TOWARD A COMPUTATIONAL SOLUTION TO THE INVERSE PROBLEM OF HOW HYPOXIA ARISES IN METABOLICALLY HETEROGENEOUS CANCER CELL POPULATIONS

by

ANDREW SUNDSTROM

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
Program in Computational Biology
New York University

September 2013

Bhubaneswar Mishra

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

DEDICATION

For Tiffany & Wren.

ACKNOWLEDGEMENTS

Let me begin by thanking my advisor, Bud Mishra, for encouraging me to "ask for forgiveness, not for permission" throughout the research process. One of the first things Bud impressed upon me was how computation can bring scale and rigor to biology, a principle from which I've derived a deeper understanding of both fields. (I would add that computation can also bring mechanism to biology, but this is arguable; that biology can bring humility to computation is beyond dispute.) Bud is one of the few genuine polymaths I've known, and he's frequently astonished me with the deep connections he perceives between seemingly dissociated phenomena and fields of research. We first spoke in his office in the fall of 2006. During that wide-ranging conversation it dawned on me that I must study computational biology under his supervision. This began in earnest that following summer, with our first collaborative publication with researchers at UCLA, and has continued unbroken to the present. I thank him for seven years (and counting) of stimulating inquiry, in an environment of honesty, self-reliance, and risk-taking that he has cultivated in his laboratory.

Next, I must thank my co-advisor, Dafna Bar-Sagi, for teaching me how to think like a biologist, and how to respect the staggering, defiant complexity of biological systems, which don't cleave to our beautiful ideas. When I first attended one of her lab meetings in 2009, I was struck by the candor and the quality of criticism that lab members offered each other in relation to their research—a crucible for asking questions and testing hypotheses. During the period that she supervised my depth

qualifying exam, I grew convinced that she should be my co-advisor. Over the past three years, conducting research in her lab, I've developed a keen appreciation for the rigors of laboratory bench work, and I treasure my exposure to an entirely separate intellectual tradition than I'd known prior to that. To bridge the disciplines of computation and biology, one must train an agile mind, and take a humble posture. I thank Dafna for helping me do both.

For their time, attention, and guidance during dissertation review, I thank the other members of my thesis committee, Leslie Greengard, Ravi Iyengar, and Ernest Davis.

I want to acknowledge past and present members of Dafna's lab who have helped me. I thank Laura Melis for training me to conduct tissue culture experiments and MTT assays. Only a saint could possess the vast reservoir of patience she had for these fumbling hands. I thank Rengin Soydaner-Azeloglu, for teaching me how to extract, prepare, and section a mouse pancreas. I thank Elda Grabocka for teaching me about hypoxia in a cancer setting, introducing me to pimonidazole staining, and for providing me with the tumor section histology slides I used to characterize hypoxia in this dissertation. My many conversations with Cosimo Commisso have enlightened me about genetics and biochemistry in general and cancer cell metabolism in particular. He has also taught me about biology as a career, and generously given his time to advise me on mine, for which I am grateful. I hold Elda and Cosimo up as model experimental biologists.

I want to acknowledge past and present members of Bud's lab who have helped me. I thank Ilya Korsunsky and Justin Jee for their efforts to inject life into the lab culture, including holding weekly student-only lab meetings that were especially productive and useful to me. And injecting culture into the lab life, like when Justin had his piano delivered to the lab and turned Ravel's *Toccata* from *Le Tombeau de Couperin* into

my writing anthem, and when Ilya graced many a late night in the lab with Latin ballroom dance. I found collaborating with Justin on the problem of the evolution of the genetic code to be a real thrill; I can only hope we have the opportunity to work together again. I thank Andreas Witzel for educating me about logic and game theory. His intense, spread-spectrum intellect, and sense of humor, made the lab an especially good place to be for the three years he inhabited the office next door. Lastly, I want to thank Samantha Kleinberg, whose passion for her research made me want to drop what I was doing and study temporal causality. She brought a vigorous and playful sense of inquiry to the lab, and she always took time to help me, including giving me black-belt time management advice—if only I'd listened!

Outside of the two labs, there are several I wish to acknowledge. First, I thank Ed Schonberg, colleague and friend, who always lent me an ear, and a squash racquet, for my tales of woe and triumph, and who consistently gave me sound advice for life inside and outside of the academy. I thank Davi Geiger for conversations related to image processing in my research, starting with my masters thesis analysis of AFM images, and continuing most recently with my gradient analysis of histology images. He impressed upon me the need to be methodical, to grow organically with the problem, learning what each algorithm means and where its limit lie, and to develop new mathematics and new algorithms when necessary. My dear friend Alex Botta, a Courant computer science doctoral veteran, always made himself available for my questions, and to offer sage advice about the Ph.D. process. Alex is another polymath of unfathomable wit, who has shaped me in profound ways. Our thirteen-year friendship testifies to this; if he hadn't brought me to the NYU graduate computer science open house in 2003, I might not be writing these words. He taught me early on that "Art needs constraint." Ever since, I've sought to identify the best constraints within

which to perform my creative work. My Aussie brother, Damian Conway, provided a steady stream of long-distance, and occasional short-range, encouragement. I met Damian in 2000, and over the course of his teaching me to program at a level I'd not dreamt was possible, I realized I was becoming friends with the Mozart of computer programming and programming language design. His singular creativity in the art is my inspiration to recurrently surpass myself at translating thought into code. It is my failing that this dissertation is implemented in Matlab and not Perl6. Lastly, I want to thank David Mordecai, whose conversations over the last three years, ranging from game theory to Bayesian statistical learning, from economics to sociology, from sentiment analysis to spatiotemporal mapping, have proven instrumental not only in stimulating my current research direction, but in exploring my next one. David coached me in invaluable ways throughout the dissertation writing process, helping me get perspective, formulate scope, and manage my time during "essay crisis."

Let me conclude by acknowledging my family. I thank Tom and Martha Mills for their unfailing support and faith in me, not only for the duration of this five-year enterprise, but for the 25 years I've known them. They have instilled in me a sense of honesty, generosity, fairness, and respect for one's work. In their example I've come to realize that nothing can approach or replace the love of family, and the personal strength that issues from it. I am blessed to call them my second parents. Denna Reilly, my sister and friend, continually encouraged me through the doctoral process. As the end approached, she would repeatedly exhort, "Stay on target! Stay on target!" I thank my mom, Martha Sundstrom, for giving me her extraordinary emotional appetite for the world, and her Sicilian characteristics through which that's inflected. Her love and faith in me are unbounded. I thank my dad, Pete Sundstrom, for instilling in me from as early as I can remember the value of intellectual curiosity, and

for cultivating my appreciation of antiquity and the long human conversations that form history, art, philosophy, and literature—emanating from Goethe's understanding that "He who cannot draw on 3,000 years is living from hand to mouth." Most importantly, he taught me how to approach the world with a sense of awe. Little wonder that he was the first to illuminate for me the beauty, and sanctity, of living things, a sensibility I carry straight into this thesis. His love and pride I can see from here. I thank Tiffany Mills, my wife and best friend, my boon companion for more than a quarter-century, for her love and support. She has always believed in me and I couldn't have made this happen without her. Finally, I thank my son, Wren Sundstrom. Every day that I love him and experience him unfurl into the world, I am gobsmacked and humbled. Before he was born, I was half of a man. It is to Tiffany and Wren, my two loves, whom I dedicate this dissertation.

ABSTRACT

As a tumor grows, it rapidly outstrips its blood supply, leaving portions of tumor that undergo hypoxia. Hypoxia is strongly correlated with poor prognosis as it renders tumors less responsive to chemotherapy and radiotherapy. During hypoxia, HIFs up-regulate production of glycolysis enzymes and VEGF, thereby promoting metabolic heterogeneity and angiogenesis, and proving to be directly instrumental in tumor progression. Prolonged hypoxia leads to necrosis, which in turn activates inflammatory responses that produce cytokines that stimulate tumor growth. Hypoxic tumor cells interact with macrophages and fibroblasts, both involved with inflammatory processes tied to tumor progression. So it is of clinical and theoretical significance to understand: Under what conditions does hypoxia arise in a heterogeneous cell population? Our aim is to transform this biological origins problem into a computational inverse problem, and then attack it using approaches from computer science. First, we develop a minimal, stochastic, spatiotemporal simulation of large heterogeneous cell populations interacting in three dimensions. The simulation can manifest stable localized regions of hypoxia. Second, we employ and develop a variety of algorithms to analyze histological images of hypoxia in xenographed colorectal tumors, and extract features to construct a spatiotemporal logical characterization of hypoxia. We also consider characterizing hypoxia by a linear regression functional learning mechanism that yields a similarity score. Third, we employ a Bayesian statistical model checking algorithm that can determine, over some bounded number of simulation executions, whether hypoxia is likely to emerge under some fixed set of simulation

parameters, and some fixed logical or functional description of hypoxia. Driving the model checking process is one of three adaptive Monte Carlo sampling algorithms we developed to explore the high dimensional space of simulation initial conditions and operational parameters. Taken together, these three system components formulate a novel approach to the inverse problem above, and constitute a design for a tool that can be placed into the hands of experimentalists, for testing hypotheses based upon known parameter values or ones the tool might discover. In principle, this design can be generalized to other biological phenomena involving large heterogeneous populations of interacting cells.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICIES

1

INTRODUCTION

1.1 PROBLEM STATEMENT

At the tissue level, what are the origins of localized hypoxia within tumors?

1.2 BIOLOGY BACKGROUND & LITERATURE REVIEW

1.2.1 *Hypoxia*

As a tumor grows, it rapidly outstrips its blood supply. High proliferation causes high cell density that overtaxes local oxygen supply. This leaves portions of the tumor with an oxygen concentration significantly lower than in healthy tissues. This stress condition is tumor hypoxia. Hypoxia is strongly correlated with poor prognosis as it renders tumors less responsive to chemotherapy and radiotherapy [65, 145, 57]. Hypoxia-inducible factors (HIFs) are transcription factors that respond to changes in available oxygen in the cellular environment, specifically to hypoxia. When activated, HIF-1 upregulates several genes to promote survival in low-oxygen conditions. These include glycolysis enzymes that allow cells to synthesize ATP in an oxygen-independent manner; and vascular endothelial growth factor (VEGF) that cells release to promote angiogenesis. So hypoxia is directly instrumental in tumor progression. Prolonged or extreme hypoxia can lead to necrosis, and tumors often have central regions called necrotic cores. Necrosis in turn activates inflammatory

responses that produce cytokines that stimulate tumor growth [57]. Recent research has been investigating the interactions between hypoxic tumor cells and immune cells (tumor-associated macrophages [116]) and cells that synthesize extracellular matrix (tumor-associated fibroblasts [52, 22]). Both are involved with inflammatory processes tied to tumor progression. In the context of the tumor microenvironment, these interactions regulate tumor properties like spatial patterns of cell localization, angiogenesis, and collective invasion and migration [132, 21]. So it is of critical theoretical and clinical significance to understand how, and under what conditions, hypoxia arises in tumors.

1.2.2 *Tumor heterogeneity*

Tumors are disorganized, heterogeneous tissues, consisting of many distinct cell types in spatially complex arrangements. Besides the polyclonal proliferating cancer cell population undergoing somatic evolution [111], tumors include non-proliferating stromal cells, fibroblasts, immune cells, extracellular matrix, collagen, blood vessels, and other structures and cell types [61, 62, 57]—these include "normal" cells that are conscripted by transformed cells to play collaborative roles in the neoplastic agenda. So there is a large degree of genotypic and phenotypic heterogeneity composing a tumor. Since tumors originate in physiological structures that range from simple epithelial sheets to ducts to neural and muscle tissue, and often invade neighboring tissues and then metastasize, colonizing distant tissues, the spatial situations and geometric structural relationships of tumors are themselves complex and heterogeneous. Add to this the dynamic character of the microenvironment, from high frequency variations in oxygen, nutrient, and signaling molecule concentrations, to longer time

scale processes like the synthesis of extracellular matrix and blood vessels during angiogenesis.

1.2.3 *Metabolic heterogeneity*

We are especially interested in metabolic heterogeneity [69]. In tumors, while we observe the Warburg effect [151, 150] commonly arise, aerobic glycolysis is not the only metabolic program cancer cells follow. In fact, there are experiments and mathematical models to suggest the metabolic strategies tumors use, when considered as a whole, is quite dynamic [146, 98, 79]. As a tumor progresses, it negotiates a course of barriers to proliferation [61, 47, 62]. Its gain of oncogenic function [157, 25, 26, 156, 44, 108, 160, 96], and loss of tumor suppressor function [78, 102, 127], give rise to changing bioenergetic and biosynthetic requirements for proliferation in the face of the new obstacles [79]. So the tumor cells' metabolic programs are varied and in flux, following from this coevolution.

1.2.4 *Hypoxia and metabolic heterogeneity*

The somatic evolution of early carcinogenesis feeds on sources of phenotypical variation present in tumor and surrounding stromal tissue, including metabolic variation. Hypoxia plays an important role in producing intra-tumor metabolic heterogeneity, as chronic hypoxia produces sustained conditions of metabolic stress that lead to phenotypical adaptation and oncogenic transformation that support growth and proliferation [81], and intermittent hypoxia produces cyclic conditions of oxygen de-

privation and reoxygenation that are only recently being detected and investigated [20].

1.2.5 *The cancer metabolism Renaissance*

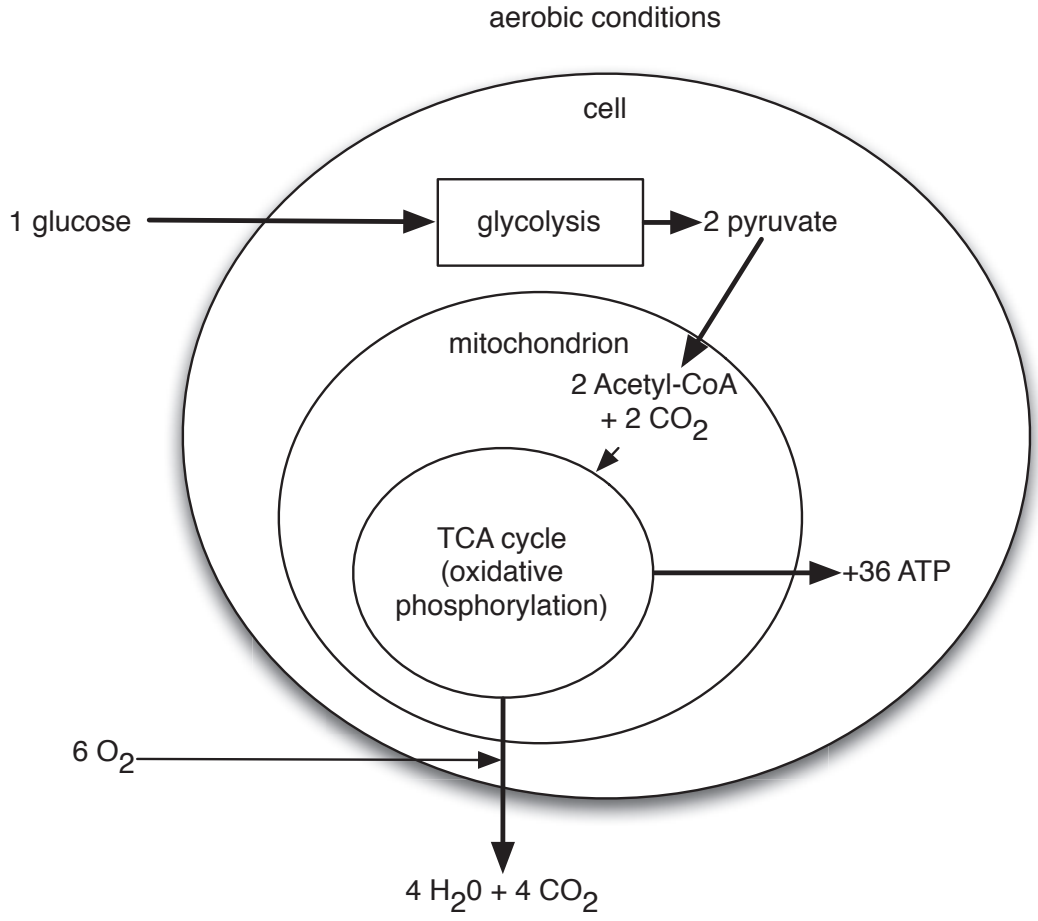

Figure 1: Nontransformed cells use mitochindrial oxidative phosphorylation to support their largely bioenergetic needs of cellular maintenance and homeostasis. When these cells consume glucose (glu), it enters the glycolytic metabolic pathway and is transformed into pyruvate; it subsequently enters the TCA cycle in the mitochondria where it is further transformed into Acetyl-CoA and CO_2. When these cells consume oxygen (O_2), it enters the TCA cycle with the Acetyl-CoA and CO_2 to produce H_2O and CO_2, eventually released by the cell, and produce ATP that the cell uses. With respect to energy production (output molecules of ATP per input molecules of glu and O_2), this process is an order of magnitude more efficient than aerobic glycolysis (Figure 2).

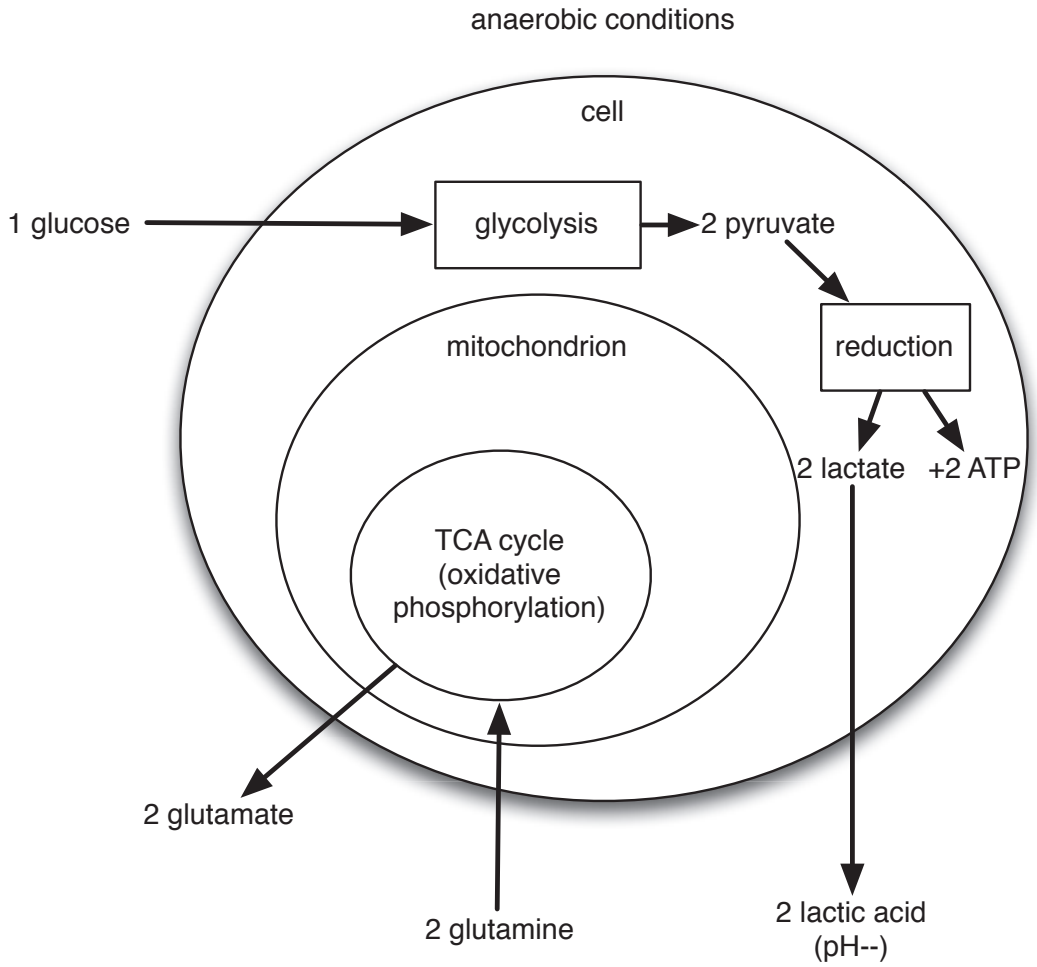

Figure 2: Tumor cells often use anaerobic glycolysis even in the presence of oxygen ("aerobic glycolysis") to support their proliferative requirements (Warburg effect). Where glucose (glu) consumption supports bioengetic needs, glutamine (gln) consumption supports biosynthetic needs (for carbon backbones, etc.). When these cells consume glu, it enters the glycolytic metabolic pathway and is transformed into pyruvate (pyr) that is subsequently reduced to lactate (lac), eventually released by the cell, and ATP that the cell uses—circumventing the mitochondria. Separately, when these cells consume glutamine (gln), it enters the TCA cycle in the mitochondria and after donating its carbon, the cell eventually releases glutamate (glu). Together, glutamic and lactic acids lower the pH of the cell's microenvironment. With respect to energy production (output molecules of ATP per input molecules of glu), this process is an order of magnitude less efficient than mitochondrial oxidative phosphorylation (Figure 1).

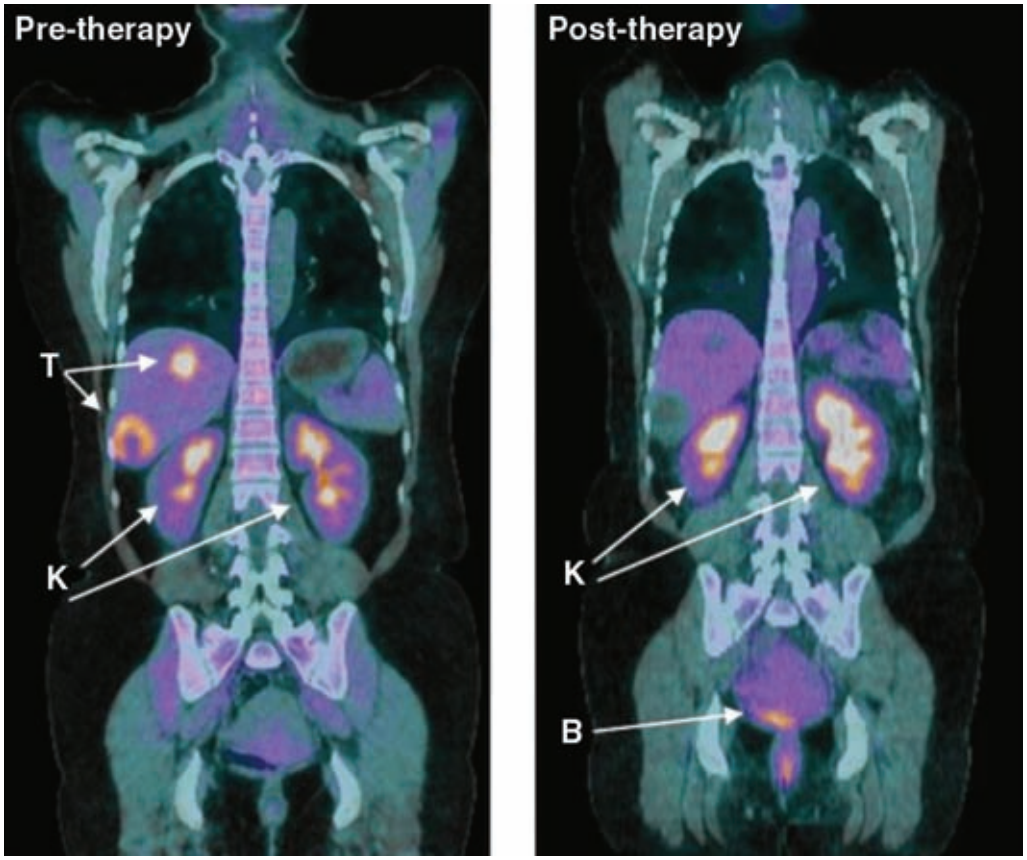

Figure 3: Whole-body 2D PET/CT scan using ^{18}F-FDG. A patient with malignant gastrointestinal stromal tumors was infused with ^{18}F-FDG, a glucose analog, and then scanned on a hybrid PET/CT scanner before (left) and after (right) 4 weeks of administering the tyrosine kinase inhibitor *sunitinib*. Before therapy, bright regions show the increased uptake of glucose by the tumors (T). Excess ^{18}F-FDG is excreted into the urine, thereby migrating and collecting in the kidneys (K) and bladder (B), which also show bright regions in left and right images. In the case shown here, the decrease in metabolism of glucose by the tumors predicts the patient's response to anticancer therapy. [Image taken from [64].]

The past decade has seen a Renaissance in the study of cancer cell metabolism [91], challenging and extending Warburg's original observation of pervasive anaerobic glycolysis in tumor tissue even in the presence of oxygen [151, 150]—please refer to Figure 1 and Figure 2. Between their celebrated syntheses of 2000 [61] and 2011

[62], Hanahan & Weinberg updated their cancer hallmarks to include metabolic reprogramming. The new frontier has been pushed by a group of research laboratories, headed by Craig Thompson, Matthew Vander Heiden, Lewis Cantley, David Sabatini, Ralph DeBerardinis, Eileen White, Joshua Rabinowitz, Chi Dang, and Tak Mak, among others.

1.2.5.1 *The primacy of metabolism*

If we take Craig Thompson to be the frontier's leading spokesperson, then a central aim of the research programme is to give causal primacy to cancer cell metabolism in the process of tumorigenesis [152]. In 2012, one of the first symposia on cancer metabolomics, hosted by the New York Academy of Sciences, presented the research of many of these laboratories [112]. During his keynote address, Craig Thompson remarked, "A decade ago, if a paper appeared with the phrase 'anaplerotic' in the title it would've been summarily rejected by the major journals...not the case any longer." It seems clear the subfield has burst from its chrysalis. Several good reviews of this research have been published over the past decade [46, 16, 154, 88, 28, 31, 30, 68, 78, 64, 80, 101, 87, 18, 19, 152, 29, 27].

1.2.5.2 *Ras and Myc*

Some research has focused on specific oncogenes, usually Ras and Myc, and their roles in transforming glucose and glutamine metabolism, respectively. On the Ras front, one study builds a case for the hypothesis that oncogenic K-Ras decouples glucose and glutamine metabolism to support cancer cell growth [44], while another builds a case for the hypothesis that oncogenic K-Ras maintains pancreatic tumors through regulation of anabolic glucose metabolism [160]. One study examines the

consequences of enhanced cell-autonomous glutamine metabolism [96]. On the Myc front, one particularly compelling line of research investigates a phenomenon known as "glutamine addiction," where cancer cells acquire an exquisite sensitivity to the concentration of glutamine, and even slight deprivation leads to widespread apoptosis [157, 25, 156, 26, 108]—yet another "Achilles' heel" of cancer to exploit therapeutically.

1.2.5.3 *Metabolic enzymes*

Much recent research has focused on the transformative effects of mutations in metabolic enzymes. One of the key players under active investigation is isocitrate dehydrogenase [122, 17]. In addition to isocitrate dehydrogenase, in the context of the most experimentally studied metabolic pathways supporting cancer cells—glycolysis, TCA cycle, pentose phosphate, glutaminolysis, and oxidative phosphorylation—other key players include: lactate dehydrogenase, pyruvate dehydrogenase, fumarate dehydrogenase, succinate dehydrogenase, and ATP-cytrate lyase. Modeling and predicting the effects of their mutation or disruption on the network of cancer metabolic pathways is the focus of other research [123]. Other studies focus on the proliferative and pro-survival roles played by specific metabolites, like glysine and serine [140, 72, 97].

1.2.5.4 *The embryonic program*

One frequently hears comparisons between the heightened proliferation in cancer and the deregulated embryonic growth program, often referred to as "reversion to the embryonic state." At least one study has examined metabolic regulation in pluripotent stem cells during reprogramming and self-renewal [161]. Another study investigated the Warburg effect in the developing retina [43].

1.2.5.5 *Hypoxia and metabolic reprogramming*

A number of studies have investigated the relationship of cancer cell metabolic reprogramming to hypoxia, including the role HIF-1 plays upstream and downstream of cancer metabolism [129, 130], and consideration of the triad of oxidative stress, tumor microenvironment, and metabolic reprogramming [42].

1.2.6 *Cell-autonomous versus non-cell-autonomous*

Most of this intense investigation has focused on the cell-autonomous view of cancer cell metabolic reprogramming. But as discussed above, a complimentary view is taking shape that tumor progression does not depend exclusively on cell-autonomous properties of the cancer cells, but also on properties that can only be observed, simulated, and analyzed at the tissue-scale of whole tumor cell population. With respect to this latter view, we seek to explore the hypoxia origin question computationally.

1.2.6.1 *Specific hypotheses*

Certain non-cell-autonomous hypotheses are being investigated that could explain observations related to evolving cancer cell metabolism in specific situations.

One is the so-called "metabolic symbiosis" between hypoxic and aerobic tumor cells, where lactate produced by hypoxic cells is taken up by aerobic cells, which use it as their principal substrate for oxidative phosphorylation. The two cell types thereby mutually regulate their access to energy metabolites [128, 136, 41, 84].

Another is the so-called "reverse Warburg effect," where epithelial cancer cells induce aerobic glycolysis in neighboring stromal fibroblasts. These cancer-associated fibroblasts then undergo myo-fibroblastic differentiation, and secrete lactate and pyru-

vate, energy metabolites resulting from aerobic glycolysis. Epithelial cancer cells could then take up these energy-rich metabolites and use them in the mitochondrial TCA cycle, thereby promoting efficient energy production—ATP generation via oxidative phosphorylation—resulting in a higher proliferative capacity [118, 137].

Still another is the so-called "secondary senescence," where non-cell-autonomous interactions between tumor cells and nonmalignant bystander cells add to cell-autonomous modes of tumor suppression during tumor development and progression. In this scenario, stroma or host immune cells convert tumor-generated signals into a response that feeds back to the tumor cell population. In particular, suppose Myc plays a primary role promoting apoptosis in a subset of the tumor cell population, which leads to the attraction of macrophages; these subsequently engulf the apoptotic remainders. Phagocytosis-activated macrophages, in turn, exhibit strongly increased secretion of various cytokines, among them transforming growth factor beta to an extent that is capable of inducing cellular senescence in surrounding malignant cells [57, 121, 34].

1.2.7 *Cancer metabolomics*

For a long time, there was no systematic characterization of metabolic pathways active in transformed cells, so the contribution of these pathways in promoting rapid cancer cell proliferation was unclear. But in 2012, Jain, *et al* [140, 72] produced a comprehensive metabolite profile for each of the NCI-60[1] cancer cell lines. To systematically characterize cancer cell metabolism, they created cellular consumption and release (CORE) profiles of 219 metabolites spanning the major pathways of in-

1 The NCI-60 is comprised of sixty well-characterized primary human cancer cell lines established from nine common tumor types.

termediate metabolism. The study contains some compelling findings[2], but we were particularly encouraged by the conceptual approach Jain, *et al* undertook in their methods, namely, that cancer cell metabolic reprogramming *manifests* as altered nutrient uptake and release. In other words, the gross quantitative properties of cellular consumption rate and release rate of metabolites (and other particles, like gasses and signaling molecules) is sufficient to characterize and distinguish cancer cell metabolic phenotypes from an extracellular perspective.

1.2.8 *Returning to Hypoxia*

The phenomena of cancer cell metabolic reprogramming and hypoxia can be studied separately. However, as noted above, hypoxia plays a role in creating the metabolic heterogeneity we see in tumors. We believe the emergence of hypoxia ought to be studied from a non-cell-autonomous perspective, as discussed above in the context of cancer metabolism. Further, we believe the two phenomena can shed light upon each other, and so we set out to create a computational framework that is flexible enough to model both. That said, it is the emergence of hypoxia that is the central focus of this dissertation and the concrete test case of our approach.

1.3 PROBLEM CONVERSION

Our aim is to transform this cell-population biological origins problem into a computational inverse problem, and then attack it using approaches from computer science. We envision a system that will drive an *in silico* model forward, from some set of ini-

2 The study implicates a role for glycine in rapid cancer cell proliferation.

tial conditions and operational parameters, to a recognizable, well-characterized state of hypoxia formation. In general, a mathematical model takes the form $G(m) = d$, where G is usually an ODE, PDE, or algorithm, m is the model, and d is the data. In the real world, $d = G(m_{true}) + \eta$, where η is noise. The *forward problem* is to find d given m, by computing $G(m)$ (by solving an ODE or PDE, or running a simulation). The *inverse problem* is to find m given d. When m and d are continuous functions of time and space, the task of estimating m from d is a continuous inverse problem. These can often times be well approximated by discrete inverse, or parameter estimation, problems, where model and data are vectors of parameters, $\vec{m}$ and $\vec{d}$, respectively.

1.4 PROBLEM RESTATEMENT

With this in mind, we restate the biological origins problem of hypoxia as a computational inverse problem: Given data in the form of histology images from which we may formally characterize the phenomenon of hypoxia ($\vec{d}$), and given an *in silico* model in the form of a stochastic, spatially-resolved simulation of a heterogeneous population of cells (G), what set of model parameters ($\vec{m}$) will drive the simulation from its initial state to one that corresponds with sufficient similarity to the formal characterization of the data?

1.5 COMPUTATIONAL BACKGROUND & LITERATURE REVIEW

In the context of systems biology, and in general, we believe, it is useful to distinguish between *parameter estimation* from experimental data sets, and *qualitative* inverse prob-

lems that aim to reverse engineer bifurcation patterns and other kinds of desired qualitative behavior [39]. Parameter estimation attempts to either provide values (or bounds on values) for unknown or difficult to determine parameters, or to determine insensitivities of data sets to certain parameters, which then are not accessible from the given data and require additional experimental information for their determination. Qualitative inverse problems attempt to explore the areas in parameter space that give rise to a given qualitative behavior, like multiple steady-state solutions, oscillations, or deterministic chaos. Our inverse problem is of the qualitative type, and our given qualitative behavior is characterized by analyzing experimental histological image data.

Outside of biology, the geosciences are the crucible for inverse problems. A review by Mosegaard, *et al* [109], gives numerous examples of qualitative inverse problems in the geosciences that are tackled with Monte Carlo methods, similar to our approach. One particularly intriguing study by Wijns, *et al* [155] is focused on qualitative inverse modeling in the absence of established numerical criteria to act as inversion targets. They employ a method of interactive evolutionary computation that provides for the inclusion of qualitative geological expertise within a rigorous mathematical inversion scheme, by asking an expert user to evaluate a sequence of forward geological models. The traditional numerical misfit is replaced by a human appraisal of misfit. They use this interactive technique to successfully invert a geodynamic model for a conceptual pattern of fault spacing during crustal extension. Though we are interested in developing an automated method for our problem, we recognize that integrating human expertise and evaluation, even if occasionally, can make for a more robust solution. We will consider exploring this in future work.

Most of the examples of biological inverse problems we find in the literature, including this review by Engl, *et al*, are parameter estimation problems. The examples of qualitative inverse problems in biology that we find [94, 100, 60] consist in much smaller dimensions than ours, are formulated as inverse bifurcation problems, and are based on ODE methods that neglect the rich spatial structure of their systems and do not handle stochastic system behavior. We have found no studies involving qualitative inverse problems related to hypoxia emergence. Nor have we found any approaches that use our combination of computational methodologies.

The closest approach we found to what we envision, is a study by Grosu, *et al* [58], who use model checking with a temporal logical characterization to tackle the problem of learning and detecting emergent behavior in networks of cardiac myocytes. They develop a hybrid-automata network environment called CellExcite [10] for the efficient simulation of excitable cells. They perform discrete mode abstraction and hierarchical superposition of the elementary units by employing a quad-tree decomposition [56]. At each time step of their simulation, this abstract representation, Q is compared to their Linear Spatial-Superposition Logic (LSSL) formula, Φ, that characterizes spatial patterns such as spirals (learned through a classification process). If there is a finite path, π, in Q that satisfies Φ, then their system detects the emergence of spiral patterns and hence the approaching state of fibrillation. While this approach demonstrates the validity and effectiveness of using temporal logic and model checking for the problems of specification and detection of an emerging complex biological property, it is not so much concerned with their version of the longer time scale computational inverse problem: what initial conditions and operational parameters drive simulated cardio myocytes to a likely state of spiral waves followed by fibrillation. And the traditional model checking approach they have taken does

not lend itself well to stochastic models—the motivation for developing the statistical model checking approach.

1.6 OUR MATERIALS & METHODS

Our approach rests on three "pillars," or general system components, which are independent of, yet related to, each other in ways we shall elaborate on shortly. The approach we take within each pillar is constrained by its defining problem statement.

1.6.1 *Pillar 1*

PROBLEM STATEMENT Implement a spatially-resolved simulation framework for modeling the emergence of localized regions of hypoxia within a tissue-scale mixed population of cells. Cellular fitness should be locally determined. Cells should consume and release diffusible particles that represent metabolites, gasses, signaling molecules, etc. These constitute a complex spatial universe of heterogeneous information, stress, and reward, to which cells may adapt using either their type-defined default behaviors, or type-defined conditionally invoked behaviors that override their default behaviors. The simulator should be algorithmically simple and efficient. It should be fully specified by an input vector of numbers that code for initial conditions and operational parameters. Its data structures should be amenable to interrogation and decomposition for the purposes of run-time feature measurers and the hypoxia detector that integrates them, as specified by Pillar 2. Accordingly, the simulator should output either {0,1}, if the detector embeds a spatiotemporal logical

proposition integrator/detector, or a numerical value in [0,1], if the detector embeds a learned functional form integrator/detector.

OUR APPROACH We develop such a framework. It is situated in a 3D regular lattice and obeys basic properties of a cellular automaton. Cells of various types occupy lattice points at every time step, are affected by diffusing concentrations of diffusing particles of various types, and undergo local-fitness-based probabilistic reproduction. Using the Cleveland, *et al* framework as a conceptual starting point, we then extend it in significant and novel ways to further suit our needs. First, we can support any number of cell types and any number of particle types (each with its own diffusion rate). Second, each cell type has default behaviors and conditionally-invoked behaviors, which can implement phenotypical adaptations and mutations, and state machines composed of two or more cell types. Third, initial, and upper- and lower-bounded basal concentrations can be set for each particle type. Fourth, each cell type can be replaceable or not, and reproductive or not. Fifth, initial lattice occupation can be delayed to establish complex diffusion gradients to form prior to simulation. The 3D lattice data structure is simple, regular, and easy to interrogate for the purposes of feature measurer and feature integrator modules we can later implement to detect emergent phenomena, such as necrotic core formation and stable local regions of hypoxia like we observe in xeno-graphed hypoxic tumor histology.

1.6.2 *Pillar 2*

PROBLEM STATEMENT Derive a spatiotemporal characterization of hypoxia in human tumor tissue from a set of histological images.

OUR APPROACH Our concrete focus into the phenomenon of tumor cell hypoxia begins with an experiment where human colon tumor cells were xeno-graphed into a nude mouse, and upon subsequent analysis were determined to exhibit localized regions of hypoxia. This data is in the form of histology images taken from anti-pimonidazol stained tumor sections. Our approach consists in extracting qualitative and quantitative features from these histology images. We classify these as: (1) features that derive from segmenting the image into the three tissue types depicted: viable tumor cells, hypoxic tumor cells, and necrotic tumor cells; (2) features related to the intra-lesion hypoxia gradient, as measured from radial distance away from the nearest vessel; (3) features that derive from multiscale analysis; and (4) features that relate to qualitative generalities about bounded and nested structure. Once we specify a set of features, we proceed in two separate but related directions. First, we attempt to construct a logical proposition to describe hypoxia in space and time using a simple spatiotemporal logic (also expressible as a model logic [70]) whose primitives are image feature predicates. This is a human-driven process, following from human learning and generalization. Second, we attempt to construct a linear regression function that learns what hypoxia is in terms of estimated linear coefficients on the image feature terms. This is an machine-driven process, kept on the rails by a combination of false-positive and false-negative control, and feature dimensionality reduction where possible.

1.6.3 *Pillar 3*

PROBLEM STATEMENT Identify the initial conditions and operational parameters of an *in silico* model (simulation) that result in hypoxia, as characterized by Pillar 2.

OUR APPROACH The nature and extent of the simulation parameter space derives from the algorithmic specification given in Pillar 1. A number of parameters define any given simulation, those that specify initial conditions and those that specify the entities that operate in the simulation. Initial condition parameters specify: the initial positions in the 3D lattice of the cells of various types; the initial and basal upper- and lower-bound concentrations of the various particle types; and the delay time indicating when to place the cells in their initial positions. Operational parameters specify: each particle type's diffusion rate; the consumption and release rates and impact factors of each cell type for each particle type; whether each cell type is replaceable and whether its reproductive; and the conditional behaviors for each cell type. Together, these constitute a high dimensional parameter space.

For our problem, we will assume that in the absence of simplifying factors or expert knowledge of the biology, each parameter should be modeled as a random variable having a uniform, independent probability distribution. Our aim here is modest: sample the large parameter space using a vanilla Monte Carlo algorithm [105] and accumulate families of nearby solutions. We do attempt to make this process more efficient by learning from each sample's truth outcome if it is in 0,1, or branching-and-bounding sampled subspaces where the sample values are in [0,1]. In this way, we propose two simple "adaptive Monte Carlo" methods, MC-Boost and MC-Walk, that employ boosting of the independent probability distributions upon successful samples, and constrained random walks around successful samples, respectively. And we propose a simple adaptation to the traditional branch-and-bound algorithm, MC-Branch-and-Bound: instead of systematically exploring a subspace of problems, we employ constrained Monte Carlo sampling of each subspace.

Our model (simulation), once specified by a coordinate from the high dimensional parameter space, is still stochastic, owing to the cells' manner of probabilistic reproduction. Suppose a feature integrator/detector is embedded in the simulator that decides when the formal characterization of hypoxia—be it a spatiotemporal logical proposition, ϕ, or a learned linear function, f—is satisfied. It will then enable the simulator to return a value in $\{0,1\}$ or $[0,1]$, respectively, where the first is a the evaluation of a logical proposition, and the second is a normalized similarity score. Thus, depending on the detection scheme being implemented, the simulator's outcome can be modeled as either a Bernoulli random variable or a numerical random variable. The first case is the province of statistical model checking. Jha, *et al* [77] give the Bayesian statistical model checking algorithm we employ here. It runs the simulator some bounded number of times until enough confidence accrues to the null hypothesis (the simulator satisfies, within some bounded probability, the spatiotemporal logical proposition describing hypoxia), or its alternative hypothesis (it does not). This verdict constitutes the Bayesian-tested outcome of the simulation with respect to satisfying ϕ. In a similar, and perhaps trivial, sense, the numerical $[0,1]$ outcome of the simulation should be tested repeatedly until some threshold on the numerical stability of its mean value is surpassed. Here the simple approach we use is to examine its mean and standard deviation, and apply a threshold to its $CV = \frac{\sigma}{\mu}$.

This pillar presents two distinct ways of implementing a two-level simulation driver. The top level explores by sampling the high dimensional parameter space, testing a coordinate in that space by passing control down to the lower level that repeatedly runs the specified simulation until a stable outcome is achieved. It then passes the binary verdict up to the top level that records and eventually responds to the coordinate's computed truth value.

1.6.4 *System description*

Now we describe how we organize the three pillars above into a solution to the computational inverse problem stated above. We show this in three views, corresponding to: the problem stages, the system relationships established during the design phase, and the structure of the system's run time execution.

1.6.4.1 *Problem view*

In the problem view shown in Figure 4, we depict the problem dependencies and methods for the computational solution presented here. Solving the inverse problem depends on exploration of the simulation configuration space, accomplished by the adaptive sampling methods of Pillar 3. Exploration depends on robust detection of the hypoxia characterizations being satisfied in the simulation, accomplished by either the Bayesian statistical model checking or the mean-variance thresholding methods of Pillar 3 that drive the simulations to a stable result. Detection depends on a description of hypoxia derived from the available evidence, accomplished by either the spatiotemporal logical or linear regression functional characterization methods of Pillar 2. And description depends on analysis and synthesis of the histology images, accomplished by the image analysis methods of Pillar 2.

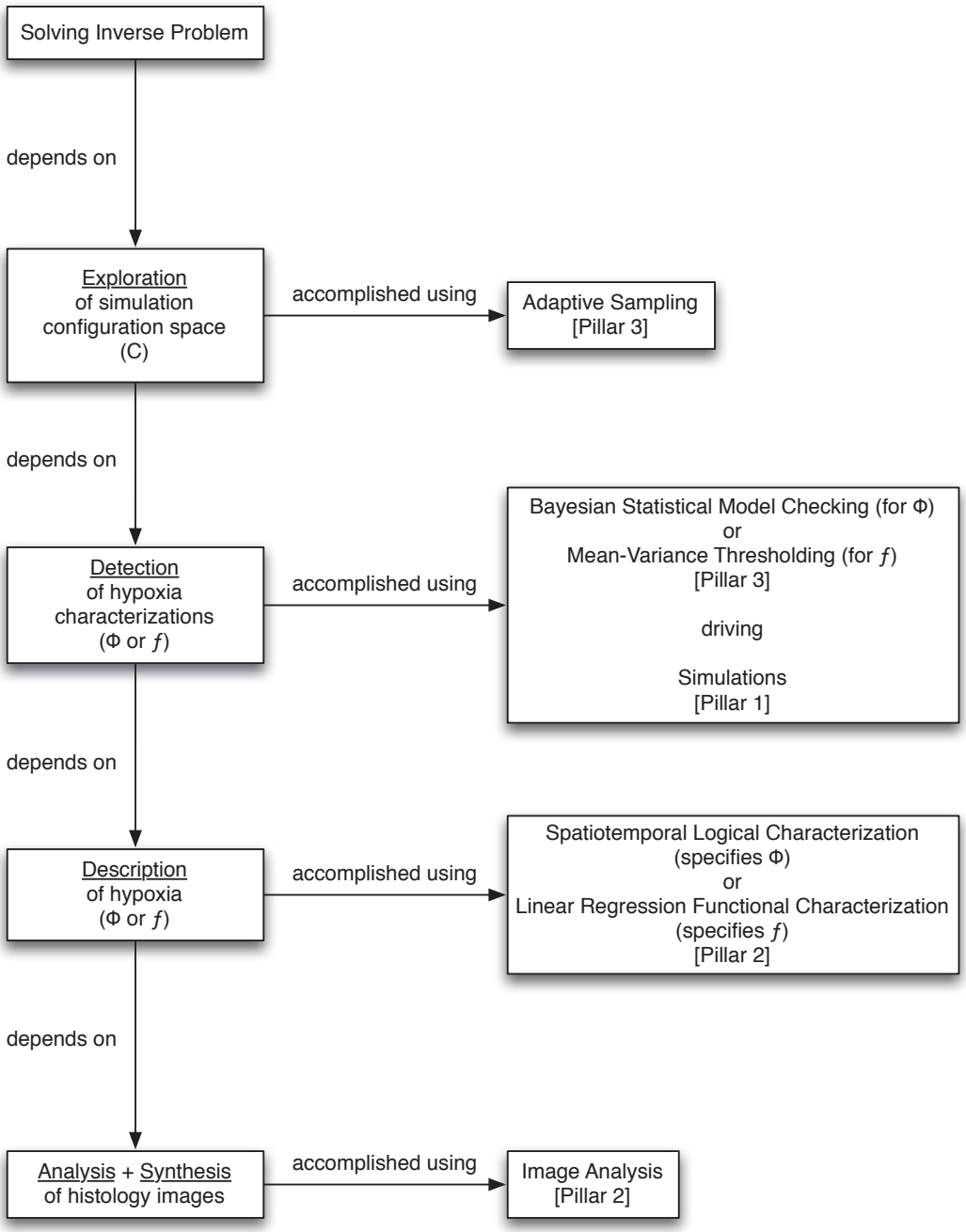

Figure 4: Problem view. System problem dependencies and methods for the computational solution presented here.

1.6.4.2 *Design-time view*

At design time, the components of the system either specify or inform some of the others, as shown in Figure 5.

A number of requirements inform the design of the simulator. If we are computationally constrained, then we cap the number of parameters by which the simulator may be configured by some constant κ. For efficiency as an "inner loop" in the overall system execution, we require the simulator halts once the embedded logical feature integrator detects hypoxia, and that 0 or 1 then be returned. The embedded feature measurers require the simulator use data structures that are easy to interrogate and decompose. Lastly, the feature integrator requires the simulator to return 0 or 1, or a real value in [0,1], depending on whether the feature integrator is logical or functional in nature.

As mentioned, the simulator embeds feature measurers and feature integrators/detectors. The simulator specifies its data structures, D, and its configuration parameter space, C. The configuration parameter space informs the adaptive sampler, along with any expert biology knowledge, K, which can prune and constrain C prior to its exploration. The data structures specify the set of simulator features, F_s, that may be implemented. These in turn inform the hypoxia characterization process, since we must try to do this in terms of possible features, which may be defined as, $F_s \cap F_i$, the intersection of simulator features and the image features. The image features, F_i, are specified by image analysis that is informed by the images themselves, I, which are specified by biological experiment. Hypoxia characterization specifies the intersection of features and subsequently informs the design of feature measurers, the spatiotemporal logical characterization, and the linear regression functional characterization. Spatiotemporal logical characterization specifies the proposition ϕ, which

then specifies the logical feature integrator/detector. Linear functional characterization specifies the functional form f, which then specifies the functional feature integrator/detector.

By execution time, we assume the system design has been settled and implemented.

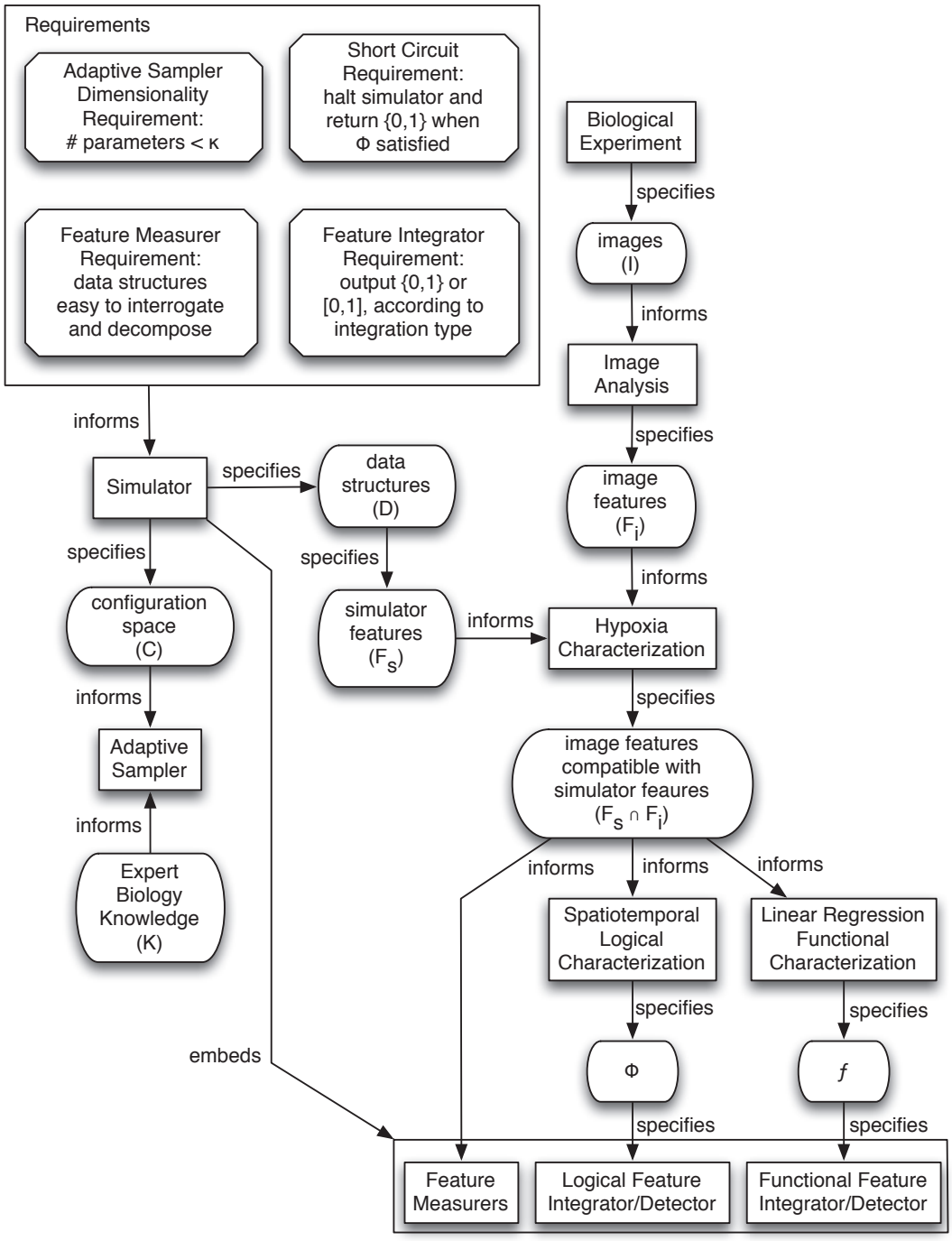

Figure 5: Design-time view. System design inputs and specifications for the computational solution presented here.

1.6.4.3 *Run-time view*

We show the two possible execution paths of the system in Figure 6. These correspond to the choice of whether to use a logical or functional characterization/detection scheme.

In the logical scheme, we assume three global parameters: the configuration parameter space, C, a probability threshold, θ, and a simulation time threshold, T. The adaptive Monte Carlo sampling method selects a $c \in C$ and calls the Bayesian statistical model checker with c and θ. This then calls the simulator with c, thereby configuring the simulator for an execution. The simulator embeds the logical feature integrator/detector and feature measurers, together which implement ϕ. The simulator either halts early, returning 1, or runs its full course (to time T), returning 0. Depending on this outcome, the Bayesian statistical model checker decides whether or not to execute another simulation of c. Once it halts, the Bayesian statistical model checker returns its binary verdict to the adaptive Monte Carlo sampler, which records this evaluation and adapts its subsequent sampling on the basis of it.

In the functional scheme, we assume three global parameters: the configuration parameter space, C, a coefficient of variation threshold, τ, and a simulation time threshold, T. The Monte Carlo branch-and-bound sampler selects a $c \in C$ and calls the mean-variance thresholder with c and τ. This then calls the simulator with c, thereby configuring the simulator for an execution. The simulator embeds the functional feature integrator/detector and feature measurers, together which implement f. The simulator halts at T and returns its "high water mark" normalized similarity score, a real number in [0,1]. Depending on this outcome, the mean-variance thresholder decides whether or not to execute another simulation of c. Once it halts, the mean-variance thresholder returns its binary verdict to the Monte Carlo branch-and-

bound sampler, which records this evaluation and adapts its subsequent sampling on the basis of it.

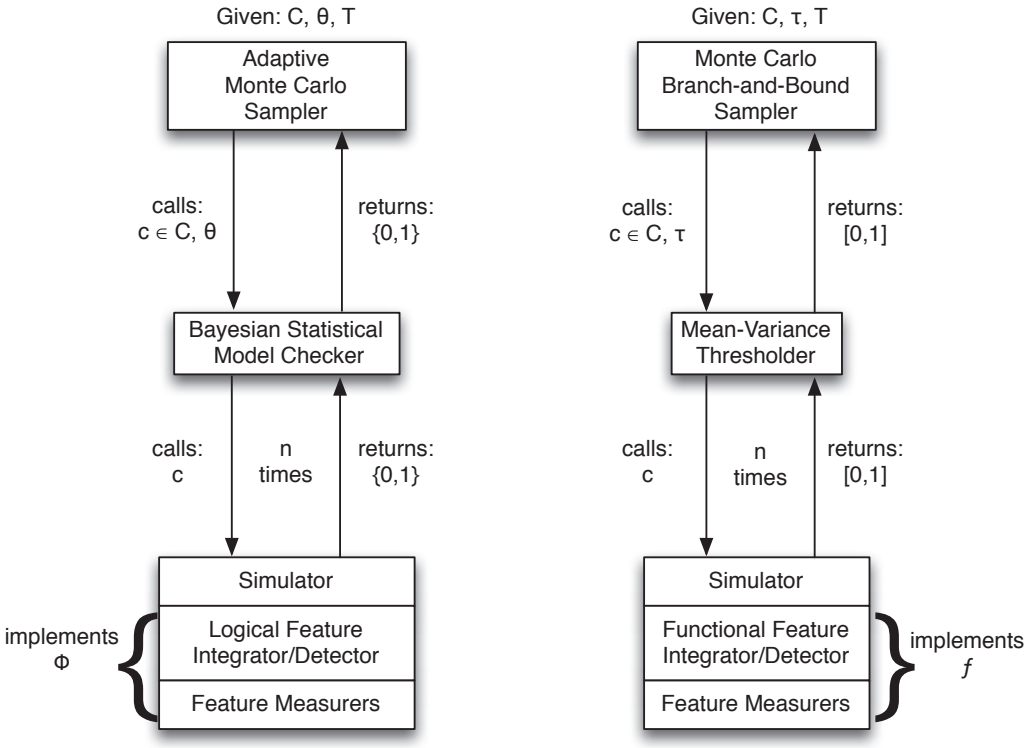

Figure 6: Run-time view. System execution calls and returns for the computational solution presented here.

1.6.5 *The pillars in detail*

We have designed a modular system; the three pillars may stand independently from the others, then be coordinated at design time. Should we find a better simulation framework, we can replace the current one and it will not jeopardize (though it will distinctly inform the design of) the other system components. Likewise for replacing the type of data used for characterization. Likewise for replacing the algorithms

employed to sample high dimensional space and ascertain stability of the stochastic simulator. These may be replaced without destroying the overall system design. The system is also extensible and scalable, especially the components using feature-based characterization, since one may select an arbitrary number of features. Taken together, these three system components formulate a novel approach to the inverse problem stated above, and constitute a design for a tool that can be placed into the hands of experimentalists, for testing existing and new hypotheses, either based on known parameter values, or on ones the tool discovers. In principle, this design can be generalized to other biological phenomena involving large heterogeneous populations of interacting cells.

Now let us consider each of the pillars in detail over the course of the next three chapters.

2

SIMULATION FRAMEWORK

2.1 INTRODUCTION

2.1.1 *Problem statement*

Implement a spatially-resolved simulation framework for modeling the emergence of localized regions of hypoxia within a tissue-scale mixed population of cells. Cellular fitness should be locally determined. Cells should consume and release diffusible particles that represent metabolites, gasses, signaling molecules, etc. These constitute a complex spatial universe of heterogeneous information, stress, and reward, to which cells may adapt using either their type-defined default behaviors, or type-defined conditionally invoked behaviors that override their default behaviors. The simulator should be algorithmically simple and efficient. It should be fully specified by an input vector of numbers that code for initial conditions and operational parameters. Its data structures should be amenable to interrogation and decomposition for the purposes of run-time feature measurers and the hypoxia detector that integrates them, as specified by Chapter 3. Accordingly, the simulator should output either {0,1}, if the detector embeds a spatiotemporal logical proposition integrator/detector, or a numerical value in [0,1], if the detector embeds a learned functional form integrator/detector.

2.1.2 *Background & literature review*

2.1.2.1 *Canonical model designs and their trade-offs*

When one is deciding how to create a computational model of a cell population, the primary choice is whether it should be continuous or discrete (or a hybrid). Usually, this breaks out into two canonical design dimensions[1]. In the first, one can represent cells as points, or as being composed of sub-elements. In the second, cells can occupy positions on a fixed, regular lattice, or positions off-lattice.

The lattice-gas cellular automata model [63] is an example of cells-as-points on a lattice. Hatzikirou, *et al.* use it to model cell migration in directional and oriented fields. It uses channels to alleviate the problem of collisions found in classical cellular automata models. Many individuals can move synchronously in each time step. It accomplishes this using a two-step transition rule. First, interaction: update channels and particles in each node. Second, propagation: deterministically move the particles based on their direction and velocity.

The off-lattice hybrid discrete-continuum model [76] is an example of cells-as-points off-lattice. Jeon, *et al.* use it to model tumor growth. It models cells as discrete points. They migrate using random and biased movement; internal and external forces determine their motion; and they grow, proliferate, die, become quiescent, and mutate. It models chemical fields as continuous functions of space and time. One can thus model extracellular matrix density, and concentrations of matrix degrading enzymes and nutrients. Cellular forces include: cell-cell soft sphere repulsion force,

1 I would like to gratefully acknowledge Terri Grosso at the CUNY Graduate Center for giving a review presentation on 26 June 2012 of these canonical modeling dimensions, and providing the example models, in the context of her research into computational modeling of cell migration.

haptotactic force, and cell-cell adhesion force. In terms of the chemical gradients: the extracellular matrix degrades based on its concentration and that of the matrix-degrading enzymes, and the rate enzyme-matrix degradation; matrix-degrading enzyme concentration depends on diffusion through the extracellular matrix, production by cells, and its natural decay rate; and nutrient concentration depends on diffusion, concentration of extracellular matrix, and cell consumption and natural decay.

The cellular Potts model [13] is an example of cells composed of sub-elements on a lattice. Bauer, *et al.* use it to model tumor-driven angiogenesis and the transwell migration assay. It was developed to study bubbles and other surface-energy-driven processes. It conceives of cells as fluid droplets. A cell is a contiguous set of lattice locations that share a unique index. The lattice evolves as a succession of attempts to exchange neighboring lattice site indices. It accomplishes this by step-wise, synchronous minimizing the total energy of the system, accepting those proposed lattice site index exchanges that do this with a Boltzmann probability. The model represents the total energy of the system as a function of cell-cell adhesion, cells' current and target volumes, and other energy terms, like chemotaxis.

The sub-cellular viscoelastic model [73] is an example of cells composed as sub-elements off-lattice. Jamali, *et al.* use it to model emergent and complex cellular morphology. The model aims to represent the internal structure of the cell and then model cellular processes such as: adhesion, growth, mitosis, migration, polarization, a distinct nucleus, cell-environment interactions, and biomechanical behavior. Cells are modeled as an approximately circular membrane and an internal nuclear membrane; the cell is then divided into some number of segments, and the mass of the cytoskeleton and nucleus are divided between the segments. Points are connected by Voigt subunits, where purely viscous elements are a damper, purely elastic elements

are a spring, and units are connected in parallel. Total force on the cell is computed as the sum of forces exerted: from inner structure of the cell; from interaction with other cells; from interaction with the extracellular matrix; from external sources; and from contractility during cell division.

Considering again the canonical modeling dimensions, one should weigh several trade-offs. When one models cells as points, one can represent large numbers of cells in a computationally efficient manner; but the model is coarse-grained, neglecting cell mechanics and other biophysical considerations. When one models cells composed of sub-elements, one can better represent cell shape, cytoskeleton, and internal structure; but this requires more computation and one can therefore represent fewer objects. Lattice-based models are computationally efficient and afford a simpler algorithmic design; but the complexity depends on lattice size, not the number of objects, and the rigid structure of the lattice can affect morphology and behavior. Off-lattice models have a complexity that depends on the number of objects being modeled, and one can model cell movement and morphology continuously; but collision detection is computationally expensive, and interactions between nearby elements can be more expensive than using a lattice.

2.1.2.2 *Biophysically realistic models*

Plank, *et al.* [119] compare lattice-based and lattice-free approaches to the problem of modeling collective cell behavior with crowding effects in individual-based (agent-based) models. They note that lattice-based models implicitly assume a proliferative population will always eventually fill the lattice. They develop their own individual-based lattice-free model that incorporates cell crowding effects, where the confluent cell density is not predefined as with a lattice-based model, but is instead an emergent

model property. As a consequence of the more realistic, irregular configuration of cells in the lattice-free model, the population growth rate is much slower at high cell densities, and the population cannot reach the same confluent density as an equivalent lattice-based model.

Xavier, *et al.* [159] developed an elaborate simulation framework for multidimensional modeling of activity and structure of multi-species biofilms. This integrates concepts from previous biofilm models into a realistic biophysical simulation that runs in a 2D or 3D off-lattice environment. They pay much attention to biomass computation, as this is one of the principal concerns in biofilm modeling. In the particular simulations they show, they use 25 model parameters that pertain to: solute species, particulate species, yield coefficients, rate parameters, and computation parameters. The simulation proceeds as a cycle over the following steps: (1) determine the time step, Δt, for the current iteration at time t; (2) grow every biomass agent, dividing if its radius surpasses a threshold, and excreting extracellular polymeric substances if these surpass a threshold; (3) spread the cellular constituents, resettling them according to a global energy minimization, advancing the biofilm front; (4) detach biomass, including erosion and sloughing; (5) update the bulk concentration of solutes, performing global mass balances for solutes with dynamic behavior; (6) update the spatial concentration fields of solutes to steady state; (7) advance the simulation time to $t + \Delta t$. One can easily define a stoichiometric table for one's simulation, specifying the constituents, reactions, and rate expressions in a concise manner. Notably, the simulation cannot explicitly perform apoptosis, nor can it implement conditional logic. It is a deterministic framework as it implements reactions as ODEs and numerically solves PDEs for the spatially resolved particle concentrations.

2.1.2.3 *Multiscale models*

In addition to the models described above, there is an extensive literature on multiscale cancer modeling [24, 32]. These models aim to incorporate biological properties that range in length and time scales pertaining to molecules, cells, tissues, organs, and whole organisms. We laud this effort; cancer is a systems disease that requires the unified consideration of a broad set of structures and environments that span many scales. Often these models, some of which are rigorously derived [24], are complicated and computationally expensive. While cancer metabolism is a multiscale phenomenon [91], we restrict our attention to the scales that define a large tumor cell population within a single tissue. While we do not seek to completely neglect biophysical detail in our model, we do seek a coarse-grained, minimal model that can capture at least some of the qualitative features that define local regions of hypoxia in histological evidence from *in vivo* experiments. For our purposes here, we also seek a model that is computationally efficient, since in the broader context of this dissertation, we understand it will be the "inner loop" of a large processing regime to (at least partially) solve the inverse problem of hypoxia formation. With all of this under consideration, we chose to implement our simulation representing cells as points based on a lattice.

2.1.2.4 *Cancer metabolism models*

Astanin, *et al.* [3] develop a mathematical model of the Warburg effect in tumor cords. It links two approaches: a continuous medium to describe the movement and the mechanical properties of the tissue, and a population dynamics approach to represent tumor-intrinsic heterogeneity and instability. While one can use their framework to build models which cover several stages of tumor progression, they focus on describ-

ing the transition from oxidative phosphorylation to purely glycolytic metabolism in tumor cords. Growth and decay of the cells and uptake of the nutrients are related through ATP production and energy costs of the cellular processes. Intriguingly, they assume the Warburg effect is an irreversible, all-or-nothing event triggered by hypoxia. Mathematically, this model leads to a free boundary problem where domains in contact are characterized by different sets of equations. They accurately stitch together the solution by developing a modified ghost fluid method. They employ PDE models for studying the boundary shape changes, exploring growth rates and resulting spatial structure of the glycolytic and oxidative phosphorylation subpopulations over a range of various parameter values. Consequently, the model is deterministic, and spatial heterogeneity can be modeled only to a limited extent, where subpopulations are contiguous and occupy adjacent layers [158].

