

A study of the Intratumoural Vascular Network by Fractal Analysis, Percolation Theory and Syntactic Structure Analysis

An Investigation of Possible Image Analysis Parameters Applied to Histological Sections in Hypoxia and Angiogenesis Related Research

LARS TORE GYLAND MIKALSEN

The Biophysics Group
Department of Physics
University of Oslo

Department of Radiation Biology
Institute of Cancer Research
Rikshospitalet – Radiumhospitalet HF



Rikshospitalet – Radiumhospitalet **HF**

MASTER THESIS

May, 2007

Preface

This thesis is submitted in partial fulfilment of the requirements for the Master's Degree at the Institute of Physics, University of Oslo, Norway. The study has been performed at the Institute of Cancer Research, The Norwegian Radium Hospital, from January 2006 to May 2007.

The programs used in this thesis are written in MATLAB as a part of this work. Figures have been processed in the Gnu Image Manipulation Program, GIMP, and the thesis itself has been written in the document preparation language L^AT_EX. Figures with citations in the caption are acquired from these references, other figures, i.e. those of section 2.3 and chapters 3 and 4, are produced by the author.

I would like to thank my supervisor Dag Rune Olsen for suggesting this thesis, and for guiding me through the process. Thanks to Øyvind Bruland and Hari Dhakal for providing the CD34 tissue sections.

Thanks to Therese Seierstad, Kathrine Røe and Hong Qu for help with the tumour xenografts and contrast enhanced micro-CT. These results are not a part of the thesis, due to a redefiniton early autumn 2006. The help with these experiments was all the same of great value, and provided the data leading up to the redefinition of the thesis. Furthermore, thanks to Therese Seierstad for help with some of the early immunohistological experiments on tumour xenografts.

Finally, I would like to give thanks to God (Col 3.17), and to my family for support through all these years, and for proofreading in the final stages.

The Norwegian Radium Hospital, Oslo, May 2007

Lars Tore Gyland Mikalsen

Abstract

The purpose of this thesis is the investigation of the intratumoural network through image analysis of histological sections. Tumour vasculature is characterized by complexity, irregularities and poorly regulated growth. Fractal analysis has been used to establish that tumour vasculature has a different network architecture from that of the normal arterio-venous system or the capillary network. The vasculature is responsible for the transportation of oxygen to tumour cells, however its many pathological features results in, among others, the presence of hypoxic regions. Hypoxia is a challenge to the treatment of cancer, both through its indirect biological effects, such as a reduced progression through the cell cycle, but also through direct chemical effects. In particular, *the oxygen effect* reduces the effectiveness of radiation therapy. Furthermore, the network morphology relates to many other parameters as well, such as the angiogenic and the metastatic capability of the cancer. This raises the possibility of using image analysis, and fractal analysis in particular, to quantify different aspects of the network morphology.

The study limits itself to parameters which may be obtained from digitized images of histological sections with endothelial-specific staining. The investigated parameters are primarily obtained through fractal analysis and syntactic structure analysis. A few more parameters, such as the number of vessels, the size of the vessels, the total vascular area, and cumulative histograms of distances to the nearest vessel, were obtained directly from the images. The investigated parameters depend on both the number of vessels in the image, and the distribution of the vessels. Two particular areas have been emphasized, the first is the identification of how strongly the parameters relate to the vessel distribution, and the second is the implementation of fractal analysis on vascular cross sections.

Four different CD34-stained immunohistological sections have been analysed. They were obtained from malignant carcinomas of the breast and exhibited qualitatively different vascular patterns. A routine has been developed to segment out the vessels from the background staining before the image analysis.

The investigated fractal dimensions include the Box Counting dimension, the Sandbox dimension, the Correlation dimension, the Mass dimension and the Fourier dimension. These have been applied to images processed in three different ways. The first contained the entire vessel lumens, the second only the outer vessel wall perimeter and the last only the vessels' geometric centre of mass. In addition fractal analysis has been performed on Gabriels's Graph and the Euclidean Minimum Spanning Tree, both of which belong to the Syntactic Structure Analysis graphs. The different methods and images provided both different dimensions and different curve shapes. Some of the curves did not have any meaningful power-law scaling regions at all, however, most of them did. The Sandbox dimension in general and the mass centre images in particular, have been considered the most promising of these methods. Although it may be argued that the term *dimension* does not, in any meaningful way, relate to most of the parameters obtained through these methods, they do most certainly appear capable of differentiating various vessel distributions from each other. In addition to the fractal analysis methods, all other investigated methods have been applied to the four cases as well.

In order to identify the relationship between the parameters and the number of vessels in a particular image, two simulations have been performed. The first simulation generated the images through a uniform random distribution probability (10.000 images), while the second used a three-dimensional invasion percolation cluster to generate the vessel positions (15.560 images). The mean and standard deviations of the results at each vessel count have been investigated. Large standard deviations have been interpreted as a strong dependency on the vessel distribution. The slope of the mean, on the other hand, shows the parameters dependency on the number of vessels. The sizes of the standard deviations are considered relative to the slopes. Most of the analysis parameters showed large variations for low vascular densities. A subset of the parameters had large variations even at very high vascular densities.

In conclusion, most of the investigated parameters appear to be promising candidates for further studies. Fractal analysis may be applied to vascular cross-sections. It is, however, important to rigorously specify how the analysis is performed, as a large number of possible results may be acquired through these methods. In particular the Sandbox dimensions of the mass centre images, Gabriel's Graph, and the Euclidean Minimum Spanning Tree, at large sandbox diameters, are recommended for further study, with the possible addition of the EMST dimension at small diameters, as this requires no extra computation time. At this point in time it is not recommended to exclude any of the SSA-parameters from further studies. The next advisable step would be to perform a correlation study, comparing these parameters to other data of clinical value, related to treatment, diagnosis or prognosis.

Contents

Preface	i
Abstract	iii
1 Introduction	1
2 Theory	3
2.1 Tumour Vasculature	3
2.1.1 Normal Vascular Formation in the Fetus; Vasculogenesis and Remodeling . .	3
2.1.2 Angiogenesis	4
2.1.3 Angiogenic research	4
2.1.4 The Molecular Biology of Angiogenesis	5
2.1.5 Pathophysiological Angiogenesis in Tumours	6
2.1.6 The Characteristics of Tumour Vasculature	11
2.2 Hypoxia	20
2.2.1 Chronic Hypoxia	20
2.2.2 Acute Hypoxia	20
2.2.3 Effects of Hypoxia; Radiotherapy	21
2.3 Fractal Theory	23
2.3.1 Dimensions	23
2.3.2 Self-Similarity	26
2.3.3 Natural Fractals	26
2.3.4 Finding the Fractal Dimension of a Natural Fractal	26
2.3.5 Box-Counting Dimension	27
2.3.6 Sandbox Dimension	27
2.3.7 Fourier Dimension	28
2.3.8 Mass Dimension and Correlation Dimension	29
2.3.9 Analysing Images	31
2.3.10 Percolation Theory	32
2.4 Fractals and Cancer	37
2.4.1 Fractal Quantification of Tumour Vasculature	37
2.4.2 Analysis of Two-Dimensional Tumour Models	37
2.4.3 Analysis of Tissue Sections	39
2.4.4 Invasion Percolation Tumour Vasculature Model	44
2.4.5 Cellular Automaton Tumour Vasculature Model	46

3	Materials and Methods	51
3.1	CD-34 Image Processing	51
3.1.1	Scanning of Images	51
3.1.2	Applying Thresholds to Image Sections	51
3.2	Basic Image Statistics	56
3.3	Syntactic Structure Analysis	57
3.3.1	Voronoi Diagram	57
3.3.2	Gabriel's Graph	58
3.3.3	Euclidean Minimum Spanning Tree	58
3.4	Linear Fitting and Graphs from the Fractal Algorithms	58
3.4.1	Linear Fitting	58
3.4.2	The Graph Set-Up	60
3.4.3	Interpreting the Results	61
3.5	Accuracy of the Fractal Algorithms	62
3.5.1	Description of the Test	62
3.5.2	Evaluation of the Algorithms	63
3.6	Random Simulations	65
3.6.1	Implementation of the Simple Random Simulation	66
3.6.2	Implementation of the Percolation Simulation	66
4	Results	69
4.1	Vessel Section Simulations	69
4.1.1	Simple Random Simulations	69
4.1.2	Percolation Simulation	71
4.2	Analysis of Histological Sections	84
4.2.1	Image Statistics	84
4.2.2	Syntactic Structure Analysis	86
4.2.3	Fractal Analysis	87
4.2.4	Comparison with the Simulation Data	92
5	Discussion	97
5.1	Fractal Properties of the Vascular System	97
5.2	Histological Images	99
5.2.1	Image Magnification	99
5.2.2	Image Processing	99
5.3	The Studied Parameters	99
5.3.1	Fractal Analysis	99
5.3.2	Cumulative Histograms of the Distance to the Nearest Vessel	102
5.3.3	Syntactic Structure Analysis	102
5.4	Vessel Simulations	103
5.4.1	The Purpose of the Vessel Simulations	103
5.4.2	The Simple Random Simulation	104
5.4.3	The Percolation Simulation	104
5.4.4	Evaluating the Two Simulations	105
5.5	Relevance to Clinical Data	106
5.6	Suggestions for Further Studies	107
5.6.1	Correlation Studies	107

5.6.2	Segmentation of the Images	107
5.6.3	Fractal Analysis	107
5.6.4	Vessel Simulation	107
6	Conclusion	109
	Bibliography	111
	Appendix	114
A	Additional Figures and Tables	A-1
A.1	Tests of the Fractal Algorithms	A-1
A.2	Fractal Analysis of the Histological Images	A-1
B	Matlab Scripts and Functions	B-1
B.1	Image Processing	B-1
B.2	Syntactic Structure Analysis	B-6
B.3	Random Site Percolation	B-19
B.4	Fractal Analysis	B-27
B.5	Random and Percolation Vessel Simulations	B-41

Chapter 1

Introduction

Cancer is one of the most common causes of death in the western world today, second only to cardiovascular diseases. In spite of the enormous amounts resources invested in cancer research and treatment, the mortality rates due to cancer are expected to increase. This is because the probability of developing cancer increases with age. Although treatments are becoming more effective, the combination of the changing demographics and a decrease in deaths caused by other diseases will make cancer treatment increasingly important in the coming years. Cancer is not the name of a specific disease. Rather it is a term which covers a wide range of diseases developing in various organs throughout the body. The defining feature of a cancerous disease is the development of cells which blatantly disregard the internal rules of the body. Somatic cells divide regardless of the body's needs, developing into a tumour. The tumour is considered cancerous when it gains the ability to invade surrounding tissue.

Radiation Therapy is, next to surgery, the most common treatment modality used on cancer. At least 50 % of all patients are believed to benefit from radiation therapy, either for curative purposes or pain relief. In order to cure a cancer, all cancerous cells must be killed, even a single survivor may be enough to cause a relapse. The challenge in radiation therapy is to kill the cancer cells with as little harm done to the healthy tissue as possible. Conventional radiation therapy typically considers the tumour as a uniform target area and attempts to deliver some specific dose to as much of this area as possible, without exceeding specified dose limits in the surrounding area, with special regard to radiosensitive risk organs in the vicinity. The treatment plan represents a trade off between the tumour control probability and the normal tissue complication probability.

The radiosensitivity of a given tumour depends on many factors, one of which is the oxygen levels in the tumour (section 2.2). Oxygen increases the biological effects of radiation, conversely hypoxic cells, which are cells deprived of oxygen, will have decreased radiosensitivity and require higher doses to kill. Hypoxia reduces the effectiveness of chemotherapy, and it is known to increase the rate of malignant tumour progression and the rate of distant metastases. Most tumours have some degree of hypoxia. The hypoxic fractions are frequently about 10 to 15%, but may vary from 0 to 50% [21].

Hypoxia is caused by the vasculatures' failure to supply the entire tumour with oxygen. As a tumour grows, regions inside the tumour will soon find themselves further away from the existing vasculature than oxygen is able to diffuse. If left deprived of oxygen and nutrients long enough, these cells will die. In order to continue growing in size, new vasculature must be formed to supply the tumour. This is a process referred to as angiogenesis and is triggered by the production of endothelial growth factors in tumour cells. The networks formed in tumours are, however, quite different from

the healthy vasculature in the normal tissues, both when it comes to the structure of the network and the individual vessels (section 2.1).

The tumour networks, in particular, are far more chaotic than normal vasculature, prompting the use of fractal analysis as a tool to quantify some aspects of this complexity (section 2.4). Fractal analysis has become a powerful tool to classify complex phenomena, and a brief introduction to the subject is given in section 2.3. Based on the fractal analysis of two-dimensional tumour models, invasion percolation has been proposed as a way to model vascular networks in tumours.

The study of a full-scale three-dimensional tumour network is difficult. There are some ways to do it, but they are generally not applicable in a clinical setting. If the goal is to identify clinical parameters, relevant to the treatment of individual patients, then the study of histological data¹ is the most promising course of action.

This study is an investigation of parameters obtainable through image analysis of histological sections stained with endothelial specific markers. The investigated parameters include the number of vessels, the stained area, and the distances to the nearest vessel, as well as a long list of parameters obtainable through fractal analysis and syntactic structure analysis. In the case of fractal analysis, several different algorithms have been used and compared.

Two different simulations have been performed, generating and analysing randomly constructed images. The first uses a uniform probability distribution, and the latter the vertical bonds of a non-trapping three-dimensional bond invasion percolation cluster (section 3.6). The simulations investigate the spread in results for the different parameters as a function of the number of vessels. The purpose is to identify how strongly these parameters are related to the number of vessels (section 4.1). This is based on the idea that parameters which have little or no variance in the results, have equally little to add to the much easier obtained number of vessels-parameter.

In addition to the simulation, four histological sections stained with CD-34, an endothelial specific marker, have been analysed (section 4.2). The tumour tissues were from four human invasive carcinomas of the breast. The study of these cases provides an example of how the image analysis may be implemented in a clinical setting. In addition it serves as the source material for the study of parameters unsuited for the simulation, either due to the large quantity of images or the simplifications involved in the simulation. These parameters are primarily related to the different fractal analysis algorithms, as well as the fractal analysis of vessels represented by the area or the perimeter of the vessels, rather than only the mass centre. Fractal analysis of Gabriel's Graph and the Euclidean Minimum Spanning Tree is performed as well. The results of these four cases are compared to those of the simulations. In order to analyse the histological data, a method has been developed to remove the background colour from CD-34-stained sections (section 3.1).

The purpose of this thesis has been to investigate the possible usefulness of image analysis in hypoxia and angiogenesis related research, with an emphasis on the use of fractal analysis. It has been important to establish whether or not it is meaningful to apply fractal analysis to histological sections. Although one must be careful as to how the resulting dimensions are interpreted, many of the fractal parameters do seem quite capable of differentiating different vessel distribution patterns from each other, if implemented correctly. From the conclusions drawn in this study, very few parameters should be excluded completely, although some clear recommendations are made as to which fractal parameters and analysis approaches that are best suited for further studies. The study, does not in itself provide any judgement on the final relevance of the parameters. A correlation study, identifying how these parameters relate to other parameters of diagnostic, prognostic or therapeutic value, should be the next step towards finding the true relevance of these methods.

¹Cross sections of a tissue sample, stained with antibody markers to highlight specific molecules in the cells

Chapter 2

Theory

2.1 Tumour Vasculature

In order to gain proper understanding of how a tumour's vasculature differs from that of healthy tissue, it necessary to start with the process of *angiogenesis*, through which tumour vasculature is developed. Against this backdrop, a list of the most important pathological features which characterise tumour vasculature will be presented. The architecture of the network itself is of particular interest as it relates directly to the oxygenation of the tumour, and the formation of hypoxic regions. The image-analysis parameters investigated in this study aim at describing some aspect of this architecture, through the analysis of histological sections.

2.1.1 Normal Vascular Formation in the Fetus; Vasculogenesis and Remodeling

The process by which the initial vasculature in the embryo develops is referred to as *vasculogenesis*. Endothelial cells within previously avascular tissues differentiate from stem cells and proliferate. Merging together, these new tubes form a single primitive network. This process forms some of the major vessels in the embryo including the aorta and major veins as well as a honeycomb-like plexus connecting these.

By a process referred to as *angiogenic remodeling*, this initial network is modified by both pruning and vessel enlargement. The results are the branching patterns typical of mature vascularization. At the same time the endothelial cells integrate tightly with supporting cells and the extracellular matrix, transforming them into mature vessels.

A third process, referred to as angiogenic sprouting, is the cornerstone in the process of *angiogenesis*. This process is responsible for the vascularization of certain structures, such as the retina, the neural tube during normal development, and most new vessels in the adult. Sprouts from existing vessels vascularize nearby avascular tissue. Vessels formed by sprouting are initially immature and must develop further. Mature vessels, at least in some cases, must first be destabilized in order to allow subsequent sprouting. [44]

In figure 2.1 most of this is shown graphically according to the angiogenic model put forward in reference [44]. Further description of the model and the involved molecular signalling components are presented in section 2.1.4.

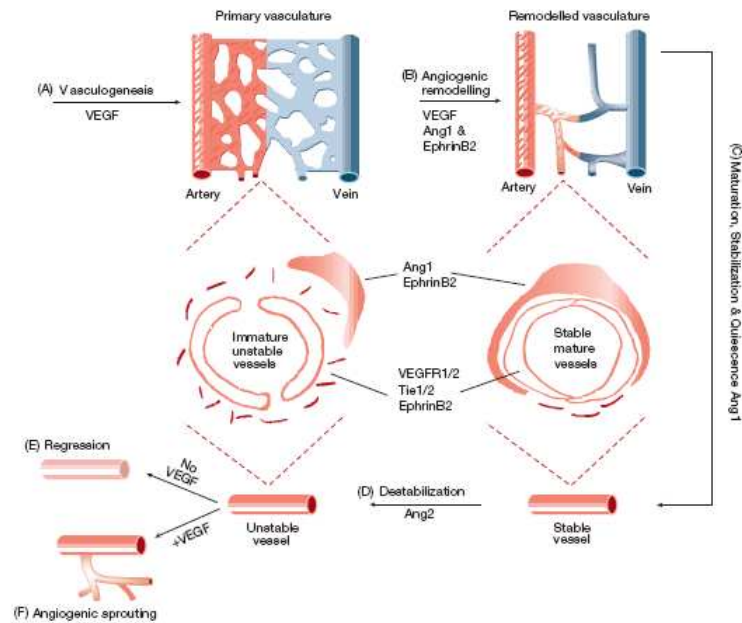


Figure 2.1: Schematic representation of important steps involved in vessel formation. These include vasculogenesis (A), angiogenic remodeling (B), stabilization and maturation (C), destabilization (D), regression (E) and sprouting (F). The role of some of the angiogenic molecules involved in these processes are shown as well. (Adapted from [44])

2.1.2 Angiogenesis

Angiogenesis is the formation and development of new blood vessels from pre-existing vessels. In the healthy body angiogenesis is responsible for the vascular remodeling during ovulation, as well as wound healing and weight gain. Apart from this, few or no changes with respect to growth, remodeling or regression of the vascular system are expected in healthy tissues. Angiogenesis is, however, involved in a long list of pathological conditions where angiogenesis either is part of the malignancy (e.g. cancer, chronic inflammatory conditions, diabetes, psoriasis, adiposity, endometriosis), or the lack thereof is a problem, i.e. where the process of angiogenesis could help cure the disease (e.g. tissue damage after reperfusion of ischemic tissue or cardiac failure).[6][19]¹

2.1.3 Angiogenic research

Much of the research of angiogenesis has been motivated by its prominent role in cancer. It has been known for almost a century that angiogenesis occurs around tumours.[6] In the early 1970s, Folkman hypothesized that angiogenesis at the tumour site was absolutely required for tumour expansion² beyond a spheroid diameter of 1-2 mm. He also postulated that inhibiting angiogenesis would inhibit tumour expansion, and that if one could get tumour vasculature to regress, then this could cause regression of the tumour mass back to the 1-2 mm spheroid diameter.[45][6]

¹See Table 1 in [6] for a more complete list.

²Tumour expansion referring to growth in tumour volume, as distinct from cell growth. Cells proliferate in avascular tumours as well, but is balanced by cell death, preventing tumour expansion.

The possibility of new ways to cure cancer spurred an intensive search for pro- and anti-angiogenic molecules, hoping to develop anti-angiogenic therapies. Today many molecules have been found and models for angiogenesis are emerging; much is, however, still unknown. Furthermore, the list of pathologies related to angiogenic research is expanding, and within oncology itself it is relevant to traditional treatment modalities as well as the anti-angiogenic treatment which motivated the research. The vascular system is the main route of cytotoxic delivery in chemotherapy. It is highly related to various modes of metabolic stress including low pO_2 , low pH, and hypoglucaemia, all of which are important to the evolution of the cancer and the current gene expression of the individual cells, but also to the outcome of treatment modalities. Low pO_2 , or hypoxia (see Section 2.2), is especially important in radiotherapy as it modifies the biological effect of radiation.

2.1.4 The Molecular Biology of Angiogenesis

The Angiogenic Switch

Physiological angiogenesis is only activated in response to ovulation, wound healing and growth. Consequently, endothelial cells have an extremely low mitotic activity in normal tissues. Only 1 in 10000 endothelial cells is in a cell division cycle at any given time [22]. Tumours, having evolved from normal tissue, start out without the ability to promote angiogenesis. In 1976 Gullino showed that cells in pre-cancerous tissue acquire angiogenic capacity on their way to becoming cancerous.[6] The onset of angiogenesis marks the transition between a dormant state (avascular phase) and the vascular phase in which the tumour grows exponentially.[35]

The ability to promote angiogenesis is not controlled by the simple presence, or lack, of growth factors, but rather the balance between various pro- and antiangiogenic molecules. Thus, it is not necessarily enough for single cancerous cells to activate genes that promote angiogenesis, but rather that enough proangiogenic factors must be produced to overcome the initial surplus of inhibitors. Likewise, not all inhibitors need to be removed from the tissue, they simply need to be suppressed by the activators. [6]

The mutations required to promote angiogenesis are usually accomplished by a subset of the cancer cells, which then induce new capillaries which converge toward the tumour. The angiogenic phenotype that triggers the vascular phase does not necessarily gain an evolutionary advantage as the new capillaries supply all nearby tumour cells regardless of phenotype. A consequence of this is that although the switch may be on in the primary tumour, small colonies of metastasized cancer cells may require a dormant phase before initiating angiogenesis and expansion. The dormant phase is not a phase in which the cancer cells are inactive, but a phase in which cell division is balanced by cell death.

A Molecular Model of Angiogenesis

As stated earlier, many molecules involved in the angiogenic signalling pathway have been found and models of angiogenesis are emerging. Five members of the Vascular Endothelial Growth Factor Family (VEGF-A/B/C/D and PlGF) have been identified, along with 3 receptors, VEGFR-1/2/3. Four angiopoietins (Ang1/2/3/4) along with one confirmed receptor (Tie2) and three of the Ephrins (Ephrin-A1/B1/B2) along with its four receptors are involved in vascular growth. The full effects of all of these are not understood, and more angiogenic molecules are expected to be found. A model based on what was known about the most important of these was presented by Yancopoulos et.al. in 2000 [44], see figure 2.1 and table 2.1.

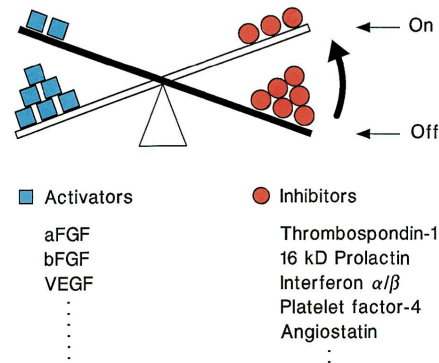


Figure 2.2: In the *Balance Hypothesis* angiogenesis will commence when the balance between activators and inhibitors tips in favour of angiogenesis. (Adapted from reference [22])

In this way a healthy angiogenic process requires the presence of a number of angiogenic factors. All of these contribute in different ways and their concentrations affect the result. Consequently, it will be very complicated and demanding to promote angiogenesis through pharmaceutical means. However, in the case of diseases like cancer where angiogenesis is undesired, at least in some of the treatment strategies, this brings hope that blocking even a few key factors may halt angiogenesis altogether. This can happen either by preventing the switch to flip, or by removing the ability of important steps to take place even if the switch is set. Furthermore, this complexity goes a long way to explain the suboptimal characteristics of tumour vasculature. This model contains only a few factors, but as the roles of the many more identified angiogenic molecules are properly understood and included, angiogenic models can only be expected to increase in complexity. (See table 2 in reference [6] for a more comprehensive list of angiogenic molecules, functions and inhibitors.)

2.1.5 Pathophysiological Angiogenesis in Tumours

Two Models of Pre-Angiogenic Tumours

The tumour growth model at the heart of Folkman's theory is a situation in which the tumour start out as an avascular mass. The developing tumour will grow at its margins pushing vessels further away from its core, causing the core to be deprived of oxygen and nutrients, and subsequently to die. The tumour growth will reach a steady state at about 1-2 mm, until the onset of angiogenesis.

This is a feasible theory for tumour (and metastasis) development, indeed it has long been thought to be the only way tumours develop. One of the reasons why this model has been left unchallenged for so long is probably the nature of many artificial tumour models used in research. By placing tumour cells in a space normally devoid of vessels, such as the subcutaneous space, the cornea pocket, the vitreous, or the tumour window, avascular tumours are forcibly created. It is also clear, however, that many natural tumours arise in this manner. [44]

In recent years, however, another way has been identified, namely by co-opting nearby vessels into the tumour mass. In this way tumours are able to expand along the vessels, growing beyond Folkman's spheroid. The vessels respond to this co-option by up-regulating Ang2, causing the vessels to destabilize and regress, and the tumour to be choked off. This leads to a secondarily avascular tumour, that upon gaining the ability to induce angiogenesis will continue its expansion in the surviving

Molecule	Receptor	Description
VEGF-A	VEGFR1/2	The most important molecule promoting vascular formation. It is required to initiate the formation of immature vessels by vasculogenesis or angiogenic sprouting. By itself it only promotes the formation of leaky, immature and unstable vessels.
Ang1	Tie2	Important for remodeling and maturation of initially immature vasculature. It also plays a role in maintaining the quiescence and stability of mature vasculature.
Ang2	Tie2	Can behave as both agonist and antagonist to Tie2 under different circumstances. Believed to provide a key destabilizing signal reverting vessels to a more plastic and tenuous state, allowing for both vascular remodeling and regression.
Ephrin-B2	EphB4	Is required for remodeling and maturation. In addition they hold an important role in distinguishing developing arterial and venous vessels. Furthermore, the presence of the arterial marker ephrin-B2 in tumour sprouting challenges the dogma that such sprouting primarily involves venous or uncommitted vessels.

Table 2.1: A description of the various roles of the molecules in Yancopoulos' model [44], cf. figure 2.1.

tumour masses. Both processes are illustrated in figure 2.3. [44],[25].

Processes of Vascular Formation in Tumours

Several qualitatively different processes of vascular formation have been found in tumours, see figure 2.4. These include angiogenic sprouting, endothelial precursor cells emigrating from the bone marrow (vasculogenesis), and intussusceptive growth [6].

Angiogenic Sprouting happens in response to local angiogenic signals. In order for sprouting to commence, the existing basement membrane and interstitial matrix are dissolved by Ang2 and proteinase mediated changes. Vessels dilate and become leaky in response to VEGF, allowing endothelial cells to escape the lumen. Endothelial proliferation, migration and assembly are stimulated by a number of molecules (VEGF, Ang1, bFGF). The sprouting vessels must then mature. A new basement membrane is formed, and the new vessels are invested with pericytes and smooth muscle cells (PDGF-BB, TGF- β 1). See figure 2.5.

Molecules that initially induce angiogenesis are subsequently processed (proteolytically) to angiogenesis inhibitors, providing a negative feedback to angiogenic processes. Most angiogenesis inhibitors promote endothelial apoptosis. Consequently, the sprouting vessels are dependent on survival factors (VEGF, Ang1).

Angiogenic sprouting is recruited from the local endothelium and is expected to give a growth of tumour mass proportional to the cube of the time for three-dimensional tumours and a quadratic growth in time for two-dimensional tumours. In other words, the mass growth is proportional to some constant factor multiplied with the time to the power of the tumour dimension, see

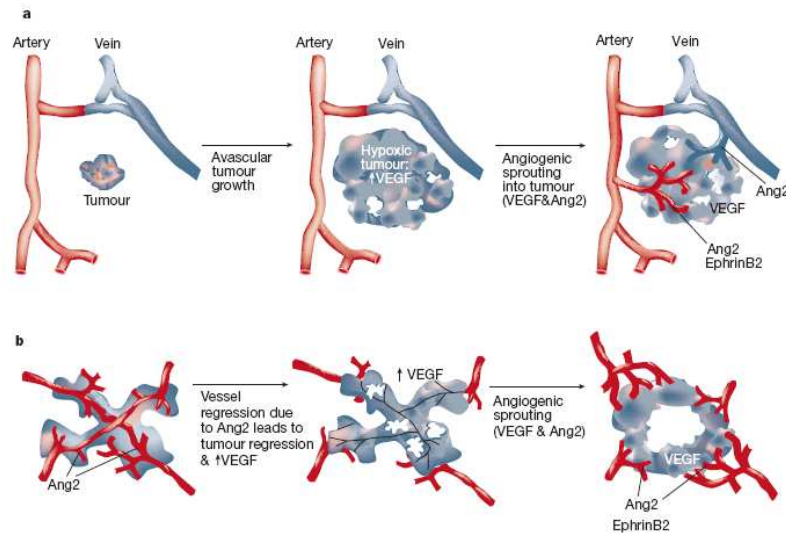


Figure 2.3: Two models of tumour angiogenesis. In **a** the tumour develops in an avascular area in accordance with Folkman's hypothesis. In **b** the tumour starts growing near vessels, these are co-opted into the tumour mass, but an upregulation of *ang2* causes vascular regression and subsequently an avascular hypoxic tumour. In the final step the onset of angiogenesis causes vascular growth and tumour expansion in both scenarios. (Reference [44])

reference [26].

$$M(t) = c \cdot t^D$$

The article did not consider the fractal characteristics of the vascular network. To reflect this, it seems reasonable to suggest the replacing of the topological dimension by the fractal dimension³ in these models.

Vasculogenesis, vascular formation from stem cells, is mediated by endothelial precursor cells (EPC) or angioblasts circulating in the blood stream. For a long time all identified postnatal angiogenic processes were due to proliferation and sprouting of differentiated endothelial cells, no postnatal vasculogenesis had been observed. In 1997 Ashara et.al. published the first paper presenting clear evidence of postnatal vasculogenesis [1]. These cells have been showed to have the ability to form endothelial colonies *in vitro*. [18]

The extent to which vasculogenesis contribute to tumour vessel formation is somewhat disputed. Conflicting results have been found in different studies and the role of tumour vasculogenesis remains unclear.

The difference between vasculogenesis and angiogenesis is, however, more than a semantic one. Not only can the molecular signalling paths be expected to be different, but also the growth processes of the vasculature and the tumour. The production of vasculature by vasculogenesis is limited by the production of EPCs in the bone marrow. This process is thus expected to give a linear growth in time of tumour mass, preceded by a short period of faster growth consuming the initial EPC buffer. [26]

³See section 2.3.1 for a description of fractal dimensions.

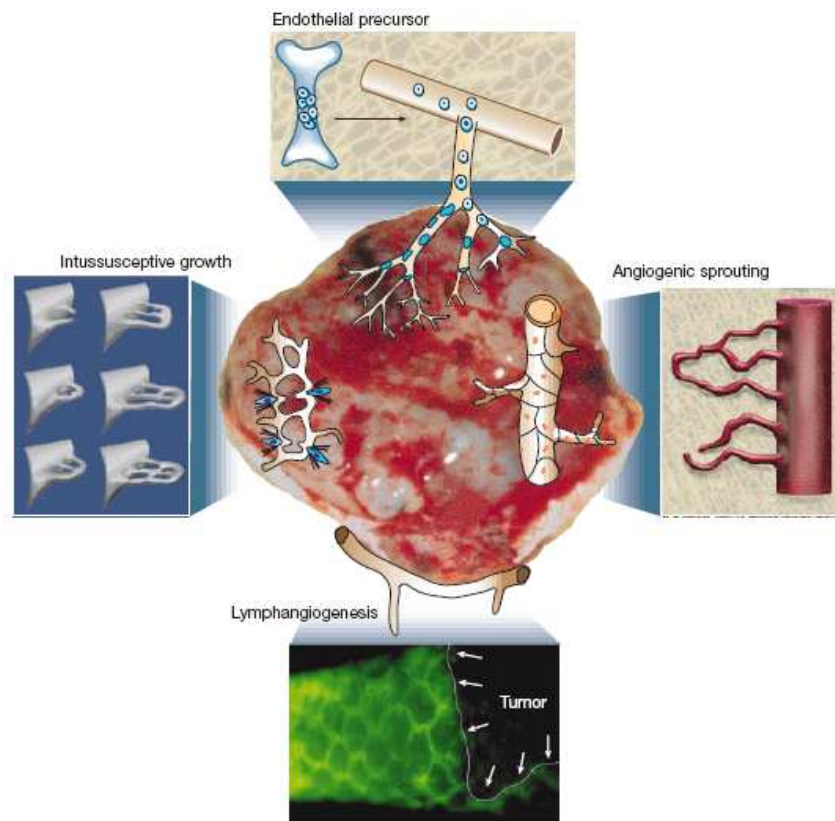


Figure 2.4: Cellular mechanisms of (lymph) angiogenesis in tumours. Vessels are formed in tumours by several mechanisms: (1) the host vascular network expands into the tumour by forming sprouts or bridges (angiogenesis); (2) interstitial tumour tissue columns are into the lumen of pre-existing vessels (intussusception); and (3) endothelial precursor cells, angioblasts, are recruited from the bone marrow into tumours and contribute to the endothelial lining of the vessels (vasculogenesis); Lymphatic vessels near the border of the tumour drain of interstitial fluid and may provide a pathway for metastasizing tumour cells. (Reference [6].)

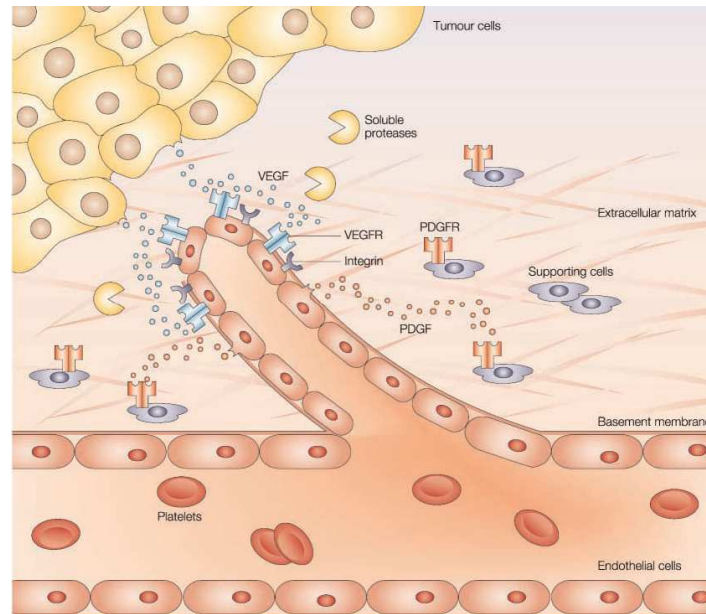


Figure 2.5: A simplified overview of tumour angiogenesis. Pro-angiogenic molecules are released from the tumour and diffuse to a nearby vessel. Upon activating the angiogenic switch endothelial cells start proliferating. The basement membrane and the extracellular matrix are weakened and integrin molecules help pull the sprouting vessel forward. A new basement membrane is then formed and supporting cells are attracted to stabilize the new vessel. (Reference [9])

Intussusceptive Growth is a process in which growth and remodeling is caused by columns of tissue partitioning the vessel lumen. This gives rise to two different processes. Inside the tumour, loop formation and remodeling give rise to new vasculature. Outside the tumour, segmentation expands and remodels the pre-existing network. Loop formation by intussusceptive growth may also be combined with sprouts superimposed on the loops, indeed this appears to be the case in the large majority of the loop systems studied by Patan et.al. [34].

Frequent remodeling in tumours by intussusceptive growth causes network architecture changes on a time scale of minutes. This might explain, or at least be one of the processes involved in causing, intermittent blood flow in tumours. [34]

Lymphangiogenesis, the development of new lymphatics, is not found in any manner comparable to that of angiogenesis in solid tumours. Indeed, hardly any lymphatic vessels are found at all; this in spite of the fact that both lymphangiogenic molecules (VEGF-C) and endothelial cells bearing their receptor are found inside tumours. Furthermore, the lymph vessels which were initially there disappear (no co-option). One hypothesis for this is that the lymph vessels collapse under the pressure of the growing tumour. Tumour cells grown as spheroids in vitro have been found to generate a pressure of 45–120 mmHg. Blood vessels are under the same stress in a tumour, but they are connected to the high-pressure arterial blood supply.

Although few lymphatic vessels can be found inside solid tumours, there are evidence of enlarged lymph vessels at the periphery of the tumour. These drain off interstitial fluid from the tumour and provide a functional network for metastasizing cells.[30]

Genetic Regulation in Endothelial Cells

The genetic expressions of endothelial cells in tumours are, although qualitatively different from normal endothelium, very similar to that of cells involved in wound healing and angiogenesis of *corpus luteum*⁴ [10]. The characteristics of tumour vasculature are, however, very different from that of vasculature developed by physiological angiogenesis. The differences that causes pathological conditions of the vascular network can therefore be assumed to be independent of the endothelial cells' gene regulation.[34]

2.1.6 The Characteristics of Tumour Vasculature

Tumour vasculature exhibits a broad range of pathological features not found in healthy vasculature. Not all of them can be expected to be found in a specific tumour, and some features are more expressed in certain types of cancers than in others. An expressed difference, when compared to normal vasculature, is, however, the rule rather than the exception. These differences relate not only to the makeup of single vessels, but also to the morphology⁵ of the network. One consequence of this is that it is meaningless to categorize tumour vessels as arterioles, capillaries, or venules. They simply lack the structural characteristics of normal vasculature that make these terms meaningful. [23]

The Characteristics of Individual Vessels

The vessels themselves exhibit several pathological features that reduce their functionality and/or increases the tumours' metastatic potential.

Increased Permeability: Tumour vessel walls are known to have a high permeability causing blood to leak. In tumours blood flow is not restricted to the vessels, but can happen in the interstitial⁶ space as well [8]. Macromolecular vessel leakiness correlates closely with the histological tumour grade [11].

Defective Endothelial Cells: In healthy vasculature the vessels are lined with a monolayer of flat endothelial cells, closely aligned with a smooth, slightly raised contour, see figure 2.6.A. In a MCA-VI mammary carcinoma, however, the endothelium exhibited a range of pathological features, see figure 2.6.B-E. These cells are thick and the smooth cell borders have been replaced by irregular edges. Where they once had a slightly raised contour, some cell borders are no longer visible. The monolayer is defective and cells have overlapping regions. Two neighbour cells may even be on top of one another in a different order at different places. Cells are found with multiple cell projections, some spanning along the wall, others across the lumen. Between cells intercellular openings are found and even transcellular holes through the cells, see figure 2.7. These openings and holes could explain the increased permeability of tumour vasculature. [23]

Blood Lakes: In some tumours blood will leak out of the vascular systems and gather up in pools known as *blood lakes*. These blood lakes are surrounded by tumour cells, not endothelia. They

⁴*Corpus luteum*, a ductless gland developed within the ovary by the reorganization of a Graafian follicle following ovulation. [Dictionary.com, November 22, 2006]

⁵Morphology, the branch of biology dealing with the form and structure of organisms. [Dictionary.com, November 23, 2006]

⁶Interstitial, *Anatomy*. situated between the cells of a structure or part: interstitial tissue. [Dictionary.com, November 23, 2006]

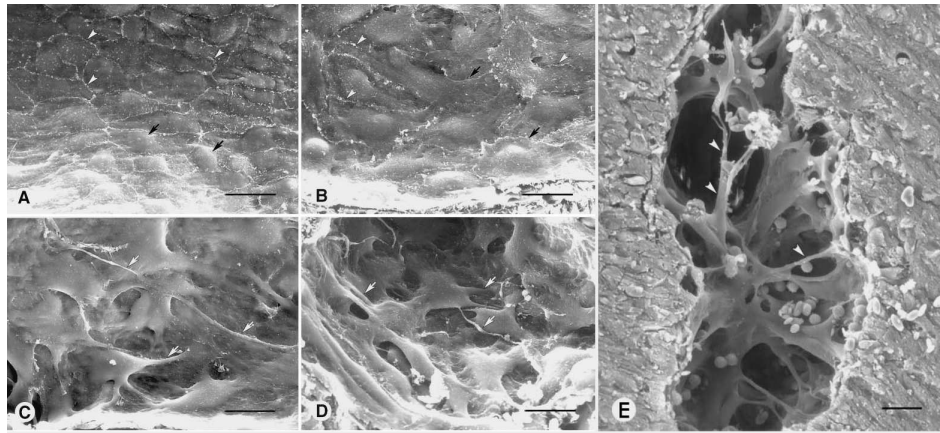


Figure 2.6: Endothelial defects in tumour vasculature. Scanning electron microscopy images of the luminal surface of endothelial cells in a normal mouse mammary gland compared to those in MCA-IV tumours. **A:** The endothelial cells in this normal venule are flat, with the exception of the region around the nucleus (arrows), and have a similar size and shape. The cells form a monolayer and the borders between individual cells (arrowheads) show very little overlap. **B:** These cells, in a tumour vessel, are irregular and overlap one another (arrows). Some of the cell borders are clearly visible (arrowheads). **C:** and **D:** More severely deformed and branched cells in a tumour. In addition to being abnormally thick, the cells overlap one another and do not have a normal connection with other cells. They do, however, have multiple cell projections (arrows) alongside the vessel walls. **E:** These abnormal lining cells (arrowheads) partition the lumen of a tumour vessel with multiple cell projections. The scale bar length represents 15 μm. (Adapted from reference [23])

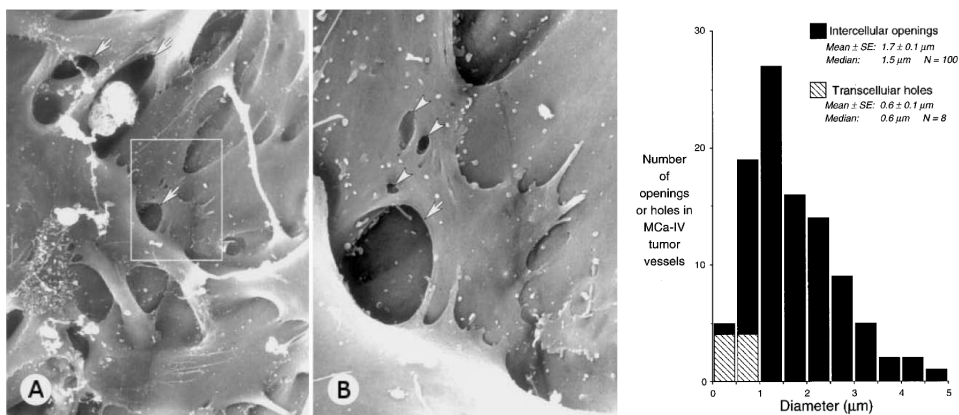


Figure 2.7: Openings in the endothelial layer. The figure shows multiple intercellular openings (arrows) or transcellular holes (arrowheads) in MCA-IV tumour vessels. **B** is an enlargement of the box in **A**. The histogram shows the distribution of openings and hole sizes of 100 openings and the holes found in the same vessels (Adapted from reference [23]).

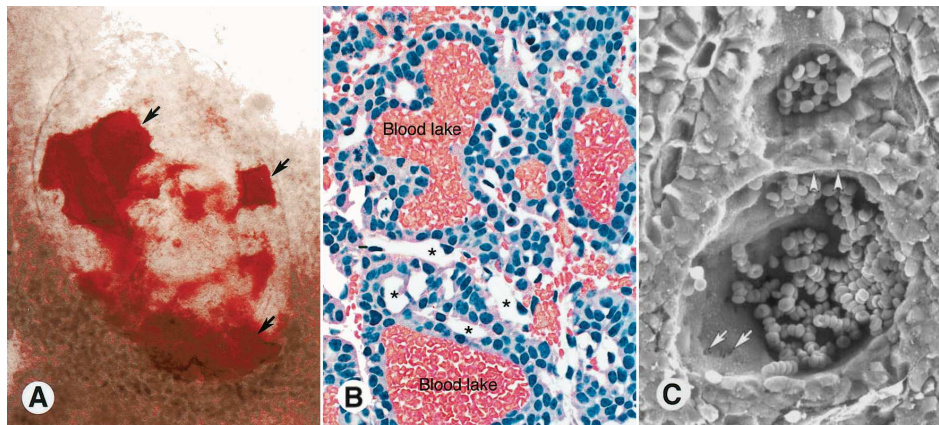


Figure 2.8: Blood lakes in pancreatic islet cell tumours of transgenic RIP-Tag2 mice. **A:** Brightfield micrograph of the whole mount of a small (~ 1 mm) RIP-Tag2 tumour containing blood lakes (arrows, red areas). The remainder of the blood has been washed out by vascular perfusion of fixative. **B:** Histological section (hematoxylin-and-eosin-stained) of RIP-Tag tumour containing blood lakes. Tumour vessels (black asterisks) are much smaller than the lakes and are emptied of blood by perfusion of fixative. **C:** Scanning electron microscope image of an extravascular blood lake which contains extravasated erythrocytes, lined by tumour cells (arrowheads) and with multiple small holes between the tumour cells (arrows). (Adapted from [23])

do not appear to be in direct contact with the vascular system as the erythrocytes⁷ in these lakes are not washed out by perfusion of fixative. Blood vessels in MCa-IV mouse mammary carcinomas are known for being unusually leaky and were used by H. Hashizume et.al. to make images of blood lakes, see figure 2.8, [23].

Mosaic Vessels: Tumour cells have been found in the lining of some tumour vessels, known as mosaic vessels. In a colon carcinoma xenograft model, Chang et.al. [7] found that about 4% of the total vascular surface area consists of cancer cells, see figure 2.9. Only 15% of the vessels were mosaic, but in these vessels approximately 25% of the perimeter consisted of cancer cells.

Several pathways by which a vessel could develop into a mosaic vessel were suggested. The one most consistent with their data, is that the endothelial cells originally lining the vessel wall are shed, consequently exposing the underlying tumour cells.

Mosaic vessels can be expected to contribute to a tumour's metastatic potential by facilitating easy access points into the vascular system. Mosaic vessels' contribution towards vessel permeability, however, remains unclear as the spore sizes identified in this study were much smaller than the areas exposed to cancer cells.

Vascular Mimicry: A phenomenon in which tumour cells develop a phenotype capable of forming vascular-like systems without endothelial cells has been labelled vasculogenic mimicry [32]. This process is distinct from that of mosaic vessels, as opposed to being a more extreme expression of the same [7].

Channels have been found forming *in vitro* cultures, obviously without the possibility of endothelial influence, see figure 2.10. In tumour xenografts similar channels have been found connected to endothelial vasculature, but without endothelial staining patterns (CD31, CD105).

⁷Erythrocyte: Red blood cell.

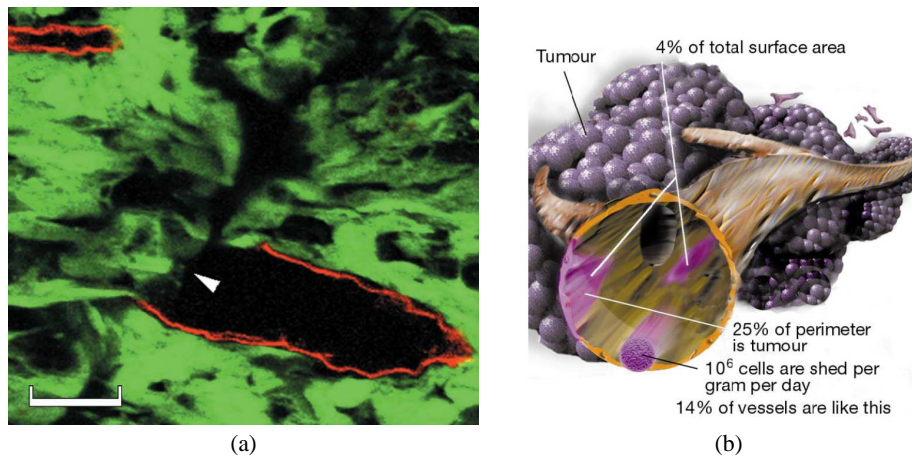


Figure 2.9: Mosaic vessels in tumours. **(a)** Cancer cells (green fluorescence) are directly exposed to the vessel lumen, note the lack of endothelial cells (red fluorescence) at the arrowhead. The gap is about $20\mu\text{m}$ long. This is referred to as mosaic vessels. (Adapted from [7]). **(b)** Quantification of mosaic vessels. In colon carcinoma $\sim 4\%$ of the total vascular surface area consists of cancer cells. If each cell intravasate in 2 days, a total number of about 10^6 cells will be shed per day per gram of tumour (Adapted from [6]).

The channels did, however, exhibit staining to the vascular-associated cell marker laminin, indicating their vascular function, see figure 2.11.B-C. In highly aggressive ovarian cancer, cells were found to form tumour cell-lined vessels, figure 2.11.A. These tumours showed minimal or no signs of necrosis. Less aggressive ovarian tumours with no sign of vascular mimicry, on the other hand, had necrosis, see figure 2.12. Patients with tumour-cell lined vasculature had a shorter overall survival [38]. Mind no such correlation was found for pT3 and pT4 cutaneous melanoma in a study by Massi et.al. [33]. The correlation cannot be assumed to be valid in general.

The formation of these fluid-conducting channels is not an angiogenic event as they do not arise from pre-existing vessels. Nor can the process be described as vasculogenesis, the channels formed, although developed *de novo*, are not blood vessels. Vascular mimicry is, strictly speaking, not a feature of the vascular system as it is not a part of it. However, although it is architecturally different, it does transport plasma and possibly red blood cells [14]. From a functional viewpoint, these systems can be regarded as extensions of the supplying vasculature. Furthermore, these systems facilitate a pathway for tumour growth without invoking the angiogenic switch, although this process may possibly require a similar switch on its own.

Morphological Characteristics

The morphological characteristics relate to the architecture of the network. This includes e.g. intervessel and interbranch distances, branching angles, vessel diameters and the network hierarchy⁸. The structure of the network, including the radiuses of the vessels, are responsible for the geometrical flow-resistance of the network. Changes in the morphology is consequently capable of changing both flow rates and pathways in local areas. This is the cause of transient (acute) hypoxia (cf. section 2.2.2). Chronic hypoxia is the result of the network's failure to supply an entire part of the tumour volume

⁸Network hierarchy: The vascular systems' functional hierarchy of arteries, arterioles, capillaries, venules and veins.

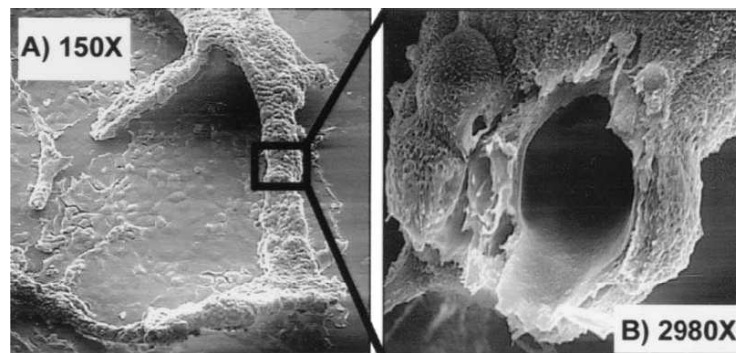


Figure 2.10: Vascular mimicry. Scanning electron micrographs of ovarian cancer cell cultures grown on three-dimensional collagen I matrices. Tubular profiles are evident in the low magnification image to the left (A). When fractured in preparation, the tubes were shown to be hollow and lined by flattened cancer cells (B). (Reference [39])

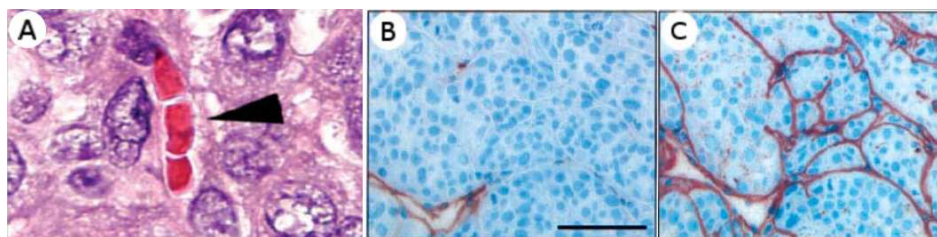


Figure 2.11: Vascular mimicry. **A:** Tumour cell-lined (pink) vasculature (red) from a H&E histological section of an invasive ovarian cancer (Adapted from reference [38]). **B** and **C:** Serial sections of xenografted Mel157 uveal melanoma cells (blue). **B** is stained for endothelium (red), **C** is stained for laminin (red). This indicates the presence of channels outside the endothelium (Adapted from reference [14]).

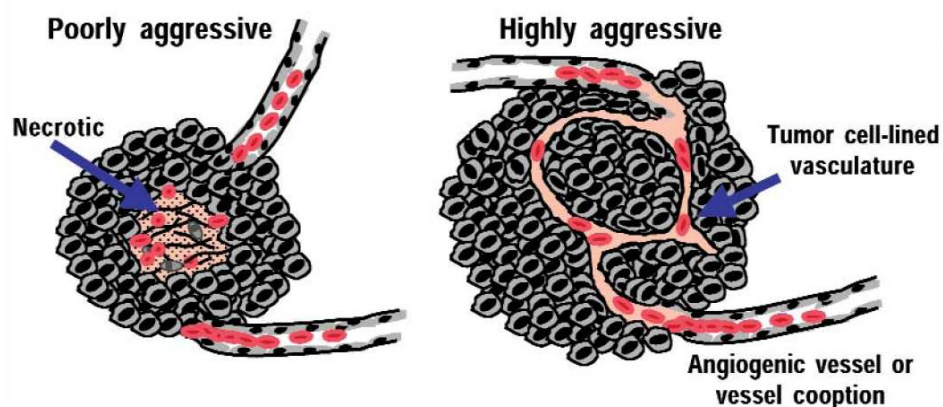


Figure 2.12: Vascular mimicry. Tumours with tumour cell-lined vasculature are more aggressive and show little sign of necrosis in human ovarian carcinoma (copied from [38]).

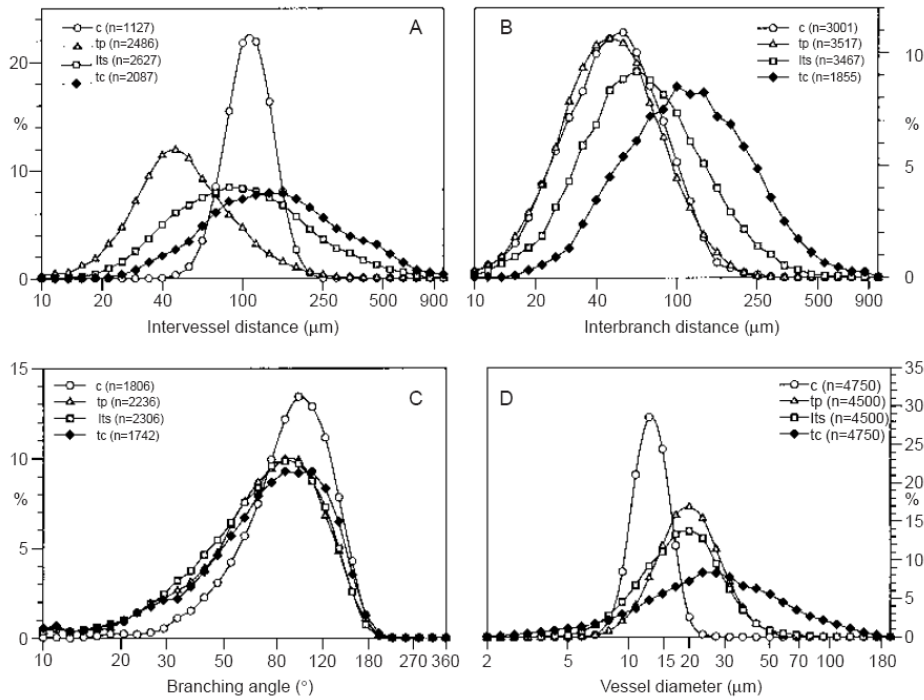


Figure 2.13: Logarithmic distributions of the intervessel distances (A), interbranch distances (B), branching angles (C), and vessel diameters (D). The data were quantified from image analysis of 3D scanning electron micrographs of corrosion casts of 20 colorectal adenocarcinomas and control mucosa. **c** = control mucosa, **tp** = tumour periphery, **lts** = luminal tumour surface, **tc** = tumour centre, (Reference [28]).

(section 2.2.1).

Corrosion cast studies suggest that the vascular network develops in a characteristic way determined by the tumour cells [27]. Although endothelial growth factors, (VEGF), correlates to the amount of new vessel formation the architecture is tumour-type specific. This architecture has been found to be qualitatively and quantitatively the same for all individual tumours, irrespective of localization and grading, in a study on colorectal cancer. Pre-cancerous lesions show architectures similar to those of invasive carcinomas, however the variability between individual adenomas is by far higher than between individual carcinomas. Metastatic tumours only display different architectures within hot spots [28].

Konerding et. al. investigated the intervessel distances, interbranch distances, branching angle and vessel diameter, see figure 2.13. Three qualitatively different areas of the tumour were investigated separately, namely the tumour periphery, luminal surface and centre. All four parameters were generally different from the control in all three areas, with the exception of the interbranch distance at the tumour periphery. Furthermore, with the exception of the branching angle, the three areas also differed from each other. The vessel diameters are in general increased and the mean branching angle is decreased. The intervessel distance is decreased at the periphery and increased in the centre. The interbranch distance is the same for the control and the tumour periphery, but increased for both the luminal surface and the tumour centre.

Normal vasculature from a skeletal muscle and subserosal capillaries of the gut are shown in figure 2.14. These architectures are quite different, yet the vessel diameters are essentially the same

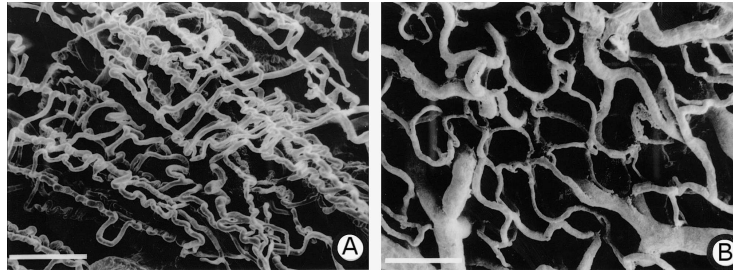


Figure 2.14: Scanning electron micrographs of corrosion cast specimens of the vascular network in a skeletal muscle (A) and of subserosal capillaries of the gut (B) draining into venules. The network morphologies show few similarities. Bars = 100 μm . (Reference [27])

with respect to both mean and variation. The branching angles are essentially the same as well, in spite of the dramatic difference in appearance between these networks. The intervessel and inter-branch distances, however, are different. In figure 2.15 these data are compared quantitatively for two murine carcinomas (CaX, CaNT), a slow growing murine sarcoma (SaS) and a human endometrial adenocarcinoma xenograft.

In figure 2.16 scanning electroscopie micrographs of the normal mucosal plexus, an adenocarcinoma,⁹ and an adenoma,¹⁰ are shown. The branching pattern of the normal *mucosal plexus* has disappeared in both the adenomas and carcinomas. The vascular densities, in general, decline from the tumour periphery to the tumour centre. In areas with low vascular density, numerous vessel compressions and elongated vessel segments are seen. A low or missing vessel hierarchy, as well as blind-ending vessels, is observed in all samples.

⁹adenocarcinoma, *pathology*: A malignant tumour arising from secretory epithelium. [Dictionary.com, 14.12.06]

¹⁰adenoma, *pathology*: A benign tumour originating in a secretory gland. [Dictionary.com, 12.14.06]

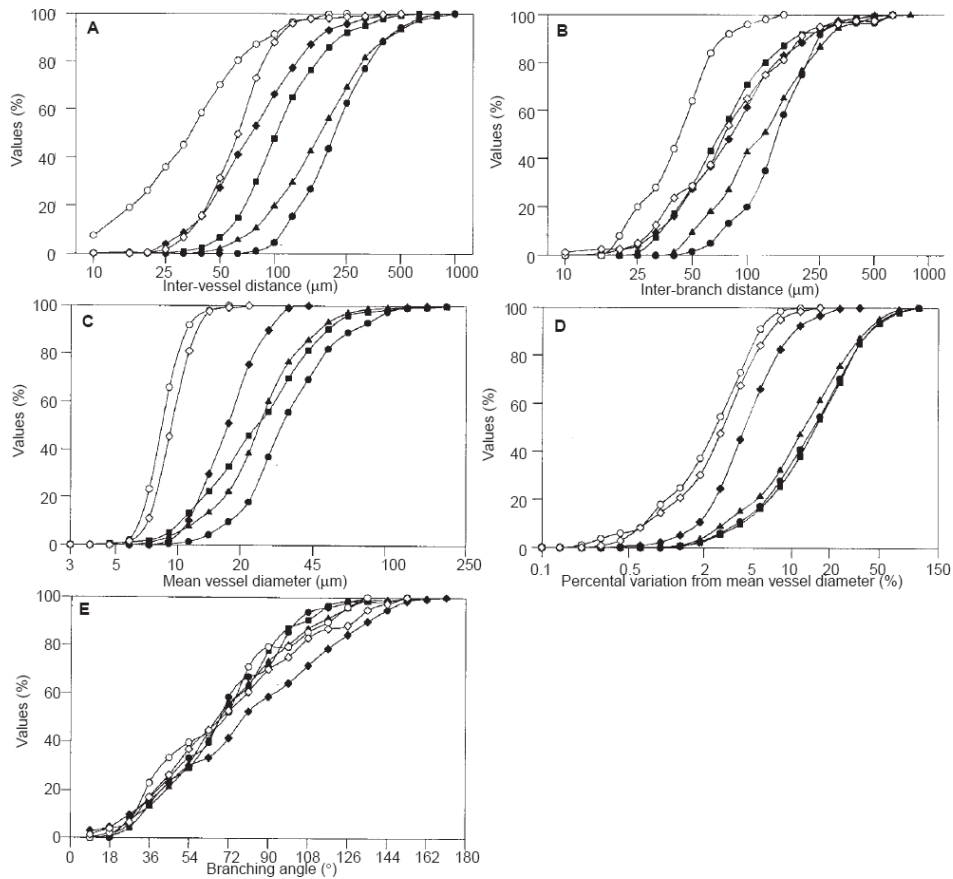


Figure 2.15: Cumulative frequency distribution plots (semi-logarithmic) of different parameters describing the vascular architecture in different tumours and normal tissues. The following tumours were used: \bullet = CaX, \blacktriangle = CaNT, \blacksquare = SaS, \blacklozenge = HEC-1B. And for comparison \circ = musculature, \diamond = subserosal gut vessels (see figure 2.14). The parameters are the inter-vessel distances, (A); inter-branch distances, (B); vessel diameters, (C); variation in percent of vessel diameters, (D); and the branching angles, (E). Different distribution patterns are observed for most of the parameters, either as a change in slope or a horizontal shift of the 50% value. With the exception of the branching angles and the inter-branch distance of the gut serosa, the tumour vessel value distributions are clearly different from the normal tissue distributions. (Reference [27])

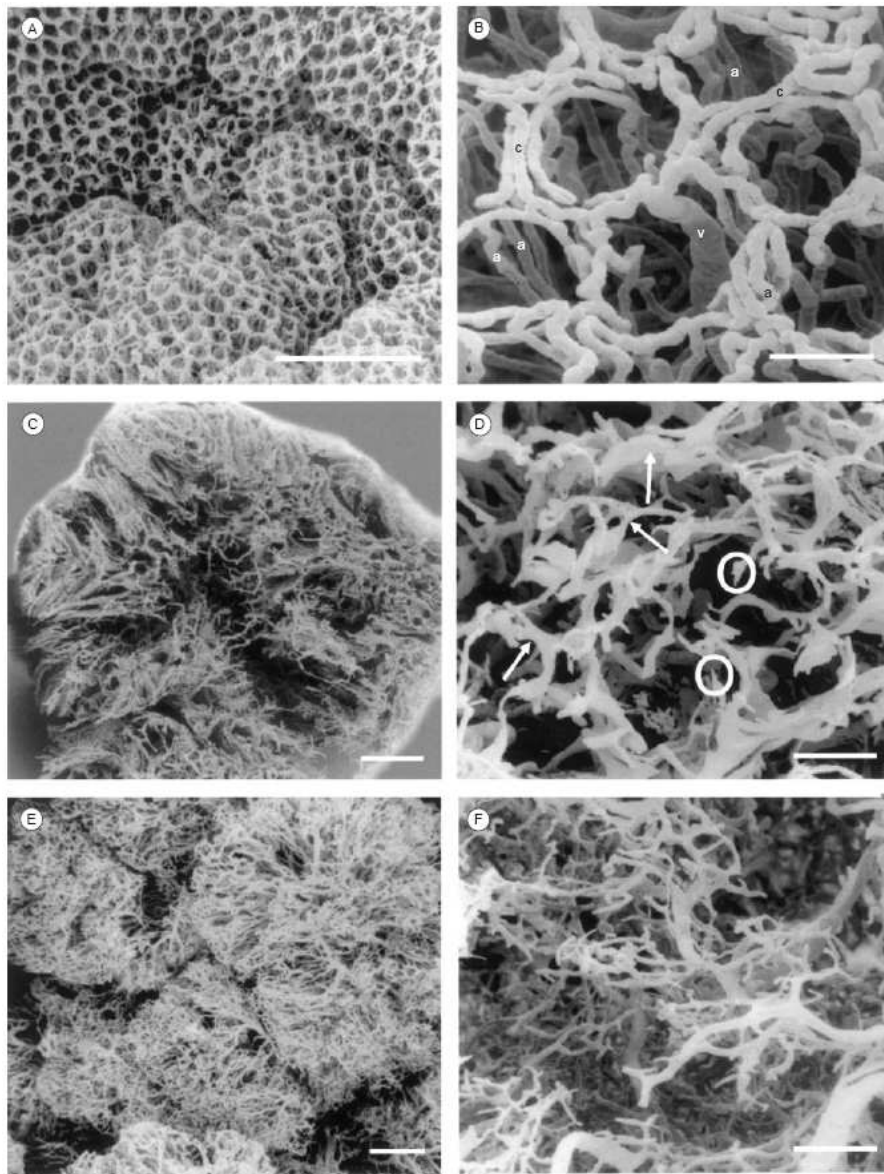


Figure 2.16: Scanning electron micrographs of corrosion casts of the colorectal vasculature in normal mucosa (A, B), carcinoma (C, D) and adenoma (E, F). The characteristic honeycomb resembling pattern around the crypts in the normal mucosa (A), with ascending arterioles supplying (B,a) and descending veins (B,v) draining the network, is very different from the vascular networks formed by angiogenesis in the carcinoma and adenomas. The carcinoma have highly expressed variations in vascular density (C), numerous blind ending vessels (D,circle) and great variations in vessel diameter within short distances (D,arrows). Furthermore, there is no expressed vascular hierarchy, that is a distinction between capillaries, veins and arterioles. The adenoma (E) has a high vascular density on the luminal surface forming from vascular networks in the centre (F). Again, there is a loss of vascular hierarchy. Bars in A, C, E = 1 mm, bars in B, D, F = 100 μm . (Reference [28])

2.2 Hypoxia

Hypoxia can loosely be described as oxygen deficient tissue. It is an important parameter for the description of tumours for at least two reasons. Oxygen was discovered to increase the biological effect of ionizing radiation as early as in 1912¹¹. Although many other substances that modify radiosensitivity have been discovered, none are as powerful as oxygen. [21] Furthermore, the presence of hypoxia will have consequences for a cell's metabolism and cell cycle progress, indeed cell division may be halted altogether until oxygen is resupplied. Oxygen therefore has a profound effect, especially on radiotherapy, but also on all treatment forms utilizing the increased cell cycle progression of cancer cells. Furthermore, hypoxia has been found to correlate with malignant tumour progression.

It is useful to differentiate between two types of hypoxia, chronic and acute. The difference is in part a functional one, acute hypoxic tissue will in time be reoxygenated without treatment, chronic will not. This offers two different challenges to treatment and warrants the differentiation. Furthermore, the two different types are created by different biological mechanisms. This difference is especially important when it comes to treatments trying to circumvent hypoxia or at least diminish its effects.

2.2.1 Chronic Hypoxia

Chronic hypoxia is found in cells too far away from any vessel capable of carrying flow, either because there are no vessels close enough or because these vessels do not carry sufficient flow to oxygenize the tissue. Both cases are results of the suboptimal arteriovenous system found in most tumours, where a branch may be so tortuous that the increased geometrical resistance forces the flow to go elsewhere. Tissue completely bereft of oxygen will in time die and form necrotic tissue. If necrotic tissue in a specific tumour has died from oxygen deprivation, there will always be a layer of hypoxic tissue between the necrotic and normoxic regions, where normoxic is defined as normally oxygenized tissue. For this reason, although the dead tumour cells are of no consequence to the patient, necrosis indicates a more severe diagnosis. This is not only due to the complicating effects of hypoxia on treatment, but also due to its role in malignant tumour progression. In summary, chronic hypoxia is diffusion limited, ie. the hypoxic tissue is out of oxygens diffusion range, about $70\mu\text{m}$, with respect to the closest vessel. [21]

2.2.2 Acute Hypoxia

Acute hypoxia is a result of blood carrying vessels being temporarily occluded, collapsed or otherwise incapable of distributing oxygen to the surrounding tissue. Tumour cells do not obey by the normal cell cycle regulation mechanisms and grow uncontrollably, this is one of the defining features of a malignant tumour. This unrestrained growth can cause the pressure in parts of the tumour to rise and become larger than the local blood pressure, resulting in the collapse of these vessels. A tumour is a dynamic system and subsequent changes in the pressure may cause vessels to open up again. Although a long-term closing is conceivable, such a vessel would be incapable of carrying flow and result in chronic hypoxia.

Another mechanism closing vessels is blockage. Normal capillaries are very narrow, only a single column of red blood cells are able to pass through. In a tumour, capillaries may be so small that they are roughly the same size as the red blood cells. Cells, either red blood cells or tumour cells which have broken into the vascular system, may then get stuck and later break free again. This causes a

¹¹It was discovered by Swartz in Germany, but only became known in the English speaking world after Mottram's research in the 1930s

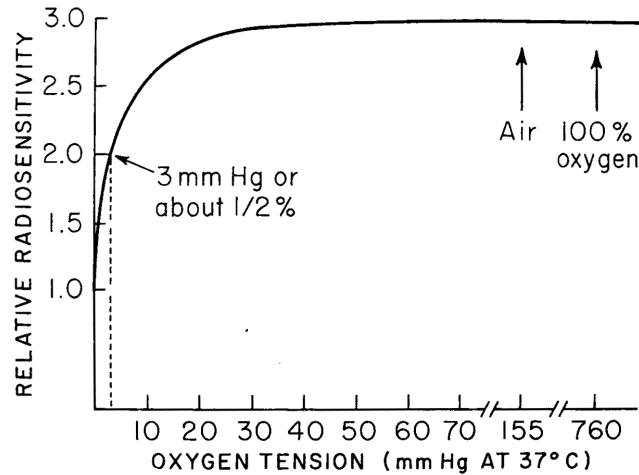


Figure 2.17: The dependence of radiosensitivity on oxygen concentration. Anoxic conditions are set to 1 and different oxygen levels are compared to this. (Reference [21])

transient hypoxia in the supplied tissue. Even capillaries normally large enough can be constricted and clog, or hamper flow due to spontaneous vasomotoric activity. [12]

Dewhirst et.al investigated acute hypoxia and found several different types in his study. [12] The first was usually confined to single vessels and was characterized by an unstable flow magnitude and direction. In this type total vascular stasis occurs for a few seconds at a time. The second observed type affected groups of vessels in a cyclic pattern with intervals ranging from 20–60 minutes. Total stasis did not occur, but there were large fluctuations in the red cell flow rate, and corresponding fluctuations in the vascular oxygen content. Finally, 9% of the investigated vessels had plasma flow, but very low or absent red blood cell flux over periods of many minutes.

In summary, acute hypoxia is perfusion limited. Although a vascular system is present, for transient periods of time, it does not supply the surrounding tissue with oxygen.

2.2.3 Effects of Hypoxia; Radiotherapy

The Oxygen Effect

Oxygen's ability to increase the biological damage of radiation is known as *the oxygen effect*. For the sake of quantification, the *oxygen enhancement ratio*, OER, has been defined as the ratio between the doses needed to produce the same biological effect in anoxic and oxic environments respectively.

Because of the way the OER is defined, it does not depend on the amount of oxygen present. The biological effect, however, does, and it has been measured for yeast, bacteria and mammalian cells in culture. Figure 2.17 shows an illustration of the results of these experiments. The curve has, initially, a very steep climb indicating how little oxygen is needed to produce the effect. At 3 mmHg half of the effect is achieved and at 30 mmHg little more is to be gained. Under atmospheric pressure this corresponds to 0.5 and 5 % of a 100 % oxygenated environment, or 1.9 and 19 % of the oxygen tension in air, respectively.

The most important hypothesis as to the cause of this effect is the *oxygen fixation hypothesis*. This theory states that oxygen reacts with the free radicals formed from an ionization and fixes the damage,

thereby preventing radicals to recombine. The oxygen needs to be present during the irradiation, or within the lifetime of the free radicals (10^{-5} s). The oxygen effect can be said to be a direct chemical effect, rather than an indirect biological one. It is not the result of reduced cell cycle progression or gene expressions.

The OER has been found to have a dependence on both doses and the *linear energy transfer*, LET, of the radiation. At high doses the OER is about 3, but at doses lower than about 2 Gy, the OER is only about 2. The OER decreases as the LET increases, and at a LET of about $160 \text{ keV}/\mu\text{m}$ it reaches unity, i.e. no oxygen effect.

Effects of Hypoxia; Chemotherapy

There is no chemical reaction, similar to that of the *oxygen fixation hypothesis*, that makes hypoxic cells less respondent to cytostatica, and oxygen concentrations does not seem to affect cells *in vitro*. *In vivo* hypoxia will, however, still reduce the effect of most chemotherapies. Many cytostatica take advantage of the cancer cells increased mitotic activity. Hypoxia reduces cell cycle progression, or in extreme cases halts all together, reducing the effect of such treatments. Furthermore, hypoxia is caused by the insufficient transportation of oxygen in the vascular system. The same mechanisms will interfere with the delivery of other blood carried agents, including both nutrition and drugs. Anticancer drugs are usually highly reactive and will in many cases have diffusion ranges shorter than that of oxygen, effectively causing the effected area to be even larger than the hypoxic. In this way, hypoxia increases cytotoxic drug resistance through indirect biological effects, and correlates with drug resistance obtained through other mechanisms. [5]

Effects of Hypoxia; Malignant Tumour Progression

Hypoxia has been shown to destabilize the genome, resulting in an increased mutation rate which increases the survival advantage of cells in adverse conditions. Cells with reduced apoptotic activity, possibly through the inactivation of tumour-suppressor genes, such as p53, or overexpression of anti-apoptotic genes such as bcl-2, will have a survival advantage in low-oxygen environments. [21] In particular hypoxia is known to influence the expression of the *hypoxia-inducible factor 1*, HIF-1. This molecule affect the expression of a large number of proteins, including the vascular endothelial growth factor, VEGF, affecting angiogenesis, molecules promoting metabolic adaptation, and genes that play a role in tumour progression, such as proliferation, invasion, and metastasis promoting genes, thereby contributing to tumour aggressiveness. [40]

Clinical studies, on advanced carcinoma of the cervix, show that local control in patients treated with radiotherapy or surgery was easier obtained in patients with oxygen probe measurements higher than 10 mm Hg, compared to those with lower pO_2 . Furthermore, the frequency of distant metastases, in patients receiving radiotherapy for soft-tissue sarcoma, were found to be 70% in patients with pO_2 s less than 10mm Hg, compared to 35% for patients with higher oxygen tension. [21]

2.3 Fractal Theory

Fractals, a term coined by Benoit Mandelbrot in his 1967 paper *How long is the coast of Britain?*[31], has received ever growing attention over the last decades. Although the development of fractal theory and dimensions started in the late nineteenth century, it is the development of computers capable of visualising fractal sets that has made this field so popular, not only in the sciences, but also in the public sphere. By the public, simply because they are pretty to look at, and in the sciences, because of the way fractal theory can be used either to model complex sets or quantitatively measuring certain aspects of a set's complexity.

Fractal geometry is a method of characterizing objects that traditional geometry is unsuited to describe. The Euclidean geometry along with calculus describes many shapes, such as parabolas, circles, triangles or ellipsoids. Many natural objects may be approximated to these shapes with great accuracy, e.g. the earth as a sphere. When it comes to more complex shapes, such as that of a snow crystal, most plants or even coastlines, traditional geometry's shortcomings become obvious. Indeed, the computer games industry readily illustrates the challenge of animating a realistic looking tree.

Fractal geometry offers not only a way of constructing many complex patterns, but also a way of characterizing a certain type of regularity in an otherwise complicated pattern, the fractal dimension. It describes how the amount of details at one size-scale relates to that on other scales. If no such relation exists in an object, the various algorithms will report this as well, as they all rely on the existence of a precise linear fit to a curve.

The concept of the fractal dimensions will be explained along with several algorithms used to estimate the fractal dimension of images, and different approaches to fractal image analysis. Percolation theory, a method used to mathematically construct structures with fractal characteristics, is discussed at the end of the section, and will be applied later in a simulation of the fractal network.

2.3.1 Dimensions

Euclidean Dimension

The Euclidean dimension, D_{euclid} , corresponds to the intuitive notion of the term, as used in everyday speech. The Euclidean dimension directly relates to the type of the object; it is zero for points, one for curves, two for planar objects and three for volumes. To put it another way, it is the minimum number of independent coordinates required to mathematically describe the figure.

Lebesgue Covering Dimension

The Lebesgue covering dimension, also known as the topological dimension, is a more stringently defined form of dimension. An object may be covered by any number of small open¹² sets. In order to completely cover the object, the open sets must overlap. Given that sufficiently small covering sets are used, the dimension of the object is one less than the least possible maximum number of circles covering any one point on the object, see figure 2.18. It is clear that this definition results in integer dimensions in accordance with the euclidean dimension.

Classical Fractals

At the end of the nineteenth century several mathematical monsters, beginning with Peanos curve, were constructed. These were objects that could not be classified by the Lebesgue Covering Dimen-

¹²Open Set, *mathematics*: An open set includes all points inside the set's boundary, but not the boundary itself.

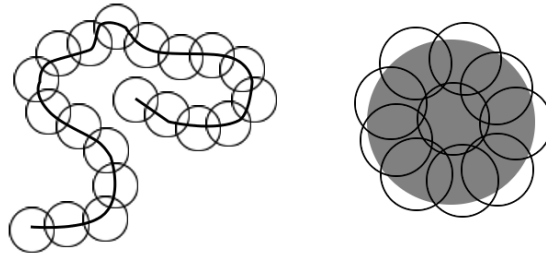


Figure 2.18: Covering sets illustrating the Lebesgue covering dimension. On the curved line a maximum of only two circles need to overlap at any one point in order to cover the entire line. The two-dimensional circle on the other hand, requires at least three circles to overlap in some areas. The covering sets need to be sufficiently small, lest a single set could cover higher dimensional parts of the object, or even the entire figure.

sion, effectively revealing the shortcomings of the traditional integer dimension. A similar curve constructed by Hilbert is shown in the top row of figure 2.19 to illustrate this. As with the other sets shown here, the Hilbert curve has a recursive definition and is the limit-object achieved when the iterations are carried on to infinity. In mathematics this is called an *iterated function system*, IFS. The result is a curve that visits all points in the plane without crossing or touching itself. The curve provides a one-to-one mapping between the plane, which requires two coordinates to represent, and a line.

The object is a curve and one would intuitively attribute it to the dimension one. On the other hand, the object visits every point in the plane, a quality that usually is attributed to the dimension two. Using the Lebesgue covering dimension, the same conclusion must be made. In order to make the result become one, infinitely small sets must be used. However, infinitely small open sets do not overlap at all, and consequently do not cover the curve. The notion that it was the type of object (point, curve, etc.) that determined its dimension, rather than its shape, had to give way for new ideas.

Fractal dimension

To deal with these new monsters Hausdorff developed the first fractal dimension. Hausdorff's dimension, D_H , can be calculated directly from the definitions of the IFS. The dimension of a recursive object is the logarithm of the number of copies, from the previous step, which are united to form the current figure, divided by the factor by which these copies are scaled down, see figure 2.19. In the case of Hilbert's curve the dimension is $\log 4 / \log 2 = 2$.¹³

An object's fractal dimension should not be considered to be a qualitative statement about what kind of object it is (points, lines, areas, etc.), but rather a quantitative measure of the object's behaviour across different size-scales. The fractal dimension does not relate to sizes such as length, area and volume. These terms are all connected to the integer dimensions, the fractal dimension is rather a recognition of the fact that some objects (i.e. fractals) do not fit these categories. The Koch Island for instance, has a finite area, but an infinite perimeter, as shown in equation 2.1, where A_0 and P_0 are

¹³There is a formal rigorous definition of the Hausdorff dimension as well, however the informal version used here is sufficient for this explanation, cf. <http://mathworld.wolfram.com/HausdorffDimension.html> for more information.

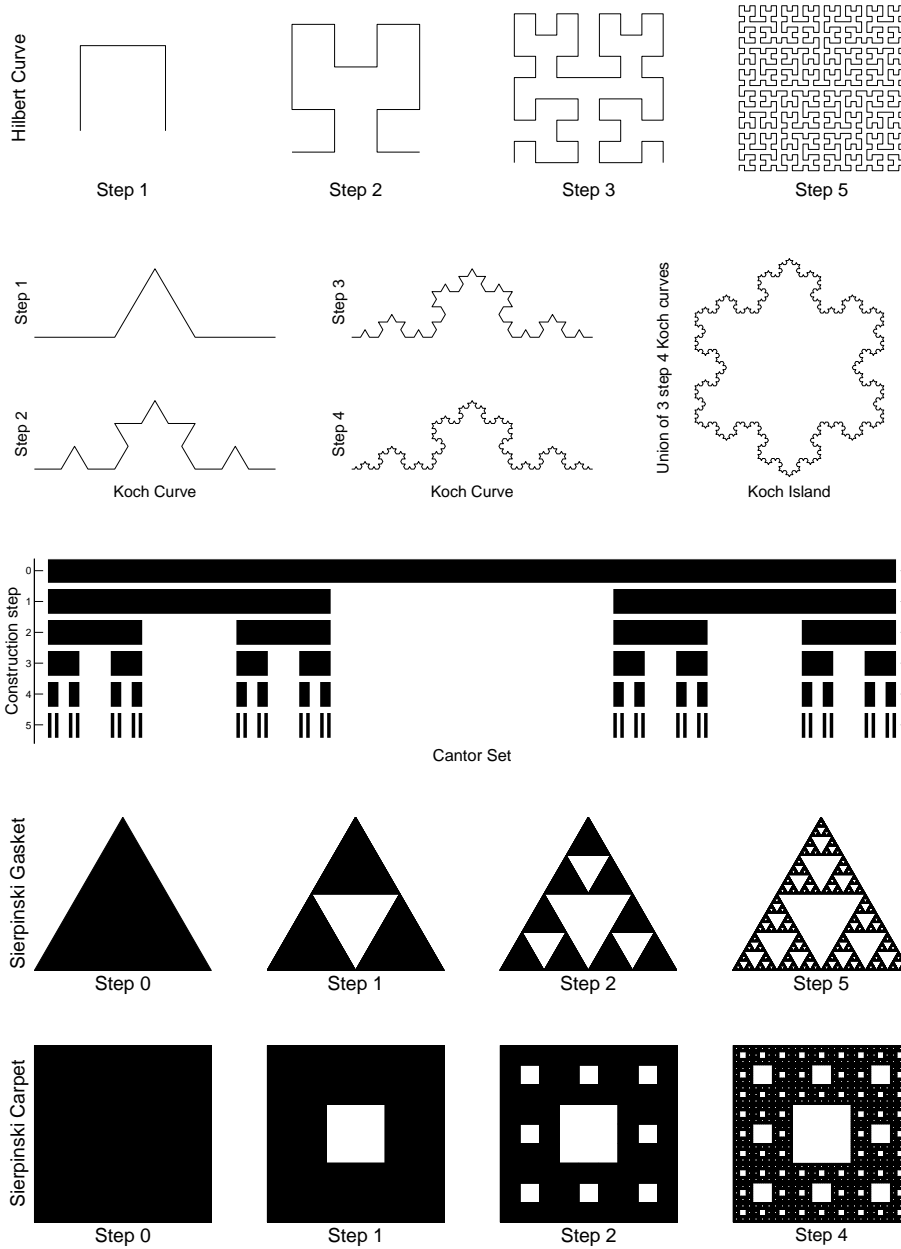


Figure 2.19: Classical fractal sets. These sets have a well-defined Hausdorff dimension (mind the set is defined as the attractor after infinitely many iterations), all of which are greater than their topological dimension: The Hilbert Curve, $D = \log 4 / \log 2 = 2$. The Koch Curve, $D = \log 4 \log 3 / \approx 1.2619$. The Cantor Set, $D = \log 2 \log 3 / \approx 0.6309$. The Sierpinski Gasket, $D = \log 3 / \log 2 \approx 1.5850$. The Sierpinski Carpet, $D = \log 8 / \log 3 \approx 1.8928$.

area and the perimeter of the initial triangle.

$$A_{\infty} = A_0 + 3A_0 \sum_{n=1}^{\infty} \frac{4^{n-1}}{6^n} = 2.5A_0 \quad P_{\infty} = P_0 + P_0 \sum_{n=1}^{\infty} 1/3 = \infty \quad (2.1)$$

Sierpinski's Gasket and Carpet have no areas at all, but likewise an infinite perimeter. A straight line will always have an area of zero no matter how long it is. However, if it is complicated and tortuous enough, caving in on itself across all length scales, such as the Hilbert Curve, it can indeed fill the entire plane. The fractal dimension, can loosely be said to refer to how much of a higher dimensional space a lower dimensional object occupies.

2.3.2 Self-Similarity

The term fractal is closely linked with that of self-similarity. Enlarging a self-similar object will reveal that it consists of many smaller objects of some smaller size that are similar to the object as a whole. For instance, all the objects in figure 2.19 are strictly self-similar, being constructed by putting together smaller parts of itself. Less strict forms of self-similarity have also been defined, such as self-affine¹⁴ sets and statistical self-similarity. Self-affine sets allow all affine operations, and not just linear operations, to be used when constructing the iterated function systems, i.e. rotations are allowed as well. The vaguer term, statistical self-similarity, refers to objects which are not made up by smaller exact copies of itself, but never the less behave in a similar manner. The term fractal dimension was indeed first used about coastlines, stating that:

Seacoast shapes are examples of highly involved curves with the property that – in a statistical sense – each portion can be considered a reduced-scale image of the whole. This property will be referred to as “statistical self-similarity”.

B. Mandelbrot [31]

A seacoast is obviously not made up by smaller parts of itself in any literal sense, yet it is impossible, from looking at a drawn outline of some coast, to see what scale the coastline has been drawn at. It is this type of self-similarity that warrants the use of fractal models for natural objects.

2.3.3 Natural Fractals

To satisfy strict mathematical definitions of a fractal, an object must satisfy an exact fractal scaling pattern all the way down to an infinitesimal scale. Never the less, it is common practice to call natural objects *fractals* even if they are only statistically self-similar over a finite range of length scales. Although natural objects may not be true fractals, it seems reasonable to model real objects as fractals in much the same way as a perfect circle may be used to represent the cross section of an artery – with full knowledge of the fact that the model is a mere approximation. [3]

2.3.4 Finding the Fractal Dimension of a Natural Fractal

Hausdorff's dimension provides a way to calculate the fractal dimensions. Its definition is, however, only applicable to mathematically defined functions. In order to quantify the fractal dimension of statistically self-similar sets, i.e. natural fractals, other methods must be used. For this purpose several algorithms have been developed.

¹⁴Affine *mathematics* : Of or pertaining to a transformation that maps parallel lines to parallel lines and finite points to finite points. [Dictionary.com, January 16, 2007]

These methods all produce dimensions similar to the Hausdorff-dimension for classical fractals, although not necessarily identical. Furthermore, the methods do not always give similar results for all conceivable image styles. Due to this and the uncertainty of the methods, the various dimensions should be denoted by the methods used to calculate them.

2.3.5 Box-Counting Dimension

The standard box-counting algorithm is the most straight forward way of calculating the fractal dimension. While the Lebesgue covering dimension considers the number of overlapping open sets, the box-counting dimension considers the number of closed square sets, i.e. boxes, which are needed to cover the object, as a function of the box size.[16]

When a given object is covered by boxes of a given size, $S > 0$, it requires some finite number, N , of boxes to cover it. For many objects N is related to S according to a simple power relation. For instance, if a straight line is covered by N_1 boxes of size S_1 , the same line would require three times as many boxes if the width of the boxes is reduced to a third, $N(S_1/3) = 3N(S_1) \Rightarrow N \propto S^{-1}$. Would the same be done to a square, nine times as many boxes would be required, $N(S_1/3) = 9N(S_1) \Rightarrow N \propto S^{-2}$. Recognizing the exponential as the negative of the dimension, one arrives at the formula

$$N \propto S^{-D} \Rightarrow D = \frac{\log N/c}{\log 1/S}, \quad \text{where } c \text{ is a constant.} \quad (2.2)$$

The procedure may seem simple. Applying the same procedure to the classical fractals, however, results in dimensions similar to the Hausdorff dimension. This can most easily be illustrated by the Sierpinski Carpet where reducing the box-size by a factor of three, results in a formula identical to Hausdorff's, $D_H = \log(8)/\log(3)$.

The somewhat arbitrary choice of what factor the box-size should be reduced by, cannot be allowed to require a lucky guess. If you start out with a single box covering the entire carpet and reduce the box-size by a factor two, then no boxes at all would be empty and $D = 2$. Reducing by a factor four will produce sixteen boxes and a size that is smaller than the empty area in the middle, but the boxes are stacked wrongly so that no boxes are empty. Shifting the boxes around, however, will allow one of the boxes to be empty and the dimension changes to $D = \log(15)/\log(4) \approx 1.95$, which is still much higher than $D_H \approx 1.89$. To overcome these challenges the Box-Counting dimension needs to be averaged over different box-sizes, and for each box-size the boxes need to be shifted around to find the configuration with the maximum number of empty boxes. The averaging is accomplished by rewriting equation 2.2 to

$$\log N = D_{box} \log 1/S + \log c \quad (2.3)$$

The dimension can now be found as the slope of a linear fit to a curve in a double-logarithmic environment.

2.3.6 Sandbox Dimension

The sandbox dimension considers the amount of vacant pixels within a neighbourhood around occupied pixels. Each occupied pixel is surrounded by boxes of size $S_1 \dots S_n$. The number of occupied pixels $m(S_j)$ inside each square of size S_j is averaged over the squares around all occupied pixels, to produce the mean number of occupied pixels $M(S_j)$. $M(S_j)$ obeys a power law relation to S_j

$$M(S_j) \propto S_j^D \Rightarrow \log M(S_j) = D \log S_j + \log c, \quad (2.4)$$

allowing the fractal dimension, D_{sb} , to be found as the slope of a linear fit in a loglog plot of $M(S_j)$ against S_j . [16]

This method has the additional advantage that, rather than calculating the global dimension from the mean occupancy of each box-size, a local fractal dimension, D_{local} , may be calculated for each occupied pixel.

$$m(S_j) \propto S_j^{D_{sb,local}}$$

2.3.7 Fourier Dimension

Fourier Transformation

The Fourier transformation of a function is defined by:

$$\mathcal{F}(f(x)) = F(\omega) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{i\omega x} dx, \quad \text{for every real number } \omega. \quad (2.5)$$

$$\mathcal{F}(f(n)) = F(k) \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} f(n) e^{\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1 \quad (2.6)$$

and its inverse function by

$$\mathcal{F}^{-1}(F(\omega)) = f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega) e^{i\omega x} d\omega, \quad \text{for every real number } x. \quad (2.7)$$

$$\mathcal{F}^{-1}(F(k)) = f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) e^{\frac{2\pi i}{N} kn} \quad n = 0, \dots, N-1 \quad (2.8)$$

The variable change in the exponential of the discrete version comes from substituting the angle frequency, ω , with the corresponding frequency: $\omega = 2\pi f$, when f is discrete $f_k = k/N$, where N is the length of $f(n)$. The discrete transform contains the frequency range $0 - (N-1)$, and requires at least $2N-1$ points to accurately describe the function. Both the zero-frequency, f_0 and the N -frequency, f_N , applies to all points in the function. Likewise a frequency slightly above f_0 and slightly below f_N will both cover almost all points. Because of this, half the frequencies are redundant, the DFT is symmetric and only $(n+1)$ frequencies are required. The phase of each frequency determines which points are included and which are excluded. The DFT returns one complex number per calculated point (usually $(2N-1)$). $|F(k)|$ is the strength of the k -th frequency component, and $\arctan(\mathcal{B}(F(k))/\mathcal{I}(F(k)))$ is the phase-shift of this frequency.

Calculation of the Fourier Power Spectrum Dimension

The power spectrum of the Fourier transform is given by [41]

$$S(\vec{k}) \propto |\mathcal{F}(\vec{k})|^2 \quad (2.9)$$

where $\mathcal{F}(\vec{k})$ is the Fourier coefficients of $f(x,y)$ in the space of frequency $\vec{k} = (k_x, k_y)$. If there are no characteristic lengths on the image, i.e. $f(x,y)$ is a random scaling fractal function, then the power spectrum averaged over all angles is related to the frequency by the power law

$$\langle S(k) \rangle_{\theta} \propto \frac{1}{k^{\beta}} \quad (2.10)$$

The authors of reference [24] provide a formula to map β to the fractal dimension, the mapping does, however, not provide meaningful results for many figures of a known dimension, nor does it provide meaningful results for their own images, which end up with fractal dimensions between two and three when this mapping is applied to β . The mapping from β to $D_{fourier}$ is further discussed in section 3.5.2, where the relation between the β and the fractal dimensions of a series of test shapes is investigated. For most shapes it appears that a direct mapping, i.e. $D_{fourier} = |\beta|$ gives the best results. This approach does, however, fail when applied to two-dimensional surfaces.

2.3.8 Mass Dimension and Correlation Dimension

Pair-correlation, Autocorrelation and Convolution

The continuous and discrete pair-correlation functions are respectively defined as:

$$\begin{aligned}(f \star g)(r) &\stackrel{\text{def}}{=} \int f^*(x)g(r+x) dx \\ (f \star g)(m) &\stackrel{\text{def}}{=} \sum_n f^*(n)g(m+n)\end{aligned}\quad (2.11)$$

where $f^*(x)$ is the complex conjugate of $f(x)$. The pair-correlation of a function with itself is called the autocorrelation. The autocorrelation, $(f \star f)$, is a quantitative measure of how the fluctuations of $f(x)$ are related at x and $x+t$, i.e. how well the function match a shifted version of itself, as a function of the amount it is shifted. Put in another way, the normalized autocorrelation is the probability of finding another mass point within a distance r from an existing point.

The pair-correlation is related to the convolution of two functions, defined as

$$\begin{aligned}(f * g)(r) &\stackrel{\text{def}}{=} \int f(x)g(r-x) dx \\ (f * g)(m) &\stackrel{\text{def}}{=} \sum_n f(n)g(m-n)\end{aligned}\quad (2.12)$$

by

$$f(x) \star g(x) = f^*(-x) * g(x).$$

If either of the two functions are even¹⁵ the operations are the same.

If the two functions f and g are the probability distributions of the two independent variable distributions, X and Y , then the correlation and the convolution correspond to the probability distributions of the difference $(-X + Y)$ and the sum $(X + Y)$ of the variables respectively.

For a finite function one must choose a way to handle the start of a convolution, or the end of a pair-correlation. There are essentially two approaches to this, a linear and a cyclic. In the linear convolution/correlation the function is expanded with zeros at all extra points needed in the algorithm. In the cyclic approach the function is regarded to be repetitive. The linear approach will be used here. [41]

The Convolution Theorem

The Convolution theorem states that the Fourier transform of a convolution is equal to the multiplication of the respective Fourier transforms of the two functions,

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g).$$

¹⁵Even function: A function is even if $f(x) = f(-x)$ and odd if $f(x) = -f(-x)$.

This property is exploited for large discrete functions by using the Fast Fourier Transform, *FFT*, algorithm to quickly calculate the convolution of two functions.

Wiener-Khintchine Theorem

The Wiener-Khintchine relations apply to most cases and state that the power spectral density, $S(k) = |\mathcal{F}(f(n))|^2$, is the Fourier transform of the corresponding autocorrelation.

$$S(\omega) = \int_{-\infty}^{\infty} (f \star f)(r) e^{i\omega r} dr$$

For a real function, f , the exponential simplifies to a cosine. For a real and discrete function, $f(n)$, the theorem takes the form:

$$S(k) = \sum_{k=-\infty}^{\infty} (f \star f)(m) \cos\left(\frac{2\pi}{N} km\right). \quad (2.13)$$

and the inverse transformation:

$$(f \star f)(m) = \sum_{k=-\infty}^{\infty} S(k) \cos\left(\frac{2\pi}{N} km\right). \quad (2.14)$$

Rewriting the theorem to a more general form gives

$$\begin{aligned} (f \star f) &= \mathcal{F}^{-1}(|\mathcal{F}(f)|^2) \\ \mathcal{F}(f \star f) &= \mathcal{F}(f) \cdot \mathcal{F}(f) \end{aligned} \quad (2.15)$$

which corresponds to the convolution theorem, but applies to autocorrelations. Furthermore, for real functions $(f \star f) = (f * f)$, allowing both convolution and correlation algorithms to be used in the calculation.

Computation of the Mass and Correlation Dimensions

In the two dimensional case, let $C(\vec{r})$ denote the autocorrelation,

$$C(\vec{r}) = C(r_x, r_y) \equiv \langle m(x, y) \cdot m(x + r_x, y + r_y) \rangle = (m(x, y) \star m(x, y))$$

where $m(x, y)$ denotes the local mass density, i.e. the image, and the brackets $\langle \dots \rangle$ denote an ensemble average. $C(r)$ represents the average of $C(\vec{r})$ over all angles such that $r^2 = r_x^2 + r_y^2$.

$$C(r) = \langle C(\vec{r}) \rangle_{\theta}$$

Keeping in mind that $C(r)$ is the probability of finding another mass-point within a distance r from an existing one, the conditional total average mass, $M(R)$, is defined. The total number of points, in a statistical sense, within a distance R from an existing point, is closely related to the dimension of the set.

$$M(R) = \int_0^R C(r) r^{(D_{euclid}-1)} dr \propto R^{D_{mass}} \quad (2.16)$$

Where $0 < D < D_{euclid}$, and D_{euclid} equals 2 for 2-D images. Note that equation 2.16 is analogous to equation 2.4, showing the power-law relation of the Sandbox dimension.

By differentiation of equation 2.16, the correlation dimension relation is obtained

$$C(r) \propto r^{D-D_{euclid}} \propto 1/r^{D_{euclid}-D_{cor}} \propto 1/r^n \quad (2.17)$$

which for two-dimensional images is

$$C(r) \propto 1/r^{2-D_{cor}} \quad (2.18)$$

The 2-D FFT algorithms provide an efficient way of computing the 2-D $S(\vec{k})$ from $f(x,y)$ and from this $C(\vec{r})$ may be calculated with the 2-D Wiener Kintchine theorem. Averaging $S(\vec{k})$ to produce $S(k)$ and then using the 1-D Wiener-Kintchine theorem to produce $C(r)$ may introduce errors, consequently the angle average should be the last step in the procedure. [41]

Local Mass Dimension

Equation (2.16), representing the global average of $M(R)$ over all positions that have mass (i.e. non-zero), calculates the global fractal dimension D . $M(R)$ may also be studied around a particular point \vec{r}_0

$$M_{\vec{r}_0}(R) \propto \int_0^R m(\vec{r}_0 + \vec{r}) d^2\vec{r} \propto R^{D(\vec{r}_0)} \quad (2.19)$$

providing the local mass dimension $D_{local} = D(\vec{r}_0)$ around the point \vec{r}_0 . [41]

2.3.9 Analysing Images

A suitable image may be considered a natural fractal. An image represents a dataset, and datasets of any dimension may be analyzed by adapting these methods. One-dimensional problems are of particular interest in signal-processing and statistics. Two-dimensional sets include all images and are of particular interest here. Three-dimensional datasets allow the true representation of e.g. vascular structures throughout a tumour. They are, however, not easily obtained and require a large amount of computer memory to process.

A digitized (uncompressed) greyscale image is usually stored as an intensity matrix $I(x,y)$. There are several different ways of storing colour images, however, because the fractal methods all operate on a single matrix, colour images must be mapped to an intensity matrix before analysis.

There are many ways of characterizing an image by fractal analysis. Not only will any image manipulation technique used prior to the analysis affect the result, but there are several different ways of analysing the image.

Defining a Measure

The fractal methods described above all apply to some measure $m(x,y)$. This measure must be chosen and the choice obviously directly affects the result. This makes it important to define the measure rigorously.

The simplest box-counting and sandbox algorithms require that a pixel must be defined as either empty or occupied, i.e. it requires a black-and-white measure. For these algorithms, threshold techniques are likely key-parts of the measure definition. Other image processing techniques may be applied as well, such as identifying border pixels between some form of distinguishable areas or calculating mass centres.

For the methods capable of handling grayscale measures $m(x,y)$ may equal $I(x,y)$. This is, however, usually undesirable because non-zero background sensitivity changes across the image, or sharp

edges, may effect the result, in particular they can seriously corrupt $S(\vec{k})$ and $C(\vec{r})$. The major difficulties in using $I(x,y)$ as the image measure may be eliminated by replacing it with the *local image gradient* $|\nabla I(x,y)|^{16}$, which emphasizes the edges in an image.[41]

For colour images unconventional mappings from colour to grayscale may be used as a part of the measure definition, for instance, specific colours can be extracted.

Global Dimension

The global dimension is a characterization of a measure across the entire image by a single fractal dimension. This is *the* fractal dimension and will consequently be denoted simply by D .

Local Dimension

Local dimensions provide another way of characterizing images. The global dimension may be inappropriate when studying features that represent a subset of the original image. The D_{local} is calculated around (all) individual occupied points \vec{r}_0 . The global distribution of D_{local} may be represented either by a histogram or a transformed intensity image, where the occupied pixel values have been replaced by their local dimension. D_{local} is not strictly a dimension and may take values outside of the usual range, $0 < D < 2$. The Sandbox and the Mass dimensions provide means for calculating the local dimension.

Locally Connected Fractal Dimension

By only including connected pixels to the local measure, $m(\vec{r}_0 + \vec{r})$, around \vec{r}_0 , the local connected fractal dimension may be calculated. The difference between D_{conn} and D_{local} is clarified by considering an image of closely spaced, but disconnected parallel lines. For scalings larger than the spacing between the columns, the image behaves like a texture and $D_{local} \approx 2$. The local connected dimension, on the other hand, only includes the pixels on single lines and $D_{conn} \approx 1$. This dimension is important in percolation theory. [41]

2.3.10 Percolation Theory

Percolation theory offers several mathematical models that have been applied to describe complex phenomena, such as liquid draining through porous media or the electrical resistance across complicated networks consisting of, or modelled as, large numbers of intertwined parallel and serial connections of varying resistance. Percolation clusters have several fractal characteristics and invasion percolation in particular has been suggested as a mechanism involved tumour angiogenesis, or at the very least a way to model it.

Random Percolation

In the random percolation model, RP, all positions on a grid are assigned a random value $r_n \in (0, 1)$, and the percolation threshold, $p \in (0, 1)$, is chosen. A percolation cluster is defined as a network of connected cells with values $r_n < p$. The clusters are located by planting a seed, i.e. choosing one or more cells as the initial cluster, and iteratively adding all neighbours that are below the threshold of the cluster.

¹⁶ $|\nabla I(x,y)|$ may be estimated by a local fit of $I(x,y)$ in the neighbourhood of x,y to the form $a\Delta x + b\Delta y + I(x,y)$. This gives $|\nabla I(x,y)| = (a^2 + b^2)^{1/2}$.

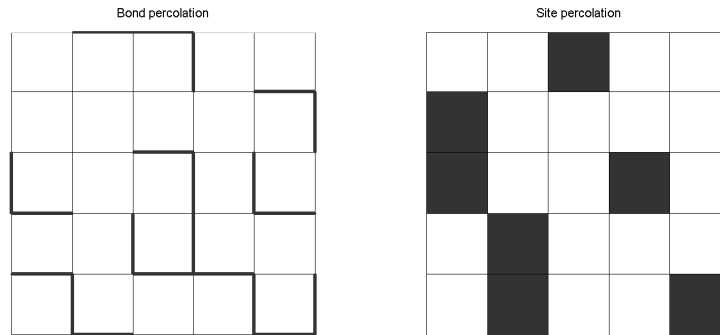


Figure 2.20: Bond and site percolation: In bond percolation the lattice is randomly open or closed. In site percolation the vertices are randomly open or closed. Both of these networks were generated with $p = 0.3$.

lattice	p_c – bond percolation	[exact]	p_c – site percolation	[exact]
square	0.50000	1/2	0.592746	
triangular	0.34729	$2 \sin\left(\frac{\pi}{18}\right)$	0.500000	1/2
honeycomb	0.65271	$1 - 2 \sin\left(\frac{\pi}{18}\right)$	0.6962	
cubic (simple)	0.2488		0.3116	

Table 2.2: List of critical percolation thresholds for a handful of common lattices.

There are two types of percolation that need to be distinguished, namely bond percolation and site percolation. Bond percolation considers the lattice edges to be the random entities. Site percolation, on the other hand, considers the vertices as such, see figure 2.20.

This method has been used to model the propagation of fluids¹⁷ through porous media. If a fluid is added to one side of a porous media (the seed), and some percentage (p) of the subvolumes in the medium have pores that the fluid potentially can propagate through, then the probability of the existence of a continuous network through the medium, allowing the fluid to reach the other side, depends on p . For large lattices and high values of p there will, almost certainly, be a cluster spanning the network, and for low values of p there is almost certainly not. A continuous network spanning across the lattice from side to side is called a spanning cluster. At some critical value p_c , the network morphology changes drastically.

The values of p_c varies for different lattices and for bond and site percolation. A few values are known exactly, but most of them have been approximated numerically. Critical percolation values for a few common lattices have been listed in table 2.2.

Random percolation spanning clusters have been shown to have a fractal shape at the critical percolation threshold. Future references to random percolation clusters will assume a critical cluster. All occupied sites belong to some cluster within the network, possibly consisting of a single site. For this reason it is important to use the locally connected fractal dimension for both global and local calculations. Usually it is the single largest cluster that is of interest.

¹⁷Fluid: the model works for both liquids and gases.

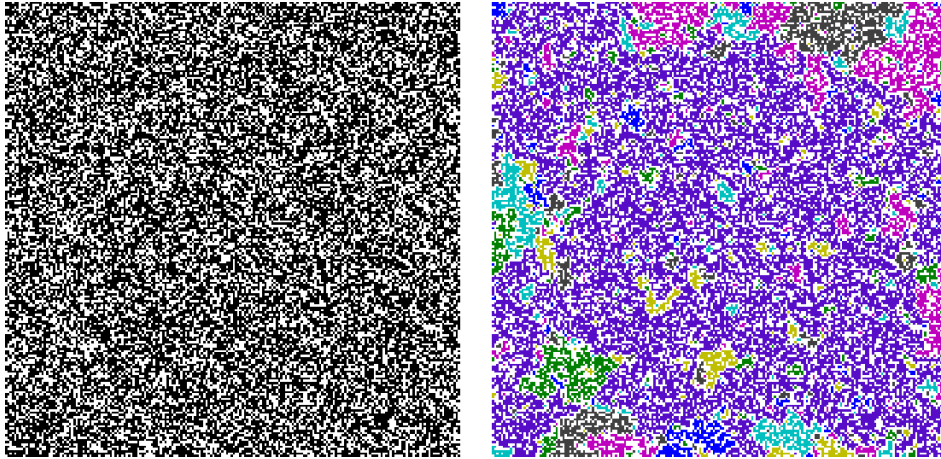


Figure 2.21: Random site-percolation cluster, 200x200 pixels. The concept of local connectivity is important in percolation theory. While the lattice itself is 'white noise' (left), fractal shapes are embedded in connected clusters throughout the grid. There are typically many small, a few medium sized, and one very large cluster spanning the lattice (right). This cluster is sometimes called an infinite cluster, even on finite lattices, because it is analogous to a cluster, which at the percolation limit exists with some probability, and spans infinite lattices.

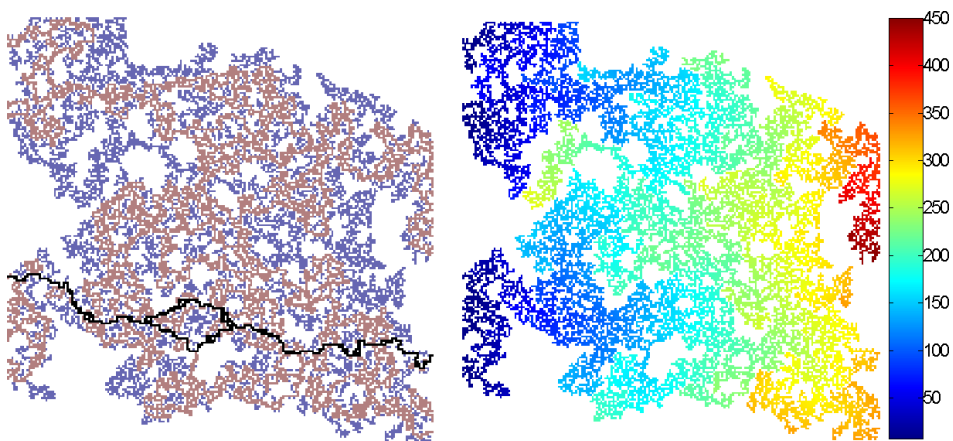


Figure 2.22: Left: The Transport Backbone (tan), Elastic Backbone (black) and Dangling Ends (blue) are shown for the largest percolation cluster. Right: The chemical distance, measured as the length of the shortest travel path to the inlet side on the left. Note that cluster sites that are geometrically close may be distant in terms of chemical distance.

Invasion Percolation

Invasion percolation, IP, is a model describing a situation where an invading fluid propagates through a porous medium filled with a defending fluid. In this model each site (or bond) is assigned a random value $r \in (0, 1)$. A set of points is defined as the invasion front at t_0 . This is often one of the lattice walls, but may also be a single point, depending on the situation being modelled. The invasion front is defined as all defending sites that are neighbours of an invading site. At each time step, the invading fluid invades the single site at the invasion front with lowest r .

Two different situations arise depending on whether or not the defending fluid is compressible. In non-trapping invasion percolation, NTIP, the defender is compressible and the invader can potentially enter any region on the interface. In trapping invasion percolation, TIP, on the other hand, the defender is incompressible and is trapped if a region of defenders is completely surrounded by the invading fluid.

The propagation of a fluid through a porous material is dependent on the fluid's ability to wet the internal surface of the medium. For a wetting fluid, the capillary pressure is large and negative, pulling the fluid into the smallest pores first, but giving it more trouble with larger capillaries. A non-wetting fluid, on the other hand, will have the most trouble with the smallest capillaries. If a porous material is modelled by a network of pores (sites) connected by throats (bonds) which have smaller radii than the pores, then a wetting fluid is best modelled by a site percolation while a non-wetting fluid is best modelled by bond percolation.[37][15]

Invasion percolation differs from random percolation in at least two ways. There is no parallel to the percolation threshold, p , determining the morphology of random percolation. Furthermore, random percolation is a static model and invasion percolation is a dynamic one allowing not only the final cluster to be studied, but also the propagation of the invader through the medium.

IP-clusters obey fractal scaling laws and the dimensions of the various types of IP are listed in table 2.3.

Fractal Characteristics of Percolation Clusters and Universality

Both random percolation and invasion clusters exhibit several fractal scaling properties. Not only are the spanning clusters fractals, but also the transport backbone of the cluster and the shortest path across the cluster. The backbone consists of all possible paths from one side of the cluster to the other, with the one constraint that no site is visited more than once. The backbone is obtained by removing all dangling ends from the cluster. It is sometimes called the transport backbone because it consists of all points that may participate in the transport across the network. The minimum path, also known as the shortest path or the elastic backbone, consists of the points in all the possible shortest routes between two points at opposite sides of the cluster. Thus, the route between the two points is a minimum route across the cluster. The elastic backbone is a part of the transport backbone.

Many complicated systems will, when approaching criticality (phase transition), behave in some scale-invariant manner. Different systems are said to belong to the same universality class if they behave according to the same power-scaling laws. For instance, even though there are two types of random percolation and numerous different 2-dimensional lattices, with different phase transitions, p_c , see table 2.2, they all have the same fractal dimension $D_f = 91/48$. Furthermore, NTIP is in the same universality class as well. For two-dimensional square lattices, there are only two different universality classes, one for RP and NTIP, and another for TIP. In three-dimensional simple cubic lattices, there are two classes as well. This time one for RP, NTIP and site TIP, and a different one for bond TIP. While there is a single universality class for random percolation, this is not the case for

2-dimensional square lattices			
Model	D – spanning cluster	D_b – backbone	D_{min} – minimum path
RP	1.89583 exact: 91/48	1.6432	1.1307
NTIP	1.89499	1.6422	1.1293
TIP site	1.825	1.217	1.214
TIP bond	1.825	1.217	1.2170
3-dimensional simple cubic lattices			
Model	D – spanning cluster	D_b – backbone	D_{min} – minimum path
RP	2.523	1.87	1.374
NTIP	2.528	1.868	1.3697
TIP site	2.528	1.861	1.3697
TIP bond	2.528	1.458	1.458

Table 2.3: Dimensions of percolation clusters. The fractal dimension of the spanning cluster, D , the backbone, D_b and the minimum path D_{min} for random percolation, RP, non-trapping invasion percolation, NTIP, and trapping invasion percolation, TIP for square lattices. With the exception of D for RP, which is exact and valid for all 2-D lattices, the values are numerically calculated. The various numbers of digits reflect the uncertainty of the calculations. See reference [37] for further information on algorithms and uncertainties.

invasion percolation, nor is it true for trapping invasion percolation in a three-dimensional space, see table 2.3. [37]

2.4 Fractals and Cancer

Complexity, irregularities and poorly regulated growth are among the defining characteristics of cancer. Tumour vasculature, in particular, defies the optimized growth patterns of healthy vasculature and is known to contain many tortuous vessels, shunts, vascular loops, widely variable intervascular distances and large avascular areas, see sections 2.1.5 and 2.1.6. [2]

The vascular features are, however, not the only interesting morphological aspect of solid tumours. The growth patterns of both the cancer cells and the tumour-parenchymal border, the latter being a highly involved surface, are both tumour-type dependent. Fractal theory provides means of characterizing complex structures and phenomena. In this respect, fractal analysis is a promising new tool for quantitative description of tumours. In the words of Baish and Jain:

“By focusing on the irregularity of tumor growth rather than on a single measure of size such as diameter or volume, fractal geometry is well suited to quantify those morphological characteristics that pathologists have long used in a qualitative sense to describe malignancies—their ragged border with the host tissue and their seemingly random patterns of vascular growth. (...)

A more quantitative and hopefully more reproducible approach, which may serve as a useful adjunct to trained observers, is to analyze images with computational tools. Herein lies the potential of fractal analysis as a morphometric measure of the irregular structures typical of tumor growth.”

James W. Baish and Rakesh K. Jain [3]

2.4.1 Fractal Quantification of Tumour Vasculature

Vasculature is in general not readily accessible to morphological studies. Tumour vasculature is in most cases a three-dimensional network embedded in tumour and host tissues. Corrosion casting or the reconstruction of numerous immunohistochemically tissue slices, sampled throughout a tumour and stained by endothelial specific markers, may be used to study the network morphology. The challenges of fractal studies of the network are, however, not just related to data access, but also to the required computations. Both computation time and memory usage increase with the size of the system. The jump from two- to three-dimensional systems effectively requires a reduction of lattice resolution so that $l_{3D} = l_{2D}^{2/3}$ in order to maintain the same number of lattice points. So far the combination of these two aspects has resulted in most studies having been carried out on two-dimensional vessel systems (retina, dorsal skinfold chamber etc.), or in the use of immunohistological essays. The amount of work and computer power required to reconstruct and analyze a complete three-dimensional network has so-far limited the immunohistological studies to single slices.

2.4.2 Analysis of Two-Dimensional Tumour Models

Expected Results

Tumour vasculature is characterized by a different morphology than both the arteriovenous system and the capillaries. It is not a superposition of the two, nor a network initially covered by these, but with areas of various sizes removed. Due to tumour vasculature's high tortuosity, its fractal dimension is expected to be higher than the arteriovenous system, yet by no means as high as the capillaries filling

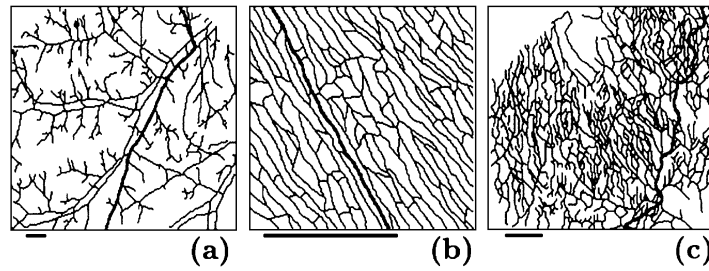


Figure 2.23: Skeletonized images of vascular networks. (a) Normal subcutaneous arteriovenous network. (b) Normal subcutaneous capillary network. (c) LS174T tumour network. The minimum paths across the images are shown in bold and the bars are $500\mu\text{m}$ long. (Reference [17])

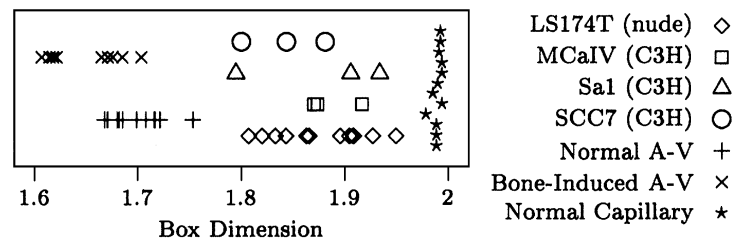


Figure 2.24: Box Counting dimensions of skeletonized vascular networks. The networks behaved scale-invariant over box-sizes ranging from approximately $50\text{-}900\mu\text{m}$. Sandbox dimensions were found to be similar. The tumour cell lines were LS174T, a human colon adenocarcinoma; MCaIV, murine mammary carcinoma; Sa1, murine sarcoma; SCC7, murine squamous cell carcinoma; all of which were implanted into dorsal chamber preparations. The bone-induced arteriovenous system was achieved by implanting whole femora into dorsal chamber preparations. (Reference [17])

the entire Euclidean space in which they are embedded. Furthermore, utilizing the minimum path dimension from percolation theory (see section 2.3.10), the increased tortuosity of the individual vessels should account for an increased D^{min} . Although insights into tumour angiogenesis and morphology allow some qualitative assumptions, they do not help in determining the size of the increase.

This is true with one exception. If the statistical growth process that produces the network could be identified, if indeed there is one, then the fractal characteristics of network should correspond to the process. What has been done is, however, exactly the opposite. Using fractal analysis on the networks, all processes with markedly different fractal properties (universality classes) can be eliminated. In this way fractal analysis has been used to give insight into the underlying growth process.

Gazit et.al. 1995 [17]—The Fractal Properties of Two-Dimensional Vessel Systems

This study investigated the fractal properties of two-dimensional vessel systems, see figure 2.23. The dimension of normal arteriovenous networks were found to be in concert with two-dimensional diffusion-limited aggregates ($D = 1.71$, $D^{\text{min}} = 1.00$). This seems to be consistent with an angiogenic process promoted by growth factors diffusing from hypoxic regions. Capillaries were found to have a dimension consistent with a space-filling curve ($D = 2.00$, $D^{\text{min}} = 1.00$). The minimum-path dimension, D^{min} , is in biological terms, a measure of the tortuosity of the vessels in the network. It was found to be significantly higher ($p < 0.0001$) for tumours than normal networks. From figure 2.24

network	tissue	D_{box}	D_{sand}	D^{min}
normal arteriovenous	subcutaneous	1.70 ± 0.03	1.70 ± 0.03	0.99 ± 0.02
	bone-induced	1.65 ± 0.04	1.66 ± 0.05	
normal capillary	subcutaneous	1.99 ± 0.01	1.97 ± 0.01	1.00 ± 0.02
tumour networks	LS174T	1.88 ± 0.04	1.89 ± 0.04	1.10 ± 0.04

Table 2.4: Fractal dimensions calculated from skeletonized images of vascular networks in reference [17], see figure 2.23.

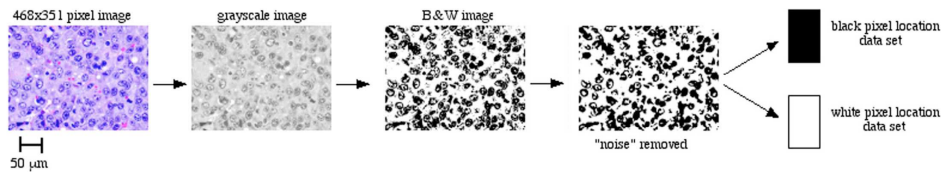


Figure 2.25: Spillman et.al. studied H&E stained immunohistological sections. The sections were processed into black and white pictures by thresholding a greyscale image at 50% of max intensity. (Reference [42])

it is, however, also clear that the dimensions of individual tumours take on a range of values. Furthermore, the observed fractal dimensions ($D = 1.88 \pm 0.04$, $D^{min} = 1.10 \pm 0.04$) coincide with the dimensions of random percolation and non-trapping invasion percolation, see table 2.3

2.4.3 Analysis of Tissue Sections

Spillman et.al. 2004 [42]—Fractal Analysis of H&E Sections

This group did not study the vasculature, but rather H&E¹⁸ stained sections of Morris 7777 hepatoma¹⁹ xenografts. Low resolution (600x400) images were further reduced in size to 468x351, converted to greyscale, and a threshold was performed at 50% of max intensity, see figure 2.25. The image foreground and background were then both analyzed (i.e. the inverted black and white images were analyzed as well as the original), for fractal characteristics with the box-counting algorithm. This double analysis is based on the assumption that the Box Counting curve is related to the distribution of voids within the analyzed image, and conversely, analysis of the inverted image relates to the distribution of the points themselves.

The fractal dimensions were compared to relative qualitative cell differentiation. A panel of four pathologists arranged nine H&E sections in ascending order from less to more differentiated, and the mean score was used as a measure of how differentiated the tumours were. The result is shown in figure 2.26. This suggests that the fractal dimensions of H&E sections go through an extremum as the tumour progresses towards higher differentiation.

Sabo et.al. 2001 [36]—Fractal Dimension and Patient Survival

Forty-nine patients with low-stage clear cell localized renal cell carcinoma were assessed in a nine-year follow-up retrospective study. Correlation between microvessel density, microvessel fractal di-

¹⁸H&E section, *immunohistochemistry*: Haematoxylin-Eosin staining, also known as HE or H+E. This method effectively stains red blood cells red, cell nuclei blue-purple, and other cellular and extracellular material pink.

¹⁹Hepatoma, *pathology*: Hepatocellular carcinoma, a cancer of the liver.

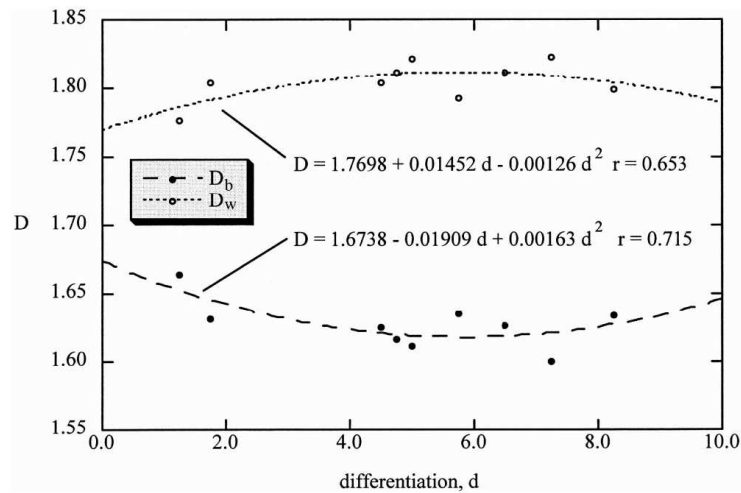


Figure 2.26: Fractal dimension of H&E sections compared to tumour cell differentiation as scored by a panel of four pathologists. The abscissa is constructed by the mean ordering of sections from the least to the most differentiated of the nine sections, making 1 the lowest and 9 the highest possible score. (Reference [42])

mension, histological grade, extent of necrosis and patient survival were tested by uni- and multivariate analysis. The microvessel densities and fractal dimensions were obtained from computerized fields of tumour sections immunohistologically stained for CD34. Microvessel density was calculated as the ratio between vascular and avascular areas in the section and the fractal dimensions by the box-counting algorithm. A high fractal dimension was found to correlate with a lower tumour grade, a higher five year survival rate and a lower incidence of high levels (>25%) of necrosis, see table 2.5. Multivariate analysis revealed that the fractal dimension was the only investigated parameter to correlate significantly to necrosis, and that necrosis was the only independent predictor of patient survival.

Grizzi et.al. 2005 [20]—Random Vessel Simulation

In this study a total of ten thousand random simulations of vessel distributions were made to investigate the behaviour of the fractal dimension as a function of vessel number, i.e. microvascular density. A fixed number of vessels (ranging from five to fifty for different groups) were placed randomly in an area without touching each other, and the fractal dimension was calculated by the box-counting method, see figure 2.28. Not surprisingly the configuration, of vessels affected the resulting dimension and the dimension increased with an increasing number of vessels.

Heymans et.al. 1999 [24]—Fractal Analysis of *Ulex Europaeus* by the Fourier Method

This group used a Fourier algorithm to calculate the power law scaling of the spectral density, β , of primary cutaneous melanoma vascular patterns. The three qualitatively different scaling patterns shown in figure 2.29 were observed. In the first (left), a power law with an exponent $\beta = 1.70$ is obtained at intermediate and low frequencies (large scales) while at higher frequencies (small scales) a different scaling is observed for a short frequency span before breaking down. This is interpreted as the characteristic distribution of vessels at large scales and the characteristic size of the vessels at smaller scales. In the second (middle), the same scaling pattern is observed almost across the entire region with $\beta = 2.10$. In the last pattern (right), two consecutive power laws are obtained. For high

	MVD ^a (mean \pm SD)	<i>P</i>	MFD ^b (mean \pm SD)	<i>P</i>
Tumour grade ^c				
Low	11.5 \pm 3.5%	0.12	1.55 \pm 0.11	0.03
High	9.4 \pm 5.2%		1.45 \pm 0.15	
Tumour necrosis ^d				
No	10.9 \pm 4.4%	0.03	1.52 \pm 0.12	0.01
Yes	7.1 \pm 4.6%		1.38 \pm 0.17	
Survival				
\geq 5 yrs	10.8 \pm 4.7%	0.03	1.56 \pm 0.11	0.02
$<$ 5 yrs	6.4 \pm 3.7%		1.46 \pm 0.15	

^a Expressed as the mean percentage of the vessel area per microscopically computerized field.

^b Expressed in absolute units.

^c Low grade tumours, Fuhrman grade I and II; high grade tumours, Fuhrman grade III and IV.

^d Tumours were considered necrotic if they exhibited $>$ 25% macroscopic necroses.

Table 2.5: Relationship between tumour grade, necrosis, patient survival, and vascular parameters in the renal cell carcinoma study of Sabo et.al. (Reference [36])

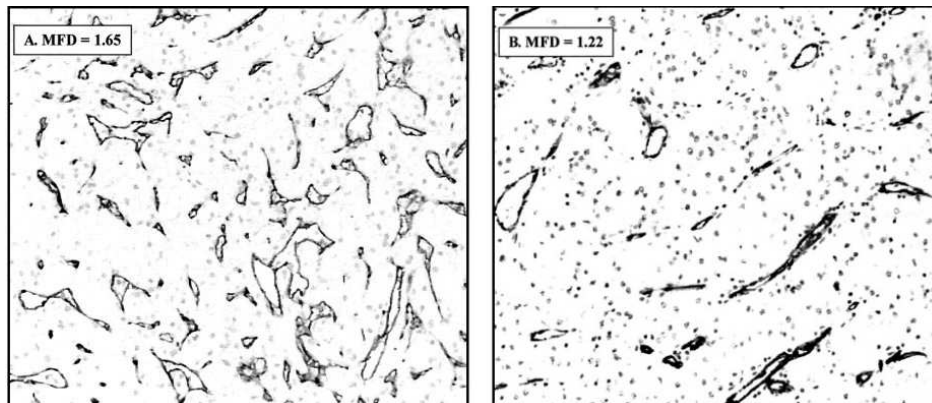


Figure 2.27: Renal cell carcinoma immunohistochemically stained for CD34 by Sabo et.al. The vasculature is clearly stained and there is some, but not much background staining (cell nuclei). It is worth noting that the background is different in the two sections, and that the authors make no comment as to having removed the background before analysis. The fractal dimensions of the two images are shown in the top left corners. (Adapted from reference [36])

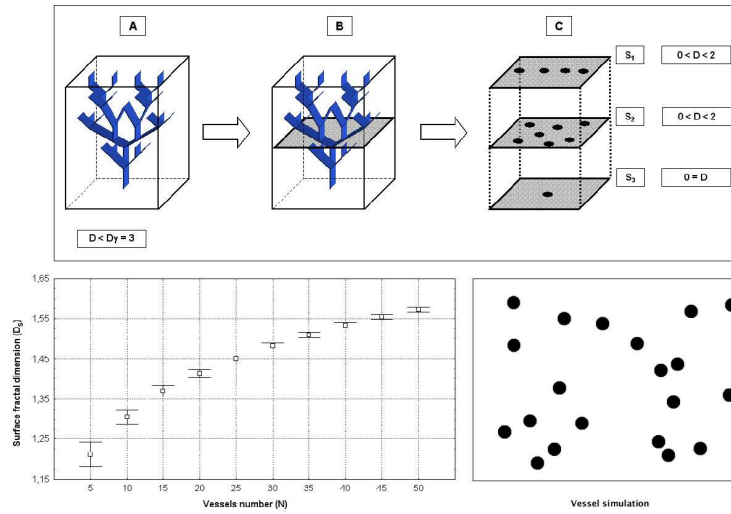


Figure 2.28: Grizzi et.al. carried out 1000 simulations for each vessel density from five to fifty with the number being increased by five for each group, and calculated the fractal dimension by the box-counting algorithm. The result is shown to the left. An example of a simulation with twenty vessels is shown to the right. An illustration of how the simulations relate to the network, is shown at the top. The dimension increases with an increasing number of vessels, however it is also dependent on the vessels' relative distributions giving rise to a standard deviation within each group. (Adapted from [20])

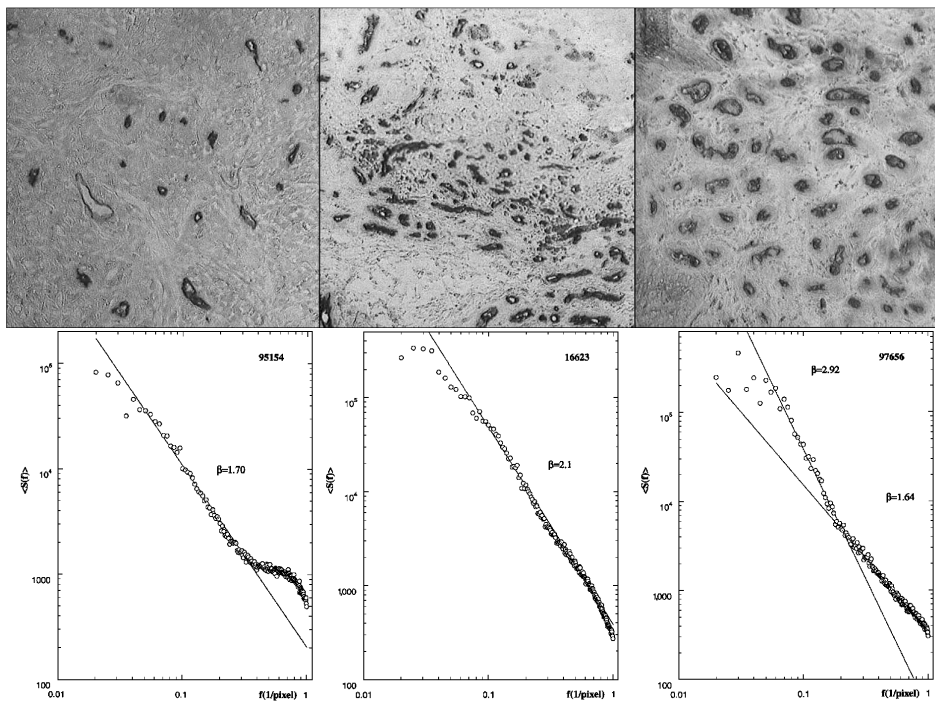


Figure 2.29: Primary cutaneous melanoma sections immunostained with *Ulex Europaeus* agglutinin-I were scanned optically at x125 magnification to 512x512 images. A gradient technique was applied to the images to enhance the edges and the Fourier dimension was calculated. The authors report the finding of these three qualitatively different scaling patterns. (Adapted from [24])

	colorectal carcinoma			malignant mesothelioma			inv. cervical carcinoma		
Confusion Matrix (%)	—	25.0	75	—	100.0	—	85.7	—	14.3
	6.3	50.0	43.8	—	91.7	8.3	9.1	81.9	14.3
	3.2	6.5	90.3	—	11.1	88.9	—	16.7	83.3
Correctly classified cases using:									
all parameters:		70.6%			86.4%			83.3%	
microvascular density:	(<i>P</i> : 0.2)	56.9%		(<i>P</i> : < 0.001)	60.9%		(<i>P</i> : 0.4)	25.0%	
fractal dimension:	(<i>P</i> : 0.2)	56.9%		(<i>P</i> : 0.4)	60.9%		(<i>P</i> : 0.8)	29.2%	

Table 2.6: Top: Confusion Matrices showing the actual prognosis on the vertical axis and identified prognosis using a panel of all good parameters ($P < 1$). Below: Correctly classified cases using all parameters, or only intervascular density or fractal dimension. The classification is done by using a panel of cases as a reference for the set of parameters measured and placing each case in the same class as the closest reference point. (Reference [43])

frequencies (small scales) a power scaling coefficient of $\beta = 1.64$ is obtained and interpreted as the scaling of the dense structure of vessel profiles with a large size distribution. The other region (large scales) has a high exponent, $\beta = 2.92$, corresponding to a uniform texture; at large scales the grey level variations are seen as small fluctuations on a smooth surface.

Weyn et.al. 2004 [43]—Fractal and Syntactic Structure Analysis—Correlation with Prognosis

This study investigated correlations between prognosis and parameters obtained from Syntactic Structure Analysis (SSA), fractal analysis, vessel numbers, area and perimeter, as well as set of clinical parameters. They investigated multiple cases of three different tumour types; colorectal carcinoma, malignant mesothelioma and invasive cervical carcinoma. The SSA was performed similar to that described in section 3.3, with the exception that the distances to the furthest neighbours in the graph were not investigated. The fractal analysis was performed using the Box Counting method, and the images were obtained by applying a threshold to CD31 stained sections. The threshold was done by a house written macro and manually inspected for errors which, if found, were manually corrected.

The correlations with prognosis was determined by constructing three survival classes. The middle survival class spanned survival longer than the mean minus one standard deviation, but shorter than the mean plus one standard deviation. The other two survival classes spanned the two regions that were further away than one standard deviations. Predictions were then made from each of the parameters using a K-nearest neighbour classifier obtained through direct comparison with survival in a subset of the cases. Confusion matrices evaluating the groups of parameters' ability to classify cases into the correct survival classes are shown in table 2.6.²⁰

For all three cancers, the predictions based on SSA generally provided much higher recognition scores compared to those obtained through the fractal dimension or the microvessel density. Survival of cervical carcinoma was best predicted by clinical data. Colorectal cancer correlated best with SSA complemented by the microvessel density. Mesothelioma showed a strong correlation with SSA.

²⁰Note that the percentages in two of the cases are the same for both microvascular density and fractal dimension. At least one of these numbers is quoted differently in the article text suggesting that the authors may have misquoted some of the numbers in the original article.

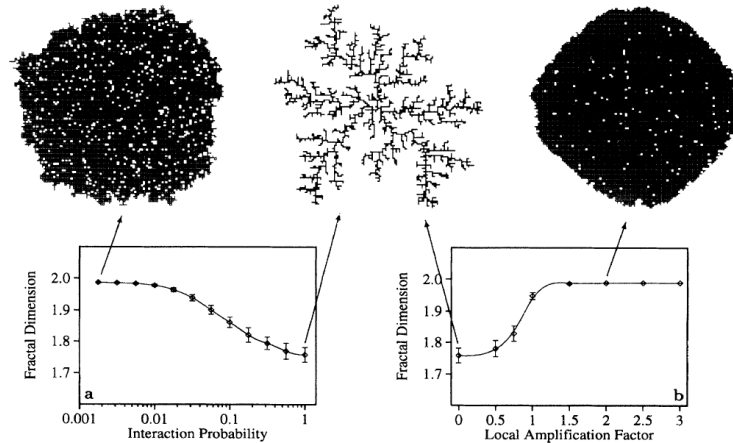


Figure 2.30: Gazit et.al. suggested two different ways that a gradient sensitive growth can be modified into producing uniform capillary beds rather than perfusion-limited aggregate patterns. The structure is grown from a seed in the middle, all network sites exposed to growth factor particles will with some probability, p_i be recorded as 'hit' and grow at the end of the time step. Growth factors are produced at distant sites (hypoxic model). Left: A low interaction probability between the growth factors and the growing vasculature would cause the growth factor levels to increase throughout the tissue and mask the diffusion field by a uniform gradient. Right: If the growing structure itself produces growth factors in response to the diffused growth factors, then this local amplification would, provided it is large enough, cause the formation of a compact uniform vasculature. [17]

2.4.4 Invasion Percolation Tumour Vasculature Model

Gazit et.al. 1995 [17]—Capillary Beds and Fractal Scaling Tumour Vasculature by Modified Invasion Percolation.

Gazit et.al. observed that the arteriovenous networks' fractal characteristics (in two dimensions) are similar to that of diffusion-limited aggregates. This is in accordance with the molecular models of angiogenesis, where growth factors diffuse from distant sites to the network and initiate growth, see figure 2.5. In experimental set-ups, however, vascular growth occurs at the capillary or postcapillary level. These structures grow like a compact mesh rather than a tree-like structure, contrary to what would be expected from diffusion limited growth. Finally, it was shown that tumour vasculature exhibit fractal characteristics similar to that of random and non-trapping invasion percolation clusters.

A simple model for network formation was proposed in which the first two observations could be brought together in one of two ways. A low interaction probability between the growing structure and the growth factors would cause the growth factors to arrive faster than they are removed, causing a near uniform diffusion field to form. The other way is by introducing a local amplification of the growth factor gradients. This can be achieved biologically by the autocrine release of growth factors from the growing structure, i.e. a positive feedback system. Both ways are able to transform a hierarchical arteriovenous network into a compact capillary mesh by adjusting a single parameter, see figure 2.30. The latter model will, however, initiate a much faster growth (mass/time) for compact networks and is therefore regarded as the best of the two models, see 2.31–left.

The networks' fractal similarity with a percolation cluster introduces the idea that the growing structure is disturbed by some locally random perturbation. The local perturbation is hypothesized to be inhomogeneity in the extracellular matrix. The fractal dimensions of networks grown for different fractions of usable growth sites is shown in figure 2.31–right. In this model the formation of different

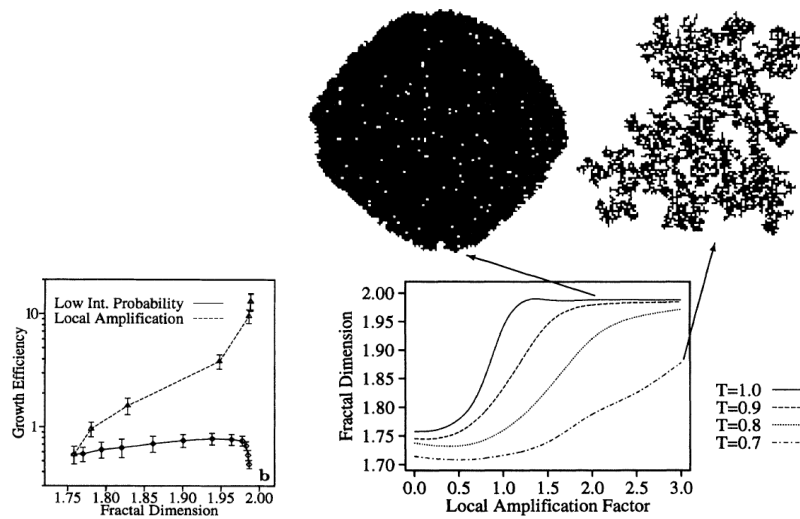


Figure 2.31: Left: The two suggested possible ways of creating a normal capillary bed in the presence of a diffusion field are shown to facilitate different growth efficiencies (mass/time) for compact growth. The local amplification model is chosen as the most appealing due to its much higher growth efficiency. Right: To produce tumour-type vasculature random values are assigned to the sites resulting in these curves for different random percolation thresholds (T). At low local amplification factors the model is insensitive to the percolation parameters and produce similar patterns/dimensions, at high amplification, however, a compact capillary structure is transformed into a percolation cluster-type network. (Adapted from [17])

types of vascular networks are dependant on two parameters only, the local amplification factor and the matrix inhomogeneity.

Baish et.al. 1996 [2]—Invasion Percolation Tumour Vasculature Model

Based on the observations of Gazit et.al [17], that the fractal dimensions of two-dimensional networks were similar to those of invasion percolation clusters, Baish et.al. [2] developed an invasion percolation-based network model for angiogenesis. Invasion percolation offers a simple rule-based way of generating network structures. This was used as an alternative to deterministic geometrical models which were unavailable as they require detailed anatomical data.

The interpretation of tumour angiogenesis as an invasion percolation process implies that the growth occurs in response to a locally random heterogeneity in the tumour, rather than a global response to physiological stimulus. The locally random property may be either biochemical or mechanical in origin. The correspondence of the fractal dimensions of tumour vasculature and invasion percolation does not prove that that these processes are related, but the authors conclude that it is strongly suggestive.

Percolation networks were used to calculate the surrounding oxygen levels, see figure 2.32. The networks were generated by an invasion percolation algorithm that is stopped when the desired occupancy is reached (not when the outlet point/side is reached as in the invasion percolation models discussed in section 2.3.10). In lack of detailed information on the diameter of tumour vessels, the assumption was made that all vessels are of identical diameter. Furthermore, a Poiseuille flow of constant-viscosity blood through rigid, impermeable vessels is assumed.

This is a gross oversimplification of what is known today about these parameters. In sections 2.1.6 and 2.2.2 (transient local hematocrit changes), all of these assumptions are shown to be wrong. The

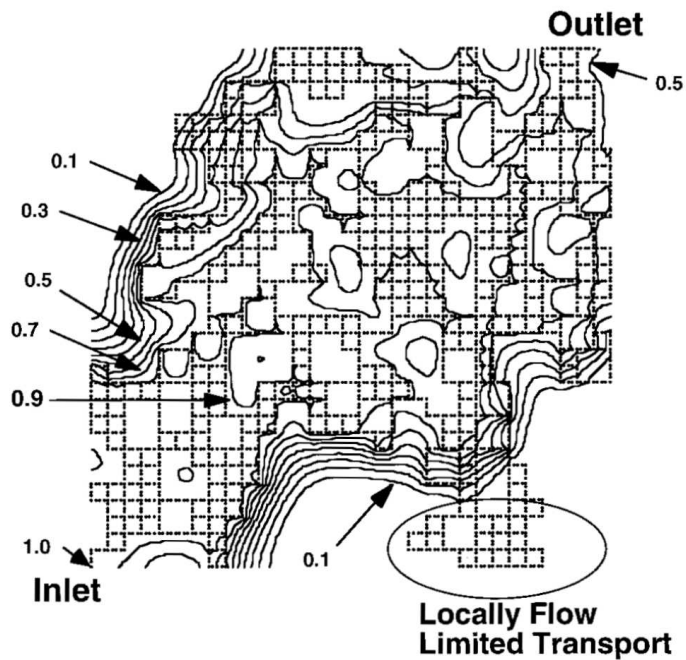


Figure 2.32: A 32×32 bond invasion percolation backbone formed on a square lattice and iterated until the cluster reached the desired 60% occupancy. The contours of the relative oxygenation ($pO_2/pO_{2\text{arterial}}$) is superimposed on the network. The inlet oxygenation is 1, and the outlet was found to be 0.57. In spite of the fact that less than half of the inlet oxygen is consumed by the network, there exists vascularized regions without blood flow or oxygen, see lower right hand corner. (Reference [2])

construction of simple models is all the same necessary, both to make it mathematically solvable, but also to provide a simple example of the potential of this kind of modelling to simulate real tumour characteristics, such as oxygen distribution patterns. Furthermore, the authors show that the introduction of more realistic assumptions would only increase the heterogeneity of the flow.

2.4.5 Cellular Automaton Tumour Vasculature Model

Bartha, Rieger and Lee have recently (2006) proposed a hybrid probabilistic cellular automaton model [4][29]. This is a Monte Carlo simulation defined on a square lattice in discrete time and space. Cellular Automata refers to a way of generating patterns, in which each cell is given a value in the next iteration based on the value of the neighbours in the previous iteration.

The Model

Each lattice point is defined by the variables in table 2.7. A set of six events is set up in figure 2.33. Each event is dependent on both a logical condition, as in normal cellular automata, and a probabilistic parameter determining the rate at which the event occurs. In this way many of the experimentally observed phenomena are included in the model. The tumour starts out as a small mass on a regular grid, co-opting vessels as it grows. The tumour produces growth factors and the vessels supply oxygen at a constant rate. Oxygen and growth factor distributions are approximated by a piecewise linear and normalized form, reaching out from its source to a specified maximum diffusion radius. Each vessel is specified by a diameter and carries a hydrodynamic blood flow exerting a shear force upon the vessel

Variable	Range	Description
e	0/1	endothelial cell or vessel absent/present
e_r	$\in [0, r_{\max}]$	radius of vessel
e_Q	≥ 0	blood flow rate through vessel segment
e_f	≥ 0	shear stress on vessel wall
t	0/1	tumour cell absent/present
t_{uO}	$\in [0, T_{\max}]$	time of tumour cell in underoxygenated state
O_2	≥ 0	oxygen concentration field
GF	≥ 0	growth factor concentration field

Table 2.7: Each lattice cell in Bartha et.al.'s model is defined by these variables at site \mathbf{r} at time τ . [4]

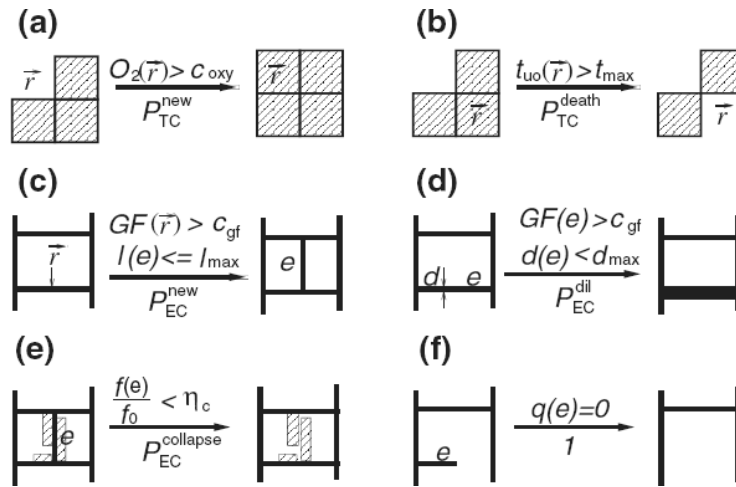


Figure 2.33: Cellular Automaton rules for Bartha et.al.'s growing tumour vasculature model. The expression over the arrows represents the cellular automaton rule. The number below the arrow is the probability that the event occurs if the rule is fulfilled. This is a hybrid model. (a) If the oxygen concentration is higher than some critical value at a given site, then that site becomes occupied by a tumour cell. (b) If a tumour cell is in an underoxygenated state longer than t_{\max} , it is removed. (c) Vessel sprouting begins at site \mathbf{r} if the growth factor concentration is greater than some threshold and the distance to the nearest vessel is smaller than the maximum sprouting distance. (d) Vessel dilation can occur when the growth factor concentration is higher than the threshold and the vessel is not already at maximum diameter. (e) A vessel can collapse if more than 80% of its surface area is covered by tumour cells and its shear force drops below a threshold. (f) All uncirculated vessels collapse.

walls. The flow is assumed to be incompressible, laminar, and stationary, consequently the flow can be calculated by Poiseuille's law, where the blood pressure in the nodes (vessel junctions) is computed using Kirchhoff's law. With these assumptions, the vessel flow, shear force, and gradient fields can be calculated at each time step and the network updated according to the rules and probabilities.

Model Results

With this model both two- (figure 2.34) and three-dimensional (figure 2.35.I) tumours have been simulated. In figure 2.35.II the radial (or angular) distribution of tumour cells and vasculature specific parameters throughout three-dimensional tumour-simulations are shown. The fractal characteristics of this network model have been calculated. Two dimensional simulations have a fractal dimension of $D = 1.85 \pm 0.05$, close to the measured dimension of 1.89 ± 0.05 . In three-dimensional simulations a dimension of $D = 2.52 \pm 0.05$ have been measured. As indicated by figure 2.35, the network is by no means uniform and most parameters vary with the distance from the tumour centre (the ∇P varies with θ as well, due to the direction bias of the model). This is true for the fractal dimension as well, which when measured for the peritumoural plexus exclusively, measured $D^{\text{periphery}} = 1.60 \pm 0.05$ in the two dimensional case. In the three dimensional case it decreases from 2.24 in the outer periphery ($125 \leq R \leq 145$) to 1.68 further in ($65 \leq R \leq 85$).

This vasculature model is able to generate vasculature with fractal dimensions close to the experimental values without incorporating any locally random variable. Furthermore, they make notice of the notion that, although the tumour network dimensions correspond to random percolation in both two- and three dimensions, they do not correspond to invasion percolation. The authors have, however, failed to realize that there are two types of invasion percolation. Non-trapping invasion percolation is in the same universality class as random percolation in both two- and three-dimensional systems (see table 2.3). Nevertheless, this model removes the need for hypothesizing that tumour vascular morphology is determined by local substrate properties such as the extracellular matrix.

Furthermore the authors emphasize that tumours usually have an elevated microvascular density near its edges, where it is growing, but a reduced density in its core. They propose that the vascular network is driven to criticality, not by the vascular growth process, but by the network remodeling process that causes this reduction in vessel density.

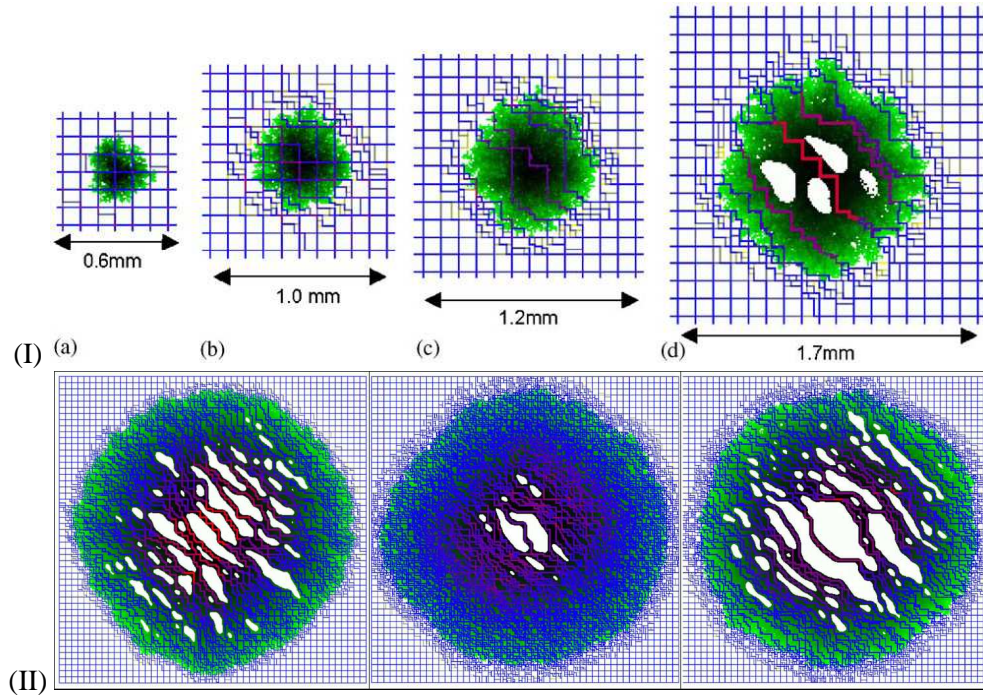


Figure 2.34: Two-dimensional networks produced by Bartha et al.'s model of angiogenesis and tumour growth. The direction bias of the vessels is a result of how the critical shear force relates to the flow, which again relates to a vascular pressure gradient. The blood pressure is set to give a homogeneous flow distribution in the original network. The boundary pressure is P_{\max} in the top left corner, P_{\min} in the bottom right, and half the difference in the two remaining corners with a linear distribution in between. (adapted from [4].)

(I) The time evolution of Bartha's tumour growth model in two-dimensions at time steps 1, 50, 100 and 200 for figures (a-d) respectively. The blue grid far from the tumour is the initial capillary bed. The colour of the blood vessels indicate their flow, blue is normal, red is high and yellow indicates low flow. The thickness of the lines indicate vessel diameters. The tumour cells are coloured from light to dark green, darker colours indicating higher age. White regions are empty sites, empty sites inside the tumour indicate necrosis. (a-b) New vessels are being formed, tumour grows. (c) Vessels inside the tumour have started to collapse, thicker vessels have started to form. (d) Necrotic areas have formed due to large avascular areas. Thick vessels run through the tumour, the tumour periphery is still characterized by an increased microvascular density.

(II) (Left): The state of the system in (I) after $t=1000$ time steps. The size of the system is 5.1 mm. (Middle): R_{GF} is increased from $200\mu\text{m}$ to $400\mu\text{m}$, but all other parameters are as in (I). (Right): All parameters are the same as in (middle), except the critical shear stress which is increased from 0.5 to 0.7. Whereas the morphology in (middle) is relatively stable even for high collapse probabilities, small changes in the critical shear force causes a rapid increase in collapses, producing large necrotic regions and decreased microvascular density within the tumour.

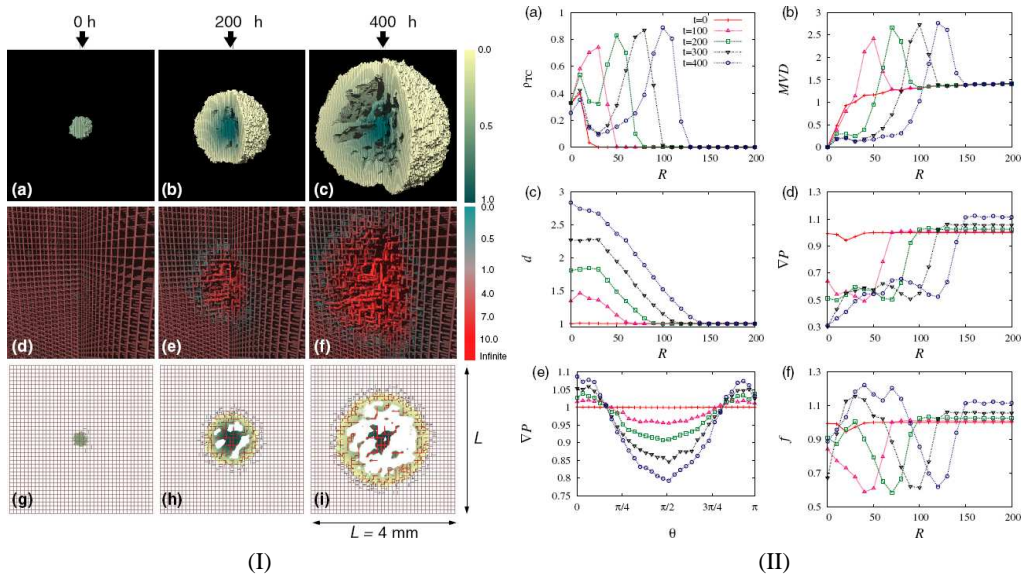


Figure 2.35: **(I)**: The time evolution of the tumour or vessel system at $t=0, 200$ and 400 . Top: tumour cells. Middle: vasculature. Bottom: cross section of tumour. **(II)**: Data averaged over all sites with same R (or θ). (a) Tumour density $\rho_{TC}(R)$. (b) Microvascular density $MVD(R)$. (c) Vessel diameter $d(R)$. (d) Blood pressure gradient $\nabla P(R)$. (e) Blood pressure gradient $\nabla P(\theta)$. (f) shear force f . The colours, red, magenta, green, black and blue represent the tumour at $t=0, 100, 200, 300$ and 400 respectively.

Chapter 3

Materials and Methods

3.1 CD-34 Image Processing

Histological cross sections of four breast carcinomas were included in this investigation. They were all obtained after informed consent by the patients.

3.1.1 Scanning of Images

The images are scanned with a Leica DFC320 digital camera for microscopy. The sensitive area of the sensor is 7.2×5.35 mm, the full resolution is 2088×1550 pixels (3.3Mpixel) and the pixel size is $3.45 \times 3.45 \mu\text{m}$. The camera is attached to the microscope with a $\times 1.0$ video adapter. The magnification is determined by the microscope alone, resulting in the field of view values shown in table 3.1 for each of the magnifications. The images were exported as 24-bit RGB files in tiff format.

Magnification	Field of View	Field of View per Pixel
$\times 1$	7.2mm x 5.35mm	$3.45 \mu\text{m} \times 3.45 \mu\text{m}$
$\times 25$	$288 \mu\text{m} \times 214 \mu\text{m}$	138 nm x 138 nm
$\times 50$	$144 \mu\text{m} \times 107 \mu\text{m}$	69.0 nm x 69.0 nm
$\times 100$	$72.0 \mu\text{m} \times 53.5 \mu\text{m}$	34.5 nm x 34.5 nm

Table 3.1: Field of View at Different Magnifications

3.1.2 Applying Thresholds to Image Sections

This part of the process is of high importance to the outcome. Poor thresholding techniques may result in high levels of false positives and/or negatives, or most likely a moderate mixture of the two. Consequently, it has been deemed important to develop not only a sufficient routine, but a mathematically defined one that does not require the scientist to make good choices of parameters on an image to image basis. This has the added benefit that a script may automatically process images without the need for manual labour. The results, however, should be checked in case any images somehow have eluded the routines. A routine based on the following steps has been used on the CD-34 sections. Note that this method has been chosen with these sections in mind, and will most likely have to be adapted for other purposes. The images are processed using the following steps:

1. The image is transformed to grayscale.

2. The histogram is expanded by a non-linear function working on a pixel-by-pixel basis throughout the image, increasing the contrast on the middle half of the histogram.
3. An average filter subtraction is applied to correct for uneven background light intensity. A copy of this image is kept for further processing.
4. Edge detection using Sobel's method estimates the image gradients in the x- and y-direction. The magnitude of the slope is used for further processing.
5. A grayscale threshold applying Otsu's method to compute the global threshold is applied to a copy of the image from before the edge detection and the result of the detection.
6. The two binary images are combined to give precise information on the edges, while keeping information on the highly stained uniform areas on the interior of vessels.

Step 1 – Grayscale Conversion

The standard rgb¹ to grayscale mapping is based on the colour sensitivity of the retina. It biases green, primarily at the expense of blue.

$$[\text{Grayscale}] = [0.2989 \quad 0.5870 \quad 0.1140] \cdot \begin{bmatrix} \text{Red} \\ \text{Green} \\ \text{Blue} \end{bmatrix}$$

A different approach is taken here. Because the positive CD34 stains are coloured in a red hue, in effect the lack of green and blue, the red channel is disregarded. Figure 3.2 shows the different colour channels of the sample case at this point in the process. Red offers the least contrast and blue the most. A crop from the original image is enlarged as well, showing the difference in resolving power for these channels. The green channel offers about twice the linear resolution as that of the other colours.² Image sensors are designed to resolve this channel best, as a response to our eyes' affinity to the colour. Most sensors use the Bayer pattern³ shown in figure 3.1. An interpolation algorithm is then applied to the raw-data to produce a regular rgb-image, interpolating the missing two colours from nearby sensor elements into the current to achieve an rgb-image of almost the same pixel-count as the number of individual photosensitive areas in the sensor. This is also why, although not clearly shown in this figure, the blue layer is shifted one pixel up and the red layer one pixel to the left, both relative to the green layer.

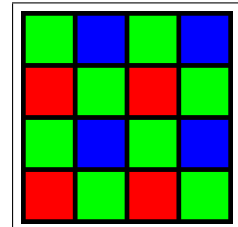


Figure 3.1: The Bayer image sensor pattern.

The aim of the grayscale conversion is to differentiate between the stained vessels and the surrounding tissue. Although the blue layer appears to have slightly better contrast, the difference in resolving power is so great that the green layer is used by itself. If, in contrast, the standard conversion rule is used, excluding the red channel, then the green channel would account for 83% and the

¹Rgb-image, i.e. a colour image stored as a set of red, green, and blue intensity images.

²This is visible because these images have been exported in a lossless tiff format. If on the other hand jpgs were used, then camera digital filters would mask this, creating the appearance that these colour channels contain information as highly detailed as the green layer, although in reality the colour values are interpolated from the nearby region

³It is hard to organize a square lattice in a way that does not bias one of the colours. The colour response of the retina, makes this a good choice.

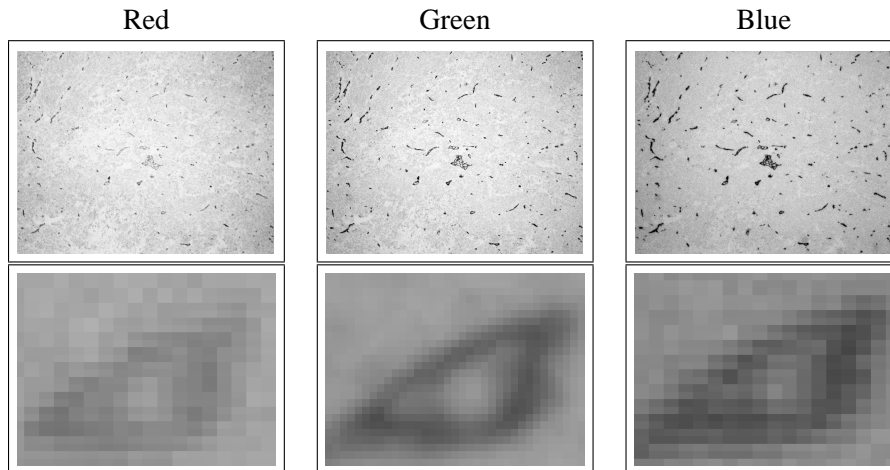


Figure 3.2: Top: The three colour channels of the image at step two. The red channel offers the least amount of contrast. The blue channel has slightly more contrast than the green, and is a bit darker. Below: Enlarged crops of a single vessel from the original image. This clearly shows that although blue offers more contrast, green has four times the resolving power.

blue for 17% of the information in the grayscale image. Even at this ratio the loss of detail appears to outweigh the gain in contrast. Consequently the following rule is used:

$$[\text{Grayscale}] = [\text{Green}]$$

This implementation suffers from one shortcoming, it is prone to false positives in highly stained blue regions. Given the low background of the sections this is not a big problem, and the 17%, or even 33% (the effective blue sensor area ratio when the red channel is excluded), is not necessarily enough to eliminate the problem. One way to compensate for this, although not applied in this work, would be to first use the green channel to identify possible vessels, and then at the end compare the blue colour levels in these regions to remove false positives.

Step 2 – Histogram Expansion

An image overlay function is defined as the combination of two images, on a pixel by pixel basis, treating each colour layer independently, in the following way: The background is inverted,⁴ multiplied⁵ by two times the foreground and added to the background again.

$$I = (B * (B + (2 * F * (255 - B)) / 255) / 255);$$

This is performed five consecutive times, each time using the original image as the foreground and the result of the previous overlay as the background, or, in the case of the first run, the original image as both foreground and background. This operation reduces the intensity of dark areas and increases the intensity of light areas in a non-linear way, see figure 3.3. The operation is carried out until at least 0.1% of the total area is either above 90% or below 10% of the maximum luminosity.

⁴Note that the inverse of an image is the image with opposite luminosity, in effect $|\text{Image} - 255|$ in an 8-bit integer colour representation, and $|\text{Image} - 1|$ in the floating point colour representation

⁵In the 8-bit integer colour space multiplication is defined as $I = B * F / 255$ to preserve the pixel values within the valid range (0, 255). This is not necessary for floating point colour representation (0, 1).

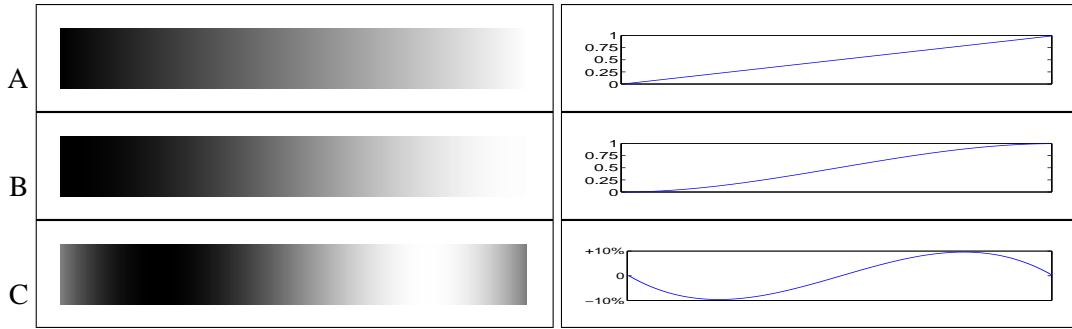


Figure 3.3: The effect of the overlay function. A: The full 8-bit grayscale spectre from 0 intensity to 255. B: The result of the overlay function. C: The effect of the overlay-function (difference between A and B). Dark areas have become darker and bright areas brighter. The gray areas in each end and the middle change little during the transformation.

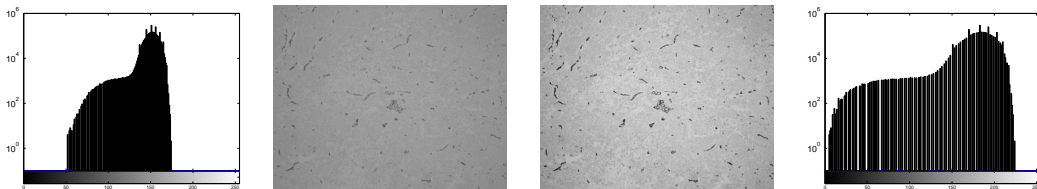


Figure 3.4: Top: The overlay function applied until the histogram hit the edges of the colour space, in this case three times.

Step 3 – Average Filter

For each pixel, the average value of a 50 pixels wide neighbourhood around it is calculated, pixels outside the image edge are disregarded (not treated as zero). The image intensity is then subtracted from the local average image. Furthermore, to get an image covering the full range of the histogram, the minimum intensity value is subtracted from the image before normalizing to the full range of the intensity space (0,1). See figure 3.5 for an example of the result. The average filter is handled by a MATLAB function.

```
% I is the intensity image matrix;
ws = 50;
mIM = imfilter(I,fspecial('average',ws),'replicate');
sIM = mIM-I;
sIM = (sIM-min(min(sIM)));
sIM = sIM./max(max(sIM));
```

Step 4 – Edge Detection

Edge detection is performed using Sobel's method. The image gradients are approximated by the convolutions

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \quad \text{and} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

These directional gradients can be combined to give the gradient magnitude using

$$G = \sqrt{G_x.^2 + G_y.^2}.$$

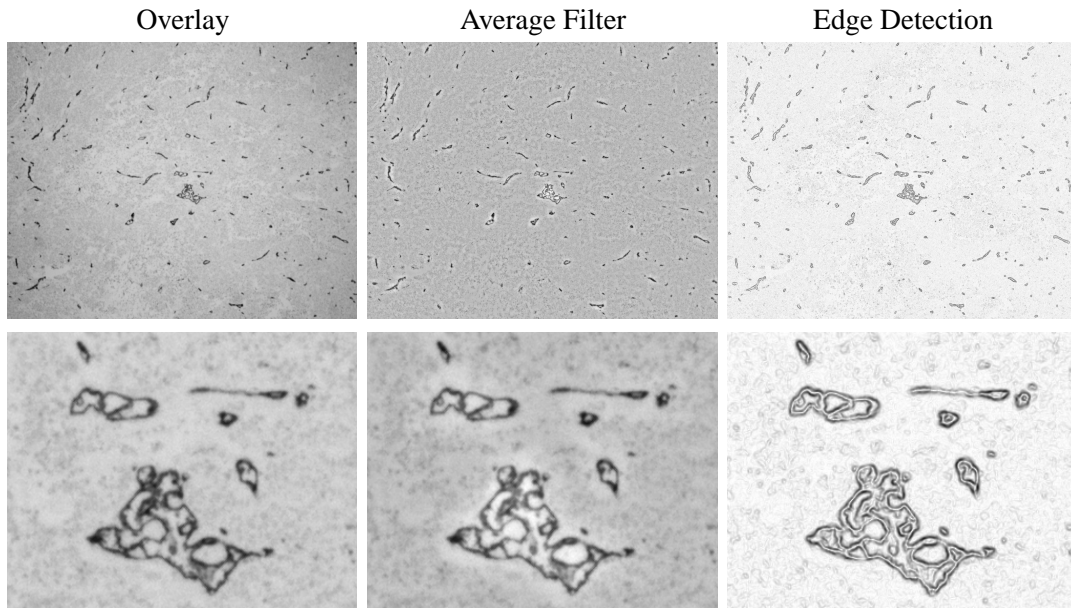


Figure 3.5: The resulting images after the overlay, average filter and edge detection steps.

See figure 3.5 for an example of the result. MATLAB provides a function to handle this operation.

```
% I is the image matrix, gv and gh are the vertical
% and horizontal gradient respectively
[Isobel, thresh, gv, gh] = edge(I,'sobel');
grad = sqrt(gh.^2+gv.^2);
```

Step 5 – Global Threshold

A global threshold is performed to one copy of the image at the end of step four and one at the end of step five. The global threshold is determined by a MATLAB function using Otsu's method, which chooses the threshold to minimize the intraclass variance of the black and white pixels. The result, superimposed on the original RGB-image, is shown in figure 3.6.

```
% global threshold of the gradient image
level = graythresh(grad);
Ibw=im2bw(grad,level);
```

Step 6 – Combining the Thresholds of Gradient and the Intensity Image

The black and white images from step 5 are combined by accepting any pixel tagged as a vessel from either one of the images as a vessel pixel. This is done using MATLAB's logical *or* operator, `|`.

```
% Ibw is from the gradient image and Ibw2 is from the intensity image.
Ibw = Ibw | Ibw2;
```

Step 7 – Filling in Closed Lumens and the Removal of Exceedingly Small Positive Areas

Any vessels containing an empty interior is filled in to give an image of the entire vessel lumens including unstained interior, rather than just the edges. Before this last step, a series of three morphological *closings* are performed. A morphological *closing* is a *dilation*, the adding of extra pixels



Figure 3.6: Global Threshold: The black and white result obtained from the intensity image at step 3, and the gradient image at step 4, superimposed on the original image. To the right the perimeter of the final result is shown. Uncropped final images (not just the perimeter) are shown in figure 4.14

around the boundaries of all objects, followed by an *erosion*, the removal of all pixels at the boundaries of the objects. The closing fills in any single pixel openings in between the object borders. This causes nearby objects (separated by one pixel) to merge. This may cause separate vessels to merge if they are very close to each other. In the vast majority of cases such gaps will be within the same vessel, and they have become separate regions due to insufficient staining and/or shortcomings of the algorithm. The merging of two vessels located very close to each other is considered better than the splitting of single vessels into multiple.

All holes, i.e. empty pixel regions unconnected with the image edge are filled. Because the image gradients are centred on the boundaries of objects their outer perimeter is a little greater than that of the vessel itself. To account for this an *erosion* is performed. Finally to remove specks of noise in the image all objects with less than 64 pixels are removed. 64 pixels is equivalent to about 20 ppm of the entire image area, or in the case of the $\times 25$ magnified images, a square covering $1.2\mu\text{m}^2$. This removes most of the false positives, due to noise. It may remove very small positively stained regions as well, thus creating false negatives, however, such stains must be very small in order to be removed this way, justifying the operation.

```
Ibw = bwmorph(Ibw, 'close',3);
Ibw = imfill(Ibw,'holes');
Ibw = bwmorph(Ibw,'erode');
Ibw = bwareaopen(Ibw,64);
```

3.2 Basic Image Statistics

Four different parameters were obtained directly from the black and white images. The first was the number of vessels, found by counting the number of individual 4-connected regions in the image. The second is the mean area of the vessels and the standard deviation, measured in both pixels and μm^2 . The relative vascular density, that is (stained area and lumens) / (total area), is calculated.

```
L=bwlabel(BWimage);
area = regionprops(L,'area');
area = cell2mat(struct2cell(stats));
meanOfArea = mean(area);
stdOfArea = std(area);
numberOfVessels = numel(area);
relativeVascularDensity = sum(sum(BWimage))./numel(BWimage)*100;
```

Finally, histograms of the non-vessel pixels' distance to the nearest vessel are calculated. The distances are calculated by adding together successively dilated bw-images of the vessels.

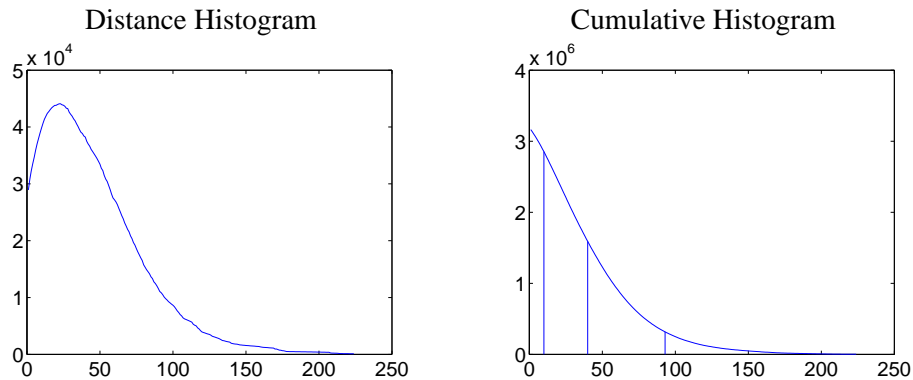


Figure 3.7: The distance histogram displays the number of pixels (y-axis) at a given distance (x-axis) from the closest vessel. The cumulative histogram shows the total number of pixels that are *further* away than the given distance. The vertical bars from left to right shows the distances at which at least 90%, 50% and 10% of the non-vessel pixels respectively, are further away. All distances are measured in the units of the pixel width which is 138nm.

```
D = int16(BWimage);
while any(any(D==0))
    C = bwmorph(C,'dilate');
    D = D + int16(C);
end
Dmax = double(max(max(D)));
distanceToNearestVessel = -D+Dmax;
```

Resulting histograms from one of the case images is shown in figure 3.7. From the cumulative histogram three parameters are extracted, namely the distances at which the histogram is at 10%, 50% and 90%. This corresponds to the distances where the given percentage is the maximum number of pixels that are further away.

3.3 Syntactic Structure Analysis

Syntactic Structure Analysis (SSA) is here used in the context of reference [43] and concerns the parameters derivable from the Voronoi Diagram, Gabriel's Graph (GG) and the Euclidean Minimum Spanning Tree (EMST), see figure 3.8. From the set of parameters investigated, in each of these methods, histograms are obtained and the mean value, standard deviation, skewness and kurtosis are calculated. All sizes are in the unit of *pixels*. The SSA-graphs themselves are calculated from a list of the *mass centre* of each vessel. The shape and size of the vessels do not effect the SSA, only their number and relative positions. The mass centre images are obtained through a black and white morphological thinning. The complete MATLAB code used to calculate the SSA-parameters is included in appendix B.2.

```
massCentreImage = bwmorph(blackAndWhiteImage,'shrink',inf);
```

3.3.1 Voronoi Diagram

The Voronoi Diagram is a set of polygons covering an area around each vessel. The Polygons are drawn so that the area inside a polygon is closer to the vessel inside the polygon than any other vessel.

The polygons at the edge of the image do not have a proper boundary on the outside and have been removed. Three histograms are obtained from this graph, the area distribution of the polygons, their shape, and their form. The form of the polygon is defined as $4\text{area}/\text{perimeter}^2$ and the shape as the smallest polygonal diameter divided by the largest. A polygonal diameter is defined as the distance from one corner to another corner that is not its direct neighbour.

3.3.2 Gabriel's Graph

The shortest distance (branch) between two nodes (vessel mass centres) uniquely defines the diameter of a circle. Gabriel's Graph is defined as the set of all branches spanning empty circles. If there is another node in a circle spanned by the branch, then it is not a part of Gabriel's Graph. Only the branches to the closest nodes in each direction can possibly fulfil this criterion. To speed up computation time the Delauney Triangulation is first calculated. It contains the set of branches not intersected by a shorter branch. Gabriel's Graph is a subset of the Delauney Triangulation.

From this graph the histograms of the branch lengths, the number of branches per node and the distances to the nearest and the furthest neighbour are calculated.

The branches per node parameter is reduced as a consequence of the edge nodes having fewer neighbours. If desired, this could be accounted for by not including the nodes removed from the Voronoi when counting the branches per node. The other parameters may also be affected by the edge. This will happen if there is a vessel beyond the edge that would remove a branch from the network. This can be corrected by not considering the branches spanning a circle which in part cover a region outside the map. For a single node this would cause some of its branches to be kept, while others are excluded for the purpose of calculating the branch lengths. No action is taken towards reducing the edge effects. The reason for this is twofold, the complexity involved is high and the effects are expected to be small, certainly much smaller than that of the Voronoi, where polygons by the edge may diverge to infinite areas. The size of the edge effects have not been investigated. This applies to the EMST as well.

3.3.3 Euclidean Minimum Spanning Tree

The Euclidean Minimum Spanning Tree is the set of branches so that all nodes are connected by the tree in the configuration providing the minimum possible total length of the tree, according to the euclidean norm. The EMST is a subset of Gabriel's Graph, and the same parameters are obtained, as from GG. The nearest neighbour data are the same for the two graphs, as all the shortest GG-branches are included in EMST. The mean number of branches per node is always $(2 \cdot n - 1)/n$, where n is the number of nodes, making this particular parameter redundant. The other three histogram parameters from the branches per node histogram are, however, not.

3.4 Linear Fitting and Graphs from the Fractal Algorithms

3.4.1 Linear Fitting

All of the fractal analysis methods uses a linear fit to a double-logarithmic plot to identify the fractal dimensions. The curves are, however, not linear across the entire range of the plot. The analysis of images restricts the curves from two sides. On one end, the investigated scales approach the size of the image, and on the other the size of a single pixel. Furthermore, natural fractals need not have fractal behaviour across all scales, and may consist of different power-law scalings at different scales.

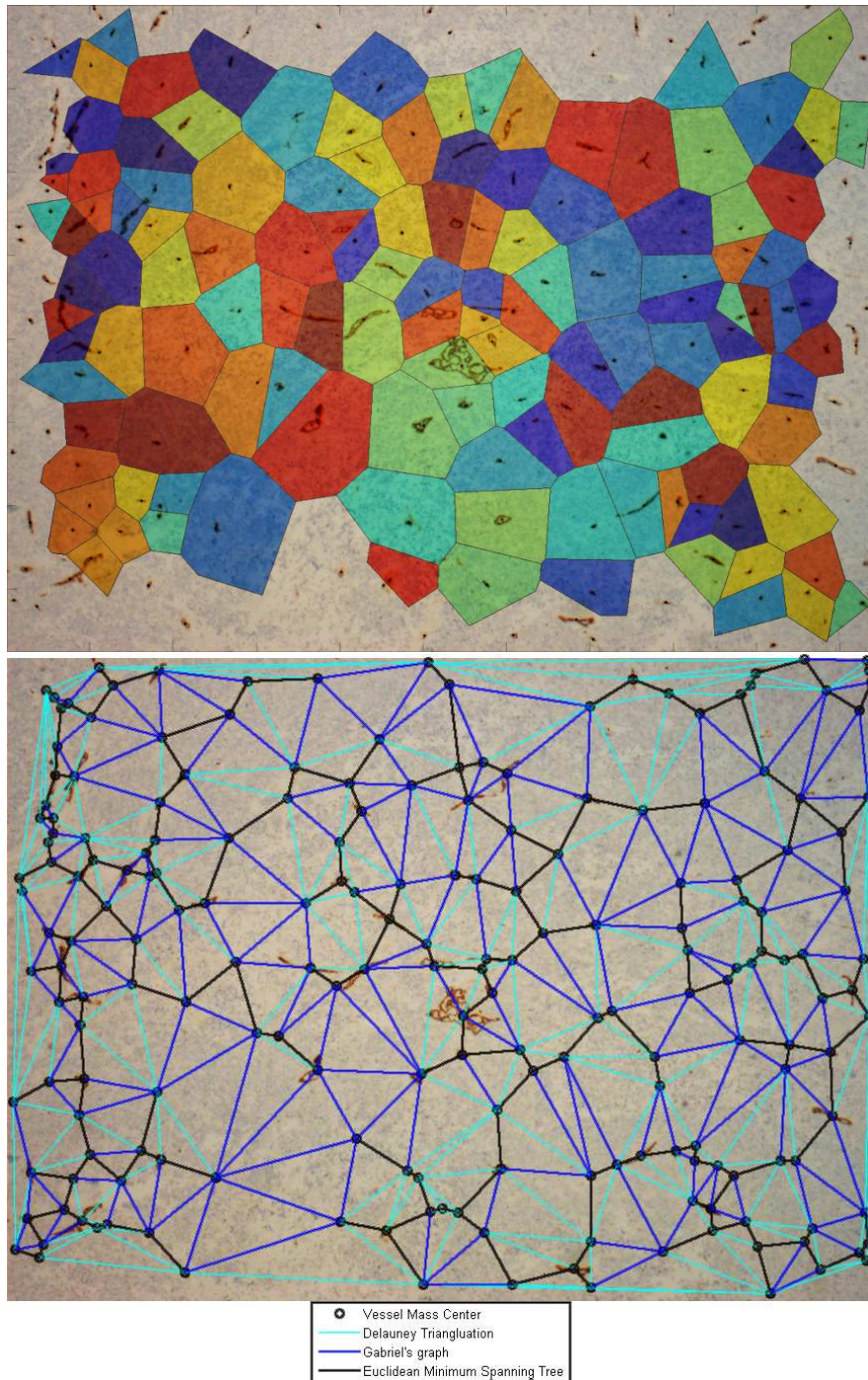


Figure 3.8: Top: *Voronoi Diagram* of a CD34 immunohistological section (background, case 2). The vessel mass centres are drawn in as black dots. The diagram polygons are randomly coloured. The *Voronoi Diagram* is the complementary graph of the *Delauney Triangulation* (below), the polygon surrounding a vessel shares one side with each other vessel it is connected to by the *Delauney Triangulation*. Below: The vessel mass-centres in this section are marked by black circles. The black lines form the *Euclidean Minimum Spanning Tree* which is a subset of *Gabriel's Graph* (blue), which is a subset of the *Delauney Triangulation* (cyan). Consequently all the lines (cyan, blue and black) are part of the *Delauney Triangulation*, but only the black lines are a part of the Minimum Spanning Tree.

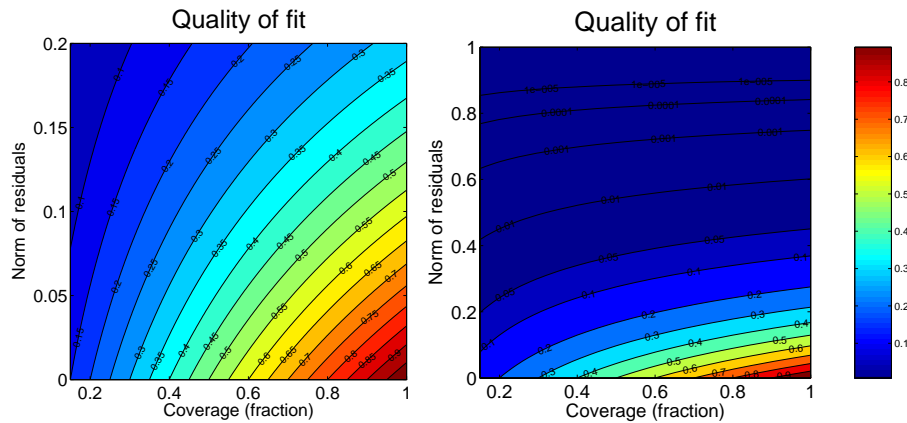


Figure 3.9: Contour lines of the *Quality of fit* function.

To find the linear areas in the image a criteria is set for what is considered a good fit. Linear fits are then calculated across all continuous regions of the graph, and the best fit is selected according to the criteria. Remaining areas of the graph, on the condition that they are larger than some minimum size, are then recursively scanned for more linear areas.

The fit is a *least squares fit* and the norm of the residuals is used as an indicator of how accurate the fit is.⁶ In addition to accuracy, the size of the linear region is used as the second component in determining how good a given fit is. The size is measured as the fraction of the total logarithmic with the linear portion spans. The quality of the fit is then defined as

$$Quality = Width \cdot (1 - NormOfResiduals)^5.$$

The width is here weighed linearly, in effect a region twice as wide is considered twice as good. The Norm of Residuals is subtracted from 1 so that the good fits will have a value close to one, and bad fits lower, possibly negative values. These values are then taken to the power of five to increase the differences. Good fits are close to one, and relatively little changes. Small values on the other hand drop considerably. The choices of factors and powers in this formula represents a trade off between the importance of the two.

A function recursively searches through all valid fits. These include all subregions with at least five data points and spanning at least 15% of the logarithmic width. The function then selects the best fit and goes on to search all remaining subregions spanning at least 20% of the logarithmic width using the same search criteria for these as well.

3.4.2 The Graph Set-Up

Two examples of the graph set-up is displayed in figure 3.10, one of the familiar cos-function, slightly shifted to the right, and one actual result from the analysis of a histological section. The curve itself is plotted in black and the fits are plotted onto the same figure. To make the areas used by the algorithm more visible, colour patches matching the linear fits are drawn beneath the curve (not the fit). The colours themselves do not contain any information.⁷ In addition the gradient is plotted in a separate

⁶The lower the norm is, the better the fit. An exact fit has a norm of 0.

⁷They are chosen from the hue-colour representation based on the total number of regions in the plot, with the aim to achieve a decent amount of contrast without getting colours that do not match next to each other, and to avoid the use of complementary colours in the same graph.

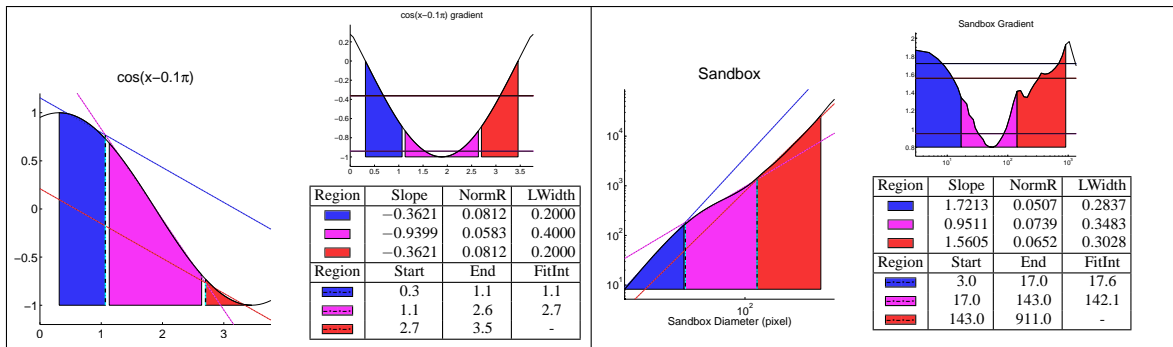


Figure 3.10: Two examples of the linear fit graph set-up. To the left the familiar cosine function is plotted from 0 to 1.2π and shifted a bit to the right. In the right hand plot the result from a sandbox-dimension analysis of a histological section slide is shown. A description of the different parts of the graph is found in section 3.4.2

smaller figure beside it. Here the linear fits are represented by horizontal lines matching the slope of the fit. If more than one fit has the same slope, as is the case here, one of them will be plotted on top, hiding the others from view.

Beneath the gradient a table shows the slope of the linear fits, the norm of residuals of the fit labelled as *NormR*, and the fraction of the logarithmic area covered by the fit, labelled as *LWidth*. The bottom part of the table shows the borders of the region. Many times the transition from one scaling pattern to another is adequately represented by the start and end positions of each region. Some times, however, especially if there is an open region between two linear, the intersection of the linear fits may better locate the transition. At other times this method gives misleading results placing the transition outside the gap. Consequently, both approaches are necessary. The x-coordinate of the intersections are listed in the *FitInt*-column. The intersection in a given row is between the current fit and the next region, consequently the bottom cell is always empty. These intersections are represented in the main graph by vertical dashed black lines with cyan filling in the gaps. The MATLAB code used to generate these graphs is listed along with the fractal algorithms in appendix B.4.

3.4.3 Interpreting the Results

An example of the resulting curve from a fractal analysis of a histological slide is shown in the right part of figure 3.10. In this case the sandbox algorithm is used on a black and white image, showing the entire lumen of the vessels, as produced by the image processing described in section 3.1. The curve has three different power-scaling regions, all with a relatively low norm of residuals. It is wise to check the *NormR* column for a given fit because the algorithm is not set to suppress poor results, but rather to show how poor they are. There is one exception, if the best region found is smaller than the specified minimum logarithmic width, it will be removed. This width is usually 15% in these calculations. In the case of the sandbox method, the slope of the graph is identical to its fractal dimension, so no conversion is required. The highest fractal dimension is found, for the first region stretching from 1 to 17 pixels wide (the sandbox diameter). This corresponds to the vessels themselves. The vessels are, at sufficiently small scales, two dimensional surfaces. The smallest vessels are, however, although at least 64 pixels large, but small dots contributing to a reduced dimension. The avascular areas are ignored at these small sandbox sizes, as all sandboxes are centred on a vessel pixel. The linear fits of the blue and the magenta regions meet at 17.6. This, being the width of a square, corresponds to an area of 310 pixels.

The next region stretches from 17 to 143 pixels large diameters. This region has a dimension a little smaller than 1 and corresponds to the size scales in which the vessels are surrounded by avascular areas. The sizes of the vessels themselves resemble small points more closely than two dimensional surfaces.

In the third region, starting at 143 pixels wide diameter, we are at a scale where the sandboxes encompass multiple vessels. This part has a scaling of 1.56 and continues almost to the edge of the graph, though not quite.

Although the first part of the curve relates to the size of the vessels and the shape of their boundaries, this is better studied at higher magnifications. The most interesting region is by far the third, especially with regard to the fractal dimension. The transition points between the regions are of significance in addition to the fractal dimension.

3.5 Accuracy of the Fractal Algorithms

The theory of the fractal methods is provided in sections 2.3.5–2.3.8, and the MATLAB implementations are listed in appendix B.4. The quality of each of the algorithms, as implemented here, needs to be tested to acquire some sense of their accuracy.

3.5.1 Description of the Test

The various algorithms calculating the fractal dimensions need to be tested against well defined images of established fractal dimensions. The chosen shapes are the Sierpinski Gasket, the Sierpinski Carpet, previously shown in figure 2.19, a circle and a square, both in filled and unfilled versions. The accuracy of the calculation will not only depend on the algorithm, but also on the resolution of the tested figures. Although these figures are mathematically well defined, allowing coordinates to be calculated with high precision, the images that will be analysed later are raster graphics of finite resolution. Raster graphics will therefore be used in these tests as well.

In addition to the classical mathematical fractals, percolation clusters, backbones and elastic backbones will be used to test the algorithms as well. These have the advantage that they are not regular and deterministic. The Box Counting Algorithm may for instance produce the exact dimension of the Sierpinski Gasket, provided that the investigated box-sizes all are dividable by 3. In other words lucky choices of box sizes may give the impression that this method is better than it really is. On the other hand the percolation clusters have a drawback as well, precisely because of their deterministic nature. A finite cluster is never guaranteed to have the exact same fractal scalings as the infinite cluster is proven to have. A deviance from the exact value, may therefore either be because of the algorithm or the cluster. Two rather large clusters at 1024x1024 are used in this test. Many more clusters could be used. Due to the long computation times involved in calculating the backbone of large clusters (~10 hours) many smaller clusters may be preferred over a few larger, but then each cluster can be expected to deviate further from the infinite cluster values. Thus, if this method is used then a mean of all the smaller clusters should be used. This would also prevent the final result (mean of all errors) from being dominated by the percolation tests. An approach with two large clusters with roughly the same size as the other test-shapes has been used here. Furthermore, note that it is the slope of the widest linear region detected for each test and algorithm that has been used in these comparisons.

Finally, it should be pointed out that these test images are quite different from the histological images. The error estimates found here should not be directly applied to the images, although they certainly indicate the relative quality of the methods and provide a rough idea of the magnitude.

Method	Mean of Error	Std. Dev. of Error
Box Counting	0.00008	0.04287
Sandbox	0.00105	0.02300
Correlation	-0.01097	0.03231
Mass	0.00642	0.03360
Fourier	0.00458	0.05856

Table 3.2: Accuracy test of the Algorithms. The algorithms were tested on a number of known geometrical shapes as well as a random site percolation cluster, see table A.1 for the full results from the dimension calculations and figures A.2-A.7 for the power scaling graphs of each test for algorithm, and the shapes (images) themselves in figure A.1. Note that the error data for the Fourier algorithm excludes the results from the filled square and circle.

Knowledge about the sizes of the errors is useful when determining whether variations in the dimension is the result of the images' dependency on some parameter, or simply the inaccuracy of the methods used to determine them.

3.5.2 Evaluation of the Algorithms

The errors in these tests are not assumed to scale with the size of the dimension. The absolute error is therefore used in the comparisons. The result of these tests are shown in table 3.2. Note that contrary to convention, the box counting curves are plotted versus the box sizes (L), rather than their linear number ($1/L$). The effect of this in a log-log plot is to reverse the x-axis. This is done to make it easier to compare with the other methods, as the large area contrast changes are now to the right.

Local Amplitude Variations: The Fourier method and to a much lesser extent, the Correlation method, exhibits huge local amplitude variations. These Curves still have a loglog-linear global trend, but local variations obscures this. To remedy this a 10-point moving average is applied twice to the Fourier data, and a 5-point moving average is applied once to the correlation data.

Mapping From Slope to Dimension: The Fourier power law scaling, although fairly well mapped for most dimensions by the norm of the slope, cannot be mapped to fit both the filled surfaces (square and circle) and the other test-shapes. The choice is then made to ignore these two data points and use the other eleven for the purpose of this test. Furthermore this mapping is used for the fractal analysis with this method, see figure 3.11.

Computation Time: The five algorithms fall into two groups with respect to computation time. The Correlation, Mass and Fourier estimates utilises the Fast Fourier Transform (FFT) for the main part of the work load. Consequently, there are few ways to affect the computation time of these methods, with the exception of one, the FFT-method which is faster for small prime factorials. The studied images are 2088×1550 with factorials $(2^3 \cdot 3^2 \cdot 29) \times (2 \cdot 5^2 \cdot 31)$, suggesting that these routines will be somewhat faster if cropped down to the nearest number consisting of only small factors ($2s$ and $3s$), i.e. $2048 \times 1526 = (2^{11} \times 2^9 \cdot 3)$. Tests on such crops show, however, that the effect is rather small. The number of points used to draw the graphs in these methods may be freely chosen, here 100 points have been used, but this does not affect the computation time. Data is calculated for all discrete distances in the transformed images and averaged into the plotted data points.

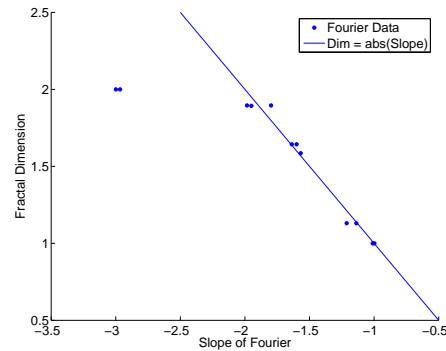


Figure 3.11: The Fourier scaling parameter β (slope of curve) versus the fractal dimension of the test images. The two-dimensional images (circle and square) will not fit into the same linear approximation as the other points.

The Box Counting and the Sandbox algorithms are different in this respect, and somewhat different from each other. The Box Counting algorithm is by far the slowest of the five, requiring six–seven times as long to compute (with thirty sampled box-sizes) as the other algorithms. Apart from scaling down the size of the image, which works on all algorithms at the expense of the information contained in the image, this method has one parameter controlling its workload, the number of different box-sizes used in the calculations. The number of points in the graphs from these two methods are the only sampled distances in the image, they do not represent averages of the surrounding region.

The Sandbox method is faster than the Box Counting Method to begin with and uses fifty sampled sandbox diameters. Furthermore, it has an additional way of reducing the workload. Both the number of the sampled sandbox diameters and the number of points sandboxes which are calculated around may be adjusted. The Sandbox algorithm used in these tests have been run with thirty sampled sandbox sizes and in images with more than 30.000 lit pixels, the number of sandbox centres has been reduced to no less than 15.000.⁸

Resolution at High Values: The box counting method has a very limited resolution at the largest box sizes. For instance, for boxes 700×700 pixels large there is only room for 3×2 boxes, so in effect, this box–size only has 7 theoretical values. Furthermore boxes at these sizes are *almost* guaranteed to contain a vessel, reducing the possible values further, zero certainly is not an option. For this reason, in addition to the slow computation speed at high box-counts, the largest box sizes used are one fifth of the shortest axis, in effect, 310×310 pixels for the histological images.

Accuracy: An overview of the results from the analysis of the test shapes is shown in table 3.2. The Sandbox method has performed best at these tests. Although the Box Counting method has the the lowest mean of error the Sandbox method has a far lower standard deviation. Of these two values the standard deviation should be considered more important. A small mean, coupled with a large standard deviation, simply indicates that the method is as likely to overestimate the dimension as it is to underestimate it. The Sandbox method may possibly be improved

⁸Note that the histological images may contain less than 15.000 points to begin with, especially the ones where each vessel is represented by its perimeter or centre of mass.

somewhat by increasing the number of sampled sandbox-sizes. Within the limits of a reasonable computation time, although much is not expected to be gained.

Different Dimensions and Fractal Transitions: Although the Sandbox method appears to be the most accurate in these implementations the other methods are not necessarily useless. All the tests were performed on fractal scaling figures. Although this may essentially be the same situation as when one is investigating two-dimensional vascular systems, or a reconstructed (or otherwise obtained) three-dimensional system, it is not the same as analyzing cross sections of three dimensional systems. The cross sections cannot be expected to exhibit fractal scaling across large regions of the log log-plot.

For this reason they may not yield the same values, keeping in mind the slightly different meaning of the different dimensions. Furthermore, the fractal dimension is not the only piece of information these methods present, they also show the transitions from one power-law scaling region to the next and at which size-scales these transitions occur.

3.6 Random Simulations

Grizzi et.al. simulated histological sections and calculated the fractal dimensions as a function of the total number of vessels in each image. They used relatively large circular dots to represent the vessels and vessel counts ranging from five to fifty in steps of five with 1000 simulations for each, see figure 2.28. The result was a curve increasing with higher vessel counts and plotted with corresponding standard deviations. The conclusion drawn was that not only the vessel count, but also the vessels relative positions contributed to the fractal dimension, a conclusion supported by the fractal theory. The standard deviations are, however, quite small, especially for vessel counts higher than fifteen. Estimating their sizes from the graph they would appear to be at most 0.025 at these vessel counts. This is roughly the same as the standard deviation of the errors of the sandbox-method, as implemented here. In effect, these results imply that fractal analysis (with this implementation of the sandbox algorithm) applied to histological slides is nothing more than an inexact and time consuming method for counting vessels.

The concept of such a test is good. Parameters with little or no variations for a given number of vessels will have little information to add to the vessel count. Consequently, not only the fractal characteristics are calculated for randomly generated vessel configurations, but also most of the other investigated parameters. Some parameters such as the relative vessel area are not considered because they are completely determined by the number of vessels in these simulations.

The terms of the random generation should be carefully chosen, in order to be as relevant to real image slides as possible. The time required to analyse large numbers of generated images calls for a reduction in resolution of the generated images. A five fold reduction in resolution has been chosen for these tests. The cases in section 4.2, deemed *high vascular* by a pathologist⁹, had vessel counts ranging from 187 to 321. Counts up to these numbers and with a solid margin for even higher vascular counts should be used.

The size and shape of the dots are also important. Most of these cases had a mean area a little higher than 300 pixels. When scaled down to the new grid size, this is equivalent to roughly 12 pixels. One must also consider whether to use the dots themselves, the perimeter of the dots or only the centre of mass. In the case of centre of mass the smallest possible size is one pixel, corresponding to

⁹This is in part a qualitative assessment, no quantitative measure of what constitutes low and high vascular is defined/agreed upon by the pathological community.

25 pixels in full size images, and the placed dots need not be expanded to full vessel areas. In order to avoid the inclusion of more variables all vessels should be of the same shape and size.

Finally, the distribution of points must be taken into account. The most obvious approach is to let each lattice point hold an equal probability of being a vessel. However, this simple random approach does not do justice to tumour vasculature's ability to form avascular regions. The second approach is to take advantage of tumour vasculature's percolation like scaling. Extrapolating Gazit et.al.'s results[17] from two to three dimensions, a percolation model may be used to generate three-dimensional networks and extract cross-sections from these.

3.6.1 Implementation of the Simple Random Simulation

Single pixels (vessel mass centres) are randomly placed on a 400×300 lattice at intervals of 10 vessels between 10 and 500 vessels. For each vessel count 200 repetitions are performed, and the values plotted at each vessel count represents the mean of these two hundred repetitions, with corresponding standard deviations, indicated by vertical bars. Two vessels may not be placed at the same location, however they are allowed to be direct neighbours. For the purpose of syntactic structure analysis direct neighbours are regarded as separate vessels. An example of an image generated in this way is shown in figure 3.13 along with an image from the percolation simulation. The results of these simulations are shown in section 4.1.1.

3.6.2 Implementation of the Percolation Simulation

Non-trapping invasion bond percolation on a simple cubic lattice has been chosen to generate the three-dimensional systems. An illustration of this type of cluster is shown in figure 3.12. The black lines represent bonds that belong to the cluster, and the coloured lines are unoccupied bonds, where blue represents weak bonds (easily invaded), and red represents strong bonds. In this model the bonds develop in three separate directions, one of these is extracted to form the cross-sections. A lattice of 43×58 in the (x,y) -plane has been found to generate vascular densities of a relevant magnitude. An additional margin of two lattice points is added on all four sides to reduce edge effects. In the z -direction 100 lattice points are used to generate 100 cross sections for each simulation. In total, for all three directions, the lattice spans roughly 870.000 bonds.¹⁰ A single point located in the middle of the x,y -plane at $z = 0$ is used as the inlet point.¹¹

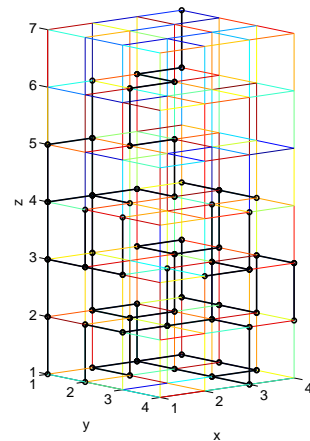


Figure 3.12: Illustration of a 3D bond invasion percolation cluster.

Only the bonds going in the z -direction are used to simulate the cross sections. Symbolic matrices showing which of the bonds that belong to the cluster are extracted at each z -coordinate. The lattice constant, i.e. the distance between each bond, is set to 7 relative to the simple random image slides, resulting in images 301×406 pixels large. To achieve this, empty rows and columns are inserted into the image increasing the minimum distance between two vessels from 1 pixel in the symbolic

¹⁰The exact number is $(46 \times 62 \times 101) + (47 \times 61 \times 101) + (47 \times 62 \times 100) = 869019$.

¹¹The bottom and top end of the cluster (in the z -direction) could potentially be excluded from the simulation results to reduce the edge effects of the inlet source and the stop condition (cluster reaches end of lattice). They are not excluded in this simulation.

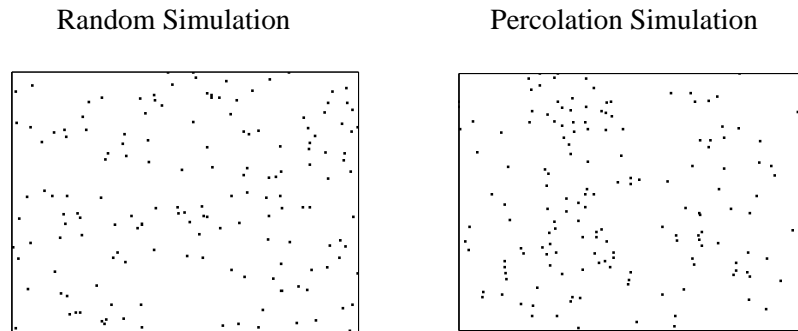


Figure 3.13: Simulated Vessel Slides. Left: Image produced with the simple random simulation. Right: Image produced with the percolation simulation. Both images have a vessel count of 160, on a grid size of 300×400 and 301×406 respectively. The vessels have been enlarged to 3×3 pixels to make them more visible.

matrices, to 8 pixels in the resulting image. At this point only one in forty-nine pixels can possibly contain a vessel, with the empty rows and columns scaling the lattice. This introduces a rigidity to the image, causing the 90° angles between the vessels' positions to be grossly overrepresented. Each vessel is then randomly shifted between zero and three pixels along each axis (x and y), causing it to occupy each of the forty-nine pixels surrounding the bond with equal probability. As in the simple random simulation, this allows pixels from neighbouring bonds to end up side by side, but not on top of each other.

The minimum spacing between vessels with this method is larger than that of the simple random simulation. In neighbourhoods smaller than 7×7 -pixels no more than four pixels can possibly contain a vessel, and in most such neighbourhoods the maximum number is one.¹² Enlarging these images up by a factor of five, to bring them up to par with the histological sections, the mean distance between two vessels in a fully occupied cross-section from this simulation is 35 pixels. As a comparison the mean branch lengths of the Euclidean minimum spanning trees of the four sections ranges from 57 to 73 pixels, subtracting one standard deviation this turns into the range 14 to 24. At size-scales larger than the lattice factor the percolation method should, however, contribute to increasing the odds that vessels are located close to each other. On these scales cross-sections can be expected to contain both vascular and avascular areas in accordance with the observed data on tumour vascular morphology.

Another important difference between the two simulation methods is that the percolation method does not allow the number of vessels in a section to be specified. Instead a large number of clusters is generated, and the number of vessels in each section is counted. This has two implications. The first being that the vessel count can take any integer value, not just steps of ten. The second that, unless a subset of a very large quantity of data is used, one cannot expect to have an equal amount of data for each vessel count. Consequently, this data may be better suited for scatter plots than the errorbar representation used in the simple random simulation, however this makes the results harder to compare. Averaging of the data into a curve with data points for every 10 vessels, i.e. a histogram drawn as an errorbar plot, makes this easier. The results from these simulations are shown in section 4.1.2

¹²One way to change this would be to use Gaussian probability distributions with a suitable standard deviation, for instance somewhere between one half and one lattice constant. This distribution would replace the uniform displacement probability when modifying the vessel positions. A further step in this direction could be to let the percolation network determine a probability distribution function and then place a specific number of vessels randomly throughout the lattice according to this distribution.

Chapter 4

Results

4.1 Vessel Section Simulations

Two sets of random simulations were performed to generate a large quantity of possible vessel configurations. A variety of parameters that can be calculated from immunohistological slides are investigated in this thesis. These simulations were performed to find out how closely the various parameters are related to the number of vessels in the slide. Although the shape of the curve is of interest, the main point is the behaviour of the standard deviations. The size of the standard deviation is used to evaluate how dependent the parameter is on the positions of the vessels. Furthermore, the size of the standard deviation is considered relative to the size of the mean. Many of the parameters have curves with local variations, small peaks and valleys. This is, most likely, caused by the fact that only two hundred samplings are performed for each point. The overall shape of the curve and the standard deviations should, however, be sufficiently accurate. The simple random simulations and the percolation simulations both generate randomly distributed vessel patterns, but the patterns are expected to have somewhat different characteristics. The resulting figures from these simulations are shown in the following subsections, along with brief descriptions of any relevant details. The coloured circles (usually red) and \times es (black and blue) in figures 4.2, 4.3 and 4.6–4.13 are the data from the four histological samples, see section 4.2.4 for further description and comment on these markers.

4.1.1 Simple Random Simulations

Cumulative Histogram Characteristics

The results from the cumulative histograms of the distances to the nearest vessel are shown in figure 4.2. Both the values and the standard deviations drop fairly quickly to a low stable value. The 10% curve has the largest variations and they are quite substantial below approximately 50 vessels. All the deviations drop to below 2 pixels ($1.38\mu\text{m}$) within the 100 vessels mark.

Fractal Characteristics

The variations of the fractal dimensions are of particular interest because the various methods calculating them are approximations. The sandbox algorithm, for instance, found to be the most stable in section 3.5.2, has a standard deviation of errors made equal to 0.0230 (see table 3.2). If the variations in fractal dimensions are not sufficiently high, then it would be impossible to deduce whether it was

the fractal approximation or the differences in the distributions that caused the variations of the result. The sandbox algorithm is the only fractal estimation tested in these simulations. In cases where multiple linear regions are found, the region spanning the highest sandbox diameters is used.

At the top of figure 4.6 the results of the fractal analysis of the simulated image slides are shown. The fractal dimension increases rapidly at the start and hits a shoulder at roughly 70 vessels. It appears to be slowly increasing even at the highest vessel counts, albeit not much. The standard deviations steadily decrease throughout the range. When approaching 500 vessels, the deviations are still a small margin above the errorlevel of the test (see table 3.2).

The position and size of the fitted region varies throughout the range. This applies especially to the region's largest diameter, i.e. its end coordinate. The mean increases rapidly for small values, and standard deviations are quite large even at high vessel counts. All in all, however, the combined region is quite significant above fifty vessels, spanning at least a third of the possible linear values, and two thirds at the long end.

The fractal analysis of Gabriel's Graph displays the same basic pattern, but the values are somewhat different. The dimension is higher throughout the range, especially at the start. The standard deviations are much smaller and appear to flatten out at a value well below the algorithm's 0.0230 test error deviation, with an average value of 0.0164 for the range 400–500 vessels region. This is the graph that uses the widest regions in its fits.

The Euclidean Minimum Spanning Tree has the slowest climb up to the shoulder part. The standard deviations are large all the way up to 300 vessels. At the lowest vessel counts, the image slide analysis has by far the greatest standard deviations. In the range starting around 100 vessels and up to around 300, vessels the EMST has the largest deviations.

For all three cases the deviations in the largest sandbox diameter used, appears to relate to the deviations of the fractal dimension.

Syntactic Structure Analysis

The syntactic structure analysis includes histogram parameters from the Voronoi diagram, Gabriel's Graph and Euclidean Minimum Spanning Tree, in total a set of 44 parameters. For all three sets the skewness and kurtosis parameters show large standard deviations. Furthermore, all parameters, with exception of the branches per node for the EMST, have significant variations below 50 vessels. The following descriptions of the individual parameters will only apply to the mean and the standard deviations, especially for vessel counts larger than 50. The fractal analysis of Gabriel's Graph and the EMST is categorized into the Fractal Characteristics section.

When the simulations were performed, a small error in the code generating the EMST caused an error in some graphs. The bug was in the initialization of the network and caused one of the potential branches, connected to the second node added, to be excluded from the potential branch list under certain circumstances. This caused some of the EMST's to be generated with up to two suboptimal branches. This is not expected to have compromised the results, and the size of the effects are readily observable by comparing the distance to the nearest neighbour parameters from GG and EMST, which should have been identical. Subtle differences in the error bars are observable below 30 vessels.

Voronoi Diagram: The Voronoi diagram parameters are shown in figure 4.8. As usual the polygons with sides exceeding the bounds of the image matrix are ignored to avoid edge effects. While the mean standard deviation of the area have quite small deviations for all but the smallest vessel counts, the shape and form of the polygons have notable deviations throughout the range.

Gabriel’s Graph: The branches per node measure is the only parameter with large variations throughout the range, see figure 4.10. Comparing the distances to the nearest and the furthest neighbours in the graph, the latter appears to have somewhat larger variations in the standard deviations. The variations in the means are similar, but the furthest neighbour has a much steeper curve (note the different y-axis).

Euclidean Minimum Spanning Tree: These results resemble those of Gabriel’s Graph. There are some differences though, see figure 4.12. The mean number of branches per node are completely determined by the number of vessels and takes the value $(n - 2)/n$, this is the only parameter without deviations. The standard deviations around the mean, within a given sample, does, however, vary, and this parameter has by far the highest variations, disregarding the skewness and kurtosis. The transformation from Gabriel’s Graph to EMST trims away the longest branches, keeping all the shortest branches between nodes. For this reason, the variations in the furthest neighbour distances are reduced, and the data on the nearest neighbour is essentially identical¹ to that of Gabriel’s Graph.

4.1.2 Percolation Simulation

As shown in section 3.6.2, the images produced by the percolation algorithm can take any integer number of vessels between 1 and 2580. The number of vessels produced is shown in figure 4.1, truncated at 100 images and 500 vessels. Images with *less* than 10 vessels (or more than 500) were not used, nor were data points with fewer than 10 images. In total, 15.560 images were used in the analysis, originating from 168 percolation clusters. By comparison, 10.000 data images were produced in the random simulation. The percolation plots make for a somewhat different visual appearance, due to the tenfold increase in data point density. The number of images used at each data point is, however, greatly reduced and the plots should be read against the background of the distribution in figure 4.1

Cumulative Histogram

The cumulative histograms show large variations, especially for images with few vessels, see figure 4.3 At vessel numbers larger than 200, the deviations are small or negligible. Both the distances and the deviations are much greater than in the random simulation. At the three values, 50, 78 and 97, both the mean value and the standard deviations drop suddenly. Furthermore, in all three cases the standard deviations rapidly increase again to a value about one third higher, relative to the initial drop. After the two first and largest peaks, the deviation drops, only to slowly increase again until it reaches the next drop. This behaviour is investigated further in figure 4.4. Here the mean and standard deviations are plotted together with the number of images at each vessel number. The bottom graph has been smoothed to emphasize the trends in the curves. At the third drop, there is a significant increase in the image material. At the second drop, there is a local increase as well, though not very large. At the position of the first and largest drop, however, there are no such changes in the material. There is a small increase nearby, it is, however, located after the fall and quickly drops back down again. The shapes of the three parameters’ curves are strongly correlated, showing many of the same local trends, with respect to slope and local extrema.

¹Close examinations reveal small differences, especially in the standard deviations at very low vessel counts. This is due to an error in the code at the time of the simulation. The error has been identified

Fractal Analysis

The fractal dimensions of the images, GG and EMST all show patterns similar to those of the random simulation. However, the standard deviations are somewhat larger, and the mean values are lower. The pattern of the linear regions used is very similar to those in the random simulation. The fractal dimension of Gabriel's Graph is larger than those of the image and the Euclidean Minimum Spanning Tree, which are fairly similar below 200 vessels, although the EMST approaches GG at high vessel counts, and is larger at the very lowest. This is contrary to the random simulation where EMST has a dimension lower than that of the images below 150 vessels, see figure 4.5. At 50 vessels, all three fractal dimensions increase suddenly. This matches the first sudden drop in the cumulative histogram. The curves appear to have a reduced increase in dimensions leading up to the sudden jump. Then, in one big leap between 0.1 and 0.2 large, it changes into a higher dimensioned path that appears to extrapolate well down to the start of the curve. As the dimension increases the variations are roughly halved, aligning the top of the standard deviations, but causing a leap, at least as big as that of the mean, at the lower end. The other two leaps in the cumulative histogram, at 79 and 98 vessels, leave no similar changes in the dimension or in the variance. If at all affected, the change is hidden by the "noise" produced by the low sampling rate.

Syntactic Structure Analysis

No plainly observable changes occur at any of the three leap points of the cumulative histogram for any of the SSA parameters. As with the random simulation, all the skewness and kurtosis parameters have large variations. The variations in these parameters are, yet again, greater than those of the random simulation images. When comparing with the random simulation, keep in mind that the y-axis is in general different. Most of the means are similarly shaped as their simple random counterparts, however, the values may be different. The variations of mean and standard deviations of many of the SSA parameters are substantial beneath 100 vessels.

Voronoi Diagram: The mean and the standard deviations of the area are smaller in this simulation than in the random. The difference is substantial, a factor 2 for the mean and 1.5 for the standard deviations. The variations are, however, much greater here. The shape and form parameters are very similar, the noticeable difference being that the mean of the form has greater dependence on the number of vessels, steadily increasing throughout the range.

Gabriel's Graph: The mean branch length is shorter in the percolation images, but the variations are greater. This is in accordance with the area distribution of the Voronoi polygons. The relatively few longer branches created by an increase in the number of short branches are not able to hold the mean length up; this causes lower means and higher deviations. The same is true for the nearest and furthest neighbour parameters. Furthermore, for all three the difference between these and those of the random simulation again appears to be roughly a factor of 2 for the mean and 1.5 for the deviations. The branches per node distributions are very similar between the two. The variations of the skewness and kurtosis of the Gabriel's Graph parameters are larger for the percolation simulation than the random simulation, with the exception of those of the branches per node which are very similar.

EMST: The EMST parameters mostly follow the same pattern as those of Gabriel's Graph, including the factor of 2 and 1.5. The mean of the branches per node is an exception.

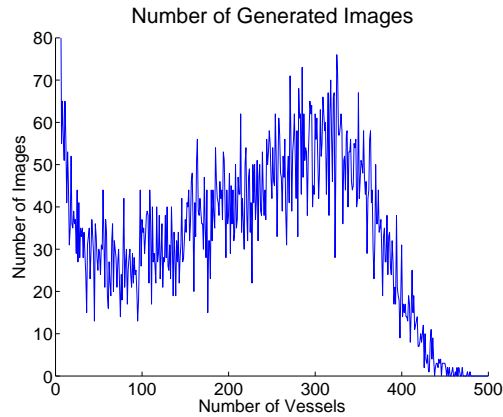


Figure 4.1: Percolation Simulation — The Number of Slides at Each Vessel Number: The percolation method does not allow the number of vessels generated in an image slide to be controlled. The number of images at each vessel count were found to have this distribution.

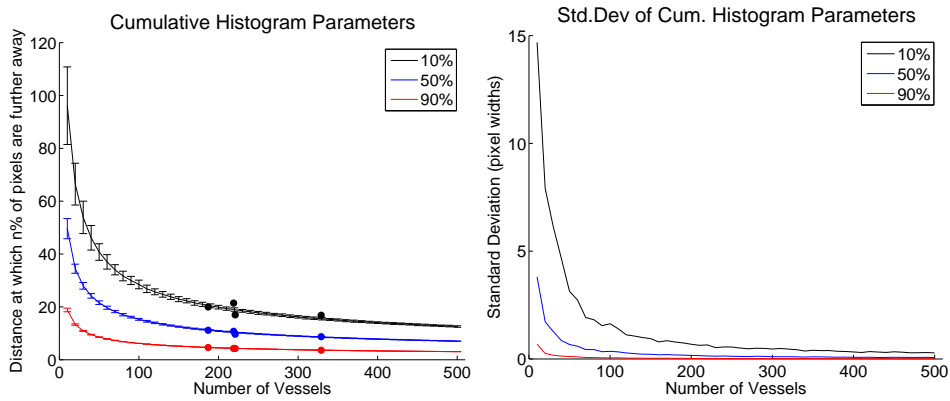


Figure 4.2: Random Simulation — Cumulative Histogram Parameters: Left, the mean of the 10%, 50% and 90% cumulative histogram values, with corresponding standard deviations. Right, the size of the standard deviations.

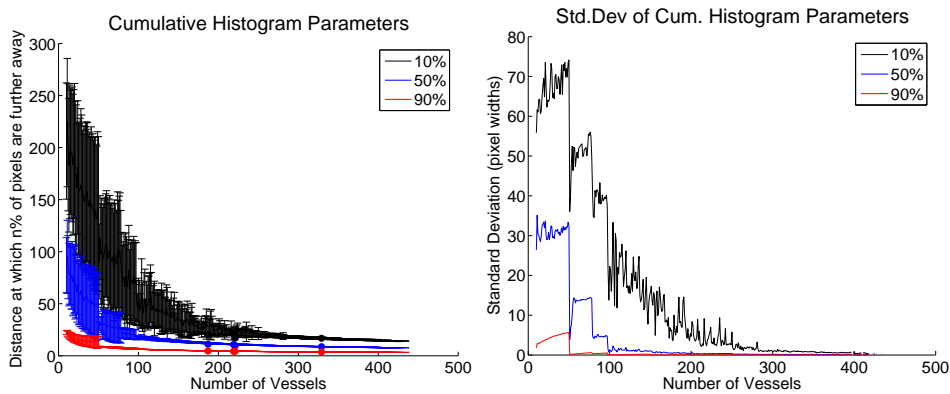


Figure 4.3: Percolation Simulation — Cumulative Histogram Parameters: Left, the mean of the 10%, 50% and 90% cumulative histogram values, with corresponding standard deviations. Right, the size of the standard deviations.

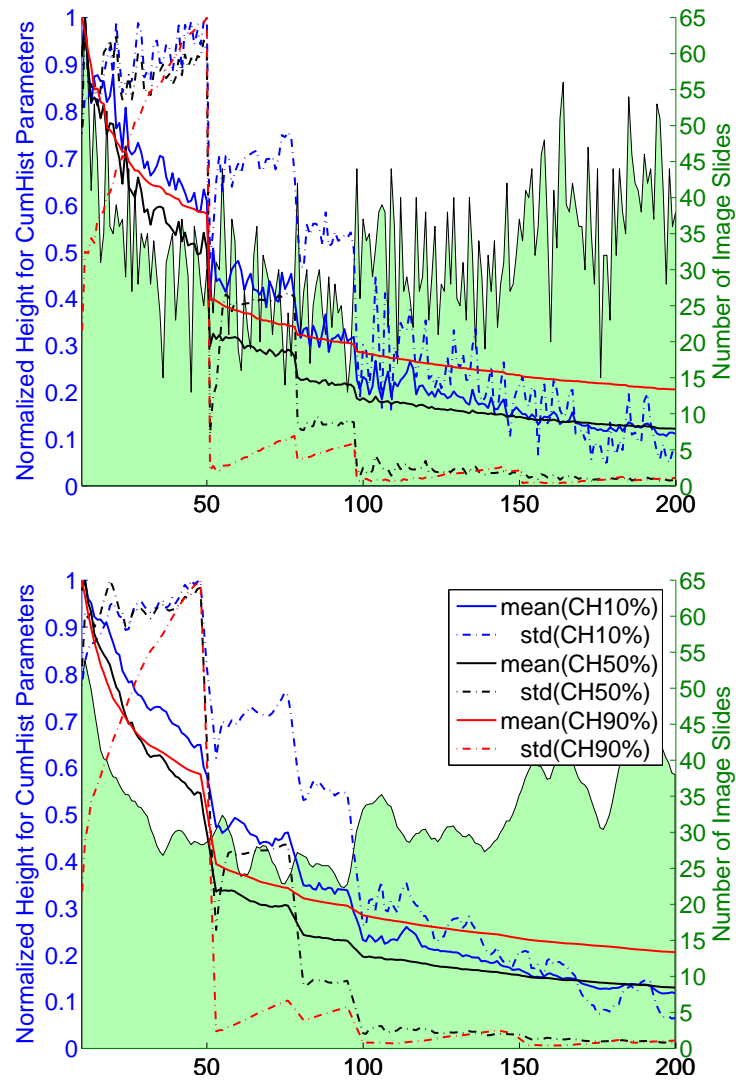


Figure 4.4: Percolation Simulation — Cumulative Histogram Parameters and the Number of Images: The two graphs show the same curves, but the plots in the bottom graph have been smoothed with a five point moving average, the image number has been smoothed twice. The normalized mean and standard deviations of the three cumulative histogram parameters are plotted with corresponding normalized values at the left hand axis. In the background the number of images containing that particular number of vessels are plotted in light green, the values are shown in the axis to the right.

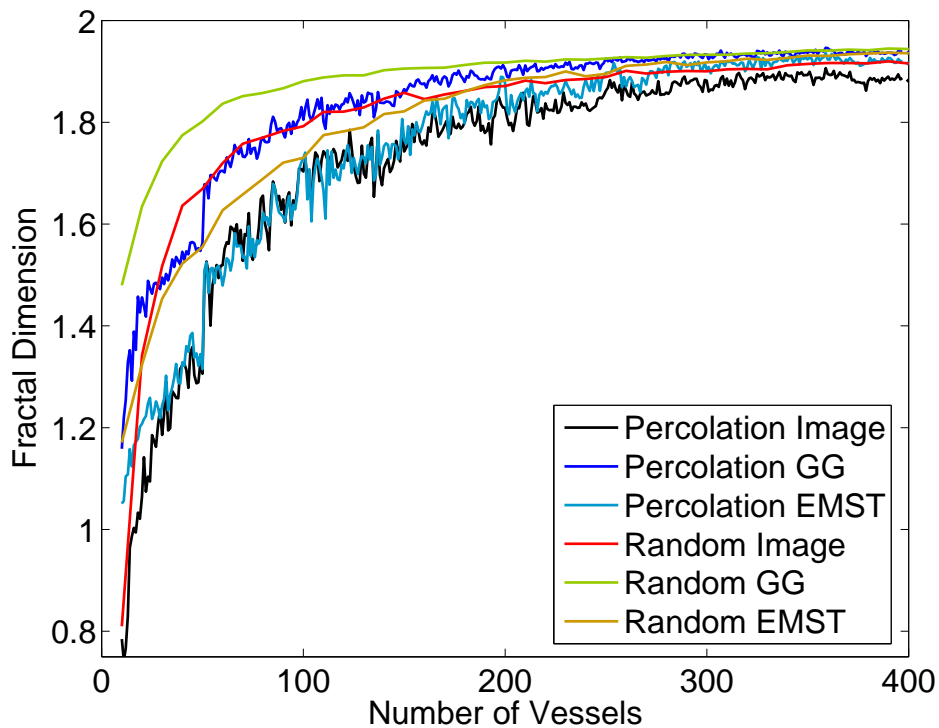


Figure 4.5: The Mean Fractal Dimension of Each Image/Graph and Both Methods: The dimension of the image is much lower for the percolation images, compared to the simple random images. Furthermore, the Euclidean Minimum Spanning Tree does not appear to have a reduced dimension compared with the image, this is, however, the case in the random simulation. The curves end near the dimensions 1.89, 1.93, 1.94, 1.93, 1.95 and 1.95 in the order of the legend from top to bottom. The graph is truncated at 400 vessels, after which the percolation data becomes sparse.

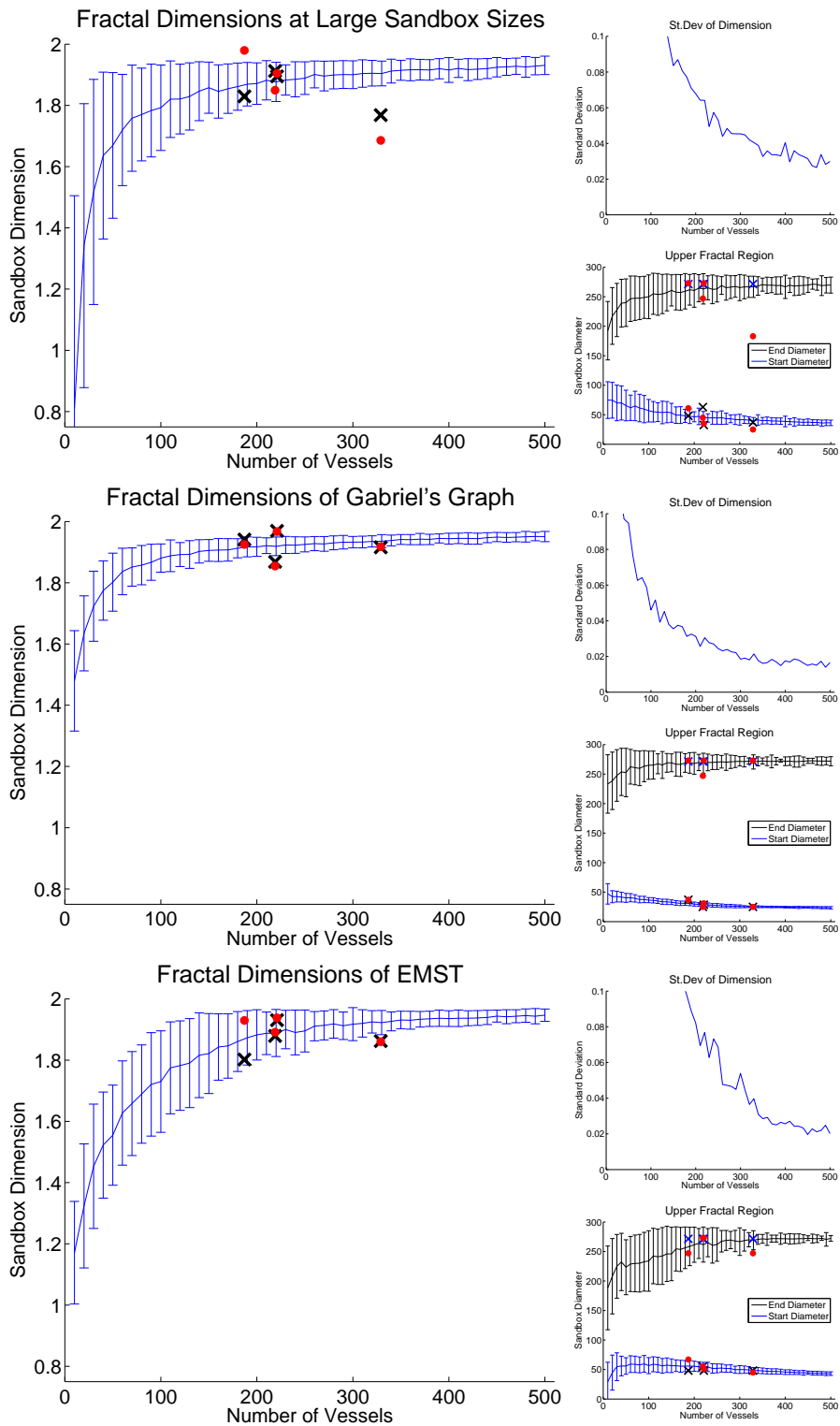


Figure 4.6: Random Simulation — The Fractal Parameters' dependence on the number of vessels in the slide. From top to bottom: the fractal characteristics of the vessel locations, Gabriel's Graph and the Euclidean Minimum Spanning Tree. For each of the three cases the fractal dimensions along with the corresponding standard deviations are shown in the left graph, note that the y-axis is truncated at 0.75. The standard deviations are plotted by themselves in the top right graph, truncated at 0.1 to emphasize the smallest values. In the lower right hand graph the smallest and largest sandbox diameter used in the linear fit is shown (linear scale).

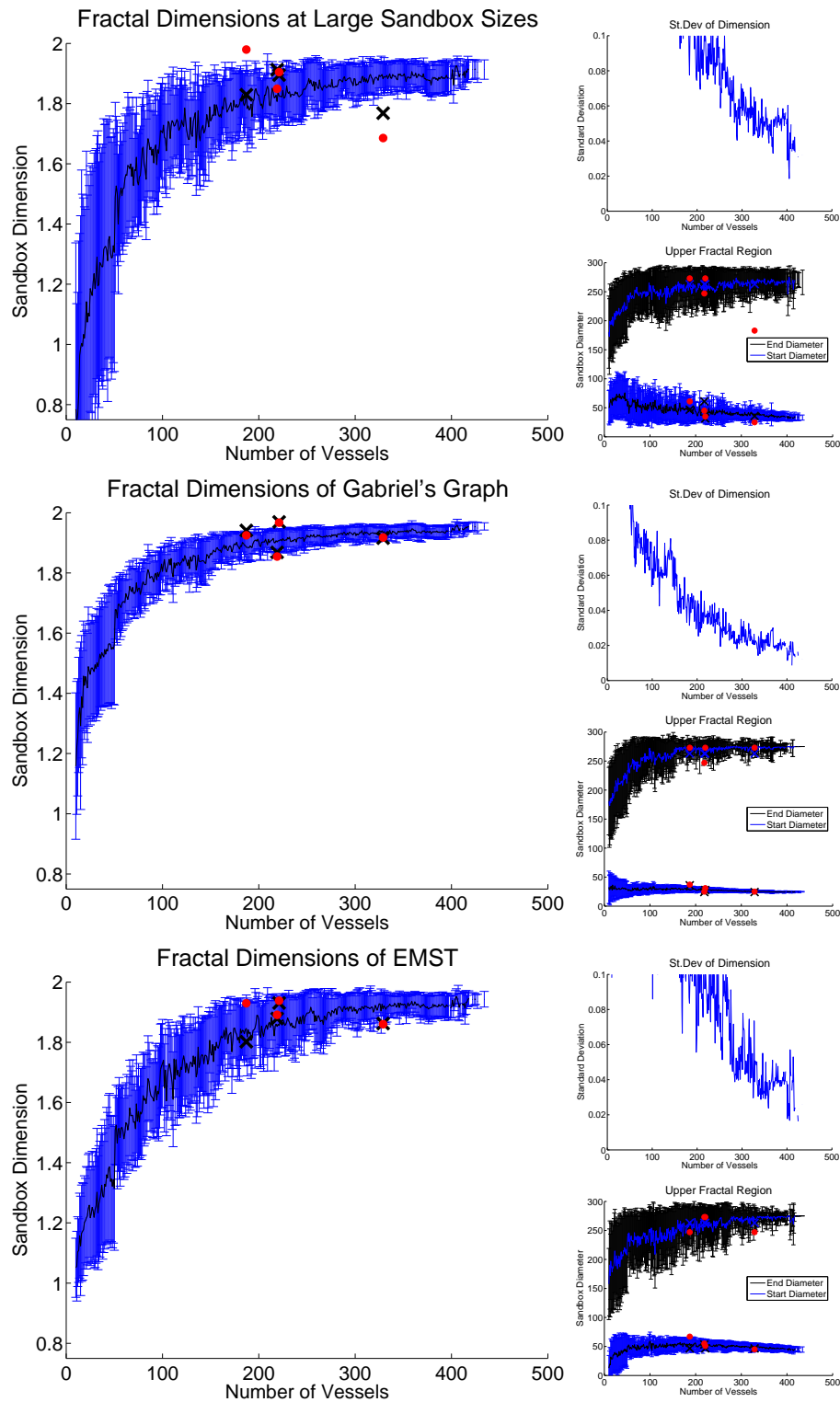


Figure 4.7: Percolation Simulation — The Fractal Parameters' dependence on the number of vessels in the slide. From top to bottom: the fractal characteristics of the vessel locations, Gabriel's Graph and the Euclidean Minimum Spanning Tree. For each of the three cases the fractal dimensions along with the corresponding standard deviations are shown in the left graph, note that the y-axis is truncated at 0.75. The standard deviations are plotted by themselves in the top right graph, truncated at 0.1 to emphasize the smallest values. In the lower right hand graph the smallest and largest sandbox diameter used in the linear fit is shown (linear scale).

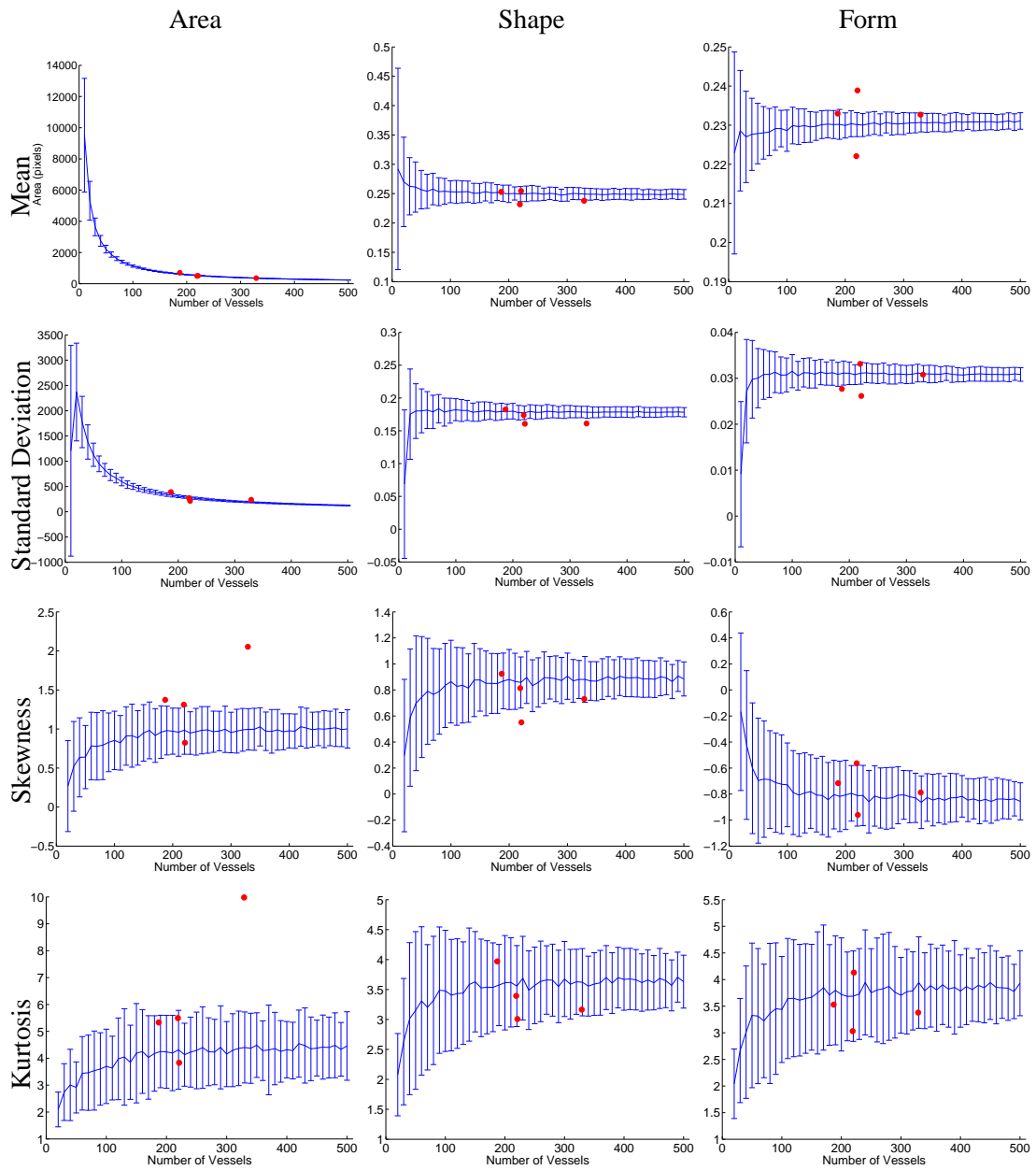


Figure 4.8: Random Simulation – Voronoi Diagram: Although variations of the mean and standard deviation of the area quickly diminish for higher vessel counts, the shape and form of the polygons show variations even for the very high vascular slides. Skewness and kurtosis exhibit large variations for all vessel counts.

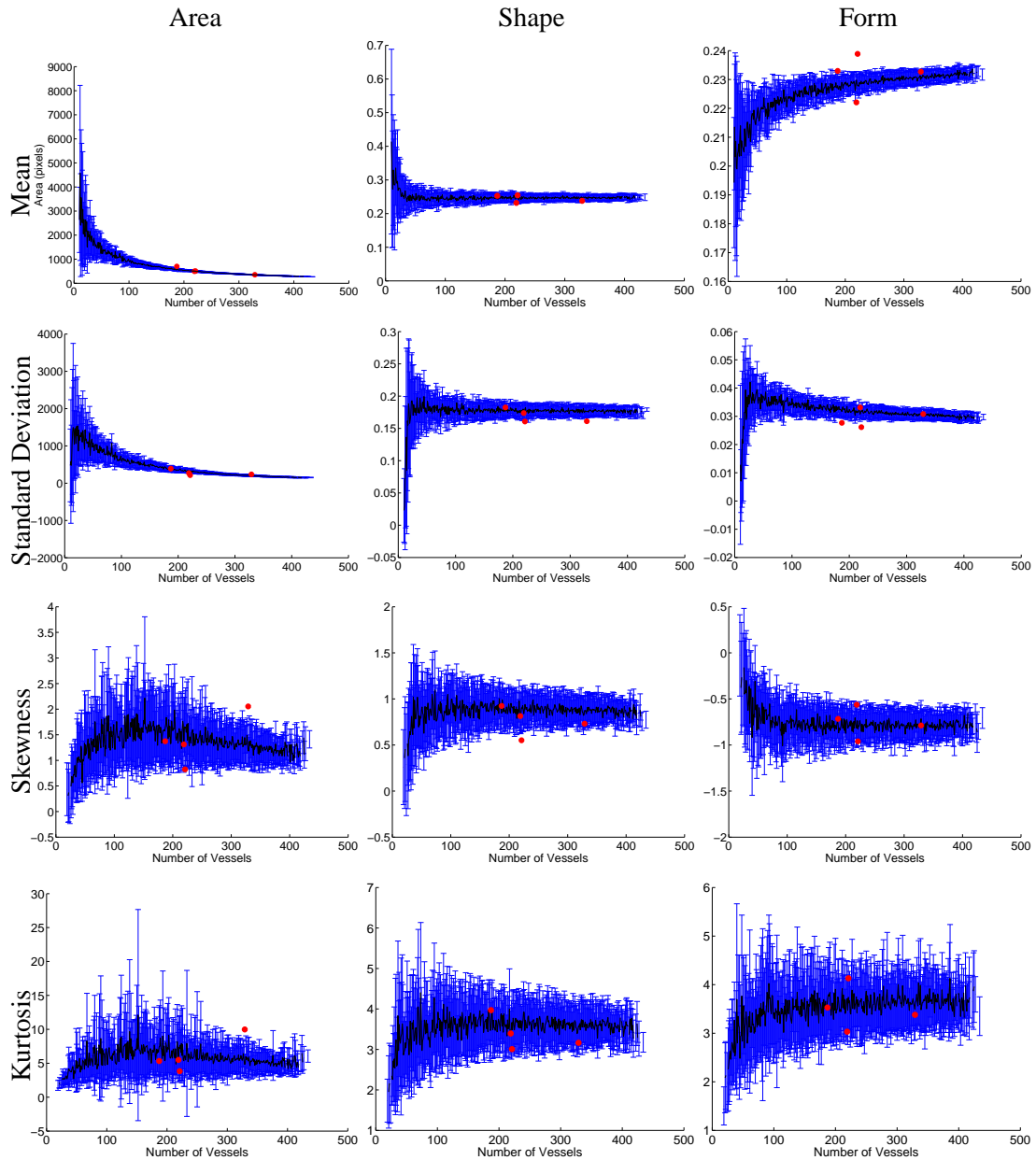


Figure 4.9: Percolation Simulation – Voronoi Diagram: Although variations of the mean and standard deviation of the area quickly diminish for higher vessel counts, the shape and form of the polygons show variations even for the very high vascular slides. Skewness and kurtosis exhibit large variations for all vessel counts.

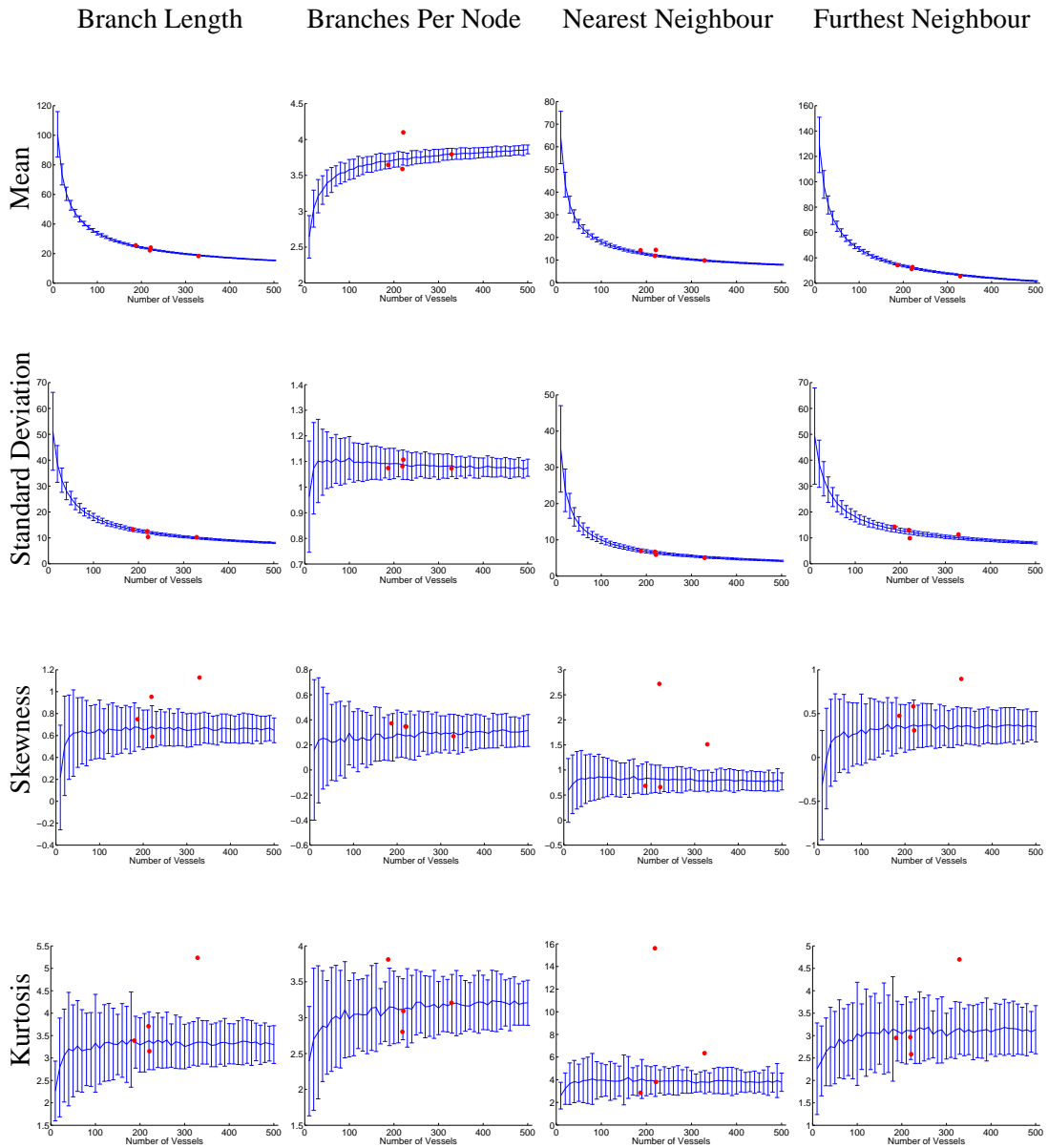


Figure 4.10: Random Vessel Simulation — Gabriel's Graph: The deviations of the branch lengths and distance to nearest and furthest neighbour all decrease rapidly with increasing vessel count. The Branches per Node count on the other hand shows much larger differences. The skewness and kurtosis have large deviations for all investigated parameters. When comparing the distances to the nearest and furthest neighbour, note that the y-axis limits are different.

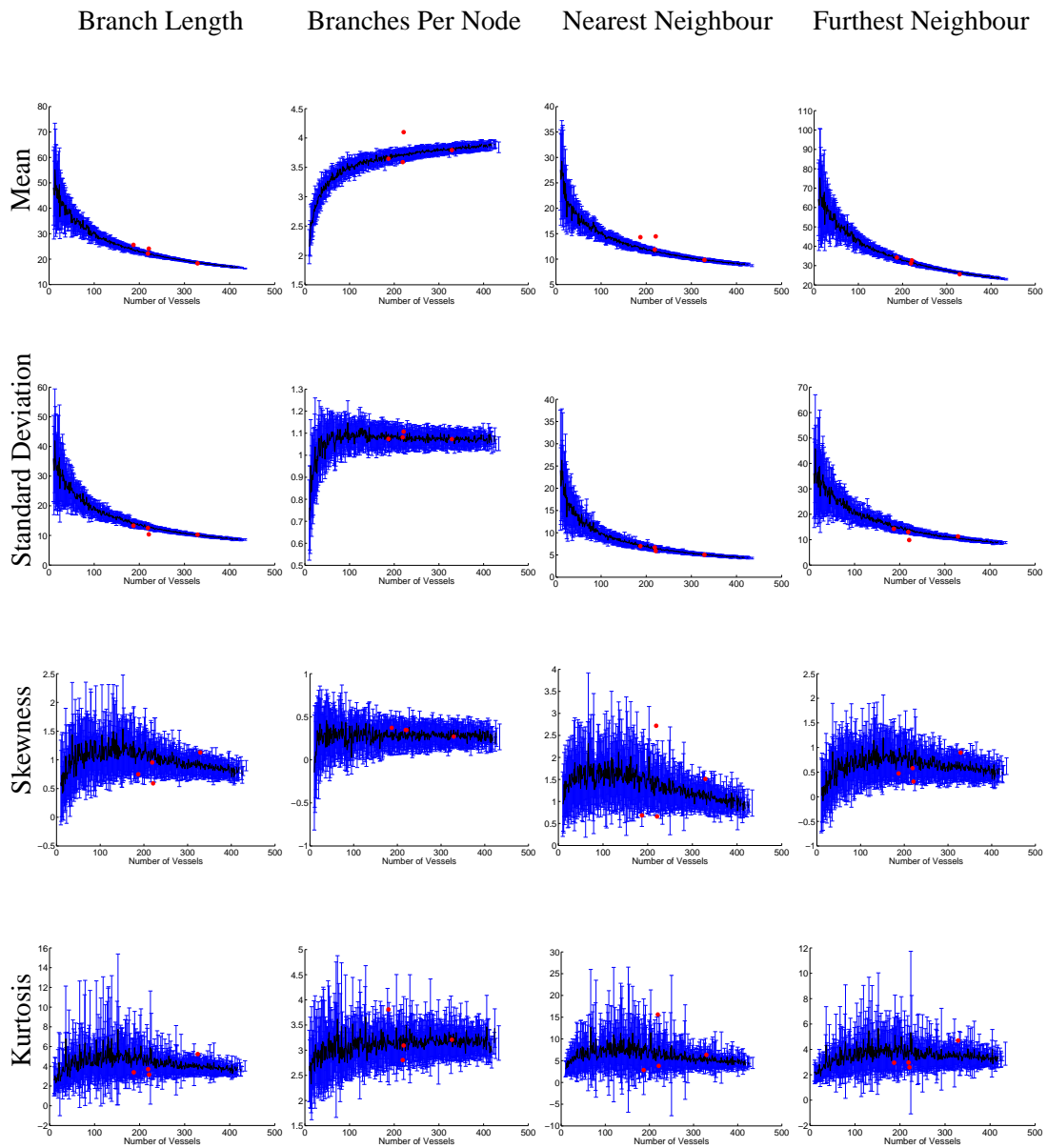


Figure 4.11: Percolation Simulation — Gabriel's Graph: The deviations of the branch lengths and distance to nearest and furthest neighbour all decrease rapidly with increasing vessel count. The Branches per Node count on the other hand shows much larger differences. The skewness and kurtosis have large deviations for all investigated parameters. When comparing the distances to the nearest and furthest neighbour, note that the y-axis limits are different.

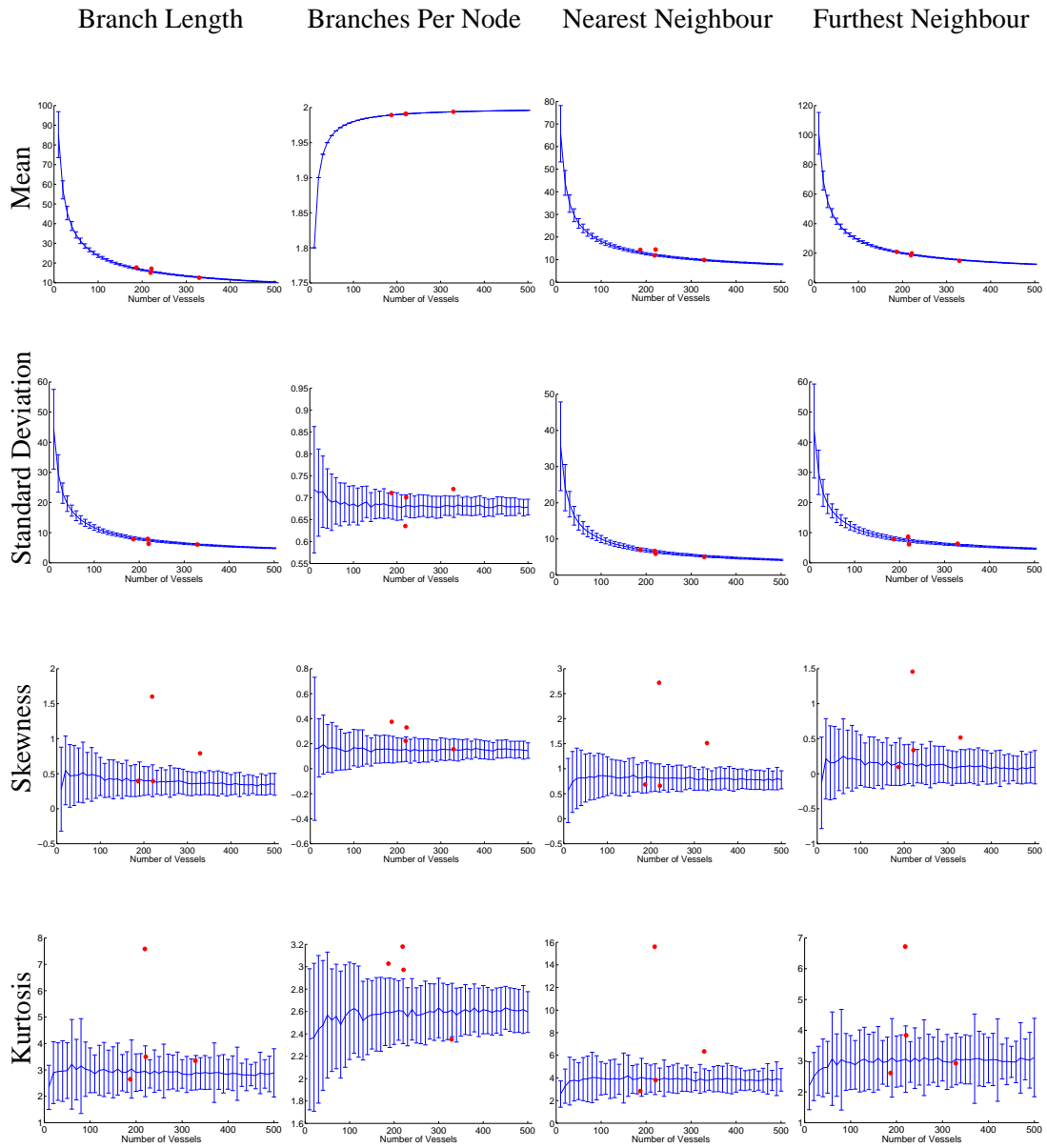


Figure 4.12: Random Vessel Simulation — Euclidean Minimum Spanning Tree: These results are quite similar to those of Gabriel’s Graph, with one exception, the mean number of branches per node is completely determined by the vessel count. The standard deviation of branches per node, on the other hand, still have variations on an order similar to that of Gabriel’s Graph. The Kurtosis and Skewness of all four parameters have large deviations.

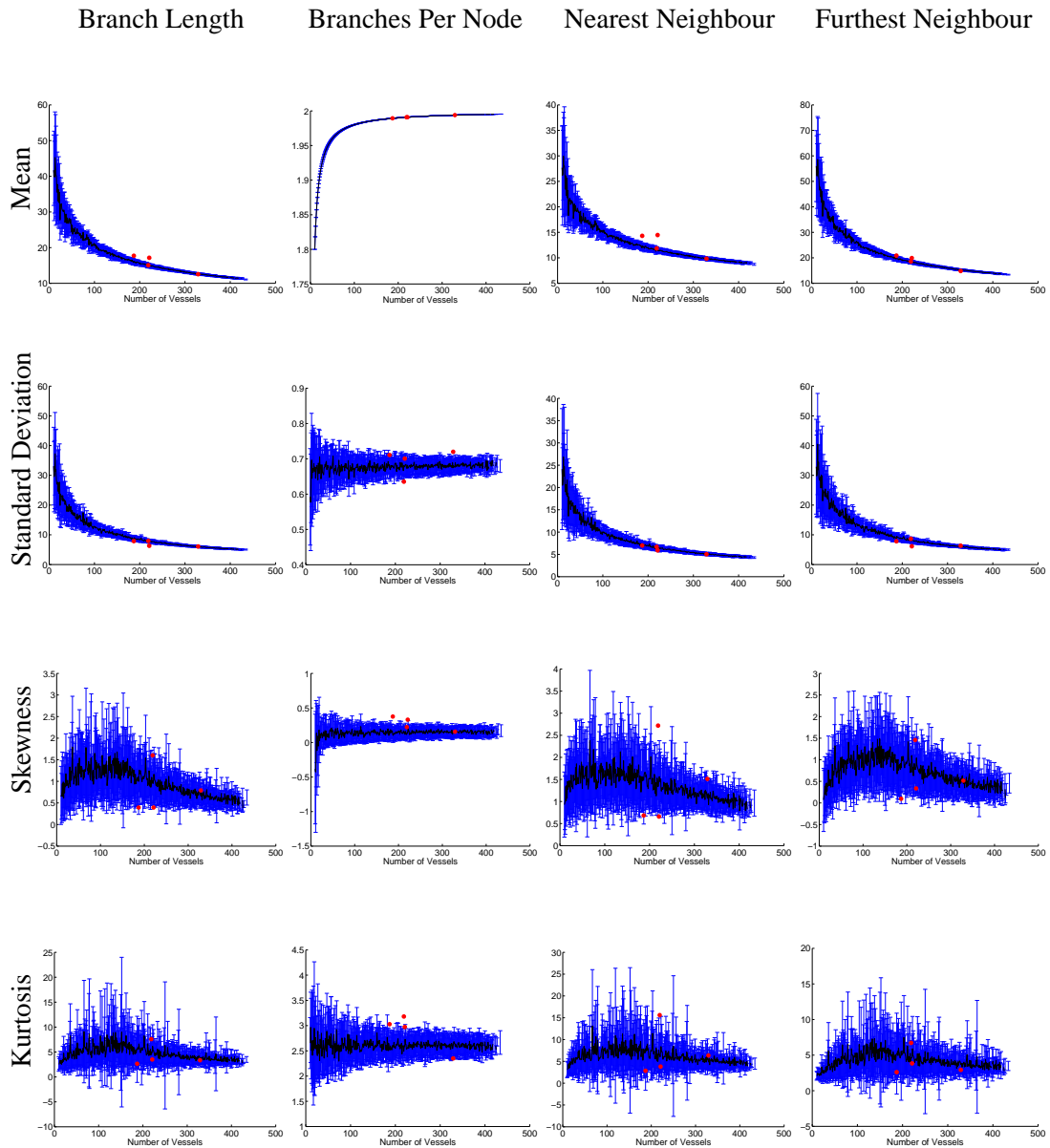


Figure 4.13: Percolation Simulation — Euclidean Minimum Spanning Tree: These results are quite similar to those of Gabriel's Graph, with one exception, the mean number of branches per node is completely determined by the vessel count. The standard deviation of branches per node, on the other hand, still have variations on an order similar to that of Gabriel's Graph. The Kurtosis and Skewness of all four parameters have large deviations.

	case1x25	case2x25	case3x25	case4x25
Number of Vessels	329	187	221	219
Area of Vessels to scale (μm^2)	312.6 ± 377.4 5.95 ± 7.19	314.0 ± 625.3 5.98 ± 11.91	185.6 ± 173.9 3.53 ± 3.31	308.8 ± 379.3 5.88 ± 7.22
Relative Vascular Density (area)	3.18%	1.81%	1.27%	2.09%
Cumulative Histogram at 90% to scale (μm)	9.0 1.24	13.3 1.83	13.0 1.79	11.3 1.56
Cumulative Histogram at 50% to scale (μm)	33.2 4.58	45.3 6.25	40.0 5.52	43.4 5.98
Cumulative Histogram at 10% to scale (μm)	75.5 10.42	89.6 12.36	76.5 10.56	97.1 13.40

Table 4.1: Basic image statistical parameters for each of the four cases. The rows labeled *to scale* show the corresponding values in μm , rather than pixels. The *x25* in the case name refer to magnification the images were acquired at.

4.2 Analysis of Histological Sections

Sections of four invasive carcinomas of the breast stained for CD34 were analysed at 25x magnification by fractal analysis and syntactic structure analysis, as well as a few more parameters relating to the number of vessels, the vascular areas and the distances to the nearest vessel throughout the image. The images were pre-processed in accordance with section 3.1. In one case, number 4, the section had an artifact, a small black area. Even though there were found, upon close examination, signs of staining at the edge of the black region, the entire area has been considered non-vascular and manually removed in an image editor. The removal was done by replacing the area with textures from the surrounding region, this to prevent it from leaving a noticeable edge which would be picked up by the gradient image.

Both the sections themselves and the imaged regions were chosen by a pathologist. They are all high-vascular and exhibit somewhat different distribution patterns. The cases before and after thresholding are shown in figure 4.14.

4.2.1 Image Statistics

Using the methods described in section 3.2, the number of vessels, the mean area of the vessels, the relative vascular density, and the three cumulative histogram values were calculated for each case. These data are shown in table 4.1. Case three is by far the one containing the highest number of vessels, while case two has the lowest number.

Cases 3 and 4 have almost the same number of vessels, but both the vessel sizes and the distribution patterns are very different between the two, see figure 4.14. The differences in the areas are clearly reflected in both the area of the vessels and the relative vascular areas listed in the table. Cases 1, 2 and 4 all have a mean vessel area close to 310 pixels, case 2, however, has a much higher standard deviation.

The influence of the mean vessel area on the cumulative histogram parameters scales as the square root of the area, and is consequently quite small. For instance, the differences in the area distributions between cases 3 and 4 should decrease the distances to the nearest vessel in case 4 by approximately 2.2 pixels compared to case 3. The different distribution patterns, however, result in an increase of 20.6

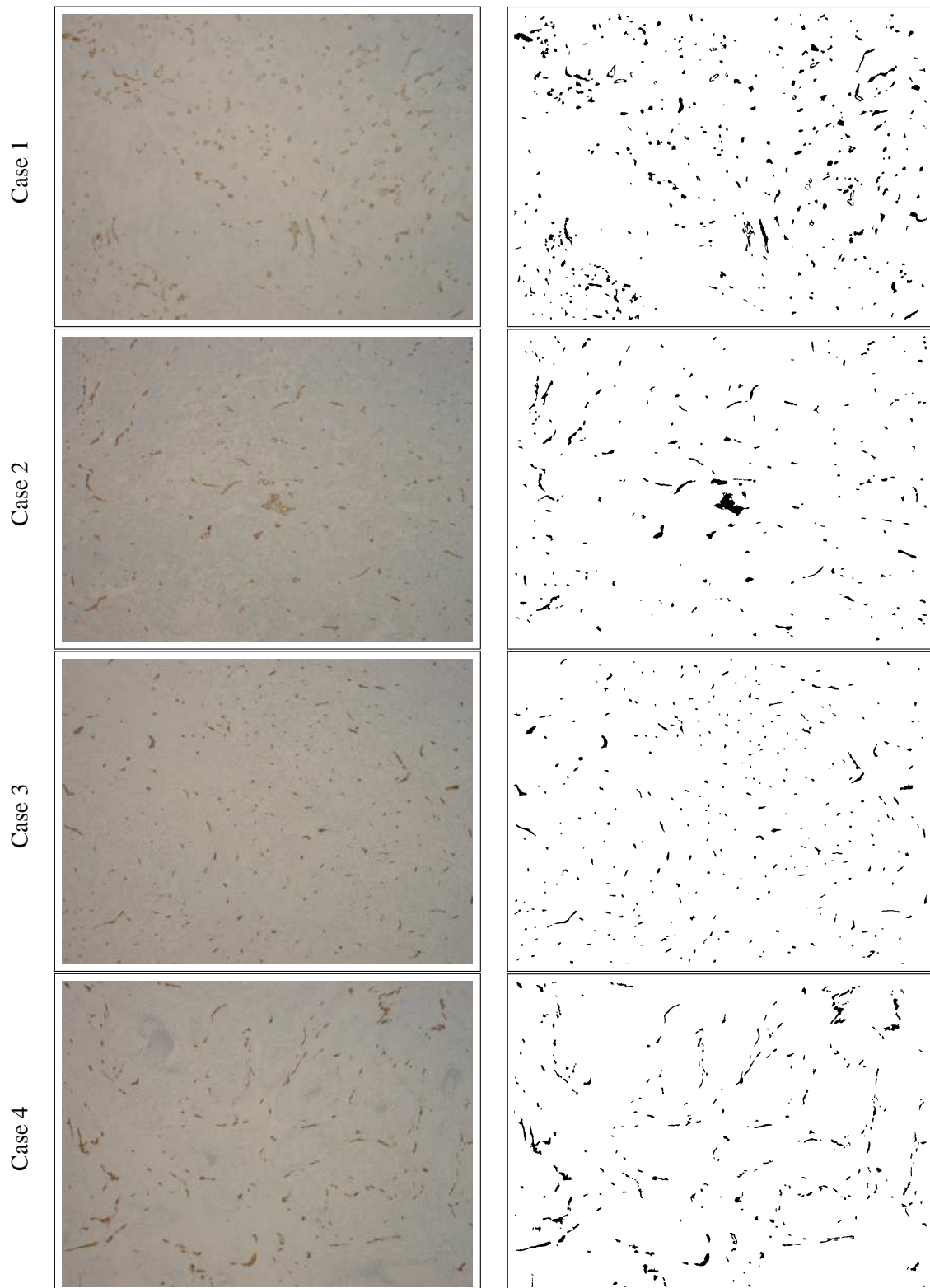


Figure 4.14: Images of the four cases. The unprocessed images are shown in the left column and the black and white results of the threshold procedure in the right.

pixels or 26.9%. Compared to the $70\mu\text{m}$ long diffusion range of oxygen, the cumulative histogram values are relatively small for all cases.

4.2.2 Syntactic Structure Analysis

The results of the syntactic structure analysis of the four cases are shown in tables 4.3–4.5. The standard deviations of the various parameters relative to the mean values are listed in table 4.2.

Voronoi Diagram

The mean area is closely and inversely related to the number of vessels in the diagram. The standard deviation of the area reflects, however, the spread of the distribution, and varies greatly between the four cases. The skewness and kurtosis show large differences for the area histograms, and somewhat less for the form and shape of the polygons. There is also less spread in the mean and standard deviations of these parameters. The size of the standard deviation of the shape, roughly 70% of the mean, suggests that this is not a highly sensitive parameter where changes in the second and third decimal are important. The form has, however, a much lower standard deviation, in the order of 10% and relatively small differences may be of some value. The number of samples is, however, far too limited to make any conclusions.

Gabriel's Graph

The mean branch length decreases for larger vessel numbers, as expected. The skewness and kurtosis of the mean and the distances to nearest and furthest neighbour, show large variations. The number of branches per node and their standard deviations are fairly large compared to the simulation results, case three in particular appears to be at least two standard deviations greater than the mean of the simulation. The standard deviations are consistent with the simulation results. The distance to the furthest neighbour appears to be dependent on the number of vessels, much like the mean branch length. The standard deviations, however, varies greatly and does not show the same correlation. The values range from 47–78 pixels and 29–47% of the mean. These values are those of cases 3 and 4 respectively, two cases with almost the same number of vessels (221 and 219). While the mean distance to the furthest neighbour is similar for cases 3 and 4, the distance to the nearest neighbour is not.

Euclidean Minimum Spanning Tree

The mean branch lengths of the EMST are 67.5, 67.7, 73.1 and 67.9% of the mean branch length in GG. The third case also have a much smaller (relative) standard deviation compared to the other cases. Case four has the highest deviations. The two cases with the least difference in vessel numbers end up at each end of the spectre again. The branches per node parameters are fairly similar in all cases. The differences in the skewness is less pronounced than in Gabriel's Graph. The differences in the kurtosis is, however, slightly larger. The relative deviations of this parameter are fairly similar from case to case, with case four having a slightly smaller deviation than the other three. The distance to nearest neighbour parameters are identical to those in Gabriel's Graph. The distances to the furthest neighbour are 55.1, 36.8, 62.6 and 57.7% of the corresponding GG parameter respectively. The relative deviations in this parameter are reduced compared to those of GG for cases 1 and 2, but increased for case 3 and fairly similar for case 4. Case 3 has the smallest relative deviations in the distances to both the nearest and furthest neighbour for both GG and EMST.

		Case 1	Case 2	Case 3	Case 4
Voronoi	Area of polygons	64.0%	56.2%	41.1%	53.4%
	Shape of polygons	70.4%	69.5%	69.4%	67.9%
	Form of polygons	12.4%	10.5%	10.0%	14.2%
GG	Branch Length	58.1%	52.1%	41.3 %	59.5%
	Branches per Node	26.3%	30.2%	36.5 %	29.6%
	Nearest Neighbour	47.7%	50.3%	39.9 %	53.5%
	Furthest Neighbour	45.7%	42.6%	29.0 %	47.5%
EMST	Branch Length	46.0%	44.8%	36.1%	51.4%
	Branches per Node	35.3%	36.2%	35.7%	33.2%
	Nearest Neighbour	48.2%	50.3%	40.0%	53.5%
	Furthest Neighbour	39.9%	36.8%	32.0%	45.3%

Table 4.2: The size of the standard deviations of the syntactic structure analysis parameters relative to the mean. The size of the standard deviations are dependent on the size of the mean, which in turn is dependent on the number of vessels. This table shows the standard deviations relative to the mean, to better facilitate comparisons between the cases.

4.2.3 Fractal Analysis

The five fractal algorithms have been applied to each of the five different image representations; the full cross sections of the vessels, the perimeter of the vessels, the vessel mass centres, Gabriel's Graph and the EMST, in total one hundred analyses. An overview of the dimensions is shown in table 4.6. The results of all the analyses are shown in figures A.8–A.17 in the appendix.

The term *dimension* is not well defined for cross sections of networks, the real investigated parameters are the power law scaling at various parts of the graphs. The term is nonetheless used, although, in a less strict sense, for ease of nomenclature. That being said, most methods provide a large spread of values between the different cases, much more so than the simulation suggests.

While the Correlation, Mass and Sandbox algorithms behave similarly, the Box Counting method is listed in the table with much lower dimensions. It is, however, the method that best reflects the *fractal dimension* of the images, in the true meaning of the words. The Fourier dimensions listed there, are even less compatible with the other dimensions as they may decrease rapidly where other dimensions increase or vice versa. For this reason, the three categories are commented separately. The dimensions in the table are selected, based on the result, as the parts considered the most interesting of the given method and image type. The same selection criteria have been used in all cases.

Box Counting: This method produces two linear regions for most of the images, the exceptions being the EMST curves and two of the cases in the perimeter analysis. The dimension of the first region corresponds to the fine details in the image, and the second to the larger features. In the Cross Section, analysis the first region has values around 1.5 and corresponds to the shapes of the vessels themselves (fine details), the values listed in table 4.6 are from the second region corresponding to less fine details, and are affected by the relative positions of the vessels. The first value is affected by the area of the vessels which increases to higher values for larger vessels, and by the vessel shape which increases with the complexity and tortuosity of the vessel perimeter. The first of these is directly measurable from the image, and the latter is better measured by the dimension of the vessel perimeter. The values in table 4.6 are low, reflecting the dot-like appearance of the vessels at large box-sizes.

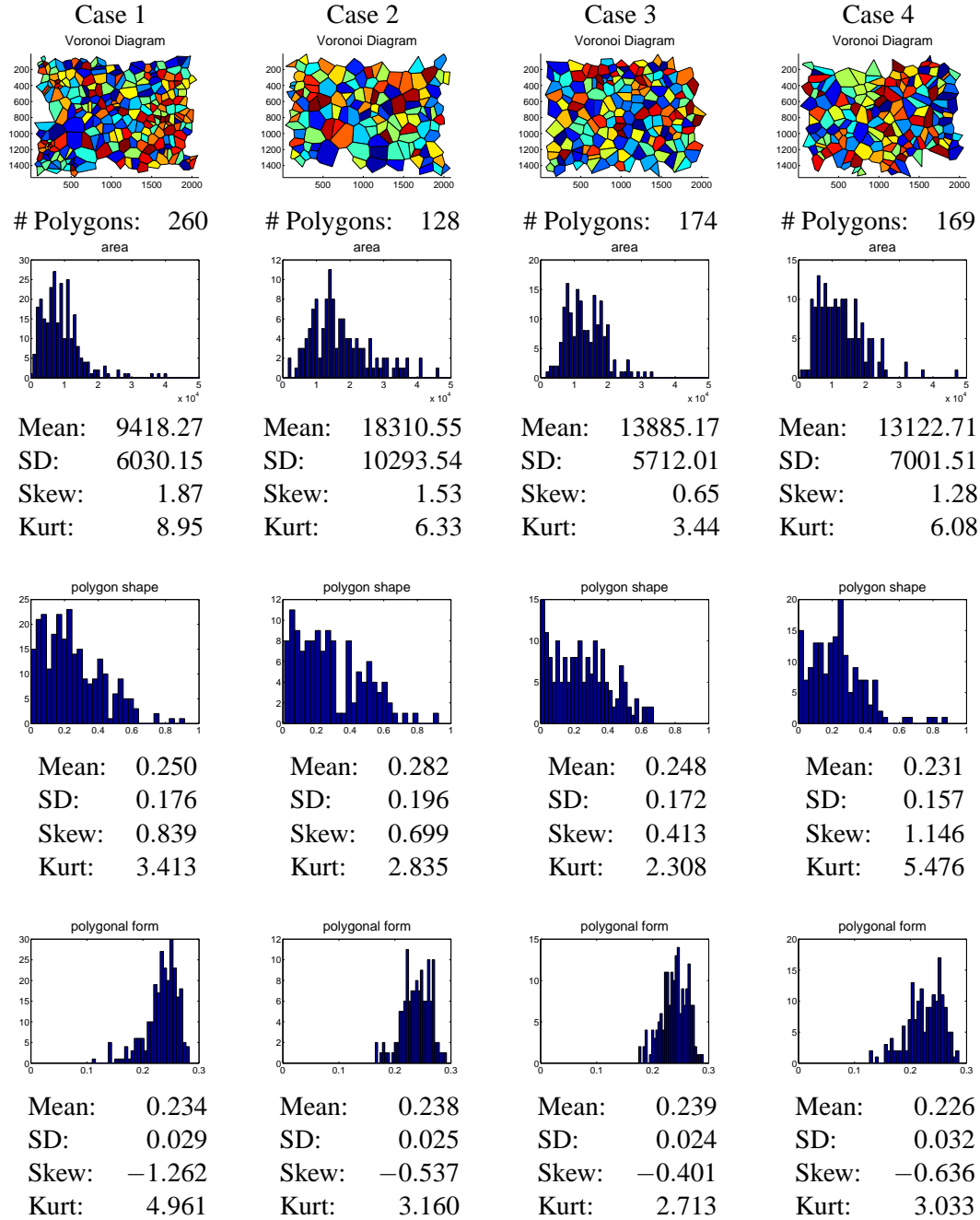


Table 4.3: Voronoi Diagrams and Histograms

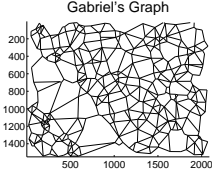
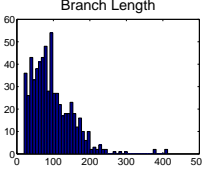
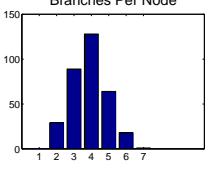
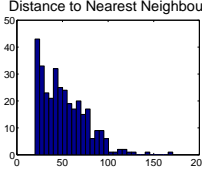
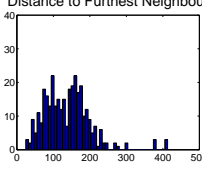
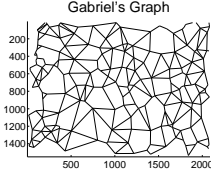
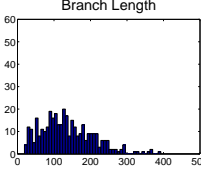
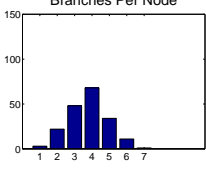
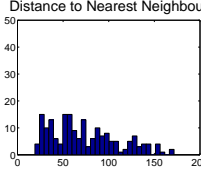
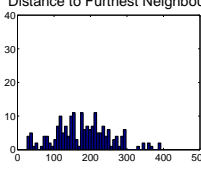
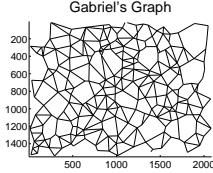
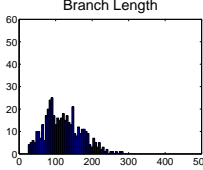
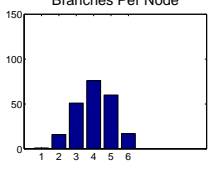
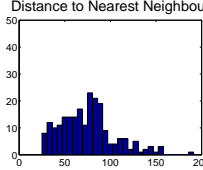
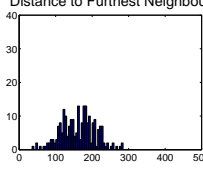
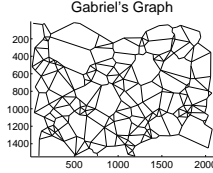
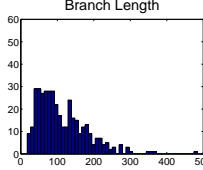
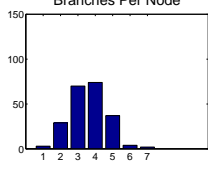
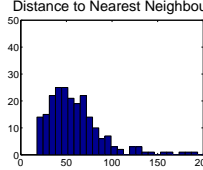
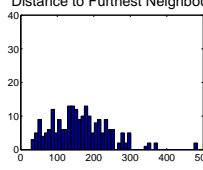
Graph	Nodes	Branches	Length	Branch Length	Branches Per Node	Distance to Nearest Neighbour	Distance to Furthest Neighbour	
	329	636	61216					
	Mean: 96.25	SD: 55.89	Skew: 1.49	Kurt: 7.46	Mean: 3.87	SD: 1.02	Skew: 0.15	Kurt: 2.72
	187	353	47960					
	Mean: 135.86	SD: 70.78	Skew: 0.73	Kurt: 3.50	Mean: 3.78	SD: 1.14	Skew: 0.03	Kurt: 2.81
	221	446	54149					
	Mean: 121.41	SD: 50.20	Skew: 0.46	Kurt: 2.80	Mean: 4.04	SD: 1.07	Skew: -0.14	Kurt: 2.52
	219	395	44601					
	Mean: 112.91	SD: 67.21	Skew: 1.29	Kurt: 5.67	Mean: 3.61	SD: 1.07	Skew: 0.18	Kurt: 3.07
	Mean: 51.84	SD: 24.72	Skew: 1.00	Kurt: 4.51	Mean: 73.97	SD: 37.18	Skew: 0.60	Kurt: 2.47
	Mean: 74.79	SD: 29.83	Skew: 0.75	Kurt: 3.76	Mean: 60.60	SD: 32.40	Skew: 1.82	Kurt: 7.77
	Mean: 138.51	SD: 63.31	Skew: 1.25	Kurt: 6.42	Mean: 179.38	SD: 76.46	Skew: 0.26	Kurt: 3.02
	Mean: 162.37	SD: 47.03	Skew: 0.00	Kurt: 2.76	Mean: 164.43	SD: 78.09	Skew: 0.84	Kurt: 4.63

Table 4.4: Gabriel's Graph Histograms

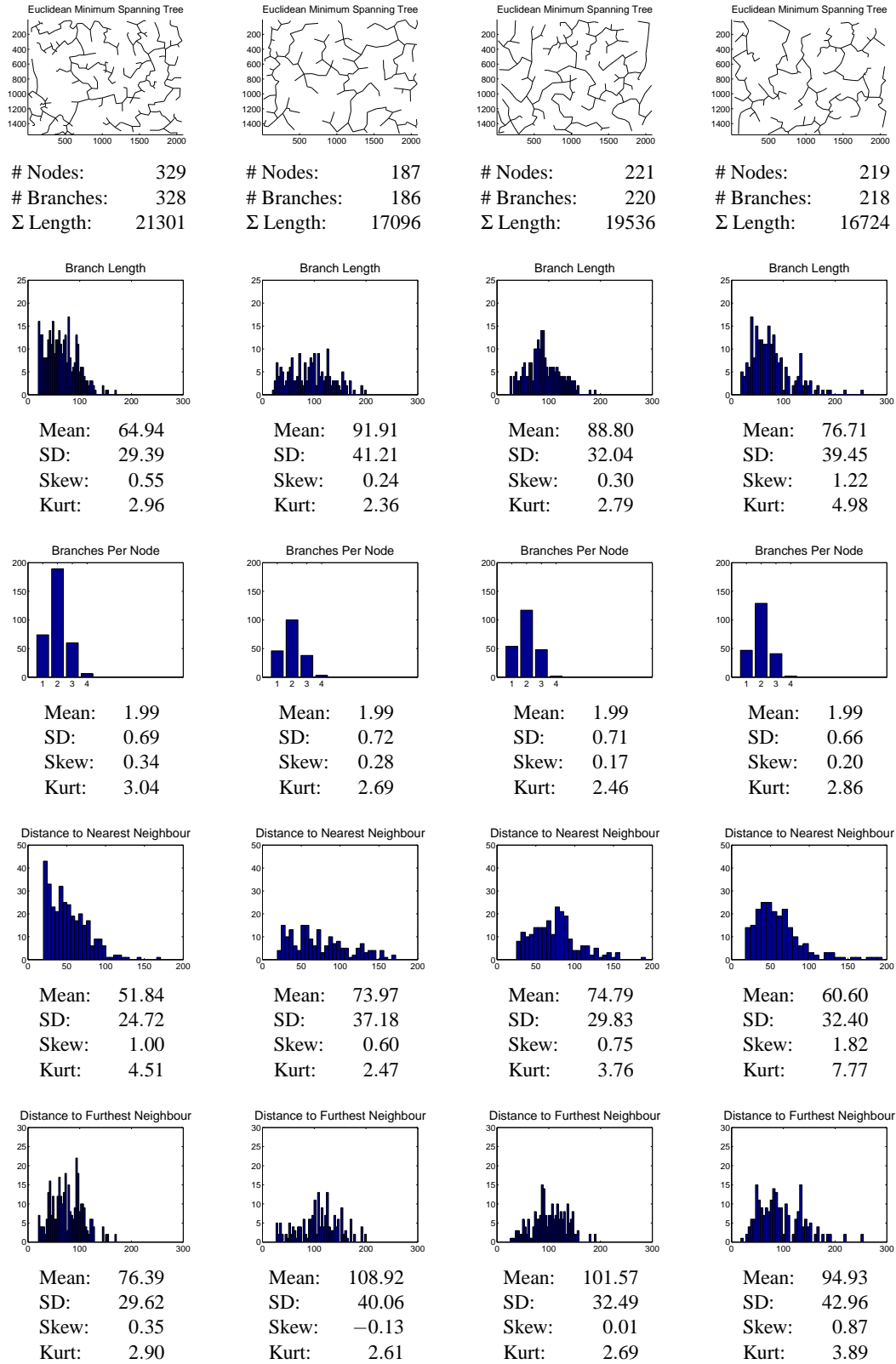


Table 4.5: Euclidean Minimum Spanning Tree Histograms

The Box Counting dimensions of the vessel perimeter are chosen as the first fitted region, two of the cases have only one region and this provides consistency from case to case. This value reflects the tortuosity of the vessel perimeter.

For the vessel centres, the first region has a slope near zero reflecting that the vessels have been reduced to points. At larger sizes, however, a box containing even a single vessel accounts for a large area. This results in a higher dimension (between 0.88 and 1.21) only affected by the relative positions of the vessels. These are the values shown in the table for the perimeter.

For Gabriel's Graph and EMST, the highest dimension in a region which includes boxes larger than 100 pixels is used. These dimensions reflect the scaling of the sizes of the holes and the tortuosity of the graphs.

Correlation, Mass and Sandbox Dimensions:

Cross Section: While the three dimensions are fairly similar in some of the cases they diverge in others. For instance, the sandbox dimension of case 3 and the correlation dimension of case three. The mass dimension has the smallest changes from case to case (although quite large) and is in general the smallest of the three. The shape of the curves are pretty consistent, with the exception of the correlation curves of cases 3 and 4 having four different regions.

Perimeter: These values follow the same pattern as that of the cross section, but are higher. The smallest values tend to have increased the most, however there is no linear relation between the cross section dimensions and the perimeter dimensions.

Centre: In this category the sandbox dimension produces the most consistent curve shape. For all cases it consists of three regions, the first with a dimension close to zero, the latter being a well defined linear region, and the third is the slope of a transition region between the two. The Correlation curve shape varies from case to case, producing between two and four regions of varying linearity. The Mass curve, being calculated from the correlation curve, is affected by this as well. It is far better defined than the correlation curve, and contains three regions (two in case 4, however, a possible third region is too small to be included). The first of these regions has a very high slope (2.11–2.83). The second region is the one included in the table. The third region has a lower slope than the second and starts at radiuses larger than 220 pixels.

Gabriel's Graph: The relative size of the dimensions is consistent regardless of the method, as the mass dimension is the smallest in all cases and the sandbox dimension the greatest. The size of the difference is also fairly consistent. Furthermore, the methods' curves have similar shapes in all the cases, in spite of the varying number of linear regions for the correlation curves, which are a result of variations within the same curve form.

EMST: In addition to the long range correlations shown in table 4.6, the short range correlations are shown in table 4.7. The short range correlations are likely to be influenced by the number of branches per node, the angle between these, and the branch lengths. The long range correlations of the EMST show larger variations in the results of the different methods than Gabriel's Graph does, as well as somewhat larger variations from case to case.

Fourier: The Fourier algorithm proves to be unsuited for this task. It produces a large number of linear regions when applied to the vessel areas or perimeter, i.e. the curve is not well approximated

by linear regions. The regions of the second largest frequencies has been listed in the table, the highest frequency slope being outside the mappable region (slope > 3). This seems to provide a value within a sensible area and one with variation from case to case they are, however, far from well defined.

For the vessel centres the curve is flat, or even increasing for all but the low frequencies. In this case the lowest frequency fit is inserted into the table, this covers, however, a *very* narrow frequency interval, and is again poorly defined.

The curve shape is far better defined for GG and EMST. The region of the second largest frequency is large and well fitted by a linear curve. The values are not directly comparable to the other dimensions, but not necessarily useless. For instance, the values of the Gabriel's Graph fits show a decent spread between 1.33 and 1.56.

4.2.4 Comparison with the Simulation Data

In order to provide a proper comparison with the simulated images, the mass centre images were reduced to 300×400 pixels, by assigning the value 1 to any of the new pixels that contained the centre of at least one mass centre pixel. This operation is similar to the rescaling done in the box counting method. The smaller images were then run through the same scripts as those used in the simulation and the results are plotted as coloured (mostly red) circles in the figures 4.2–4.13. The case names are only indicated by the number of vessels at the x-axis. From left to right, the order is case 2, 4, 3 and 1.

The four cases spread out well for most parameters. In some cases, one or two of them are even far outside the standard deviations. The spread suggests that real image sections may provide an even wider spread in results than the simulations indicate, although four cases is insufficient to make conclusions.

The fractal dimensions require some further comments. As shown in table 4.8, many of the fractal dimensions have changed significantly. To illustrate this further, \times es have been placed in figures 4.6 and 4.7, to represent the dimensions of the full resolution images. The sandbox diameters have been divided by 5.16 to achieve comparative numbers, the conversion is, however, not exact. The two ratios between the two image dimensions are approximately 1.347, not $4/3$. Furthermore, 50 different sandbox sizes are used in both cases, this provides a higher relative resolution in sandboxes for the smaller images. The maximum sandbox size of the larger image is 1357 which becomes 263 in the reduced resolution image. All of the full resolution linear fits includes the highest sandbox diameter, see table 4.15. Consequently, all the \times es are aligned along this value in the *end coordinate* plot. This value is, however, significantly smaller than the maximum value of the reduced images, which is 273. Both values are a little higher than the mean end coordinate and well inside the standard deviation of the simulated images. It should be pointed out that as these values are close to the upper boundary, the standard deviations are most likely related more strongly to the negative deviations, shifting the mean towards lower values, than the positive deviations.

The linear regions that start and end close to the equivalent coordinates of the full resolution images, have very similar fractal dimensions. The dimensions with non-overlapping markers, that is large differences between the two resolutions, also have at least one non-overlapping marker in the start-end coordinate plot. One case with non-overlapping markers in the coordinate plot still has overlapping markers in the dimension plot. The overall impression from this limited material is that linear fits across the equivalent regions result in similar dimensions, while fits from different regions may, or may not, result in large differences. The curve shapes are shown in table 4.15 for

		Case 1		Case 2		Case 3		Case 4	
		Dim	Num	Dim	Num	Dim	Num	Dim	Num
Cross Section	Box Counting	1.0577	2	0.8616	2	0.7991	2	0.9942	2
	Correlation	1.5837	3	1.4433	3	1.9369	4	1.4614	4
	Mass	1.5081	3	1.3423	3	1.6594	3	1.5381	3
	Sandbox	1.5605	3	1.4232	3	1.8080	3	1.9066	3
	Fourier	1.8464	5	1.7620	5	1.2304	5	1.3992	4
Perimeter	Box Counting	1.0676	1	1.0692	2	1.0619	2	1.0703	1
	Correlation	1.6714	4	1.5995	3	1.9474	4	1.7666	3
	Mass	1.5483	3	1.5241	3	1.7000	3	1.6097	3
	Sandbox	1.6171	3	1.7221	3	1.8540	3	1.9195	3
	Fourier	1.5503	5	1.6151	5	1.7398	5	1.5512	5
Centre	Box Counting	1.2187	2	0.8839	2	1.0195	2	0.8891	2
	Correlation	1.7484	2	1.5571	3	1.6393	3	1.8890	4
	Mass	1.7681	3	1.8472	3	2.0418	3	1.7682	2
	Sandbox	1.7684	3	1.8296	3	1.8947	3	1.9126	3
	Fourier	1.5195	4	1.5380	3	1.5902	4	1.5912	4
Gabriel's G	Box Counting	1.6965	2	1.6389	2	1.6782	2	1.5987	2
	Correlation	1.8153	3	1.7945	3	1.8544	4	1.7268	2
	Mass	1.7655	2	1.7447	2	1.8060	2	1.6894	2
	Sandbox	1.8969	3	1.9410	3	1.9695	3	1.8359	3
	Fourier	1.5663	5	1.3259	5	1.4649	5	1.4182	4
EMST	Box Counting	1.0906	1	1.0702	1	1.0744	1	1.0812	1
	Correlation	1.7183	2	1.6668	2	1.8969	3	1.5264	3
	Mass	1.6415	2	1.5950	2	1.7238	2	1.6307	2
	Sandbox	1.8852	3	1.8020	2	1.9306	2	1.9067	3
	Fourier	1.3446	5	1.3225	4	1.3964	5	1.3447	4

Table 4.6: Fractal dimensions of the four cases: These dimensions were automatically selected, one for each image and method, from the possible linearly fitted regions, see figures A.8–A.17. The number of linearly fitted regions are shown next to the dimension. The selection criteria were in general different for the box counting method and the Fourier method, compared to that used in the three other methods. See descriptions in section 4.2.3.

comparison. The non-linearity of the double logarithmic plots gives wider linear regions at the large sandbox diameter for the smaller resolution. These regions do not necessarily, however, occupy a larger portion of the linear range. The linear width is dominated by the highest logarithmic values. Provided that both regions include these values, the linear range occupy a similar percentage.

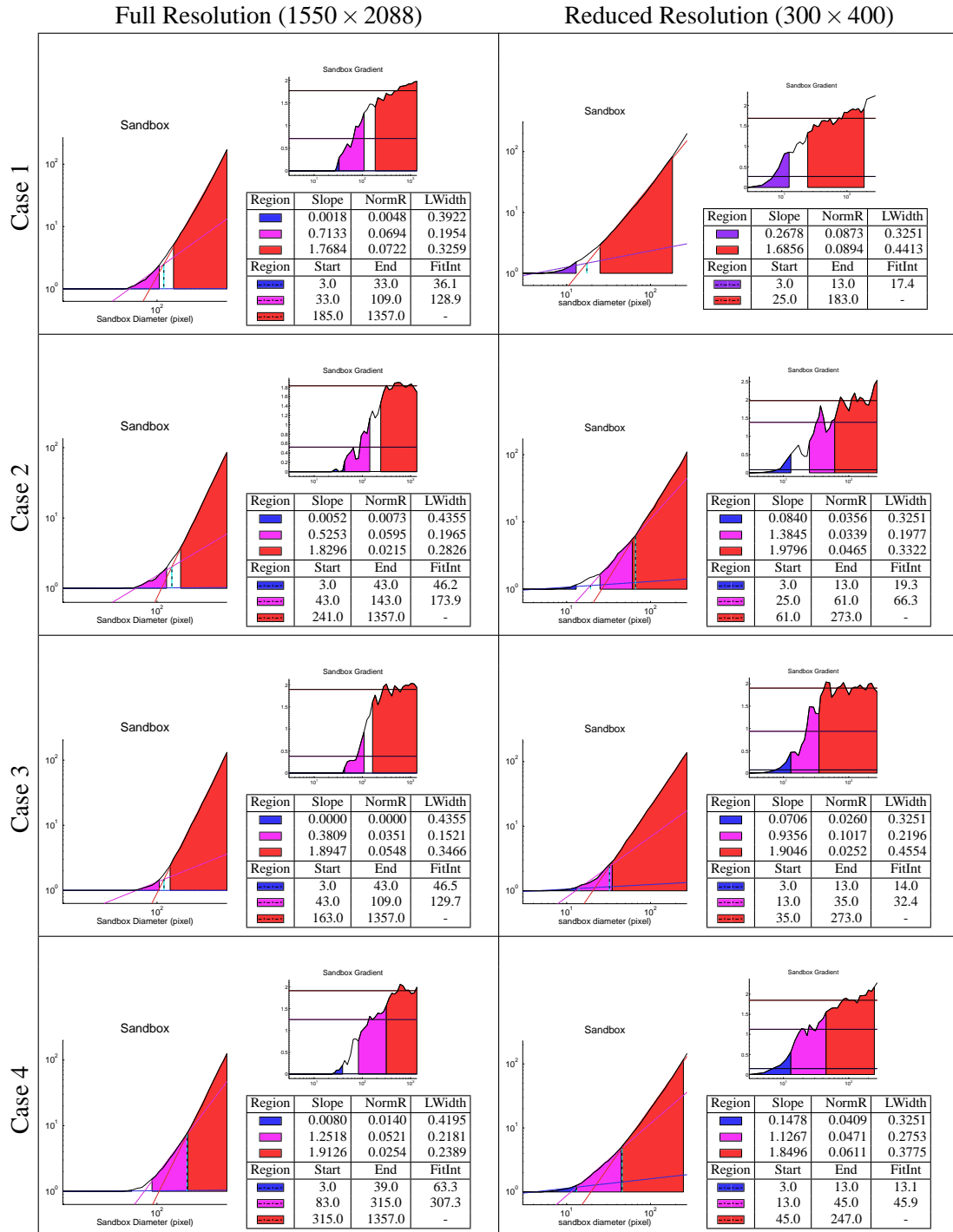


Figure 4.15: Comparison of the sandbox dimensions of the mass centre images at two different resolutions

	Case 1		Case 2		Case 3		Case 4	
	Dim	Num	Dim	Num	Dim	Num	Dim	Num
EMST Box Counting	1.0906	1	1.0702	1	1.0744	1	1.0812	1
Correlation	1.0893	2	1.0648	2	1.0641	3	1.0771	3
Mass	1.1608	2	1.1311	2	1.1357	2	1.1385	2
Sandbox	1.0726	3	1.0549	2	1.0576	2	1.0626	3

Table 4.7: Short range / high detail scaling of the Euclidean Minimum Spanning Tree. These values were obtained from the short radius / small boxes side of the graphs.

	Centre of Mass		Gabriel's Graph		EMST	
	Full	Red.	Full	Red.	Full	Red.
Case 1	1.7684	1.6856	1.8969	1.9177	1.8852	1.8602
Case 2	1.8296	1.9796	1.9410	1.9248	1.8020	1.9297
Case 3	1.8947	1.9046	1.9695	1.9672	1.9306	1.9382
Case 4	1.9126	1.8496	1.8359	1.8546	1.9067	1.8911

Table 4.8: The investigated fractal dimensions of the full and the reduced resolution images.

Chapter 5

Discussion

5.1 Fractal Properties of the Vascular System

The fractal characteristics of tumour vasculature have been investigated in several studies, see section 2.4. Images of two-dimensional tumour set-ups appear to be well-suited for the use of fractal analysis. Three-dimensional systems should work equally well, although they are more difficult to obtain, and will demand much more computational power to process. Three-dimensional network information should be most easily obtained through one of two routes. Either a cast can be made by injecting a liquid substance, which subsequently turns into plastic, into the vascular system, and then corrode away the surrounding tissue with acid. This is an established method, see examples in figure 2.14 and 2.16 (reference [27] and [28]). These studies were performed on mice and human hemicolectomy samples from patients undergoing surgery. The method has clear limitations in its use on human tumours. In these studies, the desired information was extracted through the use of photography from different angles. For the purpose of fractal analysis, another approach may provide more accessible information about the network. If a suitable contrast agent can be applied to the plastinating liquid, a high-resolution micro-CT scanner may be able to extract the vessels. The solid vessel casting allows the use of extremely long exposures, increasing the effective resolution and contrast of the result. Another approach is to use histological sections, sampled at a small, regular interval throughout the tumour. The network can then be reconstructed on a computer. The first of these approaches is most likely practically difficult, and the latter requires an enormous amount of work, due to the extremely large quantity of sections and images.

Through work on two-dimensional tumour models, Gazit et.al. [17][16] and Baish et.al. [2] observed fractal dimension similar to that of invasion percolation. They proposed that some form of percolation process, possibly linked to the extra-cellular matrix, might be responsible for the abnormal network morphology (see sections 2.4.2 and 2.4.4). Bartha et.al. [4] have recently developed what they call a hybrid probabilistic cellular automaton tumour model. In this model they propose that it is the pruning of vessels within the tumour that gives rise to the avascular pockets. The border of the tumour is highly vascularized, but as the tumour grows some of these vessels collapse or otherwise disappear, resulting in hypoxic and necrotic regions. Their model develops networks of a similar fractal dimension as that of invasion percolation, without resorting to any locally random substrate property, (see section 2.4.5).

Too much may have been made of the notion that tumour vasculature points towards a percolation-like process in the development of blood vessels, especially considering the spread in fractal dimensions observed in some of the investigated tumours, see figure 2.24. Nevertheless, invasion percolation

offers a fairly simple model to work with, at least compared to Bartha et.al.'s model. It is recognized that it may not be a process completely representative of the formation of the vascular network. Gazit et.al.'s results do, however, show that as a model used to describe the vasculature morphology in tumours, it works fairly well, at least in the two-dimensional cases. Extrapolating these results to three dimensions, this thesis hypothesizes that three-dimensional percolation clusters should produce two-dimensional cross sections with attributes closer to that of real networks, than by simply assigning vessel locations at random throughout an image.

Regardless of which models are best suited to generate a network, the fractal scaling of tumour networks provides some very interesting implications. If the networks truly are fractal across some significant size-scales, then this is equivalent to asserting a power-law relationship between the vascular and avascular regions across these scales. In other words, the number of avascular pockets of *at least* a given size relates to that of other sizes. This will be investigated further using the formulas of the box counting method from section 2.3.5:

$$\log(N) = D \log(1/S) + \log c \quad (2.3)$$

where c is the intercept of the box counting curve. That is, the number of boxes with content when the box size is maximal, this number is one for all but empty sets. Consequently, if the number of boxes of linear size S_1 is found to be N_1 , then this means that the number of boxes half the size, $N_{1/2}$, and twice the size, N_2 , will be

$$N_{1/2} = \left(\frac{2}{S_1}\right)^D \quad \text{and} \quad N_2 = \left(\frac{1}{2S_1}\right)^D \quad (5.1)$$

The avascular fractions would then be $A_{f1} = 1 - N_1 \cdot S_1^2 = 1 - S^{(2-D)}$, and

$$A_{f1/2} = 1 - 2^D S^{(2-D)} \quad \text{and} \quad A_{f2} = 1 - 2^{-D} \cdot S^{(2-D)} \quad (5.2)$$

Although great care must be exercised in interpreting these fractions, the implications are clear. Within the fractal scaling region, the number of avascular pockets of at least one size relates to that of other sizes. At the dimension of the Sierpinski Carpet for instance, see figure 2.19, which is very close to that of two-dimensional percolation clusters, the number of avascular pockets that are at least one third of some arbitrary unit in size, is eight times higher than those of at least one unit in size. There will, in other words, be a few large and an increasing number of smaller avascular regions. The number of hypoxic regions is the same as the number of avascular regions, with the exception that there are no hypoxic regions for avascular regions smaller than $140\mu m$. Furthermore, their size is smaller and given as a function of the diffusion distance of oxygen and the size and shape of the avascular pocket itself.

In this way the existence of large avascular regions predicts the existence of smaller ones. Furthermore, and perhaps more interestingly, the quantification of small avascular regions within small samples may predict the probability of finding larger ones. In each case, such predictions would be based on a number of extrapolations of data. The two most important are the extrapolations of the linear loglog-region and that of the fractal behaviour. The first is the assumption that the fractal scaling is the same at other size scales than the one investigated. The second is the assumption that the fractal behaviour is the same throughout the tumour. The available literature on the subject is not considered sufficient to make claims towards the validity of these predictions. The verification of such behaviour would require the analysis of the complete vascular system from a sufficiently large number of tumours, and obviously the data must be verified for each tumour type. At this point, they are but interesting speculations based on the properties of mathematical fractals.

5.2 Histological Images

5.2.1 Image Magnification

The four images used in this work were all digitized at $\times 25$ magnification. In addition, images at $\times 50$ and $\times 100$ were acquired from the same samples. The resolution of the images at $\times 25$ has been deemed sufficient to successfully extract the vessels from the image, and to identify the vessel centre in particular. The outline of the vessels can also be established at this resolution, providing sufficient information about the size, area, and shape of the vessels. If the shape of the vessel wall itself is to be studied, higher magnifications should be used.

5.2.2 Image Processing

In order to achieve a well defined replicable measure, some effort has been put into the process of the black and white conversion. The over all goal has been achieved, that all steps in the processes should be mathematically defined, so that it is not necessary for an operator to include a personal judgement into the process. The added benefit is that the process can be automated to a better degree. The quality of the method has not been quantitatively investigated for either false positives or false negatives. Visual inspection does, however, confirm that they are *representative* for the images in question, or at the very least possible images of this tumour type. For the purpose of this thesis, the developed method has been considered sufficient.

5.3 The Studied Parameters

The studied parameters are those available through image analysis of tumour cross-sections. The most obvious of these are the ones categorized as *basic image statistics*. These parameters provide the most basic and easily interpreted results. In particular the number of vessels and vascular density is of interest as it provides means for the comparison to current parameters used by pathologists. The histograms of the distances to the nearest vessels were done as this relates directly to hypoxia. Syntactic Structure Analysis provides a large set of parameters easily applied to this type of images. Furthermore, Gabriel's Graph and the Euclidean Minimum Spanning Tree are well suited for fractal analysis. Fractal Analysis of histological images has been the main emphasis of the this thesis. The large number of SSA-parameters should not be allowed to obscure this.

5.3.1 Fractal Analysis

There are a small, but growing, number of published papers that in one way or another applies fractal analysis to tissue sections (references [42][36][20][24][43]), see section 2.4.3. Although the use of fractal analysis on tumour vasculature appears to be well founded in the case of two-dimensional tumour models, or, if obtainable, three-dimensional tumour vascular networks, the relevance of these methods to tissue sections is less clear.

Defining the Measure

Before discussing the usefulness of the method, there is one pitfall that requires commenting. The feature of the image that is to be analysed must be rigorously defined. For instance, in the study of Spillman et.al. [42] H&E stained sections were converted to greyscale and a threshold was applied at 50% luminosity. This raises the question of how consistent the contrast in the staining is. Especially

because the sample image only appears to occupy a part of the intensity histogram, there are no dark-grey to black pixels. Any changes in the overall luminance in the image will clearly affect the densities of both black and white pixels. The same applies to Sabo et.al. [36], who applied fractal analysis to CD34 stained images. The paper does not specify if, or how, a threshold is applied, however, the box counting method used requires a threshold. Furthermore, the greyscale images shown in the paper has different amounts of background staining of what appears to be cell nuclei. In one image it is almost as dark as the vessels, while in another they are barely visible, see figure 2.27. In either case, it raises the question of whether or not this staining is included in the box counting. In yet another study, this one by Heymans et.al. [24], the Fourier method is applied to greyscale images stained with *Ulex Europaeus*. Although no threshold was required here, any changes in the overall luminance in the image are still likely to cause non-trivial changes in the dimension, see section 2.3.9 and reference [41]. For further information on these papers, see section 2.4.3.

To avoid any uncertainties as to how the image material is processed and analysed, the method in section 3.1 has been implemented. No greyscale images have been used in the fractal analysis, only black and white. See section 5.2.2 for further discussion of the method.

The Study of Tissue Sections

The meaningfulness of applying fractal analysis to tissue sections requires some debating. The relevance of the other parameters is far more evident as these methods are more or less designed for this purpose. The discussion will be limited to the study of images where a threshold has been applied to remove all pixels but those of the vasculature. It applies to all forms of such images, regardless of the staining used to highlight the vasculature. The difficulties involved in extracting the vessel information, and the means to do it, will, however, vary greatly for different staining protocols. A cross section of a tumour's vascular network is essentially a series of dots of various shapes, distributed throughout an image. Although the overlaying structure is the sum of all the cross sections, it is not clear how useful the information from a rather limited number of sections is, when it comes to describing the overall network characteristics. It is clear that any single cross section can be a part of a variety of different network morphologies. Parameters such as the vessel diameter and the intervessel distance should correlate with similar parameters in the cross sections. Other morphological parameters, however, such as the branching angles and interbranch distances (see section 2.1.6), are not obtainable at all. This suggests that networks with large variations in these parameters may result in similar cross sections. On the other hand, it is also evident that the overall network morphology will affect the distributions of vessels in the cross section. Although the fractal scaling of the sections may, or may not, correlate directly to the fractal dimension of the network, it seems more than plausible that it does relate to the overall morphology.

The Different Approaches to Fractal Analysis

The fractal analysis may be implemented in a number of different ways. The ones investigated here, see section 4.2.3, are the analyses of the combined vessel lumina and walls, the outer perimeter of the vessel walls, and the centre of mass. In addition the fractal dimensions of Gabriel's Graph and the Euclidean Minimum Spanning Tree were calculated. There is a variety of different algorithms available for these purposes, five of which have been investigated here. Table 4.6, showing the fractal analysis results of the four cases, clearly illustrates the large variety of results obtainable by these methods.

It should be pointed out that these cross sections are not fractals. The methods may, nevertheless, still be of value, provided that the curves have well defined power law obeying regions. As figures A.8–A.17 show, this certainly appears to be the case. The behaviour of the curves are, however, very different for each method and image type. Each power law scaling region occupy some 20–40% of the curves. This makes the terminology *fractal dimension* imprecise. None of these numbers can be said to be the dimension of the image as a whole. The measured quantity is the power law behaviour of that *particular* region. Nonetheless, the term *dimension* is still used of these parameters, as the alternative is an unnecessarily complicated terminology. It is, however, important to bear these distinctions in mind.

For each image, twenty-five different curves have been obtained. The sandbox dimension of the vessels' mass centres at large sandbox sizes has been considered the most attractive of these. The sandbox method was found to be the most precise for the given test images in section 3.5. It has a reasonable execution time and a well defined curve shape for the mass centre images. The use of images that only contain the mass centre, ignores the shape and sizes of the vessels, leaving their numbers and relative positions as the only involved parameters.

The fractal characteristics of the vessels themselves have not been investigated in particular, although the data is readily available from figures A.8–A.11. The left hand side of the curves would be the interesting region for this purpose. To reduce the effect of vessel sizes, which is easier measured by the vessel area parameter, the perimeter data would be preferable. Histograms of the locally connected fractal dimension, see section 2.3.9, of the vessels may be better suited for this characterization. They would show the distribution of fractal dimensions of the different vessels. Furthermore, higher magnification images should prove to be a better source of information. See section 3.1 for a list of the different area coverages and resolutions at different magnifications.

The fractal dimensions of Gabriel's Graph and the Euclidean Minimum Spanning Tree may also prove to be valuable parameters. In case of the EMST, the low regions of the graph may be considered in addition to the high, although the latter seems more promising and the differences are certainly a lot more pronounced.

Changes in the Resolution of an Image

The relative size of the pixels representing the mass centres is impossible to conserve across different image resolutions. Changes in image resolution will, however, cause changes for all methods and images. Lowering the resolution, increases the size of the smallest features it is possible to obtain information from, and, perhaps more importantly, it reduces the number of different low resolutions, or in the context of sandbox analysis of the mass centre images, it reduces the number of different large diameters. The effect of resolution changes at large sandbox diameters should, however, not cause any large effects on the result, although some is certainly present.

The effect of resizing the images has not been studied here in particular. Some data are, however, available from the comparison between the results of the simulations and the histological sections. These are, however, limited to the sandbox dimension of the mass centre at large sandbox diameters. The large deviations in these results serve to illustrate that further development is needed in order to reproduce the results at different resolutions. The differences in the results are reasonably small for many of the values, but varies greatly for others. The size and sign of the difference in value varies a lot, suggesting that the effect is not caused by the reduced resolution, and the good correlation of differences in start and end coordinates points toward the routine that automatically selected the linear regions, see figures 4.6–4.7 and section 4.2.4.

It is apparent that this method requires further development with respect to robustness. Two

suggestions for ways to improve this will be provided. 1) The power of the *NormR* factor in the quality of fit function can be increased to further emphasize good fits over large areas. 2) The selection criteria with respect to the sandbox method applied to mass centres can be modified. Rather than using the highest, automatically selected region, a region at high dimensions can be used as a requirement in the algorithm. Another possibility is to specify at least one of the sides (end or starting point) of the linear region. Other approaches to the *quality of fit* function may be considered as well.

Although the lack of robustness at a five-fold resolution change does not necessarily imply that the routine lacks robustness at high resolutions, it should certainly be addressed. Different institutions cannot be expected to have similar digital sensors. Furthermore, if stability between the high and low resolution images can be obtained, then low resolution images will be an advantage to computation speed when large quantities of images are processed.

5.3.2 Cumulative Histograms of the Distance to the Nearest Vessel

These histograms offer the most straight forward approach of estimating the magnitude of avascular pockets in a tissue section. Keep in mind, however, that these distances constitute the maximum distance to the nearest vessel, a closer vessel may easily be found outside the image plane. All four sections investigated here, are highly vascularized. This results in short distances to the vessels. The investigated areas of histological sections, at least in the context of vasculature parameters, are typically chosen from the most vascular areas in the section. These areas are known as vascular hot-spots. The relevance of this parameter depends highly on how the distance distribution within a hot-spot relate to those outside hot-spots. These parameters may additionally be studied outside the hot-spots, however, parameters that can be obtained within the same data material, are more likely to be used. In addition, the area to be studied should be chosen according to reproducible criteria. The vascular hot-spots represent such a criterion, and it is well established for histological studies. Using the same areas for all parameters, allows for correlations between parameters to be more easily established, as the additional variable of second area selection is removed. The hope is, however, that a parameter that relates directly to hypoxic fractions, or similar quantities, may be obtained. This must be established irrespective of the correlation with other parameters.

5.3.3 Syntactic Structure Analysis

The Syntactic Structure Analysis offers a broad panel of parameters, especially when each of the four histogram parameters are considered. Many of these parameters are directly dependent on the number of vessels in the image. The total length of the tree, the area of the Voronoi polygons, or the distances to the nearest neighbour are good examples. Other parameters are not, for instance the number of branches per node, or the shape of the Voronoi polygons. Because the number of vessels is established by counting them, the main interest of the SSA-parameters is the information they contain with respect to the morphology of the vessels. The relationship between these parameters and the number of vessels is shown, at least to some extent, in the simulation results, see figures 4.8–4.13. The dependencies on the number of vessels are non-linear, especially at lower vessel counts, and the size of the standard deviations and the mean value generally have different dependencies on the number of vessels.

In addition to the SSA-parameters used by Weyn et.al.[43], the distance to the furthest neighbour has been investigated. The distance to the nearest neighbour is identical in Gabriel's Graph and the Euclidean Minimum Spanning Tree. This makes it redundant to calculate it for both graphs, suggesting that one of them should be removed, or possibly replaced by the distance to the furthest neighbour.

Although the two parameters certainly relate to each other, they relate to the vessel distribution as well. The ratio between them are 1.49, 1.46, 1.36 and 1.57 for each of the four cases respectively. At this point it is recommended to keep both parameters for further studies.

5.4 Vessel Simulations

Grizzi et.al. [20] performed a simulation where circular vessels of a fixed size were distributed throughout an image, according to a uniform probability distribution, with the one restraint that no vessels were allowed to overlap, see section 2.4.3. They investigated the changes of the box counting dimension as a function of the number of vessels, for vessel counts ranging from 5 to 50. They found that the fractal dimensions increased with the number of vessels and that there were a non-zero variance for most data points. Two similar simulations have been performed, investigating the dependency of all used image analysis parameters on the number of vessels. The variance found in Grizzi et.al.'s results was quite small. The ones found in these simulations have, however, proven to be significantly larger, increasing the potential clinical value of the parameters beyond that of the vessel density.

5.4.1 The Purpose of the Vessel Simulations

The vessel simulations have served a two-fold purpose. The first is that they show which of the investigated parameters that strongly depend on image information other than the number of vessels. The size of the standard deviations is considered to relate to this dependency, with large values suggesting a strong link to the relative positions of the vessels. This was the main reason for carrying out the simulations. The purpose of this study at this point is to evaluate how well the various parameters may be suited for further studies. Any parameter that relates solely to the number of vessels is redundant. The mean number of branches per node in the Euclidean Minimum Spanning Tree is an example of this, albeit not a good one, as this was already known from theory. However, other parameters come close, especially when the number of vessels exceeds 200.

The second benefit is the obvious, that the dependency of the parameters to the number of vessels is revealed. These relationships are important for two reasons, the size of the standard deviations must be considered in relation to the slope of the curve. It is hard to avoid some uncertainty in the image processing, and false positives or negatives may affect the analysed images. If the slope is large compared to the standard deviation, then this uncertainty renders any information about the morphology of the vessel distribution useless. The second potential use of these relationships becomes important once the parameters are used to classify the morphology histological sections. Simulations, such as these, although at larger volumes, may provide the necessary curves needed to remove the information that relates to the number of vessels rather than their relative positions, i.e., the morphology. The parameters may, for instance, be replaced by their distance to the mean, either expressed in absolute values, or perhaps better, as the number of standard deviations that separate the data point from the mean. Relating it to the number of standard deviations may be an advantage because the sizes of these deviations have been found to vary with respect to the vessel count for many of the parameters. In effect, the deviations from the mean are expected to be smaller in images with many vessels, at least for most parameters.

Two series of simulations were performed, the first based on a uniform random probability distribution and the last on the cross sections of three-dimensional percolation clusters. The simplest of these approaches offers a method that is easy to understand and implement, and fast to execute. It was, however, hypothesized that variations in results from such a simple approach are not represen-

tative for real distributions. A second approach, based on three-dimensional percolation theory, was implemented with the following hypotheses:

- a) The results from the two approaches should be different.
- b) The percolation approach should, in particular, give somewhat greater variations for many of the parameters, especially the cumulative histogram parameters.
- c) The percolation method is a better model for the vascular network. It should, consequently, correlate better to the results from tissue section analysis than the random approach does.

The two first have been confirmed, but the last requires a much larger data material than the four sections used here. More data from simulations must be gathered as well, in particular the fractal dimensions of the percolation simulation are far from smooth. The limited material available does, however, certainly suggest that the variations in real samples may be even larger for real sections than either of the simulations suggest, at least for some parameters.

5.4.2 The Simple Random Simulation

The main strength of this simulation is its simplicity, and it has been named accordingly. The simple rules of the simulation implies that very few assumption are made, this is, however, misleading. To assume that vessel cross sections are well approximated by uniform probability distributions, is fairly naive. Even more so, when the entire process of angiogenesis, endothelial growth factor diffusion fields, hypoxia inducible factors and the available morphological data, is taken into consideration. This is in principle no less an assumption than the assertion made of the other simulation, namely that the vessel images are well approximated by percolation theory. The main benefit of this method is that it places very few restrictions on the output and it is easily understandable and implementable.

5.4.3 The Percolation Simulation

This simulation is based on an extrapolation of Gazit et.al.'s [17] find, that two-dimensional network models have a similar fractal behaviour as that of invasion percolation. The assumption is then made that a similar behaviour is present in three-dimensional systems. There are ways of implementing a three-dimensional percolation model. As shown in table 2.3 there are several different three-dimensional percolation models in the same universality class; site and bond random percolation, site and bond non-trapping invasion percolation, and trapping site invasion percolation. In addition there are several lattice-related parameters that must be specified, including the shape of the lattice and its size. The computation time involved increases rapidly for larger lattices, and the complexity of the code is higher for more complicated lattices, for example a hexagonal lattice.

The choice was made to use the non-trapping bond invasion percolation on a simple cubic lattice. Bond percolation on a cubic lattice consists of directional bonds and one of these directions can easily be extracted as a cross section. This gives the appearance of a situation similar to that in a histological section. The similarity is, however, superficial. In a real section, all vessels crossing a plane are included. Because the sections are extremely thin, the odds of a vessel running alongside the plane is close to zero. In the invasion bond percolation model, however, two thirds of the vessels will never cross the image plane.

The model was chosen over a site percolation model, in part because the visual representations of bond clusters bear a closer resemblance to a vessel network, and the concept is consequently easier to communicate, but primarily because the horizontal bonds serve to reduce the local dependence

between one cross section and the next. This may be achieved by other means as well, for instance, through the site percolation method by only utilizing every other section, but in a bond model it is a built in effect. The choice between bond and site percolation is not regarded as important to the model outcome. Trapping invasion percolation has, however, been ruled out, as no physical equivalence of the *trapping* process exists in the system, and random percolation introduces a percolation threshold to the model.

The local dependence of one section on the next may be considered a weakness in the percolation model. Each section does not represent an independent image. This effect could be reduced by only sampling the possible sections at some set interval. This effect is, however, closely related to the strength of the model, that the probability of finding a vessel is greater in the vicinity of other vessels. This is expected to increase the probability of avascular areas and the likelihood that patterns such as those observed in case four are generated. The network morphology will, in general, be different from that of the uniform distribution. Only 169 different clusters were used in the simulation, as the great variations in the mean show, it would have benefited greatly from a larger data material. The general outline of the behaviour of the parameters is, however, expected to be relatively representative.

5.4.4 Evaluating the Two Simulations

The predictions of the hypothesis was confirmed, the percolation model provided larger, and in part much larger, deviations for many of the observed parameters. More importantly, it has been established that the distribution used in the simulations has a large impact on some parameters, albeit less so for others. This implies that if real data material is to be classified according to its deviation from a simulation curve, as suggested above, then the model of the simulation is of great importance.

Sudden Value Changes at Specific Vessel Numbers

The sudden changes at specific values in some of the parameters of the percolation simulation were unexpected, i.e. the sudden drops in the cumulative histogram parameters at 50, 78 and 97 vessels, as well as the sudden increase in the fractal dimension at 50 vessels. The cause of these effects are unknown. However, they do not appear to affect any of the SSA-data. All parameters were calculated by the same script for each vessel count, thus removing the possibility that a mix-up of data files could affect some parameters without effecting all of them. The shape of the curves also contradict this possibility. The images near these transitions have not been inspected visually and the cause may very well be found there. The sections containing few vessels are most likely to be found near the end of the cluster, edge effects may consequently be involved. However, the sharp transitions at no less than three places would still be unexpected. Another possible cause is that, by chance, some clusters with very low occupation may have been generated, in effect that a route with relatively little resistance has been found from one side to the other in these clusters. This would cause relatively many sections with few vessels to be generated throughout the cluster, causing a single simulation to obtain a high percentage of the sections at these vessel counts. As the total number of clusters is as low as 169, this is possible, but not likely, leaving the possibility that this is an attribute of the three-dimensional percolation cluster in some way. In addition, a new simulation containing a far higher number of clusters would be a good follow-up study, in order to obtain better defined curves. This should also reveal if these features have appeared by chance, or by the nature of the percolation cluster.

Cumulative Histograms

As expected, these values are far higher than those of the random simulation. These are perhaps the parameters with the largest differences. The mean is increased by a factor of two and the deviation by at least a factor of four.

Fractal Dimension

The fractal dimensions are, as expected, different for these two simulations, confirming that this dimension, the sandbox dimension at high diameters for the mass centres, can be used to classify morphology. The fractal dimension of the percolation simulation is lower than those of the random simulation, even at peaks of the uneven percolation curve.

SSA-Parameters

Many of these values have much greater variations for the percolation simulation than the random simulation, especially at large vessel numbers. The curve shape is, however, generally quite similar.

5.5 Relevance to Clinical Data

There are a few published papers investigating the relevance of fractal analysis to clinical data. De Felice et.al. [13] investigated and found an increased fractal dimension of the oral vasculature in patients with *Lynch cancer family syndrom I and II* compared to the control group. Weyn et.al. [43] investigated the correlation of fractal analysis, syntactic structure analysis and the microvessel density's ability to predict tumour prognosis using a K-nearest neighbour test. They concluded that the SSA-parameters in particular may be useful as a prognosticator in general diagnostic pathology, but found only a mediocre prognostic value for the fractal parameters. Sabo et.al. [36] investigated the microvessel density and fractal dimension of renal cell carcinoma sections and concluded that the fractal dimension was inversely associated with tumour necrosis, and that tumour necrosis was the only investigated parameter with significant independent prognostic value. The literature on the other parameters, the microvessel density in particular, is more extensive. In addition there is some limited material on other uses of fractal analysis in cancer research (e.g. Spillman et.al.'s [42] study of H&E sections, or Craciunescu et.al.'s [8] study of dynamic contrast-enhanced magnetic resonance images)

This thesis shows the dependency of the investigated image analysis parameters on the number of vessels. It also provides an estimate of the size of the variance that can be expected for a given number of vessels. Furthermore, it provides concrete examples of what analysis results of real histological data may look like. This provides a basis for evaluating which of the parameters that should be included if a smaller panel of parameters is to be preferred. The study has not, however, investigated how these parameters relate to other clinical or tumour specific data. Although there certainly are ways to improve the methods used in this study, some of which have been addressed, it seems more important to establish the relevancy of the parameters themselves. If these parameters, or at least some subset of them, do indeed correlate to clinical data, either patient specific, e.g. prognosis, survival, metastasis incidence, treatment response, or tumour specific, e.g. tumour grade, hypoxic or necrotic fractions, or established vascular parameters, then the correlations should be identifiable with the current implementation of the methods.

5.6 Suggestions for Further Studies

This section provides suggestions for further studies within this area, based on this thesis. Some of them relate to possible improvements of the methods developed here, while other relate to further investigations regarding the possible applications of these results.

5.6.1 Correlation Studies

A correlation study should be performed, investigating how these parameters relate to other clinical parameters. These may be parameters relevant to a wide range of areas, such as prognosis, diagnosis, metastatic potential, parameters relevant to specific treatments, in particular hypoxia, and most likely other areas as well. This is considered the most important step in the continued investigation of these methods, and, consequently, the most important of the follow-up suggestions.

5.6.2 Segmentation of the Images

Other possible approaches, as to how the threshold is applied to the images, may be investigated for some or all steps of the process. For the selected method, the number of false positive and false negative vessels should be investigated. All parameters should be investigated for robustness with respect to false negatives and positives. This can be done by randomly removing or adding vessels to the image and to investigate the effects on the parameters. This should provide a clear indication of how well the segmentation routine, as a minimum requirement, needs to perform. If some or all parameters show an exceedingly large sensitivity with respect to the segmentation, this indicates that the results of the segmentations need to be manually inspected and corrected of any errors. Furthermore, and far more importantly, it may suggest that the relevant parameter is so highly dependent on the vessel distribution and/or number, that it for all practical purposes may be considered chaotic, and, subsequently, quite possibly of little use, although correlation studies should be used to confirm this.

5.6.3 Fractal Analysis

A histogram of the fractal dimensions of each vessel perimeter can be made, although images at higher magnifications are recommended. This would be relevant as a description of the individual vessel characteristics.

The effect of different image resolutions on the different dimensions and images may be studied further. Limited information for sandbox dimension of large diameters of the mass centre is available from the four cases, the other dimensions have not been investigated at all. Furthermore, this is the only dimension that has been investigated in the simulations.

5.6.4 Vessel Simulation

Both simulations may be performed for more images in order to better establish the curve shapes. The odd behaviour of some of the percolation parameters at specific vessel counts may be investigated further in a larger simulation. The other fractal algorithms can, if desired, be included in simulations as well, although this will significantly increase the computation time of the simulation.

Chapter 6

Conclusion

This thesis provides the initial investigation of a broad panel of image analysis parameters which may be used to quantify vessel distribution patterns in vascular cross sections. The computer programs required to calculate the parameters have been written as a part of the thesis, and the analysis methods have been tested on four cases. These methods are now established and adapted to the study of histological images. In this way, the thesis provides a basis for further investigations, including correlation studies.

It has been important to establish whether or not it is meaningful to perform fractal analysis on vessel cross-sections. Although one must be careful with how any resulting *dimensions* are interpreted, fractal analysis algorithms do appear to be able to differentiate between different vessel distributions. The different fractal methods investigated, all produce different results, and it is, with the possible exception of the Fourier method, difficult to completely exclude any of the parameters from further studies.

The Box Counting method provides the results that best represent the *fractal dimension* of the images. The method results in well defined curve shapes for many of the image types, but it requires an exceedingly long computation time, limiting its usefulness.

The Correlation method provides some mixed results, for some of the image types it produces well defined power law scaling regions, whilst for others, the mass centre images in particular, it fails.

The Mass method is based on the correlation method. In spite of this, it does appear to have a better defined curve shape, in particular it does not require a five point average smoothing. This method is recommended over the Correlation dimension, and is the most promising of the three algorithms utilising the Fast Fourier Transform. It is somewhat faster than the sandbox algorithm, and should be given extra consideration for very large image sizes, or three-dimensional studies.

The Sandbox method is considered the most promising of these methods, providing well defined curve shapes for all the image types.

The Fourier method has a long list of problematic features, in addition to the fact that it only provides power law scaling regions for Gabriel's Graph and the Euclidean Minimum Spanning Tree, not for the images.

Taking all algorithms and images types into consideration, the recommended fractal parameters are the Sandbox dimensions of the mass centre images, Gabriel's Graph and the Euclidean Minimum Spanning Tree, at large sandbox diameters. In addition the EMST dimension at small diameters may be investigated. At this point in time it is not recommended to exclude any of the SSA-parameters from the Syntactic Structure Analysis.

The vessel simulations provides a framework for further research, as they give an indication towards which results one can expect from histological sections. They also provide a clear demonstration of how such simulations may be used to map analysis results to a linear scale independent of the number of vessels in the image. This is important if one wishes to find parameters which solely depends on the vessel distribution pattern.

The simulations and the analysis of the four cases provide good reasons for being optimistic about the possible use of these parameters. Based on these results, it is recommended to proceed with comparative studies, as this is the only way the real usefulness of the parameters can be identified.

Bibliography

- [1] Takayuki Asahara, Toyoaki Murohara, Alison Sullivan, Marcy Silver, Rien van der Zee, Tong Li, Bernhard Witzenbichler, Gina Schatterman, and Jeffrey M. Isner. Isolation of putative progenitor endothelial cells for angiogenesis. *Science*, 275:964–67, 1997.
- [2] James W. Baish, Yuval Gazit, David A Berk, Mutsumi Nozue, Laurence T. Baxter, and Rakesh K. Jain. Role of tumor vascular architecture in nutrient and drug delivery: An invasion percolation-based network model. *Microvascular Research*, 51:327, 1996.
- [3] James W. Baish and Rakesh K. Jain. Fractals and cancer. *Cancer Research*, 60:3683–8, 2000.
- [4] K. Bartha and H. Rieger. Vascular network remodeling via vessel cooption, regression and growth in tumors. *Journal of Theoretical Biology*, 241:903–918, 2006.
- [5] J. Martin Brown. The hypoxic cell: A target for selective cancer therapy—eighteenth Bruce F. Cain memorial award lecture. *Cancer Research*, 59:5863–70, 1999.
- [6] Peter Carmeliet and Rakesh K. Jain. Angiogenesis in cancer and other diseases. *Nature*, 407:249–257, 2000.
- [7] Yong S. Chang, Emmanuelle di Tomaso, Donald M. McDonald, Rosemary Jones, Rakesh K. Jain, and Lance L. Munn. Mosaic blood vessels in tumors: Frequency of cancer cells in contact with flowing blood. *Proceedings of the National Academy of Sciences of the United States of America*, 97:14608–13, 2000.
- [8] O. I. Craciunescu, S. K. Das, and S. T. Clegg. Dynamic contrast-enhanced mri and fractal characteristics of percolation clusters in two-dimensional tumor blood perfusion. *Journal of Biomechanical Engineering*, 121:480–86, 1999.
- [9] Massimo Cristofanilli, Chusilp Charnsangavej, and Gabriel N. Hortobagyi. Angiogenesis modulation in cancer research: Novel clinical approaches. *Nature Reviews*, 1:415–26, 2002.
- [10] Brad St. Croix, Carlo Rago, Victor Velculescu, Giovanni Traverso, Katharine E. Romans, Elizabeth Montgomery, Anita Lal, Gregory J. Riggins, Cristoph Lengauer, Bert Vogelstein, and Kenneth W. Kinzler. Genes expressed in human tumor endothelium. *Science*, 289:1197–202, 2000.
- [11] Heike Daldrup, David M. Shames, Michael Wendland, Yoshitaka Okuhate, Thomas M. Link, Werner Rosenau, Ying Lu, and Robert C. Brasch. Correlation of dynamic contrast-enhanced mr imaging with histologic tumor grade: Comparison of macromolecular and small-molecular contrast media. *American Journal of Roentgenology*, 171:941–49, 1998.

- [12] M. W. Dewhirst, H. Kimura, S. W. Rehmus, R. D. Braun, D. Papahadjopoulos, K. Hong, and T. W. Secomb. Microvascular studies on the origins of perfusion-limited hypoxia. *British Journal of Cancer Supplement*, 27:247–51, 1996.
- [13] C De Felice, G Bianciardi, S Parrini, G M Fadda, M Marini, R N Laurini, and R J Kopotic. Abnormal vascular network complexity: a new phenotypic marker in hereditary non-polyposis colorectal cancer syndrome. *Gut*, 52:1764–7, 2003.
- [14] Robert Folberg and Andrew J. Maniotis. Vasculogenic mimicry. *Acta Pathologica, Microbiologica Et Immunologica Scandinavica*, 112:508–25, 2004.
- [15] Liv Furuberg, Jens Feder, Amnon Aharony, and Torstein Jøssang. Dynamics of invasion percolation. *Phys. Rev. Lett.*, 61:2117–2120, 1988.
- [16] Yuval Gazit, James W. Baish, Nina Safabakhsh, Michael Leunig, Laruence T. Baxter, and Rakesh K. Jain. Fractal characteristics of tumor vascular architecture during tumor growth and regression. *Microcirculation*, 4:395–402, 1997.
- [17] Yuval Gazit, A David, Berk Micael Leunig, Laurence T. Baxter, and Rakesh K. Jain. Scale-invariant behavior and vascular network formation in normal and tumor tissue. *Phys. Rev. Lett.*, 75:2428, 1995.
- [18] P. K. Y. Goon, G. Y. H. Lip, P. S. Stonelake, and A. D. Blann. Circulating endothelial cells, endothelial progenitor cells, and endothelial microparticles in cancer. *Neoplasia*, 8:79–88, 2006.
- [19] Arjan W. Griffioen and Grietje Molema. Angiogenesis: Potentials for pharmacologic intervention in the treatment of cancer, cardiovascular diseases, and chronic inflammation. *Pharmacological Reviews*, 52:237–68, 2000.
- [20] Fabio Grizzi, Carlo Russo, Piergiuseppe Colombo, Barbara Franceschini, Eldo E. Frezza, Everardo Cobos, and Maurizio Chiriva-Internati. Quantitative evaluation and modeling of two-dimensional neovascular network complexity: the surface fractal dimension. *Biomed Central Cancer*, 5:14, 2005.
- [21] Eric J. Hall. *Radiobiology for the Radiobiologist*. Lippincott Williams & Wilkins, Philadelphia, PA 19106 USA, fifth edition, 2000.
- [22] Douglas Hanahan and Judah Folkman. Patterns and emerging mechanisms of the angiogenic switch during tumorigenesis. *Cell*, 86:352–64, 1996.
- [23] Hiroya Hashizume, Peter Baluk, Shunichi Morikawa, John W. McLean, Gavin Thurston, Sylvie Roberge, Rakesh K. Jain, and Donald M. McDonald. Openings between defective endothelial cells explain tumor vessel leakiness. *American Journal of Pathology*, 156:1363–80, 2000.
- [24] O. Heymans, S. Blacher, F. Brouers, and G.E. Piérard. Fractal quantification of the microvasculature heterogeneity in cutaneous melanoma. *Dermatology*, 198:212–7, 1999.
- [25] J. Holash, P. C. Maisonpierre, D. Compton, P. Boland, C. R. Alexander, D. Zagzag, G. D. Yancopoulos, and S J. Wiegand. Vessel cooption, regression, and growth in tumours mediated by angiopoietins and vegf. *Science*, 284:1994–98, 1999.

- [26] Natalia L. Komarova and Vladimir Mionov. On the role of endothelial progenitor cells in tumor neovascularization. *Journal of Theoretical Biology*, 235:338–49, 2005.
- [27] M. A. Konerding, W. Malkusch, B. Klapthor, C. van Ackern, E. Fait, S. A. Hill, C. Parkins, D. J. Chaplin, M. Presta, and J. Denekamp. Evidence for characteristic vascular patterns in solid tumours: quantitative studies using corrosion casts. *British Journal of Cancer*, 80:724–32, 1999.
- [28] M.A. Konerding, E. Fait, and A. Gaumann. 3d microvascular architecture of pre-cancerous lesions and invasive carcinomas of the colon. *British Journal of Cancer*, 84:1354–62, 2001.
- [29] D.-S. Lee, H. Rieger, and K. Bartha. Flow correlated percolation during vascular remodeling in growing tumors. *Phys. Rev. Lett.*, 96:058104(4), 2006.
- [30] Anders J. Leu, David A. Berk, Athina Lymboussaki, Kari Alitalo, and Rakesh K. Jain. Absence of functional lymphatics within a murine sarcoma: A molecular and functional evaluation. *Cancer Research*, 60:4324–7, 2000.
- [31] B. B. Mandelbrot. How long is the coast of Britain? *Science*, 156:636–38, 1967.
- [32] Andrw J. Maniotis, Robert Folberg, Angle Hess, Elisabeth A. Seftor, Lynn M.G. Gardner, Jacob Pe'er, Jeffrey M. Trent, Paul S. Meltzer, and Mary J. C. Hendrix. Vascular channel formation by human melanoma cells *in Vivo* and *in Vitro*: Vasculogenic mimicry. *American Journal of Pathology*, 155:739–52, 1999.
- [33] Daniela Massi, Alessandro Franchi, Milena Paglierani, Sheyda Ketabchi, Lorenzo Borgognoni, Umberto Maria Reali, and Marco Santucci. Vasculogenic mimicry has no prognostic significance in pt3 and pt4 cutaneous melanoma. *Human Pathology*, 35(4):496–502, 2004.
- [34] Sybill Patan, Shigeru Tanda, Sylvie Roberge, Rosemary C. Jones, Rakesh K. Jain, and Lance L. Munn. Vascular morphogenesis and remodeling in a human tumor xenograft. *Circulation Research*, 89:732–39, 2001.
- [35] D Ribatti, B Nico, E Crivellato, A M Roccaro, and A Vacca. The history of the angiogenic switch concept. *Leukemia*, September 2006.
- [36] Edmond Sabo, Albina Boltenko, Yanina Sova, Avi Stein, Shira Kleinhaus, and Murray b. Resnick. Microscopic analysis and significance of vascular architectural complexity in renal cell carcinoma. *Clinical Cancer Research*, 7:533–537, 2001.
- [37] Adrian P Sheppard, Mark A Knackstedt, W V Pinczewski, and Muhammad Sahimi. Invasion percolation: new algorithms and universality classes. *Journal of Physics A: Mathematical and General*, 32:L521–L529, 1999.
- [38] Anil K. Sood, Mavis S. Fletcher, Chris M. Zahn, Lynn M. Gruman, Jeremy E. Coffin, Elisabeth A. Seftor, and mary J.C. Hendrix. The clinical significance of tumor cell-lined vasculature in ovarian carcinoma. *Cancer Biology and Therapy*, 1:6:661–64, 2002.
- [39] Anil K. Sood, Elisabeth A. Seftor, Mavis S. Fletcher, Lynn M. G. Garnder, Paul M. Heidger, Richard E. Buller, Richard E. B. Seftor, and Mary J. C. Hendrix. Molecular determinants of ovarian cancer plasticity. *American Journal of Pathology*, 158:1279–1288, 2001.

- [40] Peter Vaupel. The role of hypoxia-induced factors in tumor progression. *The Oncologist*, 9:10–17, 2004.
- [41] Richard F. Voss. Local connected fractal dimension analysis of early chinese landscape paintings and x-ray mammograms. *Fractal Image Encoding and Analysis, Ed: Fisher Y., NATO ASI series F*, 159:279–97, 1998.
- [42] Jr W. B. Spillman, J. L. Robertson, W. R. Huckle, B. S. Govindan, and K. E. Meissner. Complexity, fractals, disease time, and cancer. *Physical Review E*, 70:061911–1 – 061911–12, 2004.
- [43] B. Weyn, W. A. A. Tjalma, P. Vermeulen, A. can Daele, E. Van Marck, and W. Jacob. Determination of tumour prognosis based on angiogenesis-related vascular patterns measured by fractal and syntactic structure analysis. *Clinical Oncology*, 16:307–316, 2004.
- [44] George D. Yancopoulos, Samuel Davis, Nicholas W. Gale, John S. Rudge, Stanley J. Wiegand, and Jocelyn Holash. Vascular-specific growth factors and blood vessel formation. *Nature*, 407:242–48, 1998.
- [45] Bruce R. Zetter. Angiogenesis and tumor metastasis. *Annual Review of Medicine*, 49:407–24, 1998.

A Additional Figures and Tables

A.1 Tests of the Fractal Algorithms

1. Data Tables for the Different Algorithms A-2
2. Fractal Analysis of Sierpinski's Carpet and Gasket A-5
3. Fractal Analysis of two Random Site Percolation Clusters A-6
4. Fractal Analysis of the two Random Site Percolation Clusters' Backbones..... A-7
5. Fractal Analysis of the two Random Site Percolation Clusters' Elastic Backbones..... A-8
6. Fractal Analysis of a Filled Circle and a Filled Square A-9
7. Fractal Analysis of a Square and a Circle Perimeter A-10

A.2 Fractal Analysis of the Histological Images

1. Fractal Dimensions of the Vessel Cross Section Images A-12–A-13
2. Fractal Dimensions of the Vessel Perimeter Images A-13–A-14
3. Fractal Dimensions of the Vessel Mass Centre Images A-16–A-17
4. Fractal Dimensions of Gabriel's Graph A-18–A-19
5. Fractal Dimensions of the Euclidean Minimum Spanning Tree A-20–A-21

Box Counting Algorithm with 30 points

Shape	Norm of Res.	Slope	Dimension	True Dimension	Absolute Error
Sierpinski Carpet	0.1646	-1.8616	1.8616	1.8929	-0.0313
Sierpinski Gasket	0.0499	-1.6235	1.6235	1.5850	0.0385
Percolation Cluster 1	0.0588	-1.8426	1.8426	1.8958	-0.0532
Percolation Cluster 2	0.0540	-1.8371	1.8371	1.8958	-0.0587
Percolation Backbone 1	0.0583	-1.6490	1.6490	1.6432	0.0058
Percolation Backbone 2	0.0540	-1.6472	1.6472	1.6432	0.0040
Perc. Elastic Backbone 1	0.1135	-1.1874	1.1874	1.1307	0.0567
Perc. Elastic Backbone 2	0.1274	-1.1568	1.1568	1.1307	0.0261
Circle Perimeter	0.0842	-1.0725	1.0725	1.0000	0.0725
Circle	0.0484	-1.9571	1.9571	2.0000	-0.0429
Square Perimeter	0.0132	-1.0038	1.0038	1.0000	0.0038
Square	0.0413	-1.9797	1.9797	2.0000	-0.0203
Mean of Error		0.0001	Standard Deviation of Error		0.0429

Correlation Algorithm (smoothed)

Shape	Norm of Res.	Slope	Dimension	True Dimension	Absolute Error
Sierpinski Carpet	0.0575	-0.1423	1.8577	1.8929	-0.0352
Sierpinski Gasket	0.0870	-0.4280	1.5720	1.5850	-0.0130
Percolation Cluster 1	0.0245	-0.1152	1.8848	1.8958	-0.0110
Percolation Cluster 2	0.0275	-0.1190	1.8810	1.8958	-0.0148
Percolation Backbone 1	0.0602	-0.3302	1.6698	1.6432	0.0266
Percolation Backbone 2	0.0515	-0.3519	1.6481	1.6432	0.0049
Perc. Elastic Backbone 1	0.0703	-0.9023	1.0977	1.1307	-0.0330
Perc. Elastic Backbone 2	0.0727	-0.9394	1.0606	1.1307	-0.0701
Circle Perimeter	0.0436	-0.9982	1.0018	1.0000	0.0018
Circle	0.0361	-0.0231	1.9769	2.0000	-0.0231
Square Perimeter	0.0906	-0.9419	1.0581	1.0000	0.0581
Square	0.0351	-0.0228	1.9772	2.0000	-0.0228
Mean of Error		-0.0110	Standard Deviation of Error		0.0323

Mass Algorithm

Shape	Norm of Res.	Slope	Dimension	True Dimension	Absolute Error
Sierpinski Carpet	0.0545	1.8624	1.8624	1.8929	-0.0305
Sierpinski Gasket	0.0440	1.5848	1.5848	1.5850	-0.0002
Percolation Cluster 1	0.0485	1.8943	1.8943	1.8958	-0.0015
Percolation Cluster 2	0.0567	1.8881	1.8881	1.8958	-0.0077
Percolation Backbone 1	0.0536	1.6892	1.6892	1.6432	0.0460
Percolation Backbone 2	0.0697	1.6574	1.6574	1.6432	0.0142
Perc. Elastic Backbone 1	0.0495	1.1887	1.1887	1.1307	0.0580
Perc. Elastic Backbone 2	0.0498	1.0928	1.0928	1.1307	-0.0379
Circle Perimeter	0.0393	1.0304	1.0304	1.0000	0.0304
Circle	0.0570	1.9763	1.9763	2.0000	-0.0237
Square Perimeter	0.0457	1.0528	1.0528	1.0000	0.0528
Square	0.0556	1.9770	1.9770	2.0000	-0.0230
Mean of Error		0.0064	Standard Deviation of Error		0.0336

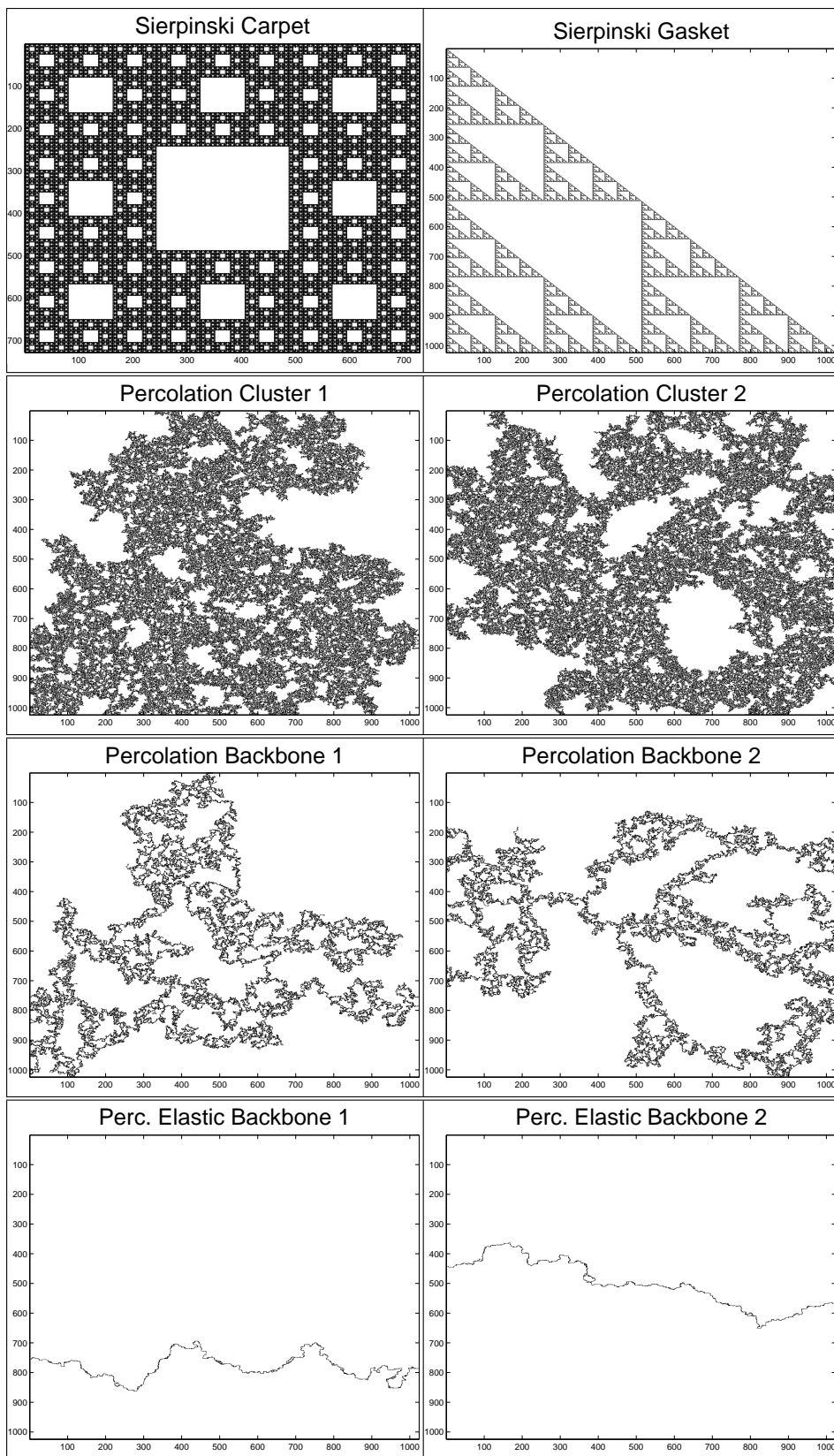
Sandbox Algorithm with 50 points

Shape	Norm of Res.	Slope	Dimension	True Dimension	Absolute Error
Sierpinski Carpet	0.0203	1.8817	1.8817	1.8929	-0.0112
Sierpinski Gasket	0.0434	1.5670	1.5670	1.5850	-0.0180
Percolation Cluster 1	0.0568	1.9076	1.9076	1.8958	0.0118
Percolation Cluster 2	0.0357	1.8949	1.8949	1.8958	-0.0009
Percolation Backbone 1	0.0419	1.6852	1.6852	1.6432	0.0420
Percolation Backbone 2	0.0374	1.6543	1.6543	1.6432	0.0111
Perc. Elastic Backbone 1	0.0432	1.1613	1.1613	1.1307	0.0306
Perc. Elastic Backbone 2	0.0536	1.1072	1.1072	1.1307	-0.0235
Circle Perimeter	0.0164	1.0034	1.0034	1.0000	0.0034
Circle	0.0323	1.9746	1.9746	2.0000	-0.0254
Square Perimeter	0.0370	1.0207	1.0207	1.0000	0.0207
Square	0.0377	1.9720	1.9720	2.0000	-0.0280
Mean of Error		0.0010	Standard Deviation of Error		0.0230

Fourier Algorithm (smoothed)

Shape	Norm of Res.	Slope	Dimension	True Dimension	Absolute Error
Sierpinski Carpet	0.6984	-1.9503	1.9503	1.8929	0.0574
Sierpinski Gasket	0.0855	-1.5682	1.5682	1.5850	-0.0168
Percolation Cluster 1	0.5202	-1.7977	1.7977	1.8958	-0.0981
Percolation Cluster 2	0.6759	-1.9838	1.9838	1.8958	0.0880
Percolation Backbone 1	0.1041	-1.6001	1.6001	1.6432	-0.0431
Percolation Backbone 2	0.0745	-1.6051	1.6051	1.6432	-0.0381
Perc. Elastic Backbone 1	0.2393	-1.2122	1.2122	1.1307	0.0815
Perc. Elastic Backbone 2	0.1474	-1.1371	1.1371	1.1307	0.0064
Circle Perimeter	0.0721	-1.0112	1.0112	1.0000	0.0112
Circle	0.0999	-2.9984	2.9984	2.0000	0.9984
Square Perimeter	0.0262	-0.9976	0.9976	1.0000	-0.0024
Square	0.1053	-2.9666	2.9666	2.0000	0.9666
Mean of Error		0.0046	Standard Deviation of Error		0.0586

Table A.1: Data tables from the testing of the fractal algorithms. See figures A.2-A.7 for graphs and further information on the power law scaling of the individual shape.



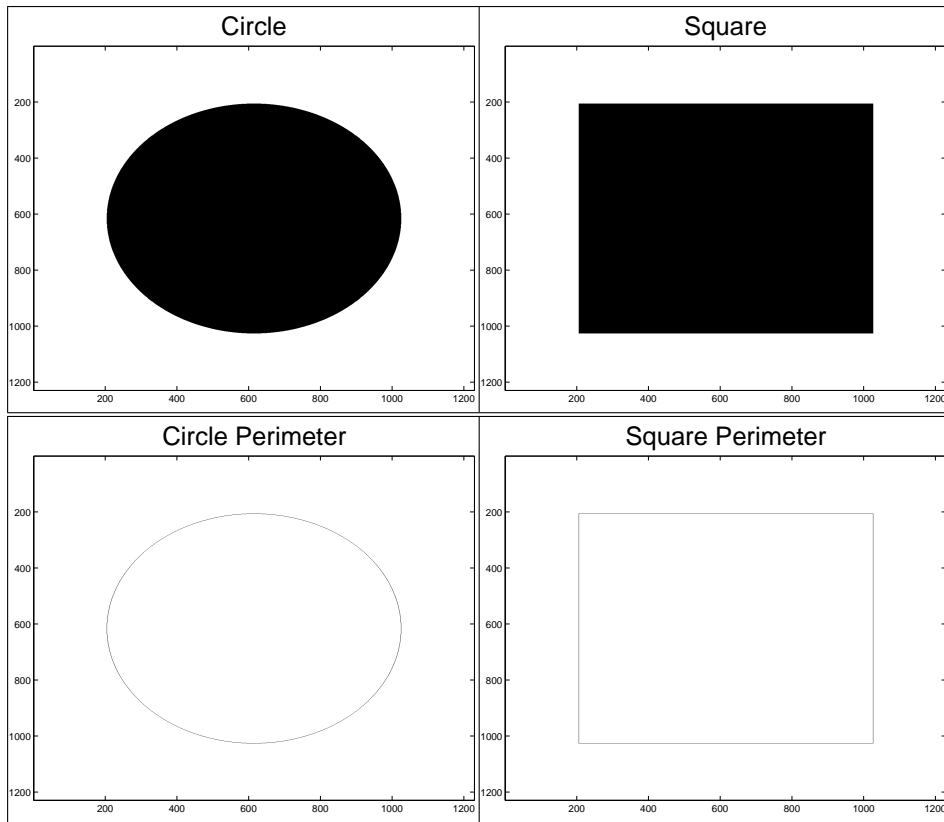


Figure A.1: The test shapes of the fractal algorithms

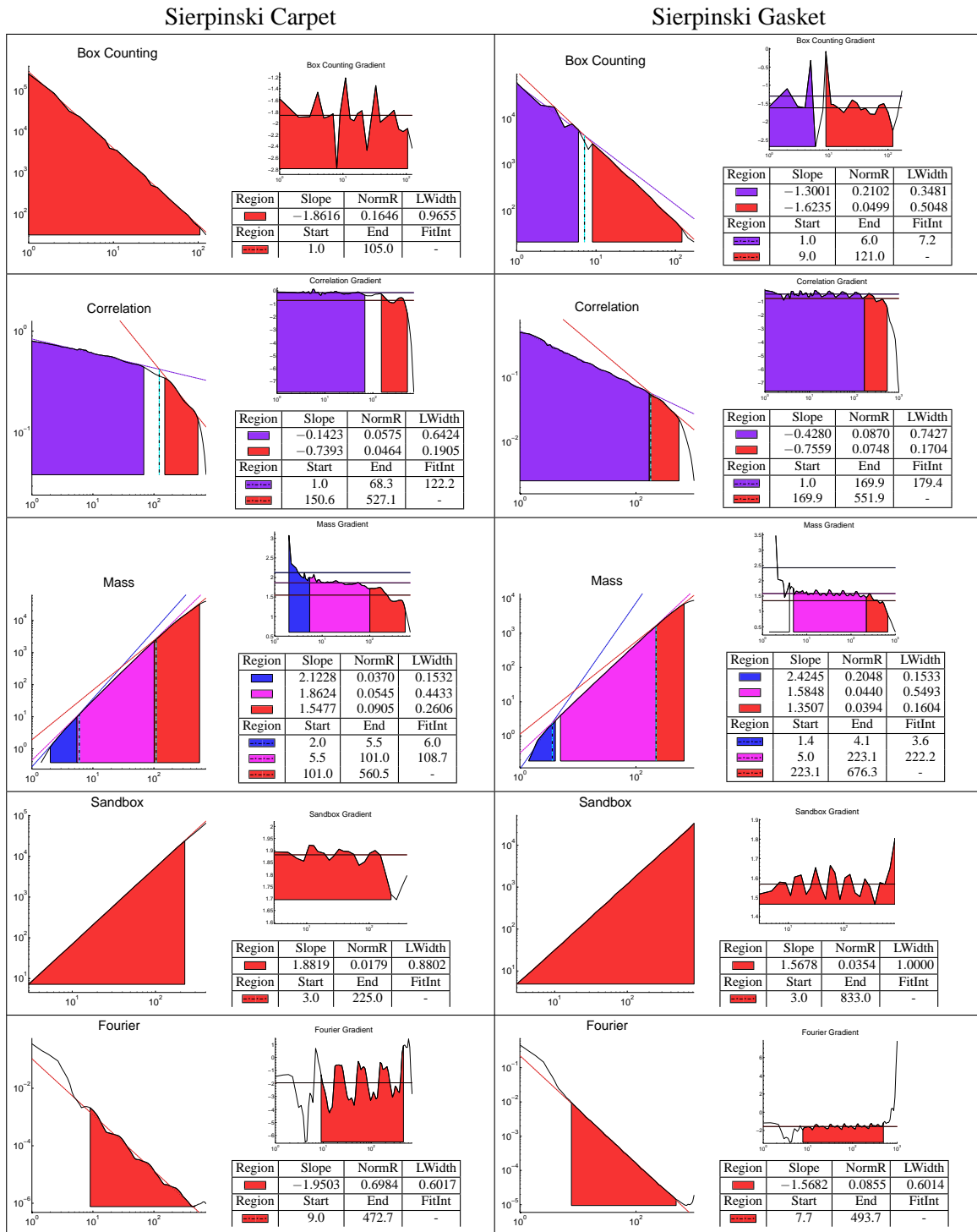


Figure A.2: Testing the algorithms on the Sierpinski Carpet and the Sierpinski Gasket

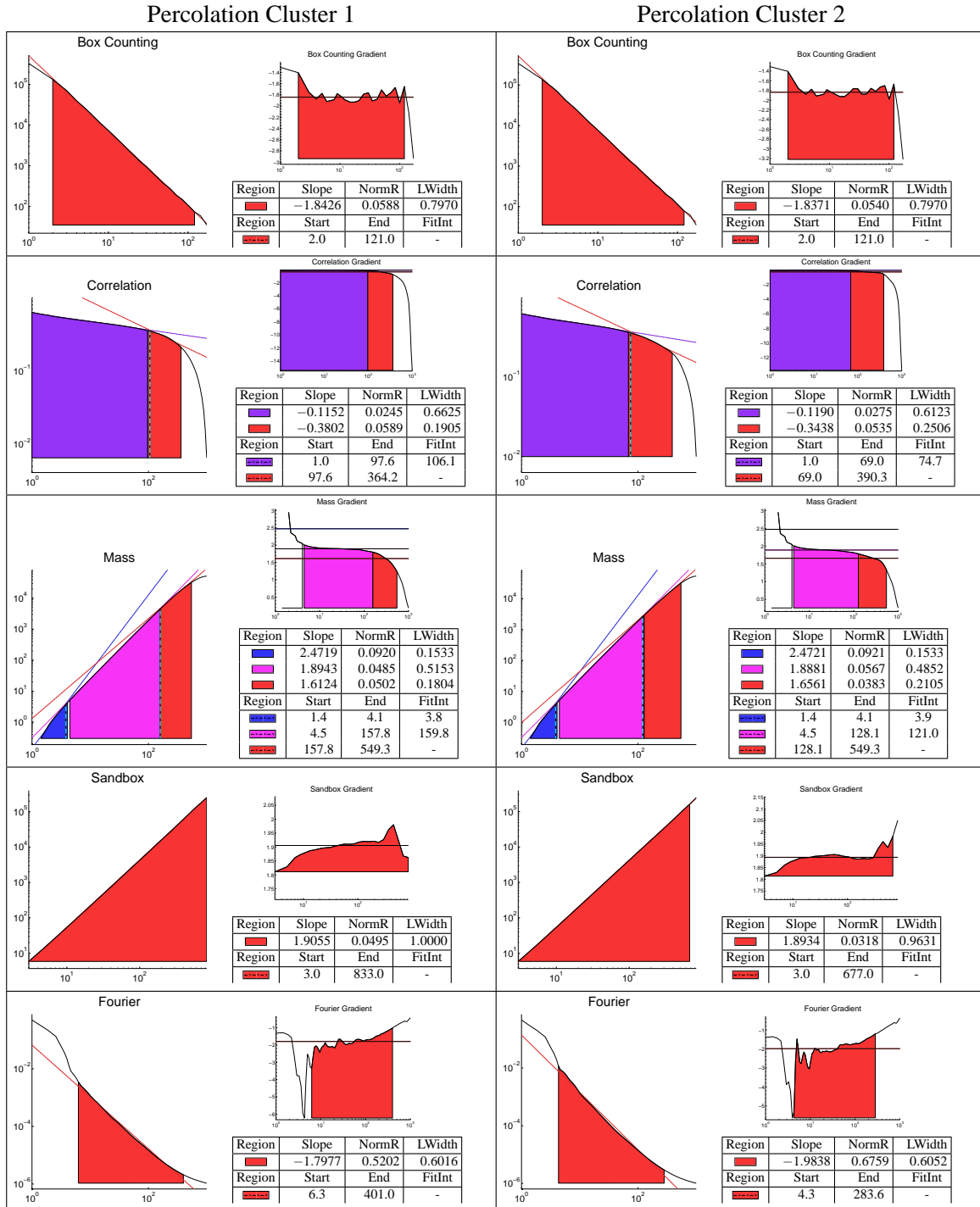


Figure A.3: Testing the algorithms on two 1024x1024 random percolation clusters

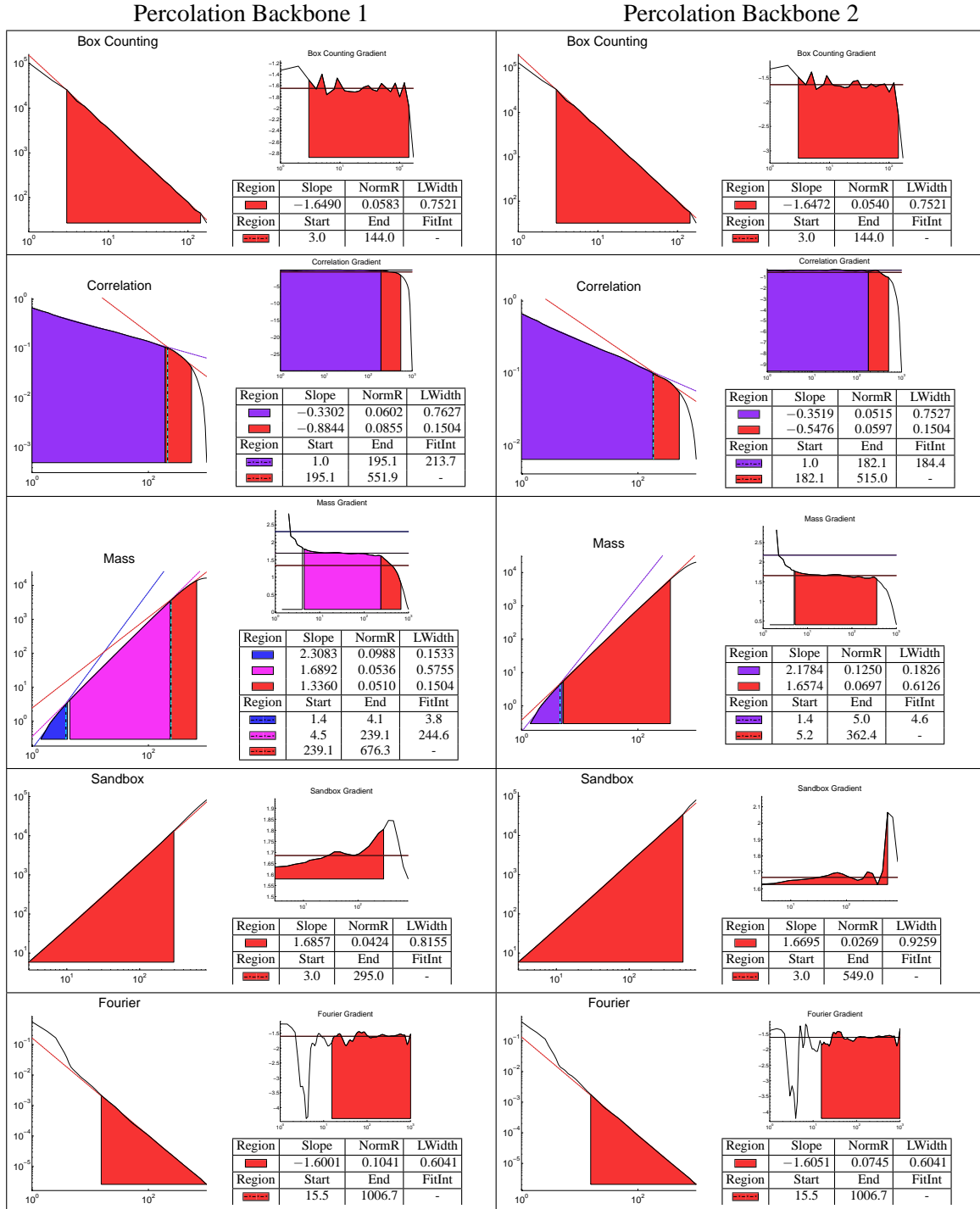


Figure A.4: Testing the algorithms on 1024x1024 percolation backbones

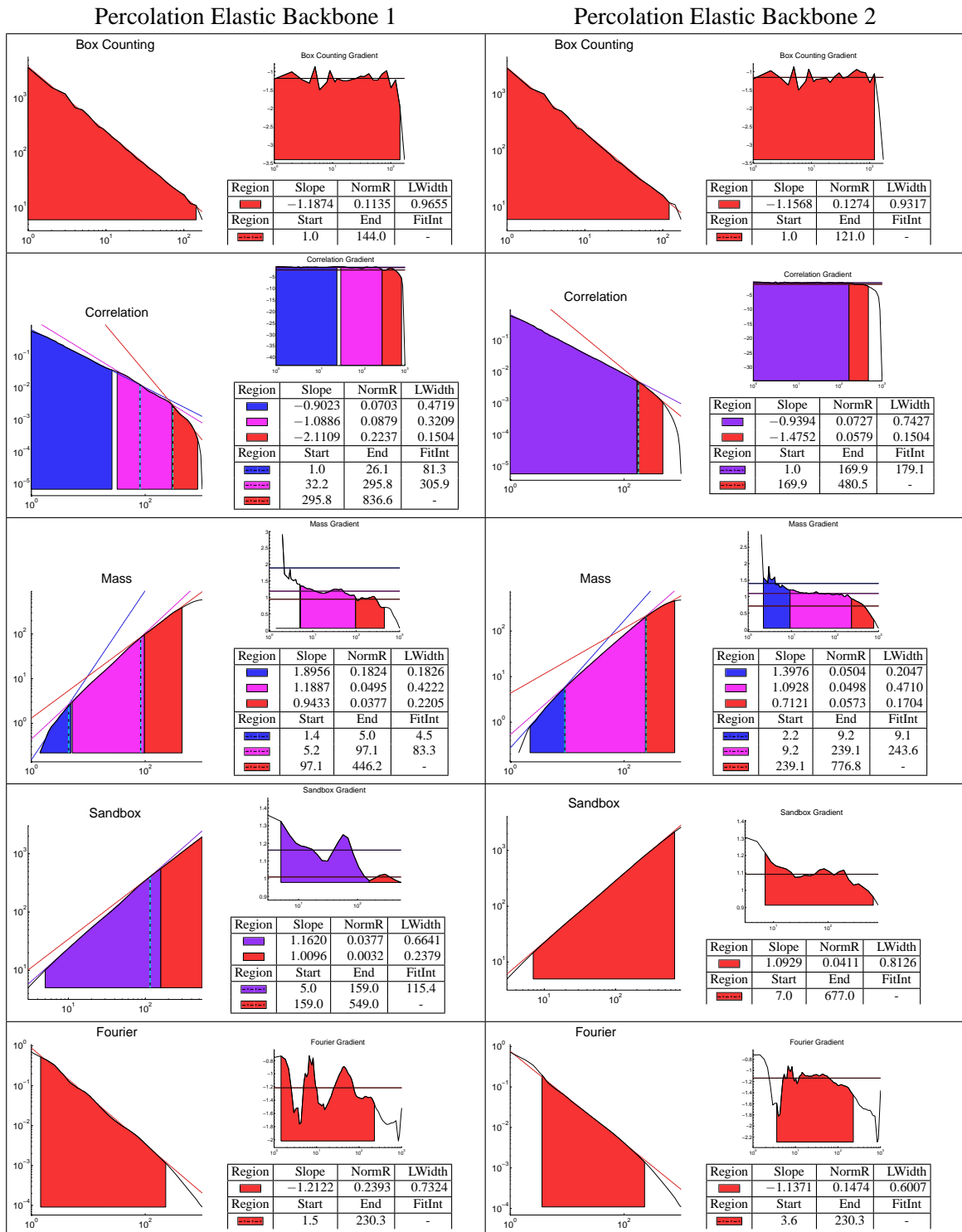


Figure A.5: Testing the algorithms on the elastic backbone of two 1024x1024 percolation clusters

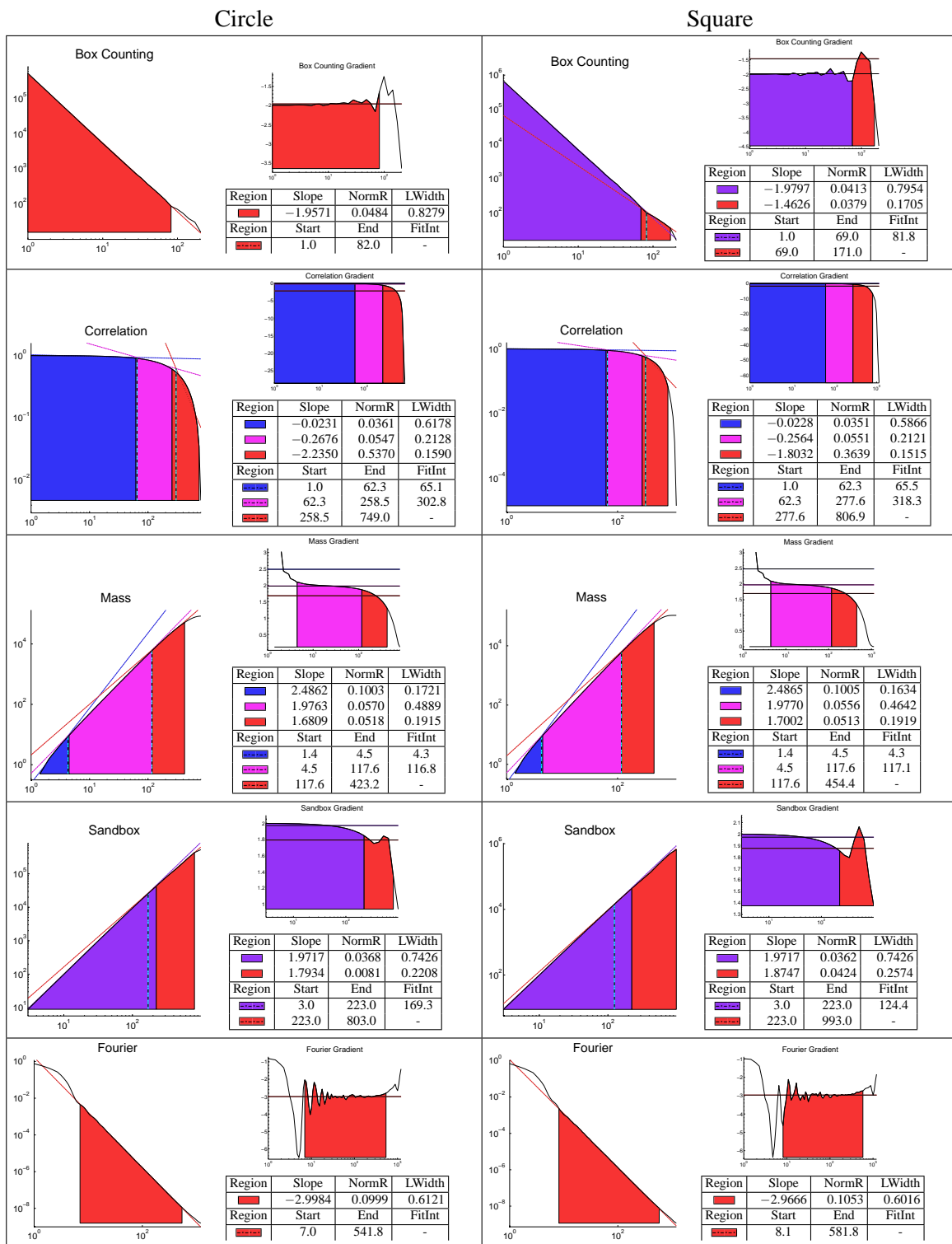


Figure A.6: Testing the algorithm on a filled circle and a filled square

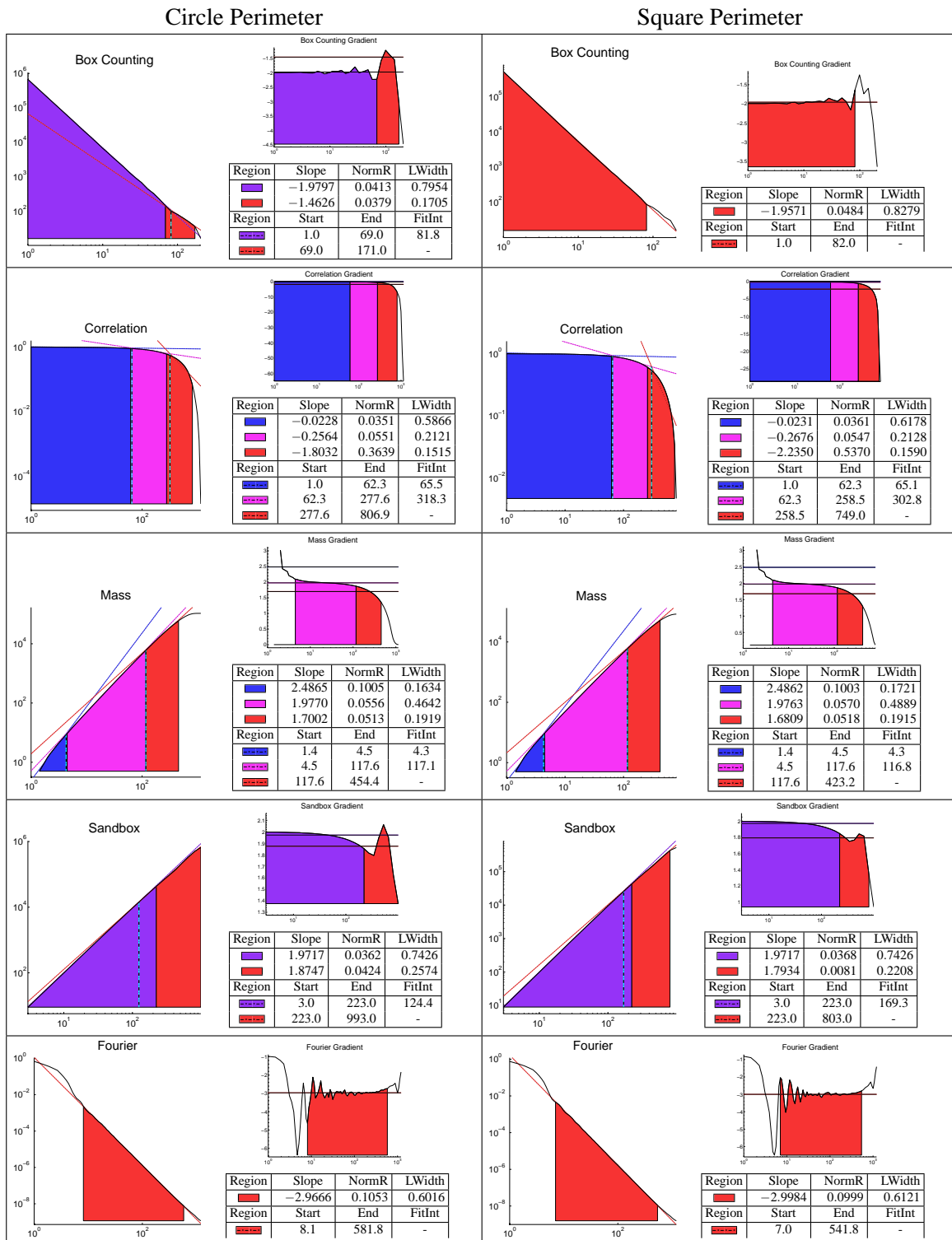


Figure A.7: Testing the algorithms on a circle and a square perimeter

Case 1

Case 2

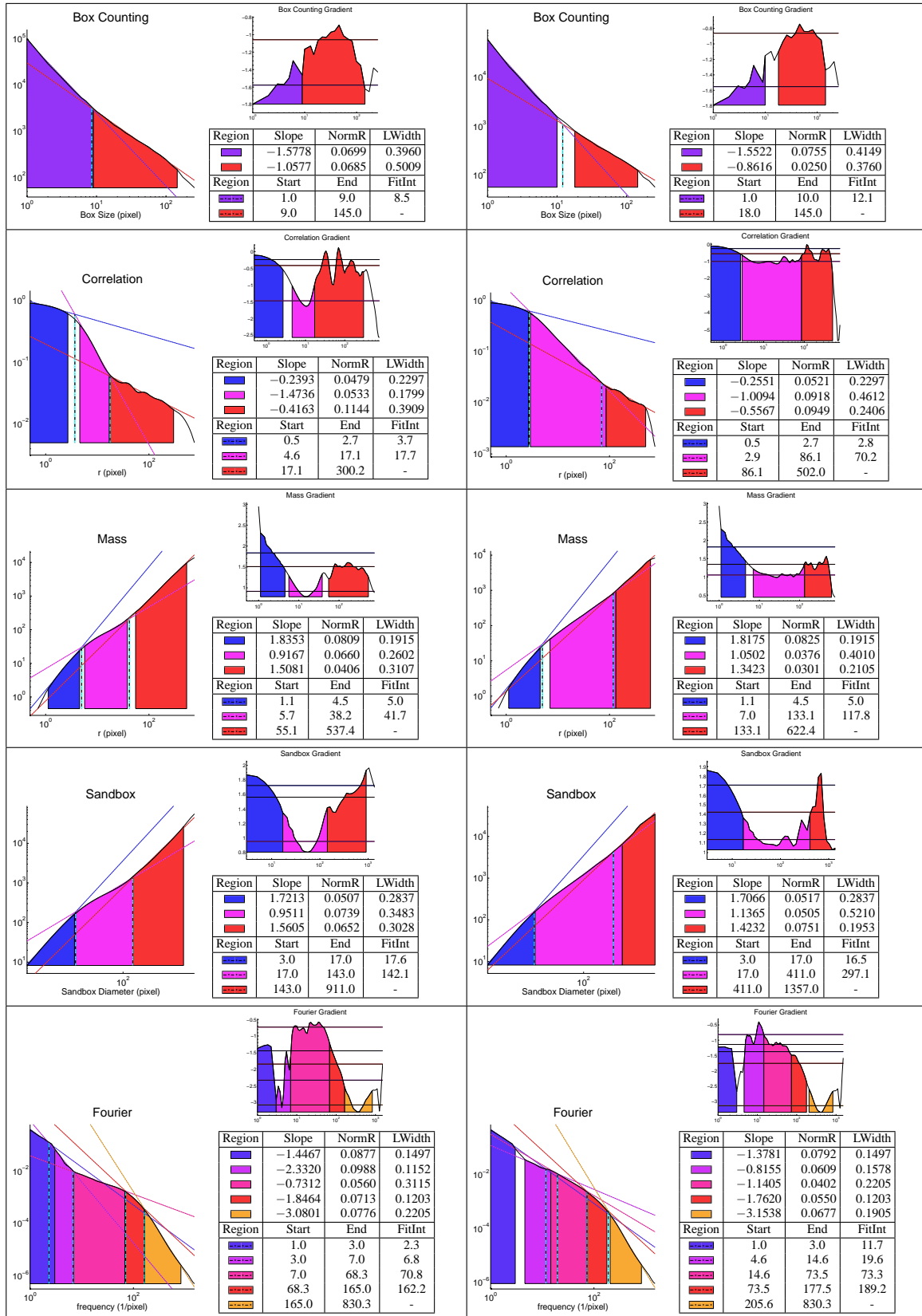


Figure A.8: Different analysis applied to the full cross section of cases 1 & 2

Case 3

Case 4

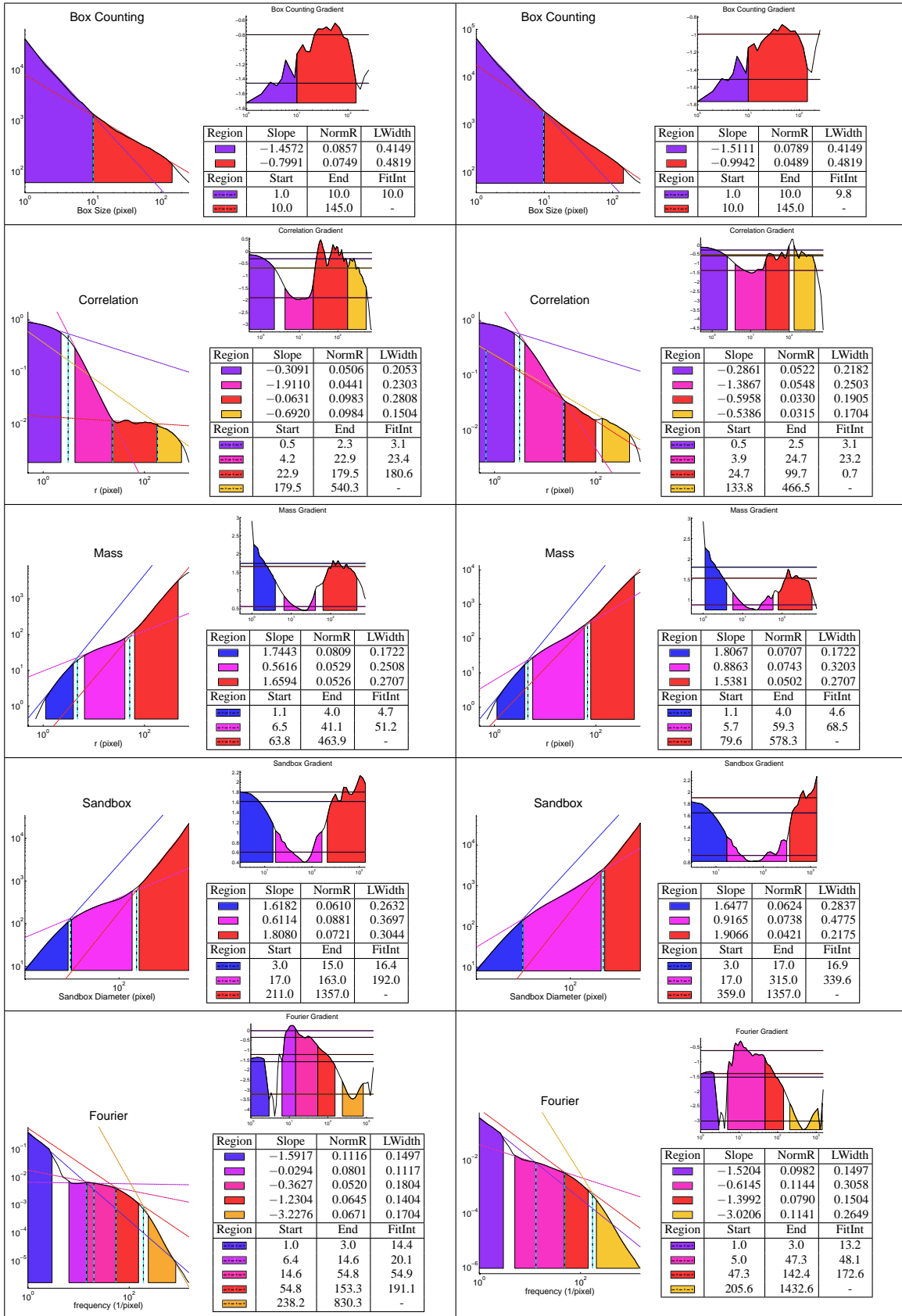


Figure A.9: Different analysis applied to the full cross section of cases 3 & 4

Case 1

Case 2

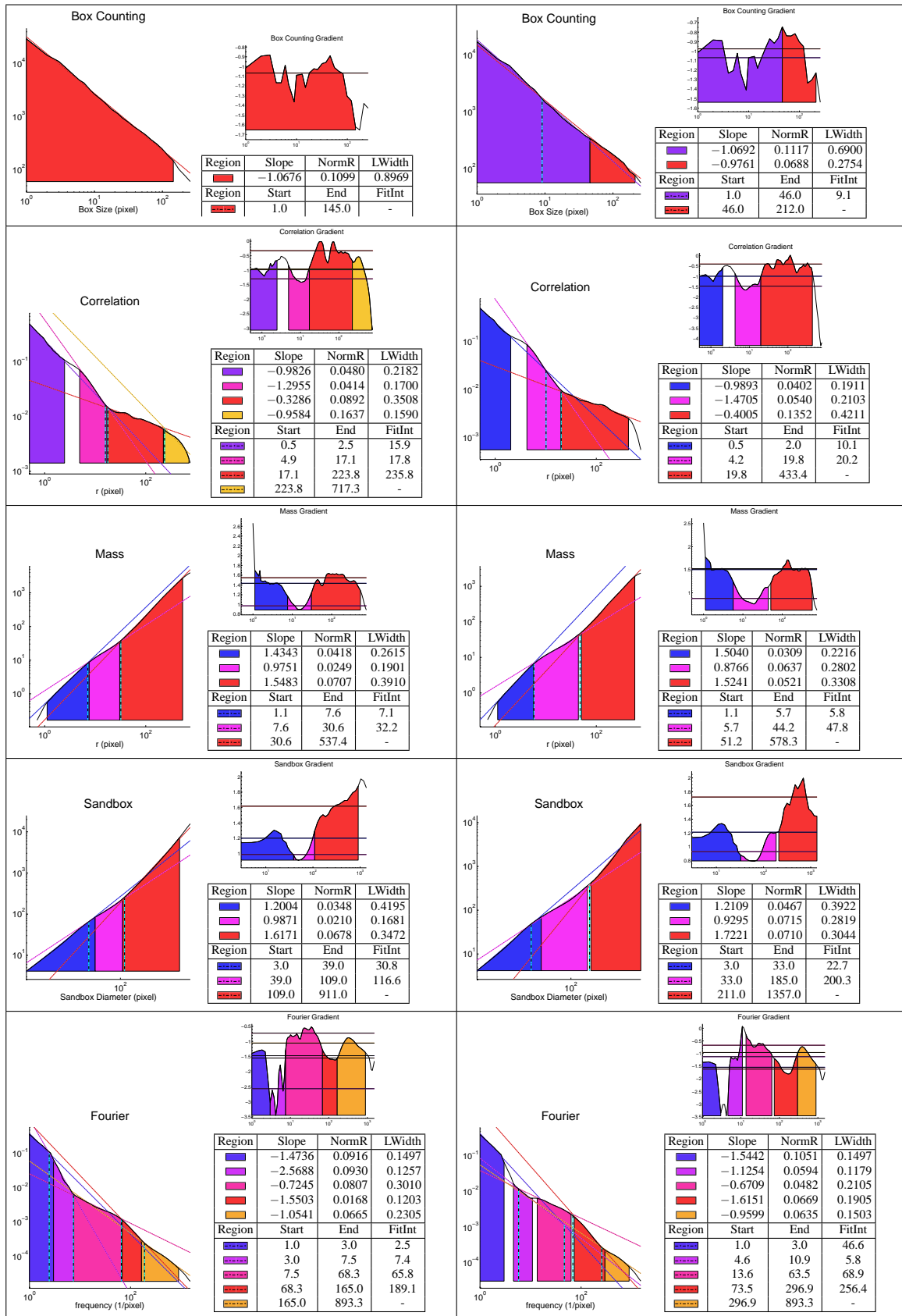


Figure A.10: Different analysis applied to the vessel perimeters of cases 1 & 2

Case 3

Case 4

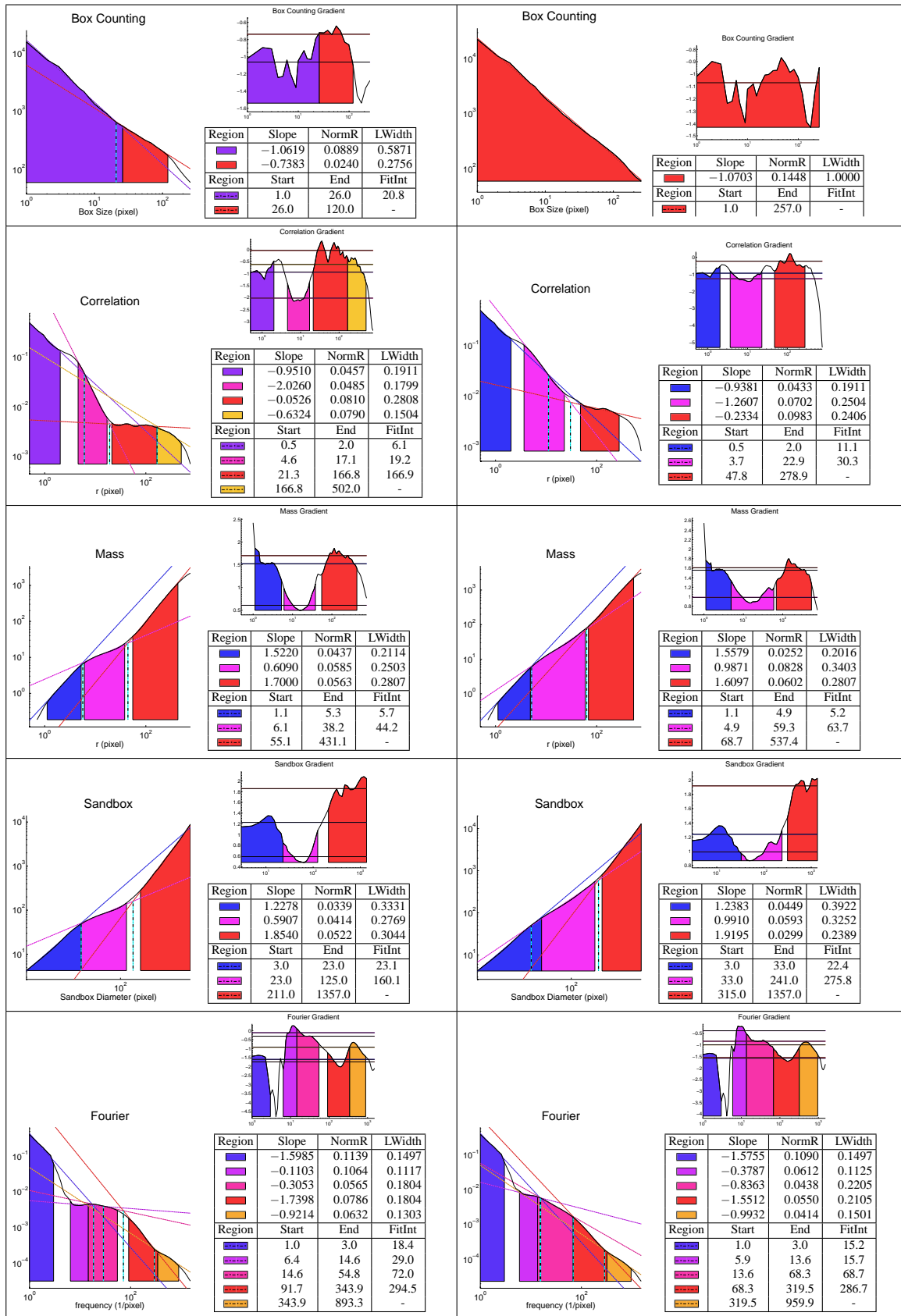


Figure A.11: Different analysis applied to the vessel perimeters of cases 3 & 4

Case 1

Case 2

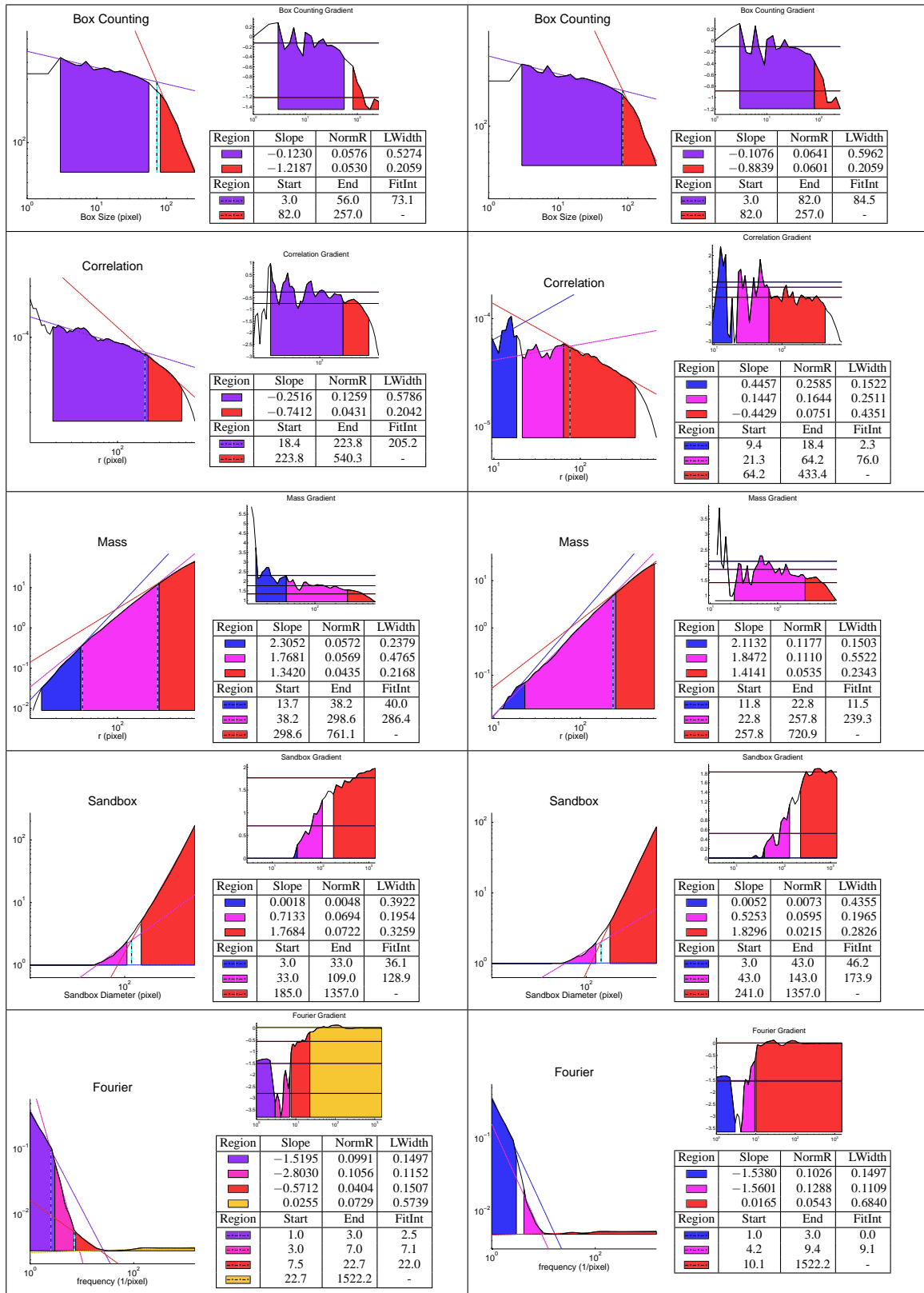


Figure A.12: Different analysis applied to the mass centers of cases 1 & 2

Case 3

Case 4

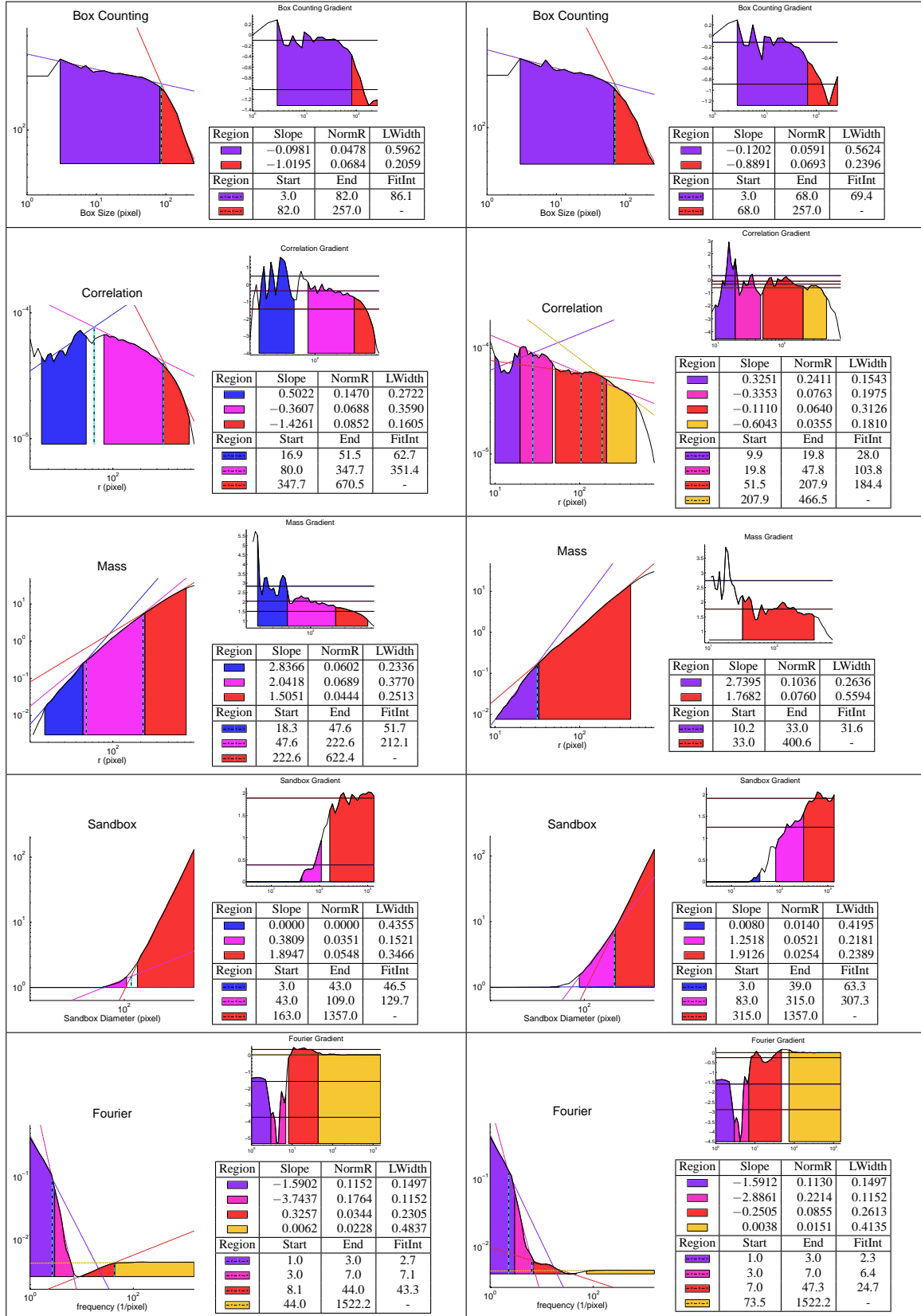


Figure A.13: Different analysis applied to the mass centres of cases 3 & 4

Case 1

Case 2

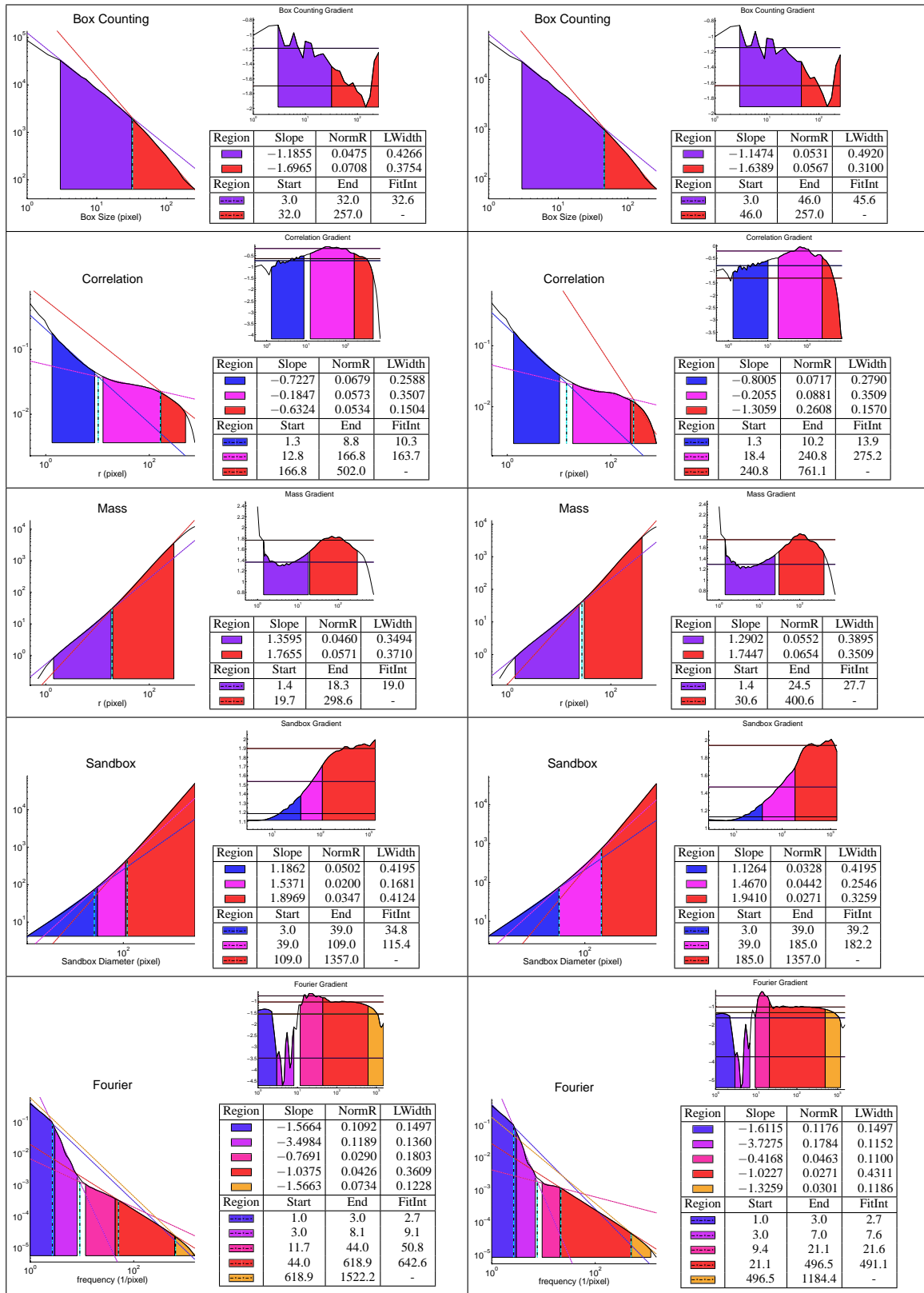


Figure A.14: Different analysis applied to the Gabriel's Graph of cases 1 & 2

Case 3

Case 4

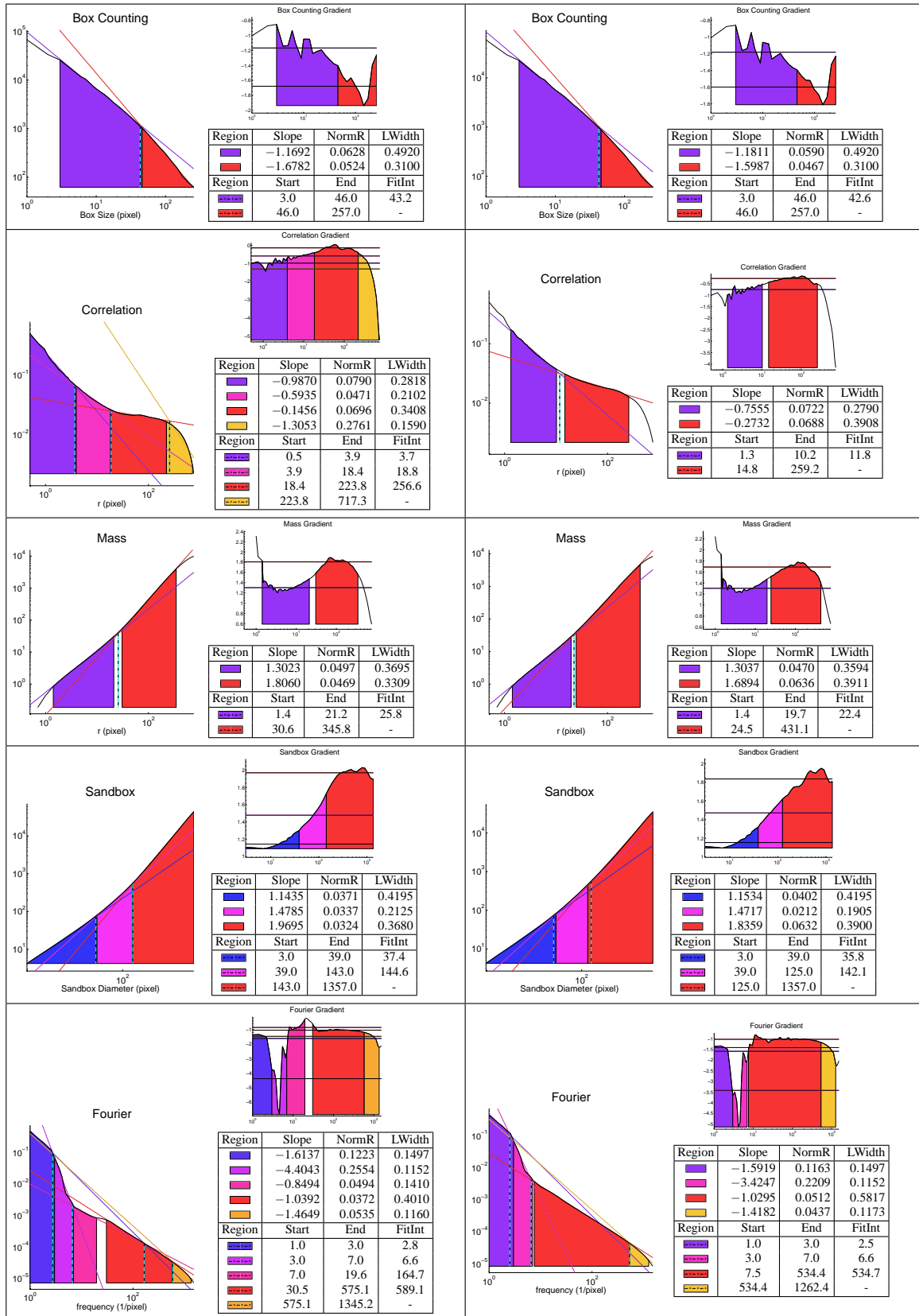


Figure A.15: Different analysis methods of Gabriel's Graph of cases 3 & 4

Case 1

Case 2

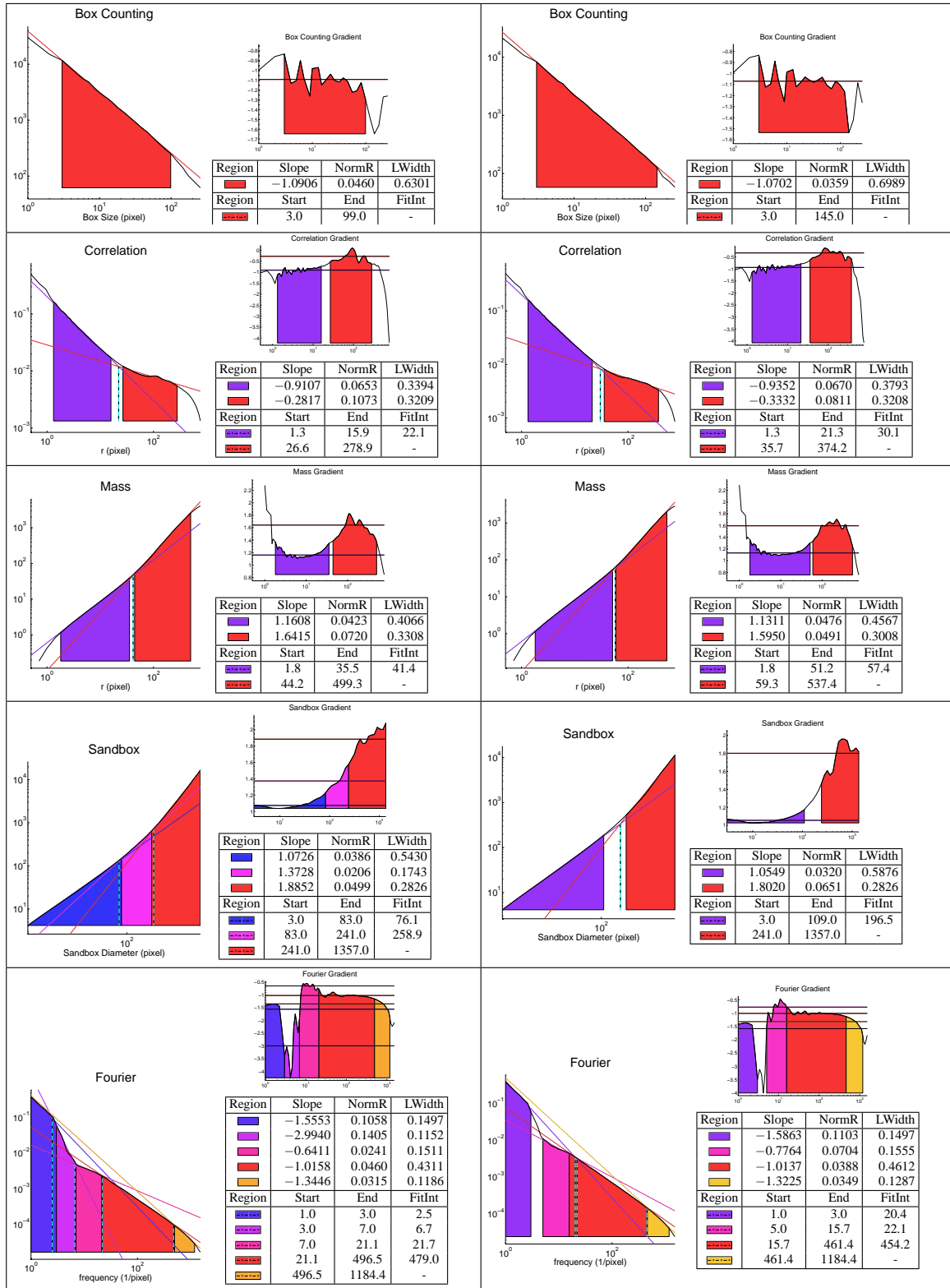


Figure A.16: Different analysis applied to the EMST of cases 1 & 2

Case 3

Case 4

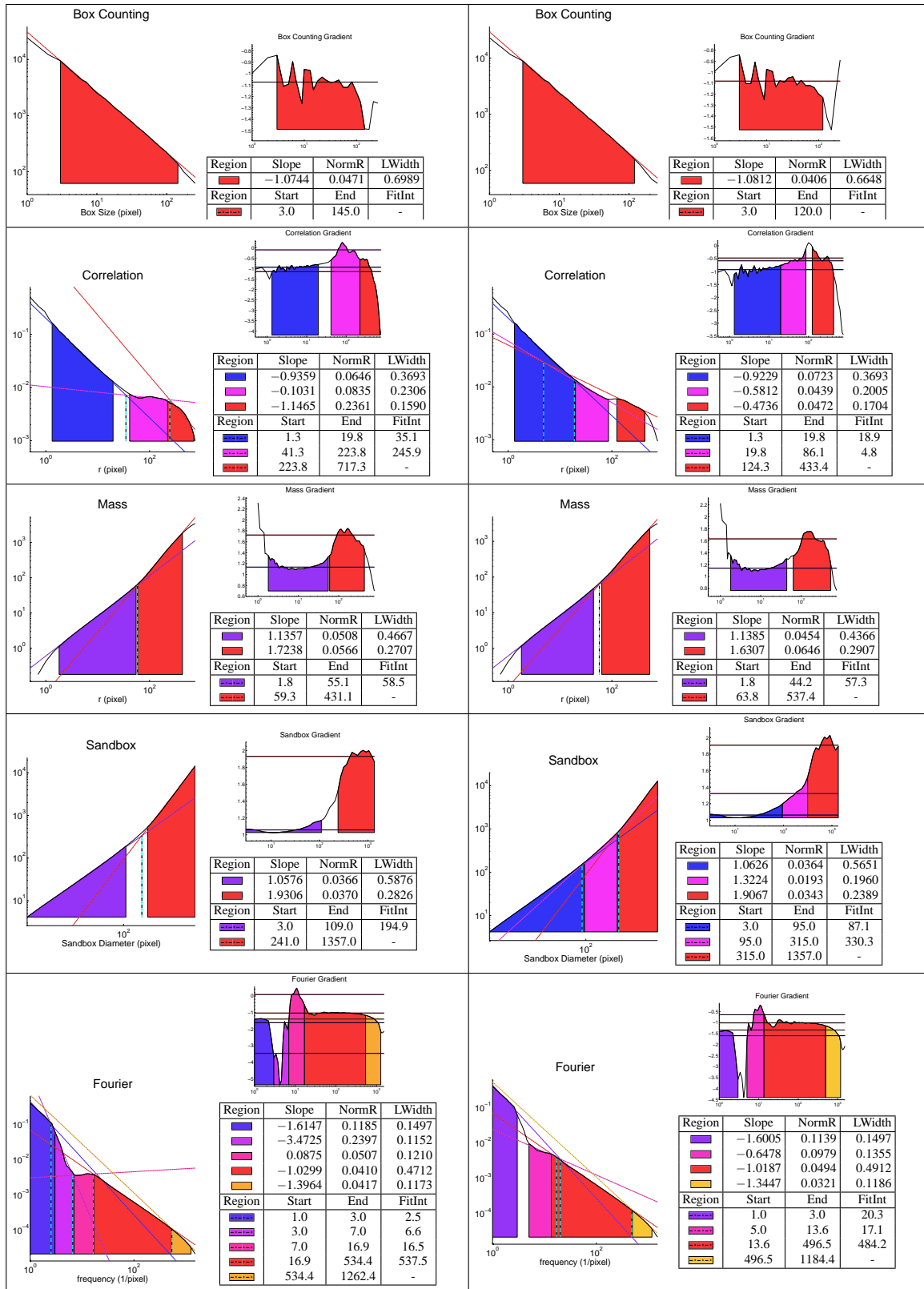


Figure A.17: Different analysis applied to the EMST of cases 3 & 4

Listings

B.1	Image Processing – Extract Vessels (function)	B-2
B.2	Image Processing – Image Layer Combine Function (function)	B-3
B.3	Image Processing – Produce Raster Image from Graph or Rescale Image (function)	B-5
B.4	Syntactic Structure Analysis – Queue Files for SSA (script)	B-6
B.5	Syntactic Structure Analysis – SSA-Analysis (script)	B-7
B.6	Syntactic Structure Analysis – Gabriels Graph (function)	B-8
B.7	Syntactic Structure Analysis – Euclidean Minimum Spanning Tree (function)	B-9
B.8	Syntactic Structure Analysis – Gabriels Graph and EMST statistics (function)	B-12
B.9	Syntactic Structure Analysis – Voronoi Statistics (function)	B-16
B.10	Syntactic Structure Analysis – Export Statistics to L ^A T _E X (function)	B-18
B.11	Random Site Percolation – Generate and Process Random Percolation Cluster (script)	B-19
B.12	Random Site Percolation – Largest Cluster in Matrix (function)	B-20
B.13	Random Site Percolation – Sort Cluster by Distance (function)	B-20
B.14	Random Site Percolation – Backbone of Cluster (function)	B-21
B.15	Random Site Percolation – Shortest Path Across Cluster (function)	B-23
B.16	Random Site Percolation – Direct Paths Across Cluster (function)	B-24
B.17	Random Site Percolation – Legal Neighbours (function)	B-25
B.18	Random Site Percolation – Lookup Table Identifying Bridging Pixels (function)	B-25
B.19	Random Site Percolation – Lookup Table Identifying Spur Pixels (function)	B-26
B.20	Fractal Analysis – Resize Image and Count Pixels for Box Counting (function)	B-27
B.21	Fractal Analysis – Count Number of Pixels for Sandbox Algorithm (function)	B-28
B.22	Fractal Analysis – Mass and Correlation Estimate (function)	B-29
B.23	Fractal Analysis – Fourier Estimate (function)	B-31
B.24	Fractal Analysis – Angle Average for Mass, Correlation and Fourier Estimate (function)	B-33
B.25	Fractal Analysis – Linear Fit of the Most Linear Region (function)	B-34
B.26	Fractal Analysis – Recursive Linear Fit (function)	B-36
B.27	Fractal Analysis – Plot Graphs and Data Table of Linear Regions (function)	B-37
B.28	Simulation – Random Simulation (script)	B-42
B.29	Simulation – 3D Bond Invasion Percolation (script)	B-45
B.30	Simulation – 3D Percolation Cross Section Processing (script)	B-47
B.31	Simulation – Sort Out Nested Cell Array Entries in the Saved Image Stacks (function)	B-49
B.32	Simulation – Merge Image Analysis Data Files of the Different Vessel Counts (script)	B-50
B.33	Simulation – Export Graphs of Percolation Simulation and Histological Data Points (script)	B-55

B Matlab Scripts and Functions

B.1 Image Processing

1.	Extract Vessels from RGB-image	B-2
2.	Image Combine Functions (floating point)	B-3

3. Prodcue Raster Image from Graph or Image of Different Resolution B-5

Function: extractvessels – Extract Vessels

```

1  function Ibw = extractvessels(A)
2  % Ibw = extractvessels(I)
3  % This function extracts the vessels out of CD34 images, by first
4  % extracting the green layer, then expanding the histogram by image overlay.
5  % A subtraction of an average filtered image is then used before
6  % thresholding this image for a BW-image based on luminance.
7  % The grayscale image is further processed by edge detection and
8  % thresholding to provide a BW-image of the edges in the image. These two
9  % BW images is then combined before the the lumens are filled in and the
10 % image cleaned up by removing areas smaller than 64 pixels.

12 A=single(A);

14 % RGB to grayscale, using green layer due to superior resolution
15 if size(A,3)==3
16     A=A(:,:,2);
17 end

19 I2=A;
20 %% IMAGECOMBINE
21 % Perform image overlays until at least 100 ppm of the image area is either
22 % above 90% luminance or belowe 10%
23 I = imcombinefp(I2,I2,'overlay',1);
24 i = 1;
25 while sum(sum(sum(I>=.90)))<.0001* numel(I) &&...
26     sum(sum(sum(I<=.10)))<.0001* numel(I)
27     i = i+1;
28     I = imcombinefp(I2,I,'overlay',1);
29 end
30 disp(['Number of overlays = ' num2str(i)]);

32 I=single(I);
33 %% AVERAGE FILTER
34 % average filter
35 ws=50; % width of averaging area
36 mIM=imfilter(I,fspecial('average',ws),'replicate');
37 % I equals the subtraction image
38 I=mIM-I;

40 % Calculating the threshold of the luminance image.
41 level = graythresh(I);
42 Ibwlum=im2bw(I,level);

44 %% EDGE DETECTION
45 [Isobel, thresh, gv, gh] = edge(I,'sobel');
46 grad = sqrt(gh.^2+gv.^2);
47 grad = grad-min(min(grad));
48 grad = grad./max(max(grad));

50 % thresholding edge image
51 level = graythresh(grad);
52 Ibw=im2bw(grad,level);

54 %% Keeping all pixels recognized by either luminance or edge thresholds
55 Ibw = Ibw | Ibwlum;

57 %% FIND ENTIRE LUMEN
58 % Merge areas very close to each other by morphological closings.
59 Ibw = bwmorph(Ibw, 'close',3);
60 % Close lumens by filling in holes
61 Ibw = imfill(Ibw,'holes');

```

```

62 % Erode off a one-pixel-thick layer at the perimeter to account for the
63 % edge function overestimates the width of the edges.
64 Ibw = bwmorph(Ibw,'erode');
65 % Remove all "vessels" smaller than 64 pixels at x25 magnification this is
66 % roughly translates into an area of (1.10E-6m)^2, i.e. false positives due
67 % to noise.
68 Ibw = bwareaopen(Ibw,64);

```

Listing B.1: extractvessels.m

Function: imcombinefp – Image Layer Combine (floating point)

```

1  function I = imcombinefp(varargin);
2  % IMAGE = imcombinefp(Foreground, Background, mode, n)
3  % Imcombinefp works with floating point images.
4  % These operations are based on the layer-modes of
5  % the Gnu-Image Manipulation Program – GIMP
6  %
7  % n > 1 applies the foreground multiple times, replacing the background with
8  % the result of last iteration.
9  %
10 % the mode can take any of the following values:
11 % -----
12 % SIMPLE FILTER    'addition'
13 %                  'subtract'
14 %                  'difference'
15 %                  'multiply'
16 %                  'divide'
17 %
18 % COMPOSITE FILTER 'dodge'
19 %                  'burn'
20 %                  'screen'
21 %                  'overlay'
22 %                  'hardlight'
23 %                  'softlight'
24 %                  'darkenonly'
25 %                  'lightenonly'
26 %                  'grainextract'
27 %                  'grainmerge'
28 %
29 % HSV FILTER       'hue'
30 %                  'colour'
31 %                  'saturation'
32 %                  'value'
33 %
34 % -----
35 %%% skipping unused image combine functions...
36 %
37 % OVERLAY
38 % B is inverted, multiplied by two times F, added to B, and then multiplied
39 % by B
40 %
41 %
42 %
43 % parse input arguments
44 switch 1
45     case numel(varargin) == 1 && isnumeric(varargin{1})
46         F=varargin{1};
47         B=F;
48         cmode='overlay';
49         n=1;
50     case numel(varargin) == 2 && isnumeric(varargin{1}) &&...
51         isnumeric(varargin{2})
52         F=varargin{1};

```



```

53     B=varargin{2};
54     cmode='overlay';
55     n=1;
56     case numel(varargin) == 3 && isnumeric(varargin{1}) &&...
57         isnumeric(varargin{2}) && ischar(varargin{3})
58         F=varargin{1};
59         B=varargin{2};
60         cmode=varargin{3};
61         n=1;
62     case numel(varargin) == 4 && isnumeric(varargin{1}) &&...
63         isnumeric(varargin{2})...
64         && ischar(varargin{3}) && isscalar(varargin{4})
65         F=varargin{1};
66         B=varargin{2};
67         cmode=varargin{3};
68         n=varargin{4};
69     otherwise
70         disp('ERROR in immode incorrect input arguments')
71         I=[];
72         return
73     end
74     cmode=lower(cmode);

76     % convert to single if uint8
77     if isequal(class(F),'uint8')
78         F = single(F);
79     elseif (isequal(class(F),'single') || isequal(class(F),'double')) &&...
80         max(max(max(F)))<=1
81         F = single(255.*F);
82     end
83     if isequal(class(B),'uint8')
84         B = single(B);
85     elseif (isequal(class(B),'single') || isequal(class(B),'double')) &&...
86         max(max(max(B)))<=1
87         B = single(255.*B);
88     end
89     F=single(F);
90     B=single(B);

92     %% IMAGE MODES %%
93     switch 1
94         %% skipping unused image combine functions

96         % OVERLAY
97         % B is inverted, multiplied by two times F, added to B, and then
98         % multiplied by B
99         case isequal(cmode,'overlay')
100             I = (B.*(B+(2.*F.*(255-B)) ./ 255) ./ 255);

102         otherwise
103             disp('mode not recognized')
104     end

106     if n > 1
107         n=n-1;
108         I = imcombine(F,I,cmode,n);
109     end

111     I= single(I ./255);

```

Listing B.2: imcombinefp.m

Function: rasterim – Produce Image Matrices from Graphs or Change the Resolution of Black and White Images

```

1  function A = rasterim(xa,xb,ya,yb,siz)
2  % A = rasterim(xa,xb,ya,yb,siz)
3  % rasterim produces an image matrix of size siz, where the pixels entered
4  % or exited by the vectors [(xa,ya),(xb,yb)] are 1, and the rest 0.
5  % This is used to produce raster images of Gabriel's Graph and the
6  % Euclidean Minimum Spanning Tree for Fractal Analysis.
7  % The function can also be used to change the resolution in an image to siz
8  % by providing the row and column indices: rasterim(C,C,R,R,siz)

10 A = false(siz);
11 xmax = max(max([xa(:);xb(:)]));
12 ymax = max(max([ya(:);yb(:)]));
13 xa = double(xa)./xmax;
14 xb = double(xb)./xmax;
15 ya = double(ya)./ymax;
16 yb = double(yb)./ymax;

18 for i = 1:numel(xa)
19     x1 = round(xa(i)*siz(2));
20     x2 = round(xb(i)*siz(2));
21     y1 = round(ya(i)*siz(1));
22     y2 = round(yb(i)*siz(1));
23     fromx = floor(min(x1,x2));
24     tox = ceil(max(x1,x2));
25     fromy = floor(min(y1,y2));
26     toy = ceil(max(y1,y2));
27     if (x2-x1)==0
28         a = inf;
29     else
30         a = (y2-y1)/(x2-x1);
31     end
32     b = y1-a*x1;
33     % assigning start and stop pixels
34     A(y1,x1) = 1;
35     A(y2,x2) = 1;

37     %% checking intersects with y-gridlines
38     ygrid = fromy+.5:toy-.5;
39     if isfinite(a)
40         x = (ygrid-b)/a;
41     else
42         x = x1*ones(size(ygrid));
43         b = 0;
44     end
45     % keeping pixels visited in two diagonal corners
46     l = ((x+.5)-fix(x+.5)<1e-10);
47     lc = l(1:numel(l)-1) & l(2:numel(l));
48     xc = x(lc)+.5;
49     yc = ygrid(lc)+.5*sign(a);
50     % removing pixels only visited in one corner (pixels are never visited in
51     % two horizontal or vertically aligned corners, because all points start in
52     % integer positions.)
53     x(1) = [];
54     ygrid(1) = [];
55     ym = [ygrid-.5,ygrid+.5];
56     ym = [ym,yc];
57     xm = round([x,x,xc]);
58     %     xm = [xm,xc]

60     %% checking intersects with x-gridlines
61     xgrid = fromx+.5:tox-.5;
62     y = a*(xgrid)+b;
63     % keeping pixels visited in two diagonal corners

```

```

64     l = ((y+.5)-fix(y+.5)<1e-10);
65     lc = l(1: numel(l)-1) & l(2: numel(l));
66     yc = y(lc)+.5*sign(a);
67     xc = xgrid(lc)+.5;
68     % removing pixels only visited in one corner
69     y(1) = [];
70     xgrid(1) = [];
71     xn = [xgrid-.5,xgrid+.5,xc];
72     yn=round([y,y,yc]);

74     % assign new pixel values
75     I = sub2ind(size(A),[ym,yn],[xm,xn]);
76     A(I) = 1;
77 end

79 A=rot90(rot90(A));

```

Listing B.3: rasterim.m

B.2 Syntactic Structure Analysis

1. Script for queueing SSA-analysis of multiple files B-6
2. Script running SSA-analysis on a single file B-7
3. Gabriels Graph B-8
4. Euclidean Minimum Spanning Tree B-9
5. GG and EMST Statistics B-12
6. Voronoi Statistics B-16
7. Statistics to \LaTeX writer B-18

Script: runallSSA.m – Script Queueing SSA-Analysis

```

1  clear, close all
2  % Script used to queue different files for syntactic structure analysis
3  % DEPENDENCIES: scriptGetGraphs.m, ggstats.m, voronoistat.m, stattex.m

5  % gg = 3-7
6  fidtotal = fopen('../analyse/images/prelim/total.tex','w');
7  fidbl = fopen('../analyse/images/prelim/branchlength.tex','w');
8  fidbpn = fopen('../analyse/images/prelim/branchpernode.tex','w');
9  fidnn = fopen('../analyse/images/prelim/distnn.tex','w');
10 fidfn = fopen('../analyse/images/prelim/distfn.tex','w');
11 % emst = 8-12
12 fidemsttotal = fopen('../analyse/images/prelim/emsttotal.tex','w');
13 fidemstbl = fopen('../analyse/images/prelim/emstbranchlength.tex','w');
14 fidemstbpn = fopen('../analyse/images/prelim/emstbranchpernode.tex','w');
15 fidemstnn = fopen('../analyse/images/prelim/emstdistnn.tex','w');
16 fidemstfn = fopen('../analyse/images/prelim/emstdistfn.tex','w');
17 % voronoi = 13-15
18 fidarea = fopen('../analyse/images/prelim/vorarea.tex','w');
19 fidarea = fopen('../analyse/images/prelim/vorshape.tex','w');
20 fidarea = fopen('../analyse/images/prelim/vorform.tex','w');

22 casestr = 'caselx25';
23 scriptGetGraphs

25 casestr = 'case2x25';
26 scriptGetGraphs;

```

```

28 casestr = 'case3x25';
29 scriptGetGraphs;

31 casestr = 'case4x25';
32 scriptGetGraphs;

34 fclose('all');
35 close all

```

Listing B.4: runallSSA.m

Script: scriptGetGraphs.m – Script Running SSA-Analysis on Image

```

1  % Script for finding the voronoi diagram, gabriels graph and the euclidean
2  % minimum spanning tree. Note the files output is written to must be opened
3  % in another script, furthermore casestr must be specified in another
4  % script.
5  %
6  % This script is used together with the runallSSA script.
7  % Other dependencies: gabrielsgraph, emst, ggstats, voronoistat.

9  titlefont = 26;
10 axisfont = 18;
11 IM = imread(strcat('../analyse/images/', casestr, 'bw.tif'));
12 IM = single(IM);
13 IM = IM./max(max(IM)); % transform possible input bw from (0,255) to (0,1)

15 K2=IM;
16 %% preprocessing image
17 % removing any holes and reducing areas to the center of mass.
18 K2 = imfill(K2, 'holes');
19 K2 = bwmorph(K2, 'shrink', inf);

21 %% locating all vessels
22 [R,C] = find(K2');
23 xmin=min(R); xmax = max(R); ymin=min(C); ymax=max(C);
24 x=[R,C];

26 %% Voronoi
27 [v,c]=voronoin(x);
28 % making random colordistribution for patches
29 [r,I]=sort(rand(1,length(c)));
30 cdist=I; % colordistribution
31 figure(1), clf, hold on,
32 % plotting all polygons that have all corners inside the image to avoid
33 % edge
34 isinside = false(size(c));
35 for i = 1:length(c)
36     %% only use polygons inside image
37     if all(c{i}~=1) && all(v(c{i},1)>=xmin) && all(v(c{i},1)<=xmax) && ...
38         all(v(c{i},2)>=ymin) && all(v(c{i},2)<=ymax)
39         isinside(i)=1;
40         % plot polygon
41         patch(v(c{i},1),v(c{i},2),cdist(i)); % use random color cdist(i).
42     end
43 end
44 c=c(isinside);
45 % format and export figure
46 axis([1 size(IM,2) 1 size(IM,1)])
47 axis on; axis tight;
48 axis([1 size(IM,2) 1 size(IM,1)])
49 set(gca, 'ydir', 'reverse', 'FontSize', axisfont);
50 title('Voronoi Diagram', 'fontsize', titlefont);
51 hold off

```

```

52 hgexport(gcf, strcat('../analyse/images/',casestr,...
53     '/',casestr,'voronoi.eps'))

55 %% Delauney triangulation
56 TRI = delaunay(x(:,1),x(:,2));

58 %% Gabriels graph
59 xind = x(:,1); yind=x(:,2);
60 [x1,x2,y1,y2]=gabrielsgraph(TRI,xind,yind);

62 figure(2), clf,
63 hold on,
64 for i=1:numel(x1)
65     plot([x1(i),x2(i)],[y1(i),y2(i)],'k'),
66 end
67 hold off
68 % format and export figure
69 daspect([1 1 1]),
70 set(gca,'ydir','reverse','FontSize',axisfont);
71 axis on; axis tight
72 axis([1 size(IM,2) 1 size(IM,1)])
73 title('Gabriel's Graph','fontsize',titlefont)
74 hgexport(gcf, strcat('../analyse/images/',casestr,...
75     '/',casestr,'gabrielsgraph.eps'))

77 %% Euclidean Minimum Spanning Tree
78 figure(8), clf,
79 [xa,ya,xb,yb] = emst(x1,y1,x2,y2);
80 hold on,
81 for i=1:numel(xa)
82     plot([xa(i),xb(i)],[ya(i),yb(i)],'k'),
83 end
84 hold off
85 % format and export figure
86 axis tight, axis on, daspect([1 1 1]),
87 axis([1 size(IM,2) 1 size(IM,1)])
88 set(gca,'ydir','reverse','FontSize',axisfont),
89 title('Euclidean Minimum Spanning Tree','fontsize',titlefont-5);
90 hgexport(gcf, strcat('../analyse/images/',casestr,'/',casestr,'emst.eps'))

92 ggstats(x1,x2,y1,y2,casestr,'gg')
93 ggstats(xa,xb,ya,yb,casestr,'emst')
94 voronoistat(v,c,casestr)
95 %close all figures
96 close all

```

Listing B.5: scriptGetGraphs.m

Function: gabrielsgraph.m – Gabriels Graph

```

1 function [x1,x2,y1,y2] = gabrielsgraph(TRI,x,y)
2 % [x1,x2,y1,y2] = gabrielsgraph(TRI,x_indices,y_indices)
3 % TRI is the Delauney triangulation.
4 % The function calculates Gabriels Graph from the delauney
5 % triangulation.
6 %
7 % Gabriels Graph is determined by keeping all branches between nodes
8 % that fullfill the following criterium:
9 % Each branch between two nodes uniquely deterimnes a circle with this
10 % branch as the diameter. If there are no other nodes inside this circle,
11 % then the branch is a part of Gabriels Graph, otherwise it is not.

13 %% turn the list of triangles into a list of braches/edges
14 S=cat(1,TRI(:,1:2),TRI(:,[1,3]),TRI(:,[2,3]));

```

```

15 S=unique(S,'rows');
16 x1=x(S(:,1)); x2=x(S(:,2));
17 y1=y(S(:,1)); y2=y(S(:,2));

19 %% testing all branches for the gabriels graph criterium
20 l=zeros(length(x1),1);
21 for i=1:numel(x1),
22     dx=abs(x1(i)-x2(i));
23     dy=abs(y1(i)-y2(i));
24     r = sqrt(dx^2+dy^2)./2;
25     xctr = min(x1(i),x2(i)) + dx/2;
26     yctr = min(y1(i),y2(i)) + dy/2;

28     %% check the nodes that has a branch to (x1,y1) or (x2,y2);
29     % alle punkter som går til eller fra
30     Ifrom = find( ((x1==x1(i)) & (y1==y1(i)))) );
31     Ito = find( ((x2==x2(i)) & (y2==y2(i)))) );

33     I=cat(1,Ifrom,Ito);
34     XN= unique( [[x1(I), y1(I)];[x2(I),y2(I)]],'rows');

36     %% checking the distance to the centre of the circlce
37     distToCtr=sqrt((xctr-XN(:,1)).^2 + (yctr-XN(:,2)).^2);
38     if all( distToCtr >= r),
39         l(i)=1;

41     else
42         l(i)=0;

44     end
45 end

47 %% removing all branches that does not meet the criterium.
48 x1=nonzeros(x1.*l);
49 x2=nonzeros(x2.*l);
50 y1=nonzeros(y1.*l);
51 y2=nonzeros(y2.*l);

```

Listing B.6: gabrielsgraph.m

Function: emst.m – Euclidean Minimum Spanning Tree

```

1 function [x1,y1,x2,y2] = emst(x1,y1,x2,y2)
2 % Funtion [x1,y1,x2,y2] = EMST(x1,y1,x2,y2)
3 %
4 % Euclidean Minimum Spanning Tree
5 % The Tree is euclidean because the weights on each branch is specified by
6 % the euclidean distance between the endpoints. The minimum spanning tree
7 % is a tree reaching all points in a graph in such a way that the sum of
8 % the weights of all individual branches is minimized.
9 %
10 % (x1,y1),(x2,y2) specify the two endpoints of the lines that make up
11 % the graph and the EMST.

13 siz=[max([y1;y2]),max([x1;x2])];

15 %% Make list of all nodes [Nodeindice, {neighbourindices},
16 %% {distanceToNeighbours}]
17 A = cat(2,sub2ind(siz,y1,x1),sub2ind(siz,y2,x2)); % all arcs
18 N = unique([y1,x1;y2,x2],'rows'); % all nodes
19 N = sub2ind(siz,N(:,1),N(:,2));
20 Neigh = cell(size(N));
21 Dist=cell(size(N));

```

```

23 % find neighbours
24 for i=1:length(N),
25     Ifrom = find( A(:,1)==N(i) );
26     Ito = find(A(:,2)==N(i));

28     neigh = [A(Ito,1);A(Ifrom,2)];
29     [neigh] = unique(neigh);
30     [m1,n1] = ind2sub(siz,N(i));
31     [m2,n2] = ind2sub(siz,neigh);
32     dist = sqrt( (m2-m1).^2 + (n2-n1).^2); % euclidean norm

34     Neigh{i}=neigh;
35     Dist{i}=dist;
36 end

38 %% Initializing network variables.
39 first = 1; % indice of the first node added to network
40 Tree = N(first);%l % The current euclidean minnum spanning tree
41 AddedTo = N(first);%l
42 Fringe = [];
43 FringeDist = [];
44 FringeConnTo = [];
45 ConnTo = cell(1,numel(N));
46 % weight to the neighbour node that connects this node to the tree
47 ConnToWeight = cell(1,numel(N));
48 lastadded = first; % starting with first node

50 %% Specifying the first arc between the startnode and the closest neighbour
51 neigh = Neigh{lastadded};
52 dist = Dist{lastadded};

54 Fringe = [Fringe;neigh];
55 FringeDist = [FringeDist;dist];
56 FringeConnTo = [FringeConnTo; ones(numel(dist),1)*lastadded];

58 [val,ind] = min(FringeDist);

60 Tree = [Tree;Fringe(ind)];
61 addedto = FringeConnTo(ind);
62 AddedTo = [AddedTo;Tree(1)];
63 lastadded = find(N==Fringe(ind));

65 ConnTo{first} = lastadded;
66 ConnToWeight{first} = FringeDist(ind);

68 ConnTo{lastadded} = first;
69 ConnToWeight{lastadded} = FringeDist(ind);

71 Fringe(ind) = []; % deleting row
72 FringeDist(ind) = []; % deleting row
73 FringeConnTo(ind) = [];% deleting row

75 %% Iterating the rest of the network
76 for i=1:length(N)-1,
77     % updating fringe
78     neigh = Neigh{lastadded};
79     dist = Dist{lastadded};

81 %% Checking if the new neighbours are in Fringe already.
82 % if any neigh already is in Fringe it should be kept with the smallest
83 % of the two weights, rather than added again
84 removeind=[];
85 for j=1:numel(neigh)
86     I = find(Fringe==neigh(j)); % never more than one indice, I is a scalar
87     if ~isempty(I)
88         removeind = [removeind; j];

```

```

89         if FringeDist(I)>dist(j)
90             FringeDist(I) = dist(j);
91             FringeConnTo(I) = lastadded;
92         end
93     end
94 end

96     neigh(removeind) = [];
97     dist(removeind) = [];

99     %% Checking if any new neighbours are in the network already.
100    % if any neigh already is in Tree it should be removed from the list
101    removeind=[];
102    for j=1:numel(neigh)
103        I=find(Tree==neigh(j));
104        if ~isempty(I)
105            removeind=[removeind; j];
106        end
107    end
108    neigh(removeind)=[];
109    dist(removeind)=[];

111    %% Adding neighbours to Fringe
112    Fringe = [Fringe;neigh];
113    FringeDist = [FringeDist;dist];
114    FringeConnTo = [FringeConnTo; ones(numel(dist),1)*lastadded];

116    % if Fringe is empty, all done
117    if isempty(Fringe)
118        break; % stop loop
119    end

121    % find the fringe node with smallest weight(s) and add to tree.
122    [val,ind] = min(FringeDist);
123    lastadded = find(N==Fringe(ind));
124    addedto = FringeConnTo(ind);
125    Tree = [Tree;Fringe(ind)];
126    AddedTo = [AddedTo;N(addedto)];

128    %% Adding the new connection to the node
129    % if this is not the first connection then earlier connections must be
130    % kept.
131    if isempty(ConnTo{addedto})
132        ConnTo{addedto} = lastadded;
133        ConnToWeight{addedto} = FringeDist(ind);
134    else
135        ConnTo{addedto} = [ConnTo{addedto};lastadded];
136        ConnToWeight{addedto} = [ConnToWeight{addedto};FringeDist(ind)];
137    end

139    %% Removing added fringe
140    Fringe(ind) = []; % deleting row
141    FringeDist(ind) = []; % deleting row
142    FringeConnTo(ind) = [];% deleting row
143    end

145    %% Calculating coordinates of start and end of all branches
146    xto=[]; yto=[]; xfrom=[]; yfrom=[]; %
147    n=0;
148    for i = 1:length(ConnTo)
149        if ~isempty(ConnTo{i})
150            connto = ConnTo{i};
151            for j=1:length(connto);
152                n=n+1;
153                [yfrom(n),xfrom(n)] = ind2sub(siz , N(connto(j)));
154                [yto(n),xto(n)] = ind2sub(siz , N(i));

```



```

155         end
156     end
157 end
159 %% Parsing output
160 %% Removing Multiple Vertice Added at the Connection of the First Node
161 x1=xfrom(:);
162 x2=xto(:);
163 y1=yfrom(:);
164 y2=yto(:);

```

Listing B.7: emst.m

Function: ggstat.m – Gabriels Graph and EMST Statistics

```

1  function ggstats(varargin)
2  % ggstats(x1,x2,y1,y2,casestr,graphstr,[texsortwrite])
3  % Calculates gabriels graph or euclidean minimum spanning tree statistics.
4  % x1,y1 and x2,y2 are the start and end coordinates respectively of all
5  % branches in the network. casestr and graphstr are used to specify
6  % filenames for the unsorted writings to file.
7  %
8  % The default value for writing sorted to tex is 1.
9  % Writing sorted to tex assumes that the output fids are numbered from
10 % 3–7 in the case of gabriels graph
11 % 8–12 in the case of euclidean minimum spanning tree
12 % see runallSSA...
14 titlefont = 26;
15 axisfont = 18;
17 x1      = varargin{1};
18 x2      = varargin{2};
19 y1      = varargin{1};
20 y2      = varargin{2};
21 casestr = varargin{5};
22 graphstr= varargin{6};
23 texsortwrite=1;
24 if numel(varargin)>6
25     texsortwrite = varargin{7};
26 end
28 siz(1) = max(max([x1;x2]));
29 siz(2) = max(max([y1;y2]));
31 %% Branch Length Parameters
32 numbranch      = length(x1);
33 branchLengths  = sqrt((x2-x1).^2 + (y2-y1).^2);
34 totalLength     = sum(branchLengths);
35 meanBranchLength = mean(branchLengths);
36 stdBranchLength = std(branchLengths);
37 skewnessBranchLength= skewness(branchLengths);
38 kurtosisBranchLength= kurtosis(branchLengths);
40 figure(14), hist(branchLengths,50),
41 title('Branch Length','fontSize',... titlefont)
42 set(gca, 'FontSize', axisfont)
43 if strcmp(graphstr, 'emst')
44     set(gca, 'XLim', [0,300], 'YLim', [0,25])
45 else
46     set(gca, 'XLim', [0,500], 'YLim', [0,60])
47 end
48 hgexport(gcf, strcat('../analyse/images/', casestr, ...
49     '/', casestr, 'hist', graphstr, 'br1.eps'))

```

```

51 %% Branches/Node Parameters
52 % number of nodes
53 nodes = unique([x1,y1;x2,y2], 'rows');
54 numnodes = size(nodes,1);
55 branches = sortrows([x1 y1; x2 y2],1);

57 nodeind = sub2ind(siz , nodes(:,1), nodes(:,2));
58 branchind = sub2ind(siz , branches(:,1), branches(:,2));

60 % numbranchespernode — numbpn
61 numbpn = zeros(size(nodes,1),1);
62 for i=1:numel(nodeind)
63     numbpn(i) = sum(sum(nodeind(i) == branchind));
64 end
65 meanbpn = mean(numbpn);
66 stdbpn = std(numbpn);
67 skewnessbpn = skewness(numbpn);
68 kurtosisbpn = kurtosis(numbpn);

70 figure(11), bar(histc(numbpn,1:max(max(numbpn))),
71 title('Branches Per Node','fontsize',titlefont)
72 set(gca,'FontSize',axisfont,'XLim',[0,10.5])
73 if strcmp(graphstr,'emst')
74     set(gca,'YLim',[0,200])
75 else
76     set(gca,'YLim',[0,150])
77 end
78 hgexport(gcf, strcat('./analyse/images/',casestr,...
79     '/','casestr','hist',graphstr,'bpn.eps'))

82 %% Distance to nearest and furthtest neighbours
83 % nodennd — node nearest neighbour distance
84 nodennd = inf.*ones(size(nodes,1),1);
85 % nodefnd — node furthest neighbour distance
86 nodefnd = zeros(size(nodes,1),1);
87 for i=1:numel(x1)
88     nodeA = sub2ind(siz ,x1(i),y1(i));
89     nodeB = sub2ind(siz ,x2(i),y2(i));
90     indA = find(nodeind == nodeA);
91     indB = find(nodeind == nodeB);
92     if branchLengths(i) < nodennd(indA)
93         nodennd(indA) = branchLengths(i);
94     end
95     if branchLengths(i) < nodennd(indB)
96         nodennd(indB) = branchLengths(i);
97     end
98     if branchLengths(i) > nodefnd(indA)
99         nodefnd(indA) = branchLengths(i);
100    end
101    if branchLengths(i) > nodefnd(indB)
102        nodefnd(indB) = branchLengths(i);
103    end
104 end

106 % nearest neighbour
107 meannn = mean(nodennd);
108 stdnn = std(nodennd);
109 skewnessnn = skewness(nodennd);
110 kurtosisnn = kurtosis(nodennd);

112 % furthest neighbour
113 meanfn = mean(nodefnd);
114 stdfn = std(nodefnd);
115 skewnessfn = skewness(nodefnd);

```

```

116 kurtosisfn = kurtosis(nodefn);
118 figure(13), hist(nodennd, 30),
119 title('Distance to Nearest Neighbour','fontsize',titlefont)
120 set(gca, 'FontSize', axisfont)
121 if strcmp(graphstr, 'emst')
122     set(gca, 'XLim', [0,300], 'YLim', [0 50])
123 else
124     set(gca, 'XLim', [0,200], 'YLim', [0 50])
125 end
126 hgexport(gcf, strcat('../analyse/images/', casestr, ...
127     '/', casestr, 'hist', graphstr, 'dnn.eps'))

129 figure(16), hist(nodefn, 50),
130 title('Distance to Furthest Neighbour','fontsize',titlefont)
131 set(gca, 'FontSize', axisfont)
132 if strcmp(graphstr, 'emst')
133     set(gca, 'XLim', [0,300], 'YLim', [0 30])
134 else
135     set(gca, 'XLim', [0,500], 'YLim', [0 40])
136 end
137 hgexport(gcf, strcat('../analyse/images/', casestr, '/' ,...
138     casestr, 'hist', graphstr, 'dfn.eps'))

140 filewrite=1;
141 if filewrite
142     % casestr = 'case3x25';
143     fid = fopen(strcat('../analyse/images/', casestr, '/', casestr, ...
144         graphstr, 'stat.txt'), 'w');
145     fprintf(fid, 'Number of nodes: \t\t\t\t %d \n', numnodes)
146     fprintf(fid, 'Number of branches: \t\t\t\t %d \n', numbranch);
147     fprintf(fid, 'Total length: \t\t\t\t %4.4f \n', totalLength);
148     fprintf(fid, 'Mean branch length: \t\t\t\t %3.4f \n', meanBranchLength)
149     fprintf(fid, 'Standard deviation of branch lengths: \t\t %3.3f \n', ...
150         stdBranchLength)
151     fprintf(fid, 'Skewness of branch lengths: \t\t\t %3.4f \n', ...
152         skewnessBranchLength)
153     fprintf(fid, 'Kurtosis of branch lengths: \t\t\t %3.4f \n', ...
154         kurtosisBranchLength)
155     fprintf(fid, 'Mean number of branches per node: \t\t %3.4f \n', meanbpn)
156     fprintf(fid, 'Standard deviation of branches per node: \t %3.4f \n', stdbpn)
157     fprintf(fid, 'Skewness of branches per node: \t\t\t %3.4f \n', skewnessbpn)
158     fprintf(fid, 'Kurtosis of branches per node: \t\t\t %3.4f \n', kurtosisbpn)
159     fprintf(fid, 'Mean distance to nearest neighbour: \t\t %3.4f \n', meannn)
160     fprintf(fid, 'St.dev. of distance to nearest neighbour: \t %3.4f \n', stdnn)
161     fprintf(fid, 'Skewness of distance to nearest neighbour: \t %3.4f \n', ...
162         skewnessnn)
163     fprintf(fid, 'Kurtosis of distance to nearest neighbour: \t %3.4f \n', ...
164         kurtosisnn)
165     fprintf(fid, 'Mean distance to furthest neighbour: \t\t %3.4f \n', meanfn)
166     fprintf(fid, 'St.dev. of distance to furthest neighbour: \t %3.4f \n', stdfn)
167     fprintf(fid, 'Skewness of distance to furthest neighbour: \t %3.4f \n', ...
168         skewnessfn)
169     fprintf(fid, 'Kurtosis of distance to furthest neighbour: \t %3.4f \n', ...
170         kurtosisfn)

172 fclose(fid)
173 end

175 %% Write to tex
176 texwrite = 1;
177 if texwrite
178     texformat1 = '2';
179     numformat = '%6.2d';
180     fid = fopen(strcat('../analyse/images/', casestr, '/', casestr, ...
181         graphstr, 'stat.tex'), 'w');

```

```

182     stattex(fid, '%06i', '0', '\\\# nodes:', numnodes, '\\\# branches' , ...
183           numbranch, 'total length' , ...
184           round(totalLength))
185     stattex(fid, numformat, texformat1, 'Mean:', meanBranchLength, 'SD:' , ...
186           stdBranchLength, 'Skew' , ...
187           skewnessBranchLength, 'Kurt', kurtosisBranchLength);
188     stattex(fid, numformat, texformat1, 'Mean:', meanbpn, 'SD:' , stdbpn, 'Skew' , ...
189           skewnessbpn, 'Kurt', kurtosisbpn);
190     stattex(fid, numformat, texformat1, 'Mean:', meannn, 'SD:' , stdnn, 'Skew' , ...
191           skewnessnn, 'Kurt', kurtosisnn);
192     stattex(fid, numformat, texformat1, 'Mean:', meanfn, 'SD:' , stdfn, 'Skew' , ...
193           skewnessfn, 'Kurt', kurtosisfn);
194
195     fclose(fid)
196
197 end
198
199 %% Write sorted to tex
200 if textsortwrite
201     texformat1 = '2';
202     texformat2 = '0';
203     numformat1 = '%6.2f';
204     numformat2 = '%6d';
205     if strcmp(graphstr, 'gg')
206         fid1=3;
207     elseif strcmp(graphstr, 'emst')
208         fid1=8;
209     end
210
211     fprintf(fid1, strcat('%%', casestr, '\n'));
212     stattex(fid1, numformat2, texformat2, '\\\# nodes:', numnodes, '\\\# branches:' , ...
213           numbranch, '$\Sigma$ length:' , round(totalLength))
214     fprintf(fid1+1, strcat('%%', casestr, '\n'));
215     stattex(fid1+1, numformat1, texformat1, 'Mean:', meanBranchLength, 'SD:' , ...
216           stdBranchLength, 'Skew:', skewnessBranchLength, 'Kurt:' , ...
217           kurtosisBranchLength);
218     fprintf(fid1+2, strcat('%%', casestr, '\n'));
219     stattex(fid1+2, numformat1, texformat1, 'Mean:', meanbpn, 'SD:' , stdbpn, 'Skew:' , ...
220           skewnessbpn, 'Kurt:', kurtosisbpn);
221     fprintf(fid1+3, strcat('%%', casestr, '\n'));
222     stattex(fid1+3, numformat1, texformat1, 'Mean:', meannn, 'SD:' , stdnn, 'Skew:' , ...
223           skewnessnn, 'Kurt:', kurtosisnn);
224     fprintf(fid1+4, strcat('%%', casestr, '\n'));
225     stattex(fid1+4, numformat1, texformat1, 'Mean:', meanfn, 'SD:' , stdfn, 'Skew:' , ...
226           skewnessfn, 'Kurt:', kurtosisfn);
227
228     for i=0:4
229         if strcmp(casestr, 'case4x25')
230             fprintf(fid1+i, '\\\');
231         else
232             fprintf(fid1+i, '&');
233         end
234     end
235 end
236 end % end function
237

```

Listing B.8: ggstats.m

Function: voronoistat.m – Voronoi statistics

```

1  function voronoistat(varargin)
2  % voronoistat(v,c,casestr,[filwrite],[texwrite],[texsortwrite])
3  % calculates voronoi statistics from the voronoi vertices, v, and cells, c,
4  % which may be obtained using the voronoin function.
5  % Default values for filwrite and texwrite is 1
6  % To manually override default all three write-logicals must be specified.
7  %
8  % NB: texsortwrite assumes that outputfiles for area, shapes and form have the
9  % numerical values of 13, 14 and 15 respectively. Texsortwrite is used to
10 % write data from all different cases into the same tables.
11 %
12 % filwrite produces ascii text output.
13 % texwrite formats output to latex tabular code for use with input.
14 % texsortwrite, same formatting as texwrite.

16  titlefont=26;
17  axisfont=18;
18  %% Parse input
19  v=varargin{1};
20  c=varargin{2};
21  casestr = varargin{3};
22  if numel(varargin)>=6
23      filwrite=varargin{4}
24      texwrite = varargin{5}
25      texsortwrite = varargin{6}
26  else % default
27      filwrite = 1;
28      texwrite = 1;
29      texsortwrite =1;
30  end

32  %% area of the polygons
33  for i=1:length(c)
34      %% area of polygons
35      area(i) = polyarea(v(c{i},1),v(c{i},2));
36      %% perimeter of the polygons
37      xtmp = v(c{i},1);
38      ytmp = v(c{i},2);
39      dxtmp2 = diff([xtmp;xtmp(1)].^2);
40      dytmp2 = diff([ytmp;ytmp(1)].^2);
41      perimeter(i) = sum(sqrt(dxtmp2+dytmp2));
42      %% number of sides
43      nsides(i)=length(c{i});
44  end
45  % format and export output
46  figure(21), bar((0:1:5).*10000, histc(area,(0:1:5).*10000)),
47  title('area','fontsize',titlefont)
48  set(gca, 'XLim',[0,50000], 'FontSize', axisfont)
49  hgexport(gcf, strcat('../analyse/images/',casestr,'/',casestr,'histvarea.eps'))

51  %% area
52  meanarea = mean(area)
53  stdarea = std(area)
54  skarea = skewness(area)
55  kuarea = kurtosis(area)

57  %% perimeter
58  figure(22), hist(perimeter,30), title('perimeter')
59  %% number of sides
60  figure(23), bar(1:max(nsides), histc(nsides, .5:1:max(nsides))),
61  title('number of sides','fontsize',titlefont)

63  %% polygonal form
64  polyform = 4*area./(perimeter).^2;

```

```

65 figure(24), hist(polyform,30), title('polygonal form','fontsize',titlefont)
66 set(gca, 'XLim',[0,0.3], 'FontSize',axisfont)
67 hgexport(gcf, strcat(' ../analyse/images/',casestr,'/',casestr,'histvform.eps'))
68 meanform = mean(polyform)
69 stdform = std(polyform)
70 skform = skewness(polyform)
71 kuform = kurtosis(polyform)

72
73 %% polygonal shape
74 maxdist = zeros(length(c),1);
75 mindist = inf(length(c),1);
76 for i=1:length(c)
77 % finding the length between the vertices
78 ccurrent = c{i};
79 vxcurrent = v(ccurrent,1);
80 vycurrent = v(ccurrent,2);
81 for j=1:length(ccurrent)
82 for k = 1:length(ccurrent)
83 % the other node must not be the same as or next to j.
84 if k~=j && k ~= mod(j-1,numel(ccurrent)) &&...
85 k~=mod(j+1,numel(ccurrent))
86 distance = sqrt((vxcurrent(j)-vxcurrent(k)).^2 +...
87 (vycurrent(j)-vycurrent(k)).^2);
88 if distance > maxdist(i), maxdist(i)=distance; end
89 if distance < mindist(i), mindist(i)=distance; end
90 end
91 end
92 end
93 end
94 polyshape = mindist./maxdist;

95
96 figure(25), hist(polyshape,30), title('polygon shape','fontsize',titlefont)
97 set(gca, 'XLim',[0,1], 'FontSize',axisfont)
98 hgexport(gcf, strcat(' ../analyse/images/',casestr,'/',casestr,'histvshape.eps'))

99
100 meanshape = mean(polyshape);
101 stdshape = std(polyshape);
102 skshape = skewness(polyshape);
103 kushape = kurtosis(polyshape);

104
105 %% Write to file
106 if filewrite
107 fid = fopen(strcat(' ../analyse/images/',casestr,'/',casestr,...
108 'voronoistat.txt'),'w');
109 fprintf(fid,'Mean area of polygon's: \t\t %1.4f \n',meanarea);
110 fprintf(fid,'Standard Deviation of area: \t\t %8.4f \n',stdarea);
111 fprintf(fid,'Skewness of area distribution: \t\t %8.4f \n', skarea);
112 fprintf(fid,'Kurtosis of area distribution: \t\t %8.4f \n', kuarea);
113 fprintf(fid,'Mean of polygonal shape: \t\t %8.4f \n', meanshape);
114 fprintf(fid,'Standard diviation of polygonal shape: \t %8.4f \n', stdshape);
115 fprintf(fid,'Skewness of polygonal shape: \t\t %8.4f \n', skshape);
116 fprintf(fid,'Kurtosis of polygonal shape: \t\t %8.4f \n', kushape);
117 fprintf(fid,'Mean of polygonal form: \t\t %8.4f \n', meanform);
118 fprintf(fid,'Standard diviation of polygonal form: \t %8.4f \n', stdform);
119 fprintf(fid,'Skewness of polygonal form: \t\t %8.4f \n', skform);
120 fprintf(fid,'Kurtosis of polygonal form: \t\t %8.4f \n', kuform);
121 fclose(fid)
122 varargout = {};
123 end

124
125 %% Write to tex
126 if texwrite
127 texformat1 = '2';
128 numformat = '%6.2d';
129 fid = fopen(strcat(' ../analyse/images/',casestr,'/',casestr,...
130 'voronoistat.tex'),'w');

```

```

131     stattex(fid,numformat,txformat1,'Mean:',meanarea,'SD:',stdarea,'Skew:',...
132           skarea,'Kurt:',kuarea);
133     stattex(fid,numformat,txformat1,'Mean:',meanshape,'SD:',stdshape,'Skew:',...
134           skshape,'Kurt:',kushape);
135     stattex(fid,numformat,txformat1,'Mean:',meanform,'SD:',stdform,'Skew:',...
136           skform,'Kurt:',kuform);
137     fclose(fid)
138 end

140 %% Write sorted to tex
141 if texsortwrite
142     txformat1 = '2';
143     txformat2 = '0'
144     numformat1 = '%6.2f';
145     numformat2 = '%6d';

147     fid1=13;
148     fprintf(fid1, strcat('%%',casestr,'\n'));
149     stattex(fid1,numformat1,txformat1,'Mean:',meanarea,'SD:',stdarea,'Skew:',...
150           skarea,'Kurt:',kuarea);
151     fprintf(fid1+1, strcat('%%',casestr,'\n'));
152     stattex(fid1+1,numformat1,txformat1,'Mean:',meanshape,'SD:',...
153           stdshape,'Skew:', skshape,'Kurt:',kushape);
154     fprintf(fid1+2, strcat('%%',casestr,'\n'));
155     stattex(fid1+2,numformat1,txformat1,'Mean:',meanform,'SD:',...
156           stdform,'Skew:',skform,'Kurt:',kuform);

158     for i=0:2
159         if strcmp(casestr,'case4x25')
160             fprintf(fid1+i,'\\');
161         else
162             fprintf(fid1+i,'%');
163         end
164     end
165 end

```

Listing B.9: voronoistat.m

Function: stattex.m

```

1 function stattex(varargin)
2 % StatTex(fid,numformat,txformat,String 1, Num 1, ... ,String n, Num n)
3 % function to simplify the tex formatting. Each string and number pair is
4 % written on a separate line in a latex tabular
5 %
6 % Example: stattex(fid,'%6.2d','2','Mean',meannum);

8 fid      = varargin{1};
9 numformat = varargin{2};
10 txformat = varargin{3};

12 fprintf(fid,['\\begin{tabular}{lD{.}{.}{',txformat,'} \\n'])
13 for i=4:2:(numel(varargin))
14     str = varargin{i};
15     cnum = varargin{i+1};
16     fprintf(fid,cell2mat(strcat({str},{ ' & '},{numformat},{ ' \\ \\ \\ \\ \\n'})),cnum);
17 end
18 fprintf(fid,'\\end{tabular}\\n');

```

Listing B.10: stattex

B.3 Random Site Percolation

1. Script for Generating and Processing a Random Percolation Cluster B-19
2. Largest Cluster in Percolation Matrix B-20
3. Sort Cluster by Distance B-20
4. Backbone of Cluster B-21
5. Shortest Path Across Cluster B-23
6. Direct Paths Across Cluster B-24
7. Locate Neighbouring Pixels that are Inside the Matrix B-25
8. Lookup Table Identifying Bridging Pixels B-25
9. Lookup Table Identifying Spur Pixels B-26

Script: siteperscript.m – Generate and Process a Random Percolation Cluster

```

1  % random site percolation at percolation threshold.
2  % A percolation matrix and; the largest (infinite) cluster ,the transport
3  % backbone and the elastic backbone of the cluster are calculated.
4  % The results are plotted to figures and the matrices saved.

6  clear all;

8  % set parameters
9  p=.592747; % critical threshold
10 siz=[1 1]*256; % size of network

12 % Make new clusters until spanning cluster occupying more than 0.3 is
13 % obtained
14 notdone=1;
15 while notdone
16     m=siz(1); n=siz(2);
17     A=rand(m,n); % percolation network
18     A=im2bw(A,1-p); % threshold
19     % Extract largest cluster
20     N = largestcluster(A);
21     if any(N(:,1)) && any(N(:,size(N,2))) && sum(sum(N))> numel(N)*.3 ,
22         notdone=0;
23     end
24 end
25 save 512fig2/A.mat A

27 %% Find Elastic Backbone
28 tic % store time at start
29 D = sortcluster(N);
30 figure(2), cla, imagesc(D), daspect([1 1 1]), colormap(-gray(256)+1),
31 [EBall, EBclose] = shortestpath(D);
32 disp('Elastic Backbone')
33 toc % report time used

35 % plot largest cluster and elastic backbone
36 EB=EBall(:, :, 1);
37 EB=max(max(EB))-EB;
38 EB(find(EB))=1;
39 figure(4), cla, imagesc(N.*.8+ EB(:, :, 1)),
40 daspect([1 1 1]), colormap(-gray(256)+1);

42 disp('Transport Backbone')
43 tic
44 %% Calculate transport backbone

```



```

45 BB=backbone(N);
46 toc
48 D=sortcluster(N);
49 D(D<10)=10;
50 D(D==max(max(D)))=0;
52 %% Plot all different clusters
53 L = bwlabeln(A,4);
54 RGB = label2rgb(L,'lines','w','shuffle');
55 map=[ 1 1 1;
56       .4 .4 .7;
57       .7 .5 .5;
58       0 0 0;
59       ];
60 figure(1)
61 subplot(1,2,1), cla, imagesc(A), colormap(-gray+1),
62 daspect([1 1 1]), axis off
63 subplot(1,2,2), cla, imagesc(RGB), daspect([1 1 1]),
64 daspect([1 1 1]), axis off
65 figure(2)
66 %% Plot Largest Cluster along with backbone and elastic backbone
67 subplot(1,2,1), cla, image(2*N+BB+EB); colormap(map);
68 daspect([1 1 1]); axis off
69 subplot(1,2,2), cla, imagesc(D), colormap([map;jet(245)]),
70 daspect([1 1 1]), axis off, colorbar
72 %% save the results
73 save 512fig2/matrices.mat A BB D EB EBall N

```

Listing B.11: sitepercsript.m

Function: largestcluster.m – Largest Cluster in a Random Percolation Matrix

```

1 function N = largestcluster(A)
2 % Remove all except largest cluster
3 N=bwlabeln(A,4);
4 S=regionprops(N, 'Area');
5 N=ismember(N, find([S.Area]==max(max([S.Area]))));

```

Listing B.12: largestcluster.m

Function: sortcluster.m – Sort Cluster by Distance

```

1 function D = sortcluster(varargin);
2 % D = sortcluster(N,[I]);
3 % D is a matrix of same size as N. N is a bw-matrix with ones at cluster
4 % location and zeros outside. D is a distance matrix with all cluster sites
5 % containing a value equal to the distance to the left edge. All
6 % non-cluster sites contain a value one greater than the longest distance.
7 %
8 % The optional second input parameter I, is a list of the indices the
9 % distances will be calculated relative to. The default value is all
10 % cluster sites on the left edge.
11
12 % initializing
13 N=varargin{1};
14 F=logical(zeros(size(N))); % percolation front
15 D=zeros(size(N)); % distance matrix
16 if numel(varargin) == 2
17     I = find([ varargin {2}]);
18 else

```

```

19     I=find(N(:,1));
20 end
21 F(I)=1; % in case
22 Done=false(size(D));

24 % Tracking Network
25 while any(any(F))

27     %Adding new sites to network;
28     I=find(F);
29     Done(find(D))=1;
30     D(I)=D(I)+1;
31     D=D+Done;
32     neighbour = legaln(size(N),I);

34     Flast=F;
35     F(neighbour) = N(neighbour).*~Done(neighbour);
36     F = F & ~Flast;
37 end

39 % transform counter to distance and label all non-cluster sites with
40 % maximum value +1
41 Dmax=max(max(D))+1;
42 D= Dmax-D;

```

Listing B.13: sortcluster.m

Function: backbone – Backbone of the Cluster

```

1 function N2 = backbone(N)
2 % BB = backbone(N)
3 % Calculates the transport backbone of the infinite cluster matrix N.
4 %
5 % The backbone consists of all cluster parts that will not break away from
6 % the cluster by the removal of a single pixel. This algorithm locates all
7 % pixels with the potential to break up the cluster from a loop-up table.
8 % These pixels are then, one by one, removed and the new number of areas
9 % with more than one pixel is counted. If the cluster broke apart then the
10 % pixel is removed along with any region not in contact with a direct
11 % route across the cluster. The pixels closest to a direct route has the
12 % highest potential for breaking off large regions which subsequently need
13 % not be checked further, these pixels are therefor checked first.
14 % Finally, because this approach doesnt find "spur-pixels", i.e. the end
15 % point of lines, all 4-connected spurs are removed by a look-up table.
16 % It is assumed that any part removed in this way is smaller than kept part
17 % containing the direct path.

19 N2=N;
20 % Find all direct paths across the cluster
21 DP=directpaths(N);

23 % removing bridging pixels using the liberal hbreakrule function (custom
24 % made) in a lookuptable.
25 lut=makelut(@lbreak,3);
26 L = applylut(N,lut);
27 % Making a list of those pixels that were found not to break up the region
28 L=N & ~L;
29 I=find(L);

31 % Sorting the cluster with the direct paths as startpoints, returning the
32 % distance from each cluster site to the closest direct path. The list of
33 % removable pixels is then sorted according to this.
34 D=sortcluster(N,DP);
35 [Y,Id]=sort(D(I));

```

```

36 I=I(Id);
37
38 for i=1:numel(I)
39     if N2(I(i))
40         N(I(i))=0;
41
42         Lbw=bwlabeln(N,4);
43         S=regionprops(Lbw,'Area');
44
45         nclusters = find([S.Area] > 1);
46         % proceed if the cluster has been split into parts
47         if numel(nclusters) > 1
48             L2(I(i))=1;
49             % checking if the the pixel removed lies on a direct path if
50             % not it can safely be removed.
51             if ~DP(I(i))
52                 N2= N2 & largestcluster(N);
53             else
54                 % cluster has been split in three on DP, the direct path
55                 % parts are to be replaced and any remaining parts
56                 % unattached to the direct path will be removed.
57                 I1 = logical(size(N),I(i));
58                 I1 = nonzeros(I1(N(I1) & N(I1)));
59
60                 I1dp = nonzeros(I1( (DP(I1) & DP(I1)) ));
61                 ondp=Lbw(I1dp);
62                 I1 = nonzeros(I1(~(DP(I1) & DP(I1))));
63                 offdp = Lbw(I1);
64                 l = ~ismember(offdp , ondp);
65                 % cluster to be removed
66                 offdp = unique(nonzeros(offdp(l)));
67                 if ~isempty(offdp)
68                     for tmp=1:numel(offdp)
69                         Itmp = find(Lbw==offdp(tmp));
70                         N2(Itmp)=0;
71                     end
72                 end
73             end
74         end
75     end
76     N2(I(i))=1;
77     N(I(i))=1;
78
79 end
80 % only keeping the largest cluster
81 N2 = largestcluster(N2);
82 % adding any removed pixels from the direct path.
83 N2 = N2 | DP;
84
85 % Continously removing four-connected spur-pixels untill all single pixel
86 % lines are removed.
87 lut=makelut(@lspur,3);
88 L1=false(size(N2));
89 L2=L1;
90 L1 = applylut(N2,lut);
91 while ~isequal(L2,L1)
92     L2=L1;
93     L1 = applylut(L1,lut);
94     L1 = L1 | DP;
95 end
96 % Removing spurs form N2
97 N2 = N2 & L1;

```

Listing B.14: backbone.m

Function: shortestpath.m – Shortest Paths Across the Cluster

```

1  function varargout = shortestpath(D)
2  % [EBall, EBclose, EBbin] = shortestpath(D)
3  % D is a sorted matrix of the cluster with values assigned according to the
4  % distance form the left side.
5  %
6  % EBall is a matrix containing all the shortest routes through the matrix,
7  % with the different routes found on the indices of the third dimension.
8  % The routes are labeled with values according to the distance to the
9  % start.
10 %
11 % EBclose is a matrix of the nearest neighbours to EBall labeled according
12 % to the distance to the start. This may be used in certain algorithms used
13 % to find the transport backbone.
14 %
15 % EBbin is a logical image of EB containing no distance information.
16 %
17 % Finds the shortest path in a sorted cluster
18 % The algorithm floods the matrix from one side and backtracks through the
19 % cluster once a path leads to the other side. This is a width-first
20 % routine checking all possible paths simultainously and aborts when a
21 % route reaches the other side.
22 %
23 % NOTE all non-cluster sites must be assigned value Dmax (not 0)

25 % FINDING ALL SHORTEST HORIZONTAL PATHS
26 Dmax=max(max(D))+1;
27 Dmin=min(D(:, size(D,2)));
28 ID=find(D(:, size(D,2))==Dmin);

30 EB=ones(size(D))*Dmax+1; % initilizing elastic backbone matrix – EB
31 EBall=zeros(size(D,1), size(D,2), numel(ID));
32 EBclose =zeros(size(D));

34 for i=1: numel(ID)
35     m=ID(i); n=size(D,2);
36     EB(m,n)=Dmin;
37     EBclose(m,n)=-1;

39     m = [m+1 m-1];
40     l = ((m<=size(D,1)).*(m>=1) );
41     m = m(find(l));

43     EBclose(m,n)=Dmin;

45     for j=1:Dmin-1
46         I=find(EB==min(min(EB)));

48         %adding legal neighbour cells.
49         r = legaln(size(D),I);
50         Dval=D(r);

52         % removing non-minimum cells
53         l = (Dval==min(min(Dval)));
54         r=r(find(r.*l));
55         Dval=min(min(Dval));

57         if ~isempty(r);
58             EB(r)=Dval;
59             EBclose(r)=-1;
60         end

62     % MAKING LIST FOR Backbone CALCULATION
63     [m,n] = ind2sub(size(D),r);
64     m = cat(1, m+1, m-1, m, m);

```

```

65     n = cat(1, n, n, n+1, n-1);
66     Itmp=[find(n<1);find(n>size(EBclose,2))];
67     if ~isempty(Itmp),
68         n(Itmp)=0; m(Itmp)=0;
69         m=m(find(m)); n=n(find(n));
70     end
71     Itmp=[find(m<1);find(m>size(EBclose,1))];
72     if ~isempty(Itmp),
73         n(Itmp)=0; m(Itmp)=0;
74         m=m(find(m)); n=n(find(n));
75     end
76
77     r2=sub2ind(size(EBclose), m,n);
78     temp=false(size(r2));
79     r=cat(1,r(:),find(EBclose==-1));
80     for teller = 1:numel(r)
81         Ir= find(r2==r(teller));
82         if ~isempty(Ir),
83             temp(Ir)=1;
84         end
85     end
86
87     r2=r2(find(r2.*~temp));
88     EBclose(r2)=Dmin-j;
89     end
90     EBall(:, :, i)=EB;
91     tmp = EB;
92     tmp(tmp==max(max(tmp)))=0;
93     tmp = tmp&tmp;
94     EBbin(:, :, i)=tmp;
95 end
96
97 % parsing output
98 varargout{1}=EBall;
99 varargout{2}=EBclose;
100 varargout{3}=EBbin;

```

Listing B.15: shortestpath.m

Function: directpaths.m – Locate All Direct Paths Across the Cluster

```

1  function DP = directpaths(N)
2  % DP = directpaths(N)
3  % Finds all direct horizontal paths in a sorted cluster. These routes need
4  % not be the globally shortest route, only the shortest route from the
5  % individual start locations at both sides and to the closest point on the
6  % other.
7  %
8  % NOTE all non-cluster sites must have value 0 in N;
9
10 % initialize
11 siz=size(N);
12 DP=false(siz);
13
14 % FINDING ALL SHORTEST HORIZONTAL PATHS
15 for j=1:2;
16     % Flip left/right to locate the shortest paths from both sides
17     N=fliplr(N);
18     D=sortcluster(fliplr(N));
19     % Changing non-cluster distance values from max to 0.
20     D(D==max(max(D)))=0;
21     start = D(:, size(D,2));
22     % start values are indices in a single column, calculating to global
23     % indice values.

```

```

24     Istart=find(start)+siz(1).*(siz(2)-1);
25     % Iterating from each start location
26     for i=1:numel(Istart)
27         current=Istart(i);
28         DP(current)=1;
29         while ~isempty(current)
30             % finding legal neighbours
31             ln=legaln(siz,current);
32             % removing neighbours not on network, or on direct path
33             ln=nonzeros(ln.*(D(ln)&D(ln)));
34             % using the minimal route, removing repetitions
35             ln = unique(ln(D(ln)==min(D(ln))));
36             current = nonzeros(ln.*(~(DP(ln)&DP(ln))));
37             DP(current)=1;
38         end
39     end
40     % Flipping the directpath matrix back
41     DP=fliplr(DP);
42 end

```

Listing B.16: directpaths.m

Function: legaln.m – Locate Legal Neighbours (inside matrix)

```

1 function neighbour = legaln(siz,I)
2 % I = legaln(siz,I)
3 % Returns all indices in I that are legal, i.e. inside the boundary siz.
4 [m,n]=ind2sub(siz,I);
5 m = [m+1 m-1 m m];
6 n = [n n n+1 n-1];
7 l = ((m<=siz(1)) & (m>=1)) & ((n<=siz(2)) & (n>=1));
8 m=nonzeros(m.*l); n=nonzeros(n.*l);
9 neighbour = sub2ind(siz,m,n);

```

Listing B.17: legaln.m

Function: lhbREAK.m – Lookup Table identifying Bridging Pixels

```

1 function r = lhbREAK(x);
2 % r = lhbREAK(x)
3 % liberal (i.e. 4-connected) break rule breaking H-connected regions for
4 % lookup table. The input x is a 3x3 binary matrix of some configuration.
5 % If the middle element can be changed from 1 to 0 without breaking up a
6 % 4-connected region then this is done. Otherwise the original value is
7 % kept.
8
9 x1=x;
10 x2=rot90(x1);
11 x3=rot90(x2);
12 x4=rot90(x3);
13 switch 1
14     % all zeros stay
15     case x(5)==0, r=0;
16     % x(5) == 1 FOR REMAINING CASES
17
18     % non bridging pixels are kept.
19     % NOT
20     % * * * * I *
21     % I I I I I *
22     % * * * * * *
23     case ~(...
24         isequal(x(4:6),[1 1 1]) || ...

```

```

25         isequal(x([2,5,8]),[1 1 1])      ||...
26         (x(2) && x(4))                    ||...
27         (x(2) && x(6))                    ||...
28         (x(4) && x(8))                    ||...
29         (x(6) && x(8))                    )
30     r=1;

32     % if 8 or more pixels are 1 r cannot break a path
33     % 1 1 0
34     % 1 1 1
35     % 1 1 1
36     case sum(x(:))>=8, r=1;

38     % if one complete side is 1, and the middle opposite is 0 then r cannot
39     % break a path
40     % 1 1 1
41     % 1 1 1
42     % * 0 *
43     case any( ...
44         [sum(x1(1:6)) sum(x2(1:6)) sum(x3(1:6)) sum(x4(1:6))]==6 ...
45         & [x1(8)==0 x2(8)==0 x3(8)==0 x4(8)==0] ...
46         ),
47     r=1;

49     % if x is on a corner it cannot break path
50     % * 1 1
51     % 0 1 1
52     % * 0 *
53     case ( x1(2) && x1(3) && x1(6) && ~x1(4) && ~x1(8) ) ||...
54         ( x2(2) && x2(3) && x2(6) && ~x2(4) && ~x2(8) ) ||...
55         ( x3(2) && x3(3) && x3(6) && ~x3(4) && ~x3(8) ) ||...
56         ( x4(2) && x4(3) && x4(6) && ~x4(4) && ~x4(8) )
57     r=1;
58     otherwise
59         r=0;
60 end

```

Listing B.18: lhbreak.m

Function: lspur.m – Lookup Table identifying Spur Pixels

```

1  function r = lspur(x);
2  % liberal (i.e. 4-connected) spur removing function for lut, makelut and applylut
3  % this removes the end points of (4-connected) lines without removing small
4  % objects completely.

6  switch 1
7      % all zeros stay
8      case x(5)==0, r=0;
9      % x(5) == 1 FOR REMAINING CASES

11     case x(2)+x(4)+x(6)+x(8)==1
12         r=0;

14     otherwise
15         r=1;
16 end

```

Listing B.19: lspur.m

B.4 Fractal Analysis

1. Regrid Image and Count Pixels for Box Counting Algorithm B-27
2. Count Number of Pixels for Sandbox Algorithm B-28
3. Mass and Correlation Estimate B-29
4. Fourier Estimate B-31
5. Angleverage for Mass, Correlation and Fourier Estimates B-33
6. Linear Fit of the Most Linear Region B-34
7. Recursive Linear Fit of All Linear Regions Using Autofit B-36
8. Plot Linear Fit Graphs and Datatable B-37

Function: regridboxcountpix.m – Count Number Occupied Pixels in Resized Image for Box Count Method

```

1  function [num] = regridboxcountpix(A, siz)
2  % num = REGRIDBOXCOUNTPIX(A, size);
3  % regridboxcountpix maps the matrix A over to a new matrix M. The size of
4  % new pixels in M is determined by the scalar size, siz, containing the
5  % number old pixels each new pixel is covering along each axis. Any cell in
6  % M that maps to a cell, or part of a cell in A, that contains a non-zero
7  % entry is set to 1. The function then counts all non empty cell,
8  % repositions the grid and counts again, compairs all possible positions and
9  % returns the minimum number of non-empty boxes

11 % If matrix is in non-logical format, all nonzero entries are converted to
12 % 1 in the conversion to logical.
13 if ~islogical(A)
14     A=logical(A);
15 end

17 % Matrix dimensions
18 x=size(A,1); y=size(A,2);
19 m=ceil(x./siz); n=ceil(y./siz);

21 % Permutations with a displacement along either row or column direction,
22 % will require a matrix one size larger for that direction. This is because
23 % these displaced grids have the last cell split between the two sides of
24 % the matrix is not joined back together as one, a 'wrapping' of the image
25 % across the edge is not wanted. This is handled by leaving the last row or
26 % column as an all-zero vector, wich will therefor not contribute to the
27 % sum of non-empty cells.
28 k = siz - 1;
29 l = siz - 1;
30 % Preallocating memory
31 M=zeros(m+1,n+1,ceil(x./m-1),'single');
32 % Using the logical bit-format to save memory.
33 M=logical(M);

35 %loops through the new matrix.
36 for i=1:m+1;
37     % calculate the indices of the first and last cells of the old matrix wich
38     % are contained in the new cells. Cells outside the matrix are ignored.
39     % from/to-x and from/to-y are vectors containg all the displaced
40     % versions as well as the original.

42     fromx = fix((i-1)*x/m+1)-(0:k);      I = find(fromx < 1); fromx(I)=1;
43     tox = ceil(i*x/m)-(0:k);              I = find(tox > x); tox(I)=x;
44     for j=1:n+1;

```



```

45     fromy = fix((j-1)*y/n+1)-(0:1);   I = find(fromy < 1); fromy(I)=1;
46     toy = ceil(j*y/n)-(0:1);         I = find(toy > y); toy(I)=y;
47     % loop through the different displacements
48     dkdl = 1;
49     for dk=1:k+1
50         for dl=1:l+1
51             % Checking if any of the cells in the old matrix, that is
52             % (partially) contained in a given cell in the new matrix,
53             % is non-zero. If so the corresponding entry in the new
54             % matrix is set to 1.
55             M(i,j,dkdl)=any(any( A(fromx(dk):tox(dk),fromy(dl):toy(dl)) ));
56             dkdl = dkdl+1;
57         end
58     end
59 end
60 end
61
62 % Find lowest possible number of non-empty cells.
63 num=min(sum(sum(M)));

```

Listing B.20: .m

Function: sandboxnumber.m – Count Number of Pixels For Sandbox Algorithm

```

1  function [num,r] = sandboxnumber(varargin)
2  % [NUM] = SANDBOXNUMBER(IMAGE,INDEX,NPOINTS)
3  % Used for calculating the sandbox dimension of IMAGE
4  %
5  % Calculates the number of occupied positions around INDEX in the image
6  % matrix A and returns an nx2 vector, where n is the number of indices in
7  % INDEX, containing the sum of the number of occupied positions over all
8  % indices in INDEX and in the second column the number of indices in INDEX
9  % that has contributed to this sum.
10 %
11 % INDEX may be a nxl vector containing the indices or a nx2 matrix containing
12 % the subscripts of the points.
13 %
14 % Optional argument NPOINTS reduces the list of possible neighbourhoods to
15 % check to a list of logarithmicly distributed subset. Default value (0)
16 % skips this redistribution and uses the full data range.
17
18 %% PARSE INPUT
19 A=varargin{1};
20 Index=varargin{2};
21 % log reduction
22 if numel(varargin)>2
23     nL = varargin{3};
24 else
25     nL = 0;
26 end
27
28 Lmax = min(round(size(A)./2));
29 if nL > 0
30     % logarithmic distribution of neighbourhood widths
31     logL = log10(1) : log10(Lmax)/nL : log10(Lmax);
32     pLs = unique(round(10.^logL));
33 else
34     % all possible neighbourhood widths
35     pLs = 1:Lmax;
36 end
37
38 % if indices are supplied, transform to subscripts
39 if size(Index,2)==1
40     [temp1,temp2]=ind2sub(size(A),Index);

```

```

41     Index=cat(2,temp1,temp2);
42     elseif size(Index,2)~=2
43         disp('ERROR: Input, wrong format')
44         return
45     end
46
47 % Initialize
48 I=Index;
49 num=[0 0];
50 for i=1:size(I,1)
51     temp=I(i);
52     m=I(i,1);
53     n=I(i,2);
54     % locate distance to nearest edge
55     edge=min( min([m-1,size(A,1)-m]) , min([n-1,size(A,2)-n]) );
56     % all possible widths inside the matrix
57     Ltmp = pLs(pLs<=edge);
58     for j=1: numel(Ltmp)
59         L = Ltmp(j);
60         % subset of matrix containing only sandbox
61         subA=A(m-L:m+L,n-L:n+L);
62         % adding sum of this sandbox to the other sandboxes of same size
63         if ( size(num,1)>=L)
64             num(L,1)=sum(sum(subA))+num(L,1);
65             num(L,2)=1+num(L,2);
66         else
67             num(L,1)=sum(sum(subA));
68             num(L,2)=1;
69         end
70     end
71 end
72
73 % parse output
74 r = pLs;

```

Listing B.21: sandboxnumber.m

Function: massestimate.m – Mass and Correlation Estimate

```

1  function varargout = massestimate(varargin)
2  % [Dim, normr, S, k] = massestimate(IMAGE, nk, kmin, fmode, distribution)
3  % [DimM, normrM, SM, kM, DimC, normrC, Sc, kc] = ...
4  %     massestimate(IMAGE, nk, kmin, fmode, distribution)
5  %
6  % Calculate fractal dimension by the mass estimate
7  % Because calculation of the mass estimate goes a long way to calculate the
8  % correlation estimate as well another optional set of output parameters is
9  % available for the correlation estimate output.
10 %
11 % NK is the number of different frequency bins the spectral density is
12 % averaged into. KMIN is the smallest frequency allowed to participate in
13 % the calculation. FMODE specifies properties of the fourier
14 % transform of the supplied image. FMODE takes the following values which
15 % specify transforms to be done before the matrix is transformed into the
16 % autocorrelation.
17 %
18 %     0 – apply fourier transform with padding
19 %     1 – apply fourier transform without padding
20 %     2 – IMAGE is transformed, apply conversion to spectral
21 %         density
22 %     3 – IMAGE is transformed and converted to spectral density,
23 %         no further transformation required before calculation.
24 %
25 % DIM is the power law scaling of the spectral density, i.e. the fourier

```

```

26 % estimate of the fractal dimension.
27 % NORMR is the norm of the residuals of the loglog linear fit.
28 % S is the angle averaged spectral density as a function of K.
29 %
30 % DEFAULTS: nk      = 20
31 %           kmin    = 10
32 %           mode    = 3 (spectral density is supplied)

33 %% parse input
34 if numel(varargin)==0
35 %     error('fourier:wronginput',strcat('Wrong number of inputs in ',...
36 %     '\n[Dim,normr,S,r]=fourierestimate(F,[nk],[kmin],[padding]'))
37 end
38 if numel(varargin) >= 5
39     distr = varargin{5};
40 else
41     distr = 'logarithmic';
42 end
43 if numel(varargin) >= 4
44     fmode = varargin{4};
45 else
46     fmode=2;
47 end
48 if numel(varargin) >= 3
49     kmin = varargin{3};
50 else
51     kmin = 10;
52 end
53 if numel(varargin) >=2
54     nk = varargin{2};
55     if nk==0,
56         nk=ceil(min(size(varargin{1}))./2);
57     end
58 else
59     nk = 20;
60 end
61 if numel(varargin) >=1
62     F = varargin{1};
63 else
64     F=rand(1024);
65 end

66 %% apply remaining transforms according to fmode
67 switch 1
68     case fmode == 0
69         F = fft2(F,2*size(F,1)-1,2*size(F,2)-1);
70         S = F.*conj(F);
71     case fmode == 1
72         F = fft2(F);
73         S = F.*conj(F);
74     case fmode == 2
75         S = F.*conj(F);
76     case fmode == 3
77         S = F;
78 end

83 %% autocorrelation - calculate , shift and normalize
84 C=ifft2(S);
85 C=fftshift(C);
86 C=real(C);
87 C=C./max(max(C));
88 %% %% cropping off rand-off errors
89 % C(C<(.00001))=0;
90 % figure(1), imagesc(log10(abs(real((C))))), colorbar, colormap(jet)
91 % pause

```

```

93 %% angle averaging
94 % f(x,y) -> f(r)
95 % [C,r] = angleaverage(C,nk,'logarithmic');
96 [C,r] = angleaverage(C,nk,distr,kmin);

98 %% cropping off rand-off errors
99 l= (abs(C) < 10.^(-10));
100 C(1) = [];
101 r(1) = [];

104 %% CORRELEATION DIMENSION
105 % Because the calculation of the mass dimension have done the hard work
106 % necessary to calculate the correlation dimension this is returned as well

108 %% loglog fitting
109 lC = log10(C);
110 lr = log10(r);
111 [P,s]=polyfit(lr,lC,1);
112 % extracting norm of residuals
113 s=struct2cell(s);
114 normr=s(3);

116 %% parsing output for correlation dimension
117 varargout{5}=P; % Slope over entire range of Correlation estimate
118 varargout{6}=normr; % norm of residulas from the regression
119 varargout{7}=C; % Angle averaged spectral density
120 varargout{8}=r; % frequency bins

122 %% integrate by piecewise cubic spline initerpolation
123 CR = C.*r;
124 ftype = fittype('spline');
125 fit1 = fit(r',CR',ftype);
126 inty = integrate(fit1,r,r(1));
127 M = inty(1:numel(inty))-inty(1);

129 %% transforming column to row vector
130 M=M(:);
131 r=r(:);

133 %% loglog fitting
134 lM = log10(M);
135 lr = log10(r);
136 [P,s]=polyfit(lr,lM,1);
137 % extracting norm of residuals
138 s=struct2cell(s);
139 normr=s(3);

141 %% parsing output
142 varargout{1}=P; % Slope of entire range of Mass estimate
143 varargout{2}=normr; % norm of residulas from the regression
144 varargout{3}=M; % Angle averaged spectral density
145 varargout{4}=r; % frequency bins

```

Listing B.22: massestimate.m

Function: fourierestimate.m – Fourier Estimate

```

1 function varargout = fourierestimate(varargin)
2 % [Dim, normr, S, k] = fourierestimate(IMAGE, nk, kmin, fmode)
3 %
4 % Calculate fractal dimension by the fourier estimate
5 %

```

```

6 % NK (num) is the number of different frequency bins the spectral density is
7 % averaged into. KMIN (num) is the smallest frequency allowed to participate in
8 % the calculation. FMODE (num) specifies properties of the fourier
9 % transform of the supplied image. FMODE takes the following values
10 %
11 %     0 - apply fourier transform with padding
12 %     1 - apply fourier transform without padding
13 %     2 - IMAGE is transformed, apply conversion to spectral
14 %         density
15 %     3 - IMAGE is transformed and converted to spectral density,
16 %         no further transformation required.
17 %
18 % DIM is the power law scaling of the spectral density, i.e. the fourier
19 % estimate of the fractal dimension.
20 % NORMR is the norm of the residuals of the loglog linear fit.
21 % S is the angle averaged spectral density as a function of K.
22 %
23 % DEFAULTS: nk           = 20
24 %           kmin        = 10
25 %           mode         = 3 (spectral density is supplied)
26 %           distribution = logarithmic
27 %
28 %% parse input
29 if numel(varargin)==0
30 %     error('fourier:wronginput',strcat('Wrong number of inputs in ',...
31 %     '\n[Dim,normr,S,r]=fourierestimate(F,[nk],[kmin],[padding]'))
32 end
33 if numel(varargin) >= 5
34     distr = varargin{5};
35 else
36     distr = 'logarithmic';
37 end
38 if numel(varargin) >= 4
39     fmode = varargin{4};
40 else
41     fmode=2;
42 end
43 if numel(varargin) >= 3
44     kmin = varargin{3};
45 else
46     kmin = 10;
47 end
48 if numel(varargin) >=2
49     nk = varargin{2};
50 else
51     nk = 20;
52 end
53 if numel(varargin) >=1
54     F = varargin{1};
55 else
56     F=rand(1024);
57 end
58 %%
59 %% apply remaining transforms according to fmode
60 switch 1
61     case fmode == 0
62         F = fft2(F,2*size(F,1)-1,2*size(F,2)-1);
63         S = F.*conj(F);
64     case fmode == 1
65         F = fft2(F);
66         S = F.*conj(F);
67     case fmode == 2
68         S = F.*conj(F);
69     case fmode == 3
70         S = F;
71 end

```

```

73 %% spectral density – shift and normalize
74 S=fftshift(S);
75 S=S./max(max(S));

77 %% angle averaging
78 % f(x,y) -> f(r)
79 [S,r] = angleaverage(S,nk,distr,kmin);

81 %% loglog fitting
82 lS = log10(S);
83 lr = log10(r);
84 [P,s]=polyfit(lr,lS,1);
85 % extracting norm of residuals
86 s=struct2cell(s);
87 normr=s(3);

89 %% parsing output
90 varargout{1}=P; % Slope of entire range
91 varargout{2}=normr; % norm of residulas from the regression
92 varargout{3}=S; % Angle averaged spectral density
93 varargout{4}=r; % frequency bins

```

Listing B.23: fourierestimate.m

Function: angleaverage.m – Angle Average for Mass, Correlation and Fourier Estimate

```

1 function [F,r] = angleaverage(F,nk,style,kmin)
2 % [F,r] = angleaverage(M,nk)
3 % [F,r] = angleaverage(M,nk,style)
4 % [F,r] = angleaverage(M,nk,style,kmin)
5 %
6 % Returns a vector of M values avaraged over nk circles around its centre
7 % style takes the following values:
8 %     'logarithmic'
9 %     'arithmetic'
10 % default value is logarithmic
11 %
12 % kmin>0 specifies an inner radius, all points within this will be ignored.

14 if isempty(style)
15     style='logarithmic'
16 end
17 if isempty(kmin)
18     kmin=0;
19 end

21 %% angle averaging
22 % locating centre pixel and finding shortest distance from centre to edge
23 mctr = ceil((size(F,1)+1)/2);
24 nctr = ceil((size(F,2)+1)/2);
25 kmax = min(mctr,nctr);

27 % initilizing the positions of the bins the data is avaraged into
28 if isequal(style,'logarithmic')
29     logk = log10(kmax)/nk : log10(kmax)/nk : log10(kmax);
30     k = 10.^logk;
31 elseif isequal(style,'arithmetic')
32 %     k = kmax/nk : kmax/nk : kmax;
33     k = kmin : kmax/nk : kmax;
34 else
35     error('angleaverage:unknownstyle',...
36         'ERROR: unrecognized style in angleaverage')
37 end

```

```

38 k=k(k>=kmin);
40 %% initializing the positions of the bins the data is averaged into
41 logk=log10(k);
43 %% initialize angle average
44 % counter array for the number of entries in each bin
45 Frc = zeros(1,numel(k));
46 % bins for storing F with reduced resolution
47 Fr = zeros(1,numel(k));
48 % bins for storing the mean distance of the entries in each bin
49 Frr = zeros(1,numel(k));
50 for m=1:size(F,1)
51     dm = mctr-m;
52     for n=1:size(F,2)
53         dn = nctr-n;
54         r = sqrt(dm^2+dn^2);
55         % centre item is out of bounds and values > kmax are only defined
56         % in corners
57         if r ~= 0 && r<=kmax && r>=kmin
58             % finding the nearest k (the v value is required in syntax)
59             [v,rk] = min(abs(logk-log10(r)));
60             Frc(rk) = Frc(rk) + 1;
61             Fr(rk) = Fr(rk) + (F(m,n));
62             Frr(rk) = Frr(rk) + (r);
63         end
64     end
65 end
67 % removing empty bins
68 l = Frc & Frc;
69 Fr = Fr(l);
70 Frr = Frr(l);
71 Frc = Frc(l);
72 k = k(l);
74 % dividing sum to mean
75 F = Fr./Frc;
76 r = Frr./Frc;

```

Listing B.24: angleaverage.m

Function: autofit.m – Linear Fit of the Most Linear Region

```

1 function varargout = autofit(varargin)
2 % [P, from, to, normr, quality] = autofit(x,y)
3 % [P, from, to, normr, quality] = autofit(x,y,dxmin)
4 % [P, from, to, normr, quality] = autofit(x,y,dxmin,Imin,Imax)
5 % [P, from, to, normr, quality] = autofit(x,y,dxmin,Imin,Imax,minfindwidth)
6 %
7 % Where dxmin is the minimum number of indices considered, Imin and Imax
8 % are the bounds of the search, points outside this range are ignored and
9 % minfindwidth is the smallest continuous region scanned for a fit, measured
10 % as the fraction of the number of the logarithmic width.
11 %
12 % Default values:    dxmin      = 0.20 * numel(x)
13 %                   Imin       = 1
14 %                   Imax       = numel(x)
15 %                   minfindwidth = 0.20
16 %
17 % This algorithm finds the single subregion of the function y(x) that can
18 % best be described by a linear fit. A fit is considered better with
19 % increasing logarithmic width multiplied by (1-normR)^5.
20 %

```

```

21 % This is brute force solution to the problem testing all legal subregions,
22 % according to its optional arguments, and comparing them to find the best
23 % one.

24
25 %% Parse input
26 x = varargin{1};
27 F = varargin{2};
28 if numel(varargin)>=3
29     dxmin=varargin{3};
30 else
31     dxmin=round(.20.*numel(x));
32 end
33 if numel(varargin)>=5
34     Imin = varargin{4};
35     Imax = varargin{5};
36 else
37     Imin = 1;
38     Imax = numel(x);
39 end
40 if numel(varargin) >= 6
41     minfindwidth = varargin{6};
42 else
43     minfindwidth = .20;
44 end
45 if Imax - Imin < dxmin
46     disp('ERROR: (Imax - Imin) < dxmin')
47     return
48 end

49
50 %% Calculate all possible continous linear fits
51 for i=Imin:Imax-dxmin
52     for j=i+dxmin:Imax
53         [Pn,sn]=polyfit((x(i:j)),(F(i:j)),1);
54         Pj(j)=Pn(1); % slope
55         sn=struct2cell(sn);
56         normrj(j)=sn{3}; % norm of residuals
57         if isnan(normrj(j))
58             normrj(j)=inf;
59         end
60         xmaxj(j-i-dxmin+1)=j;
61     end
62     P{i}=Pj;
63     normr{i}=normrj(normrj&normrj);
64     xmin{i}=i;
65     xmax{i}=xmaxj;
66     clear Pj normrj xminj xmaxj
67 end

68
69 %% Compare the quality of each fit to find the best within each range
70 for i=Imin:numel(P)
71     from=i;%xmin(i);
72     to =xmax{i};
73     norm=normr{i};
74     for j=1:numel(norm)
75         % xfactor is percentage of total range
76         xfactor = ((x(to(j)))-(x(from)))/(x(numel(x))-x(1));
77         nfactor = (1.-norm(j)).^5;
78         % checking if current location meats requiriements
79         if xfactor >= minfindwidth
80             qual(j) = xfactor.*nfactor;
81         else
82             qual(j) = 0;
83         end
84         l(j)=norm(j)<.05;
85     end
86     clear val ind

```



```

87     [ val , ind ] = max(qual);
88     qval{ i } = val;
89     qfrom{ i } = i;
90     qto{ i } = to( ind );
91     qual = [];
92 end

94 %% Find the best fit comparing all ranges
95 [ quality , ind ] = max(cell2mat(qval));
96 % adjust for any empty cells being removed before max(...)
97 ind = ind + Imin - 1;
98 from = qfrom{ ind };
99 to = qto{ ind };
100 [ P , s ] = polyfit(x( from : to ), F( from : to ), 1);
101 s = struct2cell(s);
102 normr = s{3};

104 %% Parse output
105 if (x(to) - x(from)) >= minfindwidth * (x(numel(x)) - x(1))
106     varargout{1} = P;
107     varargout{2} = from;
108     varargout{3} = to;
109     varargout{4} = normr;
110     varargout{5} = quality;
111 else
112     varargout{1} = [];
113     varargout{2} = [];
114     varargout{3} = [];
115     varargout{4} = [];
116     varargout{5} = [];
117 end
118 end % end autofit

```

Listing B.25: autofit.m

Function: reclinfilt.m – Recursive Linear Fit of all Regions (starting with the best)

```

1 function [ fits ] = reclinfilt(x,y,fromx ,tox ,sp , fits )
2 % Recursive Linear Fit
3 %
4 % {fits} = reclinfilt(x,y,from_ind , to_ind , searchcriteria )
5 %
6 % This function recursively finds all linear fits in the region of FROMX to
7 % TOX within the bounds of the searchcriteria .
8 %
9 % searchcriteria = [minn , minsw , minfw];
10 %
11 % MINN is the minimum number of points allowed in a fit , default = 5.
12 % MINSW is the minimum search width , only regions wider than this
13 % fraction of the total width is searched , default = 0.20.
14 % MINFW is the minimum accepted region a fit can be made across , again
15 % measured in fraction of total width , default = 0.15.
16 %
17 % The output is a one-dimensional cell vector where each index contains the
18 % following 7 matrices:
19 %
20 % fits{ i } = {slope , intercept , from_ind , to_ind , norm of residuals , ...
21 %             width , quality of fit};

23 %% Declare and parse input
24 if exist('fits') ~=1
25     fits = {};
26 end
27 if exist('sp') ~= 1 || isempty(sp)

```

```

28     % use defaults
29     minn = 5;
30     minsw = .2 .* ((x(numel(x)))-x(1));
31     minfw = .15;
32     sp = [minn, minsw, minfw];
33 else
34     minn = sp(1);
35     if numel(sp) >= 2
36         minsw = sp(2).*(x(numel(x))-x(1));
37     else
38         minsw = .2 * ((x(numel(x)))-x(1));
39     end
40     if numel(sp) == 3;
41         minfw = sp(3);
42     else
43         minfw = .15;
44     end
45 end

47 %% Run autofit recursively to find all linear regions in agreement with the
48 %% search criteria vector.
49 [P, from, to, normr, qual] = autofit(x,y,minn,fromx,tox,minfw);
50 if ~isempty(P)
51     fits{numel(fits)+1} = {P(1) P(2) from to normr...
52         [(x(to)-x(from))/(x(numel(x))-x(1))] qual};

54     if ((from - fromx) > minsw) && ((from - fromx) > minn)
55         fits = reclinfit(x,y,fromx,from,sp,fits);
56     end
57     if ((tox - to > minsw)) && ((tox - to) > minn)
58         fits = reclinfit(x,y,to,tox,sp,fits);
59     end

61 end

63 %% END FUNCTION RECLINFIT
64 end

```

Listing B.26: reclinfit.m

Function: `linfitplot` – Plot Graphs and Data Table of Linear Regions

```

1 function [fh1 fh2] = linfitplot(x,y,mode,fits ,fid)
2 % [figurehandle1 figurehandle2] = linfitplot(x,y,mode,fits)
3 % [figurehandle1 figurehandle2] = linfitplot(x,y,mode,fits ,fid)
4 %
5 % mode = 'normal' or 'log'
6 % mode specifies if the data should be plotted onto loglog-graphs or not.
7 %
8 % This function plots all linear fits along with the function, y(x), in a
9 % single figure, and draws patches beneath each linear region, and
10 % specifies where the fits intersect. The handle of this figure is
11 % returned in fh1.
12 %
13 % Furthermore the gradient of the function, with the same information
14 % represented in that figure is returned in fh2.
15 %
16 % If the optional parameter fid is specified the legend of the plot is
17 % written, formatted to the latex tabular environment, to the file with
18 % identification fid. Otherwise a smaller legend is written to screen.

20 %% parse input and set flags
21 if exist('fid','var') ~= 1
22     fid=[];

```

```

23 end
24 x = x(:); %->column form
25 y = y(:); %->column form
26 if strcmp(mode,'log')
27     % remove negative values, they are outside the real range of log.
28     realind = y >= 0;
29     x = x(realind);
30     y = y(realind);
31     lx = log10(x);
32     ly = log10(y);
33 end
34 if exist('fid','var') == 1 && ~isempty(fid)
35     if fid ~= 1
36         writelegendtofile = 1;
37         writelegendtoscreen = 0;
38     else
39         writelegendtoscreen = 1;
40         writelegendtofile = 0;
41     end
42 else
43     writelegendtofile = 0;
44     writelegendtoscreen = 0;
45 end
46 fh1 = figure; clf, hold on
47 fh2 = figure; clf, hold on

49 %% define colormap
50 % Use the middle portion of the hue part of the hsv-color representation to
51 % achieve a colormap without complementary colors.
52 numf = numel(fits);
53 if numf == 1,
54     cmap = [.969 .2 .2];
55 else
56     cmap = fftshift(hsv(2*numf),1);
57     cmap = cmap./1.3;
58     cmap = cmap+.2;
59     cmap(cmap>1) = 1;
60     cmap(cmap<0) = 0;
61     cmap=cmap(floor(numf./2)+1:floor(3*numf./2),:);
62 end
63 colormap(cmap);

65 %% sort fits
66 sortvec = zeros(numel(fits),1);
67 for i=1:numel(fits)
68     sortvec(i) = fits{i}{3};
69 end
70 [tmp sortind] = sort(sortvec);
71 fits = fits(sortind);

73 %% determining min legal function values, cropping at 10^-10
74 miny = min(y);
75 if miny < 10^-3 && strcmp(mode,'log')
76     if (miny == -inf)
77         ytmp = y;
78         ytmp(ytmp== -inf) = [];
79         miny = min(y);
80     elseif ((miny == 0) && strcmp(mode,'log'))
81         ytmp = y;
82         ytmp(ytmp==0) = [];
83         miny = min(ytmp);
84     end
85     if miny < 10^-10
86         miny = 10^-10;
87     end
88 end

```

```

90 %% plot patch regions
91 figure(fh1)
92 for i=1:numel(fits)
93     from = fits{i}{3};
94     to   = fits{i}{4};

95
96     px = ([x(from);x(from:to);x(to)]);
97     py = ([miny;y(from:to);miny]);
98     regionid(i) = patch(px,py,cmap(i,:));
99 end

101 %% plot slopes
102 for i=1:numel(fits)
103     slope = fits{i}{1};
104     intercept = fits{i}{2};

105
106     if strcmp(mode,'log')
107         fitid(i) = plot(x,10.^(lx.*slope+intercept),'color',...
108             cmap(i),'LineWidth',1.5);
109         plot(x,10.^(lx.*slope+intercept),'k:');
110     elseif strcmp(mode,'normal')
111         fitid(i) = plot(x,(x.*slope+intercept),'color',...
112             cmap(i),'LineWidth',1.5);
113         plot(x,(x.*slope+intercept),'k:');
114     end
115 end

117 %% locate fit intersections
118 % if upperflag
119 for i=1:numel(fits)-1
120     slope = fits{i}{1};
121     intercept = fits{i}{2};
122     slope2 = fits{i+1}{1};
123     intercept2 = fits{i+1}{2};
124     interx(i) = (intercept2-intercept)/(slope-slope2);
125     intery = slope*interx(i)+intercept;
126     if strcmp(mode,'log')
127         interx(i)=10.^interx(i);
128         intery=10.^intery;
129     end
130     plot([interx(i) interx(i)],[miny intery],'-','color',[0 0 0],...
131         'LineWidth',2);
132 end

134 %% plot function
135 plot(x,y,'k','Linewidth',1.5);

137 %% format figure
138 hold off
139 % scales
140 axis square
141 if strcmp(mode,'log')
142     set(gca, 'XScale','log','YScale','log',...
143         'YLim',[miny./10.^(.2) max(y).*10.^(.2)],'XLim',[min(x) max(x)],...
144         'XMinorTic','on','YMinorTic','on','XMinorTic','on');
145 elseif strcmp(mode,'normal')
146     set(gca, 'YLim',[miny-.2 max(y)+.2],'XLim',[min(x) max(x)]);
147 end

149 %% GRADIENT
150 figure(fh2), clf, hold on

152 if strcmp(mode,'log')
153     grad = gradient(ly,lx);
154 elseif strcmp(mode,'normal')

```

```

155     grad = gradient(y,x);
156 end
157 %% finding min and max non-inf gradient values
158 mingrad = min(grad);
159 if mingrad == -inf,
160     grad2 = grad;
161     grad2(grad==-inf) = [];
162     mingrad=min(grad2);
163 end
164 maxgrad = max(grad);
165 if maxgrad == inf,
166     grad2 = grad;
167     grad2(grad==inf) = [];
168     maxgrad=max(grad2);
169 end

171 %% plot patch regions
172 figure(fh2)
173 for i=1:numel(fits)
174     from = fits{i}{3};
175     to   = fits{i}{4};
176     px = ([x(from);x(from:to);x(to)]);
177     py = ([mingrad;grad(from:to);mingrad]);
178     regionid(i) = patch(px,py,cmap(i,:));
179 end

181 %% plot slopes
182 for i=1:numel(fits)
183     slope = fits{i}{1};
184     plot([x(1) x(numel(x))],[slope , slope],'.-','color',...
185         cmap(i,:), 'LineWidth',1.5)
186     plot([x(1) x(numel(x))],[slope , slope], 'k')

188 end

190 %% plot gradient
191 plot(x,grad,'b','LineWidth',1.5)

193 %% format gradient plot
194 if strcmp(mode,'log')
195     set(gca, 'XScale','log','XLim',([min(x) max(x)]),...
196         'YLim',[mingrad-1, maxgrad+1],...
197         'YMinorTic','on','XMinorTic','on');
198 elseif strcmp(mode,'normal')
199     set(gca, 'XLim',([min(x) max(x)]), 'YLim',[mingrad-1, maxgrad+1]);
200 end

202 %% write legend data to file
203 if writelegendtofile
204     fprintf(fid, '\\newcolumn{e}{c@{ }} \\n');
205     fprintf(fid, '\\newcolumn{d}[1]{D{.}{.}{#1}@{ }} \\n');
206     for i=1:numel(fits)
207         fprintf(fid, '\\definecolor{c%d}{rgb}{%4.4g,%4.4g,%4.4g}\\n',i,...
208             cmap(i,1),cmap(i,2),cmap(i,3));
209     end

211     fprintf(fid, '\\begin{tabular}{|@{ }e|d{4}|d{4}|d{4}|\\hline\\n');
212     fprintf(fid, strcat('\\multicolumn{1}{|@{ }e|}{Region} & ',...
213         '\\multicolumn{1}{e|}{Slope} & \\multicolumn{1}{e|}{NormR} & ',...
214         '\\multicolumn{1}{e|}{LWidth} \\ \\ \\ \\hline \\n'));

216     for i=1:numel(fits)
217         fprintf(fid, strcat('\\setlength\\fboxsep{0mm}\\n',...
218             '\\fbox{\\raisebox{\\depth}%%\\n',...
219             '\\colorbox{c%d}{\\rule{0mm}{3pt}~~~~~}}}') ,...
220             '\\n & %5.4f & %5.4f & %5.4f\\ \\ \\ \\n'),...

```

```

221         i, fits{i}{1}, fits{i}{5}, fits{i}{6});
222     end
223
224 %% write region borders
225     fprintf(fid, strcat('\hline\n\multicolumn{1}{|@{e|}{Region} & ', ...
226         '\multicolumn{1}{e|}{Start} & \multicolumn{1}{e|}{End} & ', ...
227         '\multicolumn{1}{e|}{FitInt} \\\ \hline \n'));
228
229     for i=1:numel(fits)-1
230         fprintf(fid, strcat('\setlength\fbboxsep{0mm}\ ' , ...
231             'fbox{\raisebox{\depth}%%\n' , ...
232             '{\colorbox{c%d}{\rule{0mm}{3pt}-$\cdots$}}%%\n' , ...
233             '\colorbox{c%d}{\rule{0mm}{3pt}$\cdots$-\cdots$}}%%\n' , ...
234             '}} %\n & \multicolumn{1}{e|}{%4.1f} & \multicolumn{1}{e|}', ...
235             '{e|}{%4.1f} & \multicolumn{1}{e|}{%4.1f}\n\n') , ...
236             i, i, x(fits{i}{3}), x(fits{i}{4}), interx(i));
237     end
238     i = numel(fits);
239     fprintf(fid, strcat('\setlength\fbboxsep{0mm}\ ' , ...
240         'fbox{\raisebox{\depth}%%\n' , ...
241         '{\colorbox{c%d}{\rule{0mm}{3pt}-$\cdots$}}%%\n' , ...
242         '\colorbox{c%d}{\rule{0mm}{3pt}$\cdots$-\cdots$}}%%\n' , ...
243         '}} %\n & \multicolumn{1}{e|}{%4.1f} & \multicolumn{1}{e|}', ...
244         '{%4.1f} & \multicolumn{1}{e|}{-}\n\n') , ...
245         i, i, x(fits{i}{3}), x(fits{i}{4}));
246     if numel(fits)>1,
247         fprintf(fid, '\hline');
248     end
249     fprintf(fid, '\n\end{tabular} \\\n');
250 end
251
252 %% write legend to screen
253 if writelegendtoscreen
254     fprintf(fid, 'Slope \t\t NormR \t\t LWidth \n');
255     for i = 1:numel(fits)
256         fprintf(fid, '%5.4f \t %5.4f \n', fits{i}{1}, ...
257             fits{i}{5}, fits{i}{6});
258     end
259     fprintf(fid, 'End \t\t Start \t\t FitInt \n');
260     for i = 1:numel(fits)-1
261         fprintf(fid, '%4.1f \t\t %4.1f \n', x(fits{i}{4}), ...
262             x(fits{i+1}{3}), interx(i));
263     end
264 end
265
266 %% end linfitplot
267 end

```

Listing B.27: linfitplot.m

B.5 Random and Percolation Vessel Simulations

1. Perform Random Simulation Using a Uniform Probability Distribution B-42
2. Calculate one 3D Invasion Bond Percolation Cluster B-45
3. Process and Save Sections from the 3D Percolation Cluster B-47
4. Modify Exported Image Cell Vector Produced by the Bondsprocessing Script B-49
5. Merge Data from Different Data Files from each Simulation Before Exporting B-50
6. Export Script B-55

Script: randomsimulation – Perform Random Simulation of Histological Sections

Note: The section of this program that handles the acquisition of image analysis parameters, are used in similar scripts for the percolation simulation and the downscaled histological images. These scripts will not be included again.

```

1  % script randomsimulation.
2  % This script simulates immunohistological data by randomly placing
3  % vessels on an empty image. The various image analysis parameters are
4  % calculated for each of the simulated images.
5  clear
6  nfactor = 10;
7  t0 = cputime;
8  nimages = 200; % number of images at each vessel count

10 % k may be changed to a subset, e.g. 1:10, to divide
11 % workload among several computers, output is written to file
12 for k=1:50;
13     % initialize for each vessel count
14     vareastat      = [];
15     vformstat      = [];
16     vshapestat     = [];
17     fshapestat     = [];
18     ggnumbstat     = [];
19     ggblstat       = [];
20     ggbpnstat      = [];
21     ggnnstat       = [];
22     ggfnstat       = [];
23     emstnumbstat   = [];
24     emstblstat     = [];
25     emstbpnstat    = [];
26     emstnnstat     = [];
27     emstfnstat     = [];

29     for i = 1:nimages
30         tic
31         A = false(300,400);
32         I = 1: numel(A);
33         for tmp=1:nfactor*k
34             n = round((numel(I)-1)*rand)+1;
35             I(n) = []; % delete selected so it will not be chosen again.
36             A(n) = 1;
37         end

39         %% CUMULATIVE HISTOGRAM
40         ch = cumhist(A,[.1,.5,.9]);
41         ch10(i) = ch(1);
42         ch50(i) = ch(2);
43         ch90(i) = ch(3);

45         %% SSA
46         [vstat , ggstat , emststat ,GG,EMST] = randomsimSSA(A);
47         % format of stat vectors:
48         % vstat      = {areaparams , formparams , shapeparams };
49         % gg/emststat = {numbranch , blparams , bpnparams , ...
50         %                nparams , fnparams };

52         % add voronoi statistics
53         if ~isempty(vstat)
54             vareastat = [vareastat; single(vstat{1})];
55             vformstat = [vformstat; single(vstat{2})];
56             vshapestat = [vshapestat; single(vstat{3})];
57         end
58         % add gg statistics
59         ggnumbstat     = [ggnumbstat; single(ggstat{1})];
60         ggblstat       = [ggblstat; single(ggstat{2})];

```

```

61     ggbpnstat      = [ggbpnstat; single(ggstat{3})];
62     ggnnstat      = [ggnnstat; single(ggstat{4})];
63     ggfnstat      = [ggfnstat; single(ggstat{5})];
64     % add emst statistics
65     emstnumbstat  = [emstnumbstat; single(emststat{1})];
66     emstblstat    = [emstblstat; single(emststat{2})];
67     emstbpnstat   = [emstbpnstat; single(emststat{3})];
68     emstnnstat    = [emstnnstat; single(emststat{4})];
69     emstfnstat    = [emstfnstat; single(emststat{5})];

71     %% FRACTAL ANALYSIS
72     %% of image slide
73     sp = [5,.2,.15];
74     [P,s,X,Y] = getfracdim(A,'sand',50);
75     fits = reclinfit(log10(X),log10(Y),1,numel(X),sp);

77     %% sort fits
78     sortvec = zeros(numel(fits),1);
79     for n=1:numel(fits)
80         sortvec(n) = fits{n}{3};
81     end
82     [tmp sortind] = sort(sortvec);
83     fits = fits(sortind);

85     dim = cell2mat(fits{numel(fits)}(1));
86     Dim(i) = dim;
87     epsind = cell2mat(fits{numel(fits)}(3));
88     epsilon = X(epsind);
89     Eps(i) = epsilon;
90     omegaind = cell2mat(fits{numel(fits)}(4));
91     omega = X(omegaind);
92     Omega(i) = omega;
93     normrind = cell2mat(fits{numel(fits)}(4));
94     normr = X(normrind);
95     Normr(i) = normr;

97     %% of GG
98     sp = [5,.3,.25];
99     [P,s,X,Y] = getfracdim(GG,'sand',50);
100    fits = reclinfit(log10(X),log10(Y),1,numel(X),sp);

102    %% sort fits
103    sortvec = zeros(numel(fits),1);
104    for n=1:numel(fits)
105        sortvec(n) = fits{n}{3};
106    end
107    [tmp sortind] = sort(sortvec);
108    fits = fits(sortind);

110    dim = cell2mat(fits{numel(fits)}(1));
111    DimGG(i) = dim;
112    epsind = cell2mat(fits{numel(fits)}(3));
113    epsilon = X(epsind);
114    EpsGG(i) = epsilon;
115    omegaind = cell2mat(fits{numel(fits)}(4));
116    OmegaGG(i) = X(omegaind);
117    NormrGG(i) = cell2mat(fits{numel(fits)}(5));

119    %% of EMST
120    sp = [5,.3,.25];
121    [P,s,X,Y] = getfracdim(EMST,'sand',50);
122    fits = reclinfit(log10(X),log10(Y),1,numel(X),sp);

124    %% sort fits
125    sortvec = zeros(numel(fits),1);
126    for n=1:numel(fits)

```



```

127         sortvec(n) = fits{n}{3};
128         end
129         [tmp sortind] = sort(sortvec);
130         fits = fits(sortind);

132         dim = cell2mat(fits{numel(fits)}(1));
133         DimEMST(i) = dim;
134         epsind = cell2mat(fits{numel(fits)}(3));
135         epsilon = X(epsind);
136         EpsEMST(i) = epsilon;
137         omegaind = cell2mat(fits{numel(fits)}(4));
138         OmegaEMST(i) = X(omegaind);
139         NormrEMST(i) = cell2mat(fits{numel(fits)}(5));
140     end

142     %% MEAN AND STANDARD DEVIANCE OF THE PERMUTATIONS
143     %% AT CONSTANT VESSEL COUNT.

145     %% Cumulative Histogram Parameters
146     mch10(k) = mean(ch10);
147     stdch10(k) = std(ch10);
148     mch50(k) = mean(ch50);
149     stdch50(k) = std(ch50);
150     mch90(k) = mean(ch90);
151     stdch90(k) = std(ch90);

153     %% SSA
154     mvarea(k,:) = mean(vareastat,1);
155     stdvarea(k,:) = std(vareastat,1);
156     mvform(k,:) = mean(vformstat,1);
157     stdvform(k,:) = std(vformstat,1);
158     mvshape(k,:) = mean(vshapestat,1);
159     stdvshape(k,:) = std(vshapestat,1);

161     mggnumb(k,:) = mean(ggnumbstat);
162     stdggnumb(k,:) = std(ggnumbstat);
163     mggbl(k,:) = mean(ggblstat,1);
164     stdggbl(k,:) = std(ggblstat,1);
165     mggbpn(k,:) = mean(ggbpnstat,1);
166     stdggbpn(k,:) = std(ggbpnstat,1);
167     mgggn(k,:) = mean(gggnstat,1);
168     stdgggn(k,:) = std(gggnstat,1);
169     mggfn(k,:) = mean(ggfnstat,1);
170     stdggfn(k,:) = std(ggfnstat,1);

172     memstnumb(k,:) = mean(emstnumbstat);
173     stdemstnumb(k,:) = std(emstnumbstat);
174     memstbl(k,:) = mean(emstblstat,1);
175     stdemstbl(k,:) = std(emstblstat,1);
176     memstbpn(k,:) = mean(emstbpnstat,1);
177     stdemstbpn(k,:) = std(emstbpnstat,1);
178     memstnn(k,:) = mean(emstnnstat,1);
179     stdemstnn(k,:) = std(emstnnstat,1);
180     memstfn(k,:) = mean(emstfnstat,1);
181     stdemstfn(k,:) = std(emstfnstat,1);

183     %% Fractal Parameters
184     mDim(k) = mean(Dim);
185     stdDim(k) = std(Dim);
186     mEps(k) = mean(Eps);
187     stdEps(k) = std(Eps);
188     mOmega(k) = mean(Omega);
189     stdOmega(k) = std(Omega);

191     mDimGG(k) = mean(DimGG);
192     stdDimGG(k) = std(DimGG);

```

```

193     mEpsGG(k) = mean(EpsGG);
194     stdEpsGG(k) = std(EpsGG);
195     mOmegaGG(k) = mean(OmegaGG);
196     stdOmegaGG(k) = std(OmegaGG);
197     mNormrGG(k) = mean(NormrGG);
198     stdNormrGG(k) = std(NormGG);

200     mDimEMST(k) = mean(DimEMST);
201     stdDimEMST(k) = std(DimEMST);
202     mEpsEMST(k) = mean(EpsEMST);
203     stdEpsEMST(k) = std(EpsEMST);
204     mOmegaEMST(k) = mean(OmegaEMST);
205     stdOmegaEMST(k) = std(OmegaEMST);
206     mNormrEMST(k) = mean(NormrEMST);
207     stdNormrEMST(k) = std(NormEMST);

209 end
210 clear I
211 totaltime = cputime-t0;
212 save '../analyse/test/randomsim/lastrunworkspace.mat'
213 end

```

Listing B.28: randomsimulation.m

Script: bondpercolation3d – Calculate one 3D Invasion Bond Percolation Cluster

```

1  % Script producing one three dimensional invasion bond percolation
2  % cluster. The vertical bonds are stored in the matrix P3 at the
3  % end of the script.

5  % initialize
6  siz=[46 61 100];
7  A1 = rand(siz(1),siz(2)+1,siz(3)+1);
8  A2 = rand(siz(1)+1,siz(2),siz(3)+1);
9  A3 = rand(siz(1)+1,siz(2)+1,siz(3));
10 P1 = false(size(A1));% cluster
11 P2 = false(size(A2));
12 P3 = false(size(A3));
13 N1 = false(size(A1));% neighbour
14 N2 = false(size(A2));
15 N3 = false(size(A3));

18 % Inlet = single point at the middle of bottom layer
19 N3(round(siz(1)./2),round(siz(2)./2),1)=1;

21 counter = 0;
22 while ~any(P3(:, :, siz(3)))
23     counter = counter +1;

25     % Find weakest bond for each direction; x, y and z.
26     I1 = find(N1);
27     [min1, ind1] = min(A1(I1));
28     min1 = min([min1, inf]);
29     [m1, n1, p1] = ind2sub(size(A1), I1(ind1));

31     I2 = find(N2);
32     [min2, ind2] = min(A2(I2));
33     min2 = min([min2, inf]);
34     [m2, n2, p2] = ind2sub(size(A2), I2(ind2));

36     I3 = find(N3);
37     [min3, ind3] = min(A3(I3));
38     min3 = min([min3, inf]);

```

```

39     [m3,n3,p3] = ind2sub(size(A3),I3(ind3));
40
41     % Select the weakest bond, comparing the three directions
42     [minval,minind] = min([min1,min2,min3]);
43
44     % Add bond to the cluster and update neighbours list
45     if minind == 1
46         P1(m1,n1,p1) = 1;
47         if m1<siz(1), N1(m1+1,n1,p1) = 1; end
48         if m1>1, N1(m1-1,n1,p1) = 1; end
49         if n1 <=siz(2)
50             N2(m1,n1,p1) = 1;
51             N2(m1+1,n1,p1) = 1;
52         end
53         if n1 > 1
54             N2(m1,n1-1,p1) = 1;
55             N2(m1+1,n1-1,p1) = 1;
56         end
57         if p1 <=siz(3)
58             N3(m1,n1,p1) = 1;
59             N3(m1+1,n1,p1) = 1;
60         end
61         if p1 > 1
62             N3(m1,n1,p1-1) = 1;
63             N3(m1+1,n1,p1-1) = 1;
64         end
65     end
66
67     if minind == 2
68         P2(m2,n2,p2) = 1;
69         if n2<siz(2), N2(m2,n2+1,p2) = 1; end
70         if n2>1, N2(m2,n2-1,p2) = 1; end
71         if m2 <=siz(1)
72             N1(m2,n2,p2) = 1;
73             N1(m2,n2+1,p2) = 1;
74         end
75         if m2 > 1
76             N1(m2-1,n2,p2) = 1;
77             N1(m2-1,n2+1,p2) = 1;
78         end
79         if p2 <=siz(3)
80             N3(m2,n2,p2) = 1;
81             N3(m2,n2+1,p2) = 1;
82         end
83         if p2 > 1
84             N3(m2,n2,p2-1) = 1;
85             N3(m2,n2+1,p2-1) = 1;
86         end
87     end
88
89     if minind == 3
90         P3(m3,n3,p3) = 1;
91         if p3<siz(2), N3(m3,n3,p3+1) = 1; end
92         if p3>1, N3(m3,n3,p3-1) = 1; end
93         if m3 <=siz(1)
94             N1(m3,n3,p3) = 1;
95             N1(m3,n3,p3+1) = 1;
96         end
97         if m3 > 1
98             N1(m3-1,n3,p3) = 1;
99             N1(m3-1,n3,p3+1) = 1;
100        end
101        if n3 <=siz(2)
102            N2(m3,n3,p3) = 1;
103            N2(m3,n3,p3+1) = 1;
104        end

```



```

31  %% Without expansion and randomisation
32  % count number of vessels and add to stack
33  for n=1:size(P3,3)
34      currentsum = sum(sum(P3(:, :, n)));
35      if currentsum <= numel(Imstack) && ~isempty(Imstack{currentsum})
36          Imstack(currentsum) = {[Imstack(currentsum),{P3(:, :, n)}]};
37      else
38          Imstack(currentsum) = {P3(:, :, n)};
39      end
40  end

42  % write to file
43  for n=1:length(Imstack)
44      if ~isempty(Imstack{n})
45          filename = [folder, '\originalstack\vc', num2str(n), '.mat'];
46          if exist(filename, 'file')==2,
47              stack = struct2cell(load(filename));
48              stack = stack{1};
49          else
50              stack = [];
51          end
52          for i = 1:numel(Imstack(n))
53              stack = [stack, Imstack(n)];
54          end
55          save(filename, 'stack')
56      end
57  end

59  %% With Expansion and randomization
60  fprintf(1, '\r\r\r ')
61  %% Plotting Section
62  % resetting Imstack to empty cell array
63  Imstack = cell(0);
64  for n = 1:size(P3,3)
65      Im0 = P3(:, :, n);
66      % removing border 2x2 pixels
67      Im0 = Im0(3:size(Im0,1)-2, 3:size(Im0,2)-2);

69      % increase resolution of image by adding empty rows and columns
70      Im2 = Im0;
71      expandf = 7;
72      for i = 1:1
73          Im = Im2;
74          Im2 = [Im; false(size(Im,1)*(expandf-1), size(Im,2))];
75          Im2 = reshape(Im2, size(Im,1), size(Im,2)*expandf);
76          Im2 = rot90(Im2);
77          Im2 = [Im2; false(size(Im2,1)*(expandf-1), size(Im2,2))];
78          Im2 = reshape(Im2, size(Im,1)*expandf, size(Im,2)*expandf);
79          Im2 = rot90(Im2, -1);
80      end
81      % randomizing within 7x7 square
82      rdx = (ceil(rand(sum(sum(Im0)), 1)*(expandf))-ceil(expandf/2);
83      rdy = (ceil(rand(sum(sum(Im0)), 1)*(expandf))-ceil(expandf/2);
84      [Y,X] = ind2sub(size(Im2), find(Im2));
85      X = X + rdx; X(X<1) = 1; X(X>size(Im2,2)) = size(Im2,2);
86      Y = Y + rdy; Y(Y<1) = 1; Y(Y>size(Im2,1)) = size(Im2,1);
87      Im = false(size(Im2));
88      Im(sub2ind(size(Im2), Y, X)) = 1;

90      % adding to imagestack
91      currentsum = sum(sum(Im));
92      if currentsum > 0
93          if currentsum <= numel(Imstack) &&...
94              ~isempty(Imstack{currentsum})
95              Imstack(currentsum) = {[Imstack(currentsum),{Im}]};

```

```

96         else
97             Imstack(currentsum) = {Im};
98         end
99     end
100 end
101 %% Export
102 for n=1:length(Imstack)
103     if ~isempty(Imstack{n})
104         filename = [folder, '\randomizedstack\vc', num2str(n), '.mat'];
105         if exist(filename, 'file')==2,
106             stack = struct2cell(load(filename));
107             stack = stack{1};
108         else
109             stack = [];
110         end
111         for i = 1:numel(Imstack(n))
112             stack = [stack, Imstack(n)];
113         end
114         save(filename, 'stack')
115     end
116 end
118 end % while

```

Listing B.30: bondslideprocessing.m

Function: sortslides – Further Process Saved Image Slide Arrays From the Percolation Clusters by Removing Nested Cell Array Entries

Note: After this step the images are analysed, code is similar to that of the randomsimulation at page B-42 (in appendix B).

```

1  function stack2 = sortslides()
2  % Function reading cell arrays containing images of a specific number of
3  % vessels, and sort them to become one dimensional. Percolation cluster
4  % containing several sections with the same number of vessels have produced
5  % a cell array of varying number of images at each entry.
6  % Furthermore, the vessel counts the number of images at each vessel count
7  % and stores the result in a file.
8
9  for vc=1:500
10     filename = ['randomizedstack/vc', num2str(vc), '.mat'];
11     % check if file exists
12     if exist(filename, 'file')==2
13         stack = struct2cell(load(filename));
14         stack = stack{:}(:);
15         stack = restack(stack, {});
16         save(filename, 'stack')
17         slidec(vc) = numel(stack);
18     else
19         slidec(vc) = 0;
20     end
21     vesselc(vc) = vc;
22 end
23 logfile = 'numberofslides.mat';
24 save(logfile, 'vesselc', 'slidec');
25 end
26
27 function stack2 = restack(stack, stack2)
28     for i=1:numel(stack)
29         if islogical(stack{i});
30             stack2{numel(stack2)+1} = stack{i};
31         else
32

```

```

33         stack2 = restack(stack{i},stack2);
34     end
35 end
36 end

```

Listing B.31: sortslides.m

Script: randomsimstitch – Merge the Data in the Different Stored Vessel Count Files for the Random and the Percolation Simulation

```

1  % script gathering the image analysis parameters from each vessel count
2  % file and add them to a single vector for plotting. The percolation
3  % simulation has one file per vessel count, the random simulation has five
4  % different files containing intervals of different vessel counts.

6  if ~exist('tmch10','var')
7  %% declare variables
8  % fractal
9  tmDim      = [];
10 tstdDim   = [];
11 tmEps     = [];
12 tstdEps   = [];
13 tmOmega   = [];
14 tstdOmega = [];

16 tmDimGG   = [];
17 tstdDimGG = [];
18 tmEpsGG   = [];
19 tstdEpsGG = [];
20 tmOmegaGG = [];
21 tstdOmegaGG = [];

23 tmDimEMST = [];
24 tstdDimEMST = [];
25 tmEpsEMST = [];
26 tstdEpsEMST = [];
27 tmOmegaEMST = [];
28 tstdOmegaEMST = [];

30 % cumhist
31 tmch10    = [];
32 tstdch10  = [];
33 tmch50    = [];
34 tstdch50  = [];
35 tmch90    = [];
36 tstdch90  = [];

38 % SSA
39 tmvarea   = [];
40 tstdvarea = [];
41 tmvform   = [];
42 tstdvform = [];
43 tmvshape  = [];
44 tstdvshape = [];

46 tmggnumb  = [];
47 tstdggnumb = [];
48 tmggbl    = [];
49 tstdggbl  = [];
50 tmggbpn   = [];
51 tstdggbpn = [];
52 tmggnn    = [];
53 tstdggnn  = [];
54 tmggfn    = [];

```

```

55     tstdggfn      = [];

57     tmemstnumb   = [];
58     tstdemstnumb = [];
59     tmemstbl     = [];
60     tstdemstbl   = [];
61     tmemstbpn    = [];
62     tstdemstbpn  = [];
63     tmemstnn     = [];
64     tstdemstnn   = [];
65     tmemstfn     = [];
66     tstdemstfn   = [];
67 end

69 if isequal(folder, 'percsim'),
70     readfile = '47x62x100/results/percsimvc';
71     numparts = 9;
72     partstep = 50;
73     parts = 1:9;
74     froms = partstep*(parts-1)+1; froms(1) = 10;
75     tos = partstep*(parts);     tos(numel(tos)) = 479;
76     load('47x62x100/numberofslides.mat')
77     l = ~(slidec >= 10);
78     l = l(froms(1):tos(numel(tos)));

80 elseif isequal(folder, 'randomsim'),
81     readfile = 'randomsim300x400/lastrunworkspace';
82     numparts = 5;
83     partstep = 10;
84     parts = 1:5;
85     froms = partstep*(parts-1)+1;
86     tos = partstep*(parts);
87     l = false(size(parts));
88 end

90 for part=1:numel(parts)
91     from = froms(part);
92     to = tos(part);

94     %% -----
95     %% Remove zero rows, i.e. vessel counts without any images
96     if isequal(folder, 'percsim')
97         filename = [readfile, num2str(to), 'ws.mat'];
98     elseif isequal(folder, 'randomsim')
99         filename = [readfile, num2str(from), 'to', num2str(to), '.mat'];
100    end

102    load(filename)
103    %% Fractal Parameters
104    mDim = mDim(from:to);
105    stdDim = stdDim(from:to);
106    mEps = mEps(from:to);
107    stdEps = stdEps(from:to);
108    mOmega = mOmega(from:to);
109    stdOmega = stdOmega(from:to);

111    mDimGG = mDimGG(from:to);
112    stdDimGG = stdDimGG(from:to);
113    mEpsGG = mEpsGG(from:to);
114    stdEpsGG = stdEpsGG(from:to);
115    mOmegaGG = mOmegaGG(from:to);
116    stdOmegaGG = stdOmegaGG(from:to);

118    mDimEMST = mDimEMST(from:to);
119    stdDimEMST = stdDimEMST(from:to);
120    mEpsEMST = mEpsEMST(from:to);

```



```

121     stdEpsEMST      = stdEpsEMST ( from : to );
122     mOmegaEMST     = mOmegaEMST ( from : to );
123     stdOmegaEMST   = stdOmegaEMST ( from : to );

127     %% Cumulative Histogram Parameters;
128     mch10          = mch10 ( from : to );
129     stdch10       = stdch10 ( from : to );
130     mch50         = mch50 ( from : to );
131     stdch50      = stdch50 ( from : to );
132     mch90        = mch90 ( from : to );
133     stdch90      = stdch90 ( from : to );

135     %% SSA=
136     mvarea        = mvarea ( from : to , : );
137     stdvarea     = stdvarea ( from : to , : );
138     mvform       = mvform ( from : to , : );
139     stdvform     = stdvform ( from : to , : );
140     mvshape      = mvshape ( from : to , : );
141     stdvshape    = stdvshape ( from : to , : );

143     mggnumb      = mggnumb ( from : to );
144     stdggnumb    = stdggnumb ( from : to );
145     mggbl       = mggbl ( from : to , : );
146     stdggbl     = stdggbl ( from : to , : );
147     mggbpn      = mggbpn ( from : to , : );
148     stdggbpn    = stdggbpn ( from : to , : );
149     mggnn       = mggnn ( from : to , : );
150     stdggnn     = stdggnn ( from : to , : );
151     mggfn       = mggfn ( from : to , : );
152     stdggfn     = stdggfn ( from : to , : );

154     memstnumb   = memstnumb ( from : to );
155     stdemstnumb = stdemstnumb ( from : to );
156     memstbl     = memstbl ( from : to , : );
157     stdemstbl  = stdemstbl ( from : to , : );
158     memstbpn   = memstbpn ( from : to , : );
159     stdemstbpn = stdemstbpn ( from : to , : );
160     memstnn    = memstnn ( from : to , : );
161     stdemstnn  = stdemstnn ( from : to , : );
162     memstfn    = memstfn ( from : to , : );
163     stdemstfn  = stdemstfn ( from : to , : );

165     %% -----
166     %% Stitch them together

168     %% Fractal Parameters
169     tmDim        = [ tmDim , mDim ];
170     tstDim      = [ tstDim , stdDim ];
171     tmEps       = [ tmEps , mEps ];
172     tstEps     = [ tstEps , stdEps ];
173     tmOmega     = [ tmOmega , mOmega ];
174     tstOmega    = [ tstOmega , stdOmega ];

176     tmDimGG     = [ tmDimGG , mDimGG ];
177     tstDimGG    = [ tstDimGG , stdDimGG ];
178     tmEpsGG     = [ tmEpsGG , mEpsGG ];
179     tstEpsGG    = [ tstEpsGG , stdEpsGG ];
180     tmOmegaGG   = [ tmOmegaGG , mOmegaGG ];
181     tstOmegaGG  = [ tstOmegaGG , stdOmegaGG ];

183     tmDimEMST   = [ tmDimEMST , mDimEMST ];
184     tstDimEMST  = [ tstDimEMST , stdDimEMST ];
185     tmEpsEMST   = [ tmEpsEMST , mEpsEMST ];
186     tstEpsEMST  = [ tstEpsEMST , stdEpsEMST ];

```

```

187     tmOmegaEMST = [tmOmegaEMST,mOmegaEMST];
188     tstdOmegaEMST = [tstdOmegaEMST ,stdOmegaEMST];

190     %% Cumulative Histogram Parameters
191     tmch10      = [tmch10 ,mch10];
192     tstdch10   = [tstdch10 ,stdch10];
193     tmch50     = [tmch50 ,mch50];
194     tstdch50   = [tstdch50 ,stdch50];
195     tmch90     = [tmch90 ,mch90];
196     tstdch90   = [tstdch90 ,stdch90];

198     %% SSA
199     tmvarea     = [tmvarea ;mvarea];
200     tstdvarea  = [tstdvarea ;stdvarea];
201     tmvform    = [tmvform ;mvform];
202     tstdvform  = [tstdvform ;stdvform];
203     tmvshape   = [tmvshape ;mvshape];
204     tstdvshape = [tstdvshape ;stdvshape];

206     tmggnumb   = [tmggnumb ;mggnumb];
207     tstdggnumb = [tstdggnumb ;stdggnumb];
208     tmggbl     = [tmggbl ;mggbl];
209     tstdggbl   = [tstdggbl ;stdggbl];
210     tmggbpn    = [tmggbpn ;mggbpn];
211     tstdggbpn  = [tstdggbpn ;stdggbpn];
212     tmggnn     = [tmggnn ;mggnn];
213     tstdggnn   = [tstdggnn ;stdggnn];
214     tmggfn     = [tmggfn ;mggfn];
215     tstdggfn   = [tstdggfn ;stdggfn];

217     tmemstnumb = [tmemstnumb ;memstnumb];
218     tstdemstnumb = [tstdemstnumb ;stdemstnumb];
219     tmemstbl   = [tmemstbl ;memstbl];
220     tstdemstbl = [tstdemstbl ;stdemstbl];
221     tmemstbpn  = [tmemstbpn ;memstbpn];
222     tstdemstbpn = [tstdemstbpn ;stdemstbpn];
223     tmemstnn   = [tmemstnn ;memstnn];
224     tstdemstnn = [tstdemstnn ;stdemstnn];
225     tmemstfn   = [tmemstfn ;memstfn];
226     tstdemstfn = [tstdemstfn ;stdemstfn];

228     end

230     %% Replace non slide indices with nan (from 0),
231     % this so they will not be plotted
232     tmDim(1) = nan;
233     tstdDim(1) = nan;
234     tmEps(1) = nan;
235     tstdEps(1) = nan;
236     tmOmega(1) = nan;
237     tstdOmega(1) = nan;

239     tmDimGG(1) = nan;
240     tstdDimGG(1) = nan;
241     tmEpsGG(1) = nan;
242     tstdEpsGG(1) = nan;
243     tmOmegaGG(1) = nan;
244     tstdOmegaGG(1) = nan;

246     tmDimEMST(1) = nan;
247     tstdDimEMST(1) = nan;
248     tmEpsEMST(1) = nan;
249     tstdEpsEMST(1) = nan;
250     tmOmegaEMST(1) = nan;
251     tstdOmegaEMST(1) = nan;

```

```

253 tmch10(1) = nan;
254 tstdch10(1) = nan;
255 tmch50(1) = nan;
256 tstdch50(1) = nan;
257 tmch90(1) = nan;
258 tstdch90(1) = nan;

260 %% SSA
261 tmvarea(1,:) = nan;
262 tstdvarea(1,:) = nan;
263 tmvform(1,:) = nan;
264 tstdvform(1,:) = nan;
265 tmvshape(1,:) = nan;
266 tstdvshape(1,:) = nan;

268 tmggnumb(1,:) = nan;
269 tstdggnumb(1,:) = nan;
270 tmggbl(1,:) = nan;
271 tstdggbl(1,:) = nan;
272 tmggbpn(1,:) = nan;
273 tstdggbpn(1,:) = nan;
274 tmggnn(1,:) = nan;
275 tstdggnn(1,:) = nan;
276 tmggfn(1,:) = nan;
277 tstdggfn(1,:) = nan;

279 tmemstnumb(1,:) = nan;
280 tstdemstnumb(1,:) = nan;
281 tmemstbl(1,:) = nan;
282 tstdemstbl(1,:) = nan;
283 tmemstbpn(1,:) = nan;
284 tstdemstbpn(1,:) = nan;
285 tmemstnn(1,:) = nan;
286 tstdemstnn(1,:) = nan;
287 tmemstfn(1,:) = nan;
288 tstdemstfn(1,:) = nan;

290 %% Rename all variables
291 % fractal analysis
292 mDim = tmDim;
293 stdDim = tstdDim;
294 mEps = tmEps;
295 stdEps = tstdEps;
296 mOmega = tmOmega;
297 stdOmega = tstdOmega;

299 mDimGG = tmDimGG;
300 stdDimGG = tstdDimGG;
301 mEpsGG = tmEpsGG;
302 stdEpsGG = tstdEpsGG;
303 mOmegaGG = tmOmegaGG;
304 stdOmegaGG = tstdOmegaGG;

306 mDimEMST = tmDimEMST;
307 stdDimEMST = tstdDimEMST;
308 mEpsEMST = tmEpsEMST;
309 stdEpsEMST = tstdEpsEMST;
310 mOmegaEMST = tmOmegaEMST;
311 stdOmegaEMST = tstdOmegaEMST;

313 %% cumulative histogram
314 mch10 = tmch10;
315 stdch10 = tstdch10;
316 mch50 = tmch50;
317 stdch50 = tstdch50;
318 mch90 = tmch90;

```

```

319 stdch90      =  tstch90;

321 %% SSA          %% SSA
322 mvarea       =  tmvarea;
323 stdvarea     =  tstdvarea;
324 mvform       =  tmvform;
325 stdvform     =  tstdvform;
326 mvshape      =  tmvshape;
327 stdvshape    =  tstdvshape;

329 mggnumb      =  tmggnumb;
330 stdggnumb    =  tstdggnumb;
331 mggbl        =  tmggbl;
332 stdggbl      =  tstdggbl;
333 mggbpn       =  tmggbpn;
334 stdggbpn     =  tstdggbpn;
335 mggnn        =  tmggnn;
336 stdggnn      =  tstdggnn;
337 mggfn        =  tmggfn;
338 stdggfn      =  tstdggfn;

340 memstnumb    =  tmemstnumb;
341 stdemstnumb =  tstdemstnumb;
342 memstbl      =  tmemstbl;
343 stdemstbl    =  tstdemstbl;
344 memstbpn     =  tmemstbpn;
345 stdemstbpn  =  tstdemstbpn;
346 memstnn      =  tmemstnn;
347 stdemstnn    =  tstdemstnn;
348 memstfn      =  tmemstfn;
349 stdemstfn    =  tstdemstfn;

```

Listing B.32: randomsimstitch.m

Script: plotcasesandperccomparison – Export Graphs of the Image Analysis Parameters of the Percolation Simulation and the Data Points of the Histological Sections

Note: A similar script (not included) exports the graphs from the random simulation.

```

1  % Script exporting the results from the downscaled histological
2  % data and the percolation script.
3  clear
4  folder = 'percsim';
5  % call randomsimstitch script
6  randomsimstitch
7  % load the downscaled histological image analysis data
8  load '../analyse/test/randomsim/caseparams2.mat'
9  exportflag = 1;
10 fullresfraccomp = 1;

12 %% Export Section
13 titlefs = 22;
14 axisfs  = 18;

16 folder = [folder, 'comp'];
17 linewidthnum1 = 3;
18 linewidthnum2 = 2;

20 if exportflag

22     %% Number of sections
23     figure(1), clf, plot(vesselc, slidec),
24     set(gcf, 'windowstyle', 'docked'),
25     title('Number of Generated Images', 'fontsize', titlefs)

```

```

26 set(gca, 'fontsize', axisfs, 'xlim', [0,500], 'box', 'off', 'ylim', [0 80])
27 ylabel('Number of Images'), xlabel('Number of Vessels')
28 print(gcf, '-depsc2', ['../analyse/test/', folder, '/numberofslides.eps'])

30 %% Fractal Dimension
31 %% FD of image
32 figure(2), clf, errorbar(froms(1):tos(numel(tos)), mDim, stdDim)
33 hold on, plot(froms(1):tos(numel(tos)), mDim, 'k'); hold off
34 set(gcf, 'windowstyle', 'docked'),
35 title('Fractal Dimensions at Large Sandbox Sizes', 'fontsize', titlefs)
36 set(gca, 'fontsize', axisfs, 'xlim', [0,500], 'box', 'off', 'ylim', [75,2])
37 ylabel('Sandbox Dimension'), xlabel('Number of Vessels')
38 if fullresfraccomp
39     hold on, plot(numvessels, Dim2, 'kx', 'linewidth', linewidthnum1, ...
40         'markersize', 16, 'markeredgecolor', 'k', 'markerfacecolor', ...
41         'none'), hold off
42 end
43 hold on, plot(numvessels, Dim, 'ro', 'linewidth', 2, ...
44     'markeredgecolor', 'r', 'markerfacecolor', 'r'), hold off
45 print(gcf, '-depsc2', ['../analyse/test/', folder, '/fractaldim.eps'])
46 figure(1), clf, plot(froms(1):tos(numel(tos)), stdDim), ylim([0 .1])
47 set(gca, 'fontsize', axisfs, 'xlim', [0,500], 'box', 'off')
48 title('St.Dev of Dimension', 'fontsize', titlefs)
49 ylabel('Standard Deviation'), xlabel('Number of Vessels')
50 print(gcf, '-depsc2', ['../analyse/test/', folder, '/fractalstd.eps'])

52 figure(1), clf
53 hold on
54 errorbar(froms(1):tos(numel(tos)), mOmega, stdOmega, 'k')
55 plot(froms(1):tos(numel(tos)), mOmega, 'b')
56 errorbar(froms(1):tos(numel(tos)), mEps, stdEps)
57 plot(froms(1):tos(numel(tos)), mEps, 'k')
58 set(gcf, 'windowstyle', 'docked'), title('Upper Fractal Region', 'fontsize', titlefs)
59 set(gca, 'fontsize', axisfs, 'xlim', [0,500], 'box', 'off')
60 legend('End Diameter', 'Start Diameter', 'Location', 'East')
61 ylabel('Sandbox Diameter'), xlabel('Number of Vessels')
62 if fullresfraccomp
63     hold on, plot(numvessels, Eps2./5.16, 'kx', 'linewidth', linewidthnum1, ...
64         'markersize', 22, 'markeredgecolor', 'k', 'markerfacecolor', 'none'), hold off
65     hold on, plot(numvessels, Omega2./5.16, 'bx', 'linewidth', linewidthnum1, ...
66         'markersize', 22, 'markeredgecolor', 'b', 'markerfacecolor', 'none'), hold off
67 end
68 hold on, plot(numvessels, Eps, 'ro', 'linewidth', 2, ...
69     'markersize', 10, 'markeredgecolor', 'r', 'markerfacecolor', 'r'), hold off

71 hold on, plot(numvessels, Omega, 'ro', 'linewidth', 2, ...
72     'markersize', 10, 'markeredgecolor', 'r', 'markerfacecolor', 'r'), hold off

74 print(gcf, '-depsc2', ['../analyse/test/', folder, '/fractalregion.eps'])
75 hold off

77 %% Script continues to plot graphs for all parameters, in total 918 lines.

79 close all
80 end

```

Listing B.33: plotcasesandperccomparison.m