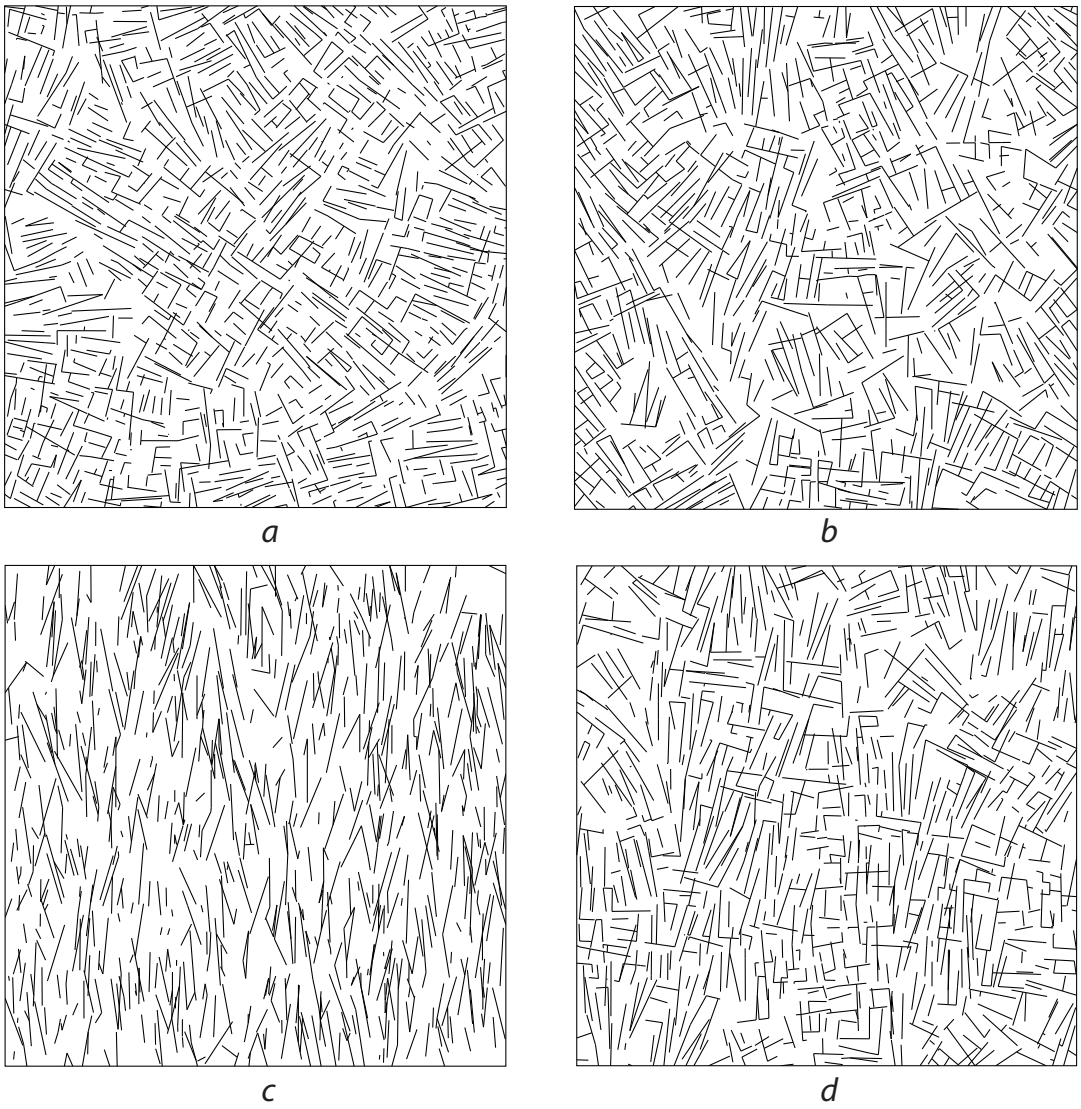


Figure 3.36: Original cross-pattern model, defined in the description to Figure 3.29-a, is superimposed with different alignment weighted models applied to the original model: (b) model am2-02, with W1 alignment weight function; (c) model am1, W1; (d) model amL(3/4), W1. Vector field VF-up was applied for (b), (c), and (d) (shown in Figure 3.40-a)

same pattern possibly with changed local density depending on the weight values. However, if the first weight W_0 is relatively small, then the affected fibres tend to repulse each other according to the inhibition model of the pair-potential, $\tilde{h}_2$, while for the second weight W to be relatively small, the configuration of two fibres become constrained Poisson distributed.

The first model, denoted by **am1**, turns to constrained Poisson for the fibres aligned with a given vector field. The fibres under a constrained Poisson distribution are aligned with a feature (in this case, to the given vector field) and randomly deviate from the feature up to a defined level. Misaligned fibres will be distributed according to the original model 3.67 and 3.68. The weight functions forming am1 model are given by

$$\textbf{am1:} \qquad W_0\Big(V;[\varphi(u),\hat{\boldsymbol{\varphi}}]\Big) = 1-\bar{w}\left(V_{\varphi(u)},\hat{\boldsymbol{\varphi}}\right), \tag{3.79}$$

$$W\Big(V;[\varphi(u),\hat{\boldsymbol{\varphi}}],\ [\psi(u),\hat{\boldsymbol{\psi}}]\Big) = \bar{w}\left(V_{\varphi(u)},\hat{\boldsymbol{\varphi}}\right) + \bar{w}\left(V_{\psi(v)},\hat{\boldsymbol{\psi}}\right), \tag{3.80}$$

where $\bar{w}(\cdot) = \frac{1-w(\cdot)}{2}$. As for amL, this model is also flexible in choosing the alignment weight function $w(\cdot,\cdot)$. An example of the fibre system distributed according to am1 with alignment weight function W1 and a target vector field VF-up is shown in Figure 3.36 (c). The resulting fibres indeed strongly follow the vector field; however, the cross-pattern of the original model [shown in Figure 3.36 (a)] is entirely lost.

Our second model, **am2-α**, while preserving the original pattern better than the first model does, generates a "substantial" number of fibres aligned with the target vector field. The weight functions which define the model am2-α are defined as follows

$$\textbf{am2-}\boldsymbol{\alpha}\textbf{:} \qquad W_0\Big(V;[\varphi(u),\hat{\boldsymbol{\varphi}}]\Big) = 1+\alpha\, w\left(V_{\varphi(u)},\hat{\boldsymbol{\varphi}}\right), \tag{3.81}$$

$$W\Big(V;[\varphi(u),\hat{\boldsymbol{\varphi}}],\ [\psi(v),\hat{\boldsymbol{\psi}}]\Big) = 1+\frac{\alpha}{2}\left\{w\left(V_{\varphi(u)},\hat{\boldsymbol{\varphi}}\right) + w\left(V_{\psi(v)},\hat{\boldsymbol{\psi}}\right)\right\}, \tag{3.82}$$

Examples of fibre systems generated under the weighted model am2-2 (i.e., $\alpha = 2.0$) applied to a version of the model for cross-pattern (whose point interaction model is given in Table 3.7) with different alignment weight function are shown in Figure 3.37.

The conceptual difference between am2-α and am1 is in that a couple of fibres, both

	$[0, w_1)$	$[w_1, w_2)$	$[w_2, w_3)$	$[w_3, w_4)$	$[w_4, w_5)$
$0 \leq d < R_1 = 0.1$	10^{-6}	10^{-6}	10^{-6}	10^{-6}	1
$R_1 \leq d < R_2 = 0.25$	1	10^{-6}	10^{-6}	10^{-6}	1

Table 3.7: Interaction rates $\gamma_{\alpha\beta}$ which define the values of point interaction function $h_{\boldsymbol{\theta}}$ constant within corresponding ranges of distances $[R_{\alpha-1}, R_\alpha)$ and angles $[w_{\beta-1}, w_\beta)$. Orientation angles are $w_1 = 18^\circ$, $w_2 = 36^\circ$, $w_3 = 54^\circ$, $w_4 = 72^\circ$, and $w_5 = 90^\circ$. Point interaction chart is shown in Figure 3.27-d.

aligned or both misaligned with the target vector field, will be distributed closely to the original model, with giving slightly more preference to the pairs of aligned fibres (in fact, the only parameter of the model, $\alpha \in [1,2]$, controls the preference rate). Indeed, for values $w_{\varphi(u)}$ and $w_{\psi(v)}$ to be both relatively small or both relatively large, the resulting values of the weight functions W_0 and W are approximately the same. Here we use the following simplification $w_{\varphi(u)} \equiv w\left(V_{\varphi(u)}, \hat{\boldsymbol{\varphi}}\right)$. When the degrees of alignment of a given pair of fibres are substantially different, a preference is given to the fibre which is better aligned with the vector field. Example of the fibre system distributed according to a am2-2 model with W1 alignment weight function is shown in Figure 3.36 (b). The Table 3.8 summaries the weighted models which we proposed so far to generate fibre systems aligned with a given target vector field.

Examples of generated fibre systems aligned with more complex vector fields are shown in Figure 3.38.

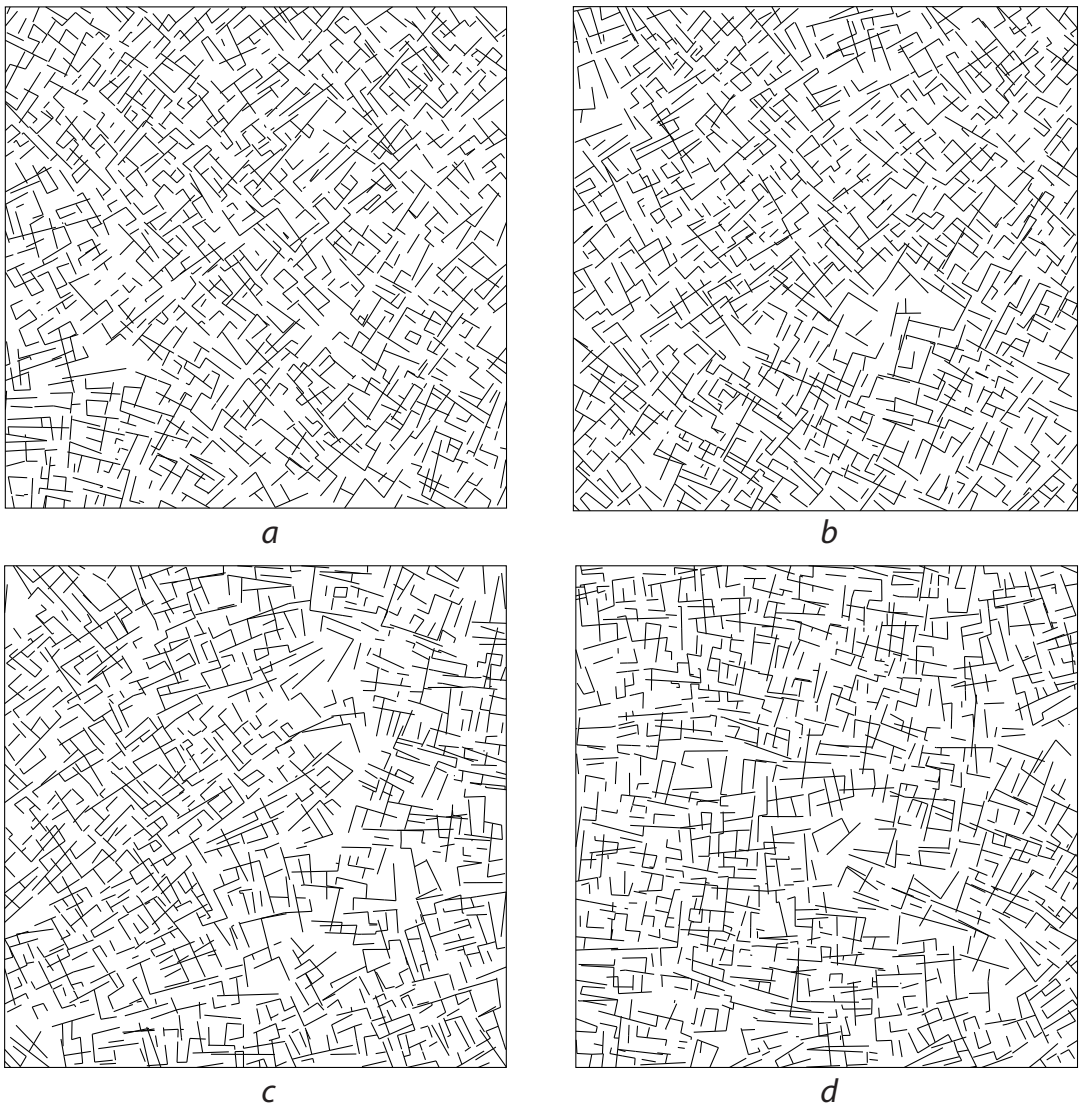
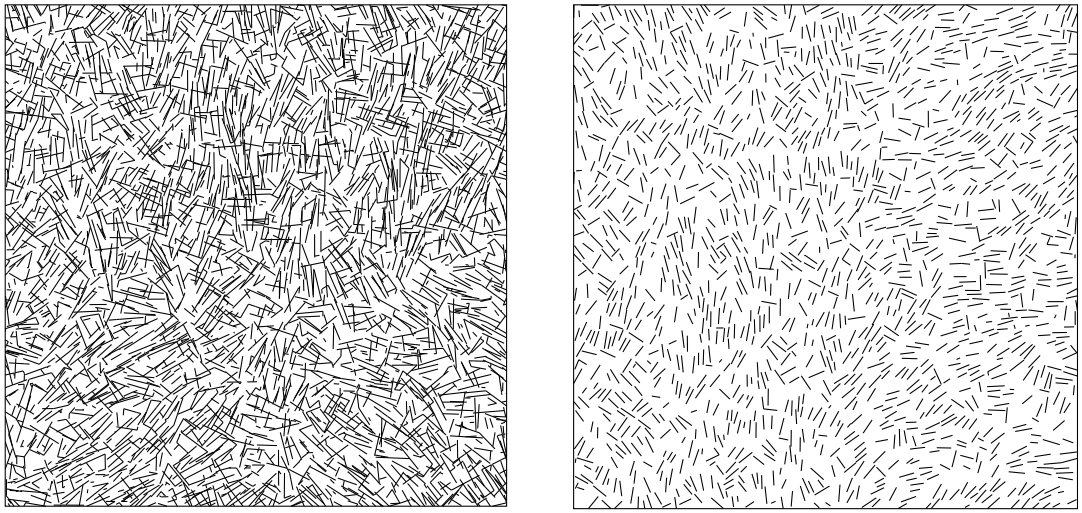


Figure 3.37: Different alignment weight functions applied to the weighted model am2-2. (a) W4, (b) W1 (vector field shown in Figure 3.40-b was used); (c) W2, (d) W3 (vector field depicted in Figure 3.40-a was used). Fibre interaction model is given in Table 3.7.

		W_0	W
amL(λ)	$\lambda \in [0,1]$	$\lambda \bullet \left(1, w_{\varphi(u)}\right)$	1
am1		$1 - \bar{w}_{\varphi(u)}$	$\bar{w}_{\varphi(u)} + \bar{w}_{\psi(v)}$
am2-α	$\alpha \in [1,2]$	$1 + \alpha\, w_{\varphi(u)}$	$1 + \frac{\alpha}{2}\left(w_{\varphi(u)} + w_{\psi(v)}\right)$

Table 3.8: Different models for weight functions W_0 and W which form a basis for the vector field alignment models.



a *b*

Figure 3.38: Examples. (a) VF-trn , model am1, W1 (b) VF-rght, model am2-2, W4; vector fields are shown in Figure 3.40-d and -c, respectively

We want to stress here that during a fibre system simulation some boundary conditions should be taken into account. Excluding such condition would introduce artifacts in the fibres close to the boundary as they lack the same level of interaction with the neighboring fibres as the fibres located within the interior of the simulation domain do. We apply a wrapping strategy in which an area outside of the simulation domain is populated by the fibres replicated from the area along the opposite side boundary. The size of the area approximately equals to the largest interaction radius R_M. Such replication in the case of a rectangular simulation domain corresponds to gluing opposite edges of the boundary rectangle. The Figure 3.39 shows that a simulation with wrapping [sub-figure (b)] results in fibre system more consistent and better aligned with the vector field along the boundaries than in the case of unwrapped simulation [sub-figure (a)].

3.7.5 Soft-constraints

All the weighted random fibre models described so far represent the fibre interaction models with what can be called external "*hard constraints*": a generated fibre system intends to be aligned to a given target vector field (or, orientation field) everywhere equally within a domain of simulation. "*Soft constraints*" are different: they are applied locally at specific locations, and their strength is modulated based on a distance function. We propose two types of soft constraints: *feature-based* and *area-based* soft constraints. For the feature-based soft constraints the fibres located within a vicinity of a feature set should be aligned with the features. The closer a fibre is to a feature, the better aligned should it be with the feature. The fibres located far enough from the feature set should be distributed according to the original model.

The area-based soft constraints are imposed in the same way as the hard con-

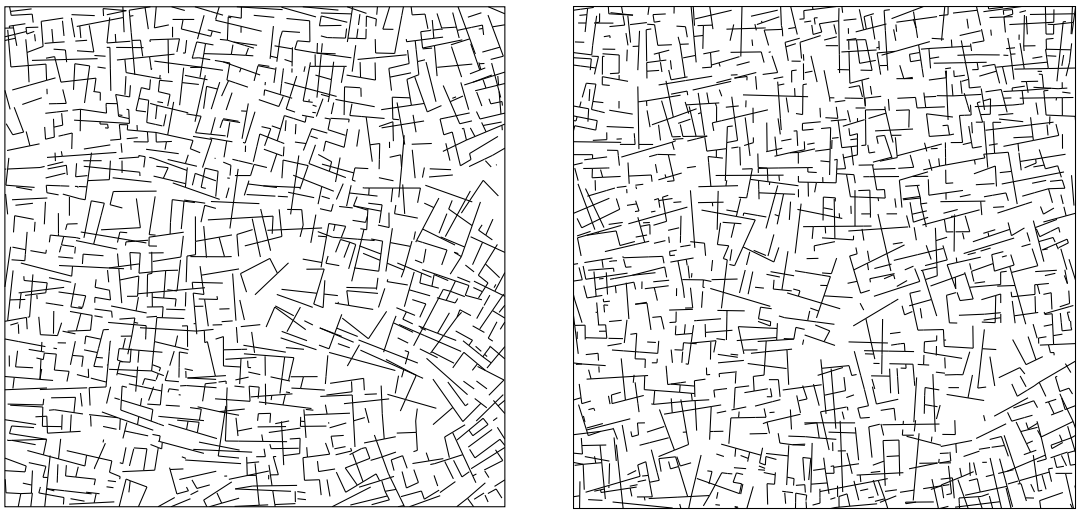


Figure 3.39: Simulation domain wrapping. (a) Without wrapping reveals misalignment of fibres along the east/north boundaries , while (b) with wrapping results in a fibre system better aligned with the target vector field VF-up (see, Figure 3.40-a). The weighted model is am2-2, W3.

straints are, but they are effective only within some (convex) region of the domain. For instance, such region can cover given features. This type of constraint does not require modifying the original fibre interaction model: one can use the model of hard constraints and only introduce a characteristic function of the region which will turn on/off the constraints based on the values of the characteristic function. Thus, for a given region $B \in X$, weight functions W_0^s and W^s defining the area-based soft constraint model can be written as

$$\begin{aligned} W_0^s\left(\boldsymbol{x}^{\varphi}(u)\right) &= \chi_B(\varphi(u)) \bullet \left(1, W_0\left(\boldsymbol{x}^{\varphi}(u)\right)\right), \\ W^s\left(\boldsymbol{x}^{\varphi}(u), \boldsymbol{x}^{\psi}(v)\right) &= \chi_B(\varphi(u))\chi_B(\psi(v)) \bullet \left(1, W\left(\boldsymbol{x}^{\varphi}(u), \boldsymbol{x}^{\psi}(v)\right)\right), \end{aligned}$$

where any hard constraint model for weight functions W_0 and W can be used. The

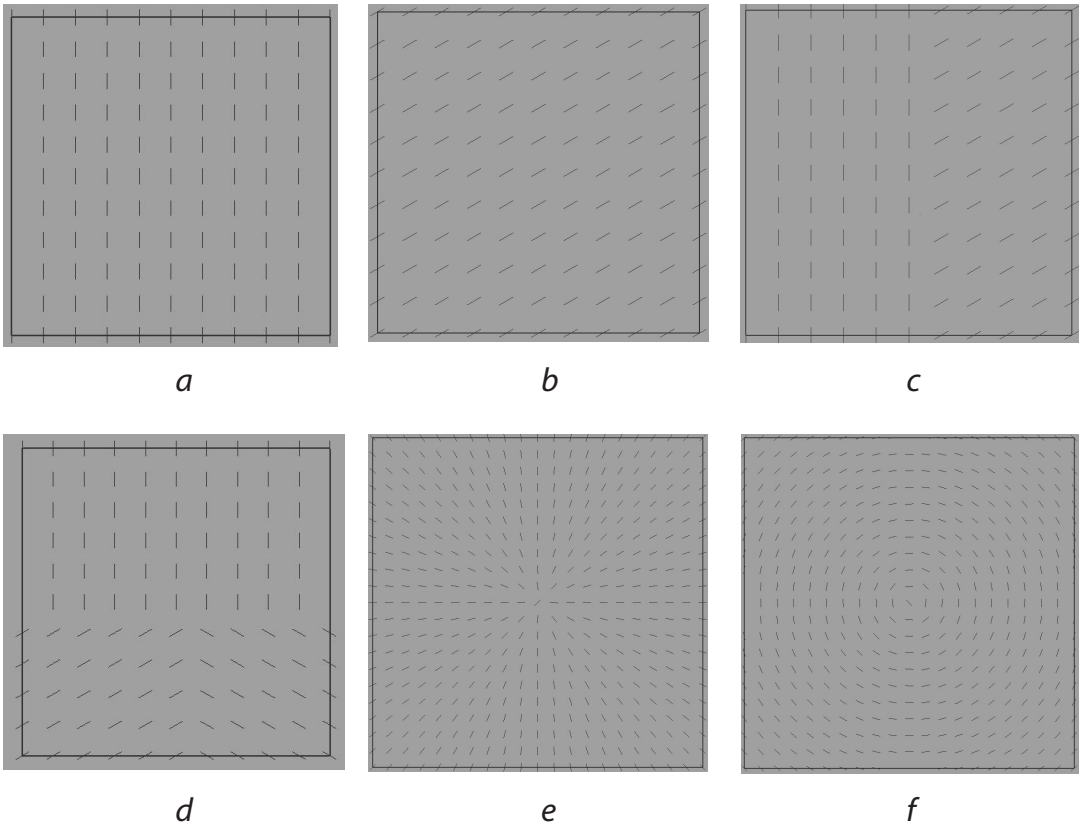


Figure 3.40: Vector fields: (a) VF-up, (b) VF-diag, (c) VF-rght, (d) VF-trn, (e) VF-rad, (f) VF-circ.

corresponding weighted models for zero- and pair-potentials $\beta^{W_0^s}$ and $\tilde{h}_2^{W^s}$ are given by Equations 3.74 and 3.75. In this work we rather investigate the feature-based soft constraint model in more detail, as it enables more gradual transition of the fibre system from being strictly aligned with the feature to become free from the aligning constraint within the effective neighborhood of the feature.

We assume that for a given set of features $\mathsf{F} = \{\mathsf{f}_j\}$ corresponding tangent vector field $V^{\mathsf{F}} : \mathbb{R}^2 \to \mathbb{E}$ and distance map $D^{\mathsf{F}} : \mathbb{R}^2 \to \mathbb{R}_+$ are provided. The set F contains curved one-dimensional features, like line segments, quadratic curves, etc. The field

V^{F} and the distance map D^{F} at a point $p \in X$ are defined as follows

$$V^{\mathsf{F}}(p) = \left\{ t_{j^*}(u^*) \mid (j^*, u^*) = \arg\min_j \min_u \left| p - \mathsf{f}_j(u) \right| \right\},$$
$$D^{\mathsf{F}}(p) = \min_j \min_u \left| p - \mathsf{f}_j(u) \right| \ ,$$

where $t_j(u)$ is a tangent vector of feature f_j at point $\mathsf{f}_j(u)$. The methods to build V^{F} and D^{F} from a given set of features was described in Section 2.5.

To implement the feature-based soft constraints model we, again, use modified version of hard constraints models amL, am1, and am2-α. We relax the hard constraints weight functions depending on the distance to the feature set, so that near the features the feature-based weight functions approximately equal to the hard constrains weight functions and gradually change to 1 when moving away from the features. Let $\alpha(D^{\mathsf{F}}; \cdot) : \mathbb{R}^2 \to [0,2]$ is a *feature influence function* which upon a given point returns a feature influence value based on the distance to the feature set F. We propose the following form for $\alpha(D^{\mathsf{F}}, d_{\max}; \cdot)$ parameterized with an additional parameter $d_{\max}$ which defines the maximum feature influence radius

$$\alpha(D^{\mathsf{F}}, d_{\max}; p) = 2\left(1 - \min\left\{ \frac{D^{\mathsf{F}}(p)}{d_{\max}}, 1 \right\}\right).$$

The Table 3.9 introduces soft constraints weight functions W_0 and W. As in the case of hard constraints, we use the following simplifications $w_{\varphi(u)} \equiv w\left(V^{\mathsf{F}}_{\varphi(u)}, \hat{\boldsymbol{\varphi}}\right)$ and $\alpha_{\varphi(u)} \equiv \alpha\left(D^{\mathsf{F}}, d_{\max}; \varphi(u)\right)$. It is easy to see that when $\alpha_{\varphi(u)}$ is zero, which happens when a fibre is far enough from the feature, all the potential models based on the weight functions described in Table 3.9, turn to the original unconstrained model 3.67 and 3.68 (excluding model asm1 which has slight misbalance towards pair-potential term). On the other hand, when fibres are close to the features the value of the influence weight $\alpha_{\varphi(u)}$ approaches 2, so that the weight functions of

	W_0	W
asmL(λ)	$\lambda \bullet \left(1, 1 + (w_{\varphi(u)} - 1)\frac{\alpha_{\varphi(u)}}{2}\right)$	1
asm1	$1 - \bar{w}_{\varphi(u)}\left(\alpha_{\varphi(u)}\right)$	$\bar{w}_{\varphi(u)}\left(\alpha_{\varphi(u)}\right) + \bar{w}_{\psi(v)}\left(\alpha_{\psi(v)}\right)$
asm2	$1 + \alpha_{\varphi(u)} w_{\varphi(u)}$	$1 + \frac{1}{2}\left(\alpha_{\varphi(u)} w_{\varphi(u)} + \alpha_{\psi(v)} w_{\psi(v)}\right)$

Table 3.9: Different models for weight functions W_0 and W which form a basis for the feature-based soft constraints. $\bar{w}_{\varphi(u)}(\alpha) \equiv \frac{1-\frac{\alpha}{2} w_{\varphi(u)}}{2}$.

asmL and asm1 models become approximately equal to the corresponding models amL and am1 of hard constraints.

The model asm2, though, has different construction: it is built from am2-α by modulating its originally fixed parameter α depending on the distance to the feature set. In short, asm2≡am2-$\alpha(D^{\mathsf{F}}, \cdot)$, with an essential change that we let parameter α vary in $[0, 1]$, so that when $\alpha(D^{\mathsf{F}}, \cdot)$ is close to zero, asm2 turns to the original unconstrained model.

An example of applying asm2 weight model of soft constraints to the linear fibre model of cross-pattern (shown in Figure 3.29-a) is illustrated in Figure 3.41-d. A feature is a line segment going along a SE-NW diagonal of the domain square; see Figure 3.41-b (the short line segments represent the vector field and their length values are proportional to the values of feature influence function α). Distance function has its maximum value $\max_{p \in \mathcal{X}_P} D^{\mathsf{F}}(p) \approx 90.5\,\text{pts}$ and $d_{\max} = 30\,\text{pts}$. It is clear from comparison of the Figures 3.41-c and -d that the fibres of the latter indeed follow the feature only in its proximity, while the fibres of the former are aligned with a diagonal everywhere.

Different alignment weight functions for a fixed weight model reveals different levels of agreement of resulting fibre systems with the feature. Thus, for the model

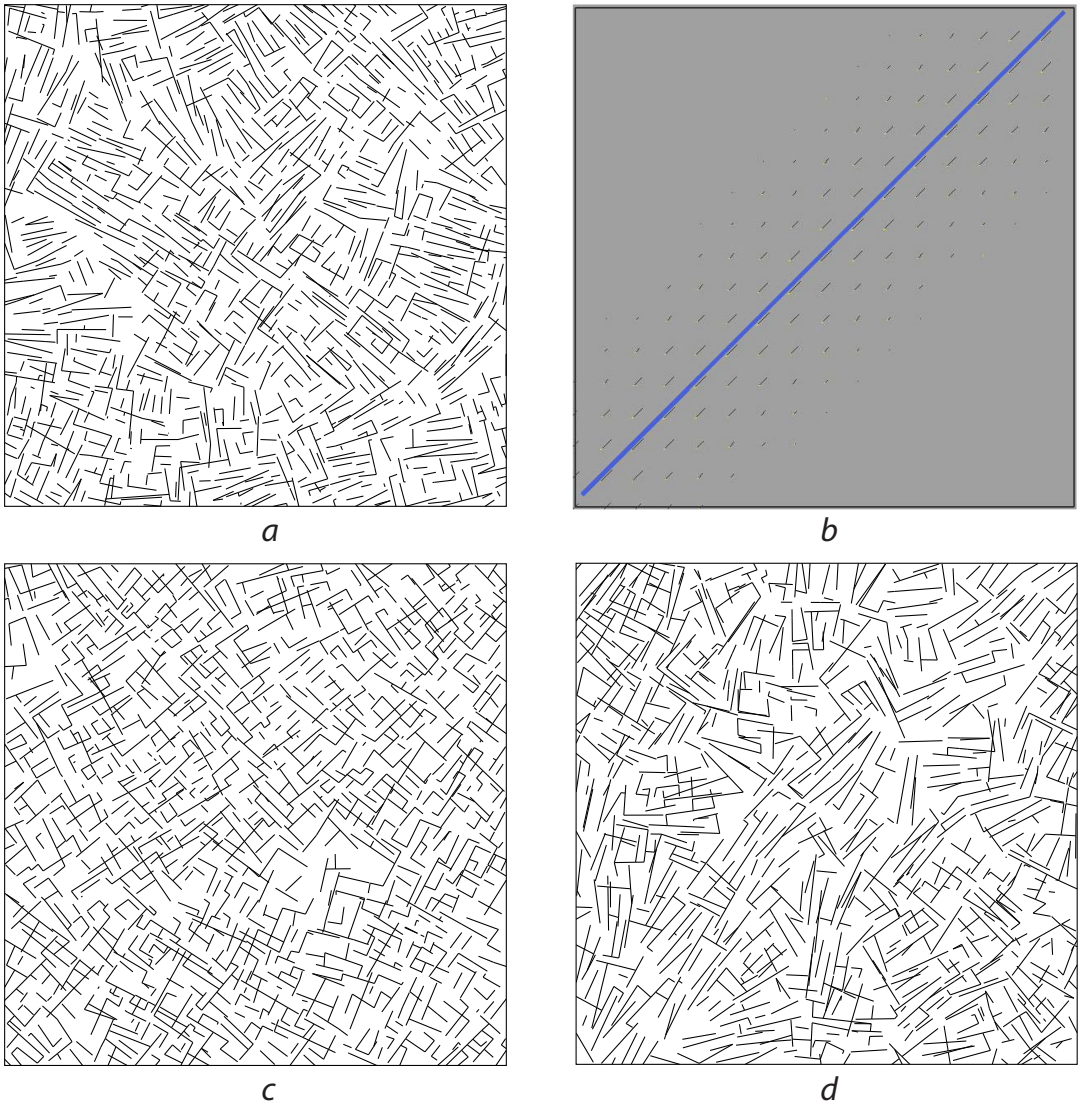


Figure 3.41: Fibre systems built by applying hard (c) and soft (d) constraints to an (original) unconstrained model (a). Soft constraint model: asm2, W1. The feature set consists of a "diagonal" line (b, in blue) connecting bottom-right and top-right corners of the domain. The size of the feature distance map is 128pts.

mentioned in the previous example functions W2 and W3 does a decent job (Figure 3.42-b,c) while the results of applying functions W1 and W4 are less satisfactory (compare with Figures 3.42-a and -d).

It is worth to mention that there are two conceptually different ways of guaranteeing local alignment of the fibre system with the feature set: make sure that a dominant orientation of the fibres matches the tangent field of the feature, or concentrate substantially more fibres along the feature. The goal of the first approach is to keep the distribution of the fibres within the feature's effective neighborhood as close to the original (unconstrained) interaction model as possible while guaranteeing that a majority of fibres are aligned with the feature. The fibre systems shown in Figure 3.43 (a)-(c) indeed have majority of fibres aligned to the "diagonal line" feature (shown in Figure 3.41-b) within its neighborhood and other fibres respecting the "cross-pattern". According to the second approach, the fibres are allowed to deviate from the original (unconstrained) model along the feature, but the concentration of the feature aligned fibres should be significant. It is clearly demonstrated in Figure 3.43 (d): the aligned fibres cluster around the feature and they do not follow the original distribution in there.

Different soft constraint models are applied to the original cross-pattern linear fibre model to make the fibres follow a feature curve outlining the letter "C" in Figure 3.44.

3.8 Summary

We have presented a new general construction for stochastic processes of random fibre systems based on the model of local interactions between the fibres. Random fi-

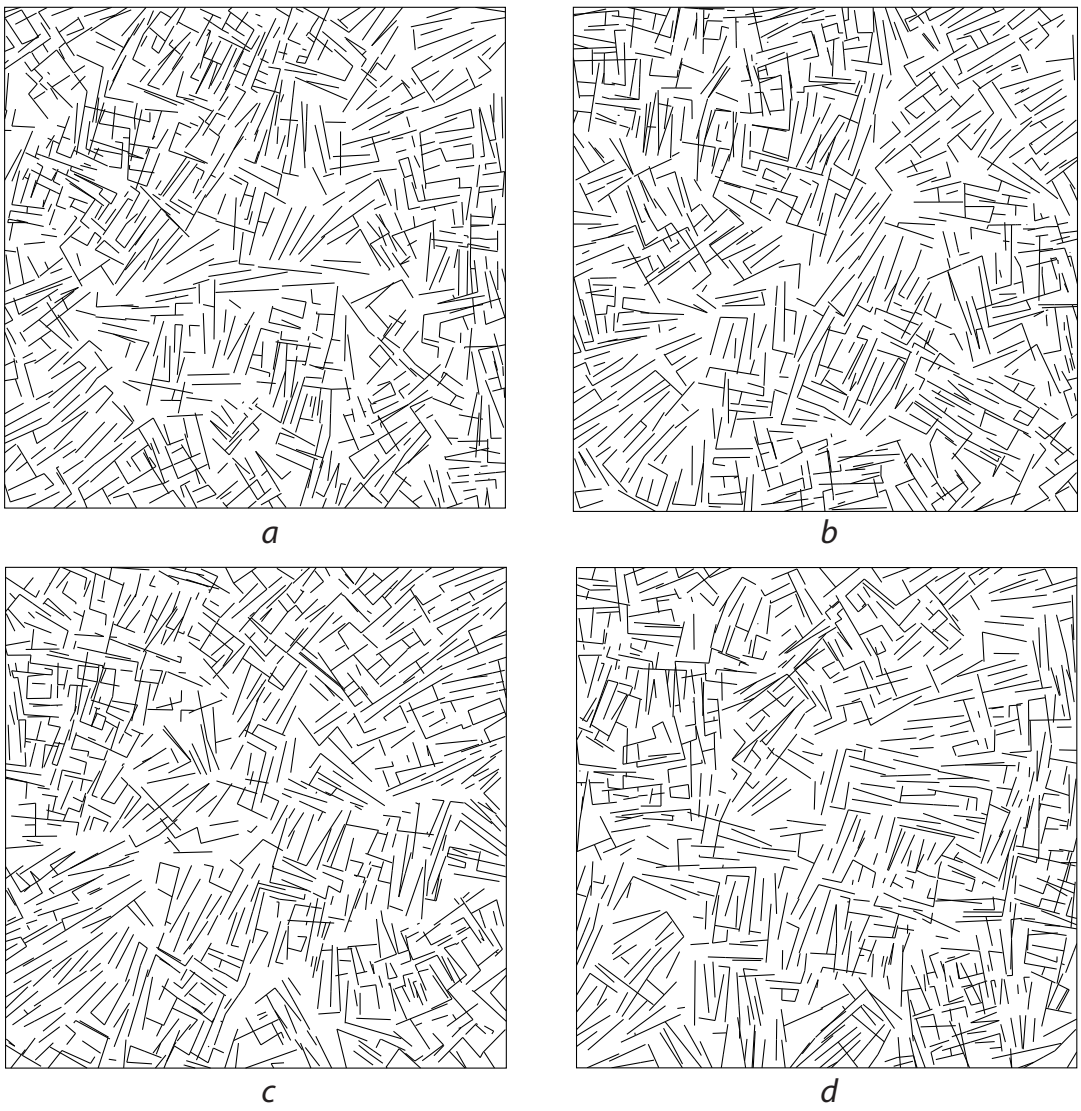


Figure 3.42: Examples, model am2-2: different alignment weight functions. (a) W1, (b) W2, (c) W3, (d) W4, feature map is depicted in Figure 3.41-b.

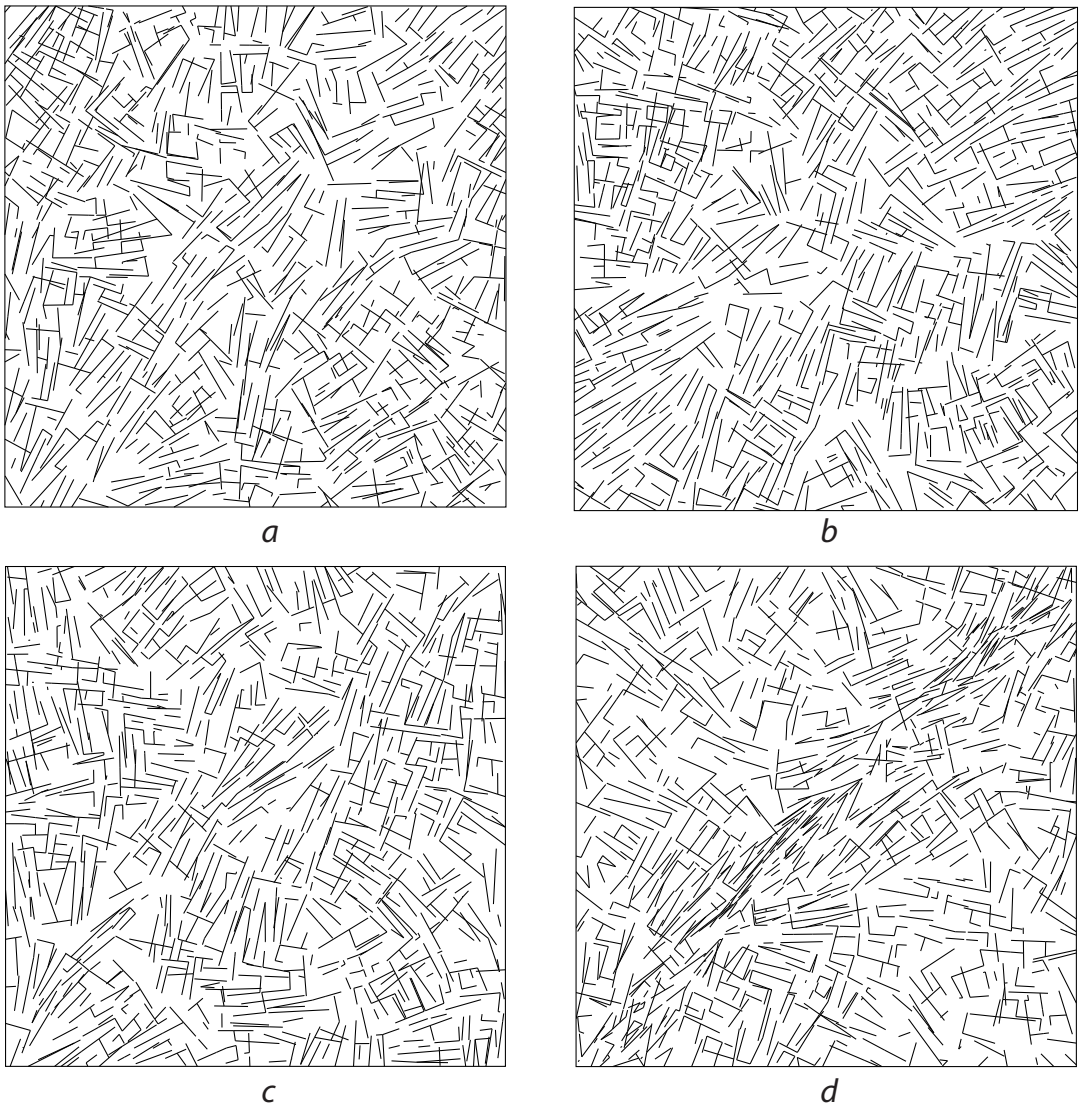


Figure 3.43: Applying different soft constraint models with the “diagonal” feature: (a) asm2-2, W1, (b) asm2-2, W2, (c) asmL(3/4), W1, and (d) asm1, W1. The feature maps is depicted in Figure 3.41-b.

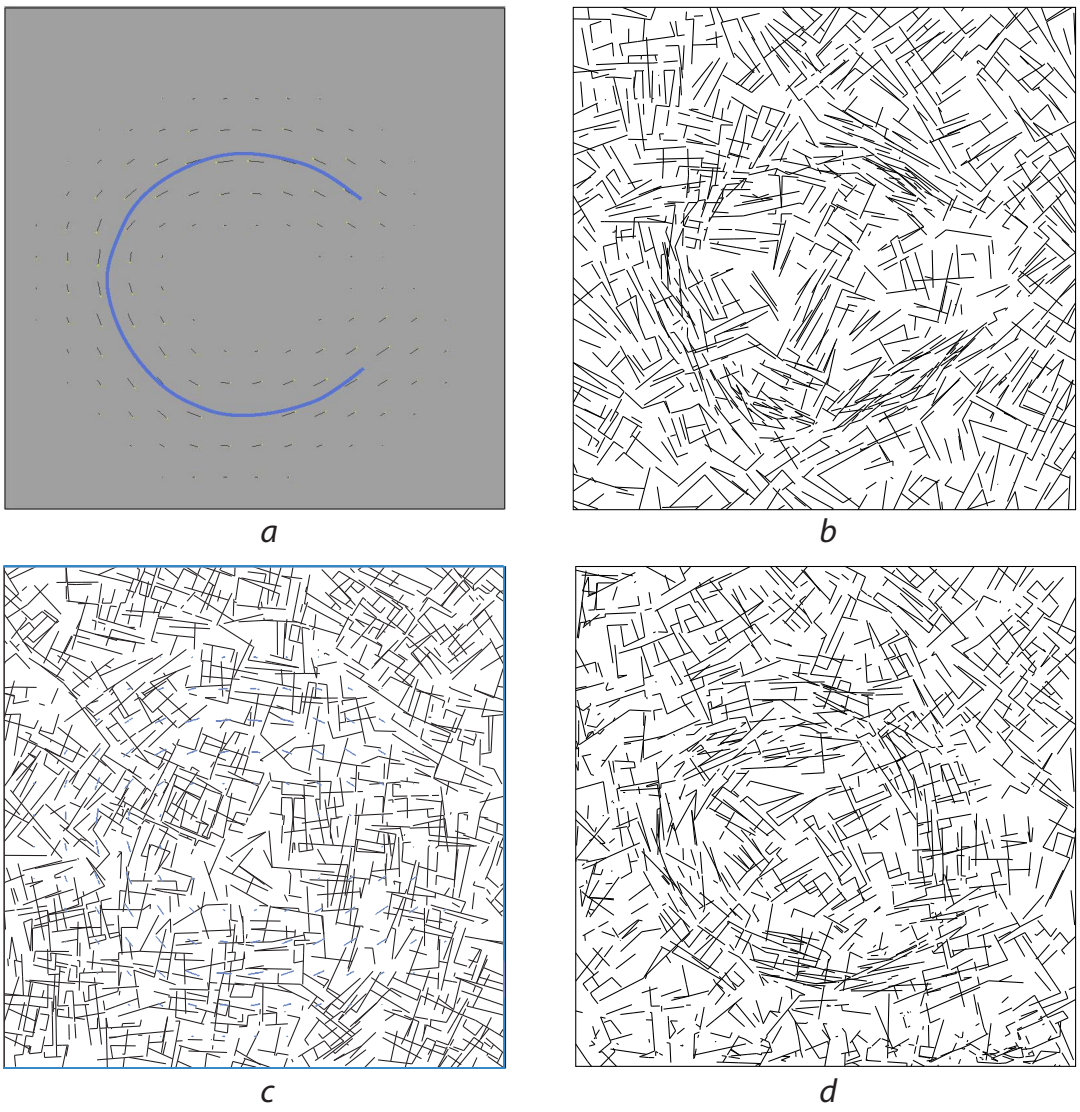


Figure 3.44: Applying soft constraint models to a feature which follows a letter “C” (a), the domain size of the feature distance map is 128pts. (b) asm1, W1, $d_{\max} = 20$pts (c) asmL(3/4), W1, $d_{\max} = 30$pts (d) asm1, W1, $d_{\max} = 30$pts.

bre systems can perfectly serve in generating textures with coherent and non-periodic patterns as a basis for forming curvilinear features which define such patterns. We have successfully adapted the Gibbs point interaction potentials to represent and model interactions for linear fibre processes, which consider line segments as individual fibres. Our linear fibre interaction model is intuitive and convenient to use, which is not the case for existing nonparametric sample-based synthesis techniques. We run a variant of the Monte Carlo Metropolis-Hastings algorithm to generate a random collection of linear fibres distributed according to a given potential-based interaction model. We have implemented a variant of the simulated annealing algorithm to include the local and global constraints to our linear fibre model. Such constraints support a control on the fibre density, the level of fibre connectivity, and the degree of alignment of the linear fibre systems with a user-defined orientation field.

The linear fibre model have shown its best performance on random fibre systems with a moderate level of connectivity between the individual fibres. We have added a post-processing algorithm which grows collections of coherent and smoothly connected curves aligned with a generated system of linear fibres to address the cases where maintaining a connection of most of the fibres is the main requirement. This algorithm is presented in the next chapter.

Chapter 4

A method of generating networks of curves aligned with random systems of linear fibres

One possible application for the random systems of linear fibres (or, fibres of a more general form) is to generate networks of curves which are aligned with a given fibre system. This is especially useful when simulating a fibre system under a given fibre interaction model does not produce a set of nicely connected, locally coherent, and slowly changing network of curves: an additional step is needed to either filter the resulting fibres or, as we propose, to generate a new system of curves coherent with the resulting fibre system. In this section we describe a general method to generate network of curves coherent with a given system of random linear fibres and demonstrate its performance on cross-fibre patterns. We describe a general framework of generating overlapping families of coherent streamlines aligned with a multi-orientation vector field in Section 4.1, and illustrate its performance and its problems for the case of cross-orientation vector fields in Section 4.2. The method of sampling the context-

dependent cross-orientation vector fields and its variations, presented in Section 4.3, solve most of the problems of the original streamline generation algorithm enabling a user to generate networks of coherent curves with a desirable degree of randomness. The following is an outline of the method:

1. *Prepare cross-orientation vector field (COVF).* Based on a given linear fibre system (generated by one of the method described in Chapter 3), a vector field tangent at every point to the system is created. Usually, the tangent vector fields are mostly discontinuous, so that applying standard streamline placement algorithms will results in undesirable sharp turns of streamlines (see Figure 4.7-a bottom-right corner, -c and -d). To resolve it we filter the tangent vector field to form a locally coherent cross-orientation vector field (*COVF*) consisting of unordered pairs of orthogonal vectors. We make sure that a vector from a COVF pair is aligned with the tangent field as much as possible. More detailed definition and the optimization details are described in Section 4.1 with examples of optimized COVFs in Figures 4.4 and 4.5;

2. *Run our CD-wCD algorithm for streamline placement.* A modified two-pass version of Mebarki et al. streamline placement algorithm [86], called CD-wCD, generates two networks of coherent curves aligned with a provided COVF. The algorithm progressively fetches directions from COVF, choosing such COVF orientation which is better aligned with current streamline — we call such process as context-dependent (*CD*) direction sampling from COVF. During the first pass — CD-pass (Section 4.3) — while placing streamlines we generate two weight fields which assign confidence levels to COVF components depending on degree of alignment of such components with just generated streamlines. During the second pass — wCD-pass (Section 4.3) — we apply

weighted context-dependent (*wCD*) direction sampling from COVF taking into account assigned confidence levels to COVF pairs. The final network of curves is formed by combining the resulting two set of streamlines. An example of such network is shown in Figure 4.11.

4.1 Multi-orientation vector fields

An algorithm of building a vector field V^{Φ}, tangent to a given fibre system Φ, is straightforward: a tangent vector at a given point is defined by a direction of the closest to the point fibre.

As regularity is an essential attribute of our model of linear fibres, the fibre systems produced by our algorithm are bound to consist of several groups of locally coherent fibres overlapped within a common region. As the result, a tangent vector field built from a typical fibre system will form a map discontinuous almost everywhere. Any streamline placement algorithm will fail to produce a reasonable set of smooth streamlines tangent to such vector field. A different way of applying the fibre-induced tangent vector field is needed.

Textures which consist of several overlapping distinct texture patterns are not new in Computer Graphics and Computer Vision. Many natural multi-directional textures were carefully investigated in [21, 65]. The phenomenon of texture laciness, when several overlapping textures can be perceived as one new texture or as distinctive textures, was analyzed in [146]. Algorithms of reconstruction of multiple texture flows sharing the same domain via relaxation labeling technique was described in [16].

One possibility to handle multi-directional vector fields is to decompose them into a set of continuous vector fields which match the fragments of the original field throughout the domain. Then, different sets of streamlines are placed independently

on the same domain — each set is aligned with one of the resulting continuous vector fields. Superposition of the these sets of streamlines will form a final network of random curves. However, overwhelming majority of fibre-induced tangent fields cannot be decomposed easily into smooth vector fields, if sometimes not at all. Figure 4.1 shows an unsuccessful attempt to classify the fibres (the corresponding fibre system is depicted in Figure 3.28-a) into two classes with smoothly changing dominant orientation. Such classification is impossible around the location where two yellow curvilinear arrows approach each other: clear continuity of the "fibre flow" at the arrows' tails is broken near the arrows' tips.

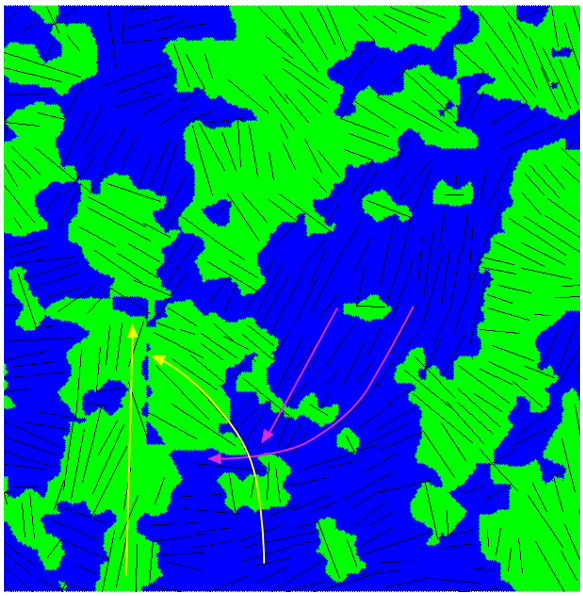


Figure 4.1: Examples of "fibre flow" singularities. Two smooth fibre flows (green versus blue background) were classified from a fibre system with interaction model which favors cross-patterns. Singularities are clearly noticeable at intersections of yellow and pink arrows.

An alternative is to form a *multi-orientation vector field* (MOVF), which assigns

multiple directions at every point. MOVF should be somewhat consistent with its underlying tangent vector field and should smoothly vary over the domain. To put it in more detail, at every point at least one of the the MOVFs directions should closely match the corresponding tangent vector at this point, and, for relatively close points, each MOVF's direction at one point should have a close counterpart at the other point. Figure 4.2 shows examples of consistent and inconsistent MOVFs: both MOVFs are smoothly varying, however, the MOVF on the right (yellow crosses) lacks of consistency with underlying tangent vector field (pink line segments). In the next section we describe how to generate streamlines from MOVF.

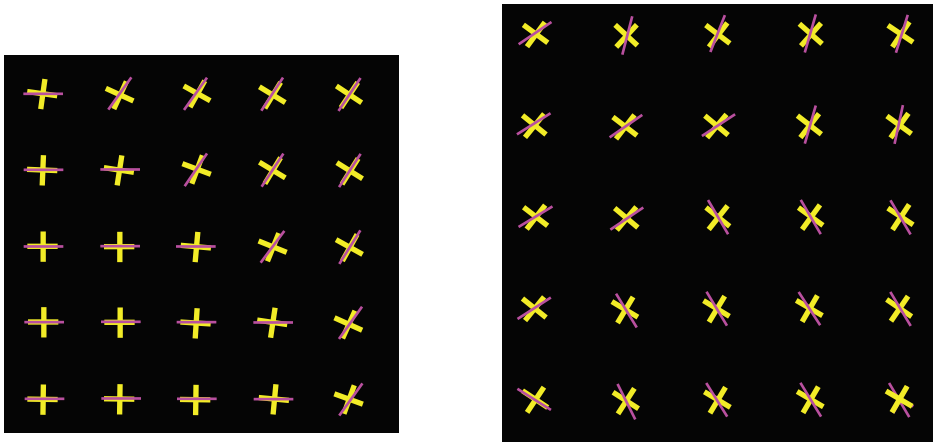


Figure 4.2: Examples of MOVFs, consistent (on the left) and inconsistent (on the right) with a given vector field. MOVF cross-orientations are represented by yellow crosses, vector fields samples — by pink line segments.

A concept of MOVF was successfully applied in [63] to construct the so called cross-hatching patterns over smooth surfaces. The authors generated cross fields with unordered pairs of orthogonal directions as their values by optimizing the directions to smoothly vary along the geodesics. We adapt this method to generate

cross-orientation vector fields (COVF) from linear fibre systems. In this work we demonstrate how to apply COVFs to generate random networks of curves with two dominant directions. For better performance of our COVF-based algorithm, the interaction model of underlying linear fibre system is expected to favor cross configurations of fibres (examples of such fibre systems are illustrated in Figures 3.29-a and -b, 3.37, and 3.39).

For a given fibre-induced tangent vector field $\mathsf{V}^{\Phi} : \mathbb{R}^2 \to \mathbb{E}$, defined within a simulation domain X, we optimize a cross-orientation vector field $V^{+} : \mathbb{R}^2 \to \mathbb{E} \circ \mathbb{E}$, to be consistent with V^{Φ} and smooth over X. COVF V^{+} assigns an unordered pair of vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ at every given point $p \in X$, such that $\boldsymbol{v} \cdot \boldsymbol{w} = 0$. To satisfy COVF's requirements, the following energy functional should be minimized over all the points form a regular grid $\Omega_X = \{p_{ij} \in X : i = 1,2,\ldots,N_i,\ j = 1,2,\ldots,N_j\}$

$$E_{ij}^{V} = \min_{k=0,1} \left\| \boldsymbol{v}_{ij}^{k} - \boldsymbol{t}_{ij} \right\| + \min_{k',k''=0,1} \left\| \boldsymbol{v}_{ij}^{k'} - \boldsymbol{v}_{ij+1}^{k''} \right\| + \min_{k',k''=0,1} \left\| \boldsymbol{v}_{ij}^{k'} - \boldsymbol{v}_{i+1j}^{k''} \right\|,$$

where the optimization runs on the grid values $\boldsymbol{v}_{ij}^{k} \equiv V_k^{+}(p_{ij})$ and $\boldsymbol{t}_{ij} \equiv \mathsf{V}^{\Phi}(p_{ij})$, and, for the sake of simplicity, we assume that there are two vector fields V_1^{+} and V_2^{+} which impose an arbitrary order for pairs of vectors in V^{+}: $V^{+} \equiv \{V_1^{+}, V_2^{+}\}$. The first term of E_{ij}^{V} relates to COVF's consistency with V^{Φ}, and the second and the third terms describe continuity of COVF along grid's east and north directions, respectively. The energy functional E_{ij}^{V} can be reformulated to a simpler form E_{ij}^{Θ} depending on orientations, rather than vectors:

$$\begin{aligned} E_{ij}^{\Theta} = &\min_{k} \sqrt{2 - 2\cos\{(\theta_{ij} - \alpha_{ij}) + k\pi/2\}} \\ &+ \min_{k} \sqrt{2 - 2\cos\{(\theta_{ij} - \theta_{ij+1}) + k\pi/2\}} \\ &+ \min_{k} \sqrt{2 - 2\cos\{(\theta_{ij} - \theta_{i+1j}) + k\pi/2\}}, \end{aligned}$$

where $\theta_{ij} = \arccos\left|\boldsymbol{v}_{ij}^{1} \cdot \boldsymbol{e}_1\right|$ and $\alpha_{ij} = \arccos\left|\boldsymbol{t}_{ij} \cdot \boldsymbol{e}_1\right|$. According to the derivation in [63], minimization of E_{ij}^{Θ} can be replaced by minimization of the following functional

$$E_{ij} = -\cos 4(\theta_{ij} - \alpha_{ij}) \; -\cos 4(\theta_{ij} - \theta_{ij+1}) \; -\cos 4(\theta_{ij} - \theta_{i+1j}) \, .$$

We apply a variant of the subspace trust region method with interior-reflective Newton method (`fminunc` in Matlab with Large scale optimization [56]) to run nonlinear unconstrained optimization on the following energy:

$$\begin{aligned} E &= \sum_{ij} E_{ij} \\ &= -\sum_{ij} \Big\{ \cos 4(\theta_{ij} - \alpha_{ij}) \; + \cos 4(\theta_{ij} - \theta_{ij+1}) + \cos 4(\theta_{ij} - \theta_{i+1j}) \Big\} \, , \end{aligned}$$

by providing a vector of gradients $\boldsymbol{g} = \left(\frac{\partial E}{\partial \theta_{ij}}\right)_{(i,j)}$ with the following $N_i N_j$ components

$$\begin{aligned} g_{(i,j)} = \frac{\partial E}{\partial \theta_{ij}} = \; & 4\sin 4(\theta_{ij} - \alpha_{ij}) \\ & + 4\sin 4(\theta_{ij} - \theta_{ij+1}) \; + 4\sin 4(\theta_{ij} - \theta_{i+1j}) \\ & - 4\sin 4(\theta_{ij-1} - \theta_{ij}) \; - 4\sin 4(\theta_{i-1j} - \theta_{ij}) \, , \end{aligned}$$

and a sparse $N_i N_j \times N_i N_j$ Hessian matrix $H = \left[\frac{\partial^2 E}{\partial \theta_{ij} \partial \theta_{i'j'}}\right]_{(i,j),(i',j')}$ with the following non-trivial components

$$\begin{aligned} H_{(i,j),(i,j)} = \frac{\partial^2 E}{\partial \theta_{ij}^2} = \; & 16\cos 4(\theta_{ij} - \alpha_{ij}) \\ & + 16\cos 4(\theta_{ij} - \theta_{ij+1}) \; + 16\cos 4(\theta_{ij} - \theta_{i+1j}) \\ & + 16\cos 4(\theta_{ij-1} - \theta_{ij}) \; + 16\cos 4(\theta_{i-1j} - \theta_{ij}) \, , \end{aligned}$$

$$H_{(i,j),(i+1,j)} = \frac{\partial^2 E}{\partial\theta_{ij}\,\partial\theta_{i+1j}} = -16\cos 4(\theta_{ij} - \theta_{i+1j}),$$
$$H_{(i,j),(i,j+1)} = \frac{\partial^2 E}{\partial\theta_{ij}\,\partial\theta_{ij+1}} = -16\cos 4(\theta_{ij} - \theta_{ij+1}),$$
$$H_{(i,j),(i,j-1)} = \frac{\partial^2 E}{\partial\theta_{ij}\,\partial\theta_{ij-1}} = -16\cos 4(\theta_{ij-1} - \theta_{ij}),$$
$$H_{(i,j),(i-1,j)} = \frac{\partial^2 E}{\partial\theta_{ij}\,\partial\theta_{i-1j}} = -16\cos 4(\theta_{i-1j} - \theta_{ij}),$$

where a two-dimensional index (i, j) is linearized by the expression $iN_j + j$. The optimization, started with the initial orientation field $\{\theta_{ij} = \alpha_{ij}\}$, turned out to converge very rapidly for the vector fields we were dealing with. The resulting optimal COVF V^+_{opt} is given by its values on the grid as follows $\hat{V}^+_{\text{opt}}[i,j] = \left\{R(\theta^{\text{opt}}_{ij})\boldsymbol{e}_1, R(\theta^{\text{opt}}_{ij} + \pi/2)\boldsymbol{e}_1\right\}$, where $R(\theta)$ is a rotation operator.

Figures 4.4 and 4.5 illustrate the results of COVFs optimization on 256×256 grid of the vector fields formed by tangents of linear fibre systems shown in Figure 4.3-a and -b, respectively. Fragments of the original vector fields are illustrated in the figures in the left columns, with the tangents colored in dark red. Optimized COVFs are depicted in the right columns colored in dark red and dark cyan. In both examples COVF optimization produced satisfactory results in smoothing the initial COVFs while keeping them consistent with the original fibre-induced tangent vector fields.

To assign an order for vector pairs from $V^+_{\text{opt}}(p)$ we run a simple greedy type algorithm which makes selections progressively starting from a given set of seed points within the domain. Formally, we decompose V^+_{opt} into $V^{2+}_{\text{opt}} : \mathbb{R}^2 \to \mathbb{E}^2$, such that for $V^{2+}_{\text{opt}} \equiv (V_{\text{opt},1}, V_{\text{opt},2})$ the cross-orientation vector field V^+_{opt} is given by $V^+_{\text{opt}} = \left\{V_{\text{opt},1}, V_{\text{opt},2}\right\}$. We start by assigning seed values for a number of grid locations $IJ^s = \left\{(i^s_0, j^s_0), \ldots, (i^s_M, j^s_M)\right\}$, $\hat{V}_{\text{opt},1}[i^s_m, j^s_m] = \boldsymbol{v}_m$ and $\hat{V}_{\text{opt},2}[i^s_m, j^s_m] = \boldsymbol{w}_m$

Figure 4.3: Linear fibre systems $\Phi_{(a)}$ and $\Phi_{(b)}$ used for creating fibre-induced tangent vector fields, $\mathsf{V}^{\Phi_{(a)}}$ and $\mathsf{V}^{\Phi_{(b)}}$, respectively, for examples which follow.

when $\hat{V}^{+}_{\text{opt}}[i^s_m, j^s_m] = \{\boldsymbol{v}_m, \boldsymbol{w}_m\}$, while leaving vector pairs at other grid locations empty, $\hat{V}_{\text{opt},1}[i,j] = \hat{V}_{\text{opt},2}[i,j] = \emptyset$, $\forall (i,j) \notin IJ^s$; and proceed by using a simple coherence criterion to classify a vector from a current cross-orientation vector pair $\hat{V}^{+}_{\text{opt}}[i,j] = \{\boldsymbol{v}, \boldsymbol{w}\}$: $\boldsymbol{v}$ will be included to $\hat{V}_{\text{opt},k}$, where $k = 1 \vee 2$, if

$$\sum_{i',j'\text{- neigh. of } i,j} \rho\left(\boldsymbol{v}, \hat{V}_{\text{opt},k}[i',j']\right) \;\geq\; \sum_{i',j'\text{- neigh. of } i,j} \rho\left(\boldsymbol{v}, \hat{V}_{\text{opt},3-k}[i',j']\right), \tag{4.1}$$

where within the summation only indices i' and j' of already assigned elements of $\hat{V}_{\text{opt},1}$ and $\hat{V}_{\text{opt},2}$ are considered, and ρ is a coherence measure between the vectors, e.g., $\rho(\boldsymbol{v}, \boldsymbol{w}) = |\boldsymbol{v} \cdot \boldsymbol{w}|$; $\boldsymbol{w}$ will be included to the opposite vector field, $\hat{V}_{\text{opt},3-k}$.

This simple selection algorithm can decompose a COVF into two smooth vector fields when the underlying tangent vector field contains up to two dominant orientations within its domain (this is supported by examples in Figure 4.4-b and -d, where the resulting vector fields $\hat{V}_{\text{opt},1}$ and $\hat{V}_{\text{opt},1}$, colored in dark red and dark cyan, respectively, are smooth). However, the algorithm fails to produce smooth vector fields

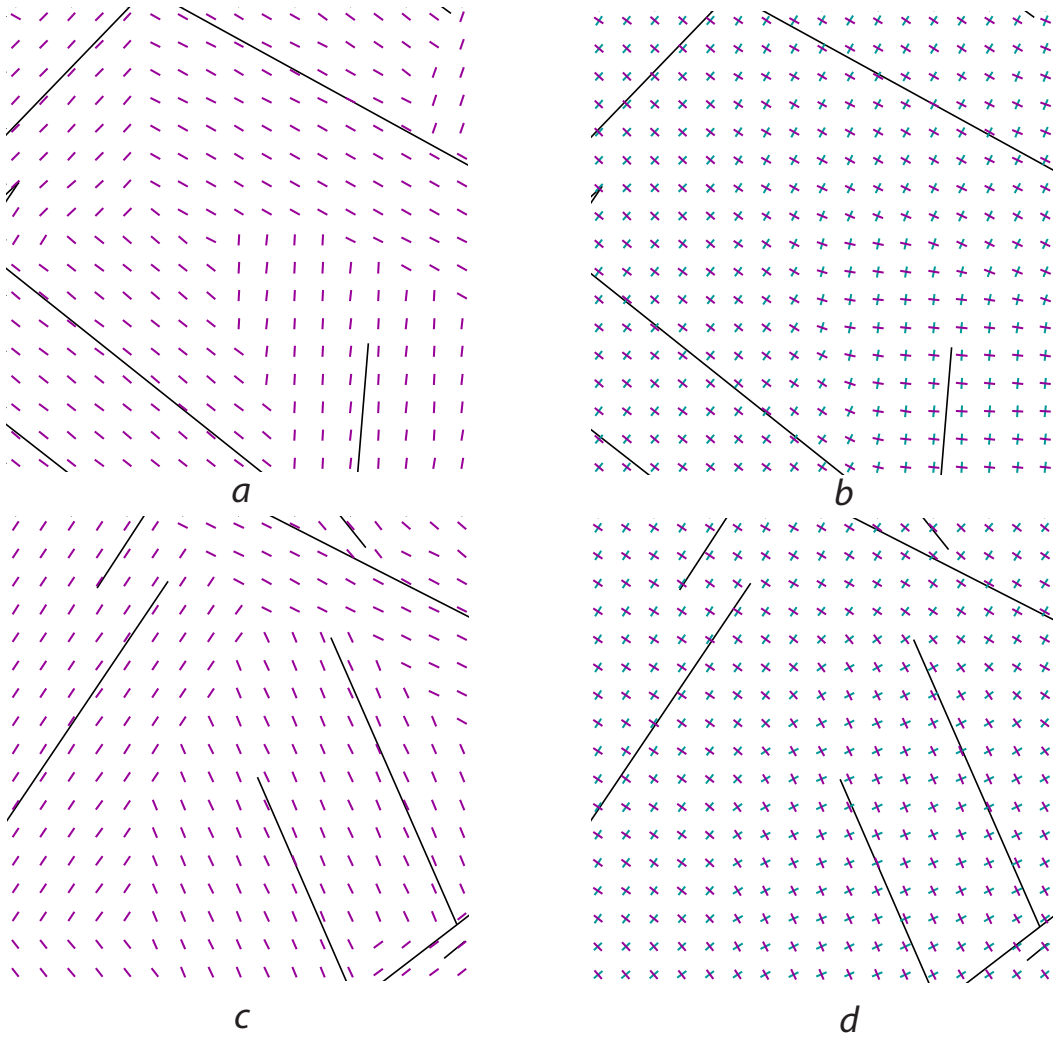


Figure 4.4: Fragments of optimal COVF (right column) against original fibre-induced tangent vector field $V^{\Phi_{(a)}}$ illustrated in Figure 4.3-a (the locations of the fragments are shown by red rectangles). COVF's grid size is 256×256.

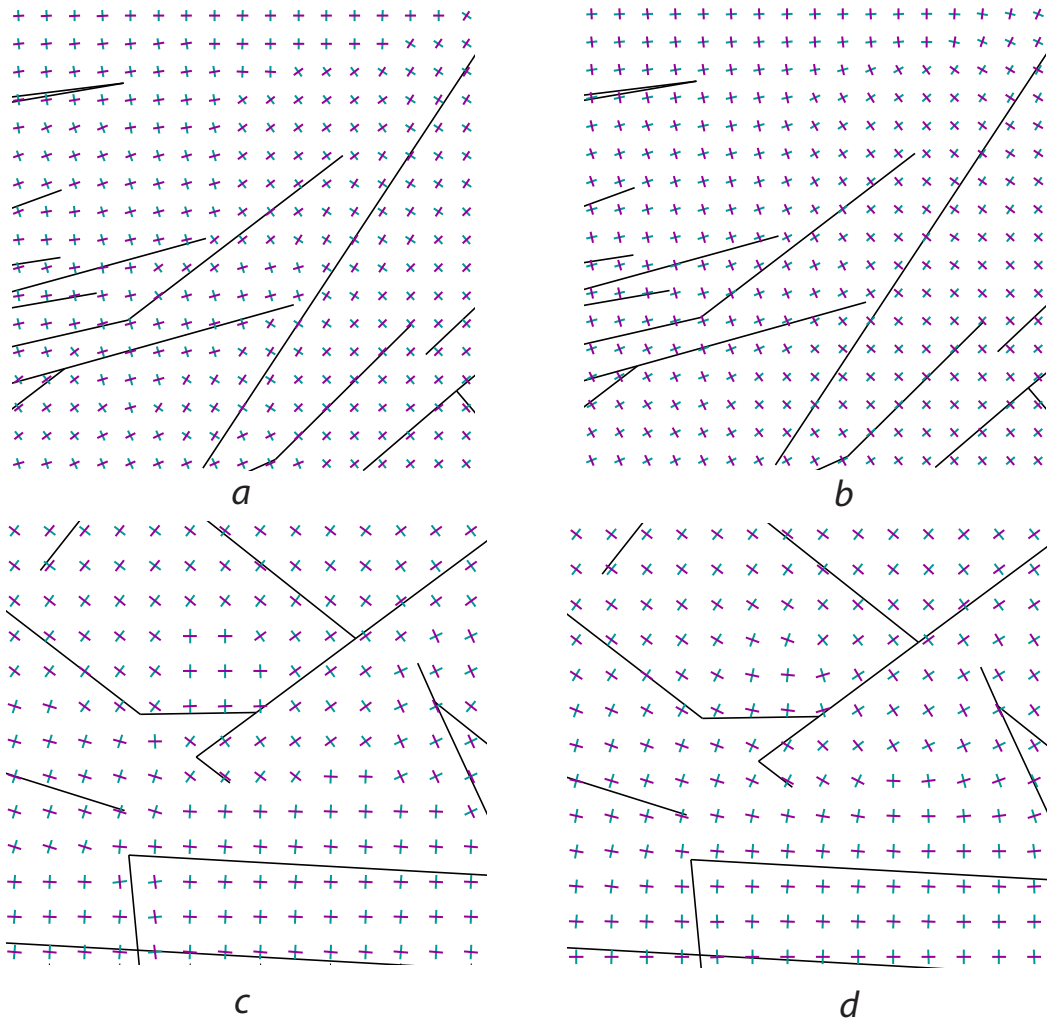


Figure 4.5: Examples of COVF optimization within two fragments of the fibre-induced tangent vector field $V^{\Phi(b)}$ illustrated in Figure 4.3-b (the locations of the fragments are shown by red rectangles). Left: fragments of initial COVF formed from the tangent vector field $V^{\Phi(b)}$. Right: corresponding fragments of optimized COVF. Due to singularities in the original tangent vector field (c), optimal COVF on (d) can not be decomposed into two smooth fields (in the center of the Figure).COVF's grid size is 256×256.

at every location when more than three dominant orientations are presented in the source linear fibre system as shown in Figures 4.5-b and -d in the top right corner and in the center, respectively. MOVFs with more than two overlapping vector fields are required in this case.

4.2 Streamlines aligned with cross-orientation vector fields

The algorithm of Mebarki et al. [86] progressively integrates streamlines — polylines, everywhere tangent to a given vector or orientation field, V — starting each new streamline from a seed point selected from the center of the current biggest empty "cavity." Here is one step during placement of n-th streamline

$$s_n(t+\Delta t) = s_n(t) + \int_t^{t+\Delta t} V(s(\tau))d\tau \equiv s_n(t) + \boldsymbol{v}(t,\Delta t)\,,$$

$$s_n(0) = \text{BigCavSearch}\Big(\{s_1, s_2, \ldots, s_{n-1}\}\Big)\,.$$

For a given pair of vector fields V_1 and V_2, the algorithm runs twice to generate streamline sets $S_1 = \{s_n^1,\ n = 1,2,\ldots,M_1\}$ and $S_2 = \{s_n^2,\ n = 1,2,\ldots,M_2\}$, respectively. The union of S_1 and S_2 gives a final network of curves $S = S_1 \cup S_2$.

As we already noticed before, COVFs which have "singularities" cannot be decomposed in two smooth fields. COVF singularities occur when more than two locally coherent "flows" with different dominant orientations meet at a point. To find singularity locations within a given cross-orientation vector field V^+, one should look for the locations around which any local traversal (ccw or cw) results in the break of continuity (an example is shown in Figure 4.6). One possible scenario of using COVFs with singularities is to perform an assignment algorithm, described in

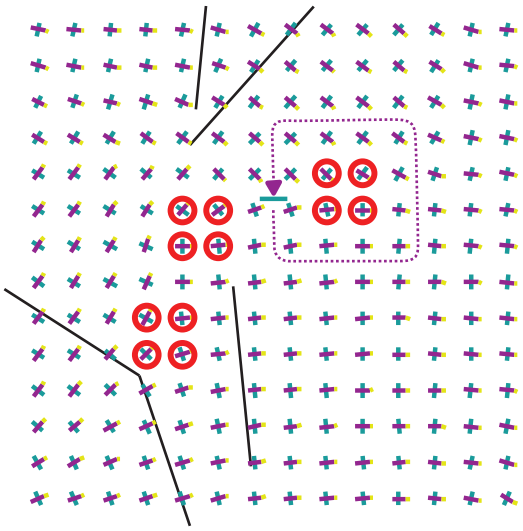


Figure 4.6: A fragment of a COVF with discontinuities: a traversal around the COVF samples, circled in red, always breaks continuity.

the previous section, which produces two vector fields $V_{\text{opt},1}$ and $V_{\text{opt},2}$ — smooth everywhere but in relatively small number of singularity locations — and to pass them to the streamline algorithm described above to generate the network of curves S. The coherence criterion given by equation 4.1 should be slightly modified, as it allows close samples of collinear vectors to be oriented in opposite directions which can abrupt iterations of a streamline placement algorithm. From a pair $\hat{V}^{+}_{\text{opt}}[i,j] = \{\boldsymbol{v},\boldsymbol{w}\}$, a vector with the largest coherence value $c(\rho;\boldsymbol{v},i,j,1)$ w.r.t. the first vector field $\hat{V}_{\text{opt},1}$ is chosen, $c(\rho;\boldsymbol{v},i,j,k) = \sum_{i',j'\text{- neigh. of } i,j} \rho\left(\boldsymbol{v},\hat{V}_{\text{opt},k}[i',j']\right)$. Suppose that it is $\boldsymbol{v}$, then, either $\boldsymbol{v}$ or $-\boldsymbol{v}$ is added to $\hat{V}_{\text{opt},1}$ depending on which one has the largest value $c(\rho_1;\boldsymbol{v},i,j,1)$, where this time a coherence measure ρ_1 takes into account the orientation: $\rho_1(\boldsymbol{v},\boldsymbol{w}) = 1+\boldsymbol{v}\cdot\boldsymbol{w}$. The same procedure is performed for $\boldsymbol{w}$ and $-\boldsymbol{w}$ against $V_{\text{opt},2}$.

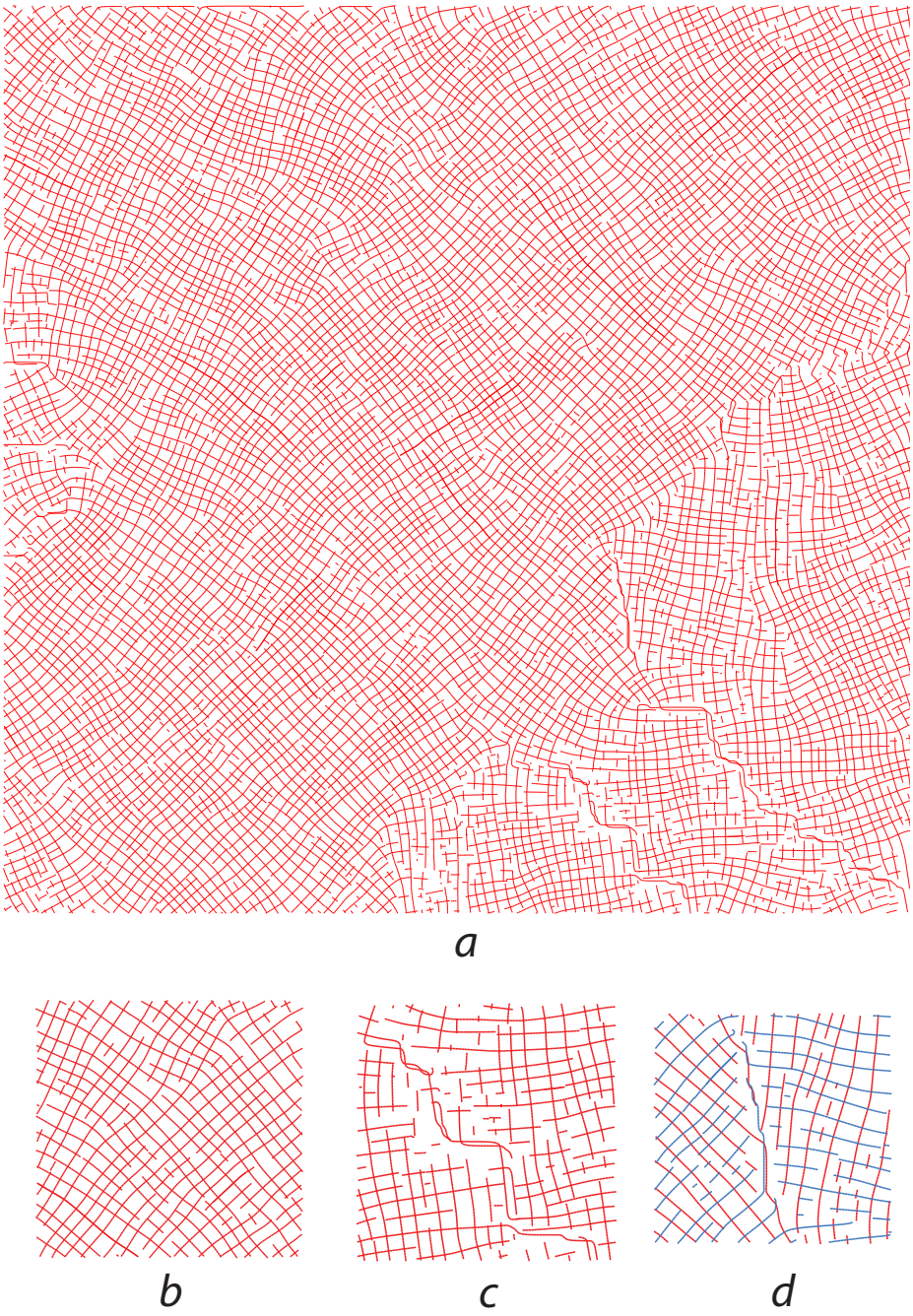


Figure 4.7: Example of curvilinear networks generated by the original streamline algorithm [86] but with flipping the vector samples when necessary to avoid sudden stops.

Figure 4.7 shows an example of a network of curves generated by applying the algorithm described above to a tangent vector field built around fibre system illustrated in Figure 4.3-a. It works well in the prevailing number of areas in the simulation domain, as shown, e.g., in Figure 4.7 (b). However, due to singularities in COVF, generating non-smooth streamlines in several places is inevitable, as shown, e.g., in Figure 4.7 (c,d). An ability to switch between the vector fields during streamline placement can partially resolve the problem of COVF singularities.

4.3 Streamlines based on context-dependent vector fields

In this section we describe several streamline placement algorithms adapted to operate on a cross-orientation vector field (COVF). An advantage of using COVF instead of pair of orthogonal vector fields is that COVF allows to resolve the problem of encountering singularities in the cross field during streamline placement, which occurs in a prevailing number of natural multi-orientation vector fields. As COVF has two directions at every point, one can choose the most appropriate direction out of the two during a current streamline integration step to make sure that the resulting streamline is smooth. This is a central idea for our context-dependent direction sampling algorithm, based on which we describe our COVF-based streamline placement algorithms.

CD-Algorithm. We start with an algorithm of streamline placement with *context-dependent direction sampling* which performs context-dependent bilinear interpolation on a given grid-valued COVF, $\hat{V}^+ = V^+|_{\Omega_X}$. We still assume that $\hat{V}^+$ has been preliminary decomposed into two vector fields $V^{2+} = (V_1, V_2)$, so that $\bar{V}^+_{ij} \equiv \{V_1[i,j], V_2[i,j]\}$. However, this decomposition is necessary only for providing unambiguous seed directions: V_1 is used as seed vector field to generate the first set of

streamlines S_1, and the same is for V_2 to generate S_2. Subsequent integration directions are drawn from COVF. Suppose that n-th streamline s_n^k of k-th set of streamlines S_k, $k \in \{1,2\}$, is being placed, and $m+1$ points of s_n^k has been already placed, $s_n^k = (p_0^n, p_1^n, \ldots, p_m^n)$. The first seed direction $\boldsymbol{v}_0^n$ has been taken from V_k as follows

$$\boldsymbol{v}_0^n = \tau\big(\lambda_0, \mu_0; V_k[i,j], V_k[i,j+1], V_k[i+1,j], V_k[i+1,j+1]\big)\,,$$

where τ is an operator of bilinear interpolation, and λ_0 and μ_0 are the texel coordinates of the n-th streamline's seed point p_0^n which happened to be located within the interior of texel T_{ij}

$$\tau\big(\lambda, \mu; v_{00}, v_{01}, v_{10}, v_{11}\big) = \lambda \bullet (\mu \bullet (v_{00}, v_{01}), \mu \bullet (v_{10}, v_{11}))\ , \tag{4.2}$$

$$p_0^n = \tau\big(\lambda_0, \mu_0; p_{ij}, p_{ij+1}, p_{i+1j}, p_{i+1j+1}\big)\ . \tag{4.3}$$

To find $(m+2)$-th point of s_n^k, we need to fetch the next direction $\boldsymbol{v}_m^n$ from COVF, V^+, at p_m^n. Suppose that p_m^n is located within texel T_{ij} and its texel coordinates are λ_m and μ_m (as in Equation 4.3). First, we choose those directions from corner cross-orientation vectors $\hat{V}_{ij}^+$, $\hat{V}_{ij+1}^+$, $\hat{V}_{i+1j}^+$, and $\hat{V}_{i+1j+1}$ which are better aligned with the current integration direction $\boldsymbol{v}_{m-1}^n$: $(\boldsymbol{w}_{00}, \boldsymbol{w}_{01}, \boldsymbol{w}_{10}, \boldsymbol{w}_{11})$, such that $\boldsymbol{w}_{IJ} = V_{k'}[i+I, j+J]$, if $\rho(\boldsymbol{v}_{m-1}^n, V_{k'}[i+I, j+J]) \geq \rho(\boldsymbol{v}_{m-1}^n, V_{3-k'}[i+I, j+J])$, $k' \in \{1,2\}$, for all index offset combinations $I = 0,1$ and $J = 0,1$. Second, every chosen vector $\boldsymbol{w}_{IJ}$ is flipped if necessary to be oriented in the same direction as the current vector $\boldsymbol{v}_{m-1}^n$ giving rise to a new quadruple of vectors $(\tilde{\boldsymbol{w}}_{00}, \tilde{\boldsymbol{w}}_{01}, \tilde{\boldsymbol{w}}_{10}, \tilde{\boldsymbol{w}}_{11})$, such that $\tilde{\boldsymbol{w}}_{IJ} = -\boldsymbol{w}_{IJ}$, if $\rho_1(\boldsymbol{v}_{m-1}^n, -\boldsymbol{w}_{IJ}) \geq \rho_1(\boldsymbol{v}_{m-1}^n, \boldsymbol{w}_{IJ})$, and is the same otherwise. And third, the next integration direction $\boldsymbol{v}_m^n$ is just a bilinear interpolation of the resulting vectors as follows

$$\boldsymbol{v}_m^n = \tau\big(\lambda_m, \mu_m; \tilde{\boldsymbol{w}}_{00},\ \tilde{\boldsymbol{w}}_{01}, \tilde{\boldsymbol{w}}_{10},\ \tilde{\boldsymbol{w}}_{11}\big)\,.$$

The algorithm of context-dependent direction sampling is schematically shown in

Figure 4.8. The Figure illustrate a situation when COVF presumably has a singularity within a texel T_{ij} which resulted in decomposition of $\hat{V}^+$ into two, locally discontinuous, vector fields V_1 (brown colored) and V_2 (green colored).

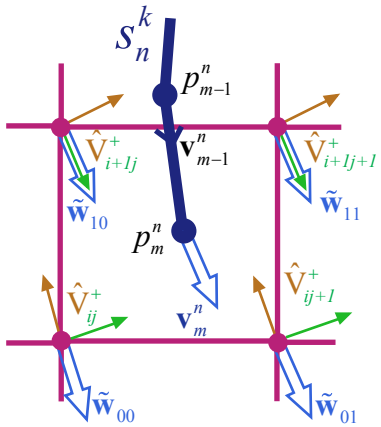


Figure 4.8: Context-dependent direction sampling from COVF $\hat{V}^+$ (pairs of brown and green vectors at the texel corners) during integration step at point p_m^n of current streamline s_n^k (dark blue). New direction $\boldsymbol{v}_m^n$ is interpolated from the adjusted corner vectors $\tilde{\boldsymbol{w}}_{IJ}$ within the texel T_{ij}.

Figure 4.9 (a) shows the result of a run of our CD-algorithm on COVF used for generating Figure 4.7. It did very good job to avoid abrupt turns of streamlines [compare Figure 4.9 (d) and Figure 4.7 (d)]. However, switching between the vector fields during streamline placement process can result in noticeable artifacts caused by merging streamlines from different sets (Figure 4.9 (a), central part). It happens when two streamlines from different sets get the integration directions from the same vector field at the same location. Also, because of vector field switching, one or both resulting streamline sets can turn out to be discontinuous at certain locations. Discontinuity can come from near-orthogonal intersection (for the case of cross-orientation

vector fields) of two streamlines of the same set, seeded from distant locations, which start by using the initial directions from the same vector field but approach each other from orthogonal directions resulted from sampling different vector fields (Figure 4.9 (c) shows a fragment of streamlines from S_1). None of these could happen when a given COVF is preliminary decomposed into two different cross vector fields as streamline placement algorithm uses only one resulting vector field at a time for each streamline set (as was demonstrated in the previous section). For a possible solution, we propose assigning weights to COVF vectors which describe the *confidence level* of each vector within cross-orientation vector pair to be appropriate when used during streamline placement run for a particular streamline set. If one makes sure that each vector in a COVF pair is not simultaneously assigned a large confidence level for both streamline sets and aligned vectors from grid-neighboring COVF vector pairs get either both large or both small confidence levels for the same streamline set, it can significantly decrease the likelihood of artifacts mentioned above to occur.

wCD-Algorithm. The confidence levels of COVF vectors is taken into account in context-dependent interpolation on a given grid-valued COVF, $\hat{V}^+$, which forms to a new version of streamline placement algorithm with *weighted context-dependent direction sampling*. In principle, the confidence level is a function of two variables, the location and the direction, $\sigma : \mathbb{R}^2 \times \mathbb{E} \rightarrow [0,1]$, and $\sigma(p,\boldsymbol{v})$ assigns the confidence level of a given direction $\boldsymbol{v}$ at point p, which will be used during a streamline placement run. In this framework there are only two possible directions at every location defined in COVF and these directions do not change during the run. Therefore, $\sigma(p,\cdot)$ is replaced by a pair of confidence level functions $\sigma^1(p)$ and $\sigma^2(p)$ which attribute weights to both vectors from COVF vector pairs, $\sigma^k : \mathbb{R}^2 \rightarrow [0,1]$, $k \in \{1,2\}$. (Comment: an order for vector in V^+ pairs can be chosen by using a technique described in the previous section; so, as before, $\bar{V}^{2+} \equiv V^{2+}|_{\Omega_X} = (V_1, V_2)$

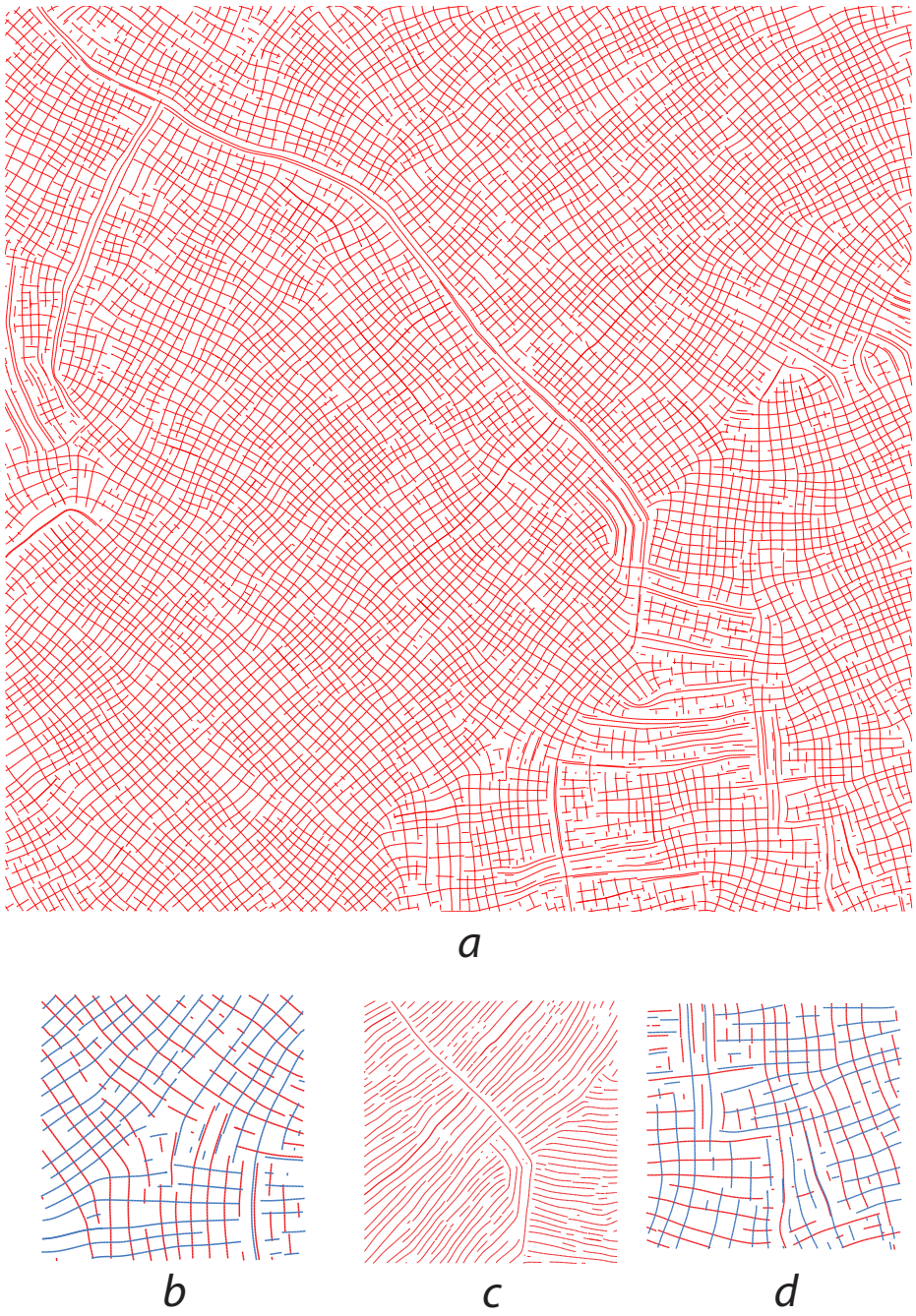


Figure 4.9: (a) Example of curvilinear network generated by the streamline algorithm with context-dependent direction sampling from the same COVF as for Figure 4.7. (b),(c) and (d) are different fragments of the network.

where $\bar{V}^+ \equiv V^+|_{\Omega_X} = \{V_1, V_2\}$).

Suppose, for a moment, that the functions $\sigma^k(\cdot)$ are already provided. We describe now how to calculate the next direction $\boldsymbol{v}_m^n$ to find $(m+2)$-th point of currently placed streamline s_n^k, when $m+1$ points have been already allocated, $s_n^k = (p_0^n, p_1^n, \ldots, p_m^n)$. First, we fetch two quadruples of the texel corner vectors $\bar{v}^k = (\boldsymbol{v}_{00}^k, \boldsymbol{v}_{01}^k, \boldsymbol{v}_{10}^k, \boldsymbol{v}_{11}^k)$ and their corresponding weights $\bar{\sigma}^k = (\sigma_{00}^k, \sigma_{01}^k, \sigma_{10}^k, \sigma_{11}^k)$ from the texel T_{ij} which covers the current point p_m^n, where $\boldsymbol{v}_{IJ}^k = V_k[i+I, j+J]$, $\sigma_{IJ}^k = \sigma^k(p_{i+I,j+J})$, $p_{i+I,j+J}$ is a grid point and a bottom-left corner of $T_{i+I,j+J}$. Second, we choose component-wisely the vectors from $\bar{v}^1$ and $\bar{v}^2$ which are more coherent with a current direction $\boldsymbol{v}_{m-1}^n$ and form $\bar{w}^1 = (\bar{v}_0^{k_0}, \bar{v}_1^{k_1}, \bar{v}_2^{k_2}, \bar{v}_3^{k_3})$ while placing less coherent vectors in $\bar{w}^2 = (\bar{v}_0^{3-k_0}, \bar{v}_1^{3-k_1}, \bar{v}_2^{3-k_2}, \bar{v}_3^{3-k_3})$: for each $t = 0,1,2,3$ the index k_t is chosen to satisfy the inequality $\rho(\boldsymbol{v}_{m-1}^n, \bar{v}_t^{k_t}) \geq \rho(\boldsymbol{v}_{m-1}^n, \bar{v}_t^{3-k_t})$ (here $\bar{v}_{t'}^k$ denotes the t'-the component of $\bar{v}^k$). Third, we calculate the average confidence level of the vectors in every resulting quadruple, $\bar{w}^1$ and $\bar{w}^2$: $\langle \bar{\sigma}^k \rangle = \sum_{t=0}^{3} \bar{\sigma}_t^{k_t}/4$ and $\langle \bar{\sigma}^{3-k} \rangle = \sum_{t=0}^{3} \bar{\sigma}_t^{3-k_t}/4$ for the same set of $\{k_t\}$ defined during formation of $\bar{w}^1$. Then, we proceed with the quadruple of vectors $\bar{w}^\sigma$ which has the largest average confidence value: $\bar{w}^\sigma = \bar{w}^k$ for such a k that $\langle \bar{\sigma}^k \rangle \geq \langle \bar{\sigma}^{3-k} \rangle$. Fourth, we flip the vectors from the quadruple $\bar{w}^\sigma$ to make them oriented in the same direction as $\boldsymbol{v}_{m-1}^n$: $\tilde{w}_t^\sigma = -\bar{w}_t^\sigma$ if $\rho_1(\boldsymbol{v}_{m-1}^n, -\bar{w}_t^\sigma) \geq \rho_1(\boldsymbol{v}_{m-1}^n, \bar{w}_t^\sigma)$, and is the same otherwise, for each $t = 0,1,2,3$. And at last, the next integration direction $\boldsymbol{v}_m^n$ is a bilinear interpolation of the vectors from just calculated quadruple $\tilde{w}^\sigma$

$$\boldsymbol{v}_m^n = \tau\big(\lambda_m, \mu_m; \tilde{w}_0^\sigma,\ \tilde{w}_1^\sigma, \tilde{w}_2^\sigma,\ \tilde{w}_3^\sigma\big)\,.$$

The third step of the weighted context-dependent direction sampling algorithm introduces a crucial difference from the previous algorithm: the more coherent direction can turn out to have smaller confidence level, so that the less coherent direction will

be chosen to father propagate the current streamline. This may eventually result in smoother set of streamlines or avoid merging streamlines from different streamline sets. This scenario is demonstrated in more detail in Figure 4.10.

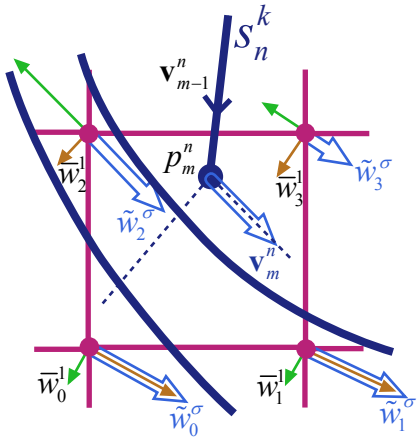


Figure 4.10: Weighted context-dependent direction sampling from a COVF (pairs of brown and green vectors at the texel corners) during integration step at point p_m^n of current streamline s_n^k (dark blue). COVF vector pairs are scaled in accordance with their weights σ_{ij}^k. Aligned with $\boldsymbol{v}_{m-1}^n$ vectors $\bar{w}_t^1$ have smaller value of average conference level, so that $\tilde{\boldsymbol{w}}_t^\sigma$ are chosen instead for interpolation of a new direction $\boldsymbol{v}_m^n$.

Variety of different criteria can be developed to calculate the weights σ_{ij}^k for a given COVF. We propose to associate the degree of local coherency of a vector field with the weights. One possible way to interpret local coherency of a given vector field is through analyzing how well the vector field is aligned with a streamline set if it were generated from COVF which bears the vector field as one of its two components. A degree of alignment of a sample from the vector field is calculated only against streamlines located in the neighborhood of the sample. Therefore, the weight

fields σ^k can substantially vary depending on the size (or, say, diameter) of the test neighborhood. For example, the confidence levels for vectors located nearby and aligned with the streamline which goes across entire streamline set, depicted in Figure 4.9-c, get approximately the same values as the corresponding confidence levels of the orthogonal vectors if the test neighborhood can be stretched only to cover up to two or three streamlines across. However, such confidence levels become significantly smaller when the test neighborhood can encompass, say, more than five or six streamlines. In the latter case, any consequent run of streamline placement algorithm of type wCD will unlikely to place a streamline across the other streamlines at the locations mentioned before, which will result in more smooth streamline set.

CD+wCD-Algorithm. Our version of streamline placement algorithm consists of two stages: (1) generating weights as a measure of local coherency of vector fields, and (2) using the weights as the confidence level in a subsequent streamline placement pass. The first stage of the algorithm consists of performing one or several passes of CD-algorithm, where each pass, while generating two streamline sets with the seed directions taken from the pair of vector fields V_1 and V_2, constituting a given COVF $V^+ = \{V_1, V_2\}$, updates corresponding weight fields σ^1 and σ^2. A possible update strategy is to update all the weights at those locations which are in the neighborhood of a streamline, being placed, by a certain value which depends on the distance to the streamline and a degree of alignment of the vector field at the location with the tangent of the streamline: for each $k = 1,2$, update $\sigma^k(p_{ij}) = U\Big(\sigma^k(p_{ij}), \rho\left(\boldsymbol{t}_{s_n^k}(u^*_{p_{ij}}), V_k(p_{ij})\right), d_{ij}\left(s_n^k(u^*_{p_{ij}})\right)\Big)$ for those $p_{ij} \in \Omega_X$ which satisfy the distance requirement $d_{ij}\left(s_n^k(u^*_{p_{ij}})\right) \equiv \left\| p_{ij} - s_n^k(u^*_{p_{ij}}) \right\| \leq R_{\text{ctxt}}$, where $u^*_{p_{ij}}$ is defined by $u^*_{p_{ij}} = \arg\min_u d_{ij}\left(s_n^k(u)\right)$, $\boldsymbol{t}_s(u)$ is the tangent vector of a curve s at its point $s(u)$, and $R_{\text{ctxt}} > 0$ is the test neighborhood radius — a *context radius*. $U(\tilde{\sigma}, \tilde{\rho}, \tilde{d})$ is any update function of the current weight $\tilde{\sigma}$, the degree of coherency of the direc-

tion with the current streamline $\tilde{\rho}$, and the distance to the streamline $\tilde{d}$. Initially, the weights $\sigma^k(\cdot)$ are zero everywhere. One can run more passes to accumulate more samples of local coherency while changing the context radius R_{ctxt}.

The second stage of the algorithm consists of running wCD-algorithm to generate two final streamline sets taking into account the confidence level weights found during the first stage. For every resulting streamline set $S_{k'}^{CD}$, $k' = 1,2$, two sets of weights were generated: $\sigma^{1,k'}$ and $\sigma^{2,k'}$, one per each vector field $V_{k'}$. wCD-algorithm performs streamline placement for each set $S_{k'}^{wCD}$ by using weights $\sigma^{1,k'}$ and $\sigma^{2,k'}$ which are attributed to V_1 and V_2.

One can also benefit from running multiple passes of the second stage of the algorithm. During the wCD-stage new pairs of weight fields $\sigma^{(0)1,k'}$ and $\sigma^{(0)2,k'}$ for every resulting streamline set $S_{k'}^{(0)}$ can be built, $k' = 1,2$. For each subsequent pass, $l > 0$, the weights from the previous pass $\sigma^{(l-1)1,k'}$ and $\sigma^{(l-1)2,k'}$ are used to generate new streamline sets $S_{k'}^{(l)}$ and, simultaneously, to create a new pair $\sigma^{(l)1,k'}$ and $\sigma^{(l)2,k'}$ for the $l+1$-th pass .

We ran our CD+wCD-algorithm with one pass per each stage on the same COVF which was used for generating networks illustrated in Figures 4.7 and 4.9. It resolved the problem of overlapping streamlines from different sets in many places (frequent for the results of applying just CD-algorithm) and reduced the number of abrupt turns to a minimum (which the original algorithm without context-dependent sampling is prone to), see Figure 4.11. The context radius was chosen to be $R_{\text{ctxt}} = 16$ for the simulation domain of the size $[0,512]^2$, when the average separation distance between the streamlines was 4.0 and the saturation ratio was 1.0 (see the original algorithm [86] for more detail on the parameters). Another example of applying our algorithm while using smaller context radius, $R_{\text{ctxt}} = 8$, is illustrated in Figure 4.12.

A clear limitation of our approach is that it can only handle cross-orientation vec-

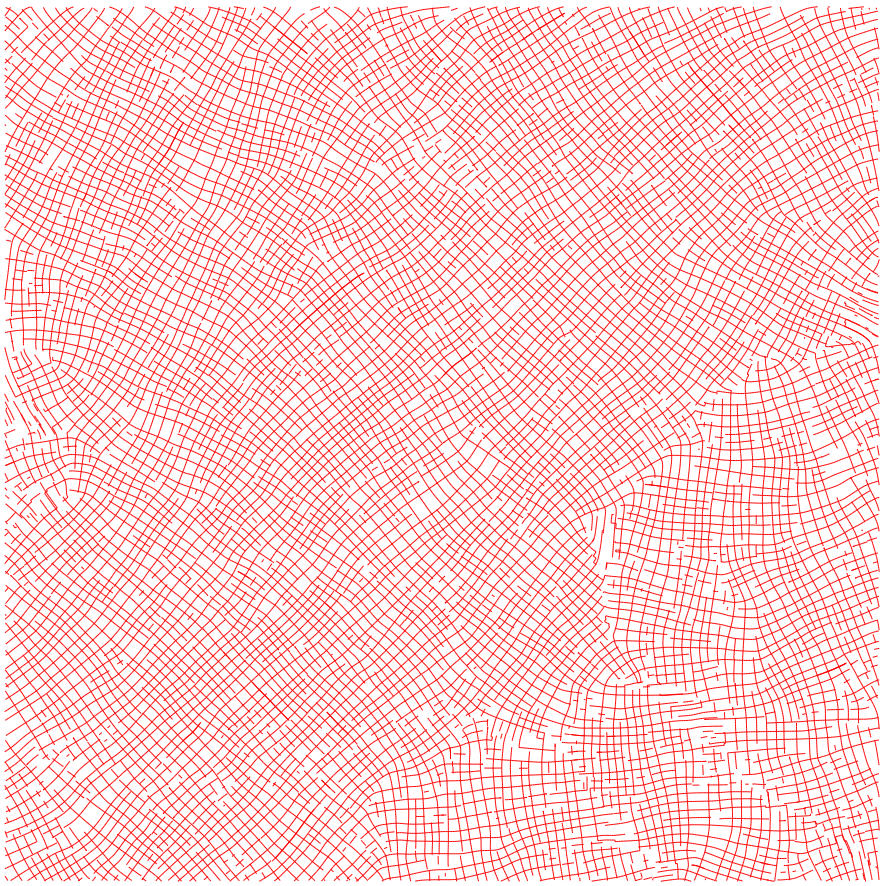


Figure 4.11: Example of curvilinear network generated by the streamline wCD-Algorithm applied to the same COVF which was used for generating networks in Figures 4.7 and 4.9.

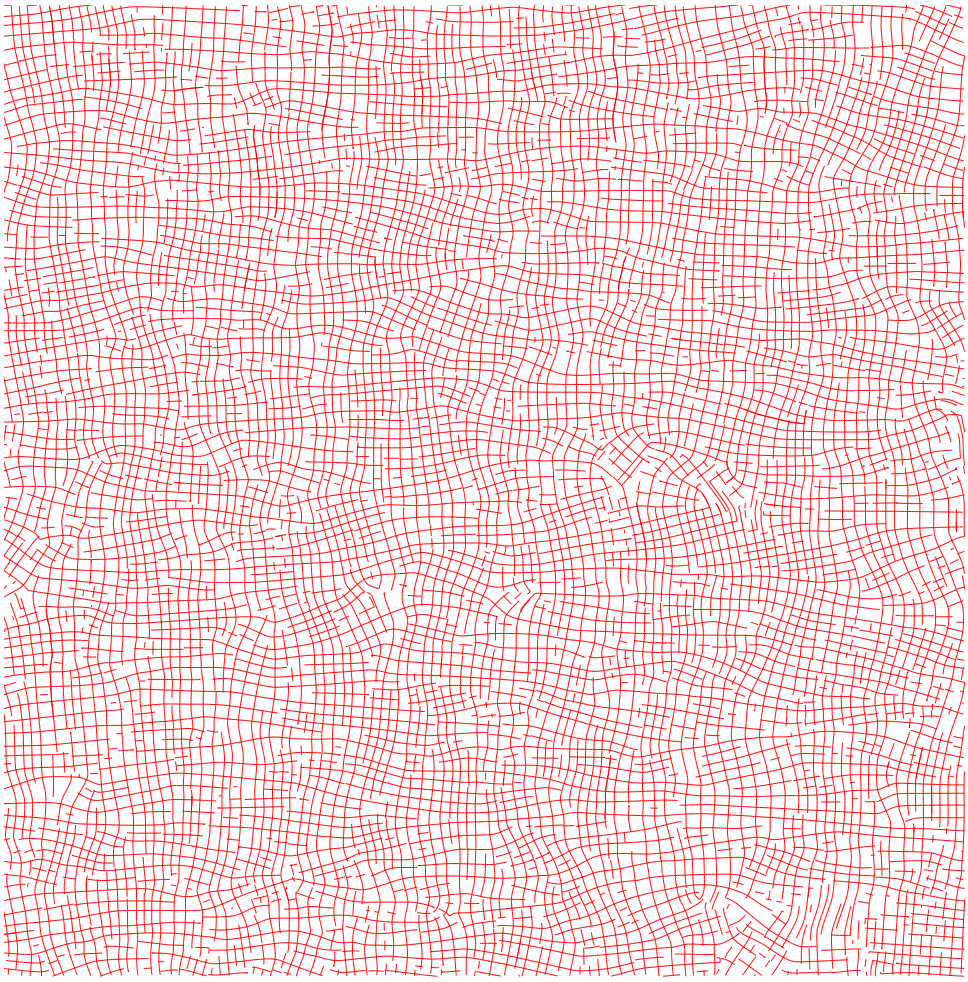


Figure 4.12: Curvilinear network generated from the linear fibre system of the “rain” model illustrated in Figure 3.36-c.

tor fields. If a given vector field has more than two dominant orientations which are highly non-orthogonal, the algorithm can significantly distort original vector field. The last example, shown in Figure 4.13, clearly demonstrate this limitation — the underlying linear fibre process favors inter-fibre angles of 45° and could form four dominant orientations locally — by revealing many spots with overlapping curves.

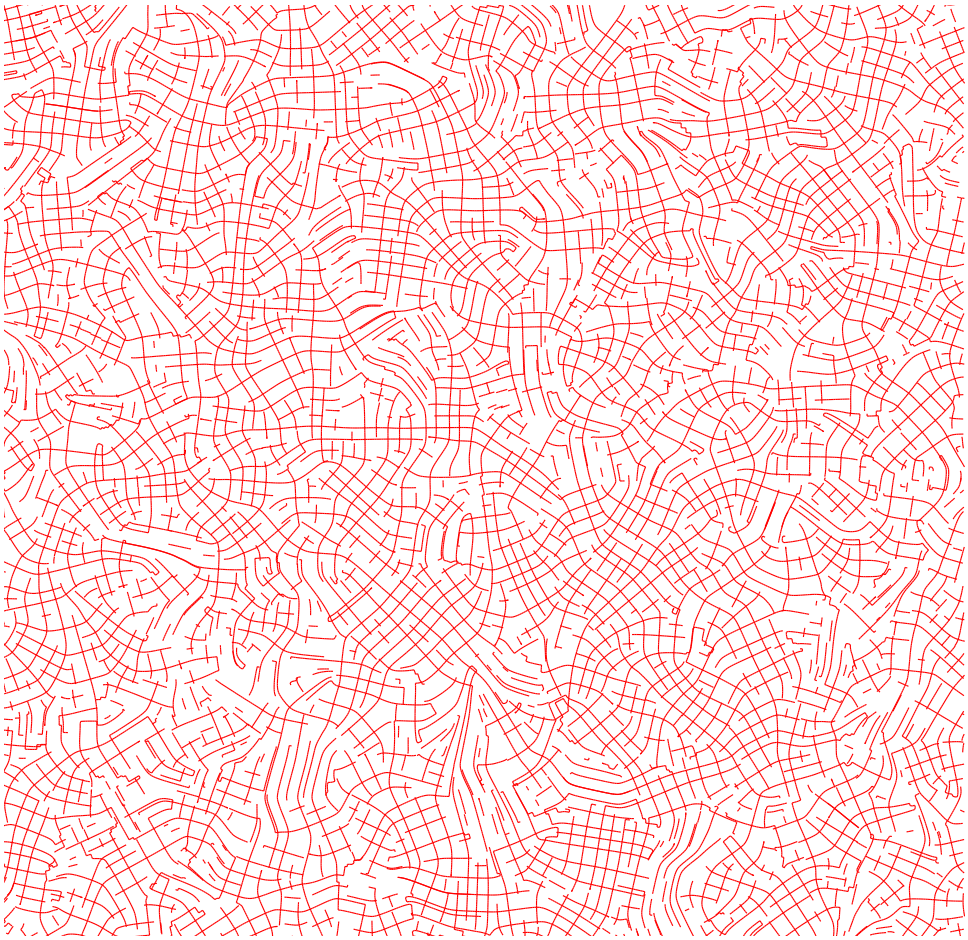


Figure 4.13: Curvilinear network generated from the linear fibre system illustrated in Figure 3.30-a.

4.4 Summary

We have developed a new synthesis algorithm which generates overlapping networks of random curves aligned with a given system of linear fibres. In its general formulation, our algorithm first converts a given fibre system into a multi-orientation vector field (MOVF) which contains more than one orientation at every grid sample with the requirement that one of the orientations is aligned with the closest fibre. MOVFs are extremely helpful in representing the incoherent underlying fibre systems, which is exactly the case for a majority of the linear fibre systems including all the systems which were presented in this work. Second part of our algorithm runs an adapted version of the Mebarki et al. streamline placement algorithm [86] to generate sets of coherent and mostly smooth curves aligned with a given MOVF which is preliminary optimized to emphasize the dominant orientations. Such construction is perfectly suited for systems of linear fibres whose orientations could be decomposed into a small number of overlapping dominant coherent orientation fields (similar to "texture flows" as in [16]). We have demonstrated effectiveness of our two-stage method on the linear fibre systems which have two dominant orientations, always perpendicular to each other, called cross-orientation vector field (COVF). The streamline placement algorithm progressively propagates streamlines towards new directions fetched from COVF by taking into account recently used propagation orientations, or what we call a "context". For better results, such algorithm runs more than one time, such that the resulting orientations from the previous iterations are used as a "context" for the next iterations.

Chapter 5

Results: synthesis and GPU rendering of a feature curve network

In this chapter we demonstrate the effectiveness of our approach in synthesizing and GPU rendering of a random network of curves. We apply our version of the Monte Carlo Metropolis-Hastings algorithm, described in Chapter 3, to generate a sample of a linear fibre process with the fibre interaction model favoring the cross fibre configurations, constrained to be globally aligned with the orientation field of vertical directions. To improve the connectivity and coherency of the resulting system of linear fibres, we apply our adaptation of the streamline placement algorithm, described in Chapter 4. It creates a new network of two overlapping sets of coherent curves, locally aligned with the original linear fibre system. By applying our feature curve algorithm described in Chapter 2, we form a normal map and an alpha transparency map with curvy discontinuity features placed around the curves of the resulting network. We demonstrate a high quality, sharp real-time rendering of the resulting texture maps, while applying arbitrary profiles for the curvy features customizable in real-time.

5.1 Synthesis of a random network of curves

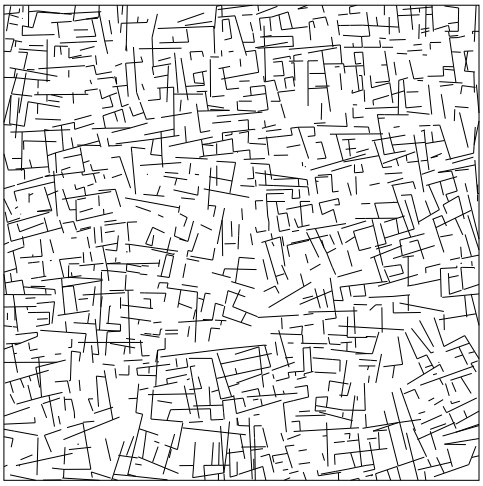


Figure 5.1: A sample of a hard-constrained linear fibre process distributed by the weighted model am2-2 (equations 3.81 and 3.82) with a "cross-pattern" point interaction model given by Table 3.7.

We generated a sample of a Gibbsian linear fibre process, given by equations 3.56 and 3.57, with a point interaction model $h_{\boldsymbol{\theta}}$ favoring orthogonal (or, cross) configurations of nearby fibres (the maximum interaction radius is $R_{\max} = 0.25$), which is given by equation 3.59 with its interaction rate values accumulated in Table 3.7. The phase space for the process had the following components: $\mathcal{X}_P = [0,7]^2$, $\mathcal{X}_L = [0,1]$, and $\mathcal{X}_W = [0,2\pi)$. The proximity radius for the simulation algorithm was set to $R_\Delta = 0.01$. The intensity of the zero-order term was chosen as $\beta_0 = 5.5$. We applied the hard-core type constraints described in our model am2-2 (equations 3.81 and 3.82 with $\alpha = 2.0$) to enforce the target fibre system to be aligned with the orientation field of vertical directions VF-up shown in Figure 3.40 (a). We have run a

simulated annealing algorithm, described in Section 3.7.1, in combination with the total length constraint $l_T = 350$, with the SA-temperature varying within the range $[0.01, 1.0]$. The resulting fibre system, which took $135,000$ iterations and about 10 minutes to generate (one can actually reach the similar quality with less iterations: $54,000$ iterations takes 4 minutes, and $27,000$ iterations takes 2 minutes), is illustrated in Figure 5.1.

It is apparent from Figure 5.1 that our algorithm achieved the crucial desirable properties: (1) the resulting fibre system indeed forms locally a "cross-pattern" — nearby not-aligned fibres are near to orthogonal to each other — and have a reasonable amount of regularity, (2) the fibre system is aligned with a given VF-up vector field which satisfies our target global constraint, (3) the fibre pattern is consistent along the domain boundary as we have applied a wrapping procedure described at the end of Section 3.7.4 (one can compare this fibre system with an unwrapped version shown in Figure 3.37, which clearly demonstrates a pattern break along the boundaries), (4) the fibre pattern is clearly non-periodic.

There is one desired property which still requires improvement: it relates to the connectivity of the aligned fibres. There is a fair amount of fibre breaks in the resulting fibre system. Formation of such breaks is the result of using the inhibiting interaction potentials, which are good in producing regular patterns but tend to repel neighboring fibres. One can try to look for better inhibiting interaction models which have a better balance of inhibiting and attraction. We have described a different approach in this thesis — one can convert linear fibres to a set of smooth coherent curvy fibres, locally aligned with the original fibres.

We ran our streamline placement algorithm, described in Chapter 4, to create a network of two overlapping sets of smooth coherent curves aligned with the linear fibres described above. The fibre orientations in this case clearly form two domi-

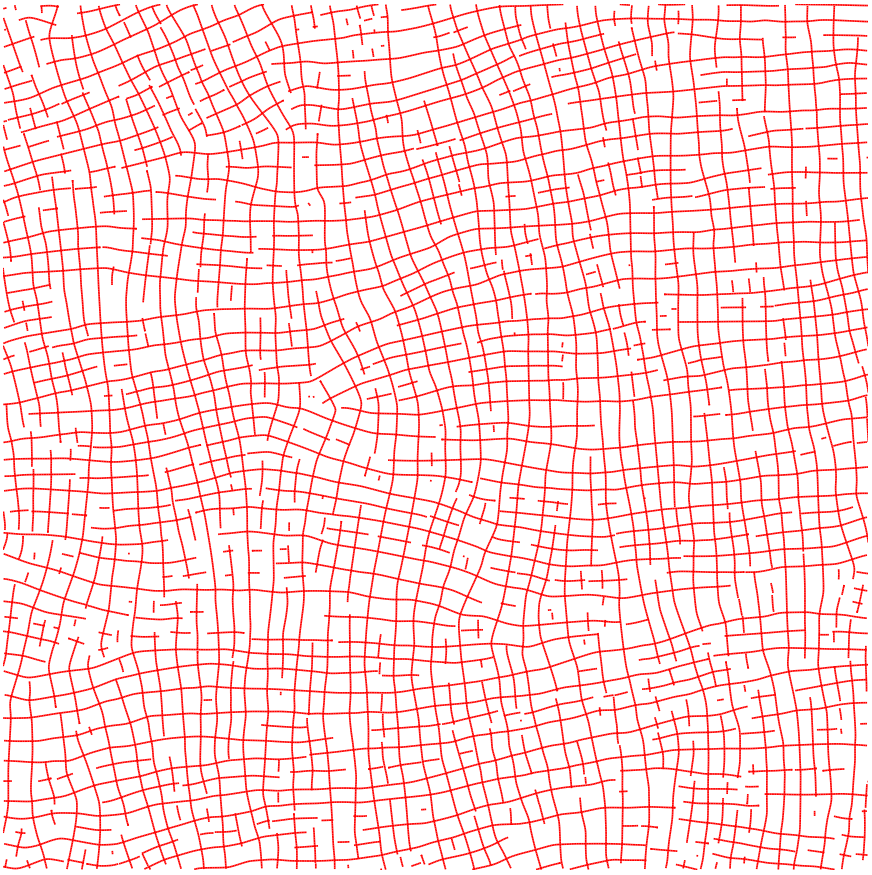


Figure 5.2: Generating a networks of overlapping sets of coherent curves by applying our two-pass streamline CD-wCD-algorithm to a COVF optimized from the linear fibre system illustrated in Figure 5.1.

nant orientation fields, approximately orthogonal to each other, so that applying our streamline algorithm is reasonable. We have optimized a 256×256 COVF from a vector field tangent to the linear fibre system and have applied one iteration of our two-pass CD-wCD algorithm (described in Section 4.3) to such COVF. For the resulting network of curves illustrated in Figure 5.2, we have used the following parameters during streamlines placement within $[0.0, 512.0]^2$ simulation domain: the separation distance between the streamlines was 8.0 and the saturation ratio was 1.0 (see details in [86]). We found that the best value for the context radius R_{ctxt} is equivalent to two separation distances, $R_{\text{ctxt}} = 16.0$ (see Section 4.3 for details).

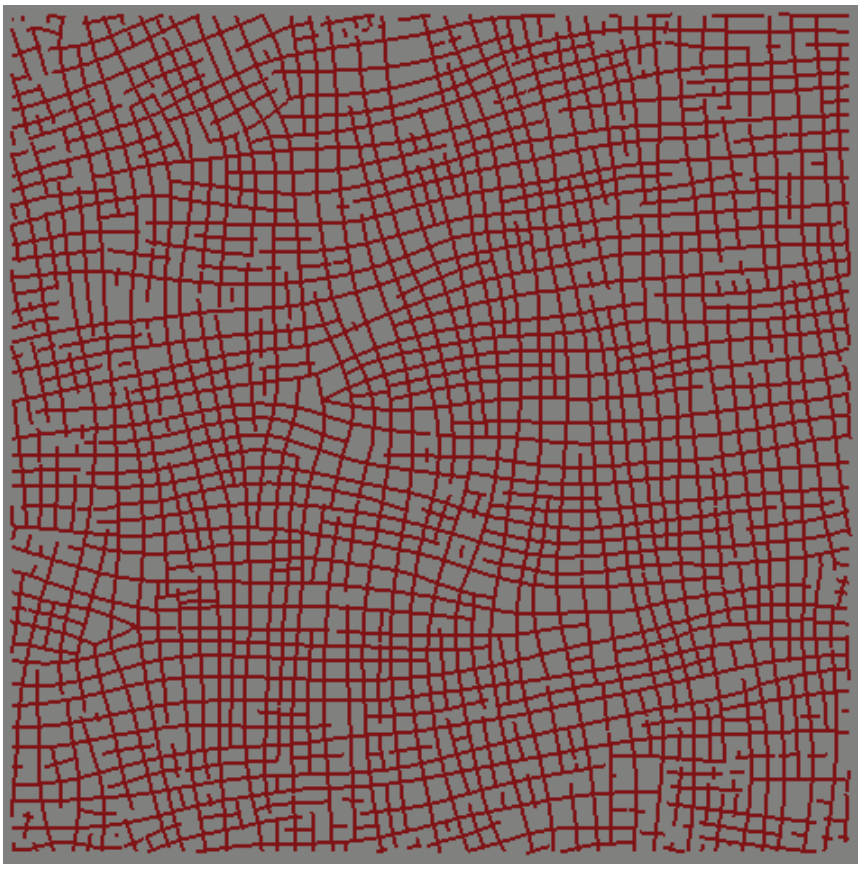


Figure 5.3: Result of post-processing the network of curves illustrated in Figure 5.2.

The quality of the resulting random curves is satisfactory — the number of coherent and smooth fibres is a way larger than that of in the original linear fibre system. However, some post-processing still needs to be done to eliminate a few remaining breaks between the curvy fibres. Figure 5.3 illustrates the result of performing a post-processing which includes the following steps: (1) delete all relatively short open (or, hanging) fibres, (2) translate open fibres towards the closest coherent neighbors and connect them, and (3) stretch one end of open fibre till reaching an open end of another coherent fibre or till running into (or, touching) another fibre.

5.2 Real-time rendering of network of curves

The first example is an embossing on a plane. We converted the network of curves generated in the previous section into discontinuity map of size 512×512 (by using our preprocessing algorithm described in Section 2.6) and formed the sharp features of "furrows" (with the width w of 3 pixels) around the discontinuities as a normal map (explained in Figure 2.6 and Section 2.4) with auxiliary textures of distance map and its gradient. We used our GPU sharp interpolation shader (Section 2.7) to render in real-time the resulting feature maps on a plane. Snapshots of the plane illustrated in Figure 5.4 clearly demonstrate that the feature curves and curve joints remain sharp at any resolution. The width of the feature and profile shape remains consistent very close to the curves joint, thanks to accurate distance function and gradient.

For a second example, we applied the same normal map with network of feature curves onto a surface of a coke cane, which is shown in Figure 5.5. This time we used a smooth "wide ditch" profile instead of the profile of a sharp furrow. The normal map is not discontinuous across the feature curves in this case, so that the sharpness would not be a concern here. However, a coherency of the ditch width and shape

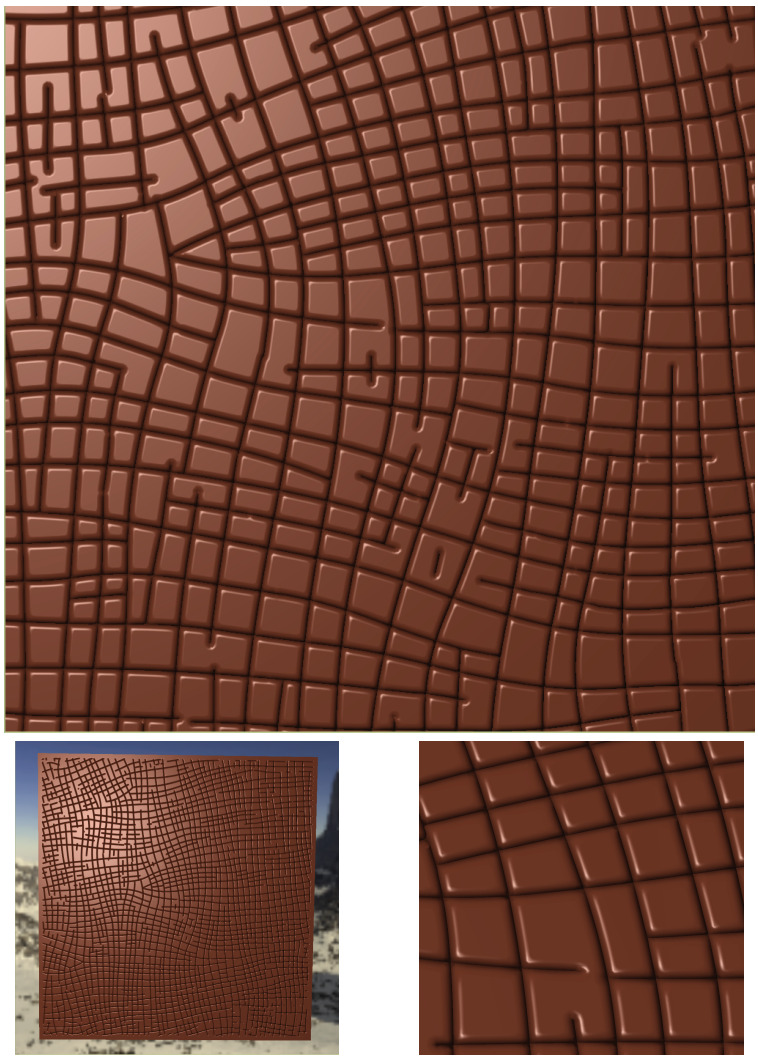


Figure 5.4: Rendering the network of features on a plane.

along the entire can's surface is a consequence of using out approach of distance and gradient close to precise interpolation.

Figure 5.5: Rendering the network of features on a can with a different embossing profile.

Being able to calculate precisely the distance to a network of discontinuity curves allows us to mimic wiremesh appearance of the originally opaque surfaces. We use the distance field to calculate α transparency values of the surface wrapped by geometric detail textures with our feature curves. For the surfaces illustrated in Figures 5.6 and 5.7, we assigned $\alpha(u,v) = 0$ for all the samples (u,v) which are within some distance $d_{\text{SHOW}} > 0$ from the feature discontinuities, $d(u,v) < d_{\text{SHOW}}$. We assign $\alpha(u,v) = 1$ for other samples in the texture domain. We implement transparency

during real-time rendering by clearing z-buffer depth values at the pixels which correspond to the texture samples with $\alpha = 1$. Other pixels are shaded according to our feature curve rendering algorithm.

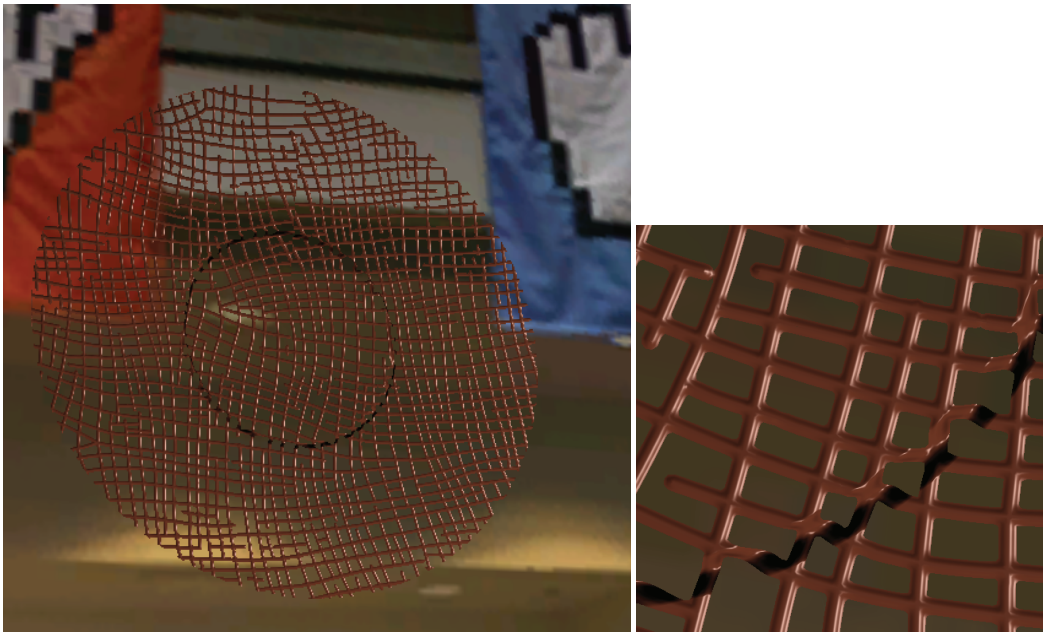


Figure 5.6: Wiremesh-like appearance of a plate (originally illustrated in Figure 2.31).

The next example shows how our technique is combined with transparency and an environment map to obtain glass appearance of the same wiremesh. Refraction direction is calculated with respect to the interpolated normal $\mathbf{n}(u,v)$ at the visible feature points, i.e, $\alpha(u,v) = 0$, and is used to sample the background intensity from a provided environment map. One can imitate changing the shape of the wires by modifying profile of the feature curves keeping the same the original (continuous) normal map.

The half-diameter of a feature "wire" (or, the visibility distance) d_{SHOW} is a pa-

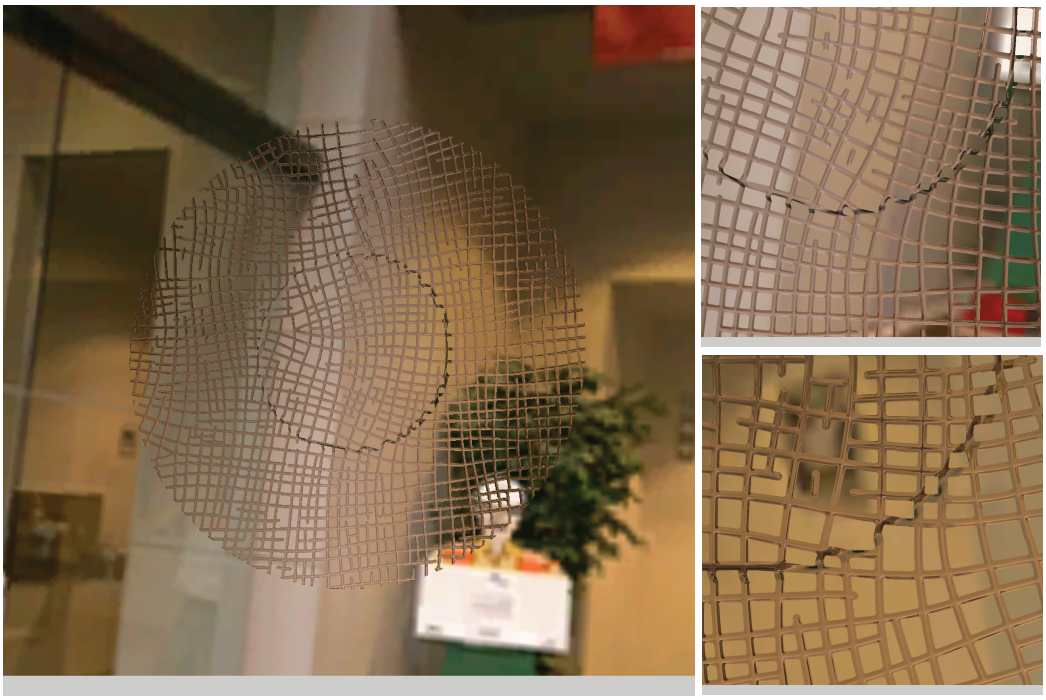


Figure 5.7: Glass wiremesh-like appearance of a plate.

rameter for our pixel shader, which can be modified interactively. The snapshots of interactive process of progressive wire thinning is illustrated in Figure 5.8. The width of the resulting visible wires is consistent within each snapshot. This is a good evidence that our real-time interpolation of given distance field is accurate, which is not always the case for recently published distance-based feature rendering techniques.

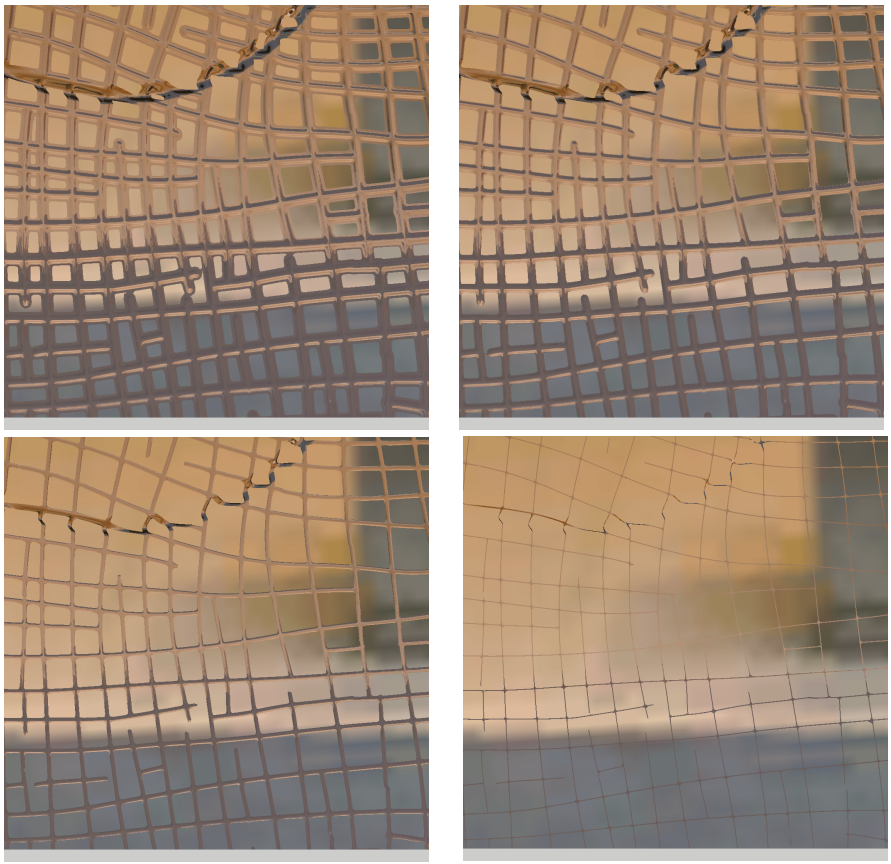


Figure 5.8: Interactive wire thinning by decreasing the visibility distance d_{SHOW}.

Such wiremesh rendering technique can be a good approximation to volumetric textures [103], which actually can be relatively more expensive to manipulate than our feature curves and which quality degrades when viewed from some (even non-

grazing) angles.

Conclusion

In this thesis, we have developed a set of powerful techniques which target curvilinear features from the following different perspectives. Our feature detection algorithm based on optimization of topologically adaptable snakes, robustly segments networks of curvilinear structures from the black and white images. Our method of real-time rendering of surfaces wrapped by textures with high concentration of detail features efficiently stores feature curves in a texture format and smoothly interpolates feature samples preserving discontinuity feature curves at any resolution. We have developed a parametric synthesis algorithm which generates coherent and non-periodic sets of linear fibres and have included comprehensive local and global control. Our adaptation of the streamline generating algorithm converts such sets into networks of smooth curves.

Snake-based detection of curvilinear structures. We have developed a robust feature detection algorithm which efficiently recognizes intricate networks of curvilinear structures from still black and white images. Unlike many relevant existing detection techniques, we avoid a time consuming manual step of localizing potential feature places by throwing a large number of short snakes in the image plane at random. This increases a chance of hitting the target structures in the image. Optimization of a large number of snakes forms wide snake bundles pulled to the image

structures. We have presented a dynamic thinning algorithm which reduces the bundles to a one snake width and link nearby snakes into a graph-like network during the original snake type optimization. To make such optimization stable, we have applied equality constraints to implement links between snakes nodes, instead of using springs proposed in the original snake algorithm.

Real-time rendering of feature curves. We have described a technique for representing and real-time rendering of textures with discontinuity feature curves. In our rendering framework, the discontinuity features are represented as a function of unsigned distance to the discontinuity curves and its gradient. We have shown that a separate interpolation of distance and gradient texture fields are crucial for obtaining high quality results in feature rendering. Our algorithm uses Bezier curve representation for features directly, and no linear approximation artifacts appear at any resolution. While only a limited number of local configurations are allowed by the rendering-time representation of the feature curves, we describe a preprocessing step that simplifies a general network to the form that can be used by the rendering algorithm. The preprocessing step is relatively simple, and, for texture with piecewise linear features, can be done interactively.

Gibbsian fibre processes. We have developed a new parametric model for representing and generating random systems of fibres distributed according to a Gibbs type interaction model. This work represents a new look at the problem of parametric synthesis of textures with rich feature content: instead of synthesizing a new texture from a reference image, where the valuable geometric information about the features is typically lost during imaging, we generate the features themselves. We use outcomes of a random fibre process to define the locations of the features, so that

the features can be formed afterwards around the fibres. Conventional models of the fibre processes are known to model only completely random Poissonian collections of fibres, which cannot produce near-regular patterns. Our random fibre process is capable to generate regular patterns. It is based on a parametric attraction/repulsion fibre interaction model which allows directly specifying the preferred and unwanted local configuration of fibres in the output fibre systems. In addition to an implicit local control to fibre interactions, we have embedded a global control as a set of constraints which force a fibre system to be aligned with a given orientation map. Hard-core global constraints influence all the fibre systems, while soft-core global constraints affect certain areas of the simulation domain (e.g., within a neighborhood of some specified thick feature).

In this work we have explored in a great detail linear fibre processes, which define distributions of line segments in the plane. We have developed a variant of the Monte Carlo Metropolis-Hastings algorithm to simulate random samples of the linear fibre systems which converged in a reasonable time for the examples we have presented in the thesis. The global constraints have been efficiently incorporated into the simulation program by adding corresponding simulated annealing terms into the probability model of the linear fibres.

Linear fibre process is a special case of a general framework of fibre processes, which could include arrangements of curves. While leaving a thorough development of models for curvy fibres for a future, we have presented a synthesis method which converts a given system of linear fibres into overlapping sets of coherent and mostly smooth curves. We assume that the linear fibres form roughly two independent orientation flows. We optimize the flows to build cross-orientation vector field (COVF) with orientation pairs being mostly continuous along each flow and perpendicular to each other. We have adapted the streamline placement algorithm by Mebarki et

al. [86] which is capable of sampling a given COVF in a way that respects the current "context" — the short history of propagation steps and some propagation neighborhood.

Future work

Snake-based curvilinear structures detection. The original deformable models have already demonstrated their effectiveness for tracking object boundaries [136]. Similarly, it is desirable to extend our snake-based detection algorithm to track curvilinear structures evolving in video sequences. The main functionality which should be implemented for a possible extension algorithm is the ability for individual snakes to shrink and to grow in length when the corresponding curvilinear structures deform across the frames with simultaneous update of the corresponding plane graph (which tracks snakes adjacency in our approach) by inserting new or deleting the old nodes and edges when it is necessary.

Our algorithm is tuned to detect thin curvilinear features only and may fail to recognize conventional feature boundaries. A generalization of our algorithm which can perform robust segmentation in the images with a mixture of curvilinear structures and feature boundaries could be very valuable.

Real-time rendering of feature curves. We demonstrated effectiveness of our method for interpolating normal maps. However, similar approaches can be applied to more advanced types of geometric mapping, such as relief and displacement maps. Our main limitation that no more than two curves can be stored in a texel, can be resolved on modern GPUs by using arbitrary length discontinuity descriptors and indirect lookups, similar to the data structures used in concurrent work [94, 105].

As discussed briefly in [138], one can remove some of the topological restrictions by considering higher resolution textures, localized to the areas of multiple curve intersections. Higher-order interpolation for the distance and its gradient would improve the quality of the result at the expense of more stored data. Hashing and a variant of the ADF approach [49] can be used to optimize memory consumption of our technique by employing the fact that a majority of texels are not affected by discontinuities.

Gibbsian fibre processes. In our model of random fibres, we mainly focus on generating systems of linear fibres, which are represented by line segments. However, our method is not restricted to just linear fibres and, in principle, can be extended to a more general case which can handle curvy primitives of bounded length. For a general model, we have already outlined a general construction of fibre distributions of the exponential family, based on interaction integrals. A challenging problem towards completing the fibre model for curves is to design a proper phase space. We believe that a very promising direction is to try working within the space of quadratic Bezier segments which are completely defined by three control points (a representation space in this case is a bounded subset of $\mathbb{R}^6$).

Our model of linear fibres is capable of producing a variety of near-regular configurations of fibres by tweaking the parameters of the fibre interaction model which are easy to interpret. However, optimizing the constraints parameters to make the linear fibres better connected may be not trivial in some cases, and may notably distort an original correlation pattern of the fibres. Considering the space of fibres connected at any time is an interesting alternative direction to explore.

Statistical inference algorithms have been provided for Gibbs type spatial point process models. In particular, the parameters of the exponential family of pair-

potential models of repulsion and attraction, based on step functions, can be rapidly recovered by computing approximate maximum pseudolikelihood estimates [5, 6, 7]. Our model of random linear fibres is formed by integrating such pair-potentials. This gives a hope that an inverse problem for linear fibres can be developed by using the same principles which were used to derive the inference algorithms for underlying point processes.

Our COVF-based streamline placement algorithm requires some improvements as in some cases it produces noticeable closely overlapping curves. On the other note, our assumption about two cross oriented coherent flows is very strong and restrictive: several mentioned in the thesis fibre systems clearly formed more than two dominant orientation fields. One can develop a more general approach capable of optimizing multi-orientation vector fields (MOVFs) from a given linear fibre system and use the ideas which shaped our COVF-based streamline algorithm in developing its extension to generate more than two overlapping sets of coherent curves aligned with a given MOVF.

Basic Notation

$\mathbb{R}^d$	:	d-dimensional Euclidean space	
$\chi_A(x)$	:	characteristic function of a set A	
$\lambda \bullet (a,b)$	:	linear interpolation function between a and b	81, 209
$\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}$	:	standard orthonormal basis in $\mathbb{R}^3$	

Detecting curvilinear features

$\mathbf{s}$	:	Snake (parametrization)	20
$\mathbf{s}(u_j)$	:	Snake's j-th node	19

Feature curves

$L[\boldsymbol{p}, \boldsymbol{q}]$	:	Line segment with end points $\boldsymbol{p}$ and $\boldsymbol{q}$	49
$B[\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2]$	:	Bezier segment with control points $\boldsymbol{b}_0$, $\boldsymbol{b}_1$, and $\boldsymbol{b}_2$	49
$\gamma_B(t)$	:	A parametrization of a Bezier segment B	49
$(\hat{C}_{ij})$	:	Texture of discontinuity configurations C	63
$(\hat{S}_{ij})$	:	Texture of discontinuity signatures S	64
$(\hat{d}_{ij})$	:	Unsigned distance texture map	71
$(\nabla \hat{d}_{ij})$	:	Texture of distance gradients	71

$\{\boldsymbol{p},\boldsymbol{q}\}$	:	Curvilinear segment, which is parameterized by some quadratic curve γ, s.t. $\gamma(0)=\boldsymbol{p}$ and $\gamma(1)=\boldsymbol{q}$	81
$[\boldsymbol{p},\boldsymbol{q})$	:	A ray $\boldsymbol{p}+t(\boldsymbol{p}-\boldsymbol{q})$, $t\geq 0$	81

Fibre processes

$\boldsymbol{X}$	:	spatial point process, $(\Omega,\mathcal{A},P)\rightarrow(X_\infty,\mathcal{F}_\infty,\mu)$	122
$\boldsymbol{X}(B)$	:	random variable of the number of points of X located in B, $B\in\mathcal{B}$	122
$\lambda(B)$	:	intensity measure of a set $B\in\mathcal{B}$	122, 127
X_∞	:	exponential space over point process simulation domain X	126
$\mathcal{F}_\infty$	:	σ-algebra of the exponential space X_∞	127
$\mathcal{B}$	:	σ-algebra generated by bounded Borel sets over X	126
$\mu(F)$	:	Poisson measure (distribution of Poisson point process) on $\mathcal{F}_\infty$	127
$\pi(F)$	:	point process measure derived from the Poisson measure	128
$f(x_1\circ x_2\circ\ldots\circ x_n)$	:	probability density function of a points arrangement	128
$\Phi=\varphi_1\circ\varphi_2\circ\ldots\circ\varphi_n$	:	fibre system of n fibres, $N(\Phi)=n$	144,146
$\Phi_{\hat{i}}\equiv\Phi\setminus\varphi_i$	:	reduction of a fibre system Φ by a fibre $\varphi_i\in\Phi$	169
$\beta(\varphi)$	:	zero-order interaction potential	154,161
$\tilde{h}_1(\varphi)$ or $(\varphi)_1$	:	first-order interaction potential	155
$\tilde{h}_2(\varphi,\psi)$	:	pair-potential,	
$(\varphi,\psi)_2$		or second-order interaction potential	151

$\varphi \underset{R}{\sim} \psi$	:	fibre neighborhood relation	154,173
$h_{\boldsymbol{\theta}}(d,w)$	:	point interaction function	161
$\mathcal{X}$	:	fibre representation space	144, 158
$\boldsymbol{\mathcal{X}}$	:	fibre process, $(\Omega,\mathcal{A},P) \to (\mathcal{X}_\infty,\widetilde{\mathcal{F}}_\infty,\mu)$	134,146
$\nu(B)$	:	intensity measure of a set $B \in \widetilde{\mathcal{B}}$	144
$\mathcal{X}_\infty$	:	exponential space over fibres process simulation domain $\mathcal{X}$	144
$\widetilde{\mathcal{F}}_\infty$	:	σ-algebra of the exponential space $\mathcal{X}_\infty$	144
$\widetilde{\mathcal{B}}$	:	σ-algebra generated by bounded Borel sets over $\mathcal{X}$	146
$\Phi(B)$	:	the length of a portion of a fibre system which overlaps with $B \in \mathcal{B}$	134
l_φ or $\varphi(X)$	:	the length of a fibre φ	134
$\lambda_c(\varphi_i;\Phi_{\hat{i}})$	:	conditional intensity for adding a new fibre ψ to a fibre system Φ	171
$Nb(\psi;\Phi,R) =$	:	R-neighborhood of a fibre ψ within a fibre system Φ	174
V	:	a vector field, $\mathbb{R}^2 \to \mathbb{E}$,$V_{\boldsymbol{p}}p = V(v)$, $\boldsymbol{p} \in \mathbb{R}^2$	206

Bibliography

[1] K. Abe, F. Mizutani, and C. Wang. Thinning of gray-scale images with combined sequential and parallel conditions for pixel removal. *IEEE Trans. Systems, Man and Cybernetics*, 24:294–299, 1994. 15

[2] R. V. Ambartsumian. Random fields of segments and random mosaics on a plane. In *Proceedings of Sixth Berkeley Symposium on Mathematical Statistics and Probability*, volume 3, pages 369–381. University of California Press, 1972. 136

[3] C. H. Arns, J. Mecke, K. Mecke, and D. Stoyan. Second-order analysis by variograms for curvature measures of two-phase structures. *European Physical Journal B*, 47:397–409, 2005. 105, 135

[4] Michael Ashikhmin. Synthesizing natural textures. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226, New York, NY, USA, 2001. ACM. 118

[5] A. Baddeley and R. Turner. Practical maximum pseudolikelihood for spatial point patterns. *Australian and New Zealand Journal of Statistics*, 42:283–322, 2000. 128, 158, 270

[6] A. Baddeley and R. Turner. Spatstat: an r package for analyzing spatial point patterns. *Journal of Statistical Software*, 12(6):1–42, 2005. 128, 158, 270

[7] A. Baddeley and R. Turner. *The spatstat Package*. http://www.spatstat.org, 2007. 128, 158, 270

[8] A. J. Baddeley and Jesper Moller. Nearest-neighbor markov point processes and random sets. *International Statistical Review*, 57:89–121, 1989. 125

[9] A. J. Baddeley and R. Turner. Practical maximum pseudolikelihood for spatial point patterns. *Advances in Applied Probability*, 30(2):273, 1998. 125

[10] Kavita Bala, Bruce J. Walter, and Donald P. Greenberg. Combining edges and points for interactive high-quality rendering. *ACM Transactions on Graphics*, 22(3):631–640, July 2003. 40, 42

[11] D. Ballard and C. Brown. *Computer Vision*, chapter 4. Prentice-Hall, 1982. 16

[12] Y. Bando, T. Kuratate, and T. Nishita. A simple method for modeling wrinkles on human skin. In *Pacific Graphics*, 2002. 107

[13] Ziv Bar-Joseph, Ran El-Yaniv, Dani Lischinski, and Michael Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):120–135, 2001. 117

[14] O. Barndorff-Nielsen, W. Kendall, and M. N. M. Lieshout. *Stochastic geometry likelihood and computation*. London, Chapman and Hall, 1999. 120, 125, 129

[15] Barry G. Becker and Nelson L. Max. Smooth transitions between bump rendering algorithms. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 183–190, August 1993. 44

[16] O. Ben-Shahar and S. W. Zucker. The perceptual organization of texture flow: a contextual inference approach. *IEEE Trans. Pattern analysis and machine intelligence*, 25(4):401–417, April 2003. 228, 252

[17] M. Bern, D. Goldberg, R. C. Stevens, and P. Kuhn. Automatic classification of protein crystallization images using a curve-tracking algorithm. *Journal of Applied Crystallography*, 37(2):279–287, 2004. 2

[18] James F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)*, volume 12, pages 286–292, August 1978. 43

[19] James F. Blinn. How to solve a cubic equation, part 5: Back to numerics. *IEEE Computer Graphics and Applications*, 27(3):78–89, 2007. 59

[20] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 361–368, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 117

[21] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover Publications, 1966. 228

[22] D. S. Carter and P. M. Prenter. Exponential spaces and counting processes. *Probability Theory and Related Fields*, 21(1), March 1972. 126

[23] Frank H. Clarke. *Optimization and Nonsmooth Analysis*. Cambridge University Press, 1987. 54

[24] Peter Clifford and Geoff Nicholls. Comparison of birth-and-death and metropolis-hastings markov chain monte carlo for the strauss process. Technical report, Department of Mathematics, University of Auckland, NZ, http://www.math.auckland.ac.nz/ nicholls/linkfiles/papers/Strauss.ps.gz, 1994. 142

[25] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. Wang tiles for image and texture generation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 287–294, New York, NY, USA, 2003. ACM. 118

[26] Robert L. Cook. Shade trees. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, volume 18, pages 223–231, July 1984. 44

[27] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, 1986. 125

[28] David Cox, John Little, and Donal O'Shea. *Using algebraic geometry*, chapter 3, pages 78–82. Graduate Texts in Mathematics. Springer-Verlag New York, 1998. 74

[29] Oana G. Cula, Kristin J. Dana, Frank P. Murphy, and Babar K. Rao. Skin texture modeling. *Int. J. Comput. Vision*, 62(1-2):97–119, 2005. 2, 14

[30] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes*, volume I. Springer Verlag, 2nd edition, 2005. 128

[31] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, second edition, 2000. 176

[32] P. J. Diggle, D. J. Gates, and A. Stibbard. A nonparametric estimator for pairwise-interaction point processes. *Biometrika*, 74(4):763–770, 1987. 125, 129

[33] Peter J. Diggle, Thomas Fiksel, Pavel Grabarnik, Yosihiko Ogata, Dietrich Stoyan, and Masaharu Tanemura. On parameter estimation for pairwise interaction point processes. *International Statistical Review*, 62(1):99–117, 1994. 125

[34] Peter J. Diggle and Richard J. Gratton. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):193–227, 1984. 129

[35] W. Donelly. *GPU Gems 2 : Programming Techniques for High- Performance Graphics and General-Purpose Computation*, chapter 8 Per-pixel displacement mapping with distance functions. Addison-Wesley Professional, 2005. 44

[36] Weiming Dong, Ning Zhou, and Jean-Claude Paul. Optimized tile-based texture synthesis. In *GI '07: Proceedings of Graphics Interface 2007*, pages 249–256, New York, NY, USA, 2007. ACM. 118

[37] R.O. Duda and P.E. Hart. Use of the hough transform to detect lines and curves in pictures. *Communications of the ACM*, pages 11–15, 1972. 16

[38] S. I. Dudov. On the generalized gradient of the distance function. *Journal of Mathematical Sciences*, 100(6):2593–2600, 2000. 54

[39] C.R. Dyer and A. Rosenfeld. Thinning algorithms for gray-scale pictures. *IEEE Trans. Pattern analysis and machine intelligence*, 1(1):88–90, January 1979. 15

[40] David S. Ebert, Steven Worley, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Kenton F. Musgrave. *Texturing and Modeling: A Procedural Approach*. Academic Press, Inc., Orlando, FL, USA, 1998. 115

[41] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, New York, NY, USA, 2001. ACM. 118

[42] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1033, Washington, DC, USA, 1999. IEEE Computer Society. 117

[43] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design*, chapter 4. Academic Press, San Diego, 4 edition, 1997. 68

[44] N. A. Fava and L. A. Santalo. Plate and line segment processes. *Journal of Applied Probability*, 15(3):494–501, 1978. 136

[45] S. F. Firsken, R. N. Perry, and T. R. Jones. *Detail-directed hierarchical distance fields*. US Patent 6,396,492, May 2002. 41

[46] M. Fischler, J. Tenenbaum, and H. Wolf. Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge inte-

gration technique. *Computer Graphics and Image Processing*, (15):201–223, 1981. 2

[47] Kurt W. Fleischer, David H. Laidlaw, Bena L. Currin, and Alan H. Barr. Cellular texture generation. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 239–248, New York, NY, USA, 1995. ACM. 115

[48] A. Fortier, D. Ziou, C. Armenakis, and S. Wang. Survey of work on road extraction in aerial and satellite images. tech report 241. Technical report, Universite de Sherbrooke, Quebec, Canada, 1999. 2

[49] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 249–254, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 41, 269

[50] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos. Dynamic programming for detecting, trackin, and matching deformable contours. *IEEE Trans. on Pattern Analysis and Machine Interlligence*, 17(3):294–302, March 1995. 17

[51] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):1–14, 1996. 2

[52] C. J. Geyer and J. Moller. Simulation procedures and likelihood inference for spatial point processes. *Scandinavian Journal of Statistics*, 21:359–373, 1994. 137, 169

[53] Stephane Gobron and Norishige Chiba. Crack pattern simulation based on 3d surface cellular automata. *The Visual Computer*, 17(5):287–309, 2001. 105

[54] Simon Green. The opengl framebuffer object exntension. In *Game Developers Conference 2005*, San Francisco, CA, Mar. 2005. 88

[55] Geoffrey R. Grimmett and David R. Stirzaker. *Probability and Random Processes*. Oxford, University Press, third edition, 2004. 122, 123

[56] User's Guide. *Matlab Optimization Toolbox 4*. http://www.mathworks.com/access/helpdesk/help/toolbox/optim/, 2008. 232

[57] Olle Haggstrom, Marie-Colette N. M. Van Lieshout, and Jesper Moller. Characterization results and markov chain monte carlo algorithms including exact simulation for some spatial point processes. *Bernoulli*, 5(4):641–658, 1999. 125

[58] Jianwei Han, Kun Zhou, Li-Yi Wei, Minmin Gong, Hujun Bao, Xinming Zhang, and Baining Guo. Fast example-based surface texture synthesis via discrete optimization. *Vis. Comput.*, 22(9):918–925, 2006. 118

[59] R. M. Haralik. Ridges and valleys on digital images. *Computer Vision, Graphics, and Image processing*, 22:28–38, 1983. 16

[60] Paul Heckbert. Discontinuity meshing for radiosity. In *3rd Eurographics Workshop on Rendering*, pages 203–226, May 1992. 40

[61] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on*

Computer graphics and interactive techniques, pages 229–238, New York, NY, USA, 1995. ACM. 4, 116, 117

[62] Wolfgang Heidrich, Katja Daubert, Jan Kautz, and Hans-Peter Seidel. Illuminating micro geometry based on precomputed visibility. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 455–464, July 2000. 44

[63] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 517–526, July 2000. 230, 232

[64] Olle Hggstrm, Marie-Colette N.M. Van Lieshout, and Jesper Mller. Characterization results and markov chain monte carlo algorithms including exact simulation for some spatial point processes. *Bernoulli*, 5(4):641–658, 1999. 142, 173

[65] C. P. Hornung. *Background Patterns, Textures, and Tints*. Dover Publications, 1976. 228

[66] P. V. C. Hough. Methods and means for recognising complex patterns. Technical Report United States Patent: 3,069,654,, December 1962. 16

[67] Ryan M. Ismert, Kavita Bala, and Donald P. Greenberg. Detail synthesis for image-based texturing. In *2003 ACM Symposium on Interactive 3D Graphics*, pages 171–175, April 2003. 45

[68] S. Karkkainen and C. Lantuejoul. Orientational analysis of planar fibre systems observed as a poisson shot-noise process. *Journal of Microscopy*, 228(1):88–96, 2007. 136

[69] F. P. Kelly and B. D. Ripley. A note on strauss's model for clustering. *Biometrika*, 63(2):357–360, 1976. 125, 129

[70] Wilfrid S. Kendall and Jesper Mller. Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Advances in Applied Probability*, 32(3):844–865, 2000. 142, 173

[71] W.S. Kendall. *Perfect simulation for the area-interaction point process*, chapter in "Probability Towards 2000", edited by L. Accardi and C.C. Heyde, pages 218–234. Springer New York, 1998. 142, 173

[72] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. *ACM Trans. Graph.*, 24(3):795–802, 2005. 118

[73] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*, 22(3):277–286, 2003. 118

[74] L. Lam, S.W. Lee, and C.Y. Suen. Thinning methodologies: A comprehensive survey. *IEEE Trans. Pattern analysis and machine intelligence*, 14(9):869–885, September 1992. 15

[75] G. M. Laslett. The survival curve under monotone density constraints with applications to two-dimensional line segment processes. *Biometrika*, 69(1):153–160, 1982. 136

[76] Peter A. W. Lewis and Gerald S. Shedler. Simulation of nonhomogeneous poisson processes by thinning. Technical Report 26, 403–413, Naval Research Logistic Quaterly, 1979. 123

[77] Jing Lin, Hui-Tang Chen, Ping Jiang, Yue-Juan Wang, and Peng-Yung Woo. Curve tracking and reproduction by a robot with a vision system. *Journal of Robotic Systems*, 16(10):547–556, 1999. 2

[78] Erik Lindholm, Mark J. Kligard, and Henry Moreton. A user-programmable vertex engine. In *Proceedings of SIGGRAPH 01*, pages 149–158, New York, NY, USA, 2001. ACM. 46

[79] Yanxi Liu, Wen-Chieh Lin, and James Hays. Near-regular texture analysis and manipulation. *ACM Trans. Graph.*, 23(3):368–376, 2004. 118

[80] Charles Loop and Jim Blinn. Resolution independent curve rendering using programmable graphics hardware. *ACM Trans. Graph.*, 24(3):1000–1009, 2005. 42

[81] Jianye Lu, Athinodoros S. Georghiades, Andreas Glaser, Hongzhi Wu, Li-Yi Wei, Baining Guo, Julie Dorsey, and Holly Rushmeier. Context-aware textures. *ACM Trans. Graph.*, 26(1):3, 2007. 118

[82] Nelson L. Max. Shadows for bump-mapped surfaces. In *Advanced Computer Graphics (Proceedings of Computer Graphics Tokyo '86)*, pages 145–156, 1986. 44

[83] Nelson L. Max. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer*, 4(2):109–117, July 1988. 44

[84] Morgan McGuire and Max McGuire. Steep parallax mapping, 2005. 44

[85] T. McInerney and D. Terzopoulos. Topologically adaptable snakes. In *Proceedings of International Conference on Computer Vision*, pages 840–845, June 1995. 18

[86] Abdelkrim Mebarki, Pierre Alliez, and Olivier Devillers. Farthest point seeding for efficient placement of streamlines. In *Proc. of IEEE Visualization*, 2005. xx, 227, 237, 239, 248, 252, 257, 268

[87] Don P. Mitchell. Generating antialiased images at low sampling densities. In *Proceedings of ACM SIGGRAPH 1987*, Computer Graphics Proceedings, Annual Conference Series, pages 65–72, 1987. 125

[88] B. S. Morse, W. A. Barrett, J. K. Udupa, and R. P. Burton. Trainable optimal boundary finding using two-dimensional dynamic programming. Technical Report MIPG180, University of Pennsylvania, Philadelphia, PA, March 1991. 17

[89] E. N. Mortensen. Adaptive boundary detection using 'live-wire' two-dimensional dynamic programming. Master's thesis, Brigham Young University, Provo, UT, August 1995. 17

[90] E. N. Mortensen and W. A. Barret. Intelligent scissors for image composition. In *ACM SIGGRAPH '95. International Conference on Computer Graphics and Interactive Techniques*, August 1995. 17

[91] E. N. Mortensen and W. A. Barret. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5), 1998. 17

[92] James Munkers. *Topology*. Prentice Hall, 2nd edition, 2000. 149

[93] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 41–50, New York, NY, USA, 1989. ACM. 115

[94] D. Nehab and H. Hoppe. Random-access rendering of general vector graphics. In *SIGGRAPH Asia 2008*. 43, 268

[95] Jerzy Neyman and Elizabeth L. Scott. Statistical approach to problems of cosmology. *Journal of the Royal Statistical Society*, 20:1–43, 1958. 124

[96] Tomoyuki Nishita, Thomas W. Sederberg, and Masanori Kakimoto. Ray tracing trimmed rational surface patches. *SIGGRAPH Comput. Graph.*, 24(4):337–345, 1990. 83

[97] Yosihiko Ogata and Masaharu Tanemura. Estimation of interaction potentials of spatial point patterns though the maximum likelihood procedure. *Annals of the Institute of Statistical Mathematics*, 33(1):315–338, 1981. 125

[98] Yosihiko Ogata and Masaharu Tanemura. Likelihood analysis of spatial point patterns. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(3):496–518, 1984. 125, 129

[99] Manuel M. Oliveira, Gary Bishop, and David McAllister. Relief texture mapping. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 359–368, July 2000. 44

[100] Pierre Parent, Steven, and W. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:823–839, 1989. 2

[101] E. Parilov, I. Rosenmberg, and D. Zorin. Real-time rendering of normal maps with discontinuities. Technical report, Courant Institute of Mathematical Sciences, NYU, 2005. 86

[102] P. Parker and R. Cowan. Some properties of line segment processes. *Journal of Applied Probability*, 13(1):96–107, 1976. 136

[103] Jianbo Peng, Daniel Kristjansson, and Denis Zorin. Interactive modeling of topologically complex geometric detail. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 635–643, New York, NY, USA, 2004. ACM. 263

[104] Ken Perlin. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 287–296, New York, NY, USA, 1985. ACM Press. 44, 115

[105] R. N. Perry and S. F. Firsken. *Method and apparatus for rendering cell-based distance fields using texture mapping*. US Patent 6,917,369, June 2005. 268

[106] R. N. Perry and S. F. Firsken. *Method for converting two-dimensional objects to distance fields*. US Patent 7,030,881, April 2006. 41

[107] Fábio Policarpo, Manuel M. Oliveira, and Jo ao L. D. Comba. Real-time relief mapping on arbitrary polygonal surfaces. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 155–162, New York, NY, USA, 2005. ACM Press. 44

[108] Kris Popat and Rosalind W. Picard. Novel cluster-based probability model for texture synthesis, classification, and compression. In *In Visual Communications and Image Processing*, pages 756–768, 1993. 117

[109] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision*, 40(1):49–70, 2000. 117

[110] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 118

[111] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 2002. 59, 167

[112] Chris Preston. Spatial birth-and-death processes. *Bull. Int. Statist. Inst.*, 46(2):371–391, 1975. 137, 142

[113] James Gary Propp and David Bruce Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. In *Proceedings of the seventh international conference on Random structures and algorithms*, pages 223–252, New York, NY, USA, 1996. John Wiley & Sons, Inc. 142

[114] Zheng Qin, Michael D. McCool, and Craig Kaplan. Precise vector textures for real-time 3d rendering. In *SI3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 199–206, New York, NY, USA, 2008. ACM. 43, 60

[115] Zheng Qin, Michael D. McCool, and Craig S. Kaplan. Real-time texture-mapped vector glyphs. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 125–132, New York, NY, USA, 2006. ACM Press. 42, 59

[116] G. Ramanarayanan, Kavita Bala, and Bruce Walter. Feature-based textures. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 265–274, June 2004. 37, 40, 42

[117] B. D. Ripley. *Statistical inference for spatial processes*. Cambridge University Press, Cambridge New York, 1988. 125

[118] B. D. Ripley and F. P. Kelly. Markov point processes. *Journal of the London Mathematical Society*, 15:188–192, 1977. 141

[119] Brian D. Ripley. *Stochastic simulation*. John Wiley & Sons, Inc., New York, NY, USA, 1987. 123

[120] Sheldon M. Ross. *Simulation*. Academic press, 2 edition, 1997. 190

[121] Andrew J. Round, Andrew W. G. Duller, and Peter J. Fish. Lesion classification using skin patterning. *Skin Research and Technology*, 6(4):183–192, 2000. 2, 14

[122] Mike Salisbury, Corin Anderson, Dani Lischinski, and David H. Salesin. Scale-dependent reproduction of pen-and-ink illustrations. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 461–468, August 1996. 40, 42

[123] Roy Saunders and Gerald M. Funk Richard J. Kryscio. Poisson limits for a hard-core clustering model. *Stoch. Proc. Appl.*, 12(2):97–106, 1982. 125

[124] Pradeep Sen. Silhouette maps for improved texture magnification. In *Graphics Hardware 2004*, pages 65–74, August 2004. 37, 40, 41, 61

[125] Pradeep Sen, Michael Cammarano, and Pat Hanrahan. Shadow silhouette maps. *ACM Transactions on Graphics*, 22(3):521–526, July 2003. 41

[126] Zhishun She and P. J. Fish. Skin lesion differentiation using skin line direction. In *Medical Image Understanding and Analysis*. University of Portsmouth. 2, 14

[127] Eero P Simoncelli and William T Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *International Conference on Image processing*, pages 444–447, 1995. 117

[128] Eero P Simoncelli and A Karasaridis. A filter design technique for steerable pyramid image transforms. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2387 – 2390, 1996. 117

[129] Peter Sloan and Michael F. Cohen. Hardware accelerated horizon mapping. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 281–286, June 2000. 44

[130] Dietrich Stoyan, Wilfrid S. Kendall, and Joseph Mecke. *Stochastic Geometry and its Applications*. Wiley, 2nd edition, 2001. 105, 124, 133, 135

[131] H. Stoyan and D. Stoyan. Simple stochastic models for the analysis of dislocation distributions. *Physica status solidi (a)*, 97(1):163–172, 1986. 135

[132] D. J. Strauss. A model for clustering. *Biometrika*, 62(2):467–475, 1976. 125, 129

[133] M. Tarini and P. Cignoni. Pinchmaps: textures with customizable discontinuities. *Computer Graphics Forum (Eurographics 2005 Conf. Issue)*, 24(3):[in press], 2005. 37, 42

[134] M. Tarini, P. Cignoni, C. Rocchini, and Roberto Scopigno. Real time, accurate, multi-featured rendering of bump mapped surfaces. *Computer Graphics Forum*, 19(3), August 2000. 43

[135] Gabriel Taubin. A signal procesesing approach to fair surface design. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 351–358. ACM SIGGRAPH, Addison Wesley, August 1995. 71

[136] D. Terzopoulos, M. Kass, and A. Witkin. Snakes: Active contour model. *International Journal of Computer Vision*, 1:321–333, 1988. 18, 19, 21, 26, 32, 268

[137] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 665–672, New York, NY, USA, 2002. ACM. 118

[138] Jack Tumblin and Prasun Choudhury. Bixels: Picture samples with sharp embedded boundaries. In *Proceedings of the 15th Eurographics Workshop on Rendering Techniques*, 2004. 37, 40, 41, 42, 61, 269

[139] Greg Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 289–298, New York, NY, USA, 1991. ACM. 115

[140] J. K. Udupa, P. K. Saha, and R. A. Lotufo. Boundary detection via dynamic programming. In *Proc. SPIE Med. Imaging*, volume 1808, pages 33–30, 1992. 17

[141] M. N. M. van Lieshout. *Markov Point Processes and Their Applications*. World Scientific Publishing Co., 2000. 124

[142] Marcelo Walter, Alain Fournier, and Daniel Menevaux. Integrating shape and pattern in mammalian models. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 317–326, New York, NY, USA, 2001. ACM. 115

[143] Lifeng Wang, Xi Wang, Xin Tong, Stephen Lin, Shimin Hu, Baining Guo, and Heung-Yeung Shum. View-dependent displacement mapping. *ACM Transactions on Graphics*, 22(3):334–339, July 2003. 44

[144] Xi Wang, Xin Tong, Stephen Lin, Shimin Hu, Baining Guo, and Heung-Yeung Shum. Generalized displacement maps. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 227–234, June 2004. 44

[145] Yiping Wang, Wencheng Wang, and Enhua Wu. Optimizing the parameters for patch-based texture synthesis. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 75–82, New York, NY, USA, 2006. ACM. 118

[146] T. Watanabe and P. Cavanagh. Texture laciness: the texture equivalent of transparency. *Perception*, 25(3):293–303, 1996. 228

[147] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00: Proceedings of the 27th annual conference*

on Computer graphics and interactive techniques, pages 479–488, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 118

[148] Klaus-Peter Wilhelm, Peter Elsner, and Enzo Berardesca. *Bioengineering of the Skin: skin surface imaging and analysis*. CRC Press, 1997. 2, 14

[149] Andrew Witkin and Michael Kass. Reaction-diffusion textures. In *Computer Graphics*, pages 299–308, 1991. 115

[150] Steven Worley. A cellular texture basis function. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294, New York, NY, USA, 1996. ACM. 115

[151] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 364–367, New York, NY, USA, 2004. ACM. 4, 118, 119, 120

[152] Y. Wu, P. Kalra, L. Moccozet, and N.M. Thalmann. Simulating wrinkles and skin aging. *The Visual Computer*, pages 183–198, 1999. 107

[153] Y. Wu, P. Kalra, and N.M. Thalmann. Simulating of static and dynamic wrinkles of skin. In *Computer Animation*, pages 90–97, 1996. 107

[154] S.S. Yu and W.H. Tsai. A new thinning algorithm for gray scale images by the relaxation technique. *Pattern recognition*, 23:1067–1076, 1990. 15

[155] Steve Zelinka and Michael Garland. Jump map-based interactive texture synthesis. *ACM Trans. Graph.*, 23(4):930–962, 2004. 118

[156] Jingdan Zhang, Kun Zhou, Luiz Velho, Baining Guo, and Heung-Yeung Shum. Synthesis of progressively variant textures on arbitrary surfaces. *ACM Transactions on Graphics*, 22(3):295–302, July 2003. 44

[157] Yan Zhang, Zhengxing Sun, and Wenhui Li. Technical section: Texture synthesis based on direction empirical mode decomposition. *Comput. Graph.*, 32(2):175–186, 2008. 118

[158] Song Chun Zhu, Ying Nian Wu, and David Mumford. Minimax entropy principle and its application to texture modeling. *Neural Comput.*, 9(8):1627–1660, 1997. 117

[159] Song Chun Zhu, Yingnian Wu, and David Mumford. Frame: Filters, random fields, and minimax entropy– towards a unified theory for texture modeling. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:686, 1996. 117

[160] D. Ziou and S. Tabbone. Edge detection techniques — an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559, 1998. 1, 13