2.1.2.5 *Game theory cancer models*

There is a growing literature of game theory models of cancer population dynamics [141, 142, 124, 8, 9, 49, 50, 48, 45, 103, 7, 12, 11]. Most of these early studies model tumor-tumor or tumor-stromal cell interactions in a generic way, or explore evolutionary dynamics of tumorigenesis, or the emergence of tumor invasiveness. None of them address cancer metabolism explicitly.

2.1.2.6 *Well-mixed models*

One recent study, by Kareva [82], develops a mathematical model based on game theory to investigate cancer metabolism. It is well understood that glycolysis is energetically inefficient relative to oxidative phosphorylation, producing 2 versus 30-36 ATP molecules, respectively, for each molecule of glucose. And glycolysis secretes 2

molecules of lactic acid for each molecule of glucose as a byproduct. The ensuing acidification is toxic for healthy tissues, enabling glycolytic cells to be better competitors at the cost of their being energetically inefficient. However, a single cell is unlikely to secrete enough lactic acid to cause significant changes to the microenvironment. The core population of glycolytic cells needs to be large enough to gain this competitive advantage. Kareva casts the problem as a prisoner's dilemma game: from the perspective of metabolic payoffs, it is better for cells to cooperate and become better competitors, but neither cell metabolic phenotype has an incentive to unilaterally change its metabolic strategy—they are in a Nash equilibrium [110], and it can be argued that metabolically, oxidative phosphorylation is an evolutionary stable strategy [134, 135] that cannot be "invaded" by the glycolytic strategy. In this formulation, Kareva addresses the question of how such a glycolytic population could arise. One intriguing aspect of the game theory perspective is the notion of "public goods" in the cell ecology. For example, intracellular stores of nutrients can be recycled by neighboring cells [37, 38, 83]. She shows that changing the environment can take cells out of their Nash equilibrium, and that it is cooperation [6, 5, 4, 114] that can lead to the cell population committing "evolutionary suicide." The author develops an ODE model for studying the population dynamics governed by the prisoner's dilemma payoff matrix, exploring growth rates of the glycolytic and oxidative phosphorylation subpopulations over a range of various parameter values. Consequently, the model is deterministic and assumes a well-mixed population, neglecting spatial modeling properties altogether [158]. For the purposes of this dissertation, we restrict our focus to spatiotemporal, mixed-population models of cancer metabolism.

2.1.2.7 *Spatially-resolved models*

We further restrict our focus to individual-based, spatially-resolved, diffusive models that can represent the gross metabolic phenotypical properties measured by Jain, *et al.* [140, 72], namely distinct consumption and release profiles, and particle types with distinct diffusion rates. This brings us to the simulation framework developed by Cleveland, *et al.* [23]. They examine from a game theoretical perspective the population dynamics of "cooperator" and "cheater" cells under metabolic stress conditions and high spatial heterogeneity. In general, cooperators obey rules of communal survival, and cheaters do not. In a cancer setting, cooperators are the highly adapted and differentiated cells that make up the body under normal conditions, while cheaters are the rapidly proliferating cells inside a tumor. The authors are not interested in how cheaters become cheaters, but instead seek to examine the dynamics once the defection to cheat has occurred. Their ultimate aim is to understand the movement and growth of a mixed tumor cell population in a complex landscape where metabolic stress is a strong function of position. They draw upon their prior work to use a simple bacterial model to gain insights into the evolution of resistance to drugs under competitive and metabolic stress conditions.

In three ways their approach is similar to that of Kareva. First, they cast the interaction dynamics between the two cell types—be they wild-type or GASP (Growth Advantage in Stationary Phase) mutant bacteria, or distinct metabolic phenotypes—as a prisoner's dilemma game, governed by a prototypical payoff matrix. Interactions between players are matrix operations composed of: each player's consumption and release rates of the different particle types, local normalized concentrations of those particle types, and the quantified impacts those particle types upon each player. Second, they are interested in modeling "public goods". Here they cite the research

C. Athena Aktipis has done in game theoretic agent-based modeling in spatially-resolved environments [1, 2], which implements a "walk away" strategy where a player cells leave a region after they determine that neighboring cells fail to produce sufficient "public goods"; this allows cooperators to gather together rather than with selfish agents, and thereby the population can avoid annihilating the cooperative subpopulation. Third, they are interested in discovering emerging patterns of cooperation between the two cell types.

In contrast to Kareva, they claim this will happen when the traditional game theoretic framework is modified to account for heterogeneous stress patterns, like in their spatially-resolved simulation; so while Kareva developed a well-mixed, non-spatial ODE model to study this phenomenon, Cleveland, *et al.* believe the spatial modeling properties are elemental to the phenomenon. Here they cite the seminal study of Nowak & May [115]—which discusses the effects of spatial resolution on the evolution of cooperation—to make a point that spatially-resolved models and their well-mixed counterparts often produce very different outcomes: diversity and coexistence result from spatial models, while homogeneous populations result from well-mixed models. They also cite a study by Kerr, *et al.* [85], to make the related point that in spatially-resolved models, fitness is determined locally by neighborhood interactions and stresses, rather than globally by uniform stresses; and their simulation results concur with those two studies that the more localized fitness is determined, the more cooperative the outcome, while the more globally fitness is determined, the more zero-sum the outcome.

Despite the appeal of implementing advanced strategies like "walk away," which require agents explicitly model their migration, the authors found that approach to be more deterministic than what their present Markovian approach based on pure

statistics aimed to implement, though they did impose spatial gradients of externally applied "public goods" in their model by introducing special "reservoir" cells that function like vessels.

2.1.3 *Our materials & methods*

We found the authors' simple, Markovian, spatially-resolved simulation framework to be well suited to our needs for a fast, minimal simulation of a metabolically and spatially heterogeneous cell population with diffusible metabolites, gasses, and signaling molecules, where we might see important emergent phenomena such as necrotic core formation, and stable local regions of hypoxia like we observe in xenographed hypoxic tumor histology. We know their framework can implement simple game theoretic strategies and give rise to emergent cooperation—as evinced in their study of coexisting subpopulations—so we can exploit this extensibility in future work that explores evolutionary game theory [134, 6, 135, 5, 66, 4, 113, 114, 53] and signaling games [95, 133]. The 3D lattice data structure is simple, regular, and easy to interrogate for the purposes of feature measurer and feature integrator modules we can later implement to detect emergent phenomena. Using the Cleveland, *et al.* framework as a starting point, we then extend it in significant and novel ways to further suit our needs. First, we can support any number of cell types and any number of particle types (each with its own diffusion rate). Second, each cell type has default behaviors, as before, and conditional behaviors, which can implement phenotypical adaptations and mutations, and state machines composed of two or more cell types. Third, initial, and upper- and lower-bounded basal concentrations can be set for each particle type. Fourth, each cell type can be replaceable or not, and reproductive or

not. Fifth, initial lattice occupation can be delayed to establish complex diffusion gradients to form prior to simulation.

2.2 MATERIALS & METHODS

2.2.1 *Cellular automaton*

The universe of the simulation is a 2D or 3D cellular automaton [148]. Hereafter, for the sake of defining the simulation, we shall assume a 3D configuration, though many of the examples appearing on the page will naturally lend themselves better to a 2D presentation. A cellular automaton works according to the following principles. Each box inside a regular 3D lattice represents a cell, whose position is specified by a three-tuple, (i, j, k). Each cell is one of a finite number of states. For simplicity, let us assume these states are "on" and "off." Each cell has a well-defined neighborhood of adjacent cells, where neighborhood can be defined in a flexible way, most commonly immediate neighbors. All cells are initialized to some state at time = 0. Then, at each subsequent time step (time = 1, 2, ...), the cells' states are updated according to fixed rules that determine the new state of the cell as a function of the cell's current state and those of its neighbors. The state update rules are applied uniformly and simultaneously to the whole lattice of cells. In this way, the cellular automaton evolves as a whole, and patterns in the population of cells emerge over time that cannot be predicted without performing the requisite computation.

2.2.2 *The spatial simulation*

In the context of simulating large, heterogeneous cell populations, we need a flexible and extensible framework. We first specify the dimensions of our universe as `x_dim`, `y_dim`, and `z_dim` parameters—running examples will execute on a 40 x 40 x 40 lattice—and the total running time of the simulator, where time steps are in arbitrary time units, and can be scaled using a time parameter, τ. We want to simulate M types of cell: $T = \{\nu, \varepsilon, \alpha, \beta, \gamma, \delta, ...\}$, where ν and ε are special types related to a blood vessel and empty (unoccupied) space, respectively. We want to simulate N types of diffusible particles that cells can consume and release. At any given time, each (i, j, k) is occupied by some $t \in T$. Each cell type, $t \in T\backslash\{\nu, \varepsilon\}$, has a two sets of parameters that specify its behavior.

First are the default parameters. These include: $c_{t,p}$, the consumption rate of particle type p; $r_{t,p}$, the release rate of particle type p; $\sigma_{t,p}$, the impact factor of each particle type p; whether or not t is replaceable; and whether or not t is reproductive. These can be thought of as implementing the cells' genotypical (native) behaviors, as viewed from an outside, cell population perspective.

Second are the conditional parameters. These are formulated as trigger-action pairs. A trigger is a set of one or more predicates that are based on particle concentrations, as measured by the cell in its locality. All trigger predicates must be true to execute the associated action. An action is a set of commands which are executed sequentially. The possible commands include: apoptosis; become ("jump to") another cell type, $t' \in T\backslash\{\nu, \varepsilon, t\}$; set the consumption rate of particle type p to a target value; set the release rate of particle type p to a target value; set the impact factor of particle type p to a target value; set the Boolean condition of being replaceable; and set the Boolean

condition of being reproductive. This conditional degree of flexibility grants us the ability to implement cell mutational events (the action to become another cell type) and the cells' phenotypical response behaviors (all of the actions), again as viewed from an outside, cell population perspective.

The initial cell population in the lattice constitutes another set of parameters. The default cell type for the lattice is empty (ε). One can manually specify cell types at individual locations, or can algorithmically do this, using arbitrary functions, including stochastic ones, to specify the initial population. An initialization delay parameter specifies when the all-empty lattice is replaced by its initial configuration. Its default value is 0, but can be set to any future time step, to allow, for example, one to establish a concentration gradient (or set of gradients), that may take many time steps, prior to introducing the cells into it.

Each time step drives the simulation through four phases. Particles are consumed and released according to the cell type's consumption and release rates, respectively, for that particle type. Particles continually diffuse in $\mathbb{R}^3$ according to the particle type's diffusion rate. A cell's fitness is first individually computed as a function of impinging concentrations of the particles in combination with the cell type's impact factors corresponding to each particle type. Then its fitness is computed from the fitness scores of individuals in its neighborhood, according to their cell types. This defines a distribution that will be statistically sampled to determine what cell type each lattice location will contain in the next time step.

2.2.3 *Phase 1: consume and release particles*

Let P denote the set of particle types, and $\rho_p(i,j,k)$ denote the normalized concentration of particle type $p \in P$ at location (i,j,k). Concentrations of particles evolve at each time step by the following. For each particle type $p \in P$, for each cell type $t \in T \backslash \{\nu, \varepsilon\}$, set

$$\rho_p(i,j,k) = \rho_p(i,j,k) \cdot (1 - c_{t,p} + r_{t,p}). \tag{1}$$

Note that cell type ν (vessel) has some special default properties related to particles. These defaults implement an assumption we make that a vessel is a perfect source for certain particles (releasing them to full concentration) and a perfect sink for others (consuming them to zero concentration). That is, if location (i,j,k) contains cell type ν, then $\forall \tau : \forall p \in P, r_{\nu,p} = 1 : \rho_p(i,j,k) = 1$ and $\forall \tau : \forall p \in P, c_{\nu,p} = 1 : \rho_p(i,j,k) = 0$. These default properties can be overridden by specifying non-unity vessel consumption and release rates for each particle type.

The simulator has additional parameters related to particle concentrations. Initial concentrations, and basal upper and lower bounds, can be set for each particle type. For the latter, the simulator enforces the bounds in this phase at each time step: those concentrations falling below the lower bound are set to the lower bound; those rising above the upper bound are set to the upper bound.

2.2.4 *Phase 2: diffuse particles*

At each time step, each lattice point's concentration of the particles it contains is updated according to the diffusion rate, D_p, of each particle type p. Each particle type's concentration field, ρ_p, undergoes an isotropic 3D Gaussian convolution, using a 5 x 5 x 5 mask with $\sigma = \sqrt{2D_p}$. The n-dimensional Gaussian kernel is defined as

$$G_n(\vec{x}, \sigma) = \frac{1}{(\sqrt{(2\pi)}\sigma)^n} e^{-\frac{|\vec{x}|^2}{2\sigma^2}}. \tag{2}$$

We assume input array values outside the bounds of the array are equal to the nearest array border value. We accomplish the convolution using Matlab's `imfilter` command.

Equivalently, consider each individual particle taking a random walk in each of the three dimensions [67]. Let q be a number drawn from a standard normal distribution, $q \sim N(0, 1)$, and τ be a scaled time variable with respect to the simulation's clock tick value. Concentrations of particles diffuse at each time step by the following. For each particle p' of type $p \in P$,

$$\Delta x_{p'} = \Delta y_{p'} = \Delta z_{p'} = q\sqrt{2D_p\tau}. \tag{3}$$

2.2.5 *Phase 3: compute fitness scores*

For each location in the lattice (i, j, k), and for each cell type $t \in T \backslash \{\nu, \varepsilon\}$, let us define a local fitness function

$$f_t(i, j, k) = [\sigma_{t,1} \cdot \rho_1(i, j, k) + \sigma_{t,2} \cdot \rho_2(i, j, k) + ... + \sigma_{t,n} \cdot \rho_n(i, j, k)] \cdot \mathcal{I}_t, \tag{4}$$

where $\rho_p(i, j, k)$ denotes the normalized concentration of particle type p at location (i, j, k); $\sigma_{t,p}$ denotes the impact of particle type p on t; and $\mathcal{I}_t$ denotes the indicator function that is *true* if and only if location (i, j, k) is occupied by cell type t, reflecting the simulator design assertion of exclusive occupancy of one cell type per location. We define $f_\varepsilon(i, j, k) = 0$.

After we compute these individual fitness scores for each lattice location, we use them to decide probabilistically what cell type each location (i, j, k) should contain in the next time step. For each location in the lattice (i, j, k), and for each cell type $t \in T \backslash \{\nu\}$, let us define a neighborhood fitness function

$$F_t(i, j, k) = \frac{1}{N} \sum_{(i', j', k') \in \mathrm{neighbors}} f_t(i', j', k'), \tag{5}$$

where $F_t(i, j, k)$ denotes the probability that the cell at location (i, j, k) becomes cell type t; and N denotes the number of neighbors in the sum. Note that since (i, j, k) may reside on an edge or corner of the lattice, the number of its immediate neighbors is bounded from above by 8 (in 2D) and 26 (in 3D).

2.2.6 *Phase 4: reproduce probilistically*

Each cell's immediate neighbors give a distribution from which to draw the target cell type. Let $S(i,j,k) = \sum_{t \in T \setminus \{v\}} F_t(i,j,k)$. In this scheme, if $S(i,j,k) > 1$, then we normalize it to 1; and if $S(i,j,k) < 1$ then there is some probability that location (i,j,k) will become empty ($t = \varepsilon$) in the next time step.

We accomplish this as follows. Let shrinkage factor $\lambda = \frac{1}{S(i,j,k)}$. For each $t \in T \setminus \{v\}$, shrink each cell type's probability contribution: $F_t(i,j,k) = \lambda \cdot F_t(i,j,k)$. We can represent each $F_t(i,j,k)$ as a subinterval of the unit interval. We place them side by side to cover the unit interval. Then we draw a random number, r, uniformly in $[0,1]$; whichever cell type's interval r resides in determines the target cell type of (i,j,k) for the next time step.

Note that cell type v (vessel) defaults to being neither replaceable (its fate is exempt) nor reproductive (it casts no vote), so it is effectively neutral with respect to this reproduction phase of the simulation. Vessels are static features of the spatiotemporal landscape.

2.2.7 *Computing and plotting statistics*

Between phases 3 and 4, the simulator computes and displays a number of useful statistics for the user. These are organized into a console style grid of plots as follows. Since we can often get a good sense of what is happening by examining 2D slices of our 3D world, and because rendering 3D plots is computational expensive, the dashboard consists of mostly 2D plots, and defaults to showing 2D slices on the

$\frac{z\_dim}{2}$ plane; for a 40 x 40 x 40 simulation, for example, the 2D plots show the plane $z = 20$.

On the *first* row, in column:

1. Spatial organization of all cell types: 2D slice, color coded
2. Time series plot of the wall clock seconds elapsed at each time step (performance diagnostic)

On the *second* row, in column:

1. Spatial organization of individual fitness of cell type 1: 3D scatter plot, color coded, intensity level denotes fitness
2. same as above for cell types 2, ..., $M - 1$
3. Spatial organization of individual fitness of cell type M: 3D scatter plot, color coded, intensity level denotes fitness
4. Time series plot of each cell type's population at each time step, color coded

On the *third* row, in column:

1. Spatial organization of individual fitness of cell type 1: 2D slice, color coded, intensity level denotes fitness
2. same as above for cell types 2, ..., $M - 1$
3. Spatial organization of individual fitness of cell type M: 2D slice, color coded, intensity level denotes fitness
4. Time series plot of each cell type's mean individual fitness ($\pm$ standard deviation) at each time step, color coded

On the *fourth* row, in column:

1. Spatial organization of neighborhood fitness of cell type 1: 2D slice, color coded, intensity level denotes fitness
2. same as above for cell types 2, ..., $M-1$
3. Spatial organization of neighborhood fitness of cell type M: 2D slice, color coded, intensity level denotes fitness
4. Time series plot of each cell type's mean neighborhood fitness ($\pm$ standard deviation) at each time step, color coded

On the *fifth* row, in column:

1. Time series plot of cell type 1's x, y, z extents, color coded
2. same as above for cell types 2, ..., $M-1$
3. Time series plot of cell type M's x, y, z extents, color coded
4. Time series plot of each cell type's whole-image Euler-Poincare characteristic (see Chapter 3) at each time step, color coded

On the *sixth* row, in column:

1. Particle type 1 concentration: 2D slice, mesh plot
2. same as above for particle types 2, ..., $N-1$
3. Particle type N concentration: 2D slice, mesh plot

Figure 7 and Figure 8 show these as they appear in the simulator console for 2D and 3D simulations, respectively.

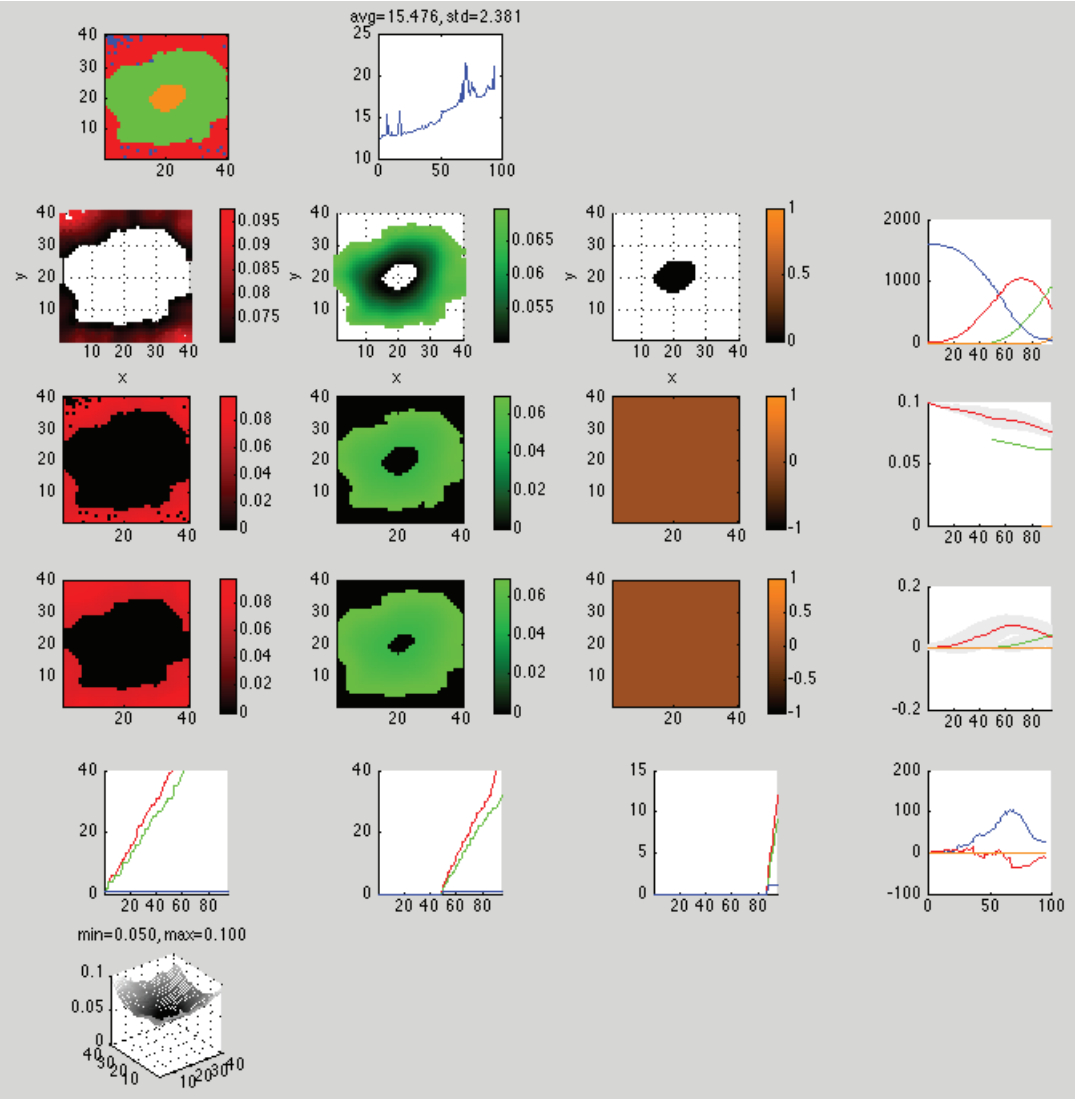

Figure 7: Simulator console displaying an evolving necrotic core in a 40 x 40 lattice. The simulation consists of four cell types—*empty*, *viable*, *hypoxic*, and *necrotic*—and one particle type, O_2. In row 1, column 1, the cell population at this point in the simulation consists of all cell types. In rows 2-5, columns 1-3, all cell types except *empty* have fitness and other spatial statistics reported. In row 6, the concentration of O_2 is reported. In column 4, rows 2-5, aggregate cell type statistics are reported in time series.

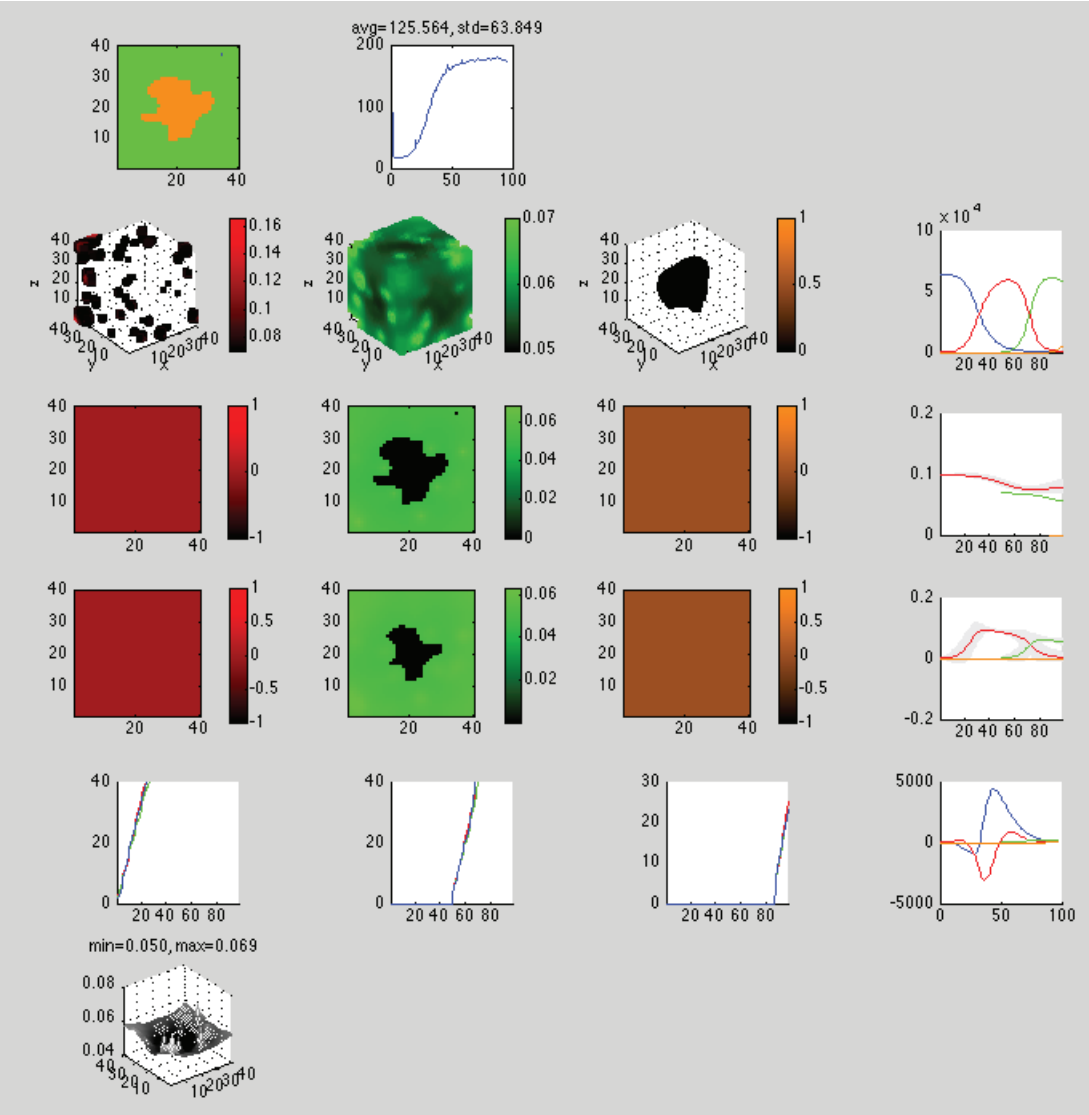

Figure 8: Simulator console displaying evolving regions of stable hypoxia with many vessels in a 40 x 40 x 40 lattice. The simulation consists of five cell types—vessel (white), *empty*, *viable*, *hypoxic*, and *necrotic*—and one particle type, O_2. In row 1, column 1, the cell population at this point in the simulation consists of *hypoxic* and *necrotic* cells. In rows 2-5, columns 1-3, all cell types except vessel and *empty* have fitness and other spatial statistics reported. In row 6, the concentration of O_2 is reported. In column 4, rows 2-5, aggregate cell type statistics are reported in time series. Components of the console displaying 2D plots show the plane $z = 20$.

2.2.8 *Code*

The simulator was coded in Matlab. The complete code listing is given in Appendix B. It depends on three modules: Colormap and Colorbar Utilities[2] (to stabilize colormaps in multi-plot figures), freezeColors/unfreezeColors[3] (to stabilize colorbars in multi-plot figures), and Geometric Measures in 2D/3D Images[4] (for computing Minkowski functionals and the Euler-Poincaré characteristic—see Chapter 3).

2.3 RESULTS & DISCUSSION

A large set of simulation results and related discussions are listed in Appendix A.

2.4 CONCLUSIONS & FUTURE WORK

2.4.1 *Conclusions*

2.4.1.1 *Our contributions*

We have created a spatially-resolved, mixed-population simulation which is minimal, fast, extensible, and adaptable. First, we can support any number of cell types and any number of particle types (each with its own diffusion rate). Second, each cell type has default behaviors, as before, and conditional behaviors, which can implement

2 http://www.mathworks.com/matlabcentral/fileexchange/24371-colormap-and-colorbar-utilities-sep-2009
3 http://www.mathworks.com/matlabcentral/fileexchange/7943-freezecolors-unfreezecolors
4 http://www.mathworks.com/matlabcentral/fileexchange/33690-geometric-measures-in-2d3d-images

phenotypical adaptations and mutations, and state machines composed of two or more cell types. Third, initial, and upper- and lower-bounded basal concentrations can be set for each particle type. Fourth, each cell type can be replaceable or not, and reproductive or not. Fifth, initial lattice occupation can be delayed to establish complex diffusion gradients to form prior to simulation.

2.4.1.2 *Our findings*

Regarding the simulation, we find that with a few simple ingredients—space; distinct particle types with their own diffusion rates; distinct cell types with default consumption and release profiles; and conditional logic to implement cell-type-specific local adaptation—we can capture a number of interesting features and phenomena. First, in Section A.3, we observe sustained coexistence of two populations; one is dominant but does not drive the other to extinction. Second, in Section A.6, we observe emergent spatial self-organization among *hypoxic* and *aerobic* cells into a stable striation pattern (without conditional logic), followed by population size rebalancing. Third, in Section A.7, we are able to implement a functioning system of autocrine and reciprocal paracrine signaling. Fourth, in Section A.9 and Section A.12, we observe emergent 2D and 3D necrotic cores, respectively. Fifth, in Section A.10, Section A.11, Section A.13, and Section A.14, we observe the emergent formation of local regions of spatially and numerically stable *viable-hypoxic* cell populations that are concentrically oriented, in 2D and 3D, over different vascular densities. In terms of relative orientation, composition, and dimensions, these simulated formations are similar to what we observe in the anti-pimonidazole stain images. This is especially true where the randomized vasculature is more dense, thereby breaking diffusion symmetries and giving rise to more realistic *viable-hypoxic* agglomerations.

Regarding the biology of hypoxia, in Section A.10, Section A.11, Section A.13, and Section A.14, we find there is an instability in the *viable-hypoxic* cell population balance. Either over-oxygenation eventually converts all *hypoxic* cells to *viable* cells, or under-oxygenation does the reverse and then eventually these *hypoxic* cells become *necrotic* cells. The oxygenation balance derives from three rates related to O_2: vessel release rate, the local population's average consumption rate, and the diffusion rate. There seem to be two possibilities: (1) the histology shows a delicate balance that is stable, and therefore in evidence everywhere; or (2) it is a transient phenomenon, and our histology happens to have caught one early stage. But which is it? In other words, what is the relationship between tumor age and average intra-tumor hypoxia? We know tumor age correlates positively with degree of vasculature, and therefore density of oxygenation, so we can perform experiments to address this question.

2.4.2 *Future work*

2.4.2.1 *To interface*

With respect to the overarching aims of this dissertation, once we have a stable characterization of hypoxia in terms of spatiotemporal features, either as a spatiotemporal logical proposition or as a learned similarity score function, then we must implement feature measurers correspond to each proposition feature predicate, or to each score function feature, respectively. These measurers will perform live measurements on the evolving simulator data structures and report their Boolean or numerical results to a live integrator/detector function that logically or functionally relates them, respectively. In the case of the spatiotemporal logical proposition, upon an integrated truth value, the simulation will terminate and return true; otherwise it will run to

the end and return false. In the case of the similarity score function, the integrator's high water mark will be maintained throughout the course of the entire simulation, then returned as a numerical value in [0,1]. We assume these modules would be embedded in the simulator for fast execution.

One likely set of feature measurers, related to 2D and 3D Minkowski functionals, including the Euler-Poincaré characteristic, is already implemented [93] as the *im-Minkowski* Matlab library. Once specified, the remaining feature measurers, so far as they could not be trivially obtained by direct interrogation into the simulator's data structures, would require implementation.

Aside from this, we would need to modify the simulator's functional interface to accept a vector of arguments that codify initial condition parameters, and default and conditional operational parameters.

2.4.2.2 *To extend and enhance*

We consider three extensions to our conditional logic handling that would enhance the simulator.

First, implement a trigger-true temporal predicate. For convenience, let us illustrate using an English language example: "If a *viable* cell's local concentration of oxygen is less than 0.05 for more than 15 clock ticks, then jump to *hypoxic*." In this example, each *viable* cell would have a local clock associated with each trigger set, including one for "local concentration of oxygen is less than 0.05". If the local concentration were to become less than 0.05, then that clock would begin running, and would run so long as the local concentration of oxygen stayed less than 0.05. If that clock reaches 15 ticks, then that *viable* cell would become *hypoxic*; otherwise, if before that local clock were to reach 15 clock ticks, the local concentration of oxygen were to become

greater than 0.05, then that clock would reset and become inactive. Modeling duration of certain local conditions, like oxygenation, would enable affected cells to adapt their behavior according to whether pO_2 levels exhibit "chronic," vessel-dominant, radially-distributed hypoxia, or "intermittent," periodic, fluctuating hypoxia, as discussed by Cárdenas-Navia, *et al.* [20].

Second, implement a cell-age temporal predicate. For convenience, let us illustrate using an English language example: "If a *viable* cell's local concentration of oxygen is less than 0.05 and the affected cell is more than 100 ticks old, then jump to *hypoxic*." At birth, each cell's local age clock would initialize to zero, then advance along with the global simulator clock. In this example, if the local concentration of oxygen were to become less than 0.05, then that cell's local age clock would be consulted to evaluate the cell-age predicate. If it too is true, then that *viable* cell would become *hypoxic*. Modeling local adaptations as a function of cell-age, like sensitivity to certain gradients, may capture some significant features related to the emergence of hypoxia.

Both of these temporal predicates depend on cell's having a unique identity, which our simulator does not use. So we would have to implement this too.

Third, implement probabilistic actions. For convenience, let us illustrate using an English language example: "If a *viable* cell's local concentration of oxygen is less than 0.05, then jump to *hypoxic* with a probability of 0.6." Or "...then jump to *hypoxic* by drawing from a Beta distribution, parameterized by...." Implementing this is trivial and would immediately give the simulator an added dimension of stochasticity, to better model natural degrees of variance in a cell population.

Another such dimension is easy to add. Instead of representing each cell type's default parameters as constants, we could represent them as, say, mean-variance pairs. This way, each newly created cell of that type would draw its parameter values from,

say, Gaussian distributions. This too would better model natural degrees of variance in a cell population.

Given the current state of the conditional logic handling, we could attempt to crudely implement certain phenomena like angiogenesis as, say, a probabilistic jumping from *empty* to *vessel*, given sufficient concentration of VEGF particles.

2.4.2.3 *To explore*

We would like to explore a number of research directions related to our modeling and simulation.

First, our findings related to local, stable regions of hypoxia, namely the balance of rates, points to the importance of vessel density in the tissue. This compels us to explore modeling neovasculature as an explicit growth process related to tumor growth. There is a broad literature on neovasculature modeling, for example, branching and anastomosis [13], and among the multiscale models [32]. Vessel cells should respond to appropriate growth factor signaling from tumor cells, like VEGF signaling, and its growth should be geometrically constrained to one-dimensional branching embedded in three dimensions, oriented along these growth factor gradients. In addition, vessel flow rates should vary. If embedded in a lattice, like our simulation, then one need not explicitly model vessel volume, since this is abstracted away, but one can easily vary vessel flow rates according to a statistical distribution that reflects *in vivo* physiological norms in the model system.

Second, it may turn out that a lattice-based simulation is too limited to capture properties related to cell crowdedness, which is arguably essential for modeling density-derived control of tumor cell population growth, and emergent geometric and spatial organization. It may also have significance for modeling emerging local,

stable regions of hypoxia. Without an explicit notion of crowdedness, one cannot correctly model contact inhibition or anoikis, for example, and how such mechanisms constrain growth rates and spatial patterns. While one could use our conditional logic apparatus to exploit an as yet unimplemented local-density predicate to crudely model constrained growth, Plank, *et al.* argue that since lattice-based methods implicitly assume uniform density, among other limitations, properly modeling crowdedness requires a lattice-free setting [119].

Third, we may wish to explicitly model individual cell migration. Our simulation presently uses a simple statistical mechanism to implement fitness-based local cell type regional takeover. As mentioned earlier, this implies that individual cells do not possess a unique identity. While distinct cell types can respond to local concentrations, and locally adapt to their environment using their conditional logic, they do so in a manner that ignores their individuality. In other words, our world is lattice-state-centric rather than cell-identity-centric. In the end, this may be too abstract a setting to properly model individual cell migration in a way that can be configured by its own set of parameters. This area too has a broad literature, for example [63, 73, 76]. The relationship between cell migration and emerging local, stable regions of hypoxia is presently unclear to us, but may well prove worth exploring.

Fourth, in a longer time scale, evolutionary dynamics with respect to phenotypical strategies, for example, complex, mixed metabolic strategies, may become important to model. As such, we would like to investigate modeling a mixed metabolic population based on the principles of evolutionary game theory [134, 135, 6, 5, 4, 114]. As we discussed at length beginning in Section 2.1.2.5, there is a broad literature on cancer game theory modeling, but little it seems has been done in the area of metabolic mixed populations. Another related direction is to model emergent signaling conven-

tions and cellular coordination and teamwork based on signaling games [95, 133]. Two particularly interesting dimensions of this are *meaning* and *credibility* in "cheap-talk" games, and the emergence of "neologism-proof" signaling conventions that are robust to "deceptive" signaling [40]. How does a tumor cell population foil this mechanism, to move the game-playing dynamics to a novel, malignant Bayesian equilibrium? As with cell migration, the relationship between game theoretic strategies for coordination, and emerging local, stable regions of hypoxia is presently unclear to us, but may well prove worth exploring. (Bud Mishra and Andreas Witzel, personal communications, 2011-2013.)

To balance matters, let us conclude with an appreciation for parsimony. Although each of these four areas of exploration merit consideration, in the scope and context of modeling emerging local, stable regions of hypoxia, our present minimal model already captures some of the salient spatial and dynamic features of the phenomenon.

3

HISTOLOGICAL IMAGE ANALYSIS AND CHARACTERIZATION OF HYPOXIA

3.1 INTRODUCTION

3.1.1 *Problem statement*

Derive a spatiotemporal characterization of hypoxia in human tumor tissue from a set of histological images.

3.1.2 *Background & literature review*

3.1.2.1 *Biological experiments & histological images*

EXPERIMENTAL PROTOCOL Our study concerns an experiment that demonstrates hypoxia arising in human colon cancer[1]. In this experiment, 2×10^6 human colon cancer cells were injected into both flanks of nude mice. When the tumor volume reached ~1500 mm^3 (~ 4 weeks post-injection), pimonidazole was administered via intraperitoneal injection. Ninety minutes after pimonidazole administration mice were euthanized, the tumors were excised and immediately fixed in formalin. Slides were then

1 I would like to gratefully acknowledge the experimental work performed by Elda Grabocka, a postdoctoral fellow in the laboratory of Dafna Bar-Sagi, in her ongoing research into the relationship between hypoxia and the formation of stress granules. This experimental work provided the tumor section slides—stained by H&E, trichrome, and anti-pimonidazole—from which we took the images that our study depends on for characterization of hypoxia.

prepared from sections 10 μm apart, alternating between H&E and anti-pimonidazole stains.

H&E STAINING Hematoxylin and eosin stain (or "H&E stain") is a common staining method in histology. The staining method applies hemalum, which colors cell nuclei blue; it also colors blue some calcified material. The method then counterstains with an aqueous or alcoholic solution of eosin Y, which colors non-nuclear, eosinophilic structures various shades of red, pink, and orange. In our study, we use H&E stains of the tumor tissue for the primary purpose of locating blood vessels and for discriminating collagen. Blood vessels appear within the boundary of the tissue as open lumens (white) populated with several to many red blood cells (small, bright pink spheroids). Collagen deposits appear as continuous structures (light pink) that infuse the tumor lesions and usually do not extend into the necrotic tissue (lightest pink, with interstitial spacing and much smaller, unenclosed nuclei). See Figure 9 for an example H&E stain image of one of our study's canonical tumor sections.

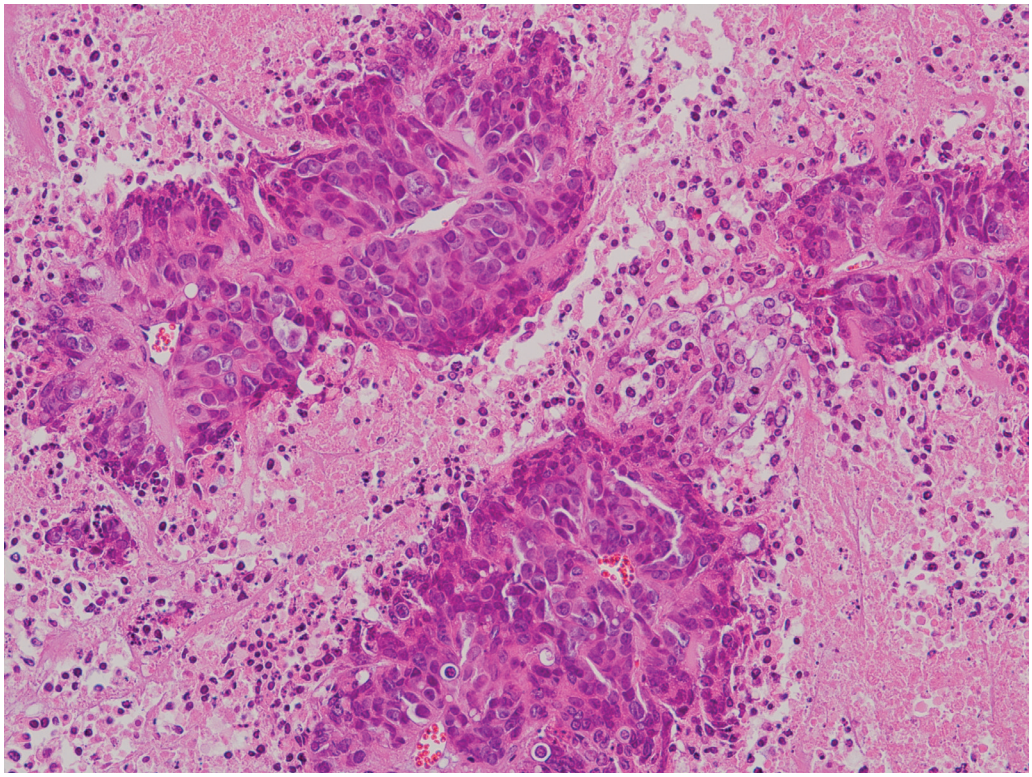

Figure 9: An H&E stain of one of our study's canonical tumor sections.

TRICHROME STAINING Masson's trichrome is a three-color staining protocol used in histology. The usual formulation stains keratin and muscle fibers red, collagen and bone either blue or green, cytoplasm wither light red or pink, and cell nuclei some gradation of dark brown to black. In our study, we use the trichrome stain to verify the presence of collagen in the tumor tissue. In Figure 10, for example, we see three lumen filled with clusters of red blood cells, indicating a transverse sectioning of three blood vessels, and a tumor lesion completely suffused with collagen. This produces a common complication in our study for two reasons that we can

see here: it spatially partitions the lesion, and it may compartmentalize (in 3D) and thereby affect oxygenation within the lesion.

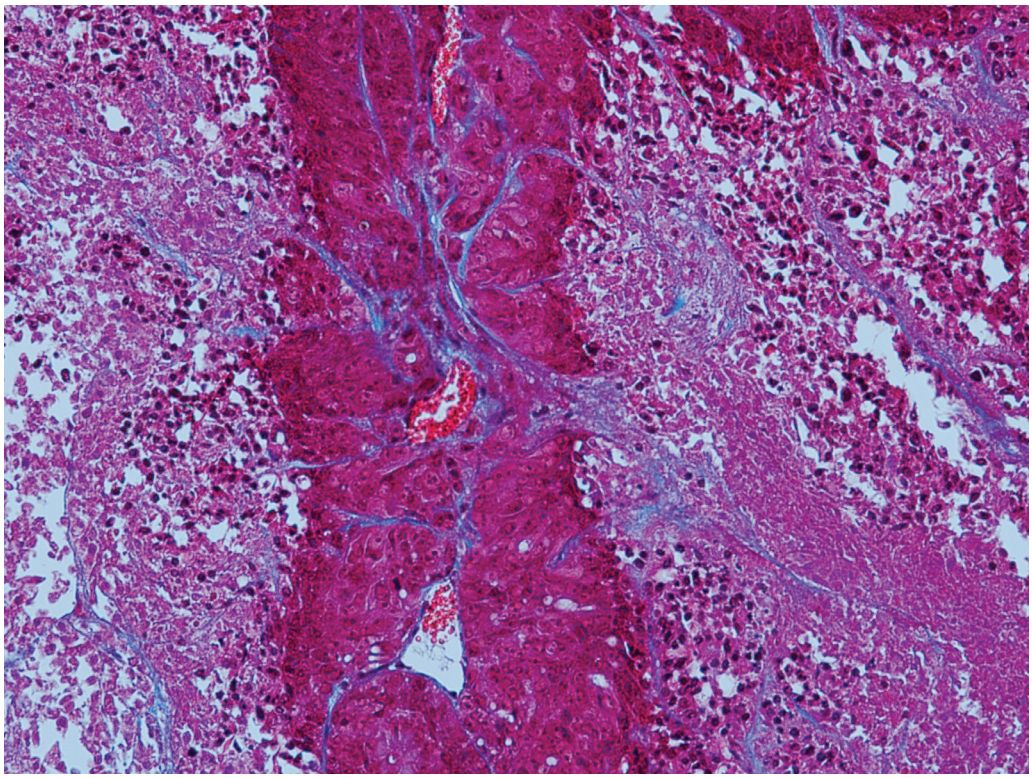

Figure 10: A trichrome stain of one of our study's tumor sections.

ANTI-PIMONIDAZOLE STAINING Anti-pimonidazole staining is an immunohistochemical stain protocol used to detect and locate live cells undergoing hypoxia [144, 126, 89]. In plasma, pimonidazole has a half-life of 25 minutes. It distributes to all tissues following injection, but it forms stable covalent adducts with thiol groups in proteins, peptides, and amino acids, only in those cells that have an oxygen concentration less than 14 micromolar (equivalent to a partial pressure pO_2 = 10 mm Hg at 37 C). In the immunohistochemistry, anti-pimonidazole binds to these adducts allow-

ing their detection. In addition to hypoxic regions in tumors, normal tissues of certain organs such as liver, kidney, and skin possess cells at or below pO_2 of 10 mm Hg; these normal tissues, and only these, will bind pimonidazole. In Figure 11, we see an anti-pimonidazole stain of one of our study's canonical tumor sections. Hypoxic cells stain brown by degree of hypoxia. Notice the blood vessels are much more difficult to locate, though it is still possible. In most cases our procedure to locate vessels is to first manually register the H&E and anti-pimonidazole images (sections of the tumor taken 10 μm apart); second, locate the vessels on the H& stain; then finally use this position on the anti-pimonidazole to approximate the vessel position, or to simply guide a more detailed examination of the anti-pimonidazole image until the vessel can be positively identified. As mentioned earlier, collagen complicates our study structurally and colorimetrically, which can be seen in the figure: collagen is difficult to distinguish from the necrotic tissue that surrounds the lesions. One consequence is that purely intensity-level-based methods of image segmentation will fail to account for the full area of any given lesion suffused with collagen, since it will classify collagen as necrotic tissue and thereby over-partition the lesion into sub-lesions and then downstream analysis will mischaracterize the larger length-scale pattern of oxygen diffusion throughout the larger lesion.

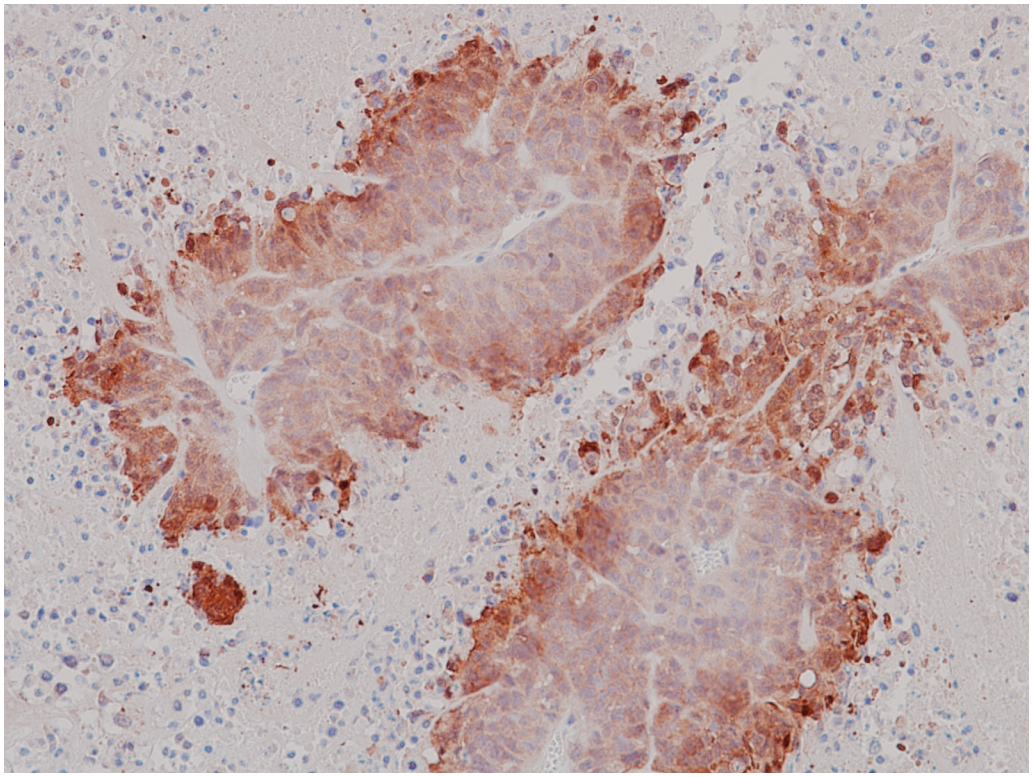

Figure 11: An anti-pimonidazole stain of one of our study's canonical tumor sections.

These types of biological experiments yield histological images of the kind shown above. We are chiefly interested in trying to characterize hypoxia from the anti-pimonidazole stain images. To our knowledge, no methods exist that can do this analysis using histology images like ours.

3.1.2.2 *One relevant study that uses time-series data*

One recent study, by Cárdenas-Navia, *et al.* [20] uses a novel generalization of a live imaging modality to study hypoxia in rat tumor tissue *in vivo*. They are motivated to draw attention to an often-neglected aspect of hypoxia study, namely the pervasive presence of fluctuating oxygenation in tumors. They distinguish between

"intermittent hypoxia," governed by this ambient fluctuation, and "chronic hypoxia," governed by a vessel-dominant oxygenation dynamics. They claim that strategies for mitigating the effects of hypoxic tumor cells were developed under the assumption that chronically hypoxic tumor cells are the central cause of treatment resistance, and so set out to demonstrate how this other paradigm of "intermittent hypoxia" might also explain treatment resistance.

In their study, they used phosphorescent lifetime imaging (PLI) to measure fluctuations in vascular pO_2 in rat 9L gliomas, fibrosarcomas, and R3230 mammary adenocarcinomas. These were grown in dorsal skin-fold window chambers (n=6 for each tumor type), and then imaged every 2.5 minutes for a duration of 60 to 90 minutes. They made a number of important observations. First, continual fluctuation in tumor oxygenation is a prevalent characteristic of these three tumor lines. Second, results in continuous reoxygenation events throughout the tumor. Third, vascular pO_2 maps show significant spatial heterogeneity at each time point, and between time points. Fourth, tumor type affects spatial distribution of oxygen. Fifth, the fluctuations in oxygenation occur with a common slow periodicity (10s of minutes) within and between tumors.

They discuss evidence to support how intermittent hypoxia alters stromal and tumor cells, and that it has important molecular effects. Lastly, they discuss how the spatial heterogeneity in temporal oxygen fluctuations has important implications for optimizing traditional therapies like radiotherapy. If we could visualize areas of fluctuating hypoxia, then high doses of radiation could be delivered to hypoxic areas at a time during which pO_2 values in that area were at the peak of their fluctuations. For this aim, we need further studies to examine spatiotemporal periodicity of tumor oxygenation at timescales relevant to treatment scheduling, like days.

Of course, we are interested in characterizing "chronic hypoxia" situations where there is a steady-state gradient of oxygen from a source vessel, since their stability renders them far more amenable to the length and time scales of our histological evidence. This situation does not preclude our interest in the more spatiotemporally dynamic processes elucidated by Cárdenas-Navia, *et al.*. In fact, as a future endeavor (but not in the scope of this study), we would like to find evidence of reoxygenation patterns in histology. While not taken in time series, histology yields far more structural and state information in the biology than the highly abstracted view that PLI provides per unit time, which is essentially only pO_2 values in space. And it would make their work more relevant to human tumors.

We appreciate the authors' position that previous characterization of hypoxia as perfusion-limited or diffusion-limited are the extreme cases of O_2-delivery-dominant or O_2-metabolism-dominant areas in tumors, respectively, and that most tumor tissue does not distinctly fall into either category; rather, the local pO_2 is heavily influenced by both. After all, the phenomenon of hypoxia as it relates to the heterogeneous metabolic profiles of the tumor population is central to our study. While it may be true that previous characterization of tumor hypoxia as being primarily diffusion-limited does not accurately portray the tumor microenvironment, it is nonetheless a clear phenomenon for which we find evidence in our study, and so we see value in our attempt to seek a spatiotemporal characterization of this "chronic hypoxia."

We also appreciate the attention the authors give to spatial heterogeneity in their analysis. In particular, their use of Moran's I statistic, to express the degree of spatial autocorrelation at each time frame, appears to be a powerful technique of immediate relevance to our study. As a matter of future work, we intend to explore this technique further.

3.1.3 *Our materials & methods*

Our approach consists in extracting qualitative and quantitative features from the histology images, namely the anti-pimonidazol stains. We classify these as: (1) features that derive from segmenting the image into the three tissue types depicted: viable tumor cells, hypoxic tumor cells, and necrotic tumor cells; (2) features related to the intra-lesion hypoxia gradient, as measured from radial distance away from the nearest vessel; (3) features that derive from multiscale analysis; and (4) features that relate to qualitative generalities about bounded and nested structure.

Once we have a set of features, we proceed in two separate but related directions. First, we attempt to construct a logical proposition to describe hypoxia in space and time using an extension of Bounded Linear Temporal Logic (BLTL), whose primitives are image feature predicates. This is a human-driven process, following from human learning and generalization. Second, we attempt to construct a linear regression function that learns what hypoxia is in terms of estimated linear coefficients on the image feature terms. This is a machine-driven process, kept on the rails by a combination of false-positive and false-negative control, and feature dimensionality reduction where possible.

3.1.4 *Literature review of constituent problems*

Given this parsing of our problem statement above into the constituent problems of image feature extraction and logical/functional construction, we should mention research in the literature that relates to them.

3.1.4.1 *Image processing*

IMAGE SEGMENTATION There has been recent progress in automated tumor segmentation on histological images, the best example of which is by Wang, *et al.* [149]. They note that existing research on immunohistochemistry quantification simplify the measurement problem by assuming expert knowledge of tumor areas, which can be used for manual segmentation of tumor cells. They discuss a variety of studies that explore the use of image analysis and machine vision techniques for tissue analysis and biomarker measurement, remarking that robust automated approaches for immunohistochemistry quantification are still under-developed. Although these studies claim to have developed algorithms that measure the intensity and distribution of biomarkers, and do so within the architecture of the tissue samples relevant to them, they all lack empirical validation. Wang, *et al.* develop a robust tumor segmentation technique and test it on H&E and immunohistochemistry stain slides. Their method is comprised of a tissue architecture extraction approach and a tumor texture learning model. The tissue architecture extraction approach uses a stain separation method and an unsupervised multistage entropy-based segmentation method, and the tumor texture learning uses a Markov random field image segmentation system. Their method allows fine pixel based segmentation for small tissue samples. Their tissue domain is human lung tumors. For their purposes they define three classes of tissue morphology: tumor, stroma, and a third catch-all category for lymphoid, inflammatory cells, and necrosis. They report achieving 80% and 78% accuracy[2] on H&E and immunohistochemistry images, respectively. They do not try their method on anti-pimonidazol stain images, which, especially in low concentrations of anti-pimonidazol, render images that have strikingly low contrast. Nonetheless,

2 accuracy = $\frac{(TP+TN)}{(TP+TN+FP+FN)}$

their approach seems to us a promising texture-learning-based alternative to the simple intensity-based means we use to segment high-concentration anti-pimonidazol images. We plan to explore this approach further in future work.

SPATIAL STRUCTURE ANALYSIS

Connectedness The Euler-Poincaré characteristic (EPC), one of the Minkowski functionals [107, 93, 59] , is a measure of structural connectedness (or alternatively, porousness), and it has been used recently in two applications. The first concerns measuring bone density. Rath, *et al.* [120] use the EPC to visualize and assess local trabecular bone structure; and Roque, *et al.* [125] use the EPC to identify low bone density from vertebral tomographic images. The second application is in classifying tumors. Hutterer, *et al.* [71] use the EPC to assign a characteristic signature curve to each AFM image of different tumor types, then use that curve as the basis of a classification method. We are intrigued by the use of characteristic EPC curves, and consider this strategy for learning our image features.

3.1.4.2 *Spatiotemporal logical characterization of biological phenomena*

STATISTICAL MODEL CHECKING AND PBLTL CHARACTERIZATION A number of recent computational studies [77, 162, 54, 55] have employed statistical model checking algorithms to verify spatiotemporal logical propositions in biological systems. They use Probabilistic Bounded Linear Temporal Logic (PBLTL) to characterize phenomena of interest in: a fibroblast growth factor signaling model, circadian rhythm, yeast heterotrimeric G protein cycle control, and the HMGB1 signaling pathway in cancer.

MODEL CHECKING AND LSSL CHARACTERIZATION One study, by Grosu, *et al.* [58] uses model checking with Linear Spatial-Superposition Logic (LSSL) to tackle the problem of learning and detecting emergent behavior in networks of cardiac myocytes. (See our discussion in Section 1.5.) We are encouraged by the success of their approach.

3.1.4.3 *Linear regression functional characterization*

In earlier work [139], in a different image processing domain, we used a linear regression learning method that we have adapted for our application here.

3.2 MATERIALS & METHODS

3.2.1 *Image analysis*

3.2.1.1 *Normalizing image data*

For our initial examination of anti-pimonidazole images, the only selection criterion we applied was to keep to the interior of the tumor, away from its extremities. Since these are xenographed tumors, there are potentially many confounding factors at work near the interface between human tumor and mouse stroma. This is a baseline criterion, applied to all of the images we investigate, regardless of any further stratification. This gave us a set of 20 high-concentration anti-pimonidazole images, taken at 20× magnification, of various regions of the tumor interior. But as we became interested in the role vessels play in oxygenation of the tissue, we decided to further stratify the data, and select just those images whose 10× fields of view are $\geqslant$90% filled with non-necrotic cancer cells, and contain at least one blood vessel. This strat-

ification gave us 8 such high-concentration anti-pimonidazole images, each taken at 10× and 20× magnification, having corresponding registered H&E images from a section 10 μm away.

3.2.1.2 *Image preprocessing*

We shall illustrate image preprocessing and further analysis using two "canonical" images as running examples: Figure 11 and Figure 17 (top). We do this for presentational convenience; our intuitions were developed examining many images, and our methods are applied to all specified images. The first step in our image preprocessing algorithm is to convert the RGB histology image into an 8-bit grayscale image. See Figure 12.

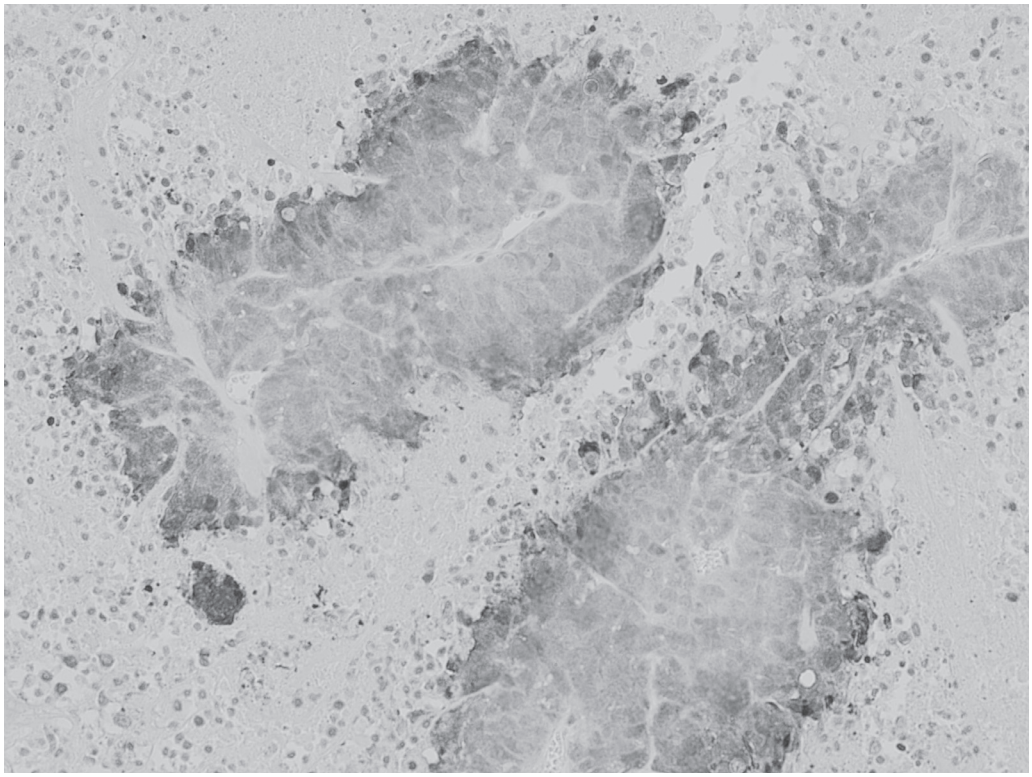

Figure 12: Our canonical image as an 8-bit grayscale image.

RGB TO GRAYSCALE

SMOOTHING Then we apply Gaussian smoothing (using a 5×5 mask and standard deviation of 5.0) iteratively until the high frequency structural information is averaged away (say 100 iterations). See Figure 13. We have used no formal criteria for establishing these parameters, assuming that a consistent protocol for smoothing all images prior to downstream processing is more important than the degree of smoothness. We will address this lack of rigor in future work.

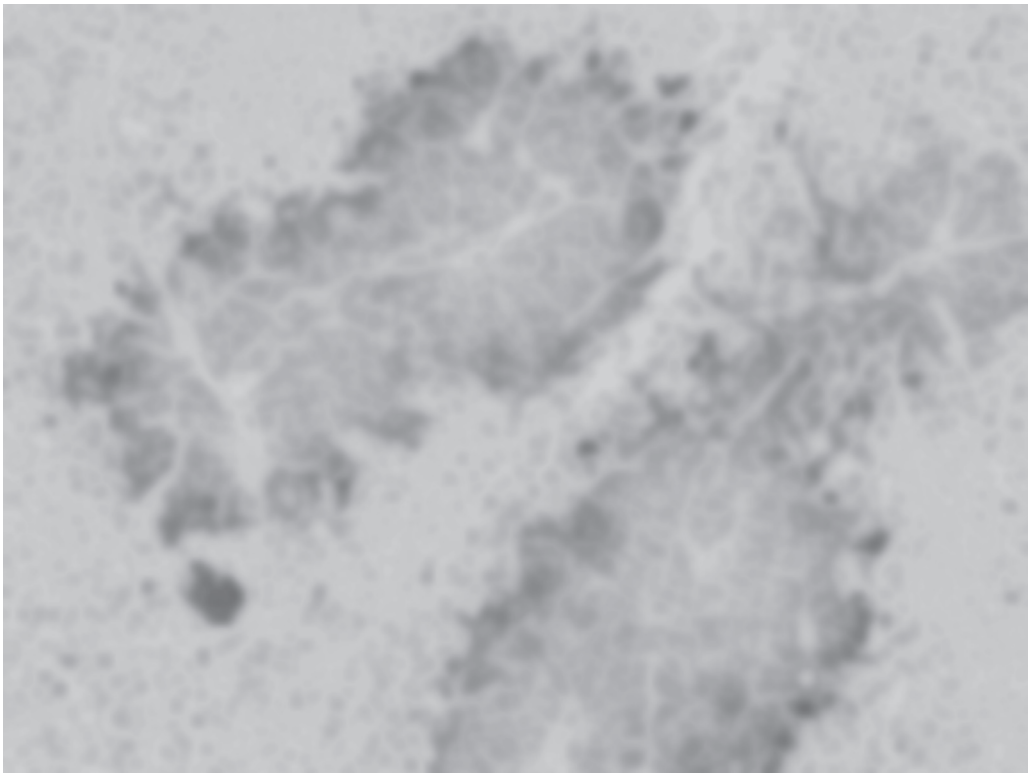

Figure 13: Our canonical image after iterative smoothing.

3.2.1.3 *Toward automated image segmentation*

MESH AND CONTOUR PLOTS When image intensity is viewed as a mesh plot, it is apparent that there are three distinct planes of intensity in the image: necrotic tissue above, hypoxia tissue in the deepest recesses along the outer contour of the lesion, and rising up from that, but not to the height of the necrotic tissue, is the viable (non-hypoxic) tissue. Note the backbone of collagen that runs along the middle of the lesion, and how its intensity levels are frequently indistinguishable from those of the necrotic tissue. More information is given in the contour plot, where the proximity of equipotential curves conveys the steepness of the gradients in intensity. See Figure 14.

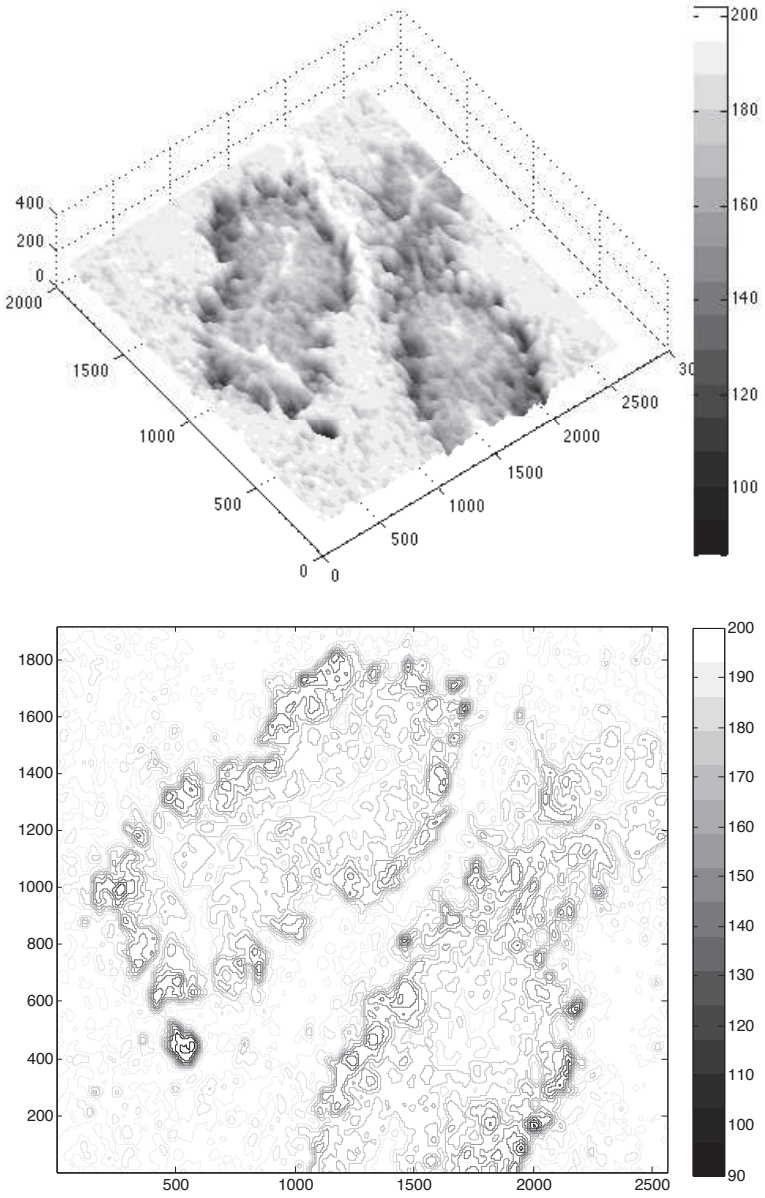

Figure 14: Our smoothed canonical image plotted as a mesh (top) and a contour (bottom).

HISTOGRAM ANALYSIS Despite what our eyes tell us in the mesh and contour plots, when we examine all of the pixels of our canonical image, we see a clear bimodal distribution in the intensity histogram. Yet, when we select a sub-image where we see roughly equal proportions of the three distinct tissue types, a trimodal distribution appears in the intensity histogram. See Figure 15.

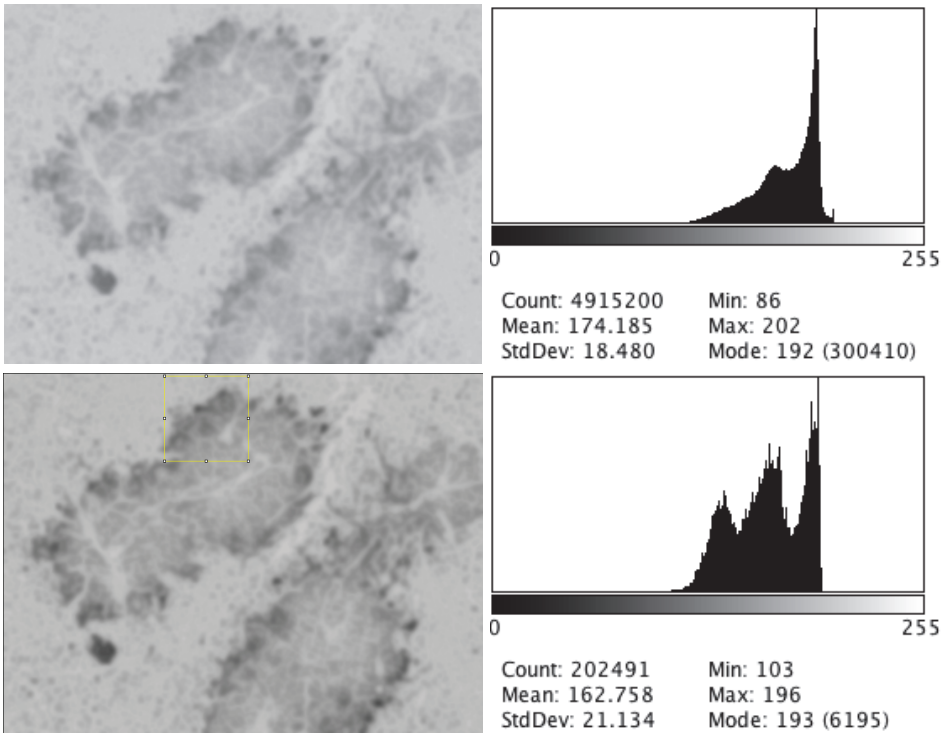

Figure 15: When we examine all of the pixels of our smoothed canonical image (upper left), we see a clear bimodal distribution in the intensity histogram (upper right). Yet, when we select a sub-image where we see roughly equal proportions of the three distinct tissue types (lower left), a trimodal distribution appears in the intensity histogram (lower right).

Using this distribution as a guideline, we proceed to segment our canonical image into three non-overlapping intensity intervals: [0,156] for hypoxic, [157,175] for viable, and [176-255] for necrotic tissue, depicted as red-colored pixels in the top, middle,

and bottom of Figure 16, respectively. Naturally, because sharp thresholds truncate neighboring distributions, false-positive and false-negative cases emerge from this crude approach. In the viable interval we see false-positive outer contours around the hypoxic tissue, and the false-negative inner backbone areas where there are collagen deposits; and in the necrotic interval we see false positive areas where collagen forms an inner backbone that partitions the viable tissue. Since our canonical image is taken from a set of high-concentration anti-pimonidazole images, where the viable-hypoxic distinction is visually and numerically easier to make, we expect this intensity interval partition approach to perform worse on the low concentration anti-pimonidazole images, which it does. Therefore, we cannot recommend it as a general method, despite seemingly high accuracy in some extreme cases of high contrast between the three tissue types' average intensity levels. It follows then that we cannot recommend conducting any downstream processing of segmented components if such components were derived using this method. However, it may still be useful for comparing gross measures of viable-like and hypoxic-like cell areas within a whole image; these might provide a characteristic ratio.

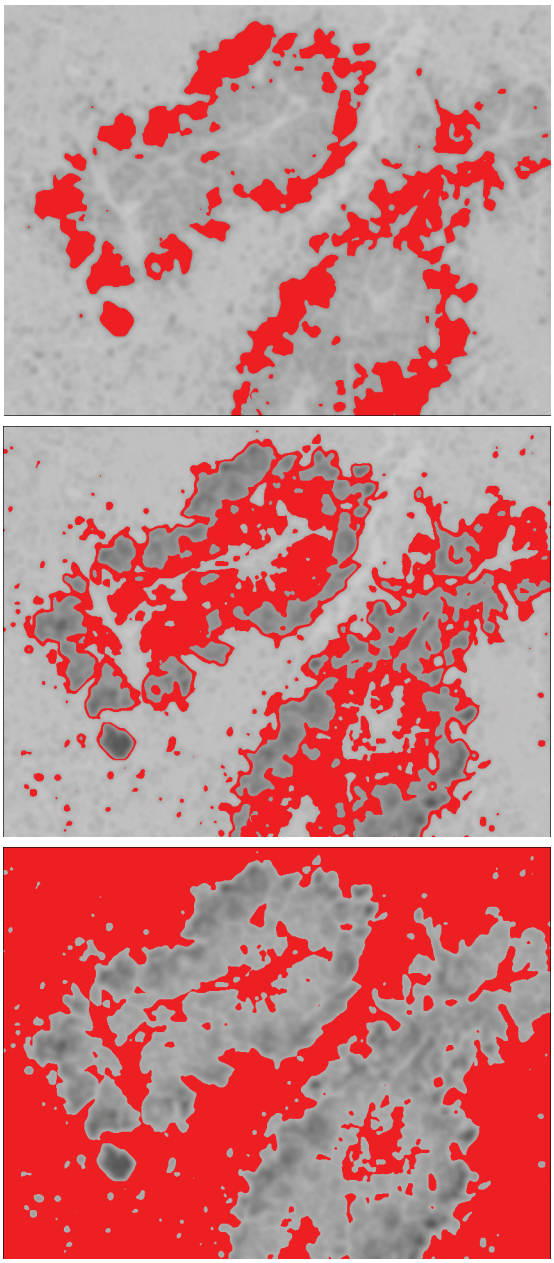

Figure 16: Tissue types by manual segmentation of our smoothed canonical image. Hypoxic tissue, as defined by the intensity interval [0,156] (top). Viable tissue, as defined by the intensity interval [157,175] (middle). Note the false-positive outer contours around the hypoxic tissue, and the false-negative inner backbone areas where there are collagen deposits. Necrotic tissue, as defined by the intensity interval [176,255] (bottom). Note the false positive areas where collagen forms an inner backbone that partitions the viable tissue.

SPATIAL PARTITIONING Next we consider spatial partitioning, where continuous boundaries that separate tissue types are introduced into the image. This requires some degree of familiarity to manually parse these histology images, and so lacks the scalability in the number of images we require for statistical analysis. In Figure 17 we have another canonical anti-pimonidazole image, its manual partitioning, and its labeled partitions. Segmentation by partitioning reveals containment properties of the different regions and leads us to infer which tissue structures are nestable, a subject we shall return to later.

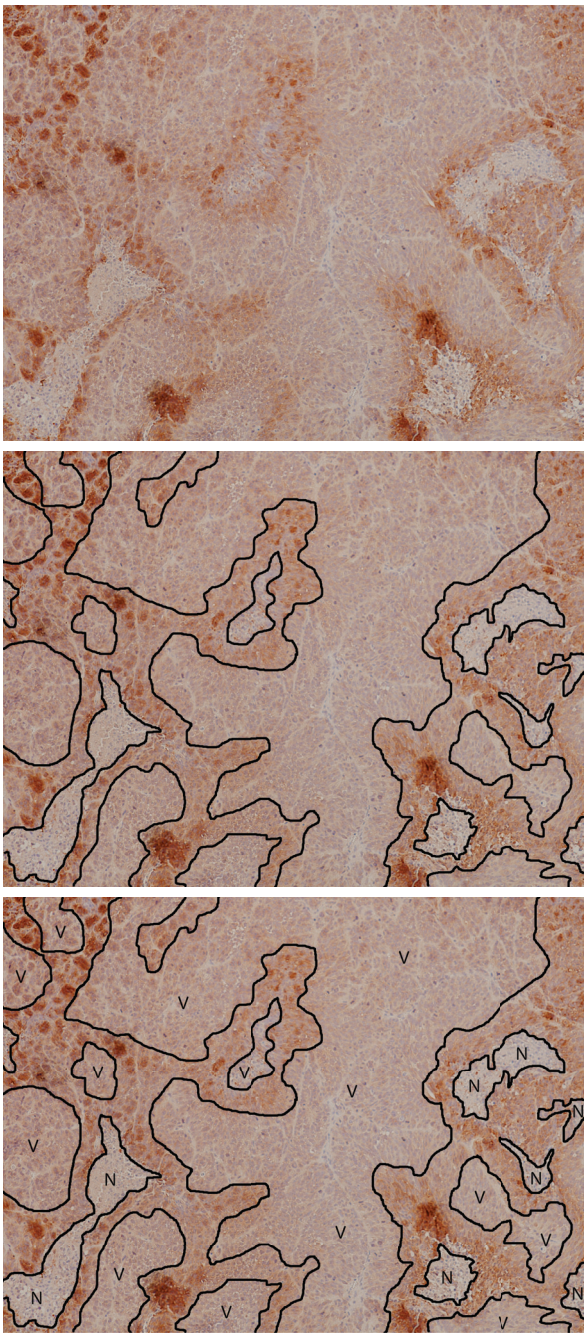

Figure 17: Tissue types by manual spatial partitioning. Another (unsmoothed) canonical antipimonidazole image (top), its manually partitioning (middle), and its labeled partitions (bottom). Key: V = viable, N = necrotic; unlabeled, brown regions are hypoxic.

3.2.1.4 *Toward characterizing gradients*

One of the most salient and consistent features of the anti-pimonidazole images under investigation is the presence of a gradient in the brown stain for hypoxia. In any given lesion, stain density is maximal at the outermost contour of the lesion, abutting necrotic tissue that surrounds it, and then diminishes steadily as a function of distance away from the extremity, toward the center (or central 1D spine) of the lesion. Equivalently, stain density decreases steadily as a function of radial distance away from the center (or orthogonally from the central 1D spine). The central area of a lesion is usually marked by a vessel.

INTENSITY SAMPLE RAY BUNDLES For our gradient measurement analysis, we designed an algorithm to perform radial intensity level sampling, along rays that extend from a given lesion center. One specifies three parameters: a center, (x_c, y_c), usually in the centroid of a blood vessel; n, the number of equal-angle-spaced rays that will sample the circle's area; and m, the number of equal-angle-defined "bundles" (sectors) into which the rays will be considered for statistical analysis. For example, if $n = 80$, then a sample ray will be extended every $\frac{\pi}{40}$ radians, and if $m = 1$, then the rays that fall within 2π radians (all of the rays) will be considered for that bundle's statistical analysis. The image is first smoothed, as before. For a given ray, intensity level is sampled radially, from the inside out, until it encounters the edge of the image. One may specify (as optional parameters) the distance between samples along the ray, d_s in pixels (1 by default), and the square neighborhood radius, r_n in pixels, over which to average for that sample (0 by default since the image is already smoothed). Once the samples have been taken along all of the rays, the rays are "stacked" and "sliced" in the following way. Each ray is an array or inte-

gers, whose index value (in the case of default value of d_s) corresponds 1:1 to pixel distance away form the center. So if we "stack" all of the rays, aligning their array representations by their start index, we will have a measurement matrix, M, that has m rows and c columns, where $c = \frac{l_{max}}{d_s}$, and $l_{max} = \sqrt{x\_dim^2 + y\_dim^2}$, the length of the hypotenuse of the triangle whose right angle sides are the x and y dimensions of the image being sampled. If $d_s = 1$ (by default), then $c = l_{max}$. To see why c takes this value, consider the following extreme case we must be prepared to handle. If we place a center in one corner of the image, then a ray may extend to the opposite corner, requiring l_{max} array locations for its measurements. Given M, we now compute mean, median, and standard deviation along column "slices" of M. This results in $\vec{mean}$, $\vec{median}$, and $\vec{std}$ vectors, whose array representation indices correspond to radial pixel distance away from the center. Since rays have different lengths—they each encounter the edge of the image in a different place, at a different distance from the center from the other rays—they each populate a row of M to a different extent, up to a certain column index; the remaining columns are populated with ∞ so that the part of our algorithm computing $\vec{mean}$, $\vec{median}$, and $\vec{std}$ knows when to drop this ray from the computation. Now we compute the radius of the measurement area, r_m, in the following way. One may specify (as an optional parameter) a threshold length, l_t (defaults to 1000 pixels), over which to locate the global minimum (darkest point) in $\vec{median}$. That is, $r_m = \min_{1 \leqslant i \leqslant l_t} \{\vec{median}(i)\}$. Our algorithm now creates three plots of the data, where the x-axis denotes distance from the center, and the y-axis denotes intensity level. The first shows every ray measurement (various colors), upon which $\vec{mean}$ (blue) and $\vec{median}$ (red) are overlaid; its title gives r_m. The second shows $\vec{mean}$ (blue) $\pm$ $\vec{std}$ (gray), overlaid with segmented least squares fits to $\vec{mean}$ (black); its title gives the length (l), slope (s), and least squares error (e)

for each fitted segment. The third shows $\vec{median}$ (red) $\pm$ $\vec{std}$ (gray), overlaid with segmented least squares fits to $\vec{median}$ (black); its title gives the length (l), slope (s), and least squares error (e) for each fitted segment. The segmented least square fits are given by a dynamic programming algorithm [86], using a cost parameter $C = 200$. We should note now that this entire process is bounded by, and repeated for, each bundle. So for example, if $m = 4$, then $\vec{mean}$, $\vec{median}$, $\vec{std}$, and r_m are computed, and plots are created, for those rays that fall within each successive $\frac{\pi}{2}$ of the circle. Figure 18 shows the circles (red) defined by the r_m found for each of the three centers specified in our canonical image ($n = 80, m = 1$), corresponding to vessel locations in the registered H&E image. The intensity analysis for the three circles' areas is given in Figure 19, Figure 20, and Figure 21. Figure 22 shows the sectors (red) defined by the r_m found for each bundle of each of the three centers specified in our canonical image ($n = 80, m = 8$), corresponding to vessel locations in the registered H&E image. We do not show the corresponding 24 intensity analysis figures.

For our implementation, see the code listing in Section C.1

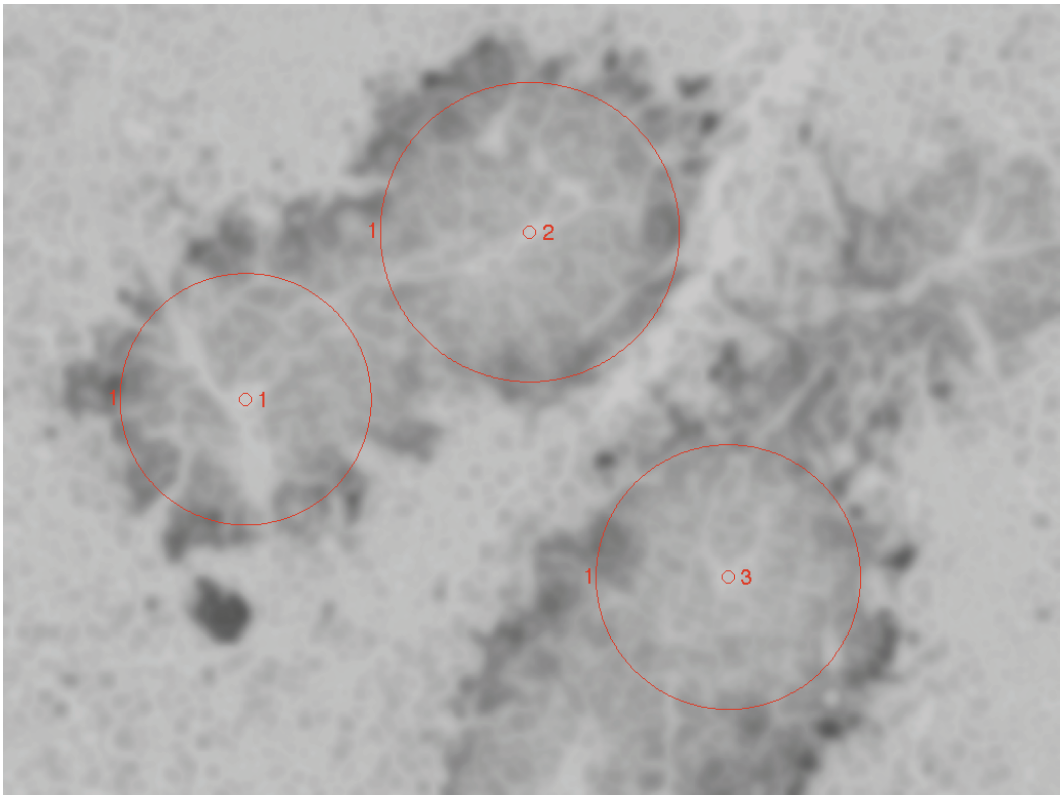

Figure 18: Circles (red) defined by the r_m found by our algorithm for each of the three centers we specifed, corresponding to vessal locations in the registered H&E image.

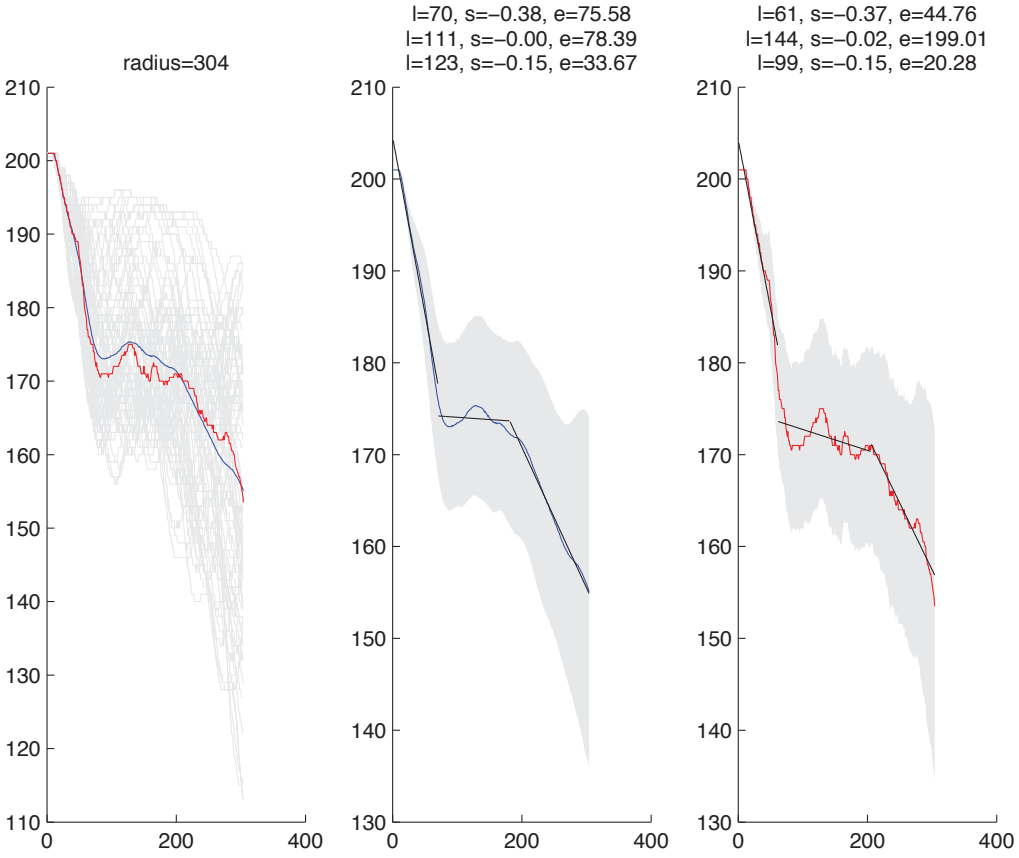

Figure 19: Intensity level analysis produced by our algorithm for center 1.

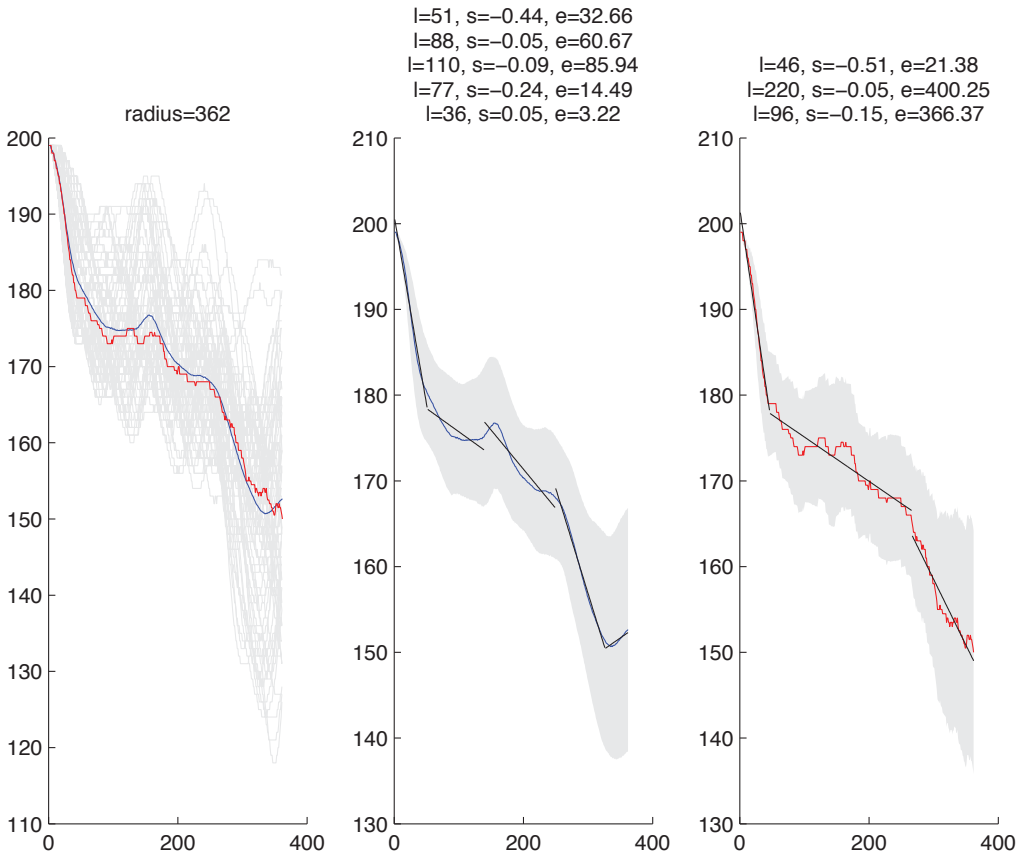

Figure 20: Intensity level analysis produced by our algorithm for center 2.

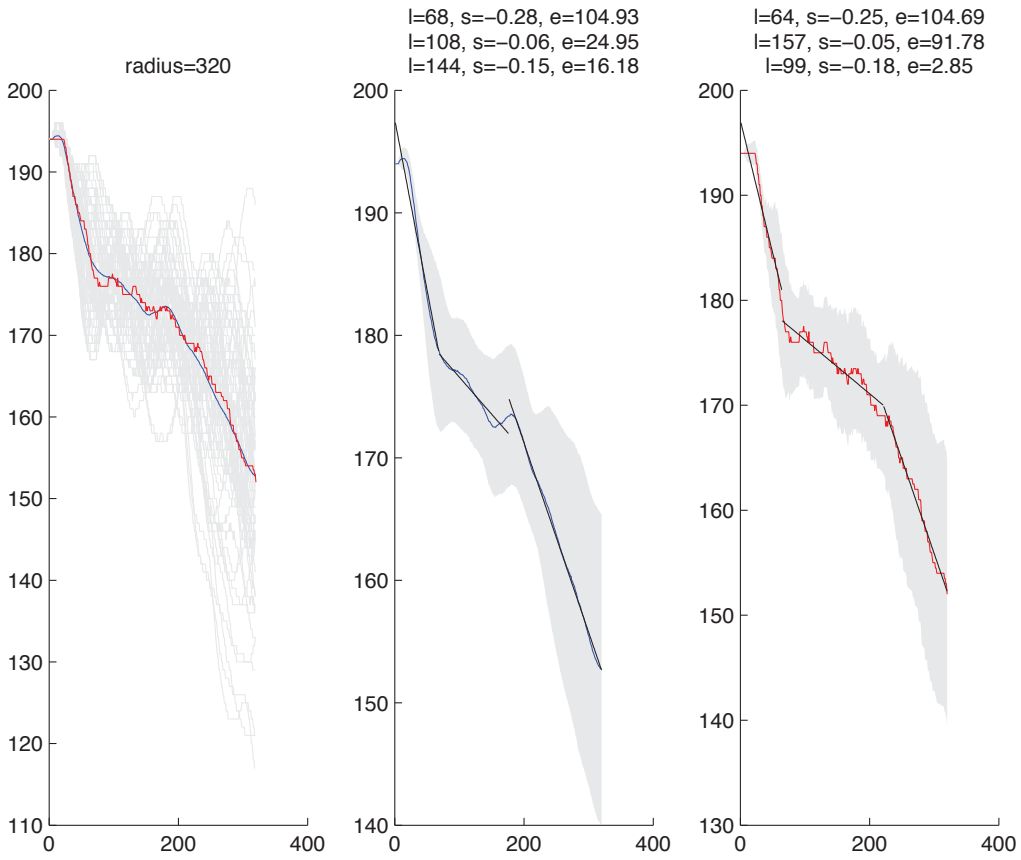

Figure 21: Intensity level analysis produced by our algorithm for center 3.

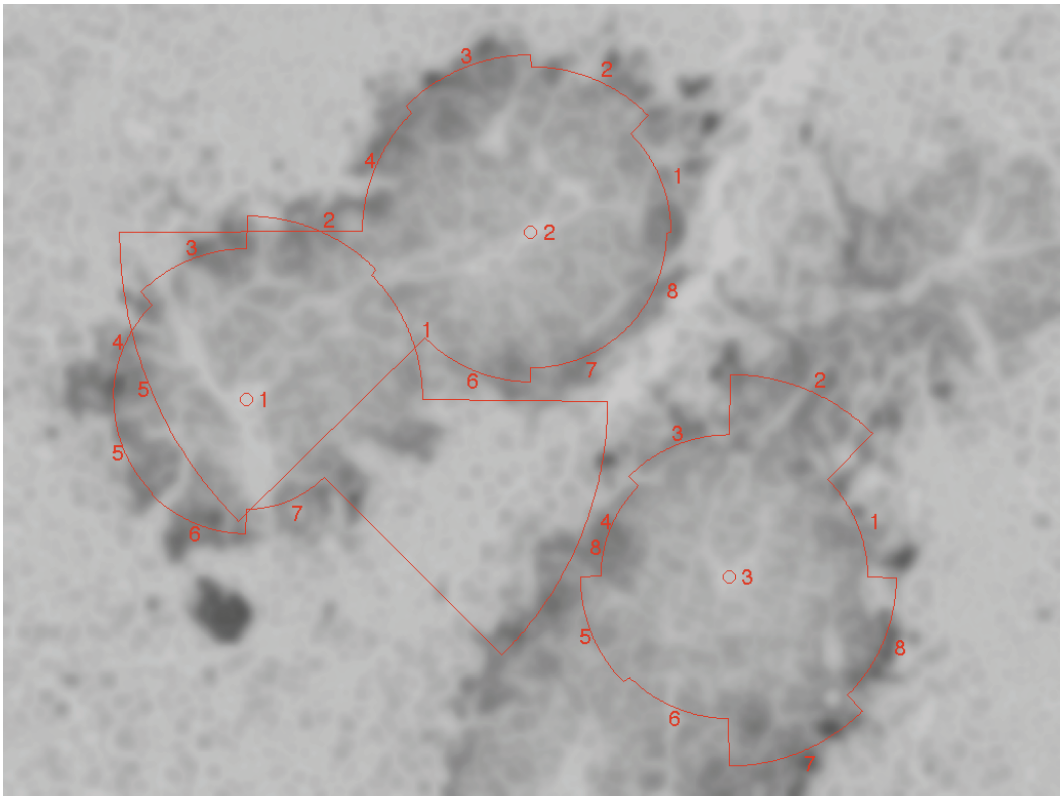

Figure 22: Circle sectors (red) defined by the r_m found by our algorithm for each bundle of each of the three centers we specifed, corresponding to vessal locations in the registered H&E image.

NORMALIZING IMAGE DATA Our first examination of high-concentration anti-pimonidazole images using this method was inconclusive. While it provided evidence for the presence of a gradient following the description above, the slopes of the relevant segments in the linear fit to the mean and median intensity measurements contained too much variation for a meaningful measurement of gradient steepness. It is common practice in many biology experiments to stain tissues using at least two concentration levels. The higher (or highest) concentration functions as a binary test for effectiveness of the stain. Is the phenomenon captured? Did it stain cor-

rectly? Provided that it did, follow up staining is conducted at lower concentrations. In the case of our data set, two concentrations, high and low, were used. Since the high-concentration images might contain excessive contrast, saturating the regions of hypoxia—beneficial for intensity-level-based image segmentation—this may swamp the more subtle gradient signal. We realized that we should attempt the same analysis on a corpus of low-concentration anti-pimonidazole images. For the purposes of measuring gradients, we sought to stratify the data differently than before, and select low-concentration anti-pimonidazole images, taken at $10times$ magnification, that contain one or more complete lesions, each containing one or more blood vessels. This gave us 23 such anti-pimonidazole images, each taken at $10\times$ magnification, having corresponding registered H&E images from a section 10 μm away.

3.2.1.5 *Toward a hierarchical or multiscale structural analysis*

QUAD-TREE RECURSIVE IMAGE DECOMPOSITION To examine the property of intensity variance at different scales in the image, we employ a quad-tree algorithm, adapting it to work with any aspect ratio, not just square images. This works in the following way. For the given rectangle R, consider the set of pixels, P, within it, and the corresponding set of intensity values, I_P. If the $CV(I_P) = \frac{\sigma(I_P)}{\mu(I_P)} > 0.02$ then decompose R into four equal-size rectangles, R_1, R_2, R_3, R_4, and perform the quad-tree algorithm on R_1, R_2, R_3, R_4. This method quickly locates those regions of the image that contain a sufficiently high noise-to-signal ratio. Figure 23 shows the quad-tree decomposition of our canonical image.

For our implementation, see the code listing in Section C.2

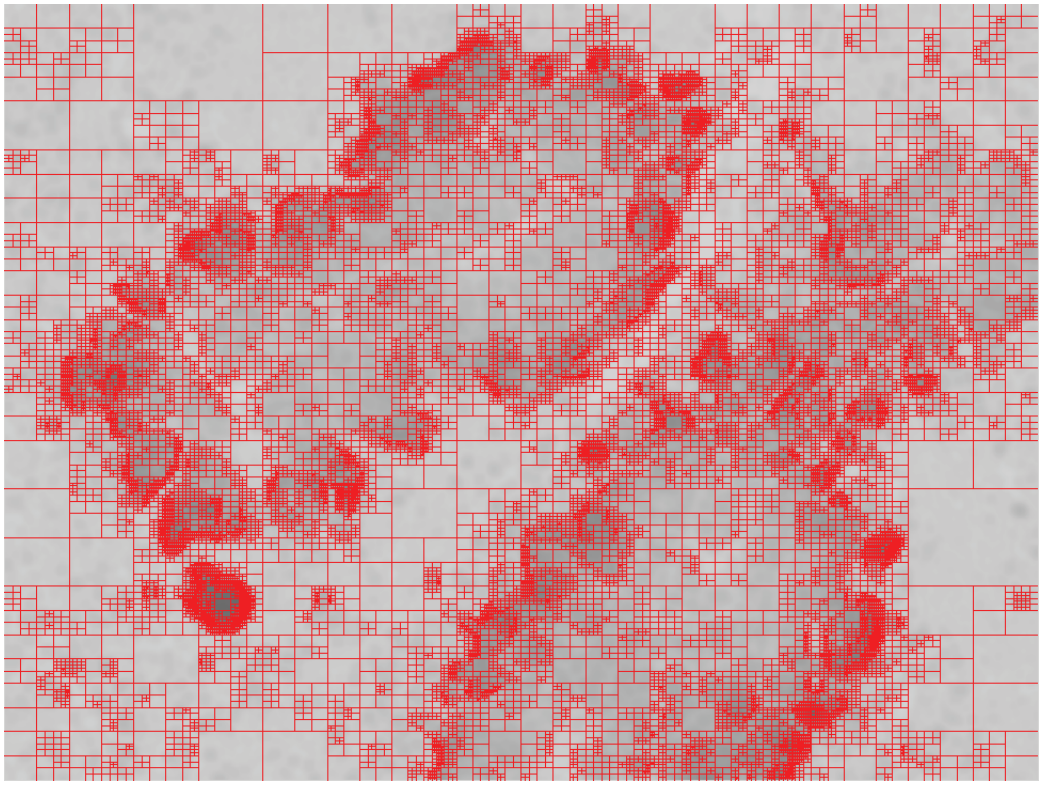

Figure 23: A quad-tree decomposition of our canonical image, where the criterion for decomposition of a given frame is a sufficiently high variation among the frame's pixels' intensity values.

3.2.1.6 *Toward a spatiotemporal grammar of nested structures*

IDENTIFYING REGIONS {V, H, N} As can be seen in Figure 17 (bottom), we can (at least manually for now), segment and unambiguously identify each tissue region in our anti-pimonidazole images: V for viable, H for hypoxic, and N for necrotic tumor tissue. Once we have performed this step on our full set of images, certain qualitative patterns emerge.

CONSISTENT OBSERVATIONS IN IMAGE DATA We observe the following generalities:

- Temporal:
  - 1. N can expand but not contract.
- Spatial:
  - 2. H always precedes N, along its outer contour.
  - 3. H gradient increases in the direction V $\rightarrow$ N.
  - 4. At any given time (in any given image), select a point in the image and proceed in a single direction away from the point; only the following two sequences will be observed
    - a. V $\rightarrow$ H (ascending) $\rightarrow$ N $\rightarrow$ H (descending) $\rightarrow$ V
    - b. V $\rightarrow$ H (ascending, descending) $\rightarrow$ V
  - 5. var(width(H)) $<<$ var(width({N,V})

AXIOMS FOR THE REGIONS These observations lead us to formulate the following axioms.

- A1. invalid-neighbors({V,N})
  - H must separate {V,N}
- A2. reg(V) $\rightarrow$ reg(H) $\rightarrow$ reg(N)
  - regional monotonicity
  - N is absorbing state

VALID SPATIOTEMPORAL GRAMMATICAL TRANSFORMATION RULES From A1 and A2, we can derive the following valid spatiotemporal grammatical transformation rules.

1. V $\rightarrow$ V H V (H origination in V) by A2
2. H $\rightarrow$ H N H (N origination in H) by A2
3. H V H $\rightarrow$ H (V elimination in H) by A2
4. N H N $\rightarrow$ N (H elimination in N) by A2

INVALID SPATIOTEMPORAL GRAMMATICAL TRANSFORMATION RULES From A1 and A2, we can derive the following invalid spatiotemporal grammatical transformation rules.

1. H $\rightarrow$ H V H (V origination in H) by A2
2. N $\rightarrow$ N H N (H origination in N) by A2
3. N $\rightarrow$ N V N (V origination in N) by A2
4. V $\rightarrow$ V N V (N origination in V) by A1
5. H N H $\rightarrow$ H (N elimination in H) by A2
6. V H V $\rightarrow$ V (H elimination in V) by A2
7. N V N $\rightarrow$ N (V elimination in N) by A1
8. V N V $\rightarrow$ V (N elimination in V) by A1

3.2.2 *Spatiotemporal logical characterization*

3.2.2.1 *What does hypoxia look like logically?*

At a high level, we have defined above some quantitative and qualitative image features that we would now like to incorporate into a modal logical proposition about what hypoxia is in space and time. We will apply thresholds to the quantitative features to render them as predicates in our proposition. We will first use these propositions with modal operators in space: S_i = eventually/always X happens in space. Then we will use these spatial modal expressions with modal operators in time: T_j = eventually/always S_i happens in time. This is the way we will build up our final proposition.

3.2.2.2 *Extending Probabilistic Bounded Linear Temporal Logic*

The logic we develop here is an adaptation of Probabilistic Bounded Linear Temporal Logic (PBLTL) [77] that accommodates the three dimensions of space as well as time.

For a stochastic model simulation S, let the set of state variables SV be a finite set of real-valued variables. A Boolean predicate over SV is a constraint of the form $u \sim v$, where $u \in SV, \sim \in \{\geqslant, \leqslant, =\}$, and $v \in \mathbb{R}$. A BLTL property is built on a finite set of Boolean predicates over SV using Boolean connectives and spatiotemporal operators. The syntax of the logic is given by the following grammar: $\phi ::= u \sim v | (\phi_1 \vee \phi_2) | (\phi_1 \wedge \phi_2) | \neg\phi_1 | (\phi_1 U^{\{x_1,x_2,x_3,t\}} \phi_2)$, where $u \in SV, \sim \in \{\geqslant, \leqslant, =\}, v \in \mathbb{Q}$, and $x_1, x_2, x_3, t \in \mathbb{Q}_{\geqslant 0}$. We can define additional spatiotemporal operators such as $F^{\{x_1,x_2,x_3,t\}}\psi = \mathrm{True} U^{\{x_1,x_2,x_3,t\}}\psi$ and $G^{\{x_1,x_2,x_3,t\}}\psi = \neg F^{\{x_1,x_2,x_3,t\}}\neg\psi$ in terms of the bounded until $U^{\{x_1,x_2,x_3,t\}}$. A PBLTL formula is a one of the form $P_{\geqslant\theta}(\phi)$, where ϕ is a BLTL formula and $\theta \in (0,1)$. We say that S satisfies PBLTL property $P_{\geqslant\theta}(\phi)$,

denoted by $S \models P_{\geqslant\theta}(\phi)$, if and only if the probability that an execution of S satisfies BLTL property ϕ is greater than or equal to θ.

Informally, the spatiotemporal operators can be interpreted as follows in this example:

- $\phi_1 U^{x_1,20} \phi_2$ means within 20 spatial units in x_1, ϕ_1 holds until ϕ_2 holds.
- $\phi_1 U^{t,20} \phi_2$ means within 20 time units, ϕ_1 holds until ϕ_2 holds.
- $F^{x_1,20} \phi$ means within 20 spatial units in x_1, ϕ holds.
- $F^{t,20} \phi$ means within 20 time units, ϕ holds.
- $G^{x_1,20} \phi$ means for 20 spatial units in x_1, ϕ holds.
- $G^{t,20} \phi$ means for 20 time units, ϕ holds.

Continuing to follow Jha, *et al.* [77], we define the semantics of our extended BLTL with respect to executions of S. Let $\sigma \models \phi$ denote that an execution trace σ of S satisfies ϕ. Let $\sigma = (s_0, t_0), (s_1, t_1), \ldots$ be an execution of the simulator along states $s_0, s_1, \ldots$ with durations $t_0, t_1, \ldots \in \mathbb{R}$. We denote the execution trace starting with state i by σ^i. The value of the state variable x in σ at state i is denoted by $V(\sigma, i, x)$. The semantics of our extended BLTL for a trace σ^k starting at the k^{th} state ($k \in \mathbb{N}$) is defined as follows:

- $\sigma^k \models x \sim \nu$ iff $V(\sigma, k, x) \sim \nu$
- $\sigma^k \models \phi_1 \vee \phi_2$ iff $\sigma^k \models \phi_1$ or $\sigma^k \models \phi_2$
- $\sigma^k \models \phi_1 \wedge \phi_2$ iff $\sigma^k \models \phi_1$ and $\sigma^k \models \phi_2$
- $\sigma^k \models \neg\phi$ iff $\sigma^k \models \phi$ does not hold

- $\sigma^k \models \phi_1 \mathcal{U}^{x_1} \phi_2$ iff $\exists i \in \mathbb{N}$ such that (1) $\sum_{0 \leqslant l < i} x_{1,k+l} \leqslant x_1$, (2) $\sigma^{k+i} \models \phi_2$, and (3) for each $0 \leqslant j < i$, $\sigma^{k+j} \models \phi_1$.
- $\sigma^k \models \phi_1 \mathcal{U}^{x_2} \phi_2$ iff $\exists i \in \mathbb{N}$ such that (1) $\sum_{0 \leqslant l < i} x_{2,k+l} \leqslant x_2$, (2) $\sigma^{k+i} \models \phi_2$, and (3) for each $0 \leqslant j < i$, $\sigma^{k+j} \models \phi_1$.
- $\sigma^k \models \phi_1 \mathcal{U}^{x_3} \phi_2$ iff $\exists i \in \mathbb{N}$ such that (1) $\sum_{0 \leqslant l < i} x_{3,k+l} \leqslant x_3$, (2) $\sigma^{k+i} \models \phi_2$, and (3) for each $0 \leqslant j < i$, $\sigma^{k+j} \models \phi_1$.
- $\sigma^k \models \phi_1 \mathcal{U}^{t} \phi_2$ iff $\exists i \in \mathbb{N}$ such that (1) $\sum_{0 \leqslant l < i} t_{k+l} \leqslant t$, (2) $\sigma^{k+i} \models \phi_2$, and (3) for each $0 \leqslant j < i$, $\sigma^{k+j} \models \phi_1$.

3.2.2.3 *Deriving* ϕ

Using the extended BLTL, we derive a preliminary spatiotemporal proposition of hypoxia in terms of the hypoxia image/simulator features discussed here.

Suppose we are in a 2D universe.

From the valid spatiotemporal grammatical transformation rules given above, we can immediately write our first proposition term. If we hold x_2 fixed and test along x_1 with respect to $x_{1,0}$, then we expect to encounter tissue types in the following order: $N \rightarrow H \rightarrow V \rightarrow H \rightarrow N$. Suppose we have a primitive state variable function $T : (x_1, x_2) \rightarrow \{H, V, N\}$ that given a coordinate in 2D returns the tissue type at

that coordinate, namely {H,V,N}. In terms of our spatiotemporal logic, this gives the spatial term

$$\begin{aligned}
&(T(x_1, x_2) = N)U^{x_1, x_N} \\
&(T(x_1, x_2) = H)U^{x_1, x_H} \\
&(T(x_1, x_2) = V)U^{x_1, x_V} \\
&(T(x_1, x_2) = H)U^{x_1, x_H} \\
&\quad (T(x_1, x_2) = N).
\end{aligned} \tag{6}$$

Until a morphological analysis provides us with robust measures of tissue type thickness in 2D, we shall leave x_H, x_V, and x_N undefined. Because the valid spatiotemporal grammatical transformation rules apply symmetrically, we can write the analogous spatial term for holding x_1 fixed and testing along x_2 with respect to $x_{2,0}$:

$$\begin{aligned}
&(T(x_1, x_2) = N)U^{x_2, x_N} \\
&(T(x_1, x_2) = H)U^{x_2, x_H} \\
&(T(x_1, x_2) = V)U^{x_2, x_V} \\
&(T(x_1, x_2) = H)U^{x_2, x_H} \\
&\quad (T(x_1, x_2) = N).
\end{aligned} \tag{7}$$

In terms of our gradient feature, suppose we have a primitive state variable function $\nabla_+ : (x_1, x_2, p) \rightarrow (\delta_{x_1}, \delta_{x_2}, \delta\rho_p)$ that given a coordinate in 2D and a particle type p returns the point $(\delta_{x_1}, \delta_{x_2})$ adjacent to (x_1, x_2) that will have the highest concentration of p, returned as δ_{ρ_p}. Thus by extending x_1 with respect to $x_{1,0}$ and x_2

with respect to $x_{2,0}$, along a contour iteratively specified by ∇_+, once we encounter the boundary of $N \to H$, we ought to be able to climb the gradient of p and measure its properties as we proceed from $H \to V$. This gives our next spatial term in the proposition:

$$\begin{aligned}
&(T(x_1, x_2) = N)\mathcal{U}^{(x_1,x_2),(x_N,x_N)}(T(x_1, x_2) = H) \\
&\qquad \wedge F^{(x_1,x_2)\leftarrow\nabla_+(x_1,x_2),(x_H+x_V,x_H+x_V)} \\
&\qquad\qquad (\\
&(F^{(x_1,x_2)\leftarrow\nabla_+(x_1,x_2),(267+126,267+126)}(\nabla_+(x_1, x_2, p) = -0.06 \pm 0.03)) \\
&\qquad\qquad \mathcal{U}^{(x_1,x_2)\leftarrow\nabla_+(x_1,x_2),(x_H+x_V,x_H+x_V)} \\
&(F^{(x_1,x_2)\leftarrow\nabla_+(x_1,x_2),(172+83,172+83)}(\nabla_+(x_1, x_2, p) = -0.21 \pm 0.19)) \\
&\qquad\qquad),
\end{aligned} \tag{8}$$

which in plain language means "We are in necrotic tissue until we are in hypoxic tissue, and then for some length along the contour of our gradient ascent (bounded from above by the expected thickness of hypoxic plus necrotic regions), we track the gradient trajectory characterized by our experimental results below, namely we follow a gradient slope of -0.06 (within bounds) for some bounded length, until we follow a gradient slope of -0.21 (within bounds) for some bounded length." The gradient segment lengths, length bounds, slopes, and slope bounds are given in Table 6. This is merely one provisional gradient term, and not intended to represent a comprehensive gradient characterization in spatiotemporal logical terms.

In terms of our segmentation feature, and the derived ratio of cell types, suppose we have a primitive state function $C : \emptyset \to \mathbb{Q}$ that gives the current ratio of hypoxic to viable cells in some spatially bounded area. Then given the results from our seg-

mentation experiment below (Table 1), namely for the mean H:V ratio, we have the following temporal term, with respect to t_0:

$$G^{t,\tau}(C = 0.36 \pm 0.24), \tag{9}$$

where τ is an upper bound on the time (e.g., the run time of our simulation).

If we assume the three spatial terms above are true for time bounded by τ, then our final spatiotemporal logical proposition characterizing hypoxia is given by:

$$
\begin{aligned}
&G^{t,\tau} \\
&(\\
&(T(x_1,x_2) = N) U^{x_1,x_N} \\
&(T(x_1,x_2) = H) U^{x_1,x_H} \\
&(T(x_1,x_2) = V) U^{x_1,x_V} \\
&(T(x_1,x_2) = H) U^{x_1,x_H} \\
&(T(x_1,x_2) = N) \\
&\wedge \\
&(T(x_1,x_2) = N) U^{x_2,x_N} \\
&(T(x_1,x_2) = H) U^{x_2,x_H} \\
&(T(x_1,x_2) = V) U^{x_2,x_V} \\
&(T(x_1,x_2) = H) U^{x_2,x_H} \\
&(T(x_1,x_2) = N) \\
&\wedge \\
&(T(x_1,x_2) = N) U^{(x_1,x_2),(x_N,x_N)} (T(x_1,x_2) = H) \wedge \\
&F^{(x_1,x_2) \leftarrow \nabla_+(x_1,x_2),(x_H+x_V,x_H+x_V)} \\
&(\\
&(F^{(x_1,x_2) \leftarrow \nabla_+(x_1,x_2),(267+126,267+126)} (\nabla_+(x_1,x_2,p) = -0.06 \pm 0.03)) \\
&U^{(x_1,x_2) \leftarrow \nabla_+(x_1,x_2),(x_H+x_V,x_H+x_V)} \\
&(F^{(x_1,x_2) \leftarrow \nabla_+(x_1,x_2),(172+83,172+83)} (\nabla_+(x_1,x_2,p) = -0.21 \pm 0.19)) \\
&) \\
&) \\
&\wedge \\
&G^{t,\tau}(C = 0.36 \pm 0.24).
\end{aligned}
\tag{10}
$$

3.2.3 *Linear regression functional characterization*

3.2.3.1 *Ergodic assumption*

We assume the tumor hypoxia process that generates our image data to be ergodic: since we see so many instances of lesions, we are likely seeing every temporal state of a typical lesion, and so, in the limit of static images, we observe the *temporal* and *spatial* phenomenon of hypoxia.

3.2.3.2 *Linear regression learning*

We would now like to incorporate the quantitative image features defined above into a linear functional form, whose weights are learned by regression [131], for a lesion hypoxia similarity metric. Our approach here is adapted from earlier work in a different domain [139]. This entails solving an overdetermined system of equations, given by $a_1 f_{1,j} + ... + a_n f_{n,j} = 1$, where the $a_i, i = 1, ..., n$ are the n feature coefficients to be learned and the $f_{i,j}, i = 1, ..., n, j = 1, ..., m$ are the corresponding n feature values over $m \geqslant n$ observations forming the feature matrix, F. We train a linear regression model on m calibrating lesions, having known similarity score 1, using values from the n features, giving $F\vec{a} = \vec{1}$. The model has the analytic solution $\vec{a} = (F^T F)^{-1} F^T \vec{1}$. This gives a trained similarity estimator, $S_T = a_1 f_1 + ... + a_n f_n$.

This formulation of S_T assumes all lesions, i.e. their associated feature values, have equal weight, owing to their equivalent validity as observations. However, such an assumption may be challenged on the grounds that upon taking into consideration the difference between the empirically measured null distribution and the actual shape of the distribution in feature measurements, certain observations appear to be

false positives, and others false negatives—a notion formally addressed by robust regression, namely, the Beaton-Tukey formulation.

3.2.3.3 *Weighting training data to address Type I and Type II errors*

Normally, false positive examples appear as ones that deviate significantly from the null-distribution, and if not discarded, can affect the statistical estimators adversely. However, instead of discarding such outliers using sharp-thresholds, and using the filtered examples in the estimator, one may assign to each data point a positive weight that signifies how likely it is that a particular example is an outlier. Such a weighting scheme could be based on the ideas underlying robust M-estimators—a class of central tendency measures that make them resistant to local misbehavior caused by outliers (e.g., false positives). We adapted the Beaton-Tukey biweight [14]—an iteratively reweighted measure — for this purpose of central tendency. We note that other schemes, such as Huber's M-estimator, could have been used with similar performance. Both the biweight and the Huber weight functions are available in standard statistical packages. Here we use Matlab's *robustfit* command with default parameters (weight function "bisquare," using a tuning constant of 4.685).

M-estimator Θ uses these weights to compute the weighted average of sample points: $\Theta = \sum w_i \cdot x_i / \sum w_i$, $0 \leqslant w_i \leqslant 1$; the weights are determined in terms a parameter descriptor $u_i = (x_i - \Theta)/\delta$, as follows: δ = MAD (median absolute deviation) and

$$w_i = \begin{cases} [1 - u^2/4.685]^2, & \text{if } |u| \leqslant 4.685; \\ 0, & \text{otherwise.} \end{cases}$$

In the context of our system, the $x_i, i = 1, ..., m$ are the feature values of the m calibrating lesions in the training set. Each lesion is assigned a weight. If its weight is zero, then the corresponding lesion is discarded from the training set. Of the m training molecules, m' remain. This gives a weighted-trained similarity estimator, $S_W = a_1 f_1 + ... + a_n f_n$.

In our modeling of estimation error above, one or more features in training may introduce too much variance (systematic error) or dependence (model error). We would like our model to have an extensible and adaptive structure, where any number of features may be used, and proceed with confidence, knowing that noisy or dependent features will have a contribution to the estimate that shrinks to zero. We now apply one of the following patterns of shrinkage to the feature coefficients, $\vec{a}$.

3.2.3.4 *Shrinking feature coefficients to reduce feature space dimensionality*

In 1961, James and Stein published their seminal paper [74] describing a method to improve estimating a multivariate normal mean, $\vec{\mu} = [\mu_1, ..., \mu_k]$, under expected sum of squares error loss, provided the degree of freedom $k \geqslant 3$, and the μ_i are close to the point to which the improved estimator shrinks.

Let $\vec{a} = [a_1, ..., a_k]$ have a k-variate normal distribution with mean vector $\vec{\mu}$ and covariance matrix $\sigma^2 I$, which we measure empirically in train mode. We would like to estimate $\vec{\mu}$ using an estimator

$$\delta(\vec{a}) = [\delta_1(\vec{a}), ..., \delta_k(\vec{a})] \tag{11}$$

under the sum of squares error loss

$$L(\vec{\mu}, \delta) = \sum_{i=1}^{k} (\mu_i - \delta_i)^2 \tag{12}$$

In terms of expected loss,

$$R(\vec{\mu}, \delta) = E_{\mu}[L(\vec{\mu}, \delta(\vec{a}))], \tag{13}$$

James and Stein show that when $k \geqslant 3$, an improved estimator is obtained by a symmetric (or spherical) shrinkage in $\vec{a}$ given by

$$\delta(\vec{a}) = \left[1 - \frac{\kappa(m-k)s^2}{\sum_{i=1}^{m} (N\vec{a})_i^2}\right]^{+} \vec{a}, \tag{14}$$

where

$$\kappa = \frac{(k-2)}{(m-k+2)}, \tag{15}$$

and s^2 is the empirical estimate of variance, σ^2, given by

$$s^2 = \frac{1}{(m-k)} \sum_{i=1}^{m} (1 - (F\vec{a})_i)^2. \tag{16}$$

and where $[x]^+ \equiv max\{0, x\}$.

When extreme μ_i are likely, then spherical shrinkage may give little improvement. This may occur, for instance, when the μ_i arise from a prior distribution with a long tail. A property of spherical shrinkage is that its performance is guaranteed only in a small subspace of parameter space, requiring that one select an estimator designed with some notion of where $\vec{\mu}$ is likely to be, such that the estimator shrinks toward it. An extreme μ_i will likely be outside of any small selected subspace, implying a large denominator and so little, if any, shrinkage in $\vec{a}$, thereby giving no improvement. To address this problem, Stein proposed a coordinate-based (or truncated) shrinkage method, given by

$$\delta_i^{(f)}(\vec{a}) = \left[1 - \frac{(f-2)s^2 min\{1, \frac{z_{(f)}}{|a_i|}\}}{\sum_{j=1}^{m} (F\vec{q})_j^2}\right]^+ a_i, \tag{17}$$

where f is a "large fraction" of k, $z_i = |a_i|, i = 1, ..., k$, $z_{(1)} < z_{(2)} < ... < z_{(f)} < ... < z_{(k)}$ forms a strictly increasing ordering on $z_1, ..., z_k$, s^2 is the empirical estimate of variance, σ^2, given by

$$s^2 = \frac{1}{(q-k)} \sum_{i=1}^{q} (1 - (F\vec{a})_i)^2, \tag{18}$$

and $\vec{q}_i = min\{a_i, z_{(f)}\}, i = 1, ..., k$. Stein shows this estimator is minimax if $f \geqslant 3$. Observe that the denominator is small even when $(k - f)$ of the μ_i are extreme.

3.2.3.5 *Applying the metric*

Once the weighted-trained model feature coefficients, a_i, have undergone shrinkage, a'_i, we have our final hypoxia similarity estimator, $\mathcal{S}_W = a'_1 f_1 + ... + a'_n f_n$ that can measure out-of-sample lesions for their similarity to hypoxic lesions. $\mathcal{S}_W$ gives a [0,1] numerical score instead of a {0,1} outcome. A simulator that implements this scoring function can then feed a branch-and-bound (optimization) process that can explore the simulator's configuration parameter space.

3.3 RESULTS & DISCUSSION

3.3.1 *Image analysis experiment 1: segmenting by histogram multi-thresholding*

3.3.1.1 *Setup*

Encouraged by our results for manually segmenting a few high-concentration anti-pimonidazole images based on simple histogram thresholding—for determining crude pixel area measures of segments, but not for producing segments that are suitable for downstream processing—we decided to apply Otsu's method [117] for automatic multiple thresholding, implemented in the Matlab Image Processing Toolkit as *multithresh*. The pixel areas of viable and hypoxic cells (and their ratio) might possibly serve as image/simulator features.

We compute this for a set of $n_T = 66$ images across stratification criteria, magnification, and high and low concentrations of anti-pimonidazole. Anticipating a likely distinction between results for the high- and low-concentration images, we place, alongside the results for the total set of images, those results for $n_H = 36$ high-

concentration images and $n_L = 30$ low-concentration images, computed separately. See Table 1. The table organization also reflects another distinction we may care to consider, namely between unsmoothed and smoothed gray images. We illustrate this distinction in Figure 24.

For our implementation, see the code listing in Section C.5.

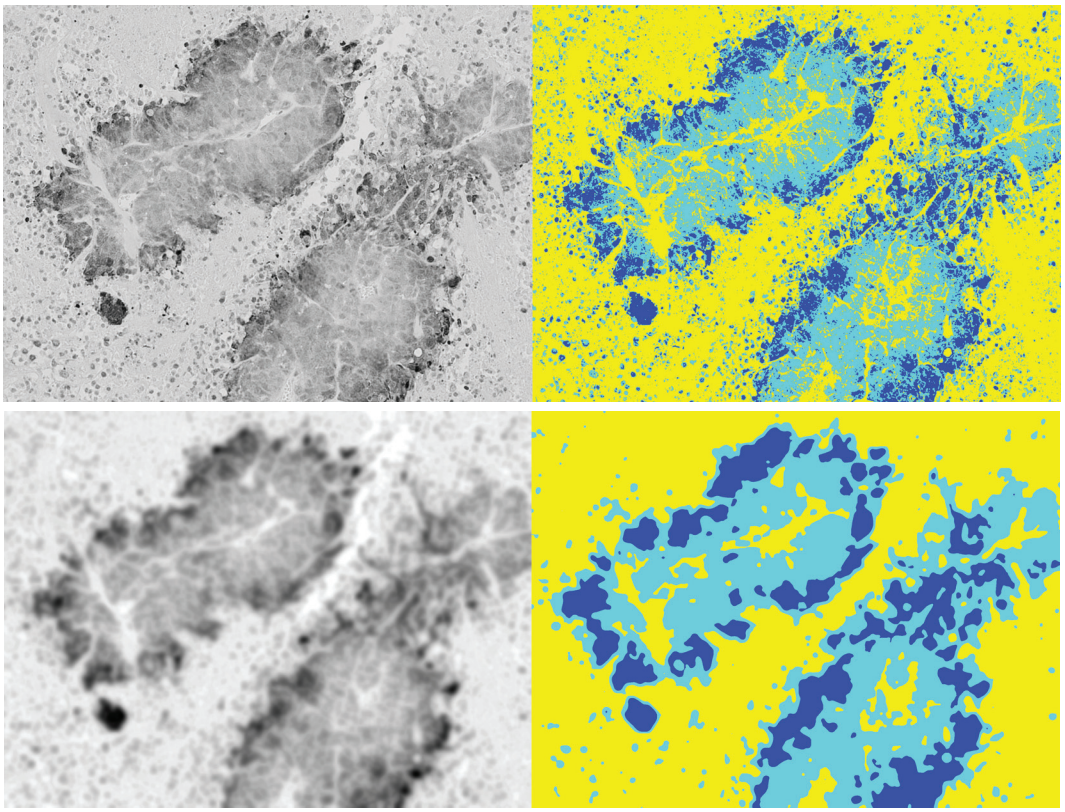

Figure 24: How Otsu's multithreshold segmentation differs between unsmoothed gray and smoothed gray images. Dark blue regions denote hypoxic cells, light blue regions denote viable cells, and yellow regions denote necrotic cells.

3.3.1.2 *Results*

See Table 1.

$n_T = 66, n_H = 36, n_L = 30$	μ_T	σ_T	CV_T	μ_H	σ_H	CV_H	μ_L	σ_L	CV_L
unsmoothed partition 1	136.23	12.25	0.09	143.47	10.74	0.07	127.53	7.31	0.06
unsmoothed partition 2	158.06	13.58	0.09	166.89	11.44	0.07	147.47	6.56	0.04
unsmoothed H:I pixels	0.14	0.04	0.27	0.14	0.03	0.25	0.15	0.04	0.29
unsmoothed V:I pixels	0.40	0.07	0.17	0.42	0.05	0.12	0.38	0.08	0.20
unsmoothed N:I pixels	0.46	0.09	0.20	0.45	0.07	0.16	0.47	0.11	0.23
unsmoothed H:V pixels	0.36	0.08	0.24	0.32	0.07	0.20	0.39	0.09	0.22
smoothed partition 1	144.76	13.13	0.09	150.39	14.11	0.09	138.00	7.55	0.05
smoothed partition 2	158.03	13.91	0.09	166.19	13.03	0.08	148.23	6.80	0.05
smoothed H:I pixels	0.21	0.09	0.41	0.16	0.06	0.38	0.27	0.07	0.27
smoothed V:I pixels	0.41	0.07	0.16	0.41	0.08	0.19	0.41	0.05	0.13
smoothed N:I pixels	0.38	0.11	0.29	0.43	0.11	0.26	0.33	0.08	0.25
smoothed H:V pixels	0.52	0.23	0.45	0.39	0.14	0.37	0.67	0.22	0.33

Table 1: Otsu's multithreshold segmentation of unsmoothed versus smoothed images over the *total* set of images ($n_T = 66$), *high* anti-pimonidazole images ($n_H = 36$), and *low* anti-pimonidazole images ($n_L = 30$). We report pixel areas as proportions of the entire set of pixels in the image (I); hence H:I, V:I, and N:I. We also report another proportion of interest, namely that of hypxic to viable cells in the image, H:V.

3.3.1.3 *Discussion*

We observe the following for unsmoothed and smoothed images. Otsu's method finds intensity level partitions whose means are remarkably stable (CV={0.09, 0.09}, {0.09, 0.09}) across such a variable total set of images. As expected, the stability of these partitions increases as we stratify the images into high-concentration (CV={0.07, 0.07},{0.09, 0.08}) and low-concentration (CV={0.06, 0.04}, {0.05, 0.05}) subsets. Of the pixel proportions, the most stable mean value is always V:I, for the total set and both strata. The mean H:V ratio is also stable across strata (CV={0.24, 0.20, 0.22}) for unsmoothed images, but not as much (CV={0.45, 0.37, 0.33}) for smoothed images. In unsmoothed images, across strata the H:V ratio has a similar mean value (σ={0.36, 0.32, 0.39}); in unsmoothed images, across strata, the mean values vary significantly (σ={0.52, 0.39, 0.67}); between unsmoothed and smoothed images the corresponding mean H:V ratio values seem to have no relationship ({0.36, 0.52}, {0.32, 0.39}, {0.39, 0.67}), though the smoothed, high-concentration mean value (0.39) does seem to fit with the cross-strata values in the unsmoothed images.

We are encouraged by the high stability of the mean partition values, and the reasonably high stability of the mean H:V ratio values for unsmoothed images. Perhaps the mean partition values could serve as image/simulator features that the simulator could measure in its 3D particle concentration lattice, and the mean H:V ratio could serve as an image/simulator feature that the simulator could measure in its 3D cell type lattice.

3.3.2 *Image analysis experiment 2: measuring gradients*

3.3.2.1 *Setup*

We use our Intensity-Sample-Ray-Bundles algorithm to measure gradient properties on high- and low-concentration anti-pimonidazole images that adhere to the stratification criterion that they contain at least one complete lesion at $10\times$ magnification, and the lesions contain at least one blood vessel. For each image, we specify one or more landmarks, (x, y) coordinates, that coincide with vessel locations on the corresponding H&E tumor sections (separated orthotopically by 10 μm). These landmarks are passed to the algorithm to be used as centers from which to extend intensity sample rays. We measure all gradients using 80 intensity sample rays per circle, centered at each landmark. We selected 9 high-concentration anti-pimonidazole images (containing 25 landmarks) and 8 low-concentration anti-pimonidazole images (containing 29 landmarks).

For each landmark the algorithm explores, it outputs the mean and median intensity levels as a function of the radial distance away from the landmark. Both curves are optimally fit using segmented least-squares with a cost parameter $C = 200$. These curves are each usually fit by one, two, or three segments, of different lengths and slopes. These are superimposed on their respective mean and median curves as part of the output. (See Figure 19, for example.) In each case, we examined the output and selected either the mean or median curve fit, depending on which fit gave fewer segments; if they gave the same number of segments, then we selected the mean curve fit. Since the length and slope of these fits characterizes the measured gradient, we would like to use these—actually the average of these, over as wide a sample as possible—as image/simulator features. However, we cannot compare, say, a one-

segment fitted curve to a three-segment fitted one, since these give distinct characterizations of the gradient and we ought to respect that observed distinction. Because of this, we report our results in six tables. The one-, two-, and three-segment fits for the high-concentration anti-pimonidazole images, and the one-, two-, and three-segment fits for the low-concentration anti-pimonidazole images. See Table 2, Table 3, Table 4, Table 5, Table 6, Table 7.

We should note two considerations we made for selecting results to show here. First, we sometimes omitted spurious short or positive-slope segments that appeared first in the sequence of segments (i.e., closest to the center of the circle), since these constitute noisy measurements, usually due to the landmark residing in the center of a high-intensity lumen or some low-intensity blob of pixels; consequently, in some of the tables, the mean segment lengths do not sum to the mean radius length, owing to the mean length of the omitted segments. Second, we selected only gradients that correspond to radii discovered by our algorithm whose length scales match those of the lesions in which they reside, i.e., the contour of the circle defined by the radius coincides with the outermost contour of the hypoxic region in the lesion. (See Figure 18, for example.)

For our implementation, see the code listing in Section C.1.

3.3.2.2 *Results*

See Table 2, Table 3, Table 4, Table 5, Table 6, and Table 7.

$n_i = 1, n_g = 2$	μ	σ	CV
radius	618.50	70.00	0.11
segment 1 length	459.00	11.31	0.02
segment 1 slope	-0.02	0.00	0.00
segment 1 error	348.38	70.70	0.20

Table 2: 1-segment radii in *high* anti-pimonidazole images. The values of n_g and n_i report that the statistics are from a sample of 2 gradients found in 1 image.

$n_i = 4, n_g = 7$	μ	σ	CV
radius	457.86	113.23	0.25
segment 1 length	84.86	35.41	0.42
segment 1 slope	-0.13	0.08	0.63
segment 1 error	106.57	126.50	1.19
segment 2 length	351.71	72.17	0.21
segment 2 slope	-0.03	0.02	0.50
segment 2 error	322.89	169.55	0.53

Table 3: 2-segment radii in *high* anti-pimonidazole images. The values of n_g and n_i report that the statistics are from a sample of 7 gradients found in 4 images.

$n_i = 8, n_g = 16$	μ	σ	CV
radius	477.25	95.07	0.20
segment 1 length	80.75	38.44	0.48
segment 1 slope	-0.22	0.12	0.56
segment 1 error	68.39	57.47	0.84
segment 2 length	219.00	85.48	0.39
segment 2 slope	-0.04	0.08	1.85
segment 2 error	158.19	123.14	0.78
segment 3 length	164.38	73.49	0.45
segment 3 slope	-0.09	0.06	0.63
segment 3 error	115.89	122.65	1.06

Table 4: 3-segment radii in *high* anti-pimonidazole images. The values of n_g and n_i report that the statistics are from a sample of 16 gradients found in 8 images.

$n_i = 2, n_g = 4$	μ	σ	CV
radius	348.00	49.29	0.14
segment 1 length	348.00	49.29	0.14
segment 1 slope	-0.09	0.03	0.30
segment 1 error	288.37	199.55	0.69

Table 5: 1-segment radii in *low* anti-pimonidazole images. The values of n_g and n_i report that the statistics are from a sample of 4 gradients found in 2 images.

$n_i = 8, n_g = 20$	μ	σ	CV
radius	454.55	138.59	0.30
segment 1 length	172.25	82.96	0.48
segment 1 slope	-0.21	0.19	0.91
segment 1 error	94.34	82.17	0.87
segment 2 length	267.40	125.64	0.47
segment 2 slope	-0.06	0.03	0.55
segment 2 error	112.97	111.27	0.98

Table 6: 2-segment radii in *low* anti-pimonidazole images. The values of n_g and n_i report that the statistics are from a sample of 20 gradients found in 8 images.

$n_i = 4, n_g = 5$	μ	σ	CV
radius	677.80	390.88	0.58
segment 1 length	153.60	74.42	0.48
segment 1 slope	-0.17	0.21	1.23
segment 1 error	61.13	37.22	0.61
segment 2 length	187.40	64.40	0.34
segment 2 slope	-0.08	0.07	0.94
segment 2 error	64.03	63.07	0.98
segment 3 length	318.20	417.14	1.31
segment 3 slope	-0.04	0.01	0.35
segment 3 error	141.12	276.56	1.96

Table 7: 3-segment radii in *low* anti-pimonidazole images. The values of n_g and n_i report that the statistics are from a sample of 5 gradients found in 4 images.

3.3.2.3 *Discussion*

For ease of discussion, let H denote the set of high-concentration anti-pimonidazole images or the segmented gradient curves that derive from them, and L denote the

set of low-concentration anti-pimonidazole images or the segmented gradient curves that derive from them.

We immediately observe that the majority of gradients in H images have 3-segment fits, whereas the majority of gradients in L images have 3-segment fits. This less complicated structure of L gradients agrees with our intuition that they are better for characterizing continuous gradients that are not punctuated by the relatively flat middle segments we see with the H gradients. This is further bolstered by a closer examination of the structure of the segmented curves. In comparing the H vs L segmented gradient curves: H 1-segment curves are flatter than those of L; H 2-segment curves are flatter in both segments than those of L; and H 3-segment curves are defined by a concave-then-convex shape, whereas those of L are decidedly concave, i.e., they tend to have monotonically decreasing slopes as a function of radial distance away from the vessel.

Suppose we focus only on L gradient curves, believing they more closely reflect real underlying hypoxia gradients. We observe that as radial distance grows, the gradient becomes nonlinear, following from its concavity. We have not performed nonlinear fits to the gradient curves but suspect a quadratic (and certainly an exponential) curve would easily fit with low error.

As a proportion of their sum, the first and second L segments tend to be 0.39 and 0.61 of their total 2-segment length, respectively; and the first, second, and third L segments tend to be 0.23, 0.28, and 0.48 of their total 3-segment length, respectively.

These segment proportions, their slopes, and the parameterized nonlinear fit to the gradient curve could serve as image features, and could easily be measured in our simulation by inspecting the 3D lattice representing particle type concentrations.

That said, we should note the statistical significance, and the degrees of error, we see in the L gradient measurements. First, with 1-, 2-, and 3-segment sample sizes of 4, 20, and 5, respectively, we acknowledge that at least the 1- and 3-segment data are less than statistically significant, and even the extreme variance—particularly with the slopes of the first and second segments ($CV = 1.23$ and $CV = 0.94$, respectively), and the length of the third segment ($CV = 1.31$) in the 3-segment fit—these might diminish with a larger sample. Looking at the more significant sample of 2-segment gradient measurements, we also see high variance in the slopes, with $CV = 0.91$ and $CV = 0.98$ for segments 1 and 2, respectively. With this in mind, we are unclear how ultimately useful these measure will be as generalized, canonical features to which we should seek comparison in our simulation. Perhaps the parameterized nonlinear fit will be more stable and therefore more suitable feature.

3.3.3 *Image analysis experiment 3: measuring quad-tree statistics*

3.3.3.1 *Setup*

Our Ply-Stats-Quad-Tree algorithm reports statistics related to the search tree for the image that it processes. These include the count, sum, mean, median, standard deviation, and CV for the number of leaves at each ply, and a histogram of the counts of leaves at each ply. We use CV in intensity value of the current frame's pixels as our splitting property, where CV exceeding a given threshold, τ, generates a split. The algorithm reports search tree statistics for the quad-tree dissection at a given value of τ. It computes these for $\tau \in [0.01, 0.10]$ by increments of 0.01, therefore generating 10 sets of statistics for a given image. We selected $\tau \in \{0.1, 0.5, 0.9\}$ to report here.

The mean histogram of ply counts (image frame sizes) might possibly serve as an image/simulator feature.

We compute this for a set of $n_T = 66$ images across stratification criteria, magnification, and high and low concentrations of anti-pimonidazole. Anticipating a likely distinction between results for the high- and low-concentration images, we produce, along with the figure reporting the results for the total set of images (Figure 25), those results for $n_H = 36$ high-concentration images (Figure 26) and $n_L = 30$ low-concentration images (Figure 27), computed separately.

For our implementation, see the code listing in Section C.3.

3.3.3.2 *Results*

See Figure 25, Figure 26, and Figure 27.

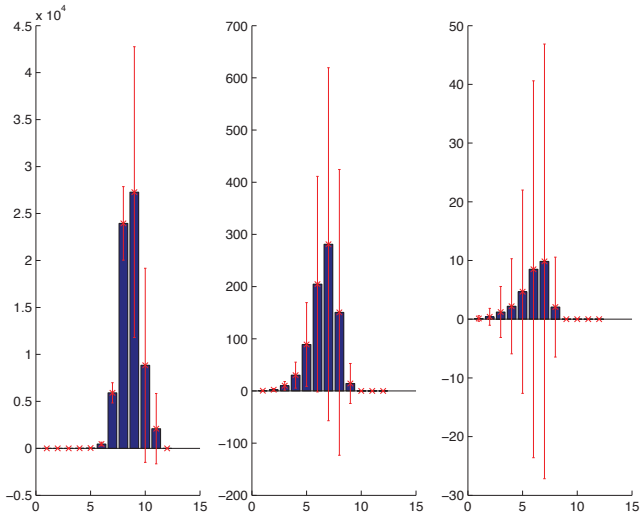

Figure 25: How Quad-Tree dissects images according to the prpoerty of CV in intensity level of a given frame's pixels: the mean window size profile across the *total* set of images ($n = 66$). The left, middle, and right mean histograms correspond to $\tau \in \{0.1, 0.5, 0.9\}$, respectively. The horizontal axis indicates ply depth, or frame size as computed by $\frac{x\_dim}{2^i} \times \frac{y\_dim}{2^i}$, where $i \in [0, 12]$ is the ply depth, and x_dim and y_dim are the x and y dimensions of the whole image, respectively. The vertical axis indicates the mean count of search tree leaves at ply depth i. Error bars show standard deviation.

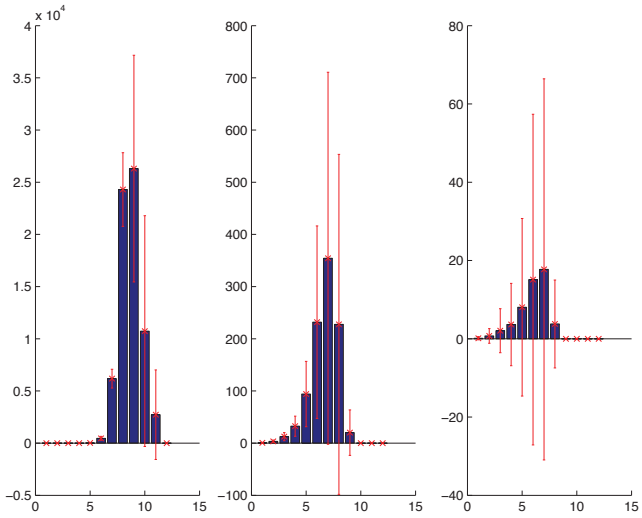

Figure 26: How Quad-Tree dissects images according to the prpoerty of CV in intensity level of a given frame's pixels: the mean window size profile across the *high* anti-pimonidazole images ($n = 36$). The left, middle, and right mean histograms correspond to $\tau \in \{0.1, 0.5, 0.9\}$, respectively. The horizontal axis indicates ply depth, or frame size as computed by $\frac{x\_dim}{2^i} \times \frac{y\_dim}{2^i}$, where $i \in [0, 12]$ is the ply depth, and x_dim and y_dim are the x and y dimensions of the whole image, respectively. The vertical axis indicates the mean count of search tree leaves at ply depth i. Error bars show standard deviation.

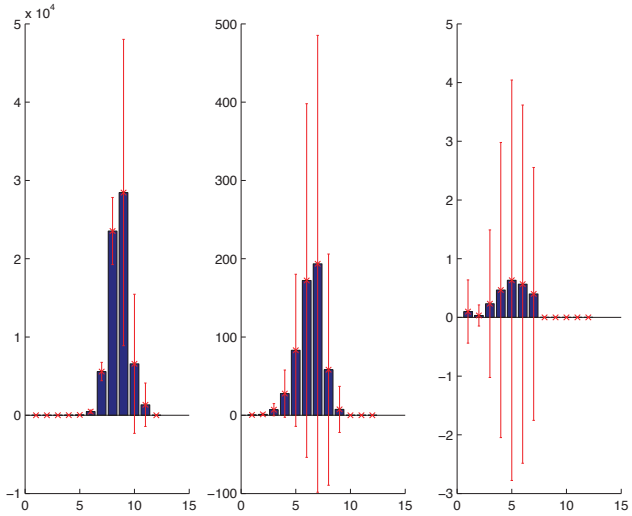

Figure 27: How Quad-Tree dissects images according to the prpoerty of CV in intensity level of a given frame's pixels: the mean window size profile across the *low* anti-pimonidazole images ($n = 30$). The left, middle, and right mean histograms correspond to $\tau \in \{0.1, 0.5, 0.9\}$, respectively. The horizontal axis indicates ply depth, or frame size as computed by $\frac{x\_dim}{2^i} \times \frac{y\_dim}{2^i}$, where $i \in [0, 12]$ is the ply depth, and x_dim and y_dim are the x and y dimensions of the whole image, respectively. The vertical axis indicates the mean count of search tree leaves at ply depth i. Error bars show standard deviation.

3.3.3.3 *Discussion*

At a glance, it is easy to observe the overwhelming degree of error in these measures, regardless of stratification. While decreasing τ consistently produces a more stable mean profile, even the minimum value of $\tau = 0.01$ we tested is too disperse. We had hoped that the frame size profiles would have converged to some stable mean "canonical" profile, but this is not the case. As such, this is unusable as an image/simulator feature.

3.3.4 *Image analysis experiment 4: deriving canonical EPC signatures*

3.3.4.1 *Setup*

We implement an algorithm that follows directly from the approach taken by Hutterer, *et al.* [71] to construct an Euler-Poincaré signature curve for an image. First, convert the RGB image $\mathcal{I}$ to an 8-bit gray level image $\mathcal{I}_g$, but do not smooth. Then for each gray level $i = 1, ..., 255$, produce a binary intensity-thresholded image $\mathcal{I}_i$ and record $\mathrm{EPC}(\mathcal{I}_i)$ for each i. This method gives a signature EPC curve for each $\mathcal{I}$.

We perform this analysis for each image in a set of $n_T = 66$ images across stratification criteria, magnification, and high and low concentrations of anti-pimonidazole. Anticipating a likely distinction between results for the high-concentration ($n_H = 36$) and low-concentration ($n_L = 30$) images, we stratify the resulting curves, and for each stratum, we compute a mean EPC curve separately. We plot these canonical stratum curves with their respective standard deviations in Figure 28.

Then we approximate each canonical stratum curve with an optimized segmented least-squares fit, and thereby compress the data into a smaller number of real-valued coefficients—slope and y-intercept for each segment—that might possibly serve as image/simulator features. The segmented least square fits are given by a dynamic programming algorithm [86], using a cost parameter $C = 50000$. We also report the sum of least-squares errors over all of the segments in the fit. (See Figure 29.) The segment fit coefficients and the error might possibly serve as image/simulator features.

For our implementation, see the code listing in Section C.4.

3.3.4.2 *Results*

See Figure 28 and Figure 29.

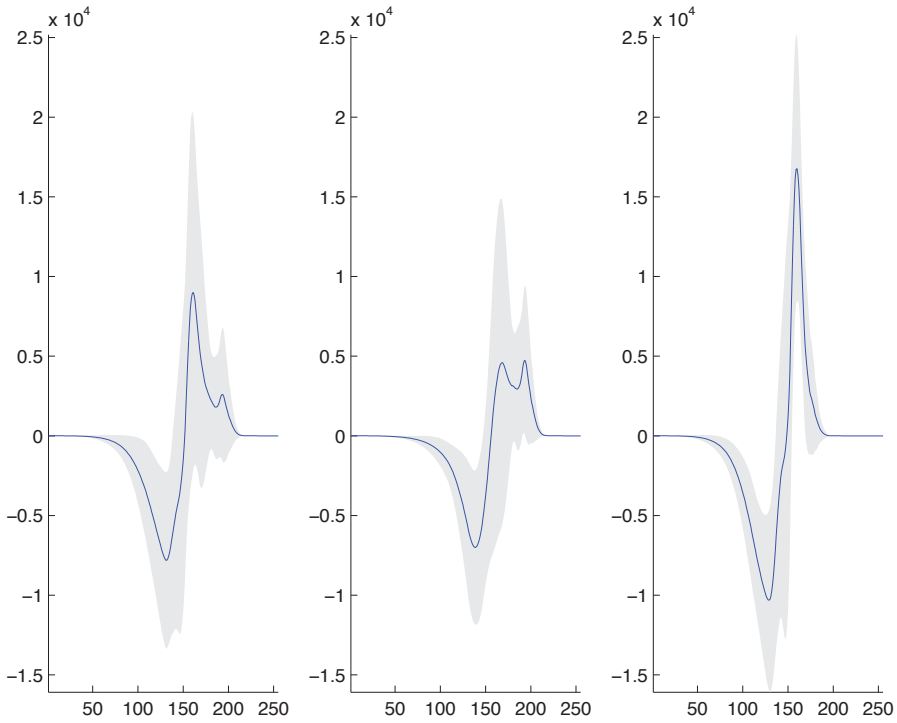

Figure 28: The mean EPC curve over the *total* set of images (left, $n_T = 66$), the *high* concentration anti-pimonidazole images (middle, $n_H = 36$), and the *low* concentration anti-pimonidazole images (right, $n_L = 30$). Segmented least-squares fits to these curves are given in Figure 29. The horizontal axis indicates the intensity level threshold $\tau \in [1, 255]$ applied to the image prior to computing χ. The vertical axis indicates the value of χ computed for each τ.

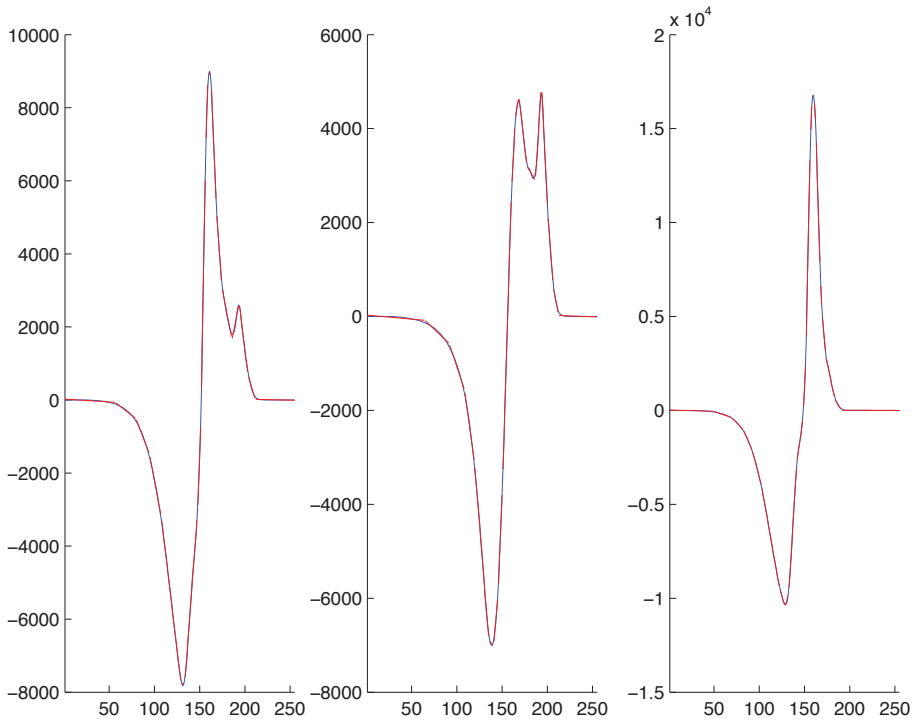

Figure 29: The segmented least-squares fits to the mean EPC curves given in Figure 28. The fit on the left is composed of 18 segments (specified by 36 coefficients), giving a compression factor of 7.08 and a normalized least-squares error of 1082.84. The fit in the middle is composed of 18 segments (specified by 36 coefficients), giving a compression factor of 7.08 and a normalized least-squares error of 933.19. The fit on the right is composed of 21 segments (specified by 42 coefficients), giving a compression factor of 6.07 and a normalized least-squares error of 1063.19. The horizontal axis indicates the intensity level threshold $\tau \in [1, 255]$ applied to the image prior to computing χ. The vertical axis indicates the value of χ computed for each τ.

3.3.4.3 *Discussion*

At a glance, it is easy to observe the overwhelming degree of error in these measures, regardless of stratification. While one can roughly discern a characteristic shape similarity in the curves, this is not rigorously established feature. We had hoped that the

EPC curves would have converged to some stable mean "canonical" signature, but this is not the case. As such, this is unusable as an image/simulator feature.

3.4 CONCLUSIONS & FUTURE WORK

3.4.1 *Conclusions*

3.4.1.1 *Our contributions*

We contribute two algorithms for this study: Intensity-Sample-Ray-Bundles (a novel algorithm) for gradient analysis with respect to a blood-vessel-centric view of hypoxia; and Ply-Stats-Quad-Tree (an extension of a common algorithm) for gathering multiscale, hierarchical statistics on images relative to a splitting criterion. In addition, we implement the EPC curve algorithm by Hutterer, *et al.* [71], extending it to approximate EPC curves with the segmented least-squares dynamic programming algorithm given in [86].

We develop a simple spatiotemporal grammar of nested structures which we use as an organizing principle for building qualitative hypoxia image/simulator features.

We extend the Probabilistic Bounded Linear Temporal Logic (PBLTL) given in [77] to accommodate our needs for spatial and temporal specification in our problem domain.

We derive a novel spatiotemporal logical proposition, ϕ, that characterizes our hypoxia images as best understood by our human domain expertise and human ability to generalize in a qualified way, with respect to the image features we have examined.

We propose a method to derive a linear regression functional characterization, f, that characterizes our hypoxia images through a simple process of machine learning, and we employ methods to avoid Type I and Type II errors, and reduce feature dimensionality where possible. This method has been rigorously tested in another image processing domain [139]. Testing it on the domain of hypoxia histology images we leave for future work.

We have not implemented ϕ or f as feature integrators embedded in the simulator. Before doing so, we must first establish a stable set of image/simulator features (the aim of the image analysis), then implement their measurement as feature measurers embedded in the simulation. So testing ϕ and f we leave for future work.

Toward establishing a set of candidate image/simulator features, we contributed an analysis of: segmentation by Otsu's multithreshold method, gradients by the Intensity-Sample-Ray-Bundles algorithm, EPC-curves by the Hutterer, *et al.* algorithm, and quad-tree ply statistics by the Ply-Stats-Quad-Tree algorithm. All of these were based on the hypoxia histology images for our problem domain. Please see our findings below.

3.4.1.2 *Our findings*

By way of stratifying our data, we affirmed our intuition that high-concentration anti-pimonidazole images are better suited for intensity histogram-based image segmentation and not well suited for gradient analysis; and the opposite is true for low-concentration anti-pimonidazole images.

Due to their implicit variance, whole-image features, such as EPC-curves and quad-tree ply statistics seem unreliable as image/simulator features. Perhaps quad-tree ply statistics would be more stable and useful if a different splitting property were used

than CV, like Moran's *I* statistic or the EPC. The best of the whole-image features we investigated emerge from Otsu's multi-threshold segmentation, namely its partition values, and the H:V ratio for unsmoothed images. Gradient measures, which focus on lesion-scale, not image-scale, properties, seem more stable and useful for image/simulator features, especially the implied parameterized nonlinear fit to the gradient. But even these measures contain significant error. We should consider these preliminary results based on preliminary image data.

3.4.2 *Future work*

3.4.2.1 *To interface*

IMPLEMENT FEATURE MEASURERS In this chapter, we have defined several image features that have natural analogues in the domain of simulator features. Chemical concentration 2D and 3D matrices lend themselves to thresholding and gradient analysis. Cell population 2D and 3D matrices give us segmentation immediately, and since simulated tissue type boundaries are trivial to detect, then so computation of qualitative nested structure invariants. Further, they lend themselves to computing cell type ratios in any size area or volume, spatial statistics (like Moran's I), structural characteristics (like the Euler-Poincaré, and other Minkowski functionals—already implemented in our simulator), and they easily decompose into sub-areas and sub-volumes for multiscale, hierarchical analysis. The more difficult part to implement is the feature integrators.

Specify the logical feature integrator Once ϕ is properly codified, it can specify the logical feature integrator. We plan to construct a spatiotemporal logical proposition that will combine those features best suited for this type of expression.

Specify the functional feature integrator Once f is properly codified, it can specify the functional feature integrator. We plan to train a similarity score function and in so doing, will select the features that give the most promising results.

3.4.2.2 *To extend and enhance*

BROADEN THE BIOLOGICAL TUMOR HYPOXIA STUDY Our study is preliminary. It is based on only two tumors xeno-graphed into a single organism. For each of these tumors, only a handful of histology slides were produced, from which we took our few H&E and anti-pimonidazole stain images. All of this points to the statistical insignificance of our sample, and therefore the potential lack of robustness in our results. Our first order of business is to conduct a broader study of hypoxia. This study would include a large enough sample (e.g., more than 5 tumors) to capture some of the natural variance we expect in tumors of different ages, proliferation rates, angiogenicity, and convergent volumes. For histology, we would use a richer grade of dilutions of anti-pimonidazole than just "high" and "low". Younger tumors will tend to have smaller necrotic cores and less vasculature. Tumor size positively correlates with blood supply (number of vessels). A broader study such as this would give us the opportunity to test many of our modeling assumptions, and the wider sample would enable us to create a more robust characterization.

Use alternative phenotype(s) The DLD-1 K-Ras cancer phenotype we used for this study is known to be particularly rich in collagen deposition. Given that collagen has

been so problematic in our efforts to segment the images, and given that we aim to broaden our study to capture the natural complexity of these tumors, we would like to expand the number of cancer phenotypes to at least three.

Use alternative stains for hypoxia We could try using pimonidazole without the counterstain, and determine if that would attenuate the structural noise in the images that comes from the counterstain's enhanced contrast.

Stain for neovasculature Newly grown blood vessels that arrive late on the scene make a contribution to tissue oxygenation that is not well understood; perhaps it is negligible. Neovasculature is difficult to distinguish from mature blood vessels. We could use anti-CD-31 (an endothelial marker) and smooth muscle actin (SMA) that labels blood vessels. We could then perform a gradient analysis between stratified data sets to determine if we see any empirical differences, and if such differences do exist, thus constituting a quantifiable principle for hypoxia.

Align with exprimental data from literature None of our findings have yet been matched or calibrated with respect to quantities in the literature such as oxygen diffusion rates in different tumor tissues and in collagen; *in vivo* consumption and release rates of various tumor cell types; and proliferation rates of said types.

IMAGE PROCESSING

Measuring gradients We would like to conduct a study on our low-concentration anti-pimonidazole images to fully examine the family of gradient slopes we extract, and their variation. We would like to relate the slope of the hypoxia gradient to the

size of the lesion (normalizing for radius length). Such a finding could shed light on another quantifiable principle of hypoxia.

Toward a hierarchical or multiscale structural analysis Considering the quad-tree recursive decomposition, we would like to try different splitting criteria based on different image properties, including: cell surface density, cell volume density, tissue connectedness (using the Euler-Poincare characteristic[107, 93, 59, 120, 125, 71]), tissue heterogeneity (using Moran's I statistic for spatial autocorrelation [20]), and ratios of cell types.

3.4.2.3 *To explore*

IMAGE PROCESSING

Segmentation We would like to explore a number of avenues toward image segmentation. First, perhaps raising contrast on original, not gray-blurred, image might improve prospects. Though it would introduce more structural information (noise) into an intensity-level-based analysis, it might steer us toward bounding regions based on identifiable markers on the cells, thereby affording a more accurate estimation of cell type area and relative orientation. Second, we could use the Beaton-Tukey biweighting method [14] to compute each pixel's contributions to the different modes. Third, Gonzalez & Woods [56] propose a way to use quad-tree processing to perform image segmentation, by merging the areas constituting the leaves of the search tree (or rather, the planes in the lowest ply). Lastly, as discussed in the introduction, we would like to explore using an MRF, texture-learning-based approach like that taken by Wang, *et al.* [149].

Characterizing segments Once we have found a suitable method for segmenting lesions into their three constituent tissue types, then we can explore properly characterizing each segment. These include: cell region thickness (minor, orthogonal axis), cell region skeleton (using erosion morphological operators), cell surface density, cell volume density, tissue connectedness (using the Euler-Poincare characteristic[107, 93, 59, 120, 125, 71]), tissue heterogeneity (using Moran's I statistic for spatial autocorrelation [20]), and ratios of cell types. The 8-connected components of each segment define distinct sets (pixel blobs). These can be filtered for size, eliminating small noisy components; those that remain can participate in a census and thus give size/area statistics. These sets can undergo morphological transformations (like dilation and erosion) that will transform them into distinct, minimal geometric objects whose properties (like shape, thickness, branching factor and angle, curvature, and relative orientation) can be computed, statistically analyzed, and mathematically (geometrically) characterized. For example, it is apparent when you look at enough complete lesions with hypoxic outer contours, that these take a similar shape. Such a shape property might be governed by an intrinsic principle, like vessel location and oxygen diffusion dynamics, or an extrinsic principle, like proximity to other lesions. In such a case, a rigorous geometric characterization of, say, the loop of hypoxic cells that bound a viable region, would help us assess the regularity of certain growth patterns, and crucially (for the problem of this dissertation), provide a signature feature that is also easy to compute in a simulation.

Measuring gradients Variance filters are a tool employed by the image processing software Image-J[3]. These highlight edges in the image by replacing each pixel with the neighborhood variance, computed at a specified radius. This might be useful as

3 http://rsbweb.nih.gov/ij/

a preprocessing step in measuring hypoxia gradients in our complex images, as they need not reference a hypothesized center. Along these lines, we would like to explore using wavelet transforms [56] for detecting and measuring image gradients. Though we have seen no studies that demonstrate this for histology images, they seem well suited to the task.

LINEAR REGRESSION FUNCTION LEARNING As an alternative to the Beaton-Tukey biweighting approach, we would like to explore the use of Efron's local empirical *fdr* algorithm [35, 36] in our effort to control Type I and Type II errors in the hypoxia image training data.

4

FUTURE WORK: EXPLORING SIMULATION PARAMETER SPACE AND EVALUATING ITS COORDINATES

4.1 INTRODUCTION

4.1.1 *Problem statement*

Identify the initial conditions and operational parameters of an *in silico* model (simulation) that result in hypoxia, as characterized by the materials & methods in Chapter 3.

4.1.2 *Background & literature review*

To our knowledge, no studies exist that address our computational inverse problem: identify the initial conditions and operational parameters of a simulation framework that drive it into a recognizable state of formally characterized hypoxia.

The closest approach we found is the study by Grosu, *et al.* [58], who use model checking with a temporal logical characterization to tackle the problem of learning and detecting emergent behavior in networks of cardiac myocytes. (See our discussion in Section 1.5.) While this approach demonstrates the validity and effectiveness of using temporal logic and model checking for the problems of specification and detection of an emerging complex biological property, it is not so much concerned with their version of the longer time scale computational inverse problem: what initial

conditions and operational parameters drive simulated cardio myocytes to a likely state of spiral waves followed by fibrillation. And the traditional model checking approach they have taken does not lend itself well to stochastic models—the motivation for developing the statistical model checking approach.

4.1.3 *Our materials & methods*

4.1.3.1 *The nature and extent of the simulation parameter space*

The nature and extent of the simulation parameter space derives from the algorithmic specification given in the materials & methods of Chapter 2. We now enumerate the parameters required to simulate n cell types and m particle types. The 2D or 3D cell lattice may be initialized in any way, but for the purposes of this estimation, let us assume that each cell type (including the vessel and empty types) require specification, and that each cell type's initial locations in the lattice can be determined by a positional function that uses one parameter for each spatial dimension. For a 2D simulation, that implies $2(2+n) = 2n+4$ parameters for specifying initial area positions. For a 3D simulation, that implies $3(2+n) = 3n+6$ parameters for specifying initial volume positions. The constant 2 comes from there being 2 preexisting cell types (vessel and empty) whose parameters must be specified in addition to those of the n cells types. Next we consider the operational parameters. Each particle type must have a diffusion rate, initial concentration, and basal lower-bound and basal upper-bound concentrations. That implies $4m$ parameters. Each cell type must have a consumption rate, release rate, and impact factor for each particle type, implying $3nm$ parameters. Each cell type must be either replaceable or not, and either reproductive or not, implying $2n$ parameters. Lastly we consider the condition

behavior specification. We assume that a modest conditional behavior programming would entail one trigger-action pair for each cell type, where each trigger-set is comprised of two three-argument predicates, and each action set is comprised of two three-argument executions. This implies $n(2 \cdot 3 + 2 \cdot 3) = 12n$ parameters. The final tally is $3nm + 16n + 4m + 4$ parameters for a 2D simulation and $3nm + 17n + 4m + 6$ parameters for a 3D simulation. For even modest n and m, this is a large configuration space, C, to explore. However, certain simulations we are interested in exist in a smaller space than the one just estimated.

For concreteness, let us consider the 2D simulation that we have observed will generate stable local regions of hypoxia. This simulation uses vessel and empty cell types and three additional ones, to represent viable, hypoxic, and necrotic cells ($n = 5$). It simulates only oxygen ($m = 1$). The initial positions of 10 vessels are randomly placed, using one parameter to specify the number of vessels and two spatial parameters to obtain random numbers in the admissible range, and two spatial parameters to give the coordinates of the first proliferating viable cell (5 parameters). There are $4m = 4$ parameters to specify oxygen diffusion rate, initial concentration, and upper- and lower-basal concentrations. There are $3nm = 15$ parameters to specify the cell consumption and release rates of oxygen, and the impact factors of oxygen upon the cell types. There are $2n = 10$ parameters to specify the replaceable and reproductive predicates of each cell type. This simulation uses a simpler conditional logic configuration than assumed above. Here, the viable cell type has one trigger-action pair: if oxygen concentration is less than 0.07 then become hypoxic (5 parameters); and the hypoxic cell type has two trigger-action pairs: (1) if oxygen concentration is less than 0.05 then become necrotic (5 parameters), and (2) if oxygen concentration is

greater than 0.07 then become viable (5 parameters). The tally for this simulation is 49 parameters.

4.1.3.2 *Attacking the curse of dimensionality*

When considering problems in dynamic optimization, Richard Bellman coined the term "curse of dimensionality" to describe the difficulties implicit in large data spaces. In general, when the dimensionality increases, the volume of the space grows so fast that the available data becomes sparse, and this becomes problematic if we hope to use methods that require statistical significance, since the amount of data required to support a result grows exponentially in the dimensionality.

If we were interested in the more difficult problem of estimating or reconstructing the joint probability distributions between parameter random variables (or even just their individual distributions), or if we were faced with the problem of obtaining a sequence of random samples from a probability distribution for which direct sampling is difficult, then we would consider using one of the Markov chain Monte Carlo (MCMC) approaches widely cited in the literature, like Metropolis-Hastings [104] or Gibbs sampling [51].

But for our problem, we will assume that in the absence of simplifying factors or expert knowledge of the biology, each random variable has a uniform, independent probability distribution. Our aim here is modest: sample the large parameter space using a vanilla Monte Carlo algorithm [105] and accumulate families of nearby solutions. We do attempt to make this process more efficient by learning from each sample's truth outcome if it is in {0,1}, or branching-and-bounding sampled subspaces where the sample values are in [0,1]. In this way, we propose two simple "adaptive Monte Carlo" methods, MC-Boost and MC-Walk, that employ boosting of

the independent probability distributions upon successful samples, and constrained random walks around successful samples, respectively. And we propose a simple adaptation to the traditional branch-and-bound algorithm, MC-Branch-and-Bound: instead of systematically exploring a subspace of problems, we employ constrained Monte Carlo sampling of each subspace.

4.1.3.3 *Bayesian statistical model checking*

Legay, *et al.* [92] provide a good overview of statistical model checking. A number of recent computational studies [77, 162, 54, 55] have employed statistical model checking algorithms to verify spatiotemporal logical propositions in biological systems. They use temporal logic to characterize phenomena of interest in: a fibroblast growth factor signaling model, circadian rhythm, yeast heterotrimeric G protein cycle control, and the HMGB1 signaling pathway in cancer. Jha, *et al.* [77] give the Bayesian statistical model checking algorithm we employ here.

The version of our stochastic simulation framework that embeds the logical feature integrator/detector will produce as its output a value in {0,1}, indicating whether or not it detected the emergence of hypoxia during its execution trace, as characterized by the spatiotemporal logical proposition ϕ. But given a fixed parametric configuration, $c \in C$, that Bernoulli outcome of the simulation is itself stochastic. So we need a method for deciding the outcome of a Bernoulli stochastic random variable over a multitude of outcomes. The best way we have found for doing such inference is the Bayesian hypothesis testing algorithm developed for the probabilistic model checking problem [77, 162, 54, 55]. We outline the Bayesian statistical model checking algorithm given in [77] below, in our materials & methods.

4.1.3.4 *Mean-variance thresholding*

The version of our stochastic simulation framework that embeds the functional feature integrator/detector will produce as its output a numerical quantity in [0,1], indicating the most similar outcome to hypoxia that the simulator has detected during its execution trace, as characterized by the similarity function f. But given a fixed parametric configuration, $c \in C$, that numerical outcome of the simulation is itself stochastic. So we need a method for deciding the stable value of a stochastic random variable in [0,1] over a multitude of outcomes. One obvious, simple approach we have adopted is to examine its mean and standard deviation, and apply a threshold to its $CV = \frac{\sigma}{\mu}$.

4.2 MATERIALS & METHODS

4.2.1 *Statistical model checking by Bayesian hypothesis testing*

Jha, *et al.* [77] formulate Bayesian statistical model checking in the following way, cast in terms of our problem.

The probabilistic model checking (PMC) problem is: given a stochastic simulator S, a starting state s_0, a BLTL proposition ϕ, and a probability threshold $\theta \in (0, 1)$, decide algorithmically whether $S, s_0 \models P_{\geqslant\theta}(\phi)$. Let p be the unknown but fixed probability of the model satisfying ϕ. We can now restate the PMC problem as deciding between two hypotheses: $H_0 : p \geqslant \theta$ and $H_1 : p < \theta$. For any trace σ_i of our simulation, S, we can deterministically decide whether σ_1 satisfies ϕ. Therefore, we can define a Bernoulli random variable X_i denoting the outcome of $\sigma_i \models \phi$.

Since each simulation trace is given by an independent execution of the model, we can assume the X_i are i.i.d. Since p is unknown, we assume it is given by a random variable, whose density $g(.)$ is called the *prior* density. The prior is usually based on our previous experiences and beliefs about S. If we are totally ignorant of the likelihood of whether S will satisfy ϕ, then we use a *non-informative* or *objective* prior.

The authors define the Bayes factor $\mathcal{B}$ of sample $d = (x_1, ..., x_n), x_i \in X_i, i = 1, ..., n$ and hypotheses H_0 and H_1 as $\mathcal{B} = \frac{P(d|H_0)}{P(d|H_1)}$. They cite Jeffreys [75], who treats $\mathcal{B}$ as a measure of the relative confidence in H_0 versus H_1, and who believes that Bayes factors in excess of 100 provide decisive evidence in favor of H_0. To test H_0 versus H_1, the authors compute the Bayes factor of the available data, then compare it against a fixed threshold, $T > 1$. They accept H_0 iff $\mathcal{B} > T$. In the dual interpretation, Jeffreys treats $\frac{1}{\mathcal{B}}$ as the measure of evidence in favor of H_1.

They show how to compute the Bayes factor using the conjugate prior to the Bernoulli distribution, namely the Beta distribution, and then give an algorithm that is essentially a sequential version of Jeffreys' test. Initially, two counters are set to 0: n, which denotes the number of traces drawn so far, and s, which denotes the number of traces satisfying ϕ so far. It iteratively draws i.i.d. simulation sample traces $\sigma_1, \sigma_2, ...$, incrementing n each time, and checks whether they satisfy ϕ. (In the case of our proposed system, this check happens in the logical feature integrator/detector that is embedded in the simulator.) The truth or falsity of whether a given simulation trace satisfies ϕ is treated as independent sampling from a Bernoulli distribution X of unknown parameter p, the actual probability of the simulation satisfying ϕ. If $\sigma_i \models \phi$, then s is incremented. At stage k the algorithm has drawn samples $x_1, ..., x_k$ i.i.d. from X. It then computes $\mathcal{B}_k$ in terms of the Beta distribution, using the counter

values of n and s: $\mathcal{B}_k = \frac{1}{F(s+\alpha, n-s+\beta)} - 1$. The algorithm stops iff $(\mathcal{B}_k > T \vee \mathcal{B}_k < \frac{1}{T})$. When this occurs, it will accept H_0 iff $\mathcal{B}_k > T$, and will accept H_1 iff $\mathcal{B}_k < \frac{1}{T}$.

The authors prove two theorems related to the algorithm's performance.

THEOREM 1 (TERMINATION). *The SMC-BHT decision algorithm terminates with probability one, for Beta priors and Bernoulli samples.*

THEOREM 2 (ERROR BOUND). *If the SMC-BHT decision algorithm terminates after observing n sample traces, then an upper bound on the probability of the Type I error is $\sum_{x=0}^{n} I_{\{\mathcal{B}(n,x)<\frac{1}{T}\}} \binom{n}{x} t_{max}^x (1-t_{max})^{n-x}$, where t_{max} is the value of t that maximizes $t^i(1-t)^{n-i}$ defined on $[\theta, 1]$, T is the Bayes Factor threshold used in the algorithm, and I is the indicator function.*

We interpret our simulator trace as a Bernoulli trial for a given model configuration, $c \in C$, implementing a fixed spatiotemporal logical definition, ϕ, of hypoxia. The SMC-BHT decision algorithm described above is thus a partitioning function for our simulator configuration space, giving hypoxia-generating versus non-hypoxia-generating subsets of simulator parameters.

In our implementation of this algorithm (Section D.1), we set $\alpha = \beta = 1$, such that the Beta distribution is a non-informative prior of $g(.)$. We follow the authors' advice in setting $T > 100$. As for θ, we will likely need to test each $c \in C$ using several values in the range $[0.75, 1)$.

4.2.2 *Adaptive Monte Carlo sampling*

Let C be our configuration parameter space, and $c^* \in C$ be a point decided true by the Bayesian statistical model checker.

4.2.2.1 *Ensembles of weak learners (EWL) using MC-Boost*

The idea here is to deploy a large ensemble of weak learners, each of which myopically homes in on the locality of the first discovered c^* points, and take the union of their discovered subsets. We implement a single weak learner in the MC-Boost algorithm. The algorithm takes four parameters: d for the number of dimensions, m for the number of samples, σ for the Gaussian reward/penalty function, and w for the weight of the Gaussian penalty function. We initialize the normalized PDFs for each of the d axes. Then we adaptively sample m points according to evolving normalized PDFs in the following way. Sample a point $c \in C$ by sampling d coordinates according to d normalized PDFs. Render a result according to a decision process: our demonstration below uses a toy 2D space within which there are two solution regions to be discovered; the real decision process would be the Bayesian statistical model checker running multiple simulations for a given $c \in C$. If the result is true, then: (1) record the true sample in the appropriate place; and (2) place a Gaussian reward upon each axis, centered at each corresponding coordinate, having σ. Otherwise, record the false sample in the appropriate place. In either case, renormalize the evolved PDFs. Then continue sampling.

For our implementation, see the code listing in Section D.2.

4.2.2.2 *Ensembles of constrained random walkers (ECRW) using MC-Walk*

The idea here is to sample C until discovering a c^* point, then deploy a large ensemble of constrained random walkers, each of which explores the locality of the c^* point for some time, then general sampling across C continues; we take the union of their discovered subsets. We implement a single random walker in the MC-Walk algorithm. The algorithm takes four parameters: d for the number of dimensions, m for the number of samples, p for the diffusion rate of the random walker, and n for the number of random walk steps taken by the random walker. We initialize the normalized PDFs for each of the d axes. Then we adaptively sample m points according to evolving normalized PDFs in the following way. Sample a point $c \in C$ by sampling d coordinates according to d normalized PDFs. Render a result according to a decision process: our demonstration below uses a toy 2D space within which there are two solution regions to be discovered; the real decision process would be the Bayesian statistical model checker running multiple simulations for a given $c \in C$. If the result is true, then: (1) record the true sample in the appropriate place; and (2) walk c^* for n steps along d dimensions using diffusion rate p. Otherwise, record the false sample in the appropriate place. Then continue sampling. However, within step (2), for each of those n walk steps we do the following, reflecting the outer loop: (1) render a result according to the same decision process; and (2) if true, then record the true sample in the appropriate place; otherwise, record the false sample in the appropriate place.

For our implementation, see the code listing in Section D.3.

4.2.2.3 *Iterative PCA (IPCA)*

Though we leave it for exploratory future work, we facilitate the discussion below by mentioning our idea for an iterative principle components analysis (PCA) algorithm. The idea is that after a Monte Carlo sampling process accumulates a sufficient number of c^* points, it performs PCA to reduce dimensionality. The number of principal components is less than or equal to the number of original variables, such that the first principal component has the largest possible variance—it accounts for as much of the variability in the data as possible—and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (uncorrelated with) the preceding components. PCA effectively incorporates a form of penalty since it is based upon the scree plot where each successive principal component receives less weight. It is a common approach to reducing dimensionality in many problem domains. Our algorithm will continue sampling until sufficiently many c^* points are accumulated to repeat the application of PCA. And so on, iteratively.

4.2.2.4 *Comparing the three algorthims*

ECRW and EWL do not perform dimensionality reductions like IPCA. In EWL, each myopic adaptive sampler in the ensemble suffers from trails of samples along each dimension that lead to the cluster but are not inside of it (inefficiency). Since the actual size of the clusters are unknown to the algorithm, using a global variance parameter for the Gaussian reward function will likely lead to either over-exploration inefficiency or overly constrained exploration (one variance does not fit all). ECRW does not suffer from this problem, since, if we deploy an ensemble of random walkers starting from the first hit in a potential cluster area, then these will diffusively spread from the inside-out rather than be constrained by a reward function from the outside-

in, where the outside is artificially assumed. However, unless an enormous number of random walkers are deployed, or each walker diffuses with a very high diffusion rate, this diffusion process is unlikely to discover the boundaries of the cluster area in high dimensions. We envision IPCA operating in a loop over two phases until some stopping criterion is met. Phase 1: collect some sufficient number of samples. Phase 2: perform PCA on the samples. Repeat. One problem with this approach is that PCA, once performed, cannot be reversed; the algorithm is locked into the new Eigen dimensions. There is no correction to gain what is lost with each reduction. In ECRW, each family of random walkers could be run in parallel. In EWL, each member of the ensemble could be run in parallel. To our knowledge, IPCA cannot be parallelized.

4.2.3 *Mean-variance threshold driver*

If our simulator embeds a functional integrator/detector, then it will return a similarity score function that indicates the hypoxia detection "high water mark" during the course of the simulation. Since our simulation is stochastic, for any given fixed $c \in C$, this returned value will fluctuate, and so constitutes a random variable in [0,1]. We assume that its value has a central tendency for any given $c \in C$ and thus, it can be measured and bounded over successive outcomes by its mean and standard deviation, or rather, by the dimensionless coefficient of variation, $CV = \frac{\sigma}{\mu}$. The mean-variance threshold driver is a simple method that decides to accept a mean value if and only if its CV is below some specified threshold by some cutoff number of measured outcomes. An accepted mean value associated with c is passed up to the MC-Branch-and-Bound sampler.

For our implementation, see the code listing in Section D.4.

4.2.4 *Monte Carlo branch-and-bound sampling*

Branch-and-bound [90, 106] is a general approach for finding optimal solutions of optimization problems, usually in discrete and combinatorial optimization. As such, it requires a metric or objective function to guide its pruning decisions. Our approach for developing this, outlined in the materials & methods in Chapter 3, is to construct a linear regression learning function, f, that characterizes hypoxia in terms of the image features we are able to extract. Linear functions are by definition monotonic, and ours gives a similarity score in the range [0,1]. We feel this is sufficient to enable the branch-and-bound approach taken here.

Our goal is to maximize $f(c)$, where $c \in C$, our space of solution candidates. Branch-and-bound requires two procedures. The first procedure *splits*, taking some set $C' \subseteq C$ and returning smaller sets $C'_1, ..., C'_n$ whose union covers C'. We observe that $\max_{c \in C'} f(c) = \max\{\mu_1, ..., \mu_n\}$, where $\mu_i = \max_{c \in C'_i} f(c), i = 1, ..., n$. This enables the *branching* step, so named since the recursive application of splitting defines a search tree whose nodes are the subsets of C. The second procedure computes the upper and lower *bounds* for $\max_{c \in C'} f(c)$, for some $C' \subseteq C$. The branch-and-bound *pruning* principle is this: if the *upper* bound for some search tree node, U, is less than the *lower* bound of some other search tree node, V, then we may safely remove U from the search space. Recursion halts when $|C'| = 1$ or when $\min_{c \in C'} f(c) = \max_{c \in C'} f(c)$, ensuring any $c \in C' = \max_{c' \in C'} f(c')$.

We adapt the traditional branch-and-bound algorithm in the following way. The *bound* step computes the upper and lower bounds not by exhaustively evaluating the

subset C'_i, but instead by Monte Carlo sampling within C'_i, thereby giving the bounds on $\max_{c \in C'_s} f(c)$, for some sampled subset $C'_s \subseteq C'_i$. This adaptation to use sampling defines our MC-Branch-and-Bound algorithm.

For our implementation, see the code listing in Section D.5.

4.3 DEMONSTRATIONS & DISCUSSION

4.3.0.1 *The toy system*

To illustrate the way these adaptive methods work and how they perform, we will use a toy version of the actual problem, consisting in 2 parameters, whose values are normalized to [0,1], and which contains two solution areas, a rectangle (at center bottom) and a smaller circle (at upper right). We use a toy system for two reasons. First, these algorithms are easier to visualize in 2D than in a higher dimensional space. Second, two phases of future work must first be completed—characterizing hypoxia by a stable set of image/simulator features, and implementing the derived embedded feature measurers and integrators in the simulation—before we can test these algorithms on the high-dimensional data for our problem.

4.3.1 *Ensembles of weak learners using MC-Boost*

4.3.1.1 *Setup*

We demonstrate the MC-Boost algorithm for a range of parameters. The first set of 8 images correspond to the algorithm's myopic discovery of one solution area; the second set of 8 images correspond to the other solution area. Each set of 8 images is

broken down into two sets of 4 images. The first set of 4 images correspond to 100 sample points; the second set of 4 images correspond to 1000 sample points. Each set of 4 images explores two values of two parameters: boost weight $\in \{1, 100\}$ (x-axis) and $\sigma \in \{0.1, 0.01\}$ (y-axis). See Figure 30 and Figure 31 and Figure 32 and Figure 33.

4.3.1.2 *Demonstration*

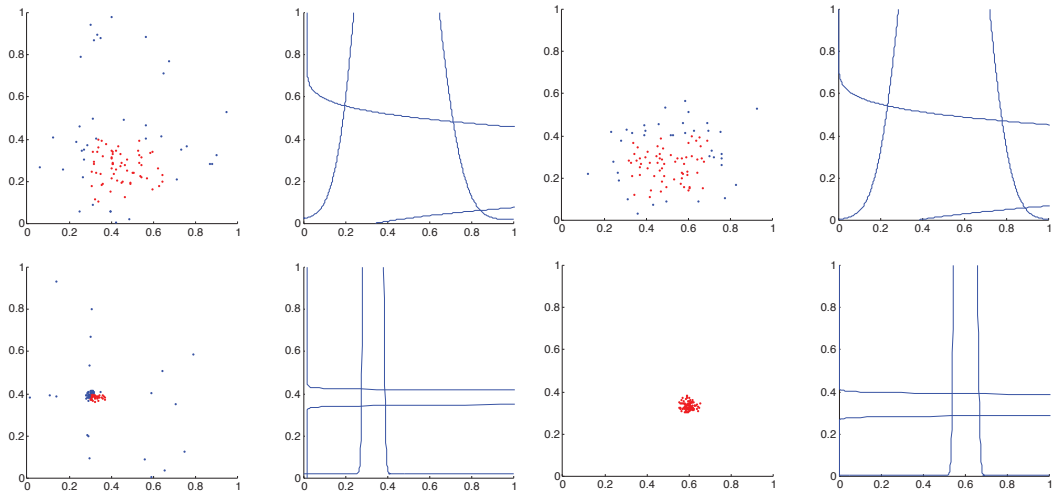

Figure 30: MC-Boost: solution area #1, 100 sample points, x-axis indicates boost weight: 1 or 100, y-axis indicates σ: 0.1 or 0.01.

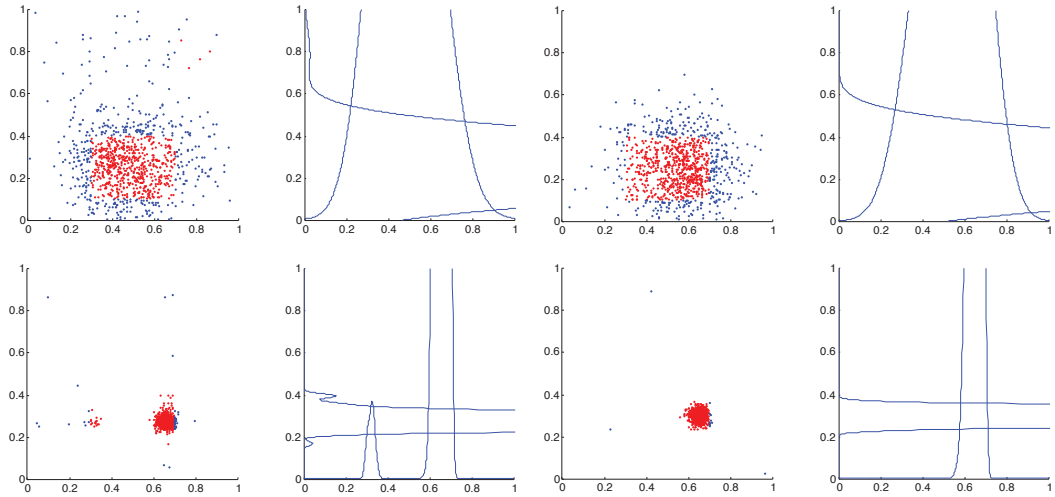

Figure 31: MC-Boost: solution area #1, 1000 sample points, x-axis indicates boost weight: 1 or 100, y-axis indicates σ: 0.1 or 0.01.

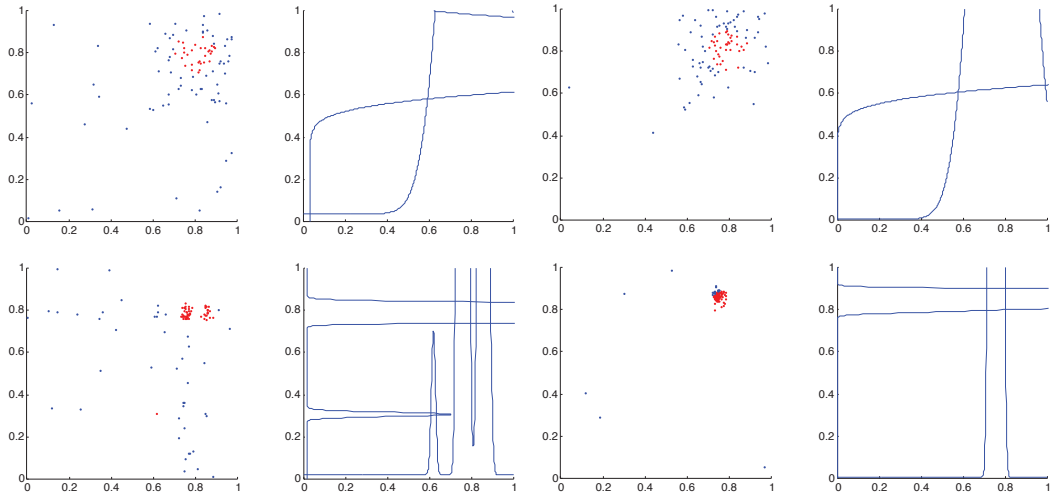

Figure 32: MC-Boost: solution area #2, 100 sample points, x-axis indicates boost weight: 1 or 100, y-axis indicates σ: 0.1 or 0.01.

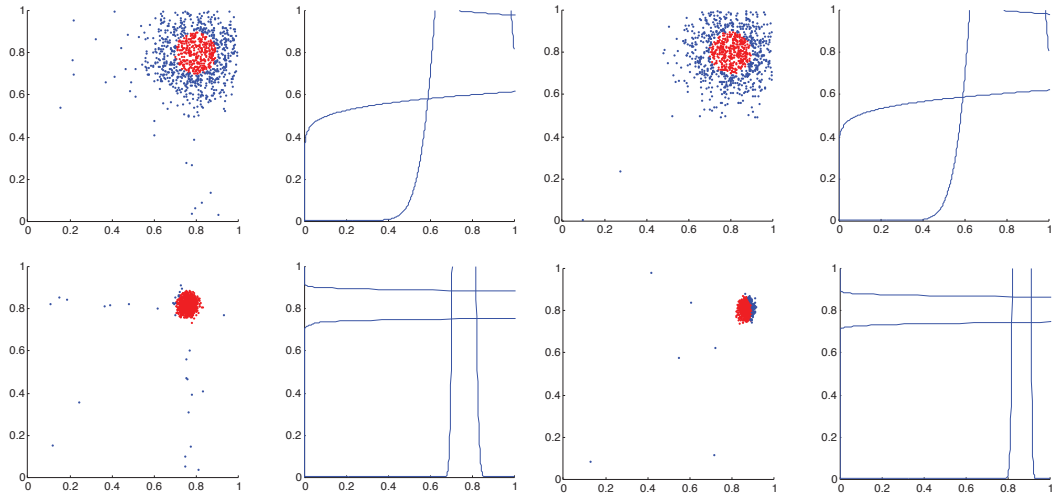

Figure 33: MC-Boost: solution area #2, 1000 sample points, x-axis indicates boost weight: 1 or 100, y-axis indicates σ: 0.1 or 0.01.

4.3.1.3 *Discussion*

The basic trade-offs are evident in the figures. Holding boost weight fixed, increasing σ broadens coverage, at the potential cost of exploring areas outside of the target zone; while narrowing σ may force the algorithm to explore only a subspace of the solution. Holding σ fixed, increasing the boost weight reigns in the signature trails of samples along each dimension, at the potential cost of forcing the algorithm the explore on a subspace of the solution. Increasing the number of sample points amplifies this pattern, which holds across both solution areas.

4.3.2 *Ensembles of constrained random walkers using MC-Walk*

4.3.2.1 *Setup*

We demonstrate the MC-Walk algorithm for a range of parameters. The first set of 4 images correspond to 100 sample points; the second set of 4 images correspond to 1000 sample points. Each set of 4 images explores two values of two parameters: walker steps $\in \{10, 100\}$ (x-axis) and diffusion rate $\in \{0.0001, 0.00001\}$ (y-axis). See Figure 34 and Figure 35.

4.3.2.2 *Demonstration*

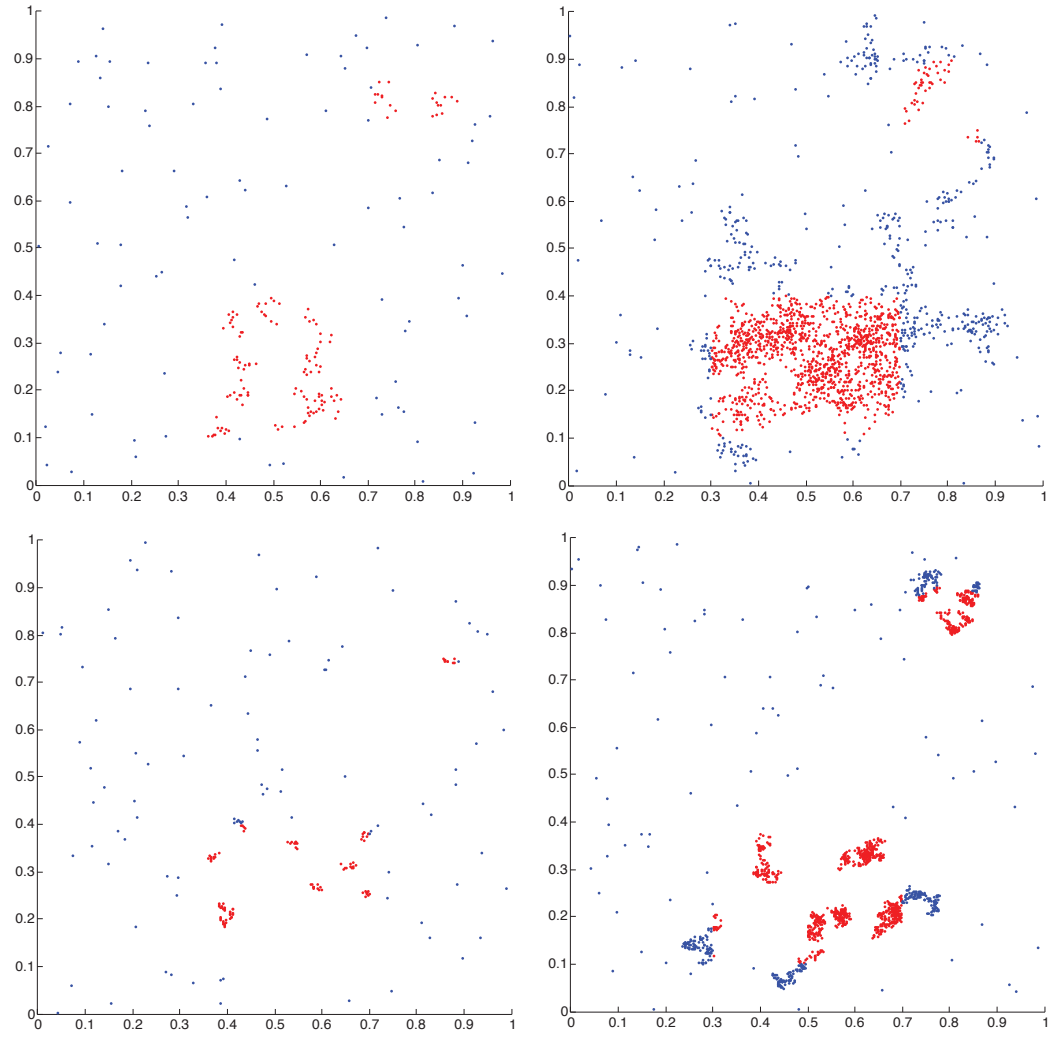

Figure 34: MC-Walk: 100 sample points, x-axis indicates walker steps: 10 or 100, y-axis indicates diffusion rate: 0.0001 or 0.00001.

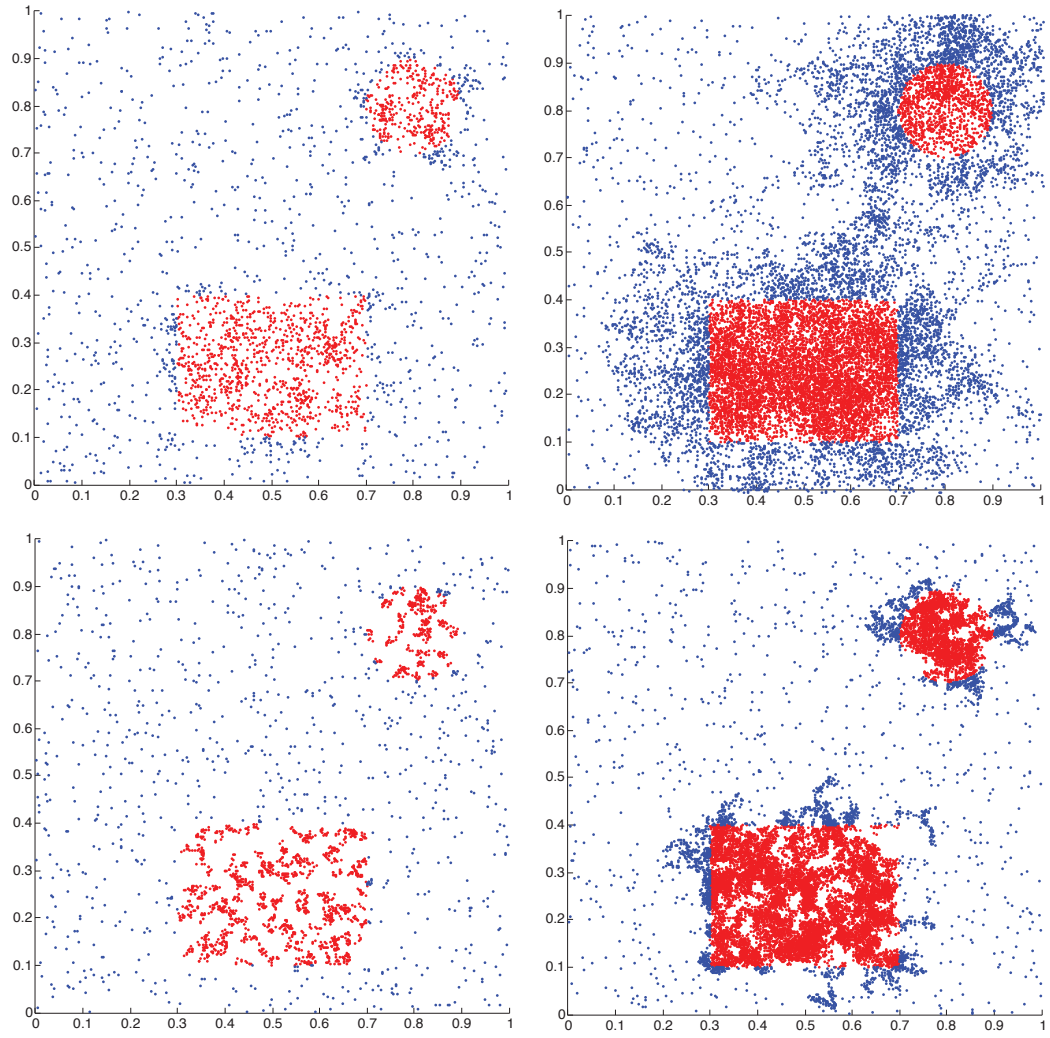

Figure 35: MC-Walk: 1000 sample points, x-axis indicates walker steps: 10 or 100, y-axis indicates diffusion rate: 0.0001 or 0.00001.

4.3.2.3 *Discussion*

The basic trade-offs are evident in the figures. Holding walker steps fixed, increasing the diffusion rate increases the span of exploration at the potential cost of lowering the concentration of hits. Holding diffusion rate fixed, increasing the number of

walker steps does not change the concentration of hits much but does increase the coverage. Increasing the number of sample points amplifies this pattern.

4.3.3 *MC-Branch-and-Bound*

4.3.3.1 *Setup*

We implement MC-Branch-and-Bound on an image, whose sampled 8-bit intensity values ($I(x,y) \in [0,255]$), when normalized, represent the return value of a decision process giving outcome [0,1]. We use the quad-tree decomposition as the *split* procedure, where any given frame of candidate pixels is recursively subdivided on the basis of a splitting decision. This decision is the *bound* procedure and works as follows. Each frame is sampled some number of times. We chose to sample $2^{(7-d)}$ times, where d is the depth, or ply index, starting at 0. Thus, the original image is sampled 128 times, then, if considered, each quadrant frame is sampled 64 times, etc. We use a recursive depth cutoff of $n = 6$. After sampling, the minimum, I_{min}, and maximum, I_{max}, sampled intensity values are computed and compared to a global minimum value, T, initialized to 0. If $I_{min} > T$ then $T = I_{min}$. Next, if $I_{max} > 0$ and $I_{max} \geqslant T$ and $d < n$, then we decompose the current frame into quadrants; otherwise we do not, effectively pruning this frame from the search space. We blur the original solution image on which the algorithm runs, to produce gradients of sample values between 0 and 255; this reflects our belief that the real parameter space is not rough and discontinuous in its hypoxia-similarity values, as given by $f(c)$. Figure 36 shows the quad-tree-based MC-Branch-and-Bound result on a blurred "solution image".

4.3.3.2 *Demonstration*

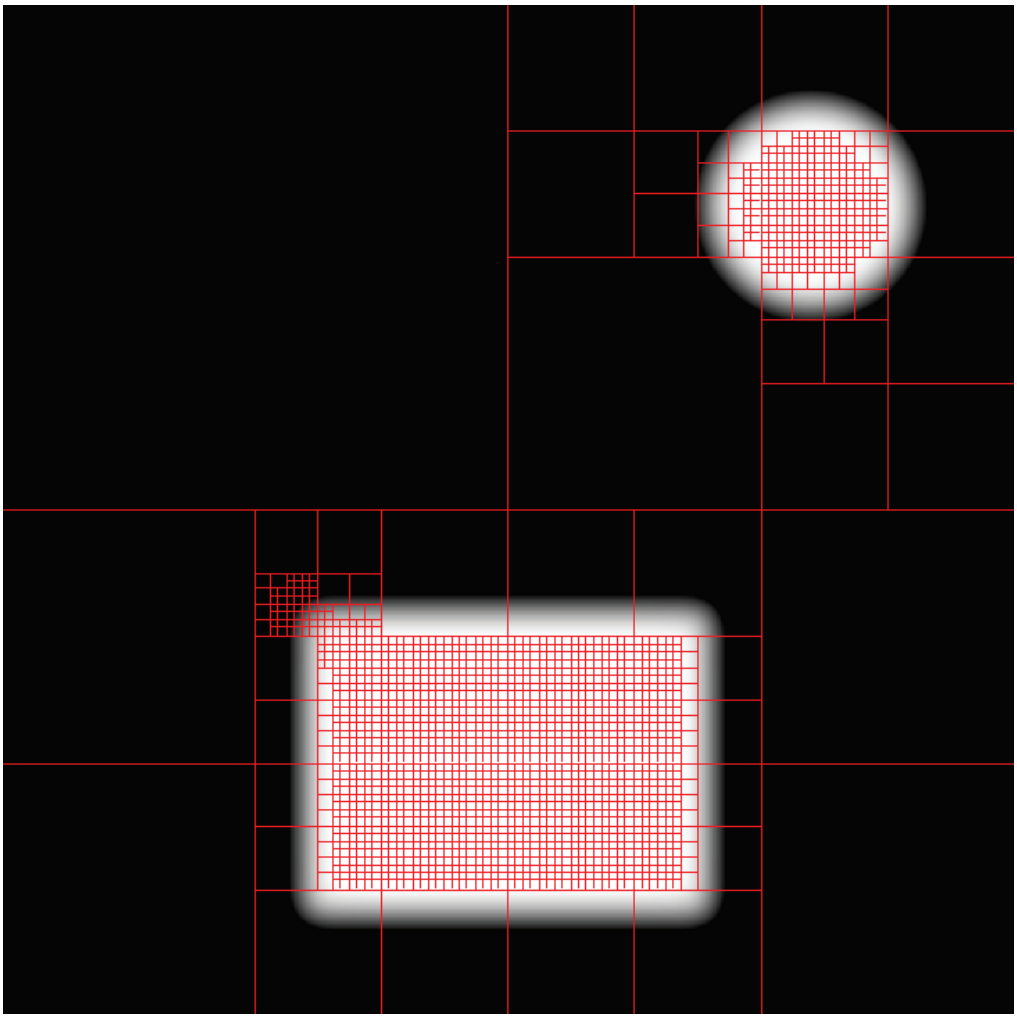

Figure 36: MC-Branch-and-Bound: search results for an underlying "solution image".

4.3.3.3 *Discussion*

The initial 128 samples sufficed for MC-Branch-and-Bound to home in on the solution areas within three recursive splits.

4.4 CONCLUSIONS & FUTURE WORK

4.4.1 *Conclusions*

4.4.1.1 *Our contributions*

We have proposed three adaptations to the Monte Carlo sampling algorithm to suit our needs for this problem, which is really one about adaptive sampling through feedback, where the values of the points govern the adaptation. These are MC-Boost, MC-Walk, and MC-Branch-and-Bound. While these do not pose fundamental algorithmic innovations, they are a push in the right direction for our problem domain.

4.4.2 *Future work*

4.4.2.1 *To extend and enhance*

INFORMATIVE PRIORS We would next like to incorporate biological domain knowledge to constrain our parameter configuration space, C. Every parameter distribution need not be uniform. Knowing the probable and improbable pre-hypoxic tumor tissue structure, for example, would shear away a large set of parameters pertaining to initial cell positions in space, as would considering the physiological *in vivo* norms for oxygen diffusion, and oxygen and nutrient consumption rates in different human tumor tissues. For tissue culture simulations, we would like to use the NCI-60 CORE profile [140, 72] for establishing *in vitro* norms for oxygen and nutrients in different human tumor tissues.

4.4.2.2 *To explore*

4.4.2.3 *Reducing parameter space dimensionality*

As noted earlier, we would like to explore developing an iterated PCA algorithm for the adaptive sampling problem. Along these lines, perhaps there is a role James-Stein shrinkage [74, 138, 33] could play in the other Monte Carlo-based algorithms that would lend itself to iterative dimension reduction.

A

SIMULATOR EXAMPLES

A.1 SYMMETRIC MUTUAL NEED (2D)

A.1.1 *Setup*

In this simulation, we have two cell types, α and β. They each release one particle type that the other consumes. The release rates are higher than their respective consumption rates, so that concentrations will not diminish too quickly. Each particle type consumed affects fitness in a positive way. All rates and impact factors are symmetric. Hence, this situation reflects symmetric mutual need between two cell types. Both cell types are replaceable and reproductive. The specific quantities mentioned here are given in the tables below.

A.1.1.1 *Configuration parameters*

pt 1	*pt 2*
0.1	0.1

Table 8: Diffusion rate of each particle type.

pt 1	*pt 2*
0.1	0.1

Table 9: Initial concentration of each particle type.

cell type	*pt 1*	*pt 2*
vessel	0	0
empty	0	0
α	0	0
β	0	0

Table 10: Basal lower-bound concentration of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	∞	∞
empty	∞	∞
α	∞	∞
β	∞	∞

Table 11: Basal upper-bound concentration of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	0	0
empty	0	0
α	0.1	0
β	0	0.1

Table 12: Consumption rate of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	0	0
empty	0	0
α	0	0.2
β	0.2	0

Table 13: Release rate for of particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	0	0
empty	0	0
α	2	0
β	0	2

Table 14: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
α	Yes
β	Yes

Table 15: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
α	Yes
β	Yes

Table 16: Reproductive predicate of each cell type.

A.1.1.2 *Initial conditions*

The simulation opens with the initial concentrations of both particle types specified in Table 9. These diffuse with rates specified in Table 8. We initialize the 2D lattice with a random mixture of *empty*, α, and β cells.

A.1.2 *Results*

A.1.2.1 *Spatial cell population evolution*

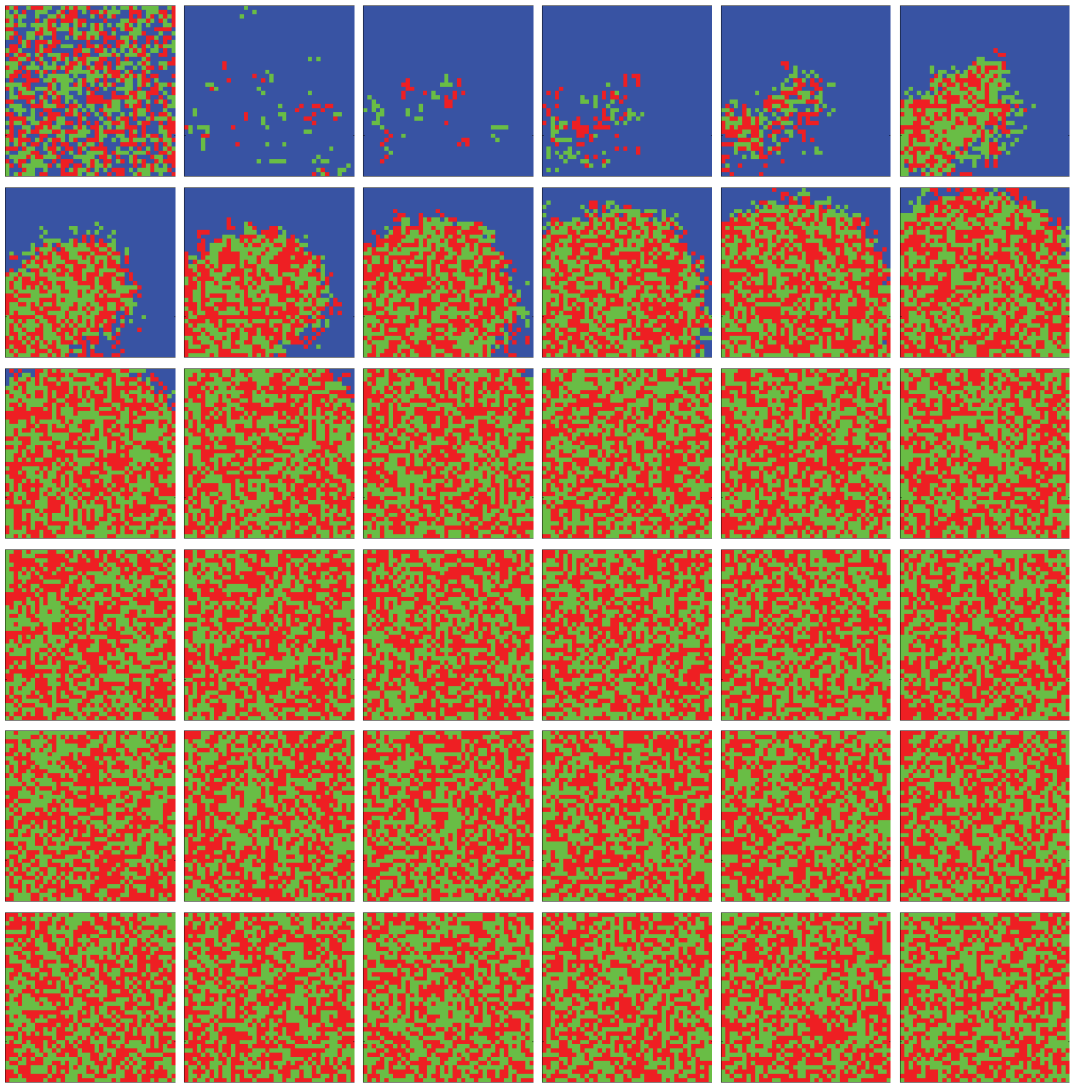

Figure 37: Time evolution of cell populations. Left-to-right, top-to-bottom: 500 generations shown in 36 frames (t = 1, 15, 29, 43, 57, 71, 85, 99, 113, 127, 141, 155, 169, 183, 197, 211, 225, 239, 254, 268, 283, 297, 312, 326, 341, 355, 370, 384, 399, 413, 428, 442, 457, 471, 486, 500). Key: *vessel* (white), *empty*, α, β.

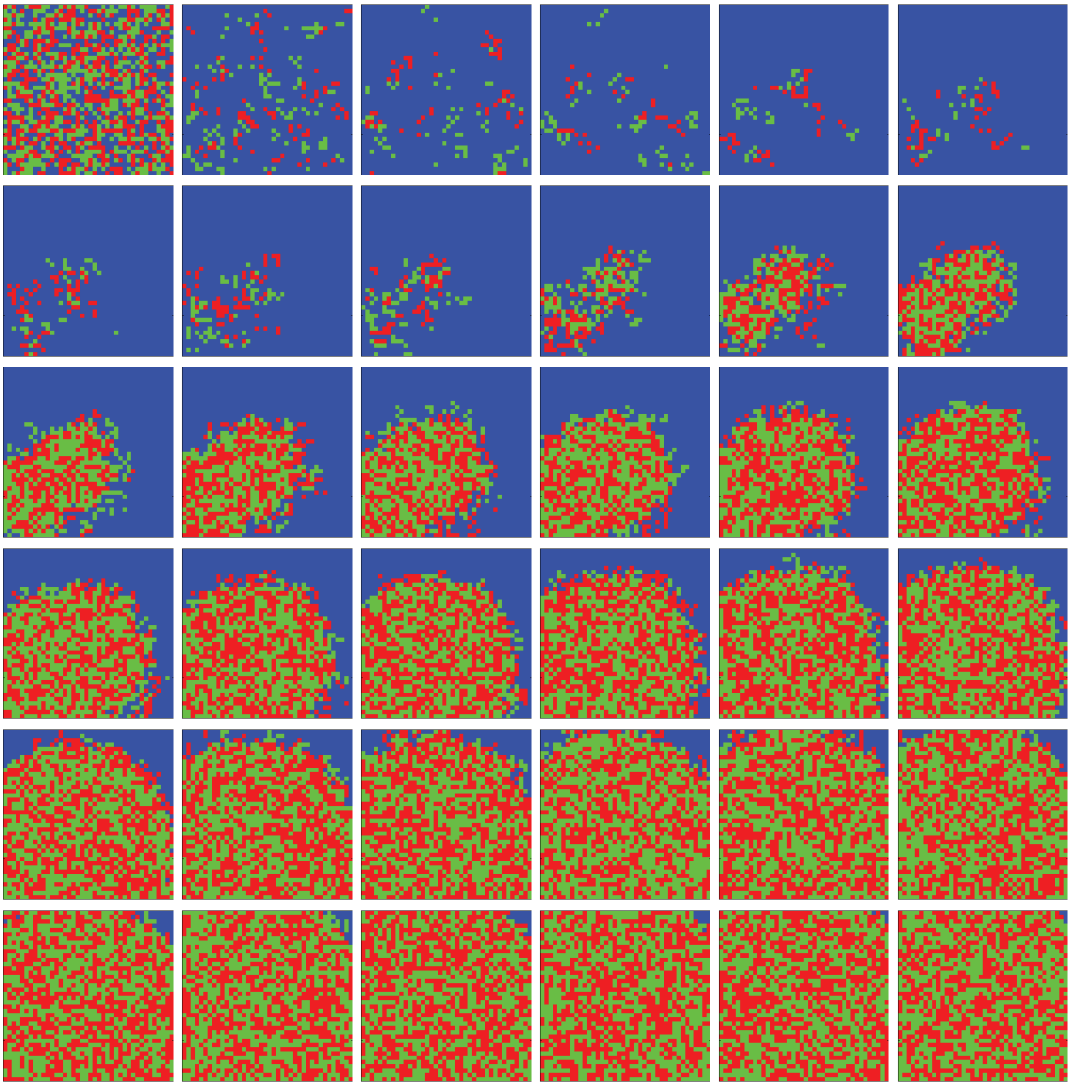

Figure 38: Time evolution of cell populations. Left-to-right, top-to-bottom: 200 generations shown in 36 frames (t = 1, 7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 90, 96, 101, 107, 112, 118, 123, 129, 134, 140, 145, 151, 156, 162, 167, 173, 178, 184, 189, 195, 200). Key: *vessel* (white), *empty*, α, β.

A.1.2.2 *Temporal cell population evolution*

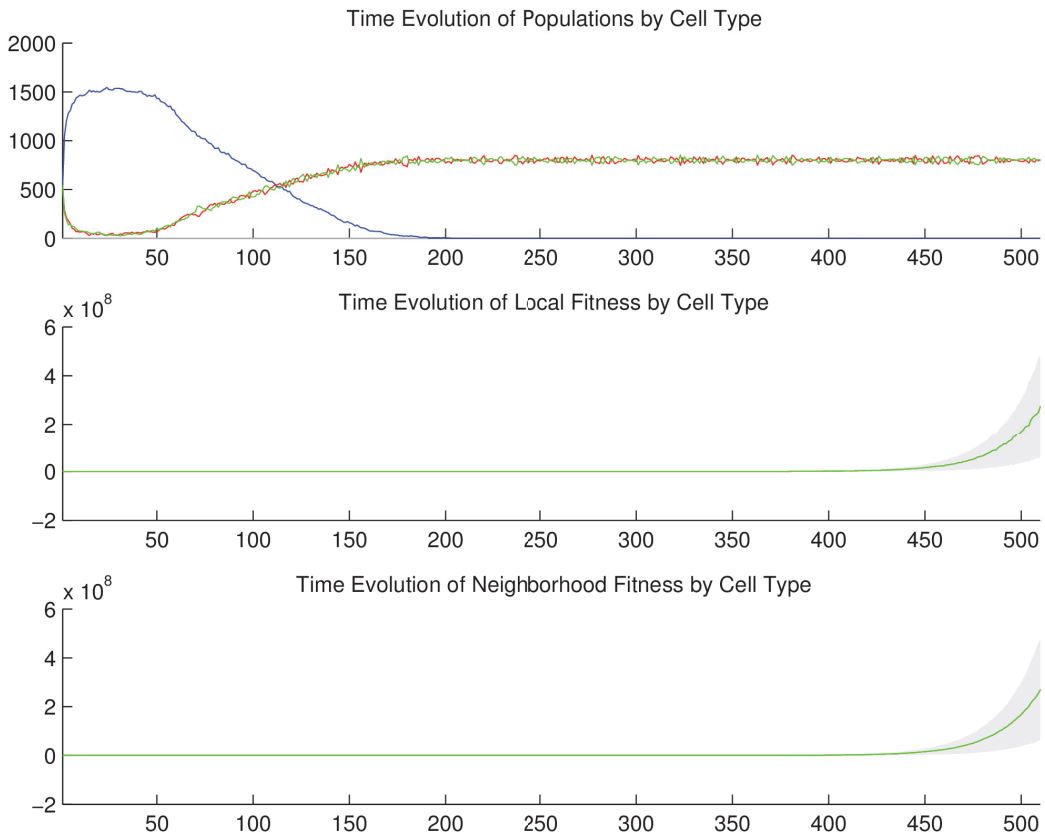

Figure 39: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, α, β.

A.1.3 *Discussion*

We observe in Figure 37 and Figure 38 that the α and β cell populations die back to near extinction at first, given the low initial concentrations of particles required

for their fitness. Then some clusters of α and β cells form, and the local concentrations of particles accumulates to a point that surpasses their consumption rates, supporting proliferation of both cell types simultaneously and reciprocally. In Figure 39 we observe that from generation 25 to 150, as the clusters congeal and begin to take over the space, but prior to their fully populating the space, the α and β cell population sizes oscillate with a similar period but a small time delay, suggesting a Lotka-Volterra ("predator-prey") [99, 147] population dynamics has emerged. After generation 150, the time delay between the two oscillations begins to vanish as the population dynamics necessarily becomes zero-sum in character. By generation 175, the two populations have converged at, and oscillate about, the same mean size, which continues indefinitely.

A.2 SYMMETRIC FITNESS WITH ONE VESSEL (2D)

A.2.1 *Setup*

In this simulation, we have two cell types, α and β. They both release no particles, and both consume the same two particle types at the same rate, which affect their fitness in a positive way to the same extent. All rates and impact factors are symmetric. The one vessel that extends down from the top-middle consumes no particle types and releases both particle types. Both cell types are replaceable and reproductive. The specific quantities mentioned here are given in the tables below.

A.2.1.1 *Configuration parameters*

pt 1	*pt 2*
10.0	10.0

Table 17: Diffusion rate of each particle type.

pt 1	*pt 2*
0	0

Table 18: Initial concentration of each particle type.

cell type	*pt 1*	*pt 2*
vessel	0.1	0.1
empty	0.1	0.1
α	0.1	0.1
β	0.1	0.1

Table 19: Basal lower-bound concentration of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	∞	∞
empty	∞	∞
α	∞	∞
β	∞	∞

Table 20: Basal upper-bound concentration of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	0	0
empty	0	0
α	0.1	0.1
β	0.1	0.1

Table 21: Consumption rate of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	1.0	1.0
empty	0	0
α	0	0
β	0	0

Table 22: Release rate for of particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	0	0
empty	0	0
α	0.8	0.8
β	0.8	0.8

Table 23: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
α	Yes
β	Yes

Table 24: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
α	Yes
β	Yes

Table 25: Reproductive predicate of each cell type.

A.2.1.2 *Initial conditions*

The simulation opens with initial concentrations of both particle types set to zero, as specified in Table 18, and lower-bounded basal concentrations of both particle types, as specified in Table 19. These diffuse with rates specified in Table 17. We initialize the 2D lattice with a random mixture of *empty*, α, and β cells, and place one vessel extending down from the top-middle.

A.2.2 *Results*

A.2.2.1 *Spatial cell population evolution*

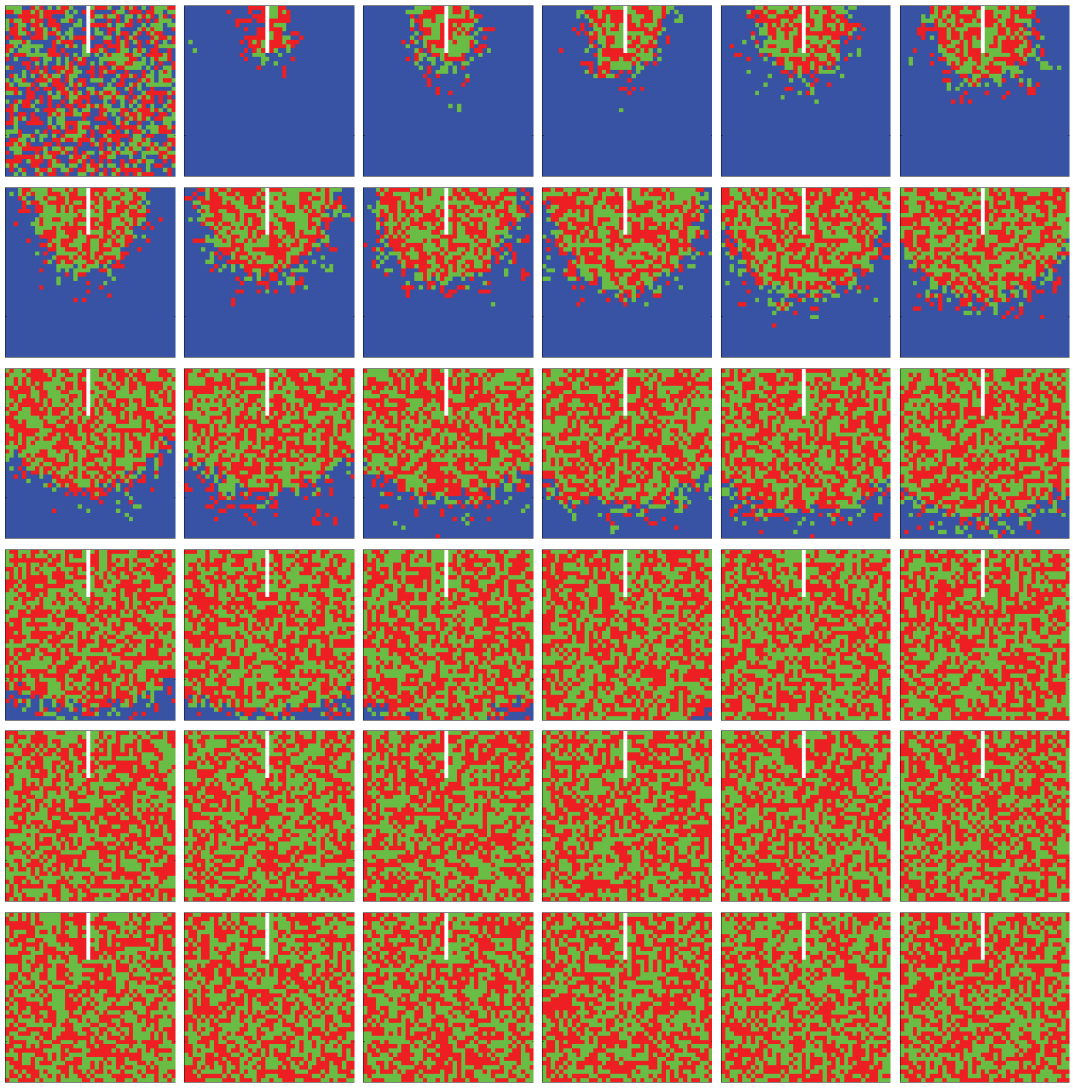

Figure 40: Time evolution of cell populations. Left-to-right, top-to-bottom: 660 generations shown in 36 frames (t = 1, 20, 39, 58, 77, 96, 115, 134, 153, 172, 191, 210, 229, 248, 267, 286, 305, 324, 343, 362, 381, 400, 419, 438, 457, 475, 494, 512, 531, 549, 568, 586, 605, 623, 642, 660). Key: *vessel* (white), *empty*, α, β.

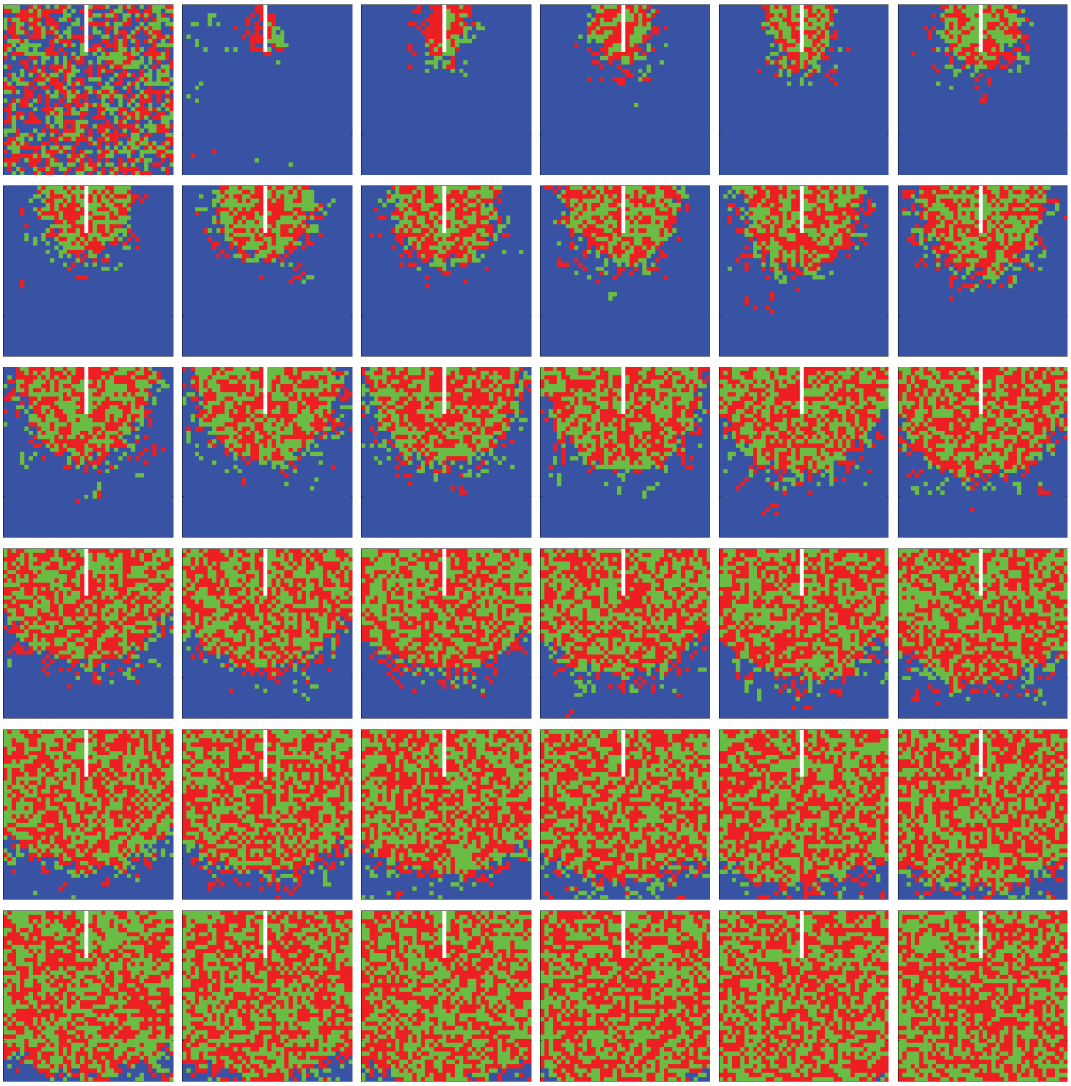

Figure 41: Time evolution of cell populations. Left-to-right, top-to-bottom: 420 generations shown in 36 frames (t = 1, 13, 25, 37, 49, 61, 73, 85, 97, 109, 121, 133, 145, 157, 169, 181, 193, 205, 217, 229, 241, 253, 265, 277, 289, 301, 313, 325, 337, 349, 361, 373, 385, 397, 409, 420). Key: *vessel* (white), *empty*, α, β.

A.2.2.2 *Temporal cell population evolution*

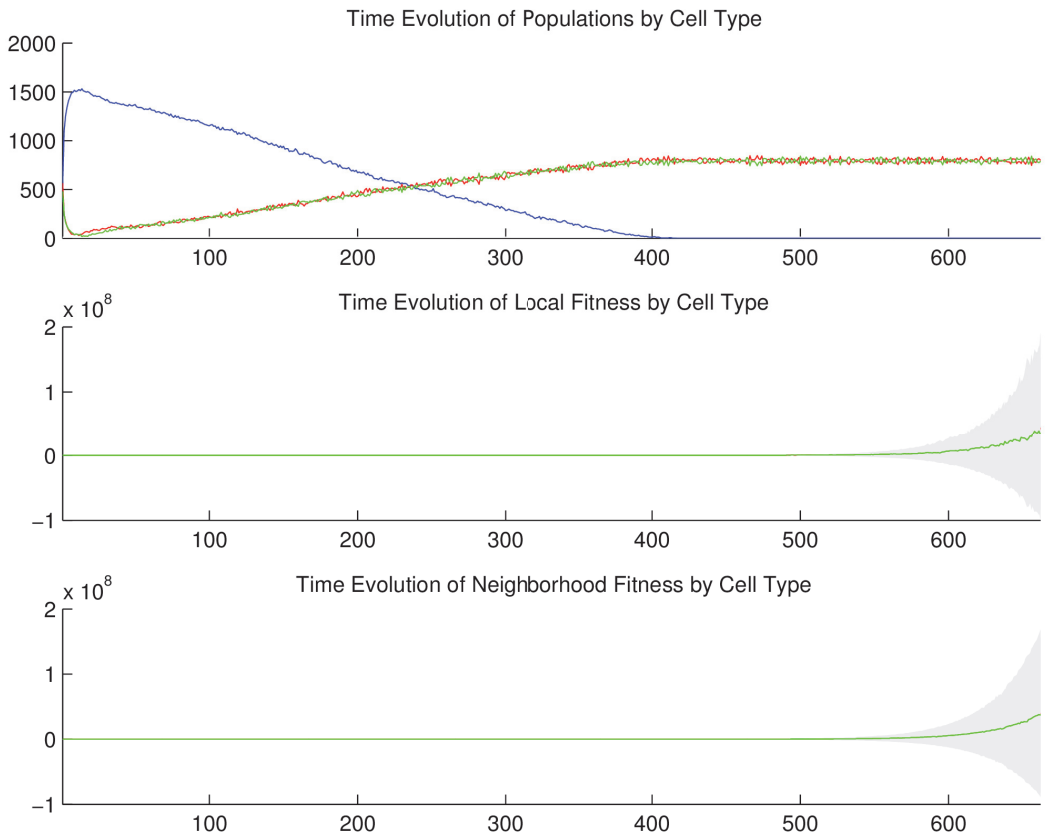

Figure 42: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, α, β.

A.2.3 *Discussion*

We observe in Figure 40 and Figure 41 that the α and β cell populations die back to near extinction at first, given the low basal concentrations of particles required for

their fitness, and their high diffusion rates. Then some clusters of α and β cells form by chance since they are near the vessel, and the local concentrations of particles accumulates to a point that surpasses their consumption rates, supporting proliferation of both cell types simultaneously but not reciprocally. In Figure 42 we observe that from generation 10 to 400, as the clusters separately begin to take over the space, but prior to their fully populating the space, the α and β cell population sizes oscillate with no observable relationship to each other, as we expect from symmetric but independent growth. After generation 400, the population dynamics necessarily becomes zero-sum in character. By this time, the two populations have converged at, and oscillate about, the same mean size, which continues indefinitely. Notice the significant difference in the two convergence times: symmetric mutual need is generation 175; symmetric fitness with one vessel is generation 400.

A.3 ASYMMETRIC FITNESS WITH ONE VESSEL (2D)

A.3.1 *Setup*

In this simulation, we have two cell types, α and β. They both release no particles, and both consume the same two particle types at the same rate, which affect their fitness in a positive way, but to different extents. All rates are symmetric. The one vessel that extends down from the top-middle consumes no particle types and releases both particle types. Both cell types are replaceable and reproductive. The specific quantities mentioned here are given in the tables below.

A.3.1.1 *Configuration parameters*

pt 1	*pt 2*
10.0	10.0

Table 26: Diffusion rate of each particle type.

pt 1	*pt 2*
0	0

Table 27: Initial concentration of each particle type.

cell type	*pt 1*	*pt 2*
vessel	0.1	0.1
empty	0.1	0.1
α	0.1	0.1
β	0.1	0.1

Table 28: Basal lower-bound concentration of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	∞	∞
empty	∞	∞
α	∞	∞
β	∞	∞

Table 29: Basal upper-bound concentration of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	0	0
empty	0	0
α	0.1	0.1
β	0.1	0.1

Table 30: Consumption rate of each particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	1.0	1.0
empty	0	0
α	0	0
β	0	0

Table 31: Release rate for of particle type by cell type.

cell type	*pt 1*	*pt 2*
vessel	0	0
empty	0	0
α	0.8	0.8
β	1.5	1.5

Table 32: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
α	Yes
β	Yes

Table 33: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
α	Yes
β	Yes

Table 34: Reproductive predicate of each cell type.

A.3.1.2 *Initial conditions*

The simulation opens with initial concentrations of both particle types set to zero, as specified in Table 27, and lower-bounded basal concentrations of both particle types, as specified in Table 28. These diffuse with rates specified in Table 26. We initialize the 2D lattice with a random mixture of *empty*, α, and β cells, and place one vessel extending down from the top-middle.

A.3.2 *Results*

A.3.2.1 *Spatial cell population evolution*

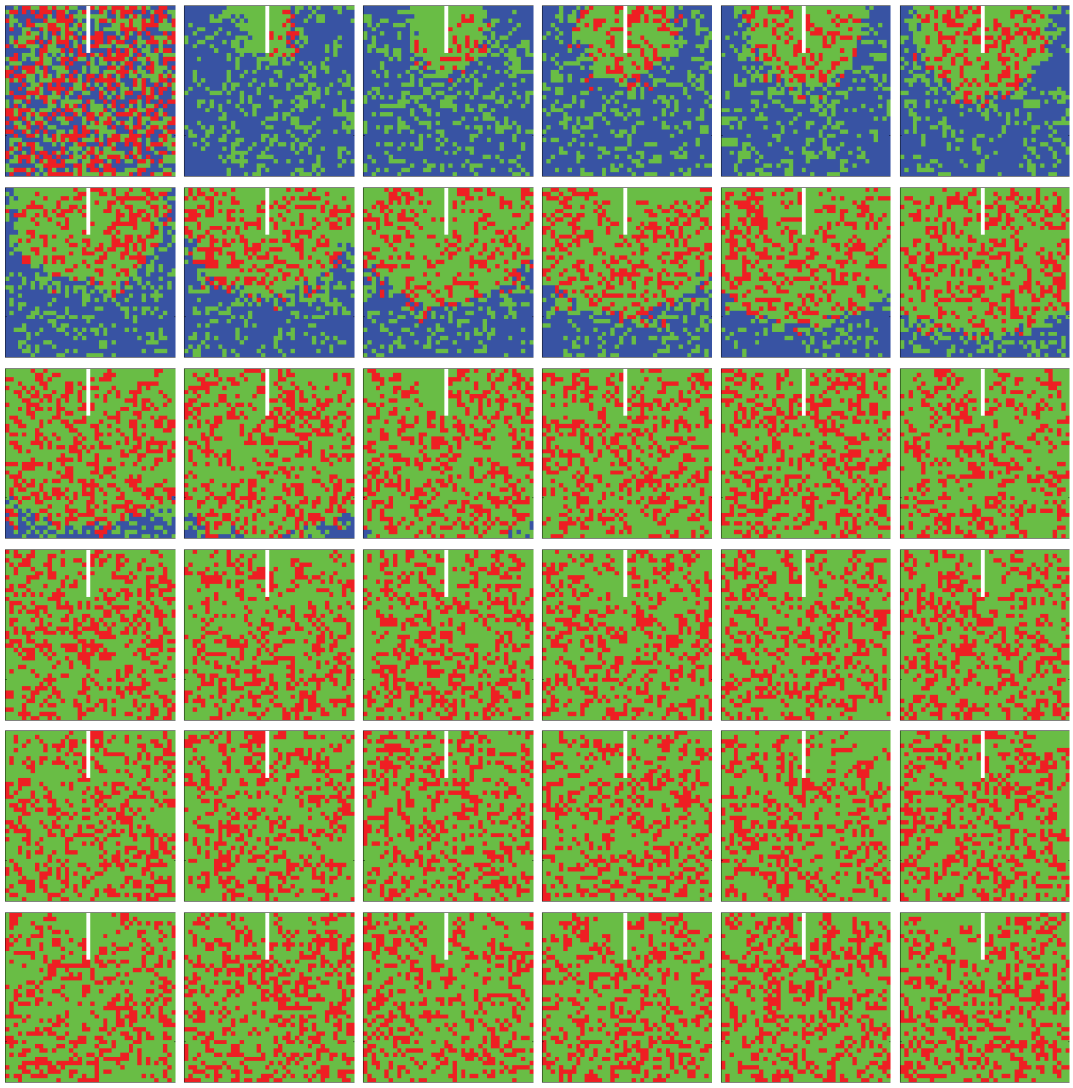

Figure 43: Time evolution of cell populations. Left-to-right, top-to-bottom: 1000 generations shown in 36 frames (t = 1, 30, 59, 88, 117, 145, 174, 202, 231, 259, 288, 316, 345, 373, 402, 430, 459, 487, 516, 544, 573, 601, 630, 658, 687, 715, 744, 772, 801, 829, 858, 886, 915, 943, 972, 1000). Key: *vessel* (white), *empty*, α, β.

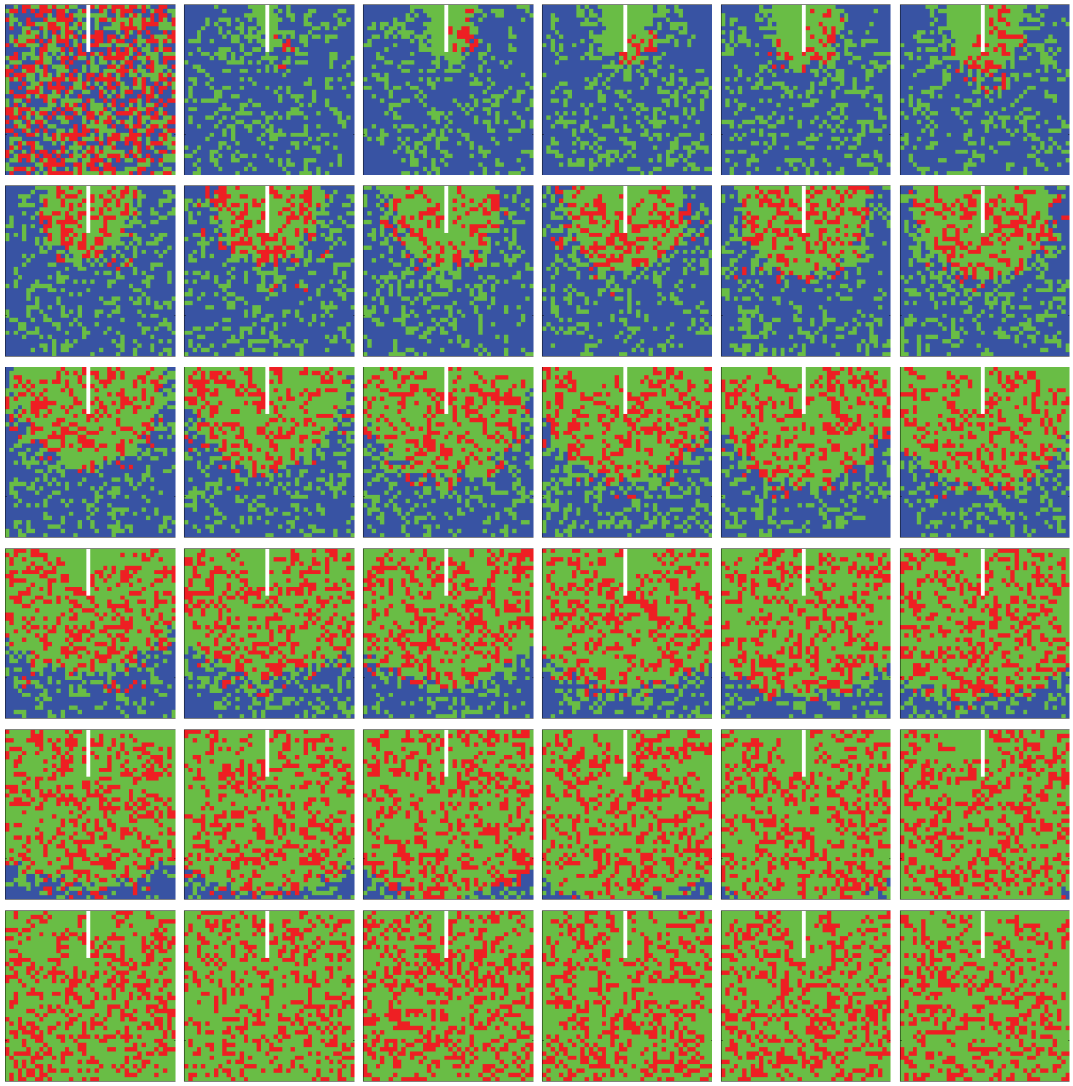

Figure 44: Time evolution of cell populations. Left-to-right, top-to-bottom: 500 generations shown in 36 frames (t = 1, 15, 29, 43, 57, 71, 85, 99, 113, 127, 141, 155, 169, 183, 197, 211, 225, 239, 254, 268, 283, 297, 312, 326, 341, 355, 370, 384, 399, 413, 428, 442, 457, 471, 486, 500). Key: *vessel* (white), *empty*, α, β.

A.3.2.2 *Temporal cell population evolution*

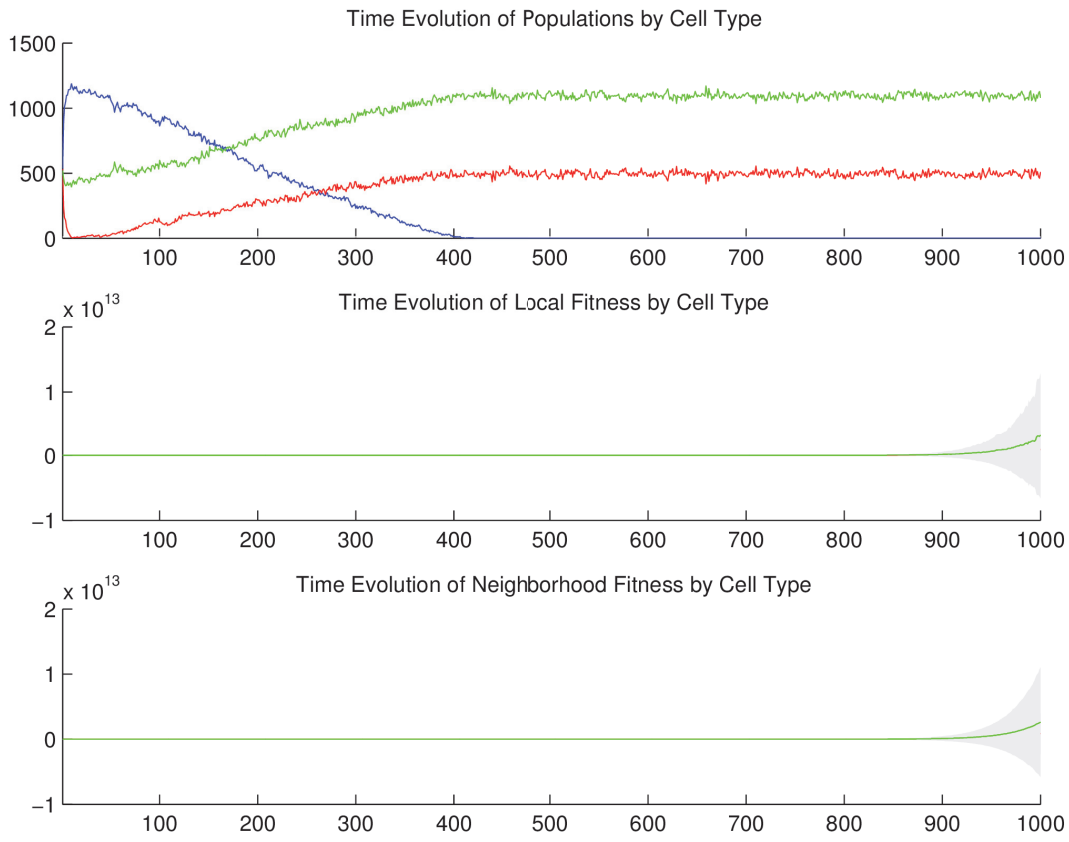

Figure 45: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, α, β.

A.3.3 *Discussion*

We observe in Figure 43 and Figure 44 that the α and β cell populations die back to near extinction at first, given the low basal concentrations of particles required for

their fitness, and their high diffusion rates. Then some clusters of α and β cells form by chance since they are near the vessel, and the local concentrations of particles accumulates to a point that surpasses their consumption rates, supporting proliferation of both cell types simultaneously but not reciprocally. In Figure 45 we observe that from generation 10 to 400, as the clusters separately begin to take over the space, but prior to their fully populating the space, the α and β cell population sizes oscillate with no observable relationship to each other, as we expect from independent growth. But the populations are of different proportions to the whole; as we expect, the population of β is proportionally larger than that of α. What we did not expect is that the two populations would grow at the same rate. After generation 400, the population dynamics necessarily becomes zero-sum in character. By this time, the two populations have converged at, and oscillate about, distinct mean sizes, and continue to coexist indefinitely—something else we did not expect. Notice the significant difference in the two convergence times: symmetric mutual need is generation 175; symmetric (and asymmetric) fitness with one vessel is generation 400.

A.4 OXIDATIVE PHOSPHORYLATION VS AEROBIC GLYCOLYSIS (2D)

A.4.1 *Setup*

In this simulation, we have *oxidative phosphorylation* and *aerobic glycolysis* cells, representing a typical Warburg effect [151, 150] mixed population. We have glucose (glu), oxygen (O_2), and lactate (lac). *Oxidative phosphorylation* is governed by $1\mathrm{glu} + 6O_2 \rightarrow 4CO_2 + 4H_2O + 36\mathrm{ATP}$. Accordingly, *oxidative phosphorylation* cells consume glu and O_2 in a rate ratio of 1:6. *Aerobic glycolysis* is governed by $1\mathrm{glu} \rightarrow 2\mathrm{lac} + 2\mathrm{ATP}$. Accord-

ingly, *aerobic glycolysis* cells consume glu and release lac in a rate ratio of 1:2. Since we assume aerobic glycolysis proceeds 100 times faster than oxidative phosphorylation, then *aerobic glycolysis* and *aerobic glycolysis* cells consume glu in a rate ratio of 1:100. In this way we implement the basic stoichiometry of the two metabolic phenotypes, taken together. In terms of particles and fitness, glu and O_2 positively impact to a large extent, and lac negatively impacts to a negligible extent, the fitness of *oxidative phosphorylation* cells; and glu positively impacts to a large extent the fitness of *aerobic glycolysis* cells (but to a lesser extent than with *oxidative phosphorylation* cells). Both cell types are replaceable and reproductive. The specific quantities mentioned here are given in the tables below.

A.4.1.1 *Configuration parameters*

glu	O_2	lac
1	1	1

Table 35: Diffusion rate of each particle type.

glu	O_2	lac
0	0	0.01

Table 36: Initial concentration of each particle type.

cell type	glu	O_2	lac
vessel	0.1	0.1	0
empty	0.1	0.1	0
oxidative phosphorylation	0.1	0.1	0
aerobic glycolysis	0.1	0.1	0

Table 37: Basal lower-bound concentration of each particle type by cell type.

cell type	glu	O_2	lac
vessel	∞	∞	∞
empty	∞	∞	∞
oxidative phosphorylation	∞	∞	∞
aerobic glycolysis	∞	∞	∞

Table 38: Basal upper-bound concentration of each particle type by cell type.

cell type	glu	O_2	lac
vessel	0	0	0
empty	0	0	0
oxidative phosphorylation	0.01	0.06	0
aerobic glycolysis	1.0	0	0

Table 39: Consumption rate of each particle type by cell type.

cell type	glu	O_2	lac
vessel	0	0	0
empty	0	0	0
oxidative phosphorylation	0	0	0
aerobic glycolysis	0	0	2.0

Table 40: Release rate for of particle type by cell type.

cell type	glu	O_2	lac
vessel	0	0	0
empty	0	0	0
oxidative phosphorylation	3.0	3.0	-0.1
aerobic glycolysis	2.8	0	0

Table 41: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
oxidative phosphorylation	Yes
aerobic glycolysis	Yes

Table 42: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
oxidative phosphorylation	Yes
aerobic glycolysis	Yes

Table 43: Reproductive predicate of each cell type.

A.4.1.2 *Initial conditions*

The simulation opens with the initial concentrations of glu and O_2 set to zero, and lac set to a negligible amount—otherwise, if it too were set to zero, then given the nature of how our simulation computes particle concentrations after cellular consumption and release, it would remain zero throughout the simulation—as specified in Table 36, and lower-bounded basal concentrations of glu and O_2, as specified in

Table 37. These diffuse with rates specified in Table 35. We initialize the 2D lattice with a random mixture of *empty*, *oxidative phosphorylation*, and *aerobic glycolysis* cells.

A.4.2 *Results*

A.4.2.1 *Spatial cell population evolution*

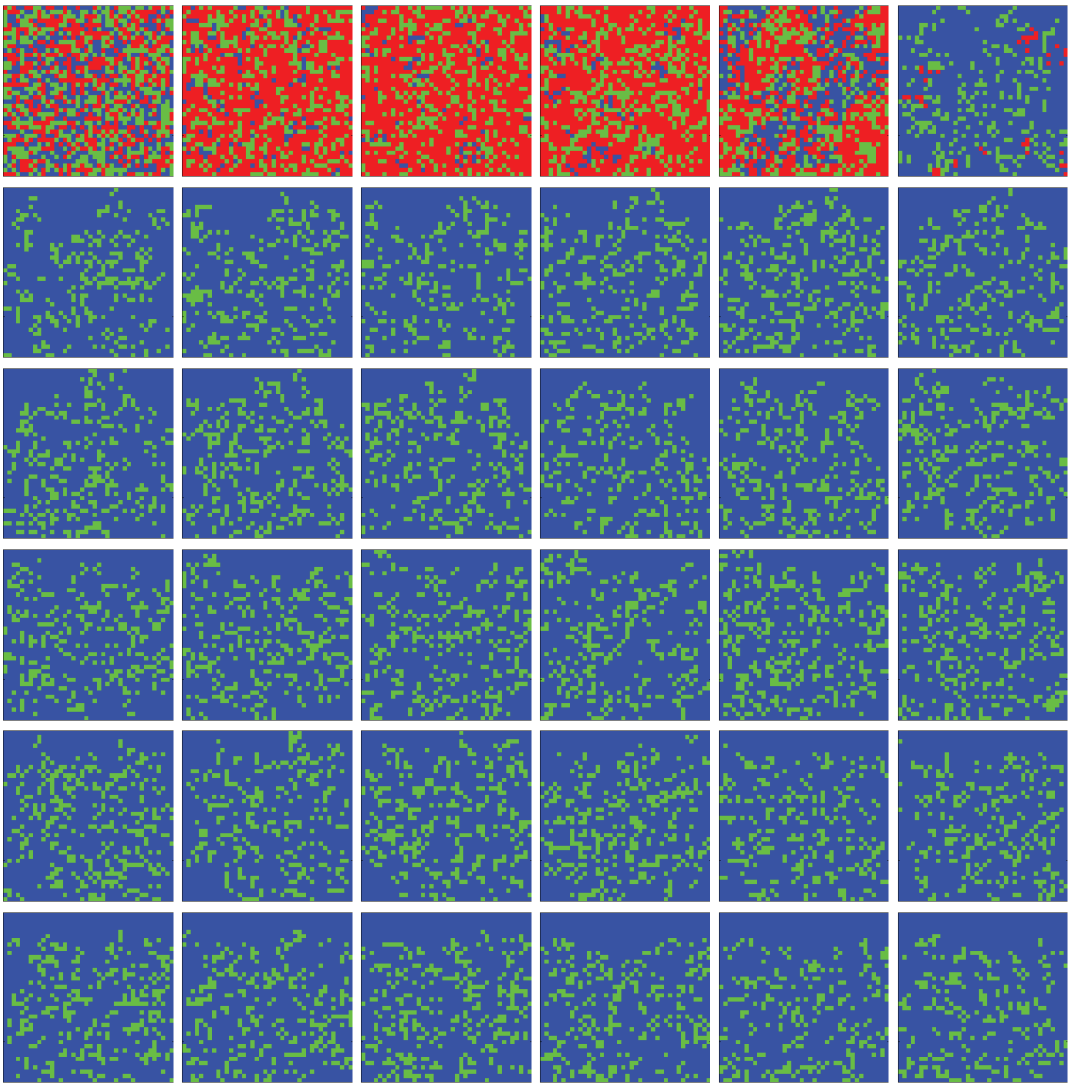

Figure 46: Time evolution of cell populations. Left-to-right, top-to-bottom: 100 generations shown in 36 frames (t = 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55, 58, 61, 64, 67, 70, 73, 75, 78, 80, 83, 85, 88, 90, 93, 95, 98, 100). Key: *empty*, *oxidative phosphorylation*, *aerobic glycolysis*.

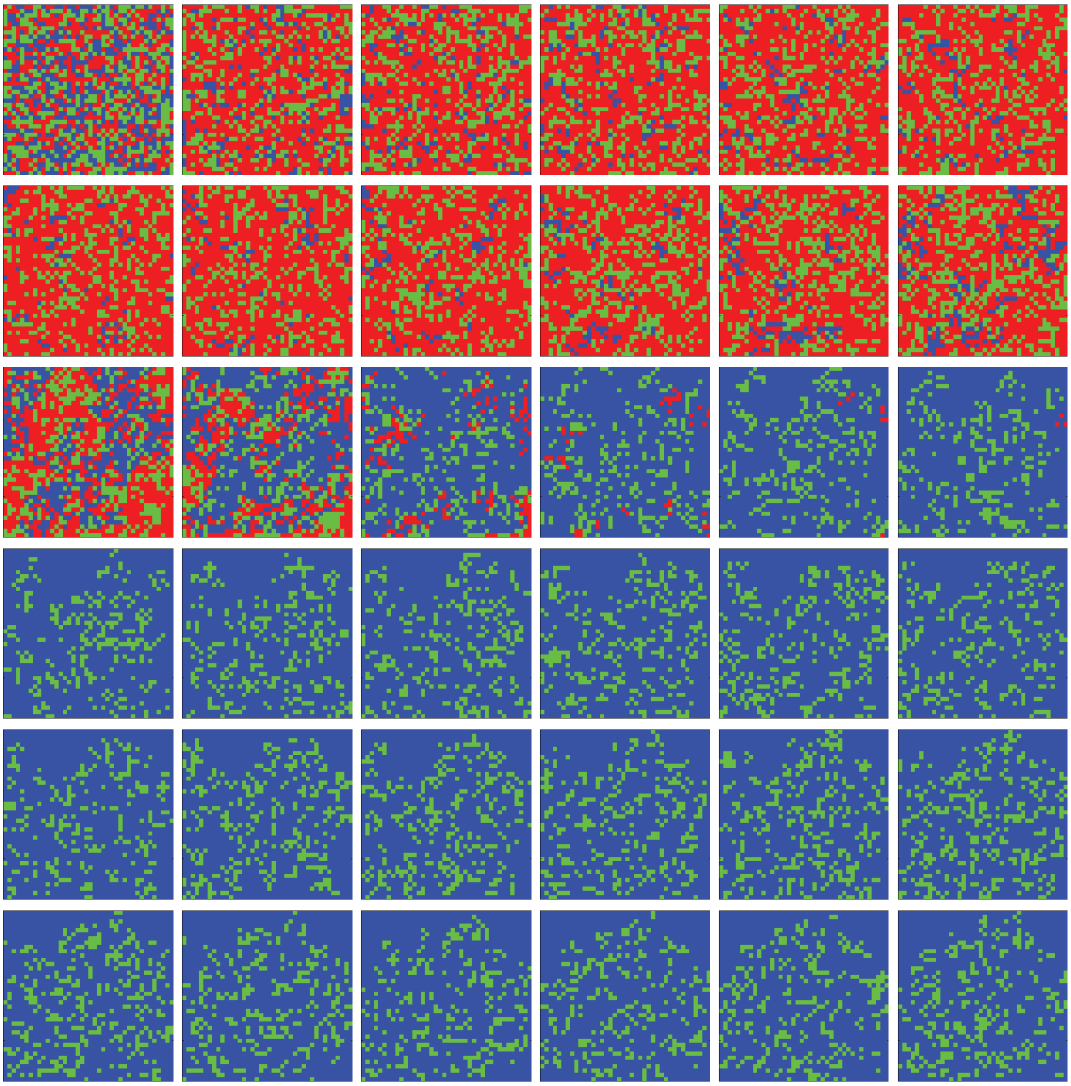

Figure 47: Time evolution of cell populations. Left-to-right, top-to-bottom: 36 generations shown in 36 frames (t = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36). Key: *empty*, *oxidative phosphorylation*, *aerobic glycolysis*.

A.4.2.2 *Temporal cell population evolution*

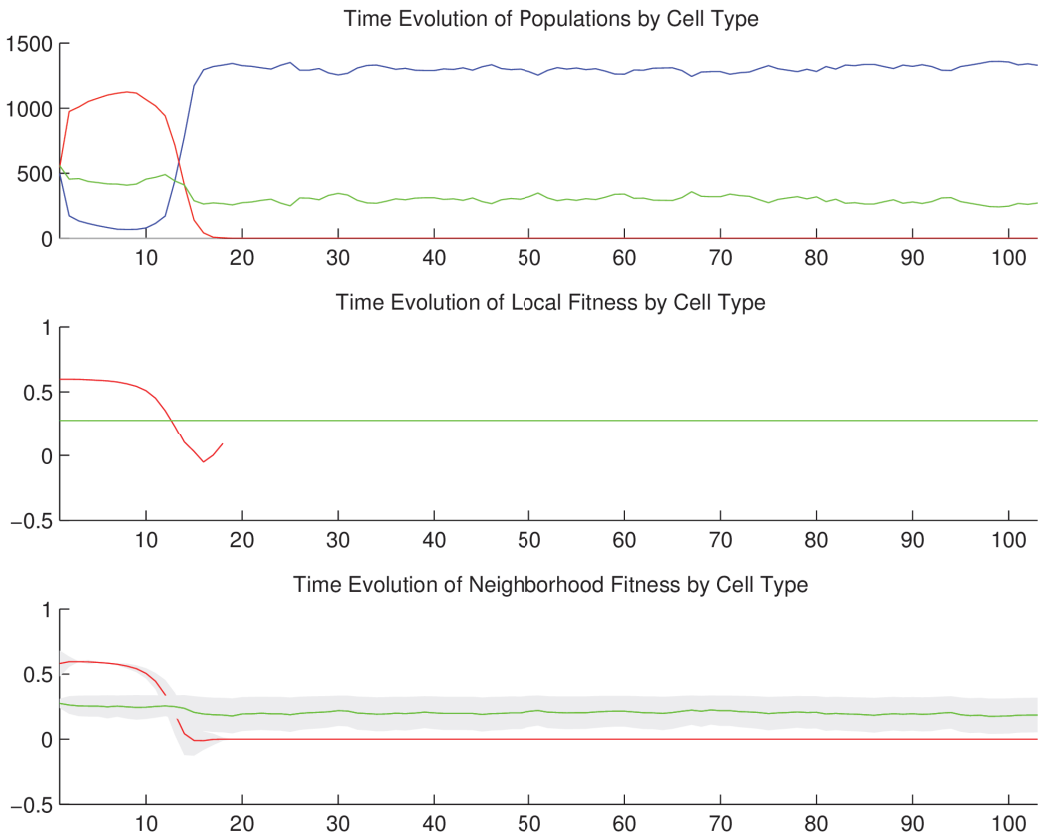

Figure 48: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *oxidative phosphorylation*, *aerobic glycolysis*.

A.4.3 *Discussion*

We observe in Figure 46 and Figure 47 that the *oxidative phosphorylation* and *aerobic glycolysis* cell populations immediately grow to cover most of the space, with *oxidative*

phosphorylation cells dominating initially due to higher mean fitness. However, the clusters of *aerobic glycolysis* cells which gain a foothold quickly release enough lac into their locales to eliminate their local *oxidative phosphorylation* competitors due to the negative impact of lac on *oxidative phosphorylation* fitness. As lac diffuses, this effect is increasingly widespread, until in short order all of the *oxidative phosphorylation* cells are eliminated in a situation that resembles acidosis. In Figure 48 we observe that from generation 15 onward, the mean size of the *aerobic glycolysis* cell population is constant. What interests us is that this mean population size is noticeably lower than it is from generations 1 to 15. Why would the *aerobic glycolysis* cell population size go down precisely when their *oxidative phosphorylation* competitors went extinct? Notice that at this point (generation 15), there is an increase in variance in the neighborhood fitness of the *aerobic glycolysis* cell population. Perhaps since so much of the area was covered by slower glu-consuming *oxidative phosphorylation* cells, and then these were rapidly eliminated, the *aerobic glycolysis* cells were no longer kept apart from each other, and would thereby raid each other's local sources of glu (depleting at each iteration the basal glu concentration), and so suffer a mean fitness decrease. There is something more to be said for the sheer bulk of cell clustering, even mixed type clustering, in that perhaps *aerobic glycolysis* cells competed better against *oxidative phosphorylation* cells (which surrounded them prior to *oxidative phosphorylation* type extinction) than *empty* cells (which surrounded them after this extinction).

A.5 OXIDATIVE PHOSPHORYLATION VS AEROBIC GLYCOLYSIS WITH ONE VESSEL (2D)

A.5.1 *Setup*

In this simulation, we have two cell types, *oxidative phosphorylation* and *aerobic glycolysis*, that correspond to cancer cells that use *oxidative phosphorylation* and *aerobic glycolysis*, respectively, representing a typical Warburg effect [151, 150] mixed population. Particle types 1, 2, and 3 correspond to glucose (glu), oxygen (O_2), and lactate (lac), respectively. *Oxidative phosphorylation* is governed by $1\mathrm{glu} + 6O_2 \rightarrow 4CO_2 + 4H_2O + 36\mathrm{ATP}$. Accordingly, *oxidative phosphorylation* cells consume glu and O_2 in a rate ratio of 1:6. *Aerobic glycolysis* is governed by $1\mathrm{glu} \rightarrow 2\mathrm{lac} + 2\mathrm{ATP}$. Accordingly, *aerobic glycolysis* cells consume glu and release lac in a rate ratio of 1:2. Since we assume *aerobic glycolysis* proceeds 100 times faster than *oxidative phosphorylation*, then *oxidative phosphorylation* and *aerobic glycolysis* consume glu in a rate ratio of 1:100. In this way we implement the basic stoichiometry of the two metabolic phenotypes, taken together. In terms of particles and fitness, glu and O_2 positively impact to a large extent, and lac negatively impacts to a negligible extent, the fitness of *oxidative phosphorylation* cells; and glu positively impacts to a large extent the fitness of *aerobic glycolysis* cells (but to a lesser extent than with *oxidative phosphorylation* cells). The one vessel that extends down from the top-middle consumes lac and releases glu and O_2. Both cell types are replaceable and reproductive. The specific quantities mentioned here are given in the tables below.

A.5.1.1 *Configuration parameters*

glu	O_2	lac
1	1	1

Table 44: Diffusion rate of each particle type.

glu	O_2	lac
0	0	0.01

Table 45: Initial concentration of each particle type.

cell type	glu	O_2	lac
vessel	0.1	0.1	0
empty	0.1	0.1	0
oxidative phosphorylation	0.1	0.1	0
aerobic glycolysis	0.1	0.1	0

Table 46: Basal lower-bound concentration of each particle type by cell type.

cell type	glu	O_2	lac
vessel	∞	∞	∞
empty	∞	∞	∞
oxidative phosphorylation	∞	∞	∞
aerobic glycolysis	∞	∞	∞

Table 47: Basal upper-bound concentration of each particle type by cell type.

cell type	glu	O_2	lac
vessel	0	0	1.0
empty	0	0	0
oxidative phosphorylation	0.01	0.06	0
aerobic glycolysis	1.0	0	0

Table 48: Consumption rate of each particle type by cell type.

cell type	glu	O_2	lac
vessel	1.0	1.0	0
empty	0	0	0
oxidative phosphorylation	0	0	0
aerobic glycolysis	0	0	2.0

Table 49: Release rate for of particle type by cell type.

cell type	glu	O_2	lac
vessel	0	0	0
empty	0	0	0
oxidative phosphorylation	3.0	3.0	-0.1
aerobic glycolysis	2.8	0	0

Table 50: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
oxidative phosphorylation	Yes
aerobic glycolysis	Yes

Table 51: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
oxidative phosphorylation	Yes
aerobic glycolysis	Yes

Table 52: Reproductive predicate of each cell type.

A.5.1.2 *Initial conditions*

The simulation opens with the initial concentrations of glu and O_2 set to zero, and lac set to a negligible amount—otherwise, if it too were set to zero, then given the nature of how our simulation computes particle concentrations after cellular consumption and release, it would remain zero throughout the simulation—as specified in Table 36, and lower-bounded basal concentrations of glu and O_2, as specified in Table 37. These diffuse with rates specified in Table 35. We initialize the 2D lattice with a random mixture of *empty*, *oxidative phosphorylation*, and *aerobic glycolysis* cells, and place one vessel extending down from the top-middle.

A.5.2 *Results*

A.5.2.1 *Spatial cell population evolution*

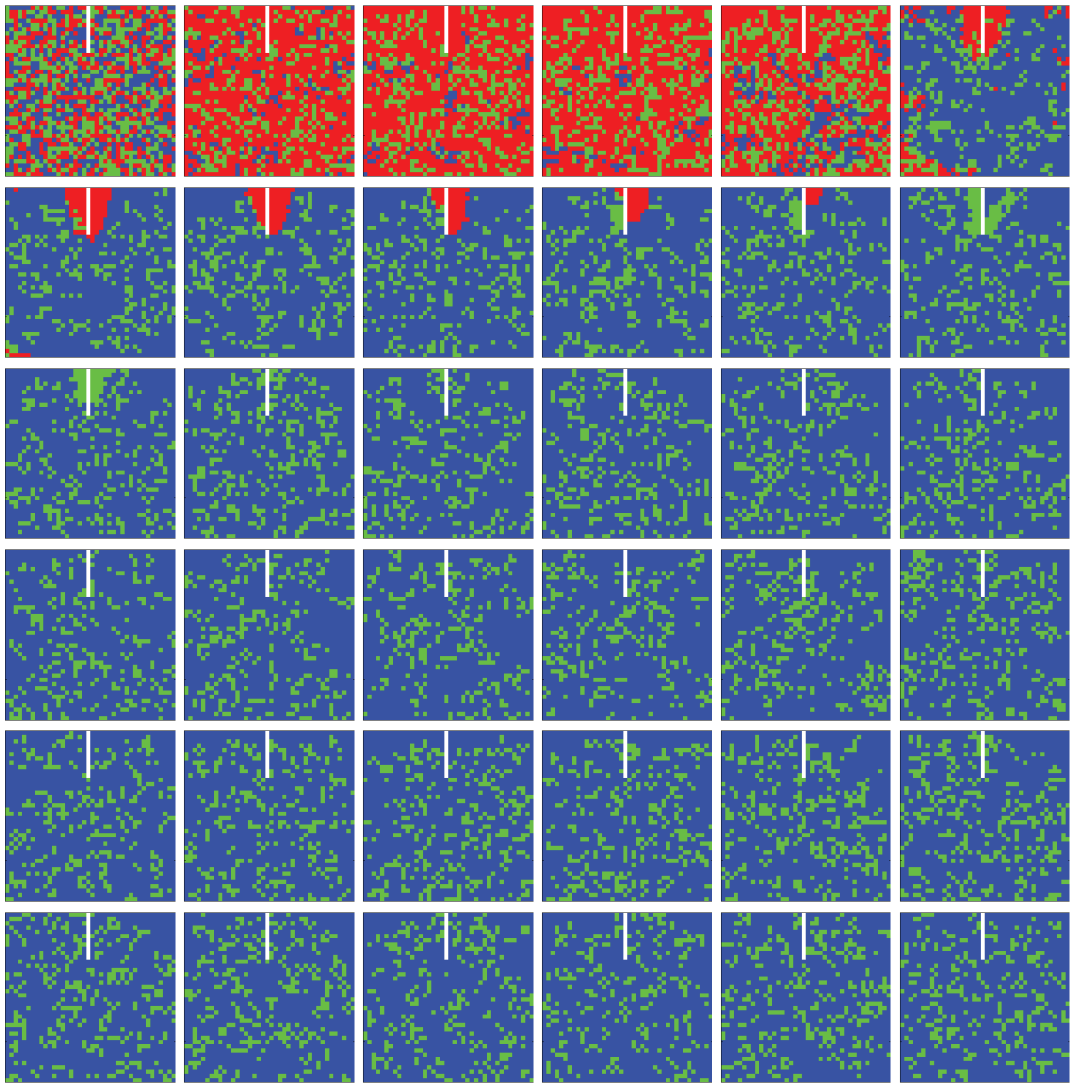

Figure 49: Time evolution of cell populations. Left-to-right, top-to-bottom: 100 generations shown in 36 frames (t = 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55, 58, 61, 64, 67, 70, 73, 75, 78, 80, 83, 85, 88, 90, 93, 95, 98, 100). Key: *vessel* (white), *empty*, *oxidative phosphorylation*, *aerobic glycolysis*.

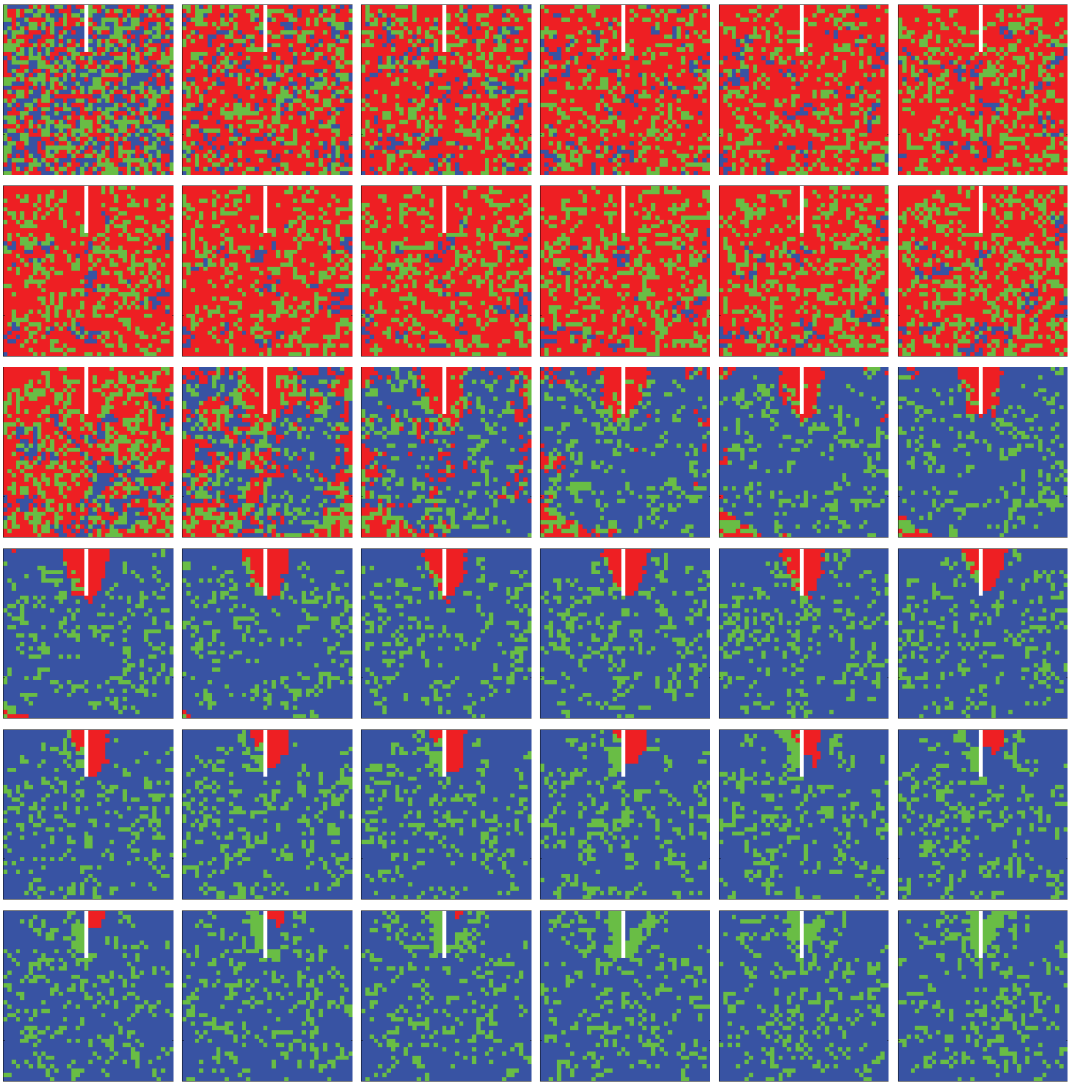

Figure 50: Time evolution of cell populations. Left-to-right, top-to-bottom: 36 generations shown in 36 frames (t = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36). Key: *vessel* (white), *empty*, *oxidative phosphorylation*, *aerobic glycolysis*.

A.5.2.2 *Temporal cell population evolution*

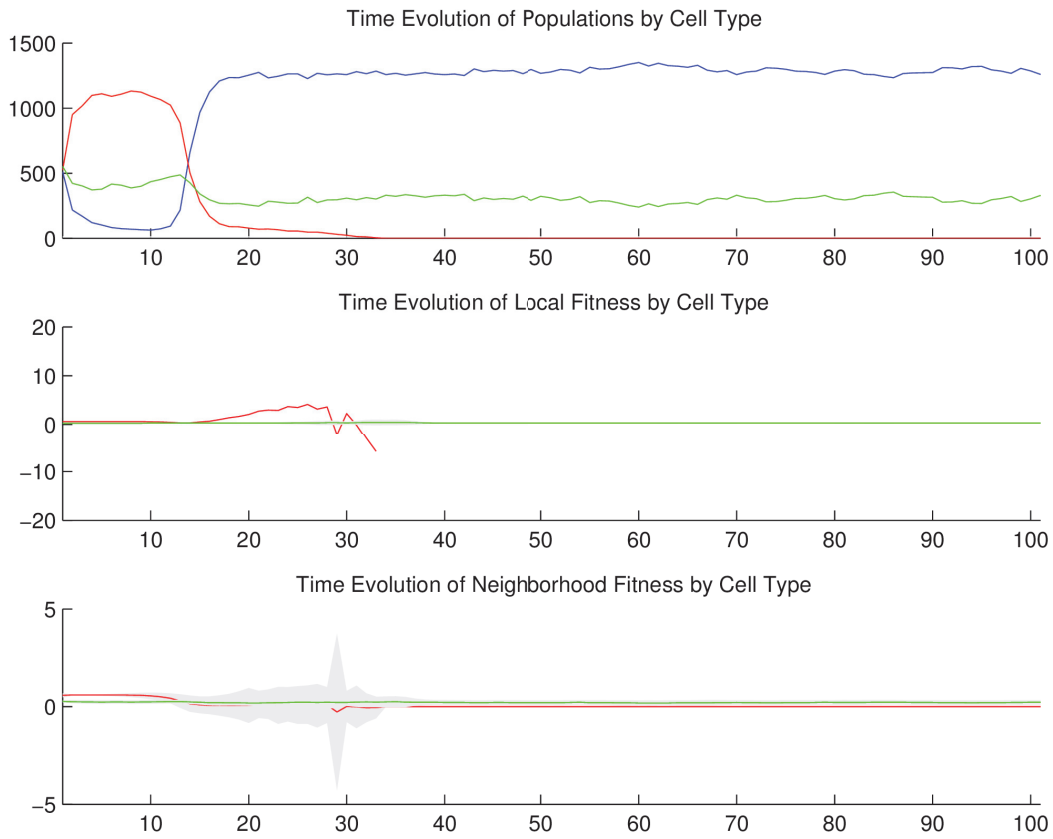

Figure 51: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *oxidative phosphorylation*, *aerobic glycolysis*.

A.5.3 *Discussion*

We observe in Figure 46 and Figure 47 that the *oxidative phosphorylation* and *aerobic glycolysis* cell populations immediately grow to cover most of the space, with *oxidative*

phosphorylation cells dominating initially due to higher mean fitness. However, the clusters of *aerobic glycolysis* cells which gain a foothold quickly release enough lac into their locales to eliminate their local *oxidative phosphorylation* competitors due to the negative impact of lac on *oxidative phosphorylation* fitness. As lac diffuses, this effect is increasingly widespread, until in short order all of the *oxidative phosphorylation* cells are eliminated in a situation that resembles acidosis. However, since the vessel consumes the local lac, we observe an expected clustering of *oxidative phosphorylation* cells directly around the vessel. These are temporarily "rescued" from the ensuing acidosis, but eventually succumb like the others. In effect, the vessel's removal of waste product only postpones the inevitable *oxidative phosphorylation* type extinction. In Figure 48 we observe that from generation 15 onward, the mean size of the *aerobic glycolysis* cell population is constant. What interests us is that this mean population size is noticeably lower than it is from generations 1 to 15. Why would the *aerobic glycolysis* cell population size go down precisely when their *oxidative phosphorylation* competitors went extinct? Notice that at this point (generation 15), there is an increase in variance in the neighborhood fitness of the *aerobic glycolysis* cell population. Perhaps since so much of the area was covered by slower glu-consuming *oxidative phosphorylation* cells, and then these were rapidly eliminated, the *aerobic glycolysis* cells were no longer kept apart from each other, and would thereby raid each other's local sources of glu (depleting at each iteration the basal glu concentration), and so suffer a mean fitness decrease. There is something more to be said for the sheer bulk of cell clustering, even mixed type clustering, in that perhaps *aerobic glycolysis* cells competed better against *oxidative phosphorylation* cells (which surrounded them prior to *oxidative phosphorylation* type extinction) than *empty* cells (which surrounded them after this extinction).

A.6 METABOLIC SYMBIOSIS (2D)

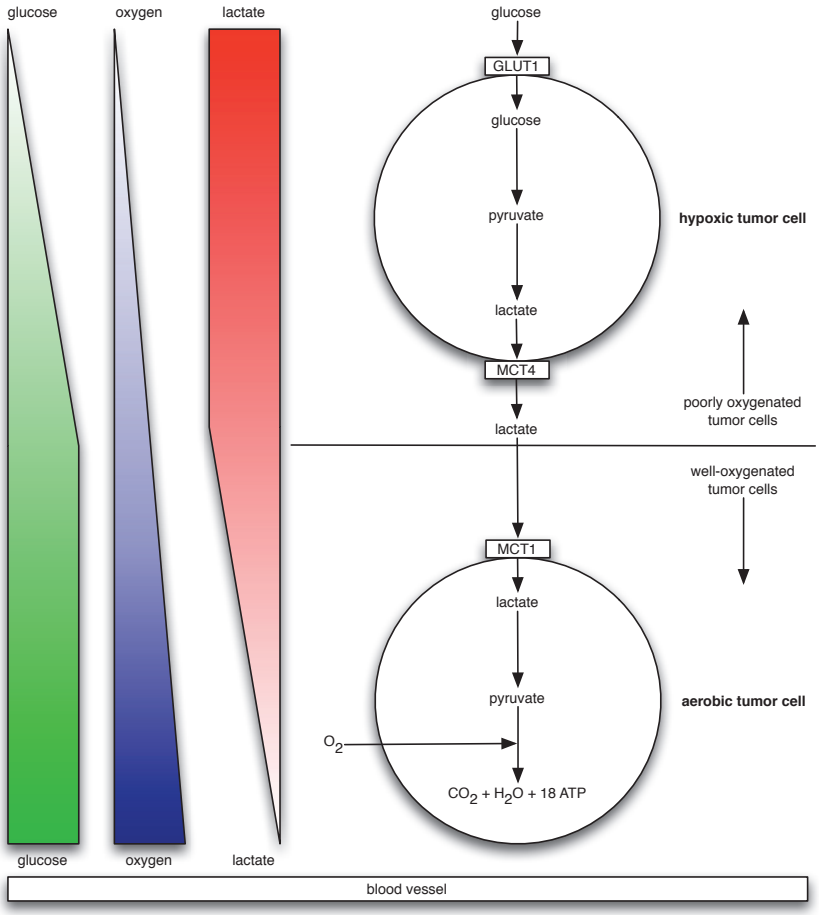

Figure 52: A schematic view of the "metabolic symbiosis" [128, 136] between *hypoxic* and *aerobic* tumor cells, where lactate produced by *hypoxic* cells is taken up by *aerobic* cells, which use it as their principal substrate for oxidative phosphorylation. The two cell types thereby mutually regulate their access to energy metabolites. Note the orientation of glucose, oxygen, and lactate gradients with respect to the blood vessel at the bottom.

A.6.1 *Setup*

In this simulation, we have *hypoxic* and *aerobic* tumor cells in the "metabolic symbiosis" scenario illustrated in Figure 52. We have glucose (glu), oxygen (O_2), and lactate (lac) particles. *Hypoxic* cells consume glu at a certain rate, and release lac at the same rate. *Aerobic* cells consume O_2 and lac at the same rate, and release no particles. In terms of particles and fitness, glu positively impacts the fitness of *hypoxic* cells; and O_2 and lac positively impact the fitness of *aerobic* cells. All rates and impacts are the same quantities for the two cell types. The one vessel that extends along the bottom row consumes lac and releases glu and O_2. Both cell types are replaceable and reproductive. The specific quantities mentioned here are given in the tables below.

A.6.1.1 *Configuration parameters*

glu	O_2	lac
1	1	1

Table 53: Diffusion rate of each particle type.

glu	O_2	lac
0	0	0.01

Table 54: Initial concentration of each particle type.

cell type	glu	O_2	lac
vessel	0.1	0.1	0
empty	0.1	0.1	0
hypoxic	0.1	0.1	0
aerobic	0.1	0.1	0

Table 55: Basal lower-bound concentration of each particle type by cell type.

cell type	glu	O_2	lac
vessel	∞	∞	∞
empty	∞	∞	∞
hypoxic	∞	∞	∞
aerobic	∞	∞	∞

Table 56: Basal upper-bound concentration of each particle type by cell type.

cell type	glu	O_2	lac
vessel	0	0	10.0
empty	0	0	0
hypoxic	1.0	0	0
aerobic	0	1.0	1.0

Table 57: Consumption rate of each particle type by cell type.

cell type	glu	O_2	lac
vessel	10.0	10.0	0
empty	0	0	0
hypoxic	0	0	1.0
aerobic	0	0	0

Table 58: Release rate for of particle type by cell type.

cell type	glu	O_2	lac
vessel	0	0	0
empty	0	0	0
hypoxic	1	0	0
aerobic	0	1	1

Table 59: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
hypoxic	Yes
aerobic	Yes

Table 60: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
hypoxic	Yes
aerobic	Yes

Table 61: Reproductive predicate of each cell type.

A.6.1.2 *Initial conditions*

The simulation opens with the initial concentrations of glu and O_2 set to zero, and lac set to a negligible amount—otherwise, if it too were set to zero, then given the nature of how our simulation computes particle concentrations after cellular consumption and release, it would remain zero throughout the simulation—as specified in Table 54, and lower-bounded basal concentrations of glu and O_2, as specified in

Table 55. These diffuse with rates specified in Table 53. We initialize the 2D lattice with the top half *hypoxic* cells, the bottom half *aerobic* cells, and place one vessel that extends along the bottom row.

A.6.2 *Results*

A.6.2.1 *Spatial cell population evolution*

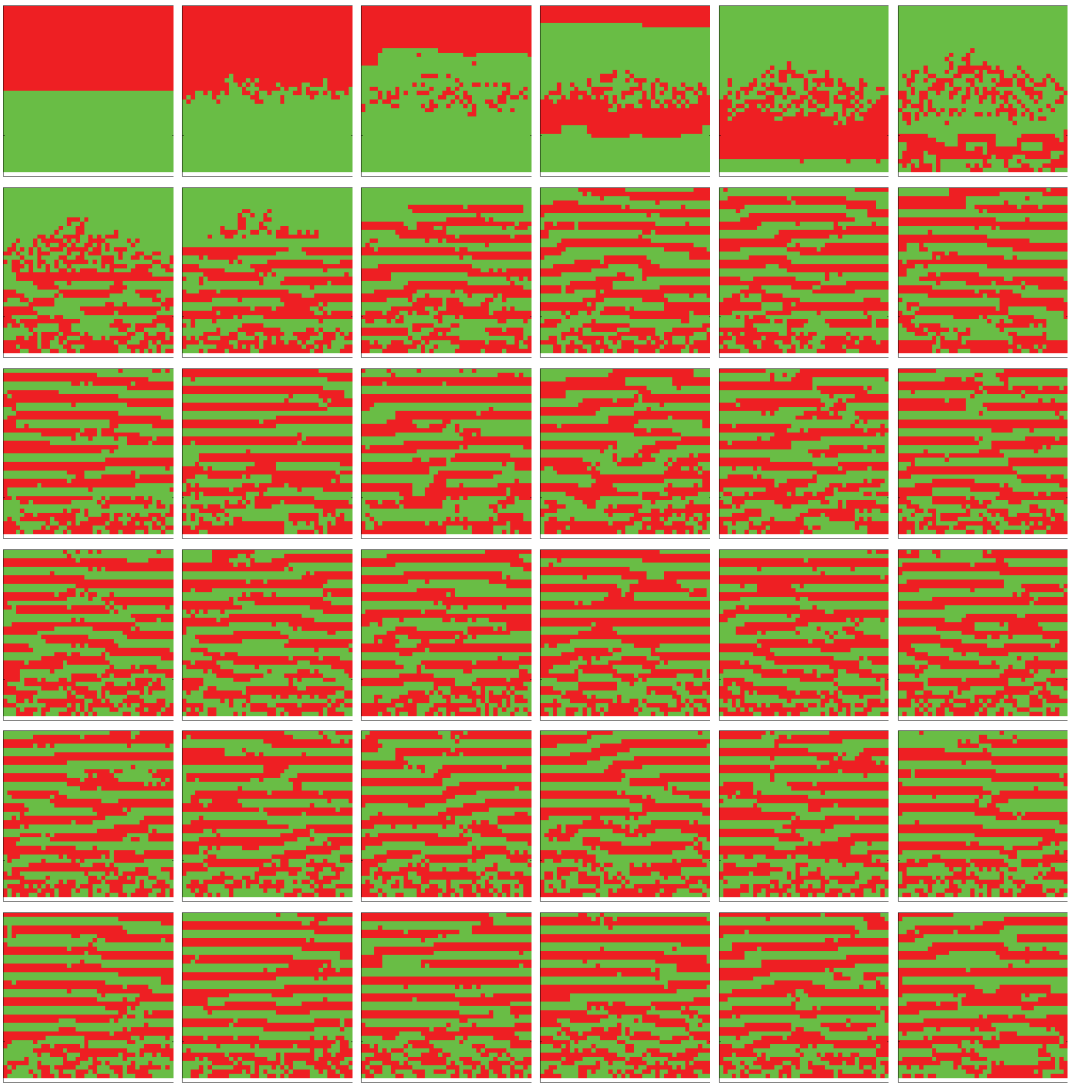

Figure 53: Time evolution of cell populations. Left-to-right, top-to-bottom: 210 generations shown in 36 frames (t = 1, 7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 91, 97, 103, 109, 115, 121, 127, 133, 139, 145, 151, 157, 163, 169, 175, 181, 187, 193, 199, 205, 210). Key: vessel (white), *empty*, *hypoxic*, *aerobic* cells.

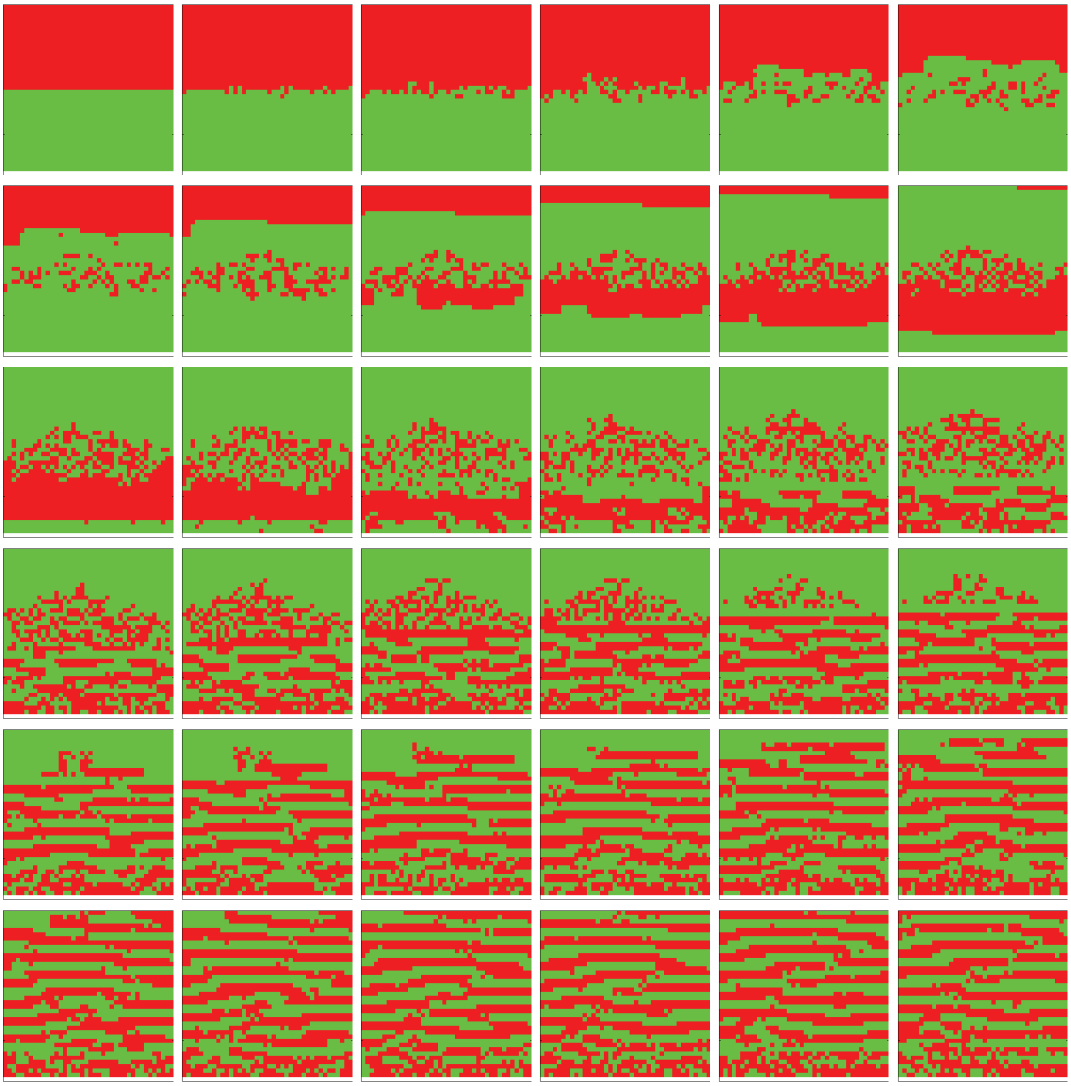

Figure 54: Time evolution of cell populations. Left-to-right, top-to-bottom: 60 generations shown in 36 frames (t = 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 30, 32, 33, 35, 36, 38, 39, 41, 42, 44, 45, 47, 48, 50, 51, 53, 54, 56, 57, 59, 60). Key: vessel (white), *empty*, *hypoxic*, *aerobic* cells.

A.6.2.2 *Temporal cell population evolution*

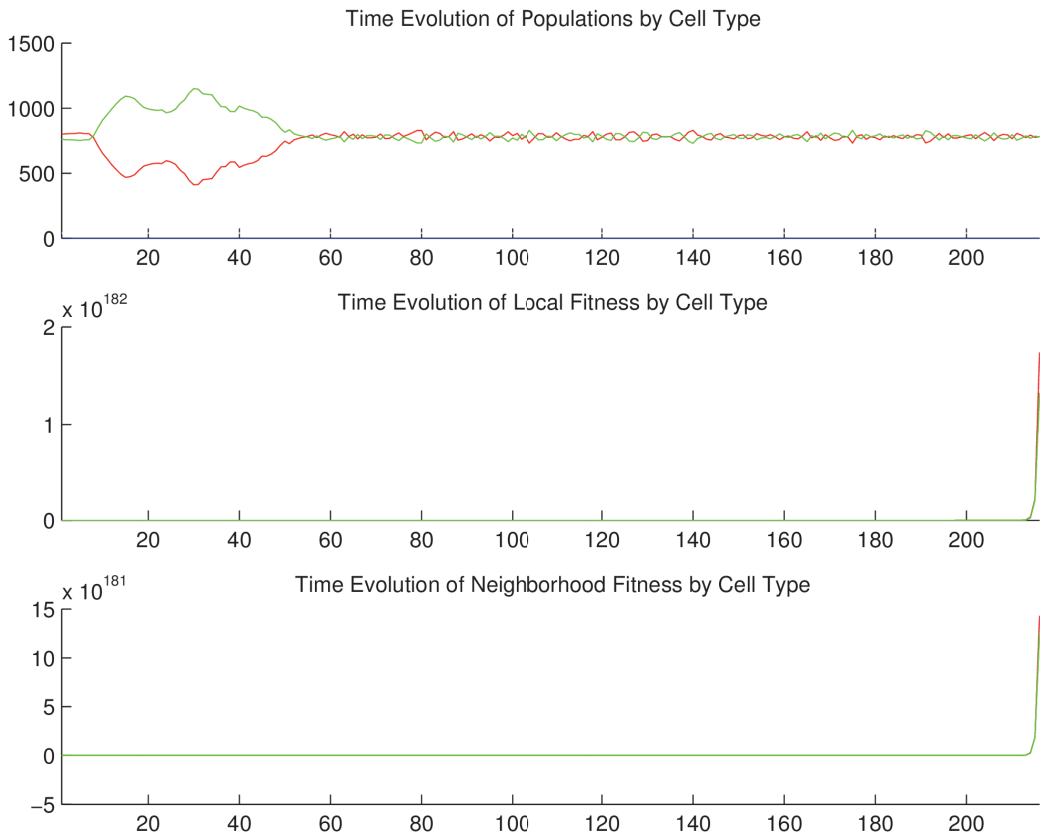

Figure 55: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *hypoxic*, *aerobic* cells.

A.6.3 *Discussion*

We observe in Figure 53 and Figure 54 that the *hypoxic* and *aerobic* cell populations immediately begin to mix at their horizontal interface. Then a thick horizontal layer of

aerobic cells pulls away and migrates upward; transiently, it appears that the populations of *hypoxic* and *aerobic* cells have swapped positions. Then a thick horizontal layer of *aerobic* cells pulls away and migrates downward. This inversion and pull-away migration repeats at successively smaller length scales, all the while maintaining its near perfect horizontality, until the entire area is covered in a regular striation pattern: horizontal, equal thickness layers of *aerobic* and *hypoxic* cells alternate from top to bottom. This pattern is stable for the duration of the simulation, and perhaps indefinitely. In Figure 55 we observe that since the two cell populations always fill the entire area, their population dynamics is always zero-sum in character. From generation 10 to 50, *aerobic* cells dominate; but from generation 50 onward, the two populations converge at, and oscillate about, the same mean size. We expect neither the reaction-diffusion [143] type emergent striation pattern nor the population size rebalancing.

A.7 TUMOR-STROMA SIGNALING (2D)

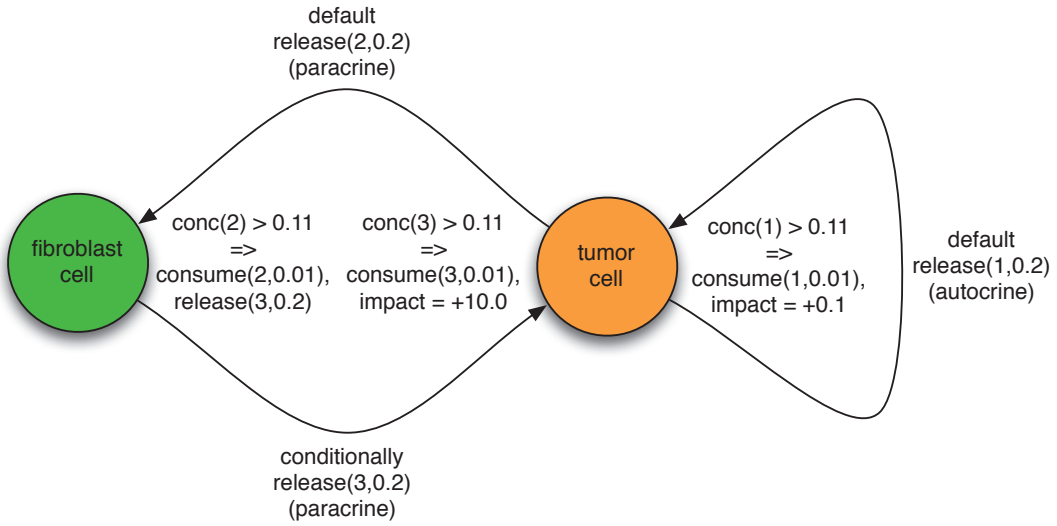

Figure 56: A schematic view of the tumor-stroma signaling described below, and quantified in Table 67 and Table 71.

A.7.1 *Setup*

In this simulation, we have *epithelial*, *fibroblast*, *tumor*, and *inert* cells. Three particle types play roles in the following scenario. In a middle of a row of *epithelial* cells, one of them transforms into a *tumor* cell. The *epithelial* cells are separated from the *fibroblast* cells as these are embedded in a thick layer of (*inert*) extracellular matrix. Proliferating (*tumor*) cells (autocrine) signal themselves with particle type 1. When this reaches a sufficient concentration, affected *tumor* cells begin consuming it and their fitness increases. Simultaneously, *tumor* cells (paracrine) signal nearby *fibroblast* cells residing in the (*inert*) extracellular matrix with particle type 2. When this reaches a sufficient concentration, affected *fibroblast* cells begin consuming it and immediately releasing

particle type 3. When this reaches a sufficient concentration, affected *tumor* cells begin consuming it and their fitness increases by a substantial factor, above and beyond that from its autocrine signaling. In this way, our conditional logic implements a simple signaling and adaptation system between *tumor* and *fibroblast* cells, illustrated schematically in Figure 56. By default, no cells consume particles, and only *tumor* tumor cells release particle types 1 and 2 at the same rate, to kick off autocrine and paracrine signaling, respectively. By default, no impact factors are defined; they are only conditionally applied. Only *empty* cells are replaceable, and only *tumor* cells are reproductive; the others are effectively inert. The specific quantities mentioned here are given in the tables below.

A.7.1.1 *Configuration parameters*

pt 1	*pt 2*	*pt 3*
10	0.1	0.1

Table 62: Diffusion rate of each particle type.

pt 1	*pt 2*	*pt 3*
0.1	0.1	0.1

Table 63: Initial concentration of each particle type.

cell type	*pt 1*	*pt 2*	*pt 3*
vessel	0	0	0
empty	0	0	0
epithelial	0	0	0
fibroblast	0	0	0
tumor	0	0	0
inert	0	0	0

Table 64: Basal lower-bound concentration of each particle type by cell type.

cell type	*pt 1*	*pt 2*	*pt 3*
vessel	∞	∞	∞
empty	∞	∞	∞
epithelial	∞	∞	∞
fibroblast	∞	∞	∞
tumor	∞	∞	∞
inert	∞	∞	∞

Table 65: Basal upper-bound concentration of each particle type by cell type.

cell type	*pt 1*	*pt 2*	*pt 3*
vessel	0	0	0
empty	0	0	0
epithelial	0	0	0
fibroblast	0	0	0
tumor	0	0	0
inert	0	0	0

Table 66: Consumption rate of each particle type by cell type.

cell type	*pt 1*	*pt 2*	*pt 3*
vessel	0	0	0
empty	0	0	0
epithelial	0	0	0
fibroblast	0	0	0
tumor	0.2	0.2	0
inert	0	0	0

Table 67: Release rate for of particle type by cell type.

cell type	*pt 1*	*pt 2*	*pt 3*
vessel	0	0	0
empty	0	0	0
epithelial	0	0	0
fibroblast	0	0	0
tumor	0	0	0
inert	0	0	0

Table 68: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
epithelial	No
fibroblast	No
tumor	No
inert	No

Table 69: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
epithelial	No
fibroblast	No
tumor	Yes
inert	No

Table 70: Reproductive predicate of each cell type.

cell type	*triggers*	*actions*
fibroblast	$\rho_2 > 0.11$	$c_{\textit{fibroblast},2} \leftarrow 0.01$, $r_{\textit{fibroblast},3} \leftarrow 0.2$
tumor	$\rho_1 > 0.11$	$c_{\textit{tumor},1} \leftarrow 0.01$, $\sigma_{\textit{tumor},1} \leftarrow 0.1$
tumor	$\rho_3 > 0.11$	$c_{\textit{tumor},3} \leftarrow 0.01$, $\sigma_{\textit{tumor},3} \leftarrow 10$

Table 71: Conditional triggers and actions for those cell types so configured.

A.7.1.2 *Initial conditions*

The simulation opens with the initial concentrations of particle types 1, 2, and 3 set to the same value, as specified in Table 63, with no lower- or upper-bounded basal concentrations set. These diffuse with rates specified in Table 62. We initialize the 2D lattice with a horizontal monolayer of *epithelial* cells, located about two-thirds down from the top, in the middle of which we place a single *tumor* cell. Below this, we lay down a thick layer of sub-*epithelial* (*inert*) extracellular matrix. Within this layer, we place a random scattering of *fibroblast* cells such that their density increases nonlinearly in a rightward direction—this amounts in most of the *fibroblast* cells populating the rightmost third of the area.

A.7.2 *Results*

A.7.2.1 *Spatial cell population evolution*

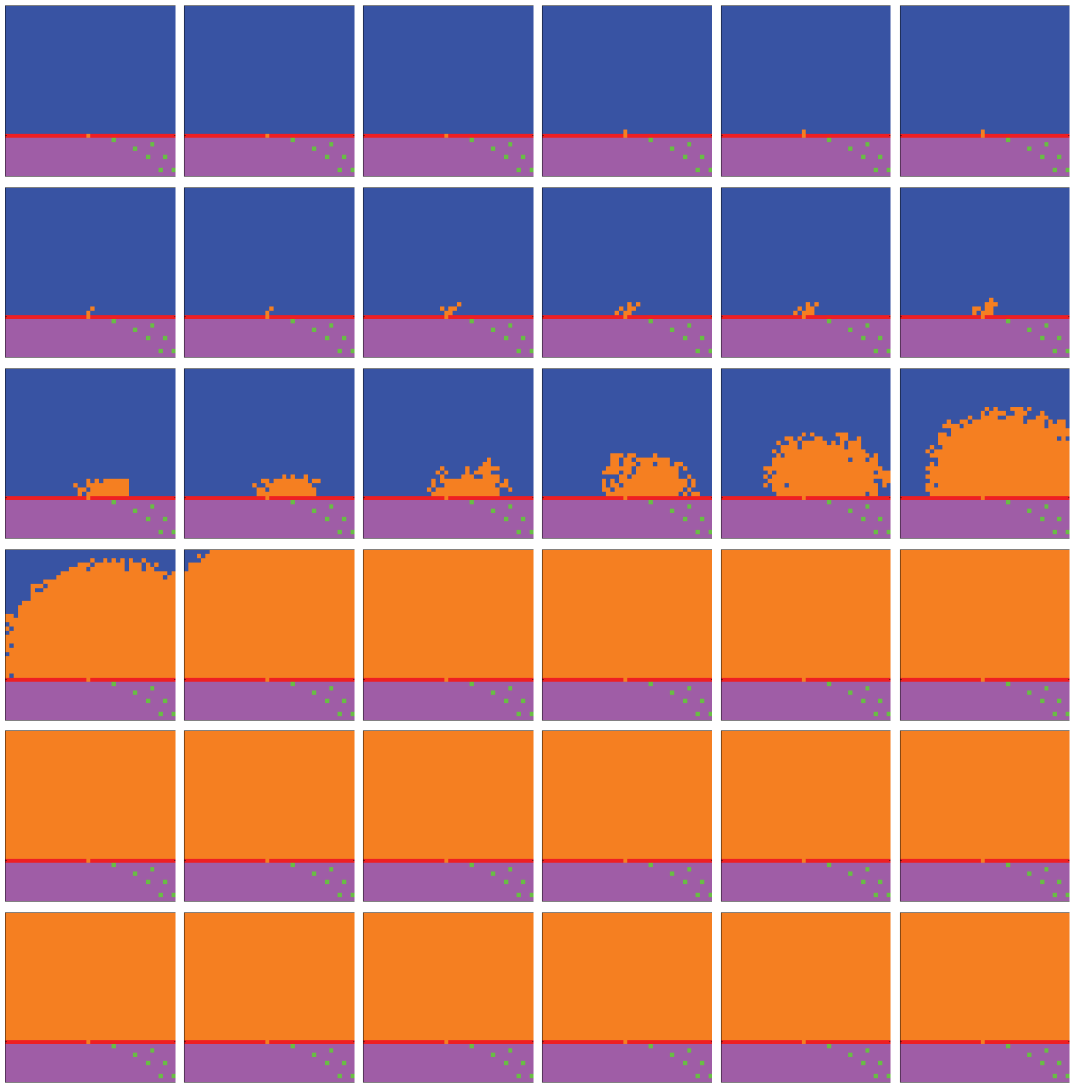

Figure 57: Time evolution of cell populations. Left-to-right, top-to-bottom: 440 generations shown in 36 frames (t = 1, 14, 27, 40, 53, 65, 78, 90, 103, 115, 128, 140, 153, 165, 178, 190, 203, 215, 228, 240, 253, 265, 278, 290, 303, 315, 328, 340, 353, 365, 378, 390, 403, 415, 428, 440). Key: *vessel* (white), *empty*, *epithelial*, *fibroblast*, *tumor*, *inert*.

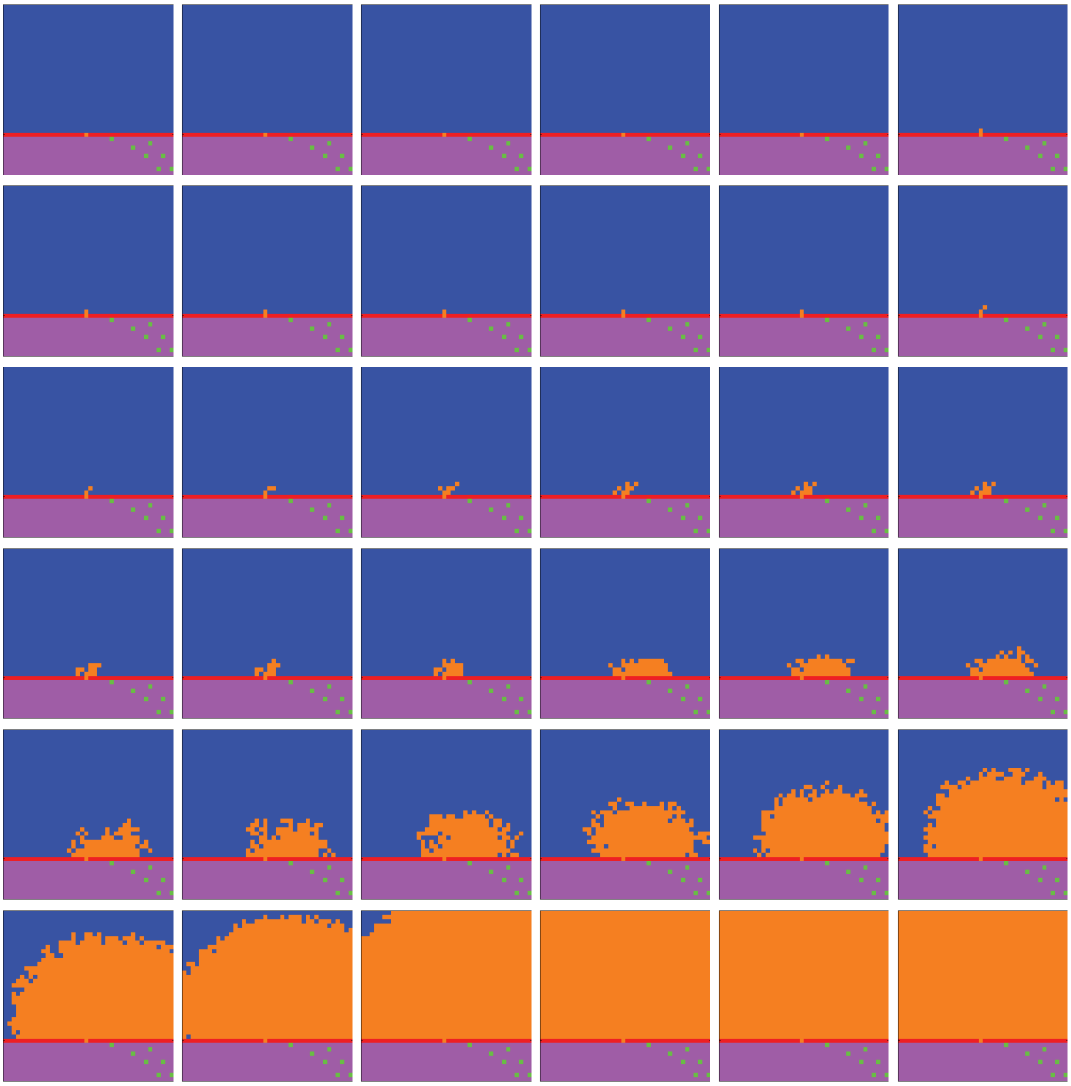

Figure 58: Time evolution of cell populations. Left-to-right, top-to-bottom: 260 generations shown in 36 frames (t = 1, 8, 15, 22, 29, 36, 43, 50, 58, 65, 73, 80, 88, 95, 103, 110, 118, 125, 133, 140, 148, 155, 163, 170, 178, 185, 193, 200, 208, 215, 223, 230, 238, 245, 253, 260). Key: *vessel* (white), *empty*, *epithelial*, *fibroblast*, *tumor*, *inert*.

A.7.2.2 *Temporal cell population evolution*

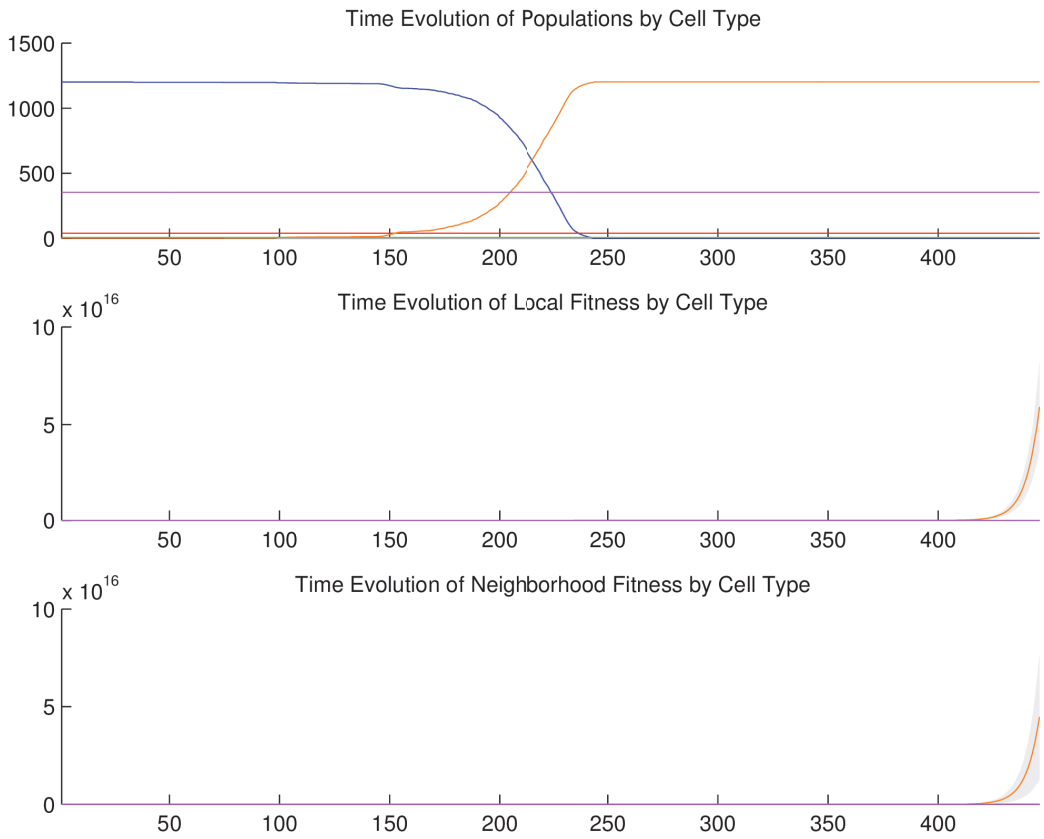

Figure 59: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *epithelial*, *fibroblast*, *tumor*, *inert*.

A.7.3 *Discussion*

We observe in Figure 57 and Figure 58 that *tumor* cell proliferation is very slow at first, as the autocrine signaling causes only a small increase in affected *tumor* cell fitness. It

is not until much later, when *fibroblast* paracrine signaling reaches the *tumor* cells that their proliferation becomes noticeable, due to the consequent much higher fitness impact on the affected *tumor* cells, and they eventually take over the replaceable area. Notice that this faster growth moves in a decidedly rightward direction, toward the source of the signaling gradient, the rightwardly dense *fibroblast* cells. After a fashion, radial outward growth dominates and the *tumor* contour becomes circular. In Figure 59 we observe linear growth in the *tumor* population size from generation 145 to 155, then exponential growth from generation 155 to 250, where the population size saturates at the full replaceable area.

A.8 APOPTOTIC CORE (2D)

A.8.1 *Setup*

In this simulation, we have non-replaceable and reproductive *tumor* cells. The *tumor* consume O_2 at a certain rate, and release no particles. The conditional logic implements *tumor* cell apoptosis wherever O_2 falls below a threshold. The specific quantities mentioned here are given in the tables below.

A.8.1.1 *Configuration parameters*

O_2
0.01

Table 72: Diffusion rate of each particle type.

O_2
0.1

Table 73: Initial concentration of each particle type.

cell type	O_2
vessel	0
empty	0
tumor	0

Table 74: Basal lower-bound concentration of each particle type by cell type.

cell type	O_2
vessel	∞
empty	∞
tumor	∞

Table 75: Basal upper-bound concentration of each particle type by cell type.

cell type	O_2
vessel	0
empty	0
tumor	0.01

Table 76: Consumption rate of each particle type by cell type.

cell type	O_2
vessel	0
empty	0
tumor	0

Table 77: Release rate for of particle type by cell type.

cell type	O_2
vessel	0
empty	0
tumor	1

Table 78: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
tumor	No

Table 79: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
tumor	Yes

Table 80: Reproductive predicate of each cell type.

cell type	*triggers*	*actions*
tumor	$\rho_1 < 0.05$	apoptosis

Table 81: Conditional triggers and actions for those cell types so configured.

A.8.1.2 *Initial conditions*

The simulation opens with the initial concentration of O_2 specified in Table 73. This diffuses with a rate specified in Table 72. We initialize the 2D lattice with *empty* cells, and we place one *tumor* cell in the center.

A.8.2 *Results*

A.8.2.1 *Spatial cell population evolution*

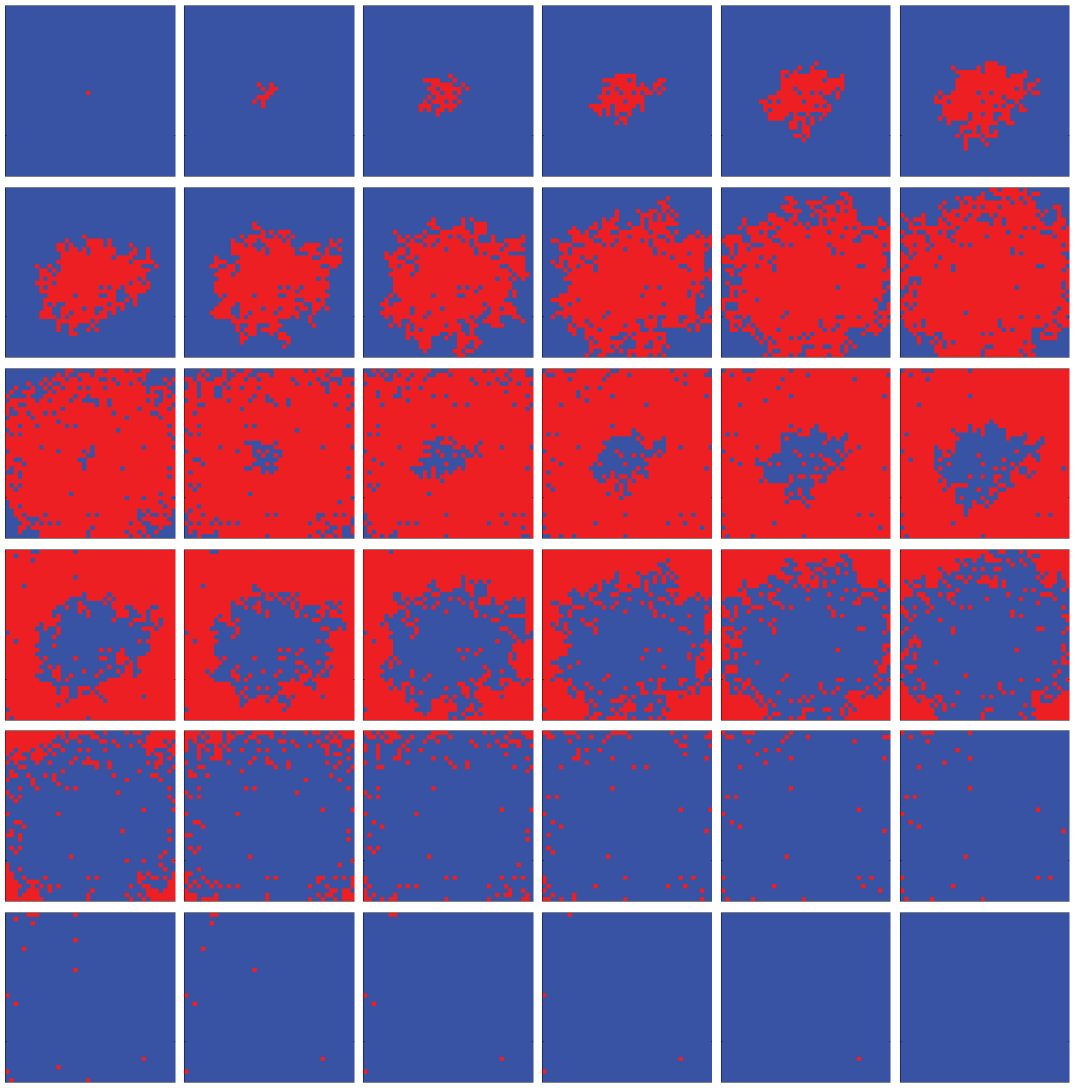

Figure 60: Time evolution of cell populations. Left-to-right, top-to-bottom: 200 generations shown in 36 frames (t = 1, 7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 90, 96, 101, 107, 112, 118, 123, 129, 134, 140, 145, 151, 156, 162, 167, 173, 178, 184, 189, 195, 200). Key: *empty*, *tumor*.

A.8.2.2 *Temporal cell population evolution*

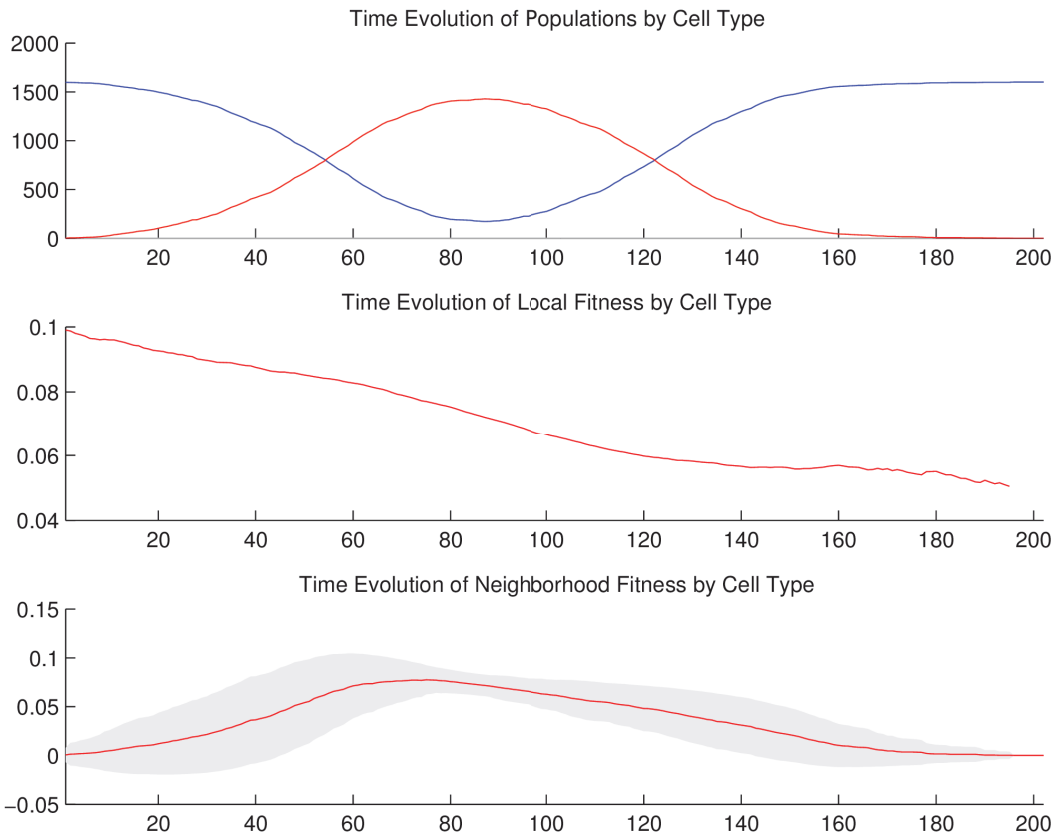

Figure 61: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *tumor*.

A.8.3 *Discussion*

We observe the following sequence of events in Figure 60. *Tumor* cells proliferate radially outward from the center. Once most of the area is covered with *tumor* cells,

then *empty* cells appear in the center—where *tumor* cells have undergone apoptosis. *Empty* cells grow radially outward at a similar rate to the *tumor* cells just beyond them. We observe the succession of *tumor* → *empty* cell populations in Figure 61. The decay of *tumor* cell populations is due to their proliferating outside of the spatial dimensions of the simulation while simultaneously being replaced by the succeeding population. By generation 200, *empty* cells have taken over the area.

A.9 NECROTIC CORE (2D)

A.9.1 *Setup*

In this simulation, we have *viable*, *hypoxic*, and *necrotic* tumor cells that correspond to those we see in the anti-pimonidazole stain images. The *viable* and *hypoxic* cells consume O_2 at a certain rate, and release no particles. The conditional logic implements a simple state machine in the following way. Wherever O_2 falls below a threshold, *viable* cells become ("jump" to) *hypoxic* cells—identical in every way except as follows. Wherever O_2 rises above that threshold, *hypoxic* cells become *viable* again; and wherever O_2 falls below an even lower threshold, *hypoxic* cells become *necrotic* cells. Once a cell becomes *necrotic* it has entered an absorbing state and its behavior is completely inert: it has no consumption or release profile, and it is neither replaceable nor reproductive. The specific quantities mentioned here are given in the tables below.

A.9.1.1 *Configuration parameters*

O_2
0.1

Table 82: Diffusion rate of each particle type.

O_2
0.1

Table 83: Initial concentration of each particle type.

cell type	O_2
vessel	0
empty	0
viable	0
hypoxic	0
necrotic	0

Table 84: Basal lower-bound concentration of each particle type by cell type.

cell type	O_2
vessel	∞
empty	∞
viable	∞
hypoxic	∞
necrotic	∞

Table 85: Basal upper-bound concentration of each particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	0.01
hypoxic	0.01
necrotic	0

Table 86: Consumption rate of each particle type by cell type.

cell type	O_2
vessel	0.2
empty	0
viable	0
hypoxic	0
necrotic	0

Table 87: Release rate for of particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	1
hypoxic	1
necrotic	0

Table 88: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
viable	No
hypoxic	No
necrotic	No

Table 89: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
viable	Yes
hypoxic	Yes
necrotic	No

Table 90: Reproductive predicate of each cell type.

cell type	*triggers*	*actions*
viable	$\rho_1 < 0.07$	jump to *hypoxic*
hypoxic	$\rho_1 < 0.05$	jump to *necrotic*
hypoxic	$\rho_1 > 0.07$	jump to *viable*

Table 91: Conditional triggers and actions for those cell types so configured.

A.9.1.2 *Initial conditions*

The simulation opens with the initial concentration of O_2 specified in Table 83. This diffuses with a rate specified in Table 82. We initialize the 2D lattice with *empty* cells, and we place one *viable* cell in the center.

A.9.2 *Results*

A.9.2.1 *Spatial cell population evolution*

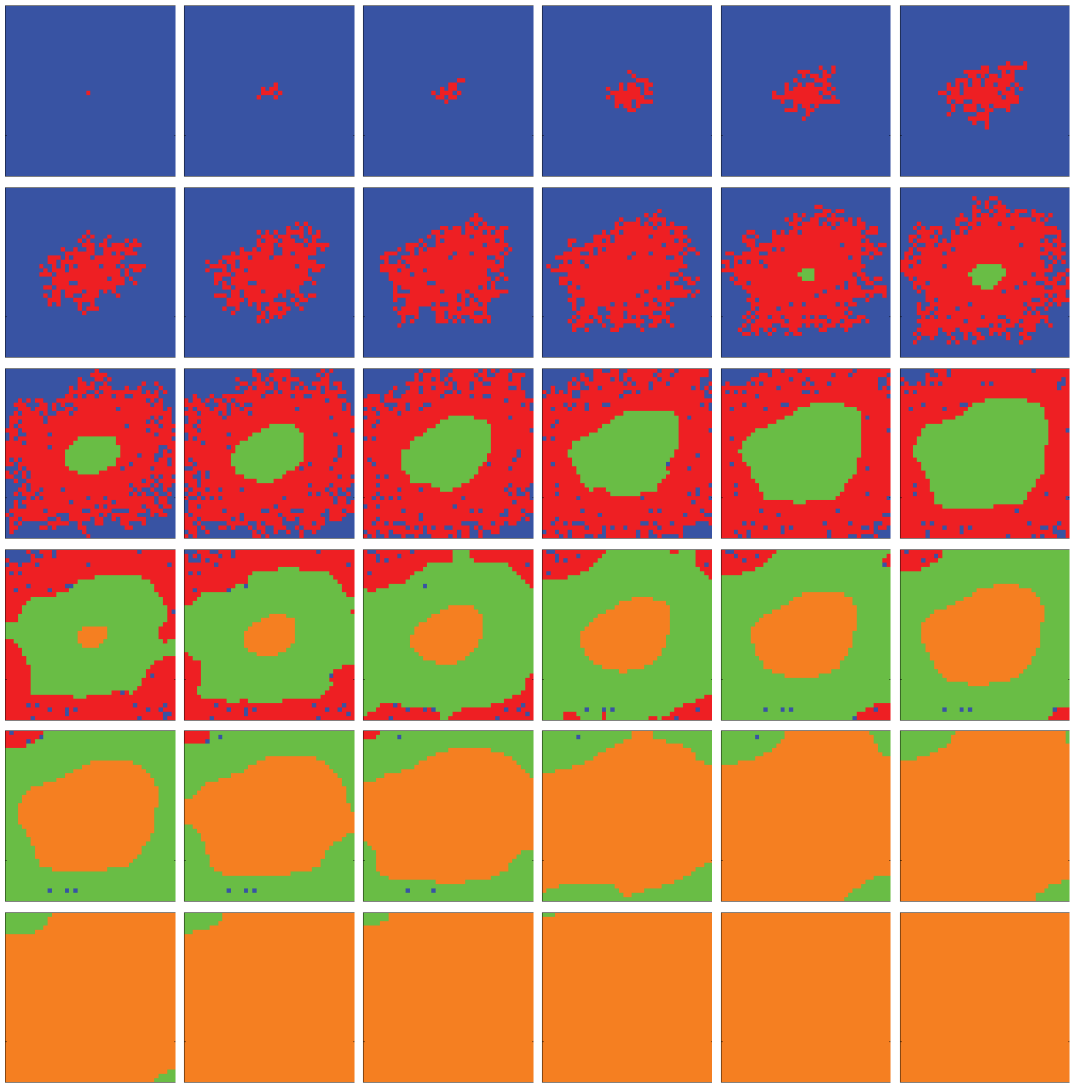

Figure 62: Time evolution of cell populations. Left-to-right, top-to-bottom: 170 generations shown in 36 frames (t = 1, 6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56, 61, 66, 71, 76, 81, 86, 91, 96, 101, 106, 111, 116, 121, 125, 130, 134, 139, 143, 148, 152, 157, 161, 166, 170). Key: *empty*, *viable*, *hypoxic*, *necrotic*.

A.9.2.2 *Temporal cell population evolution*

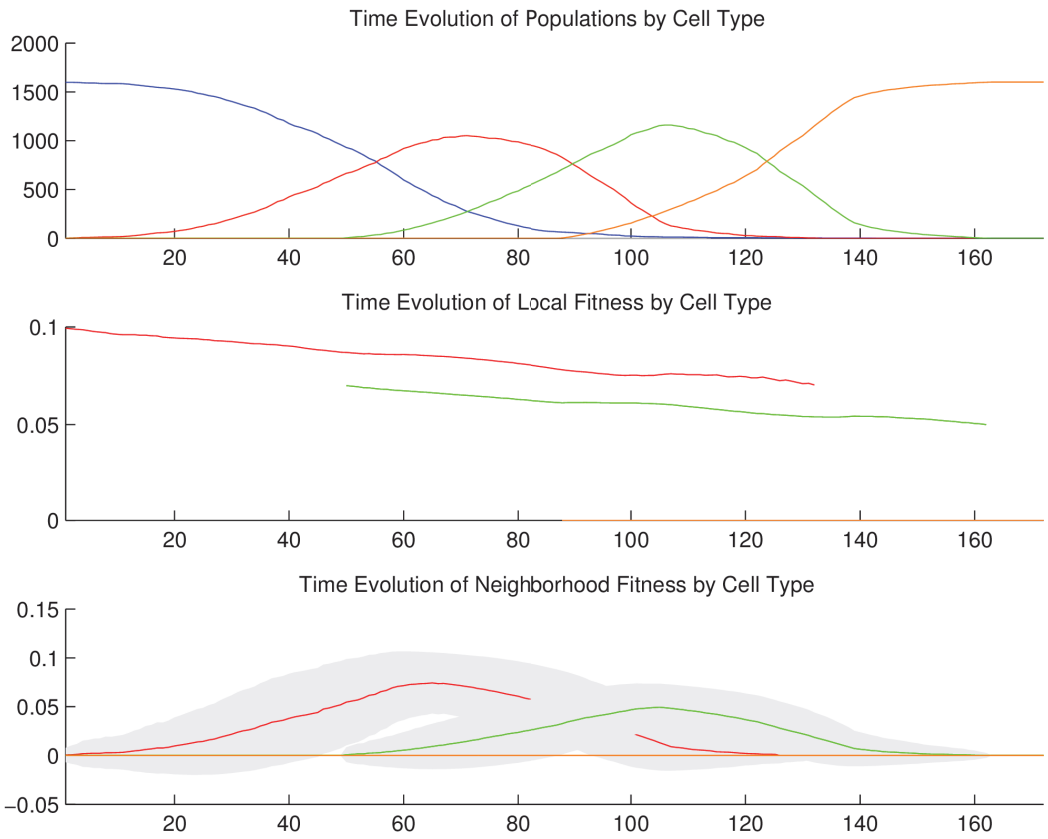

Figure 63: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *viable*, *hypoxic*, *necrotic*.

A.9.3 *Discussion*

We observe the following sequence of events in Figure 62. *Viable* cells proliferate radially outward from the center. Once most of the area is covered with *viable* cells,

then *hypoxic* cells appear in the center. *Hypoxic* cells grow radially outward at a similar rate to the *viable* cells just beyond them. Once about half of the area is covered with *hypoxic* cells, then *necrotic* cells appear in the center. *Necrotic* cells grow radially outward at a similar rate to the *hypoxic* and *viable* cells just beyond them. We observe the succession of *viable* $\rightarrow$ *hypoxic* $\rightarrow$ *necrotic* cell populations in Figure 63. The decay of *viable* and *hypoxic* cell populations is due to their proliferating outside of the spatial dimensions of the simulation while simultaneously being replaced by the succeeding population. Since *necrotic* cells cannot be replaced, their population size monotonically increases. By generation 160, *necrotic* cells have taken over the area.

A.10 STABLE LOCAL HYPOXIA WITH TWO VESSELS (2D)

A.10.1 *Setup*

In this simulation, we have *viable*, *hypoxic*, and *necrotic* tumor cells that correspond to those we see in the anti-pimonidazole stain images. The *viable* and *hypoxic* cells consume O_2 at a certain rate, and release no particles. The conditional logic implements a simple state machine in the following way. Wherever O_2 falls below a threshold, *viable* cells become ("jump" to) *hypoxic* cells—identical in every way except as follows. Wherever O_2 rises above that threshold, *hypoxic* cells become *viable* again; and wherever O_2 falls below an even lower threshold, *hypoxic* cells become *necrotic* cells. Once a cell becomes *necrotic* it has entered an absorbing state and its behavior is completely inert: it has no consumption or release profile, and it is neither replaceable nor reproductive. The specific quantities mentioned here are given in the tables below.

A.10.1.1 *Configuration parameters*

O_2
0.1

Table 92: Diffusion rate of each particle type.

O_2
0.1

Table 93: Initial concentration of each particle type.

cell type	O_2
vessel	0
empty	0
viable	0
hypoxic	0
necrotic	0

Table 94: Basal lower-bound concentration of each particle type by cell type.

cell type	O_2
vessel	∞
empty	∞
viable	∞
hypoxic	∞
necrotic	∞

Table 95: Basal upper-bound concentration of each particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	0.01
hypoxic	0.01
necrotic	0

Table 96: Consumption rate of each particle type by cell type.

cell type	O_2
vessel	0.2
empty	0
viable	0
hypoxic	0
necrotic	0

Table 97: Release rate for of particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	1
hypoxic	1
necrotic	0

Table 98: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
viable	No
hypoxic	No
necrotic	No

Table 99: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
viable	Yes
hypoxic	Yes
necrotic	No

Table 100: Reproductive predicate of each cell type.

cell type	*triggers*	*actions*
viable	$\rho_1 < 0.07$	jump to *hypoxic*
hypoxic	$\rho_1 < 0.05$	jump to *necrotic*
hypoxic	$\rho_1 > 0.07$	jump to *viable*

Table 101: Conditional triggers and actions for those cell types so configured.

A.10.1.2 *Initial conditions*

The simulation opens with the initial concentration of O_2 specified in Table 93. This diffuses with a rate specified in Table 92. We initialize the 2D lattice with *empty* cells; we place one *viable* cell in the center, and two vessels, one mid-way along the diagonal from the center to the SW corner, the other mid-way along the diagonal

from the center to the NE corner. Vessels consume no particles and release O_2 at the rate specified in Table 97.

A.10.2 *Results*

A.10.2.1 *Spatial cell population evolution*

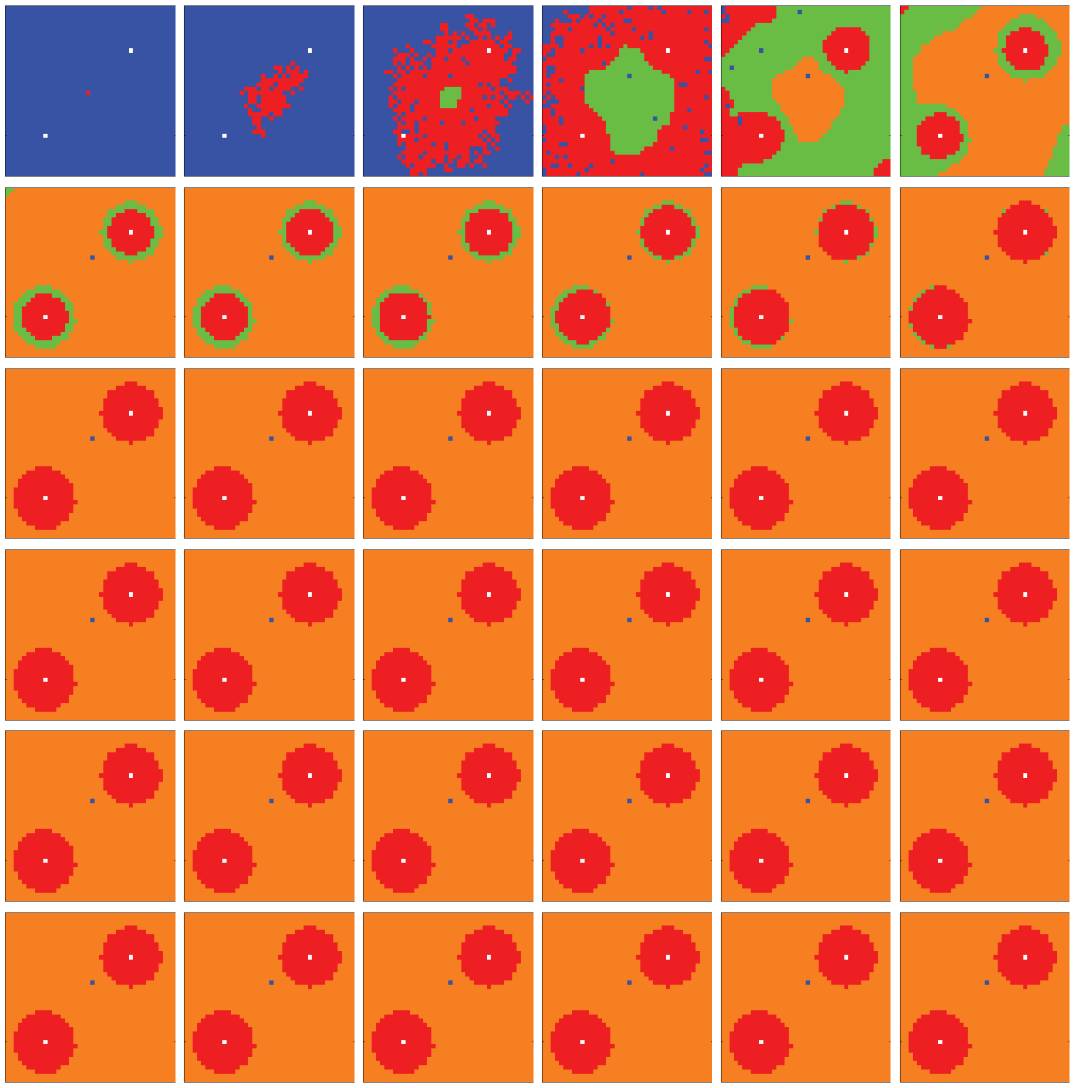

Figure 64: Time evolution of cell populations. Left-to-right, top-to-bottom: 1000 generations shown in 36 frames (t = 1, 30, 59, 88, 117, 145, 174, 202, 231, 259, 288, 316, 345, 373, 402, 430, 459, 487, 516, 544, 573, 601, 630, 658, 687, 715, 744, 772, 801, 829, 858, 886, 915, 943, 972, 1000). Key: *vessel* (white), *empty*, *viable*, *hypoxic*, *necrotic*.

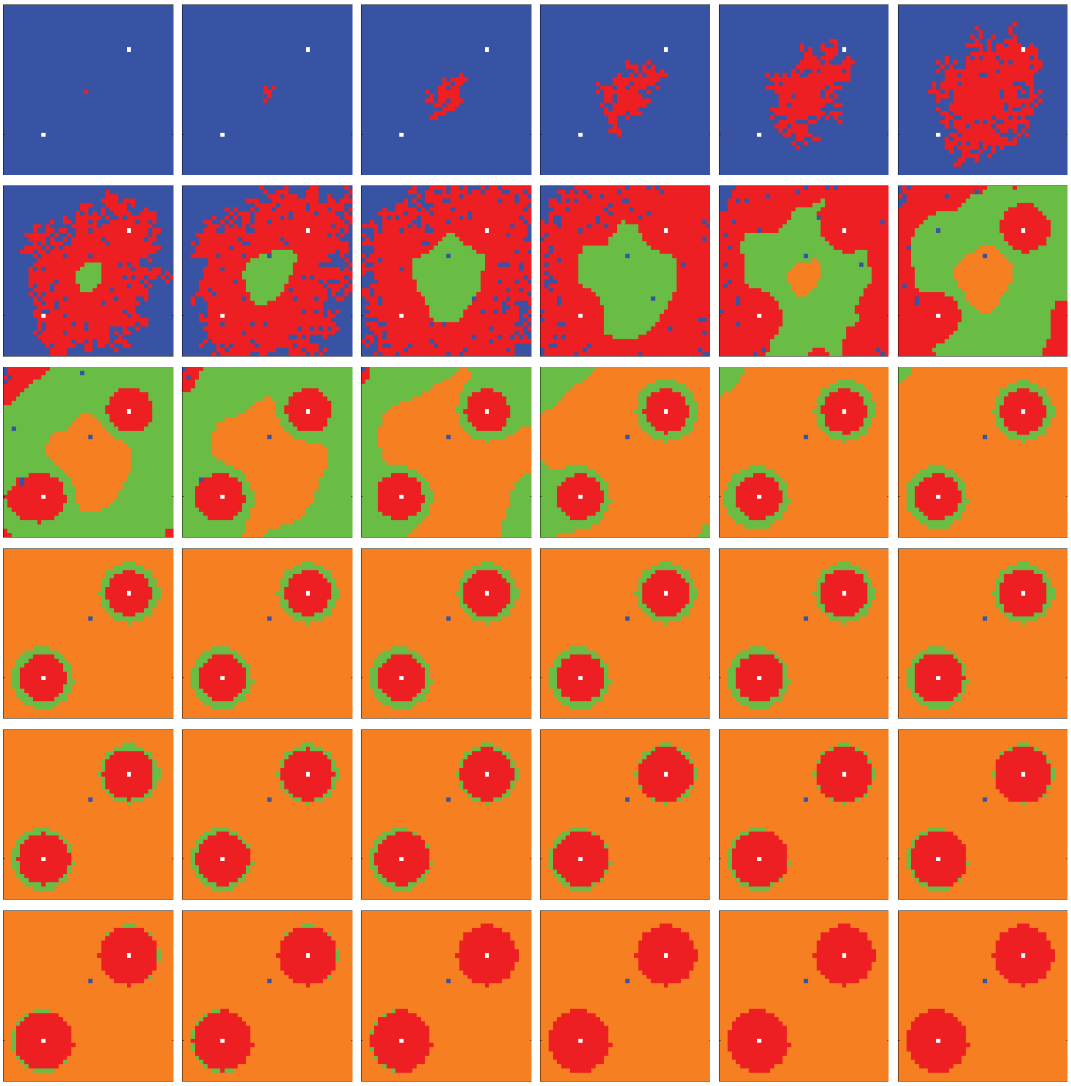

Figure 65: Time evolution of cell populations. Left-to-right, top-to-bottom: 350 generations shown in 36 frames (t = 1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191, 201, 211, 221, 231, 241, 251, 261, 271, 281, 291, 301, 311, 321, 331, 341, 350). Key: *vessel* (white), *empty*, *viable*, *hypoxic*, *necrotic*.

A.10.2.2 *Temporal cell population evolution*

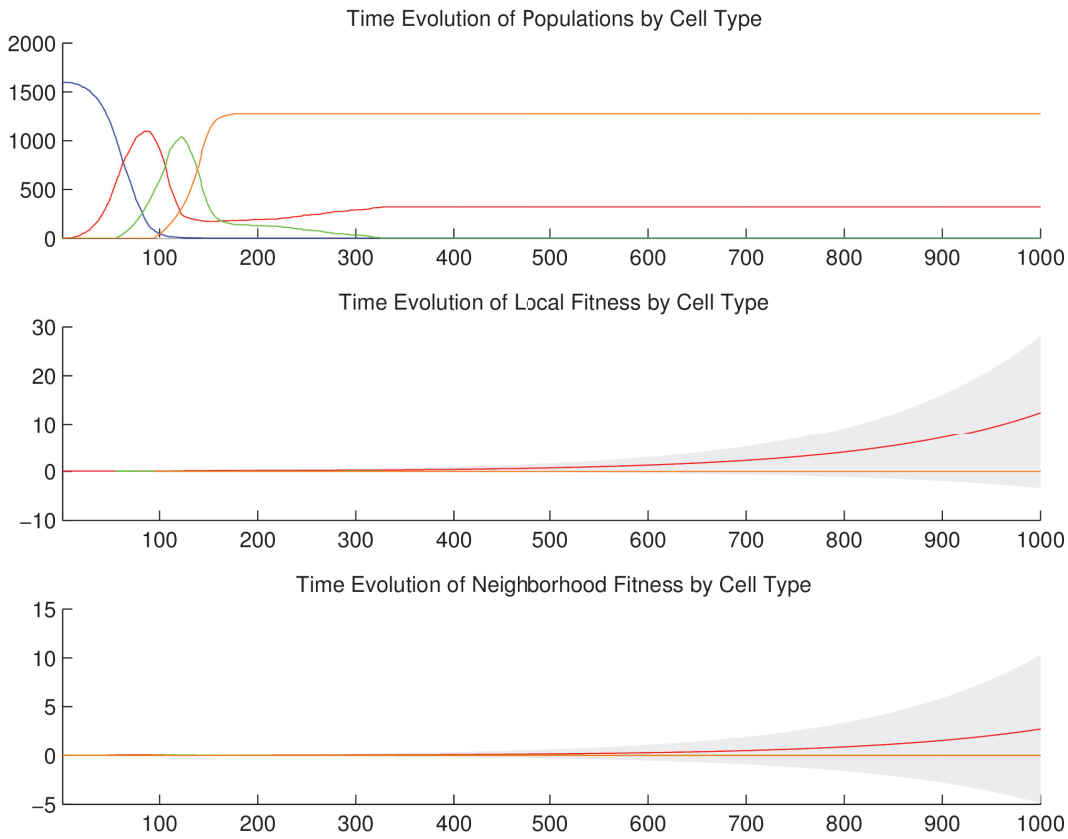

Figure 66: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *viable*, *hypoxic*, *necrotic*.

A.10.3 *Discussion*

We observe the following sequence of events in Figure 64 and Figure 65. *Viable* cells proliferate radially outward from the center. Once most of the area is covered with *vi-*

able cells, then *hypoxic* cells appear in the center. *Hypoxic* cells grow radially outward at a similar rate to the *viable* cells just beyond them. They approach vessels (surrounded by *viable* cells) up to some radial distance away from each vessel, the zone into which O_2 diffuses with a sufficient concentration to maintain *viable* cells. Once about half of the area is covered with *hypoxic* cells, then *necrotic* cells appear in the center. *Necrotic* cells grow radially outward at a similar rate to the *hypoxic* and *viable* cells just beyond them. They approach vessels (surrounded by concentrically situated *viable* and *hypoxic* cells) up to some radial distance away from each vessel, the zone into which O_2 diffuses with a sufficient concentration to maintain *viable* and *hypoxic* cells. We eventually observe islands of concentrically situated *viable* and *hypoxic* cells, surrounded by a sea of *necrotic* cells, similar to what we see in the anti-pimonidazole stain images. This arrangement is stable for awhile, before the imbalance of three factors related to O_2—diffusion rate (Table 92), vessel release rate (Table 97), and *viable* and *hypoxic* consumption rate (Table 96)—allow O_2 concentration to climb out of control, and eventually convert all of the *hypoxic* cells back into *viable* cells. We observe the succession of *viable* $\rightarrow$ *hypoxic* $\rightarrow$ *necrotic* cell populations in Figure 66. The initial decay of *viable* and *hypoxic* cell populations is due to their proliferating outside of the spatial dimensions of the simulation while simultaneously being replaced by the succeeding population. Since *necrotic* cells cannot be replaced, their population size monotonically increases. Beginning at generation 200 and continuing to the end of the simulation, the growth in the *viable* cell population at the expense of the *hypoxic* cell population occurs for the reasons just discussed.

A.11 STABLE LOCAL HYPOXIA WITH MANY VESSELS (2D)

A.11.1 *Setup*

In this simulation, we have *viable*, *hypoxic*, and *necrotic* tumor cells that correspond to those we see in the anti-pimonidazole stain images. The *viable* and *hypoxic* cells consume O_2 at a certain rate, and release no particles. The conditional logic implements a simple state machine in the following way. Wherever O_2 falls below a threshold, *viable* cells become ("jump" to) *hypoxic* cells—identical in every way except as follows. Wherever O_2 rises above that threshold, *hypoxic* cells become *viable* again; and wherever O_2 falls below an even lower threshold, *hypoxic* cells become *necrotic* cells. Once a cell becomes *necrotic* it has entered an absorbing state and its behavior is completely inert: it has no consumption or release profile, and it is neither replaceable nor reproductive. The specific quantities mentioned here are given in the tables below.

A.11.1.1 *Configuration parameters*

O_2
0.12

Table 102: Diffusion rate of each particle type.

O_2
0.1

Table 103: Initial concentration of each particle type.

cell type	O_2
vessel	0
empty	0
viable	0
hypoxic	0
necrotic	0

Table 104: Basal lower-bound concentration of each particle type by cell type.

cell type	O_2
vessel	∞
empty	∞
viable	∞
hypoxic	∞
necrotic	∞

Table 105: Basal upper-bound concentration of each particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	0.01
hypoxic	0.01
necrotic	0

Table 106: Consumption rate of each particle type by cell type.

cell type	O_2
vessel	0.2
empty	0
viable	0
hypoxic	0
necrotic	0

Table 107: Release rate for of particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	1
hypoxic	1
necrotic	0

Table 108: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
viable	No
hypoxic	No
necrotic	No

Table 109: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
viable	Yes
hypoxic	Yes
necrotic	No

Table 110: Reproductive predicate of each cell type.

cell type	*triggers*	*actions*
viable	$\rho_1 < 0.07$	jump to *hypoxic*
hypoxic	$\rho_1 < 0.05$	jump to *necrotic*
hypoxic	$\rho_1 > 0.07$	jump to *viable*

Table 111: Conditional triggers and actions for those cell types so configured.

A.11.1.2 *Initial conditions*

The simulation opens with the initial concentration of O_2 specified in Table 103. This diffuses with a rate specified in Table 102. We initialize the 2D lattice with *empty* cells; we place one *viable* cell in the center, and 10 vessels randomly about. Vessels consume no particles and release O_2 at the rate specified in Table 107.

A.11.2 *Results*

A.11.2.1 *Spatial cell population evolution*

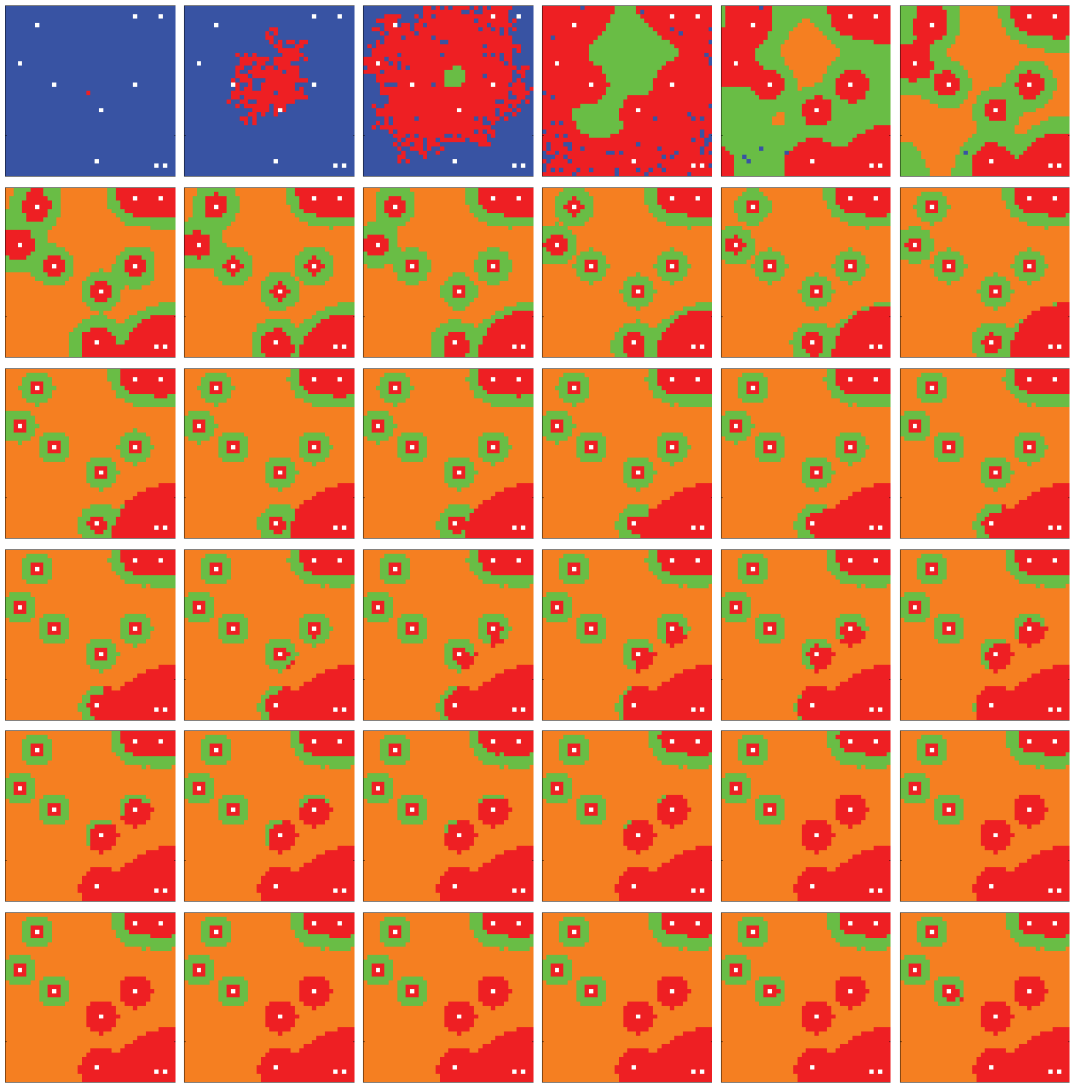

Figure 67: Time evolution of cell populations. Left-to-right, top-to-bottom: 1000 generations shown in 36 frames (t = 1, 30, 59, 88, 117, 145, 174, 202, 231, 259, 288, 316, 345, 373, 402, 430, 459, 487, 516, 544, 573, 601, 630, 658, 687, 715, 744, 772, 801, 829, 858, 886, 915, 943, 972, 1000). Key: *vessel* (white), *empty*, *viable*, *hypoxic*, *necrotic*.

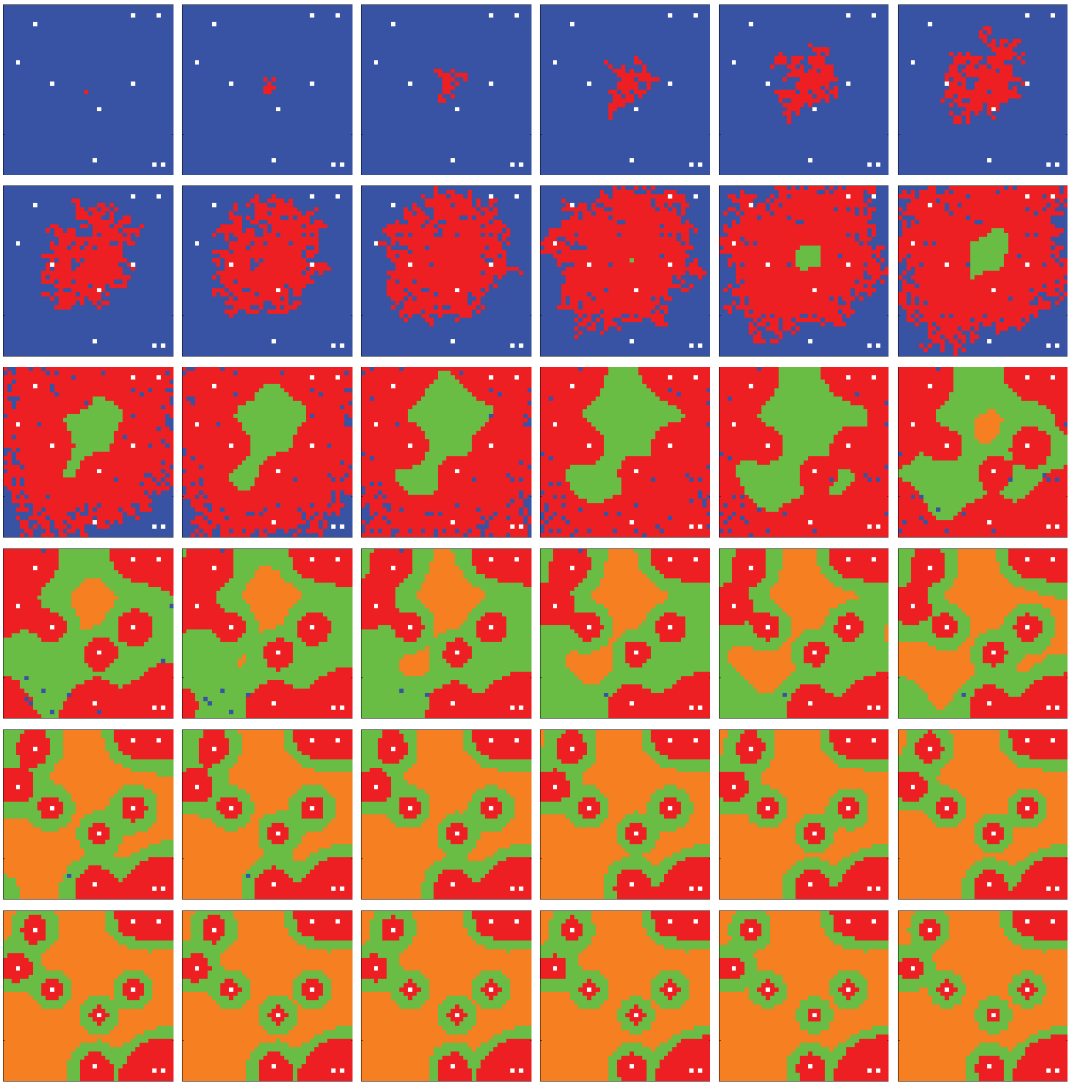

Figure 68: Time evolution of cell populations. Left-to-right, top-to-bottom: 220 generations shown in 36 frames (t = 1, 7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 91, 97, 103, 110, 116, 123, 129, 136, 142, 149, 155, 162, 168, 175, 181, 188, 194, 201, 207, 214, 220). Key: *vessel* (white), *empty*, *viable*, *hypoxic*, *necrotic*.

A.11.2.2 *Temporal cell population evolution*

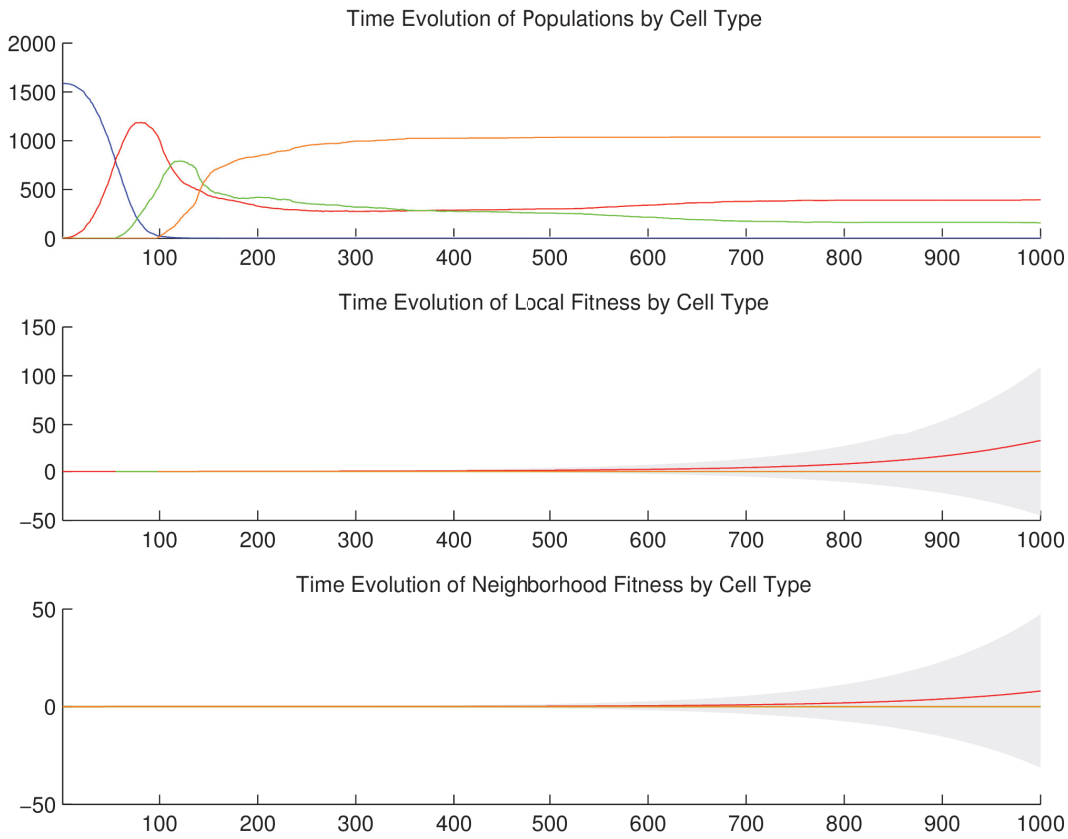

Figure 69: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *viable*, *hypoxic*, *necrotic*.

A.11.3 *Discussion*

We observe the following sequence of events in Figure 67 and Figure 68. *Viable* cells proliferate radially outward from the center. Once most of the area is covered with *vi-*

able cells, then *hypoxic* cells appear in the center. *Hypoxic* cells grow radially outward at a similar rate to the *viable* cells just beyond them. They approach vessels (surrounded by *viable* cells) up to some radial distance away from each vessel, the zone into which O_2 diffuses with a sufficient concentration to maintain *viable* cells. Once about half of the area is covered with *hypoxic* cells, then *necroic* cells appear in the center. *Necrotic* cells grow radially outward at a similar rate to the *hypoxic* and *viable* cells just beyond them. They approach vessels (surrounded by concentrically situated *viable* and *hypoxic* cells) up to some radial distance away from each vessel, the zone into which O_2 diffuses with a sufficient concentration to maintain *viable* and *hypoxic* cells. We eventually observe islands of concentrically situated *viable* and *hypoxic* cells, surrounded by a sea of *necrotic* cells, similar to what we see in the anti-pimonidazole stain images. This arrangement is stable for awhile, before the imbalance of three factors related to O_2—diffusion rate (Table 102), vessel release rate (Table 107), and *viable* and *hypoxic* consumption rate (Table 106)—allow O_2 concentration to climb out of control, at least in certain locales where random vessels may be close together, and eventually convert all of the *hypoxic* cells back into *viable* cells. We observe the succession of *viable* $\rightarrow$ *hypoxic* $\rightarrow$ *necrotic* cell populations in Figure 69. The initial decay of *viable* and *hypoxic* cell populations is due to their proliferating outside of the spatial dimensions of the simulation while simultaneously being replaced by the succeeding population. Since *necrotic* cells cannot be replaced, their population size monotonically increases. Beginning at generation 200 and continuing to the end of the simulation, the growth in the *viable* cell population at the expense of the *hypoxic* cell population occurs for the reasons just discussed.

A.12 NECROTIC CORE (3D)

A.12.1 *Setup*

In this simulation, we have *viable*, *hypoxic*, and *necrotic* tumor cells that correspond to those we see in the anti-pimonidazole stain images. The *viable* and *hypoxic* cells consume O_2 at a certain rate, and release no particles. The conditional logic implements a simple state machine in the following way. Wherever O_2 falls below a threshold, *viable* cells become ("jump" to) *hypoxic* cells—identical in every way except as follows. Wherever O_2 rises above that threshold, *hypoxic* cells become *viable* again; and wherever O_2 falls below an even lower threshold, *hypoxic* cells become *necrotic* cells. Once a cell becomes *necrotic* it has entered an absorbing state and its behavior is completely inert: it has no consumption or release profile, and it is neither replaceable nor reproductive. The specific quantities mentioned here are given in the tables below.

A.12.1.1 *Configuration parameters*

O_2
0.1

Table 112: Diffusion rate of each particle type.

O_2
0.1

Table 113: Initial concentration of each particle type.

cell type	O_2
vessel	0
empty	0
viable	0
hypoxic	0
necrotic	0

Table 114: Basal lower-bound concentration of each particle type by cell type.

cell type	O_2
vessel	∞
empty	∞
viable	∞
hypoxic	∞
necrotic	∞

Table 115: Basal upper-bound concentration of each particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	0.01
hypoxic	0.01
necrotic	0

Table 116: Consumption rate of each particle type by cell type.

cell type	O_2
vessel	0.2
empty	0
viable	0
hypoxic	0
necrotic	0

Table 117: Release rate for of particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	1
hypoxic	1
necrotic	0

Table 118: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
viable	No
hypoxic	No
necrotic	No

Table 119: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
viable	Yes
hypoxic	Yes
necrotic	No

Table 120: Reproductive predicate of each cell type.

cell type	*triggers*	*actions*
viable	$\rho_1 < 0.07$	jump to *hypoxic*
hypoxic	$\rho_1 < 0.05$	jump to *necrotic*
hypoxic	$\rho_1 > 0.07$	jump to *viable*

Table 121: Conditional triggers and actions for those cell types so configured.

A.12.1.2 *Initial conditions*

The simulation opens with the initial concentration of O_2 specified in Table 113. This diffuses with a rate specified in Table 112. We initialize the 3D lattice with *empty* cells, and we place one *viable* cell in the center.

A.12.2 *Results*

A.12.2.1 *Spatial cell population evolution*

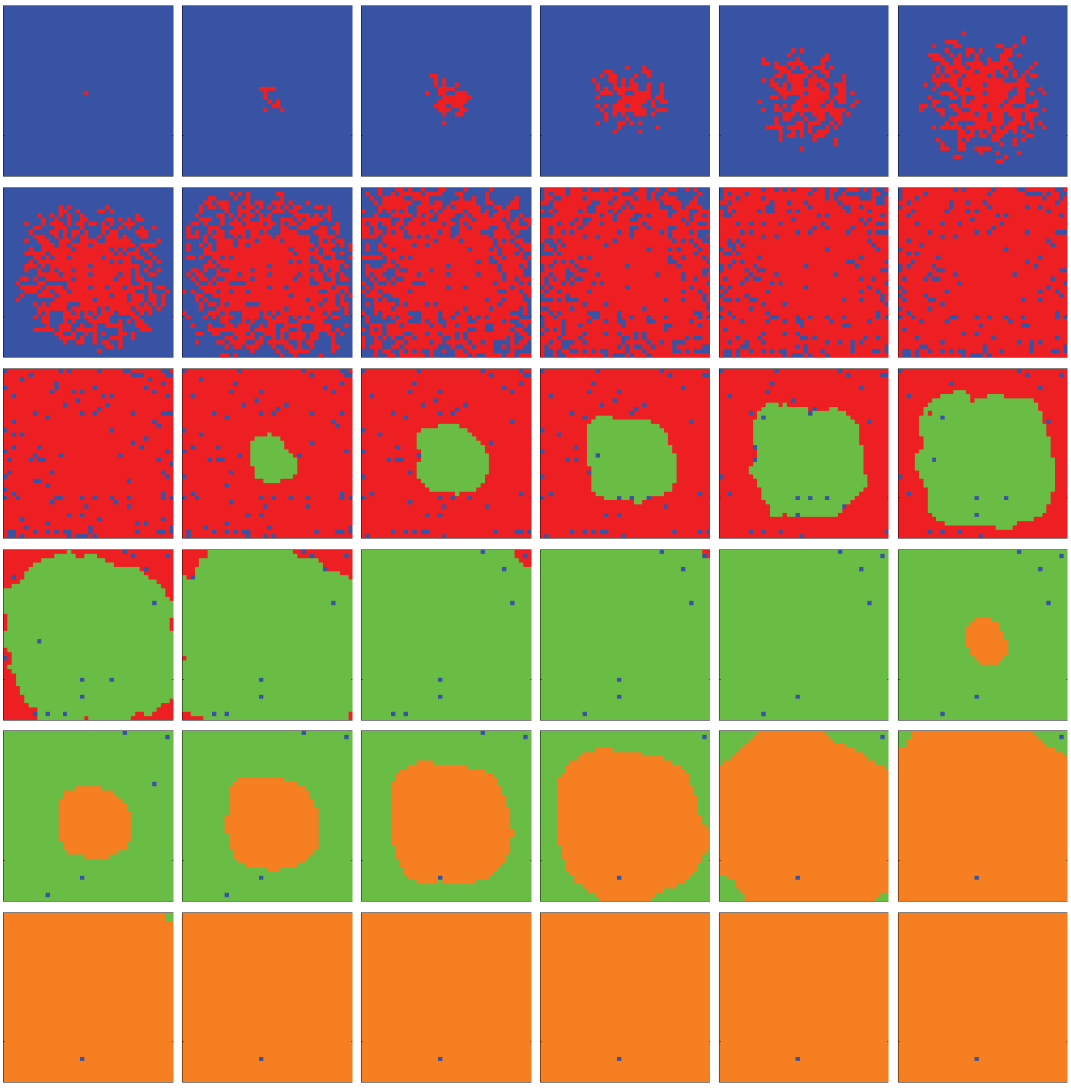

Figure 70: Time evolution of cell populations on the plane $z = 20$. Left-to-right, top-to-bottom: 130 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 60, 64, 67, 71, 74, 78, 81, 85, 88, 92, 95, 99, 102, 106, 109, 113, 116, 120, 123, 127, 130). Key: *empty*, *viable*, *hypoxic*, *necrotic*.

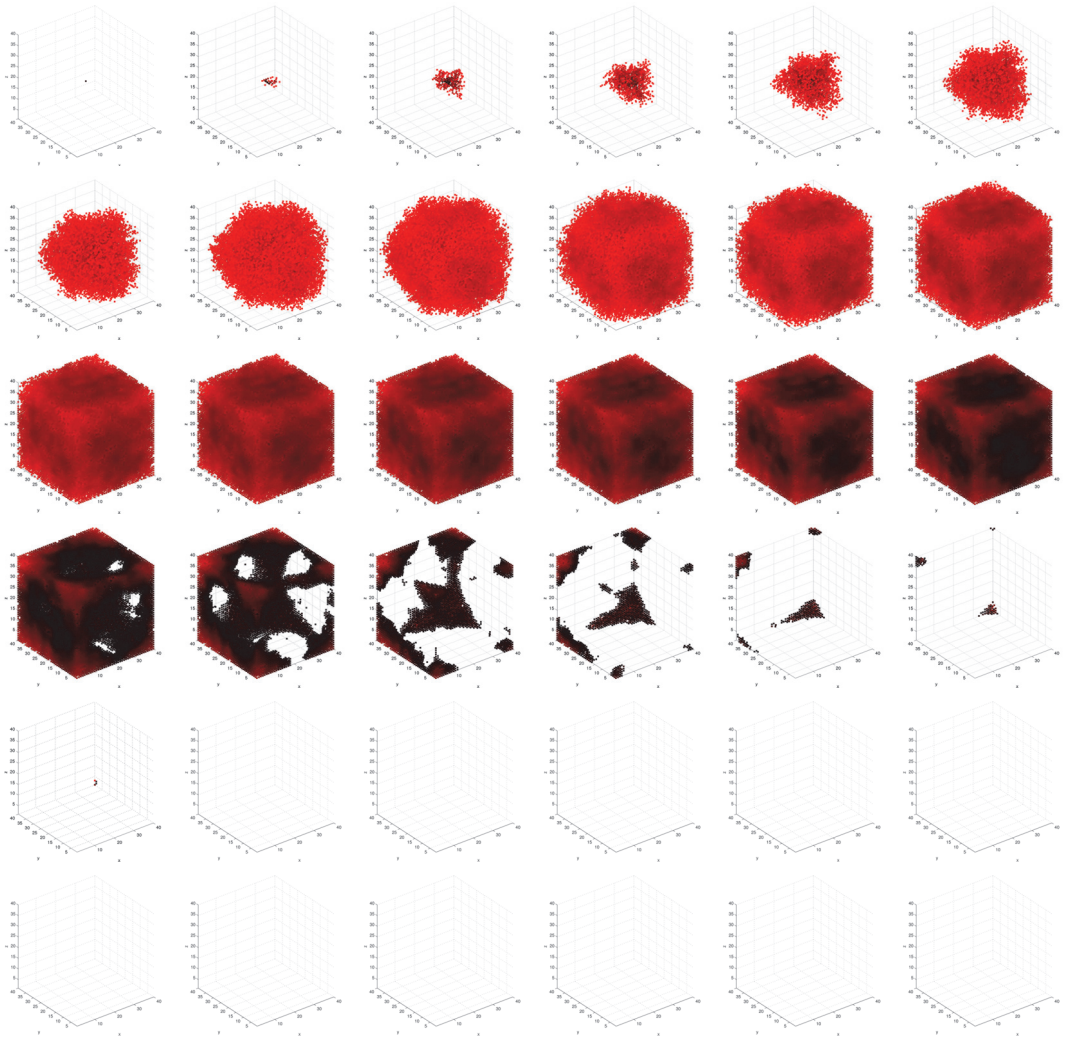

Figure 71: Time evolution of local fitness for *viable* cells. Left-to-right, top-to-bottom: 130 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 60, 64, 67, 71, 74, 78, 81, 85, 88, 92, 95, 99, 102, 106, 109, 113, 116, 120, 123, 127, 130).

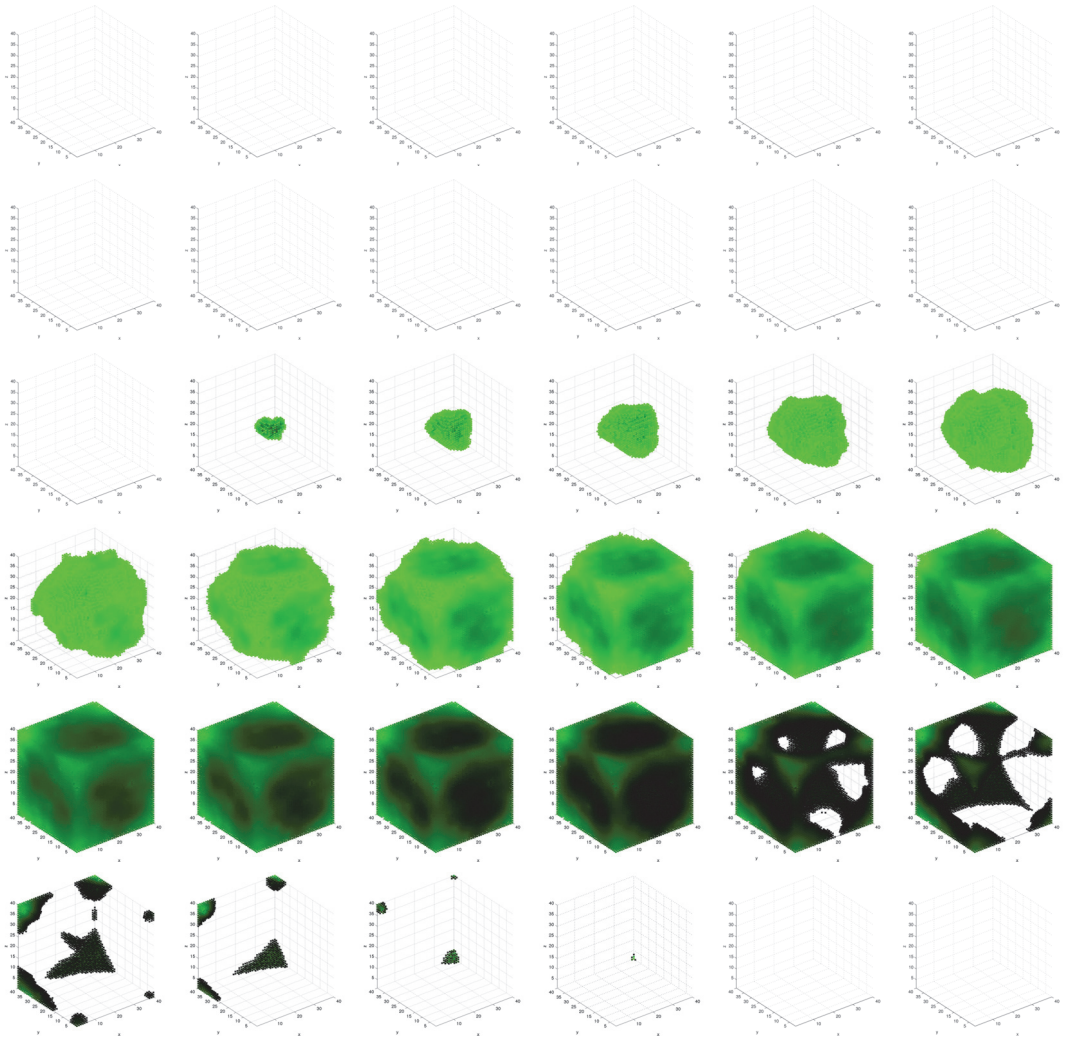

Figure 72: Time evolution of local fitness for *hypoxic* cells. Left-to-right, top-to-bottom: 130 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 60, 64, 67, 71, 74, 78, 81, 85, 88, 92, 95, 99, 102, 106, 109, 113, 116, 120, 123, 127, 130).

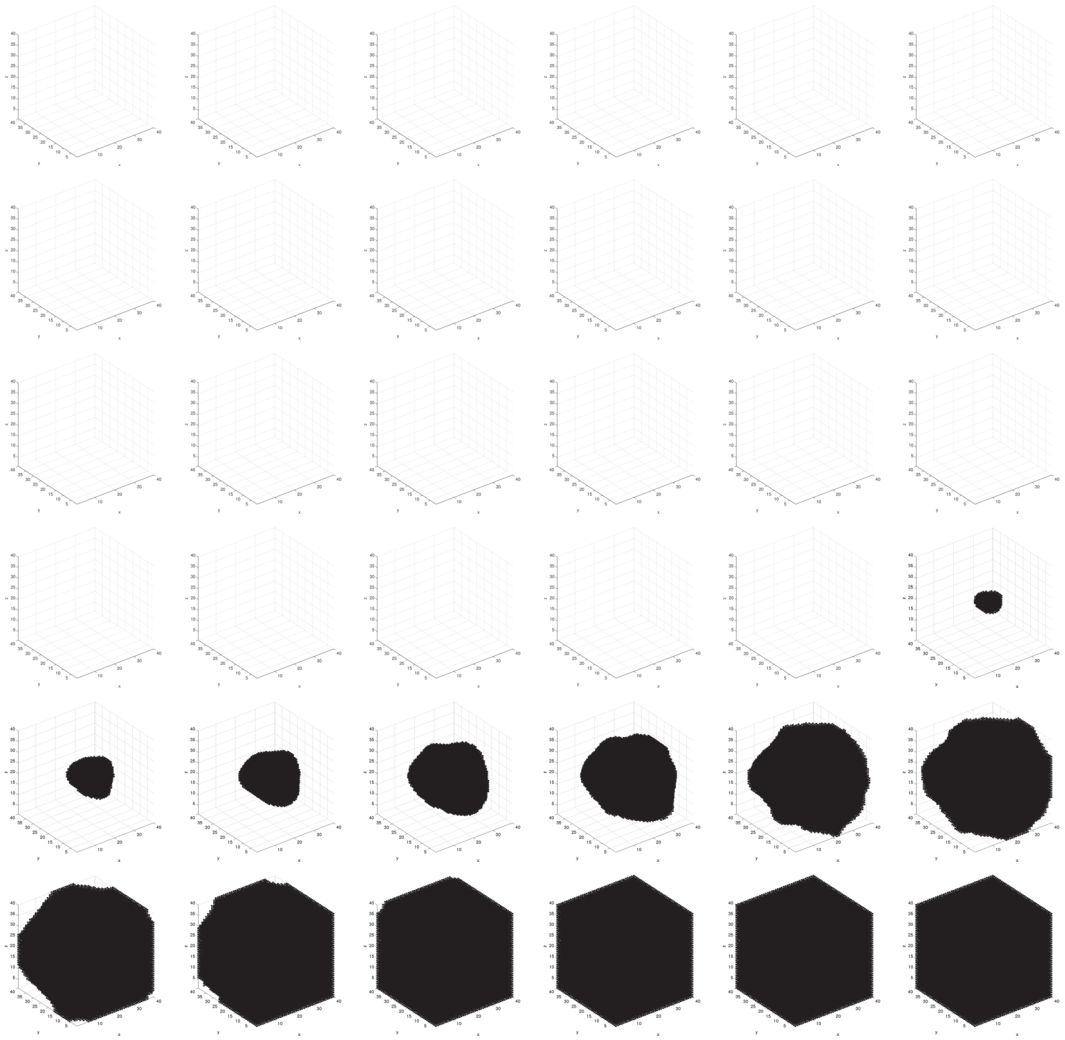

Figure 73: Time evolution of local fitness for *necrotic* cells. Left-to-right, top-to-bottom: 130 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 60, 64, 67, 71, 74, 78, 81, 85, 88, 92, 95, 99, 102, 106, 109, 113, 116, 120, 123, 127, 130).

A.12.2.2 *Temporal cell population evolution*

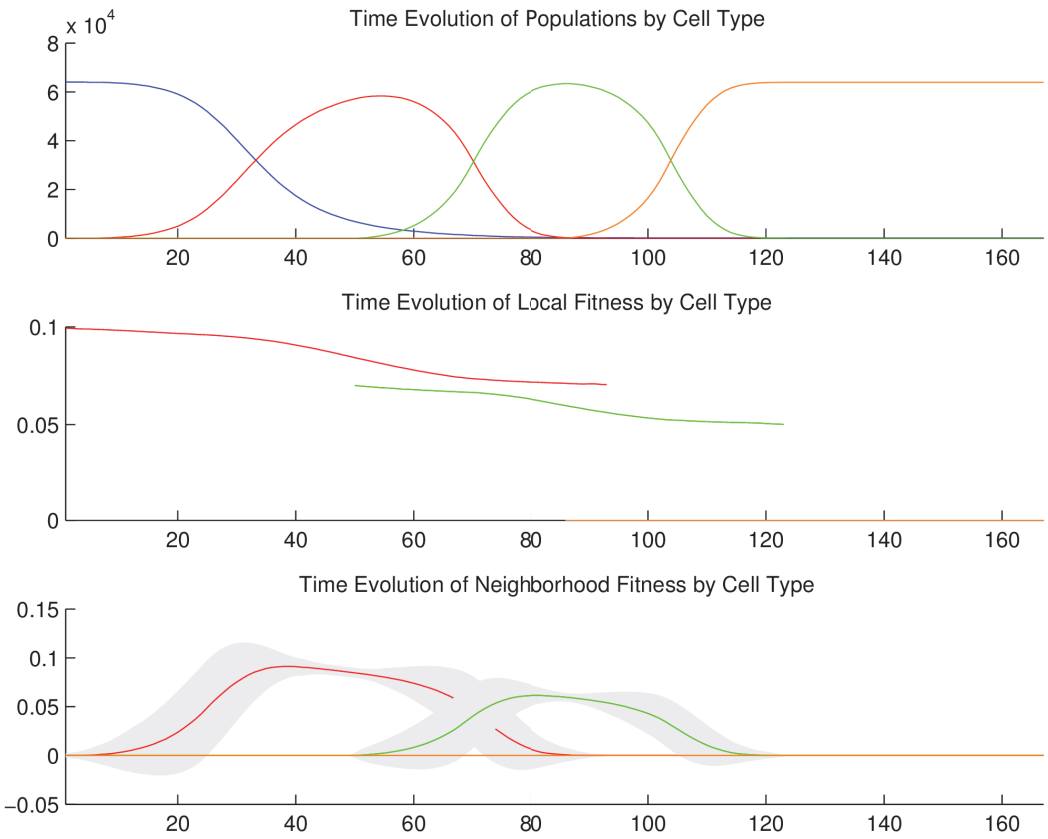

Figure 74: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *viable*, *hypoxic*, *necrotic*.

A.12.3 *Discussion*

We observe the following sequence of events in Figure 70, Figure 71, Figure 72, and Figure 73. *Viable* cells proliferate radially outward from the center. Once most of the

volume is covered with *viable* cells, then *hypoxic* cells appear in the center. *Hypoxic* cells grow radially outward at a similar rate to the *viable* cells just beyond them. Once about half of the volume is covered with *hypoxic* cells, then *necrotic* cells appear in the center. *Necrotic* cells grow radially outward at a similar rate to the *hypoxic* and *viable* cells just beyond them. We observe the succession of *viable* $\rightarrow$ *hypoxic* $\rightarrow$ *necrotic* cell populations in Figure 74. The decay of *viable* and *hypoxic* cell populations is due to their proliferating outside of the spatial dimensions of the simulation while simultaneously being replaced by the succeeding population. Since *necrotic* cells cannot be replaced, their population size monotonically increases. By generation 120, *necrotic* cells have taken over the volume.

A.13 STABLE LOCAL HYPOXIA WITH TWO VESSELS (3D)

A.13.1 *Setup*

In this simulation, we have *viable*, *hypoxic*, and *necrotic* tumor cells that correspond to those we see in the anti-pimonidazole stain images. The *viable* and *hypoxic* cells consume O_2 at a certain rate, and release no particles. The conditional logic implements a simple state machine in the following way. Wherever O_2 falls below a threshold, *viable* cells become ("jump" to) *hypoxic* cells—identical in every way except as follows. Wherever O_2 rises above that threshold, *hypoxic* cells become *viable* again; and wherever O_2 falls below an even lower threshold, *hypoxic* cells become *necrotic* cells. Once a cell becomes *necrotic* it has entered an absorbing state and its behavior is completely inert: it has no consumption or release profile, and it is neither replaceable nor reproductive. The specific quantities mentioned here are given in the tables below.

A.13.1.1 *Configuration parameters*

O_2
0.1

Table 122: Diffusion rate of each particle type.

O_2
0.1

Table 123: Initial concentration of each particle type.

cell type	O_2
vessel	0
empty	0
viable	0
hypoxic	0
necrotic	0

Table 124: Basal lower-bound concentration of each particle type by cell type.

cell type	O_2
vessel	∞
empty	∞
viable	∞
hypoxic	∞
necrotic	∞

Table 125: Basal upper-bound concentration of each particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	0.01
hypoxic	0.01
necrotic	0

Table 126: Consumption rate of each particle type by cell type.

cell type	O_2
vessel	0.2
empty	0
viable	0
hypoxic	0
necrotic	0

Table 127: Release rate for of particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	1
hypoxic	1
necrotic	0

Table 128: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
viable	No
hypoxic	No
necrotic	No

Table 129: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
viable	Yes
hypoxic	Yes
necrotic	No

Table 130: Reproductive predicate of each cell type.

cell type	*triggers*	*actions*
viable	$\rho_1 < 0.07$	jump to *hypoxic*
hypoxic	$\rho_1 < 0.05$	jump to *necrotic*
hypoxic	$\rho_1 > 0.07$	jump to *viable*

Table 131: Conditional triggers and actions for those cell types so configured.

A.13.1.2 *Initial conditions*

The simulation opens with the initial concentration of O_2 specified in Table 123. This diffuses with a rate specified in Table 122. We initialize the 3D lattice with *empty* cells; we place one *viable* cell in the center, and two vessels, one mid-way along the diagonal from the center to the SW corner of the plane $z = 20$, the other mid-way along the

diagonal from the center to the NE corner of the plane $z = 20$. Vessels consume no particles and release O_2 at the rate specified in Table 127.

A.13.2 *Results*

A.13.2.1 *Spatial cell population evolution*

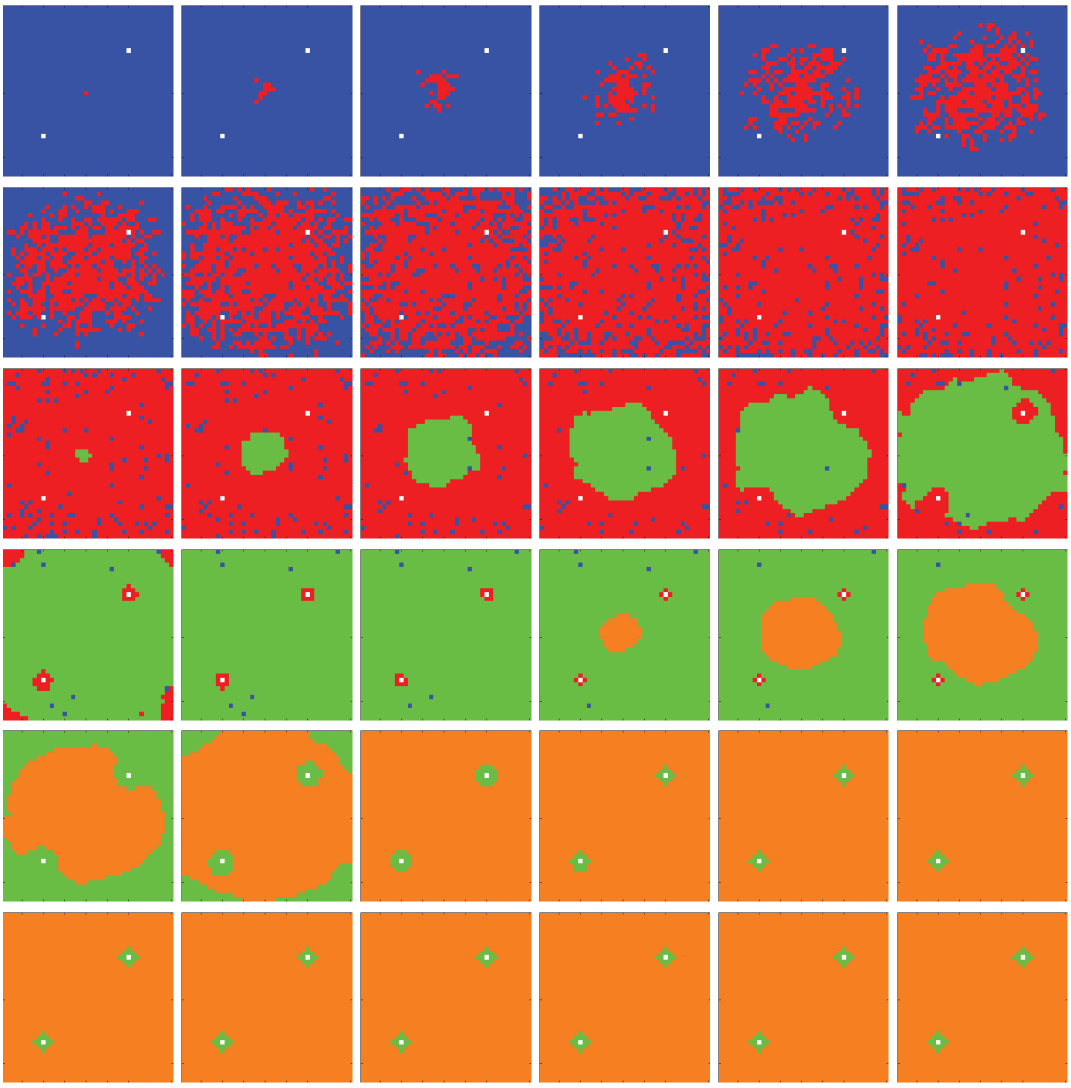

Figure 75: Time evolution of cell populations on the plane $z = 20$. Left-to-right, top-to-bottom: 150 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 74, 78, 83, 87, 92, 96, 101, 105, 110, 114, 119, 123, 128, 132, 137, 141, 146, 150). Key: *vessel* (white), *empty*, *viable*, *hypoxic*, *necrotic*.

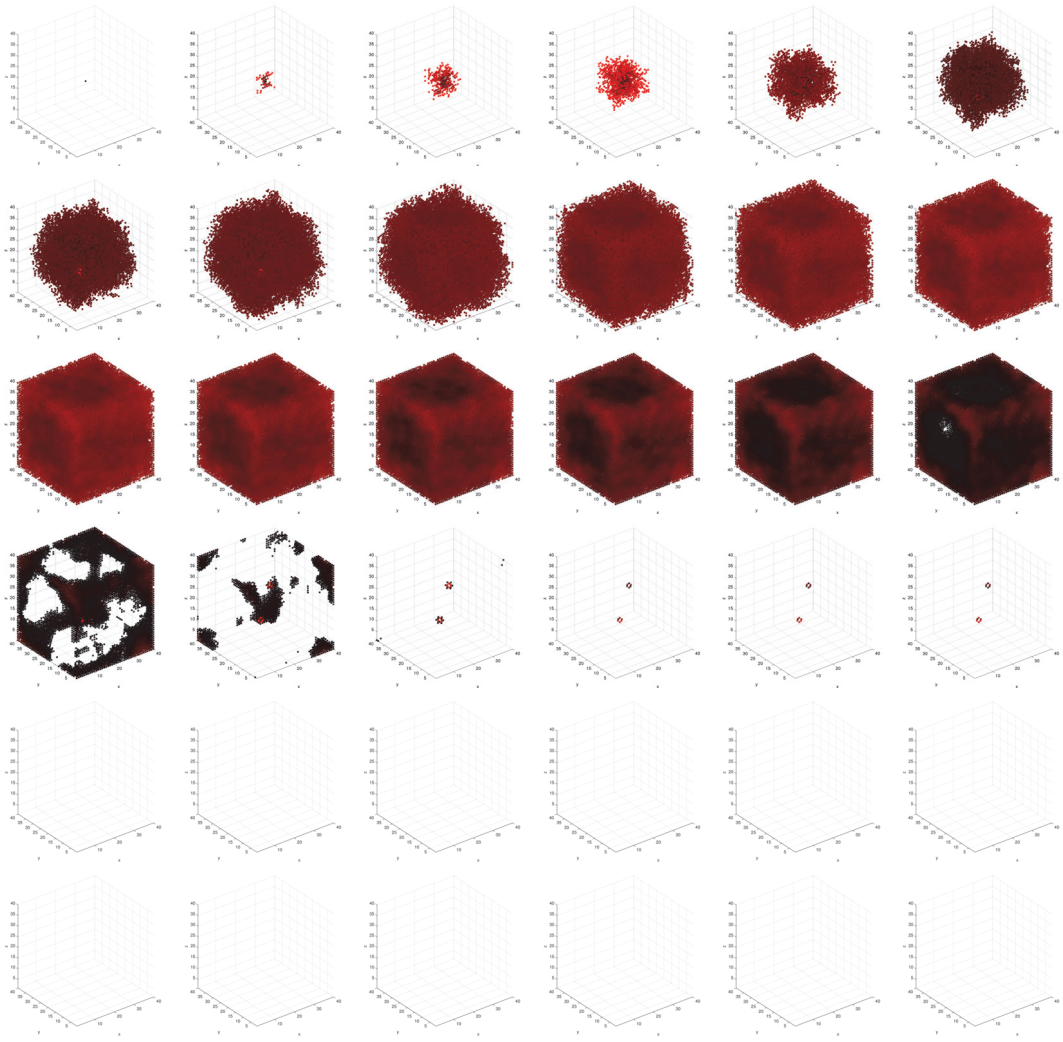

Figure 76: Time evolution of local fitness for *viable* cells. Left-to-right, top-to-bottom: 150 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 74, 78, 83, 87, 92, 96, 101, 105, 110, 114, 119, 123, 128, 132, 137, 141, 146, 150).

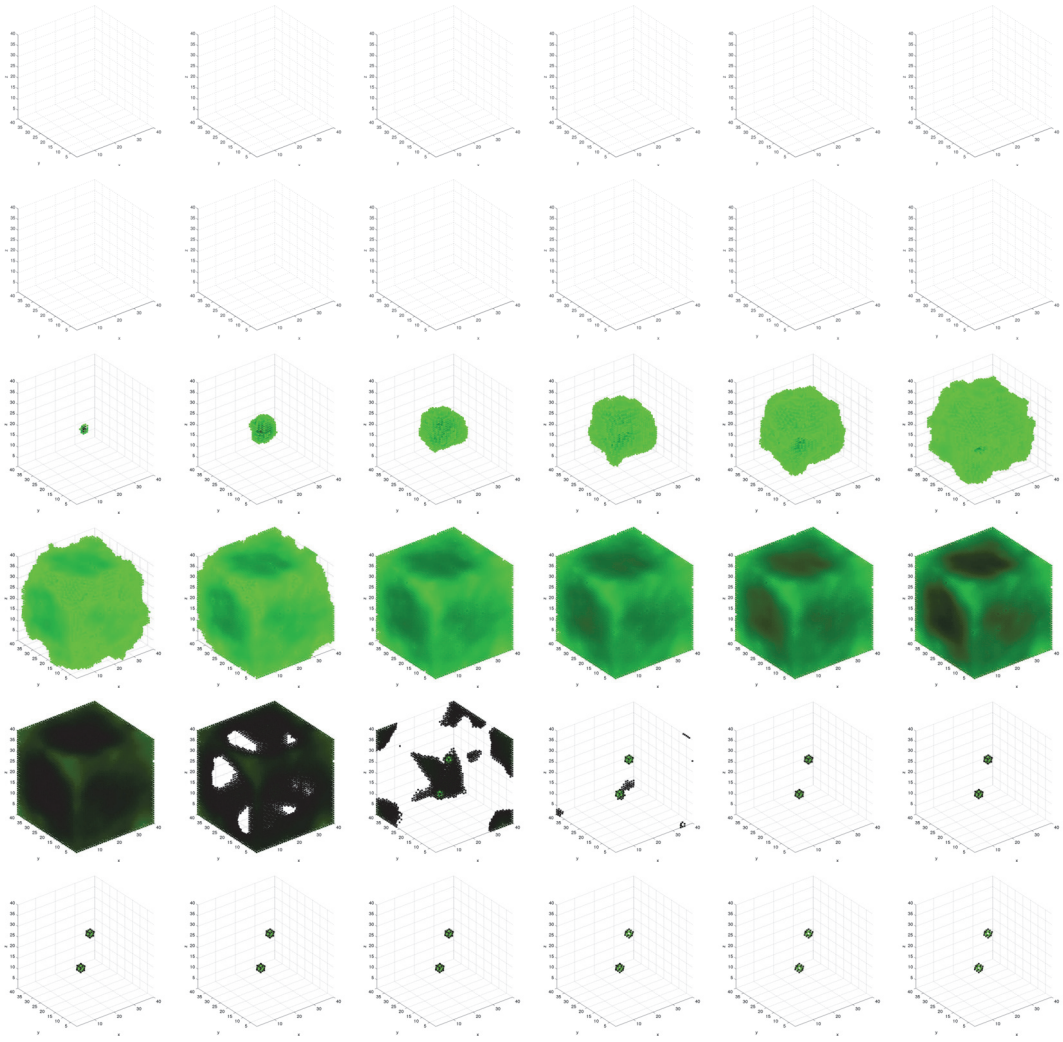

Figure 77: Time evolution of local fitness for *hypoxic* cells. Left-to-right, top-to-bottom: 150 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 74, 78, 83, 87, 92, 96, 101, 105, 110, 114, 119, 123, 128, 132, 137, 141, 146, 150).

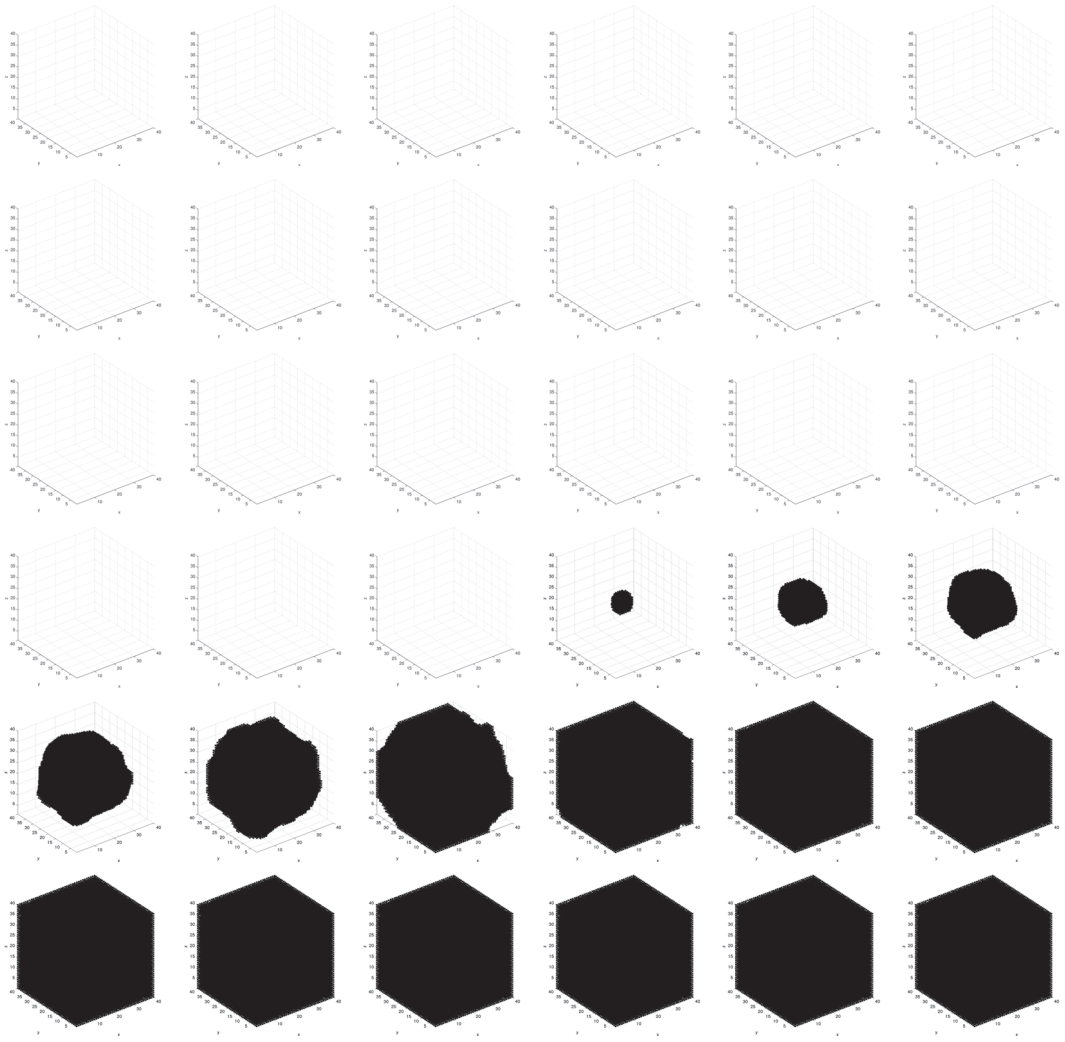

Figure 78: Time evolution of local fitness for *necrotic* cells. Left-to-right, top-to-bottom: 150 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 74, 78, 83, 87, 92, 96, 101, 105, 110, 114, 119, 123, 128, 132, 137, 141, 146, 150).

A.13.2.2 *Temporal cell population evolution*

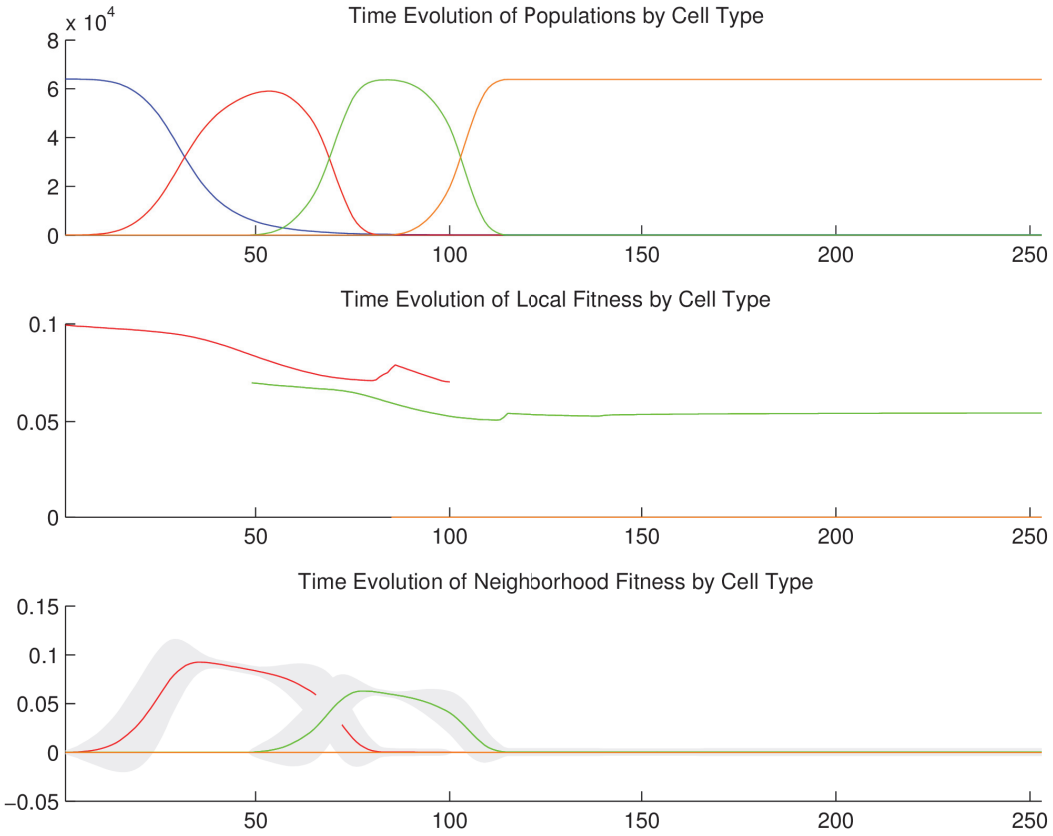

Figure 79: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *viable*, *hypoxic*, *necrotic*.

A.13.3 *Discussion*

We observe the following sequence of events in Figure 75, Figure 76, Figure 77, and Figure 78. *Viable* cells proliferate radially outward from the center. Once most of the

volume is covered with *viable* cells, then *hypoxic* cells appear in the center. *Hypoxic* cells grow radially outward at a similar rate to the *viable* cells just beyond them. They approach vessels (surrounded by *viable* cells) up to some radial distance away from each vessel, the zone into which O_2 diffuses with a sufficient concentration to maintain *viable* cells. Once about half of the volume is covered with *hypoxic* cells, then *necrotic* cells appear in the center. *Necrotic* cells grow radially outward at a similar rate to the *hypoxic* and *viable* cells just beyond them. They approach vessels (surrounded by concentrically situated *viable* and *hypoxic* cells) up to some radial distance away from each vessel, the zone into which O_2 diffuses with a sufficient concentration to maintain *viable* and *hypoxic* cells. We eventually observe islands of concentrically situated *viable* and *hypoxic* cells, surrounded by a sea of *necrotic* cells, similar to what we see in the anti-pimonidazole stain images. This arrangement is stable for awhile, before the remaining *viable* cells become *hypoxic* cells. These *hypoxic* cell islands persist indefinitely. We observe the succession of *viable* $\rightarrow$ *hypoxic* $\rightarrow$ *necrotic* cell populations in Figure 79. The initial decay of *viable* and *hypoxic* cell populations is due to their proliferating outside of the spatial dimensions of the simulation while simultaneously being replaced by the succeeding population. Since *necrotic* cells cannot be replaced, their population size monotonically increases. By generation 115, *necrotic* cells have taken over the volume, except for the islands of *hypoxic* cells.

A.14 STABLE LOCAL HYPOXIA WITH MANY VESSELS (3D)

A.14.1 *Setup*

In this simulation, we have *viable*, *hypoxic*, and *necrotic* tumor cells that correspond to those we see in the anti-pimonidazole stain images. The *viable* and *hypoxic* cells consume O_2 at a certain rate, and release no particles. The conditional logic implements a simple state machine in the following way. Wherever O_2 falls below a threshold, *viable* cells become ("jump" to) *hypoxic* cells—identical in every way except as follows. Wherever O_2 rises above that threshold, *hypoxic* cells become *viable* again; and wherever O_2 falls below an even lower threshold, *hypoxic* cells become *necrotic* cells. Once a cell becomes *necrotic* it has entered an absorbing state and its behavior is completely inert: it has no consumption or release profile, and it is neither replaceable nor reproductive. The specific quantities mentioned here are given in the tables below.

A.14.1.1 *Configuration parameters*

O_2
0.12

Table 132: Diffusion rate of each particle type.

O_2
0.1

Table 133: Initial concentration of each particle type.

cell type	O_2
vessel	0
empty	0
viable	0
hypoxic	0
necrotic	0

Table 134: Basal lower-bound concentration of each particle type by cell type.

cell type	O_2
vessel	∞
empty	∞
viable	∞
hypoxic	∞
necrotic	∞

Table 135: Basal upper-bound concentration of each particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	0.01
hypoxic	0.01
necrotic	0

Table 136: Consumption rate of each particle type by cell type.

cell type	O_2
vessel	0.2
empty	0
viable	0
hypoxic	0
necrotic	0

Table 137: Release rate for of particle type by cell type.

cell type	O_2
vessel	0
empty	0
viable	1
hypoxic	1
necrotic	0

Table 138: Impact factor of each particle type upon each cell type.

cell type	*replaceable?*
vessel	No
empty	Yes
viable	No
hypoxic	No
necrotic	No

Table 139: Replaceable predicate of each cell type.

cell type	*reproductive?*
vessel	No
empty	Yes
viable	Yes
hypoxic	Yes
necrotic	No

Table 140: Reproductive predicate of each cell type.

cell type	*triggers*	*actions*
viable	$\rho_1 < 0.07$	jump to *hypoxic*
hypoxic	$\rho_1 < 0.05$	jump to *necrotic*
hypoxic	$\rho_1 > 0.07$	jump to *viable*

Table 141: Conditional triggers and actions for those cell types so configured.

A.14.1.2 *Initial conditions*

The simulation opens with the initial concentration of O_2 specified in Table 133. This diffuses with a rate specified in Table 132. We initialize the 3D lattice with *empty* cells; we place one *viable* cell in the center, and 100 vessels randomly about. Vessels consume no particles and release O_2 at the rate specified in Table 137.

A.14.2 *Results*

A.14.2.1 *Spatial cell population evolution*

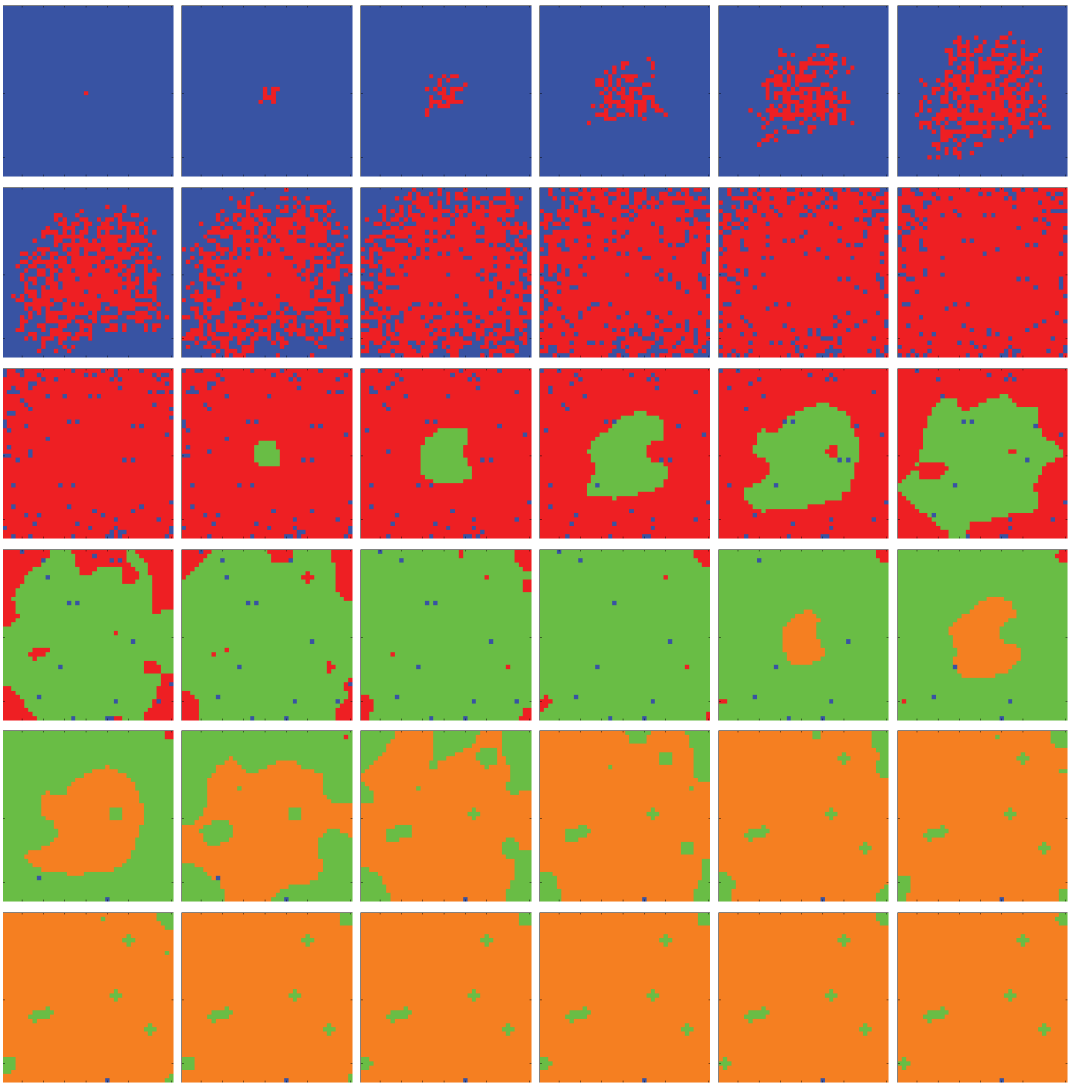

Figure 80: Time evolution of cell populations on the plane $z = 20$. Left-to-right, top-to-bottom: 150 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 74, 78, 83, 87, 92, 96, 101, 105, 110, 114, 119, 123, 128, 132, 137, 141, 146, 150). Key: *vessel* (white), *empty*, *viable*, *hypoxic*, *necrotic*.

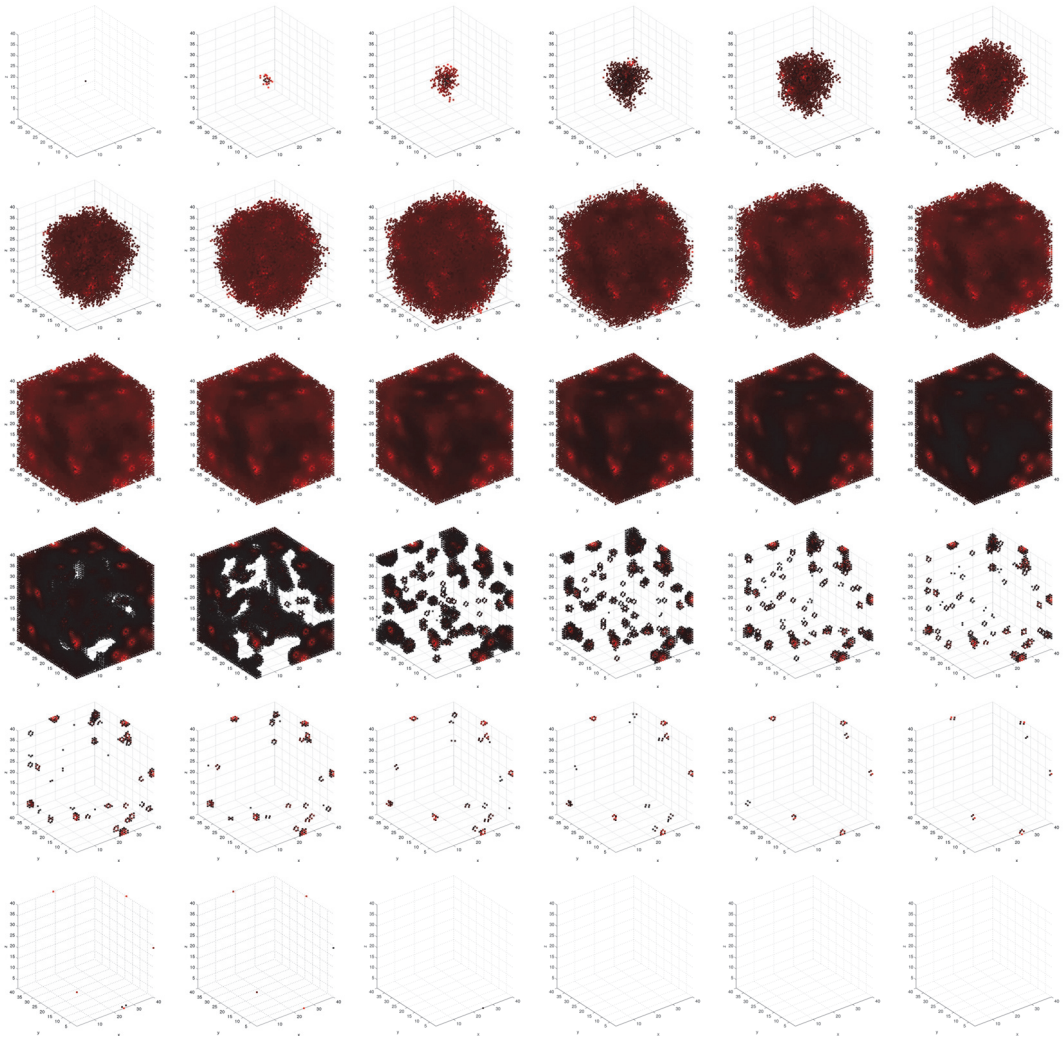

Figure 81: Time evolution of local fitness for *viable* cells. Left-to-right, top-to-bottom: 150 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 74, 78, 83, 87, 92, 96, 101, 105, 110, 114, 119, 123, 128, 132, 137, 141, 146, 150).

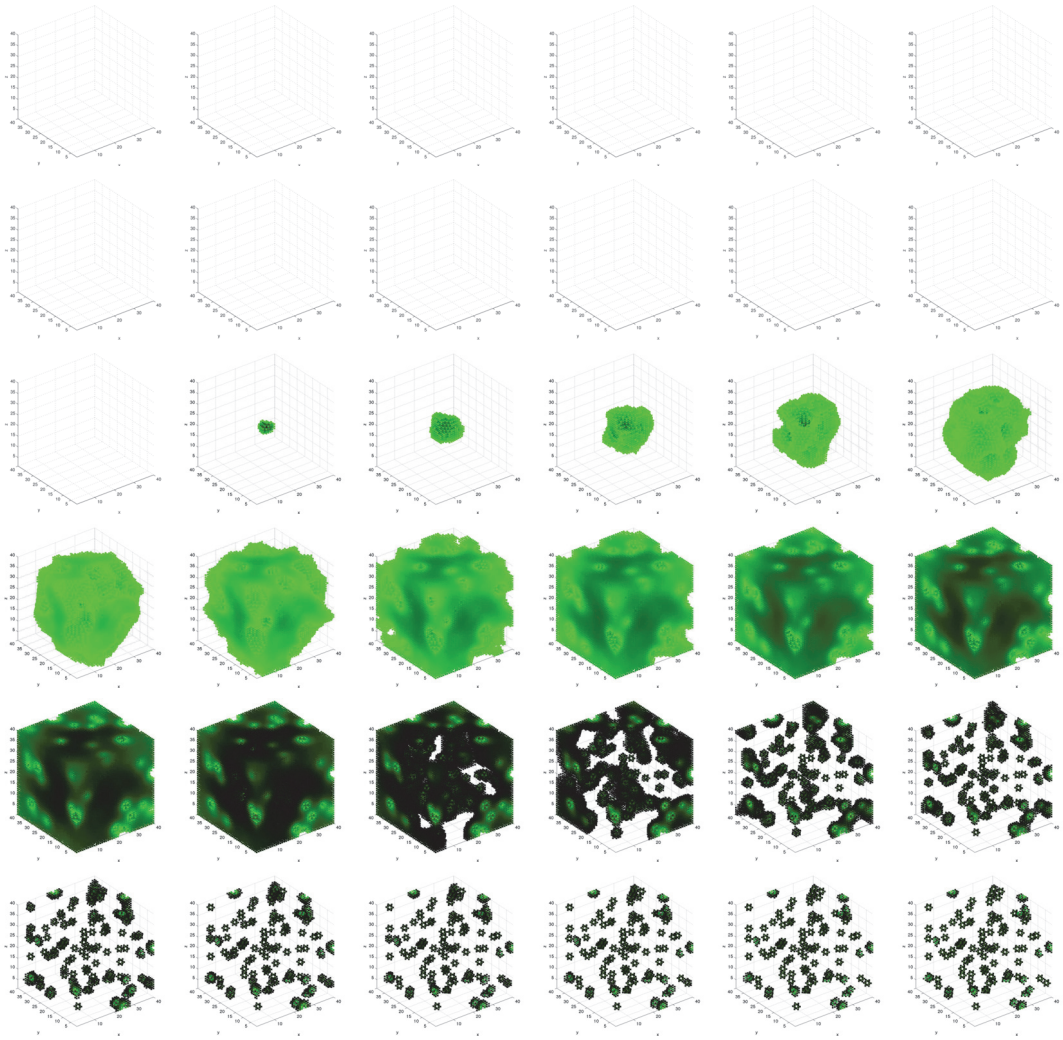

Figure 82: Time evolution of local fitness for *hypoxic* cells. Left-to-right, top-to-bottom: 150 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 74, 78, 83, 87, 92, 96, 101, 105, 110, 114, 119, 123, 128, 132, 137, 141, 146, 150).

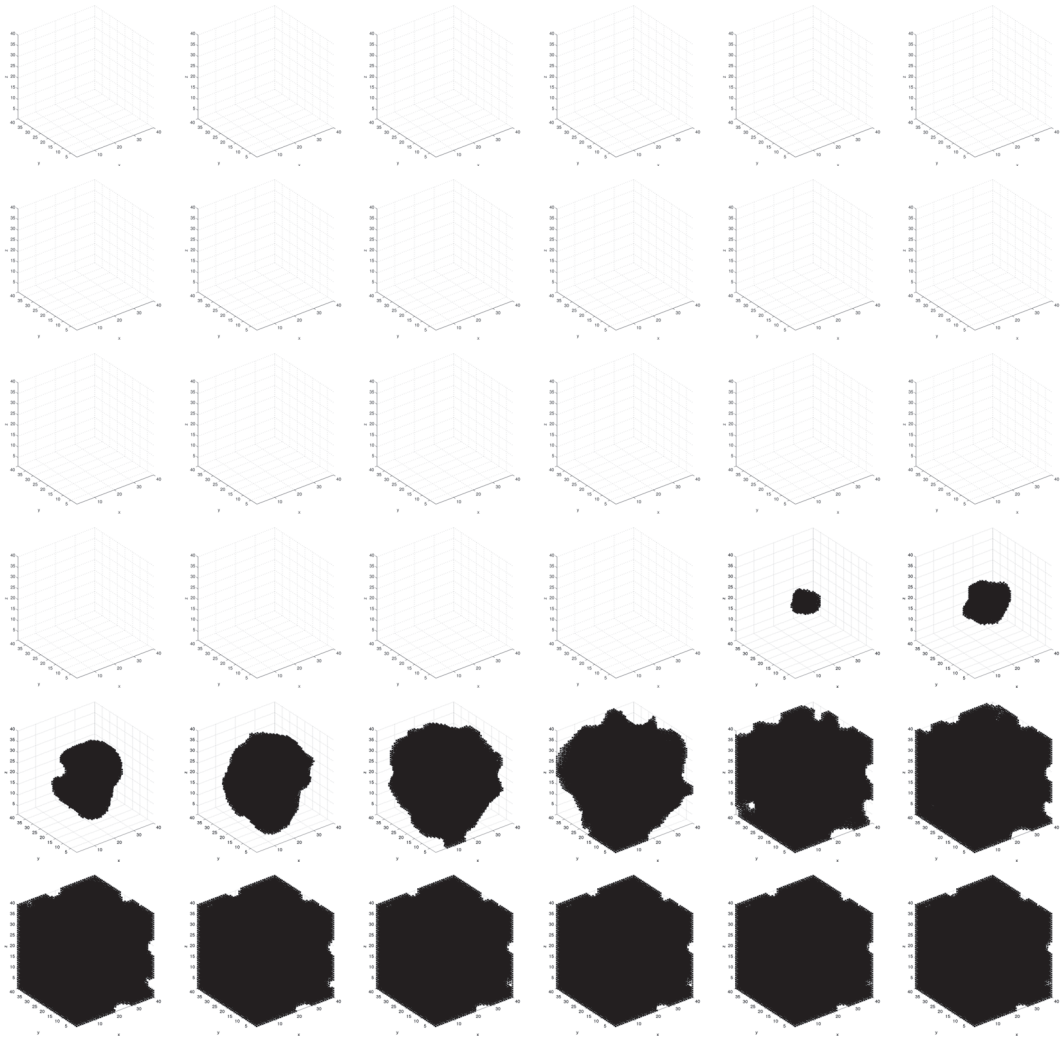

Figure 83: Time evolution of local fitness for *necrotic* cells. Left-to-right, top-to-bottom: 150 generations shown in 36 frames (t = 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61, 65, 69, 74, 78, 83, 87, 92, 96, 101, 105, 110, 114, 119, 123, 128, 132, 137, 141, 146, 150).

A.14.2.2 *Temporal cell population evolution*

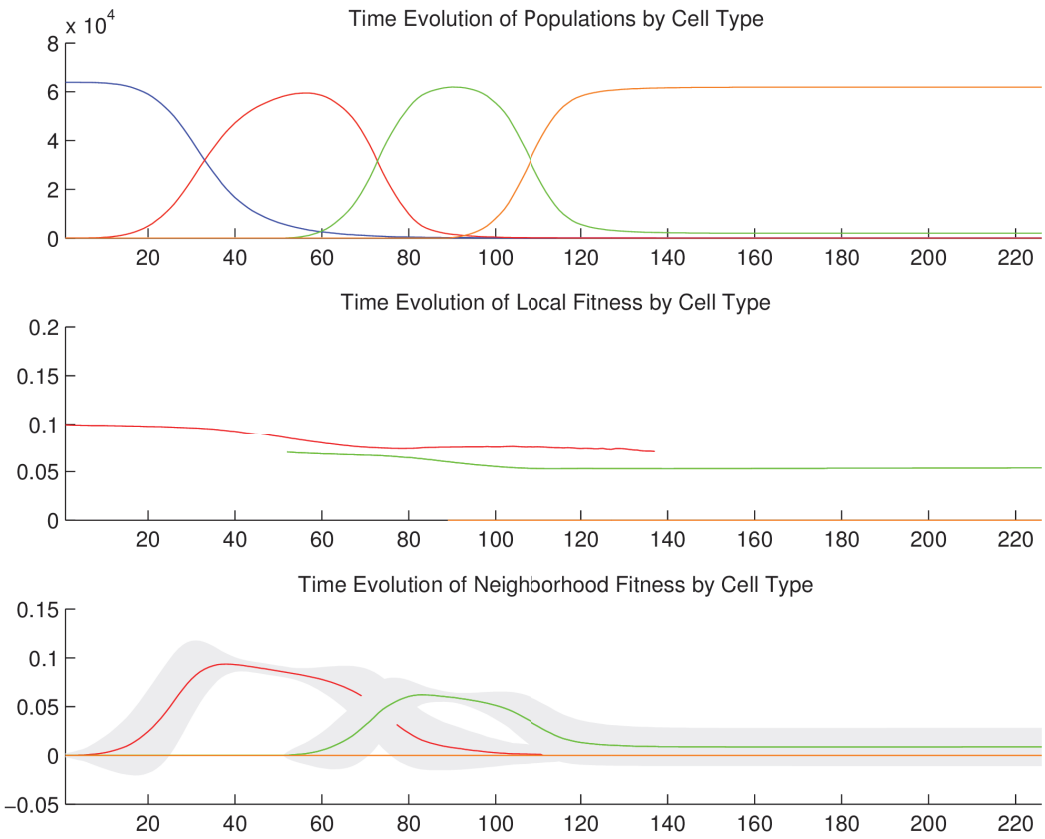

Figure 84: Time evolution of: populations by cell type (top), local fitness by cell type (middle), and neighborhood fitness by cell type (bottom). In the latter two, mean curves are plotted and gray regions above and below show the respective standard deviations. Key: *empty*, *viable*, *hypoxic*, *necrotic*.

A.14.3 *Discussion*

We observe the following sequence of events in Figure 80, Figure 81, Figure 82, and Figure 83. *Viable* cells proliferate radially outward from the center. Once most of the

volume is covered with *viable* cells, then *hypoxic* cells appear in the center. *Hypoxic* cells grow radially outward at a similar rate to the *viable* cells just beyond them. They approach vessels (surrounded by *viable* cells) up to some radial distance away from each vessel, the zone into which O_2 diffuses with a sufficient concentration to maintain *viable* cells. Once about half of the volume is covered with *hypoxic* cells, then *necrotic* cells appear in the center. *Necrotic* cells grow radially outward at a similar rate to the *hypoxic* and *viable* cells just beyond them. They approach vessels (surrounded by concentrically situated *viable* and *hypoxic* cells) up to some radial distance away from each vessel, the zone into which O_2 diffuses with a sufficient concentration to maintain *viable* and *hypoxic* cells. We eventually observe islands of concentrically situated *viable* and *hypoxic* cells, surrounded by a sea of *necrotic* cells, similar to what we see in the anti-pimonidazole stain images. This arrangement is stable for awhile, before the remaining *viable* cells become *hypoxic* cells. These *hypoxic* cell islands persist indefinitely. We observe the succession of *viable* $\rightarrow$ *hypoxic* $\rightarrow$ *necrotic* cell populations in Figure 84. The initial decay of *viable* and *hypoxic* cell populations is due to their proliferating outside of the spatial dimensions of the simulation while simultaneously being replaced by the succeeding population. Since *necrotic* cells cannot be replaced, their population size monotonically increases. By generation 125, *necrotic* cells have taken over the volume, except for the islands of *hypoxic* cells.

B

SIMULATOR CODE

Below is the Matlab code for the simulator, configured for stable local hypoxia with many vessels (3D).

```
function [] = simulator()

%% declare parameters and data structures

% global parameters
s_dim               = 40;     % symmetric dimension (used for convenience, when
    x, y, z dims are identical)
x_dim               = s_dim;  % x dimension
y_dim               = s_dim;  % y dimension
z_dim               = s_dim;  % z dimension
num_cell_types      = 5;      % number of cell types
num_particle_types  = 1;      % number of particle types
num_iters           = 1000;   % number of simulation clock ticks
d_tau               = 1;      % time scale separation parameter
plot_every          = 1;      % number of clock ticks between plottings
reproduce_every     = 1;      % number of clock ticks between probabilistic
    reproductions
delay_occupation_by = 0;      % number of clock ticks before initializing cell
    type occupations (used to establish gradients prior to exposing cells)
plot_3d             = 1;      % predicate for plotting 3D occupation per cell
    type
```

```
% output predicates
output_results                = 1;        % predicate for outputting plots (
    global switch for those below)
output_cell_types             = 1;        % predicate for outputting cell types
output_time_series            = 1;        % predicate for outputting time series
output_local_fitness_3d       = 1;        % predicate for outputting local
    fitness in 3D
output_local_fitness          = 1;        % predicate for outputting local
    fitness
output_neighborhood_fitness   = 0;        % predicate for outputting
    neighborhood fitness
output_particle_concentration = 0;        % predicate for outputting particle
    concentration
output_only_cell_types        = [3 4 5];  % subset of cell types to output
output_only_particle_types    = [1];      % subset of particle types to output

% diffusion rate of each particle type
diffusion_rate      = [0.1];

% initial concentration of each particle type
init_concentration  = [0.1];

% basal lower bound of each particle type for each cell type
basal_lower         = [0;   % vessel
                       0;   % empty
                       0;   % alpha
                       0;   % beta
                       0];  % gamma
```

```
% basal upper bound of each particle type for each cell type
basal_upper         = [Inf;   % vessel
                       Inf;   % empty
                       Inf;   % alpha
                       Inf;   % beta
                       Inf];  % gamma

% consume rate of each particle type for each cell type
consume_rate        = [0;      % vessel
                       0;      % empty
                       0.01;   % alpha
                       0.01;   % beta
                       0];     % gamma

% release rate of each particle type for each cell type
release_rate        = [0.2; % vessel
                       0;   % empty
                       0;   % alpha
                       0;   % beta
                       0];  % gamma

% impact factor of each particle type for each cell type
impact_factor       = [0;   % vessel
                       0;   % empty
                       1;   % alpha
                       1;   % beta
                       0];  % gamma
```

```
% replacement status of each cell type
replaceable         = [0;   % vessel
                       1;   % empty
                       0;   % alpha
                       0;   % beta
                       0];  % gamma

% reproductive status of each cell type
reproductive        = [0;   % vessel
                       1;   % empty
                       1;   % alpha
                       1;   % beta
                       0];  % gamma

% color of each cell type
cell_type_color_map = [1    1    1;   % vessel (white)
                       0    0    1;   % empty  (blue)
                       1    0    0;   % alpha  (red)
                       0    1    0;   % beta   (green)
                       1    0.5  0];  % gamma  (orange)

% declare conditional triggers and actions:
%     trigger = { <particle type> <comparative operator = {<,=,>}> <value> }
%     action  = { <operator = {a,j,-,+,c,r,i}> <operand 1 = {jump cell type,
    boolean target value, particle type}> <operand 2 = {particle concentration,
    impact factor}> }
% note:
%     operator:  number of operands:
%     {a}        0
```

```
%     {j,-,+}    1
%     {c,r,i}    2
% example conditional triggers and actions:
%     conditionals(n) = { { { {t1} {t2} {t3} } { {a1} {a2}      } }
%                         { { {t1} {t2}      } { {a1} {a2} {a3} } }
%                       };
conditionals    = containers.Map('KeyType', 'int32', 'ValueType', 'any');
conditionals(3) = {
                    { { {1 '<' 0.07} } { {'j' 4} } }
                  };  % alpha
conditionals(4) = {
                    { { {1 '<' 0.05} } { {'j' 5} } }
                    { { {1 '>' 0.07} } { {'j' 3} } }
                  };  % beta

% define essential simulation arrays
occupation          = zeros(x_dim, y_dim, z_dim);                    % cell
    types
concentration       = zeros(x_dim, y_dim, z_dim, num_particle_types);  %
    particle concentrations
impact_factor_cond  = zeros(x_dim, y_dim, z_dim, num_particle_types);  % impact
    factor of each particle type for each cell
replaceable_cond    = zeros(x_dim, y_dim, z_dim);                    %
    replacement status of each cell
reproductive_cond   = zeros(x_dim, y_dim, z_dim);                    %
    reproductive status of each cell
fitness_1           = zeros(x_dim, y_dim, z_dim);                    % local
    fitness (used in conjunction with occupation; each coordinate has one local
    fitness value)
```

```
fitness_n           = zeros(x_dim, y_dim, z_dim, num_cell_types);      %
    neighborhood fitness (each coordinate has a neighborhood fitness value for
    each cell type)

% define statistics arrays
population          = zeros(num_cell_types, num_iters);  % population of each
    cell type
fitness_1_avg       = zeros(num_cell_types, num_iters);  % mean local fitness
    for each cell type
fitness_1_std       = zeros(num_cell_types, num_iters);  % standard deviation
    local fitness for each cell type
fitness_n_avg       = zeros(num_cell_types, num_iters);  % mean neighborhood
    fitness for each cell type
fitness_n_std       = zeros(num_cell_types, num_iters);  % standard deviation
    neighborhood fitness for each cell type
diameter_x          = zeros(num_cell_types, num_iters);  % global x-extent for
    each cell type
diameter_y          = zeros(num_cell_types, num_iters);  % global y-extent for
    each cell type
diameter_z          = zeros(num_cell_types, num_iters);  % global z-extent for
    each cell type
epc                 = zeros(num_cell_types, num_iters);  % global Euler-Poincare
     characteristic for each cell type
wallclock           = zeros(num_iters);                  % wallclock time for
    each tick's processing


%% set initial conditions


% define some useful landmarks
```

```
mid_x     = floor(x_dim / 2);
mid_y     = floor(y_dim / 2);
mid_z     = floor(z_dim / 2) + 1;
mid_mid_x = floor(x_dim / 4);
mid_mid_y = floor(y_dim / 4);
mid_mid_z = floor(z_dim / 4) + 1;

% initialize particle concentrations by particle type
for pt = 1 : num_particle_types
    concentration(:,:,:,pt) = init_concentration(pt);
end

% initialize the cell type occupations
occupation_delay_elapsed = false;
if ~delay_occupation_by
    initialize_occupation;
    occupation_delay_elapsed = true;
end

%% setup for outputting results

if output_results

    % create timestamp
    [year, month, day, hour, minute, second] = datevec(now);
    years   = num2str(year);
    months  = num2str(month);
    days    = num2str(day);
    hours   = num2str(hour);
```

```
    minutes = num2str(minute);
    seconds = num2str(floor(second));
    if length(months) == 1
        months = strcat('0', months);
    end
    if length(days) == 1
        days = strcat('0', days);
    end
    if length(hours) == 1
        hours = strcat('0', hours);
    end
    if length(minutes) == 1
        minutes = strcat('0', minutes);
    end
    if length(seconds) == 1
        seconds = strcat('0', seconds);
    end
    ts = sprintf('%s_%s_%s__%s_%s_%s', years, months, days, hours, minutes,
        seconds);

    % create new results directory and copy this code into it
    path = strcat('/tmp/simulations/', ts);
    mkdir(path);
    copyfile('gd.m', path);

end

%% simulation loop
```

```
for tick = 1 : num_iters

    %% prologue

    % set the timer for this tick
    tic;

    % set the time scale separation parameter
    tau = tick * d_tau;

    % initialize once the cell type occupations after the delay
    if tick > delay_occupation_by && ~occupation_delay_elapsed
        initialize_occupation;
        occupation_delay_elapsed = true;
    end

    % initialize the conditional matrices after the delay
    if tick > delay_occupation_by
        impact_factor_cond = reshape(impact_factor(occupation,:), x_dim, y_dim,
            z_dim, num_particle_types);
        replaceable_cond   = replaceable(occupation);
        reproductive_cond  = reproductive(occupation);
    end

    %% phase I: consume and release particles

    % if past the occupation delay
    if tick > delay_occupation_by
```

```
% DEFAULT: consume and release particles according to the cells'
    respective consumption and release rates
for pt = 1 : num_particle_types
    concentration_pt = concentration(:,:,:,pt);
    for ct = 1 : num_cell_types
        concentration_pt(occupation == ct) = concentration_pt(occupation
            == ct) * (1 - consume_rate(ct,pt) + release_rate(ct,pt));
    end
    concentration(:,:,:,pt) = concentration_pt;
end

% CONDITIONAL: for each cell type defined in the conditionals
for cond_key = conditionals.keys

    % define the cell type from the conditional key
    ct = cond_key{1};

    % fetch the conditional block
    cond = conditionals(ct);

    % for each trigger-action defined for this cell type
    for ta = 1 : numel(cond)

        % parse out the trigger set and the action set
        triggers = cond{ta}{1};  % trigger set
        actions  = cond{ta}{2};  % action set

        % declare a result matrix for this trigger set
        ts_result = ones(x_dim, y_dim, z_dim);
```

```
% for each trigger in the trigger set
for t = 1 : numel(triggers)

    % fetch the trigger out of the trigger set
    trigger = triggers{t};

    % parse out the trigger elements
    trigger_pt = trigger{1};  % this trigger's particle type
    trigger_op = trigger{2};  % this trigger's operator
    trigger_pc = trigger{3};  % this trigger's particle
        concentration

    % obtain the appropriate concentrations for this trigger's
        particle type
    concentration_trigger_pt = concentration(:,:,:,trigger_pt);

    % determine the results of this trigger based on this
        trigger's operator
    switch trigger_op
        case '<'
            ts_result = ts_result & (occupation == ct) & (
                concentration_trigger_pt <  trigger_pc);
        case '>'
            ts_result = ts_result & (occupation == ct) & (
                concentration_trigger_pt >  trigger_pc);
        case '='
            ts_result = ts_result & (occupation == ct) & (
                concentration_trigger_pt == trigger_pc);
```

```
        end

    end  % for each trigger in the trigger set

% for each action defined for this trigger set
for a = 1 : numel(actions)

    % fetch the action out of the action set
    action = actions{a};

    % parse out the action elements
    action_op = action{1};  % this action's operator
    if ~isequal(action_op, 'a')
        action_o1 = action{2};  % this action's operand 1 (e.g.,
             boolean target value, particle type, jump cell type
            )
    end
    if ~isequal(action_op, 'a') && ~isequal(action_op, 'j') && ~
        isequal(action_op, '-') && ~isequal(action_op, '+')
        action_o2 = action{3};  % this action's operand 2 (e.g.,
             particle concentration)
    end

    % for the trigger set result cells, perform the action
    switch action_op
        case 'a'
            occupation(ts_result) = 2;
            for pt = 1 : num_particle_types
```

```
                impact_factor_cond_pt = impact_factor_cond
                    (:,:,:,pt);
                impact_factor_cond_pt(ts_result) = impact_factor
                    (2,pt);
                impact_factor_cond(:,:,:,pt) = impact_factor_
                    cond_pt;
            end
            replaceable_cond(ts_result) = replaceable(2);
            reproductive_cond(ts_result) = reproductive(2);
        case 'j'
            occupation(ts_result) = action_o1;
            for pt = 1 : num_particle_types
                impact_factor_cond_pt = impact_factor_cond
                    (:,:,:,pt);
                impact_factor_cond_pt(ts_result) = impact_factor
                    (action_o1,pt);
                impact_factor_cond(:,:,:,pt) = impact_factor_
                    cond_pt;
            end
            replaceable_cond(ts_result) = replaceable(action_o1)
                ;
            reproductive_cond(ts_result) = reproductive(action_o
                1);
        case '-'
            replaceable_cond(ts_result) = action_o1;
        case '+'
            reproductive_cond(ts_result) = action_o1;
        case 'c'
            concentration_o1 = concentration(:,:,:,action_o1);
```

```
                        concentration_o1(ts_result) = concentration_o1(ts_
                            result) * (1 - action_o2);
                        concentration(:,:,:,action_o1) = concentration_o1;
                    case 'r'
                        concentration_o1 = concentration(:,:,:,action_o1);
                        concentration_o1(ts_result) = concentration_o1(ts_
                            result) * (1 + action_o2);
                        concentration(:,:,:,action_o1) = concentration_o1;
                    case 'i'
                        impact_factor_cond_o1 = impact_factor_cond(:,:,:,
                            action_o1);
                        impact_factor_cond_o1(ts_result) = action_o2;
                        impact_factor_cond(:,:,:,action_o1) = impact_factor_
                            cond_o1;
                end

            end  % for each action defined for this trigger set

        end  % for each trigger-action defined for this cell type

    end  % for each cell type defined in the conditionals

end  % if tick > delay_occupation

%% phase II: diffuse particles

% diffuse particles according to the particles' respective diffusion rates
for pt = 1 : num_particle_types
    sigma = sqrt(2 * diffusion_rate(pt));
```

```
        if z_dim > 1
            G = gauss3([5 5 5], [sigma sigma sigma]);
        else
            G = fspecial('gaussian', [5 5], sigma);
        end
        concentration(:,:,:,pt) = imfilter(concentration(:,:,:,pt), G, '
            replicate', 'same', 'conv');
    end

    % restore concentrations to within their basal bounds
    for pt = 1 : num_particle_types
        concentration_pt = concentration(:,:,:,pt);
        for ct = 1 : num_cell_types
            bl = basal_lower(ct, pt);
            bu = basal_upper(ct, pt);
            concentration_pt((occupation == ct) & (concentration_pt < bl)) = bl;
            concentration_pt((occupation == ct) & (concentration_pt > bu)) = bu;
        end
        concentration(:,:,:,pt) = concentration_pt;
    end

    %% phase III: compute cell fitness

    % if past the occupation delay
    if tick > delay_occupation_by

        % compute local fitness of each voxel
        fitness_1 = zeros(x_dim, y_dim, z_dim);
        for i = 1 : x_dim
```

```
        for j = 1 : y_dim
371         for k = 1 : z_dim

                % get the cell type at this voxel
                cell_type = occupation(i,j,k);

376             % skip vessel and empty cell types, since these cannot
                    obtain a local fitness
                if cell_type == 1 || cell_type == 2
                    continue;
                end

381             % comute local fitness based on particle type concentrations
                     and their respective impact factors
                impact_factors = reshape(impact_factor_cond(i,j,k,:), 1, num
                    _particle_types);
                concentrations = reshape(concentration(i,j,k,:), num_
                    particle_types, 1);
                fitness_1(i,j,k) = impact_factors * concentrations;

386         end
        end
    end

    % compute neighborhood fitness of each voxel
391 fitness_n = zeros(x_dim, y_dim, z_dim, num_cell_types);
    for i = 1 : x_dim
        for j = 1 : y_dim
            for k = 1 : z_dim
```

```
% get the cell type at this voxel
cell_type = occupation(i,j,k);

% skip vessel cell type, since they cannot be targeted for a
    new cell type (but empty cells still can)
if cell_type == 1
    continue;
end

% compute neighborhood fitness; loop from cell type 3 (post-
    vessel, post-empty) to the last type
for ct = 3 : num_cell_types
    num_neighbors = 0;
    sum_neighbors = 0;
    for ii = i-1 : i+1
        if ii < 1 || ii > x_dim
            continue;
        end
        for jj = j-1 : j+1
            if jj < 1 || jj > y_dim
                continue;
            end
            for kk = k-1 : k+1
                if kk < 1 || kk > z_dim
                    continue;
                end
                if occupation(ii,jj,kk) ~= ct
                    continue;
```

```
                                    end
                                    num_neighbors = num_neighbors + 1;
                                    sum_neighbors = sum_neighbors + fitness_1(ii
                                        ,jj,kk);
                                end  % for kk
                            end  % for jj
                        end  % for ii
                        if num_neighbors == 0
                            fitness_n(i,j,k,ct) = 0;
                        else
                            fitness_n(i,j,k,ct) = sum_neighbors / num_neighbors;
                        end
                    end  % for ct

                end  % for k
            end  % for j
        end  % for i

    end  % if tick > delay_occupation

    %% interlude: compute and plot statistics

    % compute statistics
    for ct = 1 : num_cell_types

        % occupations for this cell type
        occupation_ct = occupation == ct;

        % population of this cell type
```

```
    population(ct, tick) = sum(sum(sum(occupation_ct)));

    % local fitness of this cell type
    fitness_1_avg(ct, tick) = mean(fitness_1(occupation_ct));
    fitness_1_std(ct, tick) = std(fitness_1(occupation_ct));

    % neighborhood fitness of this cell type
    fitness_n_ct            = fitness_n(:,:,:,ct);
    fitness_n_avg(ct, tick) = mean(fitness_n_ct(occupation > 1));
    fitness_n_std(ct, tick) = std(fitness_n_ct(occupation > 1));

    % x,y,z-extent for this cell type
    [cti, ctj, ctk] = ind2sub(size(occupation_ct), find(occupation_ct));
    if numel(cti) == 0
        diameter_x(ct, tick) = 0;
    else
        diameter_x(ct, tick) = max(cti) - min(cti) + 1;
    end
    if numel(ctj) == 0
        diameter_y(ct, tick) = 0;
    else
        diameter_y(ct, tick) = max(ctj) - min(ctj) + 1;
    end
    if numel(ctk) == 0
        diameter_z(ct, tick) = 0;
    else
        diameter_z(ct, tick) = max(ctk) - min(ctk) + 1;
    end
```

```
        % Euler-Poincare characteristic for this cell type
        if z_dim > 1
            epc(ct, tick) = imEuler3d(occupation_ct);
        else
            epc(ct, tick) = imEuler2d(occupation_ct);
        end

    end

    % plot statistics
    if ~mod(tick, plot_every)
        plot_statistics();
    end

    %% phase IV: reproduce cells probabilistically

    % if past the occupation delay
    if tick > delay_occupation_by

        % if it's time to reproduce
        if ~mod(tick, reproduce_every)

            % probabilistically determine the fate of each voxel
            for i = 1 : x_dim
                for j = 1 : y_dim
                    for k = 1 : z_dim

                        % if this cell type is not replaceable, then skip it
                        if ~replaceable_cond(i,j,k)
```

```
        continue;
    end

    % declare a probability vector; the first two elements (
        representing vessel and empty cell types) won't be
        used
    probabilities = zeros(1, num_cell_types);

    % obtain the probabilities from the neighborhood fitness
         values; loop from cell type 3 (post-vessel, post-
        empty) to the last type
    for ct = 3 : num_cell_types
        probabilities(ct) = fitness_n(i,j,k,ct);
    end

    % sum the probabilities
    sum_probabilities = sum(probabilities);

    % if the sum of probabilities is greater than one, then
        normalize the probabilities to one
    shrinkage_factor = 1;
    if sum_probabilities > 1
        shrinkage_factor = 1 / sum_probabilities;
    end
    probabilities = shrinkage_factor * probabilities;

    % partition the [0,1] interval by the set of
        probabilities, contiguously placed along the
        interval, and randomly point into the interval to
```

```
    select the target cell type for this cell; loop from
     cell type 3 (post-vessel, post-empty) to the last
    type
essay = rand;
beg_num = 0;
end_num = 0;
target_cell_type = 0;
for ct = 3 : num_cell_types
    beg_num = end_num;
    end_num = end_num + probabilities(ct);
    if essay >= beg_num && essay < end_num
        target_cell_type = ct;
        break;
    end
end

% if the random point lies beyond the last cell type's
    probability range, then it must become an empty cell
if target_cell_type == 0
    target_cell_type = 2;
end

% if any of the neighboring cells are of the target type
     and are reproductive, then mutate the current cell
    type to the target cell type
neighbor_is_of_target_cell_type_and_reproductive = false
    ;
for ii = i-1 : i+1
    if ii < 1 || ii > x_dim
```

```
                            continue;
                        end
                        for jj = j-1 : j+1
                            if jj < 1 || jj > y_dim
                                continue;
                            end
                            for kk = k-1 : k+1
                                if kk < 1 || kk > z_dim
                                    continue;
                                end
                                if (occupation(ii,jj,kk) == target_cell_type
                                    ) && reproductive_cond(ii,jj,kk)
                                    neighbor_is_of_target_cell_type_and_
                                        reproductive = true;
                                end
                            end  % for kk
                        end  % for jj
                    end  % for ii
                    if neighbor_is_of_target_cell_type_and_reproductive
                        occupation(i,j,k) = target_cell_type;
                    end

                end  % for k
            end  % for j
        end  % for i

    end  % reproduce every

end  % if tick > delay_occupation
```

```
    %% epilogue

    % record the timer for this tick
    wallclock(tick) = toc;

end  % for tick

    %% plot the statistics of interest
    function [] = plot_statistics()

        % define some useful constants
        gray_256    = grade([1 1 1], 256);
        num_rows    = 6;
        num_ct_cols = (num_cell_types - 2) + 1;  % no columns for vessel and
            empty cell types + 1 column for corresponding time series
        num_cols    = max(num_ct_cols, num_particle_types);
        del_cols    = num_cols - num_ct_cols;
        ct_range    = 3 : num_cell_types;
        pt_range    = 1 : num_particle_types;
        cursor      = 1;

        % set up the figure
        figure(1);
        clf;

        % cell types (2-D slice)
        subplot(num_rows, num_cols, cursor);
        imagesc(occupation(:,:,mid_z), [1 size(cell_type_color_map,1)]);
```

```
        colormap(cell_type_color_map);
        freezeColors;
        set(gca, 'YDir', 'normal');
        axis square;
611     cursor = cursor + 1;

        % performance (time series)
        subplot(num_rows, num_cols, cursor);
        domain = 1:tick-1;
616     wc     = wallclock(domain);
        wc_avg = mean(wc);
        wc_std = std(wc);
        plot(domain, wc);
        freezeColors;
621     title(['avg=' sprintf('%3.3f', wc_avg) ', std=' sprintf('%3.3f', wc_std)
            ]);
        axis square;
        cursor = cursor + 1;

        % pad, if necessary
626     pad_cursor();

        % cell type 3D plot
        for ct = ct_range
            if plot_3d
631             subplot(num_rows, num_cols, cursor);
                occ_ct = occupation == ct;
                fit_ct = fitness_1(occ_ct);
                min_fit_ct = min(fit_ct);
```

```
max_fit_ct = max(fit_ct);
if min_fit_ct < max_fit_ct
    norm_fit_ct = (fit_ct - min_fit_ct) ./ (max_fit_ct - min_fit
        _ct);
else
    norm_fit_ct = 0;
end
color_density = 256;
color_grade = grade(cell_type_color_map(ct,:), color_density);
color_idx = floor((color_density - 1) * norm_fit_ct) + 1;
scatter_colors = color_grade(color_idx,:);
[x,y,z] = ind2sub(size(occ_ct), find(occ_ct));
% note the axis order for plotting: <y,x,z>
scatter3(y, x, z, 20, scatter_colors, 'filled', 's');
colormap(grade(cell_type_color_map(ct,:), color_density));
ch = colorbar;
if min_fit_ct < max_fit_ct
    caxis([min_fit_ct max_fit_ct]);
end
cbfreeze(ch);
xlabel('x');
ylabel('y');
zlabel('z');
if z_dim > 1
    axis([1 x_dim 1 y_dim 1 z_dim]);
else
    axis([1 x_dim 1 y_dim]);
end
axis square;
```

```
        end
        cursor = cursor + 1;
    end

    % population by cell type (time series)
    subplot(num_rows, num_cols, cursor);
    hold on;
    for ct = 1 : num_cell_types
        plot(1:tick, population(ct, 1:tick), 'color', cell_type_color_map(ct
            ,:));
    end
    hold off;
    freezeColors;
    if tick > 1
        xlim([1 tick]);
    end
    axis square;
    cursor = cursor + 1;

    % pad, if necessary
    pad_cursor();

    % cell type local fitness (2-D slice)
    for ct = ct_range
        subplot(num_rows, num_cols, cursor);
        imagesc((occupation(:,:,mid_z) == ct) .* fitness_1(:,:,mid_z));
        colormap(grade(cell_type_color_map(ct,:), 256));
        cbfreeze(colorbar);
        freezeColors;
```

```
691             set(gca, 'YDir', 'normal');
                axis square;
                cursor = cursor + 1;
            end

696         % local fitness by cell type (time series)
            subplot(num_rows, num_cols, cursor);
            plot_time_series(fitness_1_avg(ct_range,:), fitness_1_std(ct_range,:),
                cell_type_color_map(ct_range,:), 1, tick);
            axis square;
            cursor = cursor + 1;
701
            % pad, if necessary
            pad_cursor();

            % cell type neighborhood fitness (2-D slice)
706         for ct = ct_range
                subplot(num_rows, num_cols, cursor);
                imagesc((occupation(:,:,mid_z) > 1) .* fitness_n(:,:,mid_z,ct));
                colormap(grade(cell_type_color_map(ct,:), 256));
                cbfreeze(colorbar);
711             freezeColors;
                set(gca, 'YDir', 'normal');
                axis square;
                cursor = cursor + 1;
            end
716
            % neighborhood fitness by cell type (time series)
            subplot(num_rows, num_cols, cursor);
```

```
        plot_time_series(fitness_n_avg(ct_range,:), fitness_n_std(ct_range,:),
            cell_type_color_map(ct_range,:), 1, tick);
        axis square;
721     cursor = cursor + 1;

        % pad, if necessary
        pad_cursor();

726     % cell type x,y,z-extent
        for ct = ct_range
            subplot(num_rows, num_cols, cursor);
            hold on;
            % note the axis order for plotting: <y,x,z>
731         plot(1:tick, diameter_y(ct, 1:tick), 'r');
            plot(1:tick, diameter_x(ct, 1:tick), 'g');
            plot(1:tick, diameter_z(ct, 1:tick), 'b');
            hold off;
            if tick > 1
736             xlim([1 tick]);
            end
            axis square;
            cursor = cursor + 1;
        end
741
        % Euler-Poincare characteristic by cell type (time series)
        subplot(num_rows, num_cols, cursor);
        hold on;
        for ct = 1 : num_cell_types
746         plot(1:tick, epc(ct, 1:tick), 'color', cell_type_color_map(ct,:));
```

```
        end
        hold off;
        freezeColors;
        axis square;
        cursor = cursor + 1;

        % pad, if necessary
        pad_cursor();

        % particle type concentrations (2-D slice)
        for pt = pt_range
            subplot(num_rows, num_cols, cursor);
            con_pt = concentration(:,:,mid_z,pt);
            min_con_pt = min(min(con_pt));
            max_con_pt = max(max(con_pt));
            mesh(con_pt);
            colormap(gray_256);
            min_rel = '=';
            if min_con_pt > 1000
                min_rel = '>';
                min_con_pt = 1000;
            end
            max_rel = '=';
            if max_con_pt > 1000
                max_rel = '>';
                max_con_pt = 1000;
            end
            title(['min' min_rel sprintf('%3.3f', min_con_pt) ', max' max_rel
                sprintf('%3.3f', max_con_pt)]);
```

```
            axis square;
            xlim([1 x_dim]);
            ylim([1 y_dim]);
            cursor = cursor + 1;
        end

        % if outputting results, then create and save a stand-alone figure
        if output_results
            if output_cell_types
                save_cell_types();
            end
            if output_time_series
                save_time_series(ct_range);
            end
            if output_local_fitness_3d
                save_local_fitness_3d(ct_range);
            end
            if output_local_fitness
                save_local_fitness(ct_range);
            end
            if output_neighborhood_fitness
                save_neighborhood_fitness(ct_range);
            end
            if output_particle_concentration
                save_particle_concentration(gray_256, pt_range)
            end
        end

        function [rc] = row_cursor(cursor, num_cols)
```

```
            rc = mod(cursor, num_cols);
            if rc == 0
806             rc = num_cols;
            end
        end  % function row_cursor

        function [] = pad_cursor()
811         for p = 1 : (num_cols - row_cursor(cursor - 1, num_cols))
                cursor = cursor + 1;
            end
        end  % function pad_cursor

816 end  % function plot_statistics

    %% plot cell types and save them to disk
    function [] = save_cell_types()

821     % set up an invisible figure
        h = figure('Visible', 'off');

        % cell types (2-D slice)
        imagesc(occupation(:,:,mid_z), [1 size(cell_type_color_map,1)]);
826     colormap(cell_type_color_map);
        freezeColors;
        set(gca, 'YDir', 'normal');
        axis square;
        title({'Cell Population' ['t=' num2str(tick)]});
831
        % create figure file name and save figure
```

```
        len_num_iters = length(num2str(num_iters));
        len_tick = length(num2str(tick));
        len_pad = len_num_iters - len_tick;
        pad = '';
        for pi = 1 : len_pad
            pad = strcat('0', pad);
        end
        fig_file = strcat(path, '/', 'cells_', pad, num2str(tick));
        saveas(gcf, fig_file, 'pdf');

        % delete figure
        close(h);

    end  % function save_cell_types

    %% plot certain time series and save them to disk
    function [] = save_time_series(ct_range)

        % set up an invisible figure
        h = figure('Visible', 'off');

        % population by cell type (time series)
        subplot(3,1,1);
        hold on;
        for ct = 1 : num_cell_types
            plot(1:tick, population(ct, 1:tick), 'color', cell_type_color_map(ct
                ,:));
        end
        hold off;
```

```
        if tick > 1
            xlim([1 tick]);
        end
        title('Time Evolution of Populations by Cell Type');

        % local fitness by cell type (time series)
        subplot(3,1,2);
        plot_time_series(fitness_1_avg(ct_range,:), fitness_1_std(ct_range,:),
            cell_type_color_map(ct_range,:), 1, tick);
        title('Time Evolution of Local Fitness by Cell Type');

        % neighborhood fitness by cell type (time series)
        subplot(3,1,3);
        plot_time_series(fitness_n_avg(ct_range,:), fitness_n_std(ct_range,:),
            cell_type_color_map(ct_range,:), 1, tick);
        title('Time Evolution of Neighborhood Fitness by Cell Type');

        % create figure file name and save figure
        fig_file = strcat(path, '/', 'time_series');
        saveas(h, fig_file, 'fig');
        saveas(h, fig_file, 'pdf');

        % delete figure
        close(h);

    end  % function save_time_series

    %% plot local fitness 3D and save them to disk
    function [] = save_local_fitness_3d(ct_range)
```

```
        % cell type 3D plot
        for ct = ct_range

            % process only designated cell types
            if ~numel(find(output_only_cell_types == ct))
                continue;
            end

            % set up an invisible figure
            h = figure('Visible', 'off');

            % plot the image
            occ_ct = occupation == ct;
            fit_ct = fitness_1(occ_ct);
            min_fit_ct = min(fit_ct);
            max_fit_ct = max(fit_ct);
            norm_fit_ct = fit_ct;
            if min_fit_ct < max_fit_ct
                norm_fit_ct = (fit_ct - min_fit_ct) ./ (max_fit_ct - min_fit_ct)
                    ;
            end
            color_density = 256;
            color_grade = grade(cell_type_color_map(ct,:), color_density);
            color_idx = floor((color_density - 1) * norm_fit_ct) + 1;
            scatter_colors = color_grade(color_idx,:);
            [x,y,z] = ind2sub(size(occ_ct), find(occ_ct));
            % note the axis order for plotting: <y,x,z>
            scatter3(y, x, z, 20, scatter_colors, 'filled', 's');
```

```
            colormap(grade(cell_type_color_map(ct,:), color_density));
            drawnow;
            ch = colorbar;
            if min_fit_ct < max_fit_ct
                caxis([min_fit_ct max_fit_ct]);
            end
            cbfreeze(ch);
            xlabel('x');
            ylabel('y');
            zlabel('z');
            if z_dim > 1
                axis([1 x_dim 1 y_dim 1 z_dim]);
            else
                axis([1 x_dim 1 y_dim]);
            end
            axis square;
            title({['Local Fitness of Cell Type ' num2str(ct) ' (3D View)'] ['t
                =' num2str(tick)]});

            % create figure file name and save figure
            len_num_iters = length(num2str(num_iters));
            len_tick = length(num2str(tick));
            len_pad = len_num_iters - len_tick;
            pad = '';
            for pi = 1 : len_pad
                pad = strcat('0', pad);
            end
            fig_file = strcat(path, '/', 'local_fitness_3d_', num2str(ct), '_',
                pad, num2str(tick));
```

```
            saveas(gcf, fig_file, 'pdf');

            % delete figure
            close(h);

        end

    end  % function save_local_fitness_3d

    %% plot local fitness and save them to disk
    function [] = save_local_fitness(ct_range)

        % cell type local fitness (2-D slice)
        for ct = ct_range

            % process only designated cell types
            if ~numel(find(output_only_cell_types == ct))
                continue;
            end

            % set up an invisible figure
            h = figure('Visible', 'off');

            % plot the image
            imagesc((occupation(:,:,mid_z) == ct) .* fitness_1(:,:,mid_z));
            colormap(grade(cell_type_color_map(ct,:), 256));
            cbfreeze(colorbar);
            freezeColors;
            set(gca, 'YDir', 'normal');
```

```
            axis square;
            title({['Local Fitness of Cell Type ' num2str(ct)] ['t=' num2str(
                tick)]});

            % create figure file name and save figure
            len_num_iters = length(num2str(num_iters));
            len_tick = length(num2str(tick));
            len_pad = len_num_iters - len_tick;
            pad = '';
            for pi = 1 : len_pad
                pad = strcat('0', pad);
            end
            fig_file = strcat(path, '/', 'local_fitness_', num2str(ct), '_', pad
                , num2str(tick));
            saveas(gcf, fig_file, 'pdf');

            % delete figure
            close(h);

        end

    end  % function save_local_fitness

    %% plot neighborhood fitness and save them to disk
    function [] = save_neighborhood_fitness(ct_range)

        % cell type neighborhood fitness (2-D slice)
        for ct = ct_range
```

```
% process only designated cell types
if ~numel(find(output_only_cell_types == ct))
    continue;
end

% set up an invisible figure
h = figure('Visible', 'off');

% plot the image
imagesc((occupation(:,:,mid_z) > 1) .* fitness_n(:,:,mid_z,ct));
colormap(grade(cell_type_color_map(ct,:), 256));
cbfreeze(colorbar);
freezeColors;
set(gca, 'YDir', 'normal');
axis square;
title({['Neighborhood Fitness of Cell Type ' num2str(ct)] ['t=' num2
    str(tick)]});

% create figure file name and save figure
len_num_iters = length(num2str(num_iters));
len_tick = length(num2str(tick));
len_pad = len_num_iters - len_tick;
pad = '';
for pi = 1 : len_pad
    pad = strcat('0', pad);
end
fig_file = strcat(path, '/', 'neighborhood_fitness_', num2str(ct),
    '_', pad, num2str(tick));
saveas(gcf, fig_file, 'pdf');
```

```
1026
            % delete figure
            close(h);

        end
1031
    end  % function save_neighborhood_fitness

    %% plot particle concentration and save them to disk
    function [] = save_particle_concentration(gray_256, pt_range)
1036
        % particle type concentrations (2-D slice)
        for pt = pt_range

            % process only designated particle types
1041        if ~numel(find(output_only_particle_types == ct))
                continue;
            end

            % set up an invisible figure
1046        h = figure('Visible', 'off');

            % plot the image
            con_pt = concentration(:,:,mid_z,pt);
            min_con_pt = min(min(con_pt));
1051        max_con_pt = max(max(con_pt));
            mesh(con_pt);
            colormap(gray_256);
            min_rel = '=';
```

```
        if min_con_pt > 1000
            min_rel = '>';
            min_con_pt = 1000;
        end
        max_rel = '=';
        if max_con_pt > 1000
            max_rel = '>';
            max_con_pt = 1000;
        end
        axis square;
        xlim([1 x_dim]);
        ylim([1 y_dim]);
        title({['Concentration of Particle Type ' num2str(pt)] ['t=' num2str
            (tick)] ['min' min_rel sprintf('%3.3f', min_con_pt) ', max' max_
            rel sprintf('%3.3f', max_con_pt)]});

        % create figure file name and save figure
        len_num_iters = length(num2str(num_iters));
        len_tick = length(num2str(tick));
        len_pad = len_num_iters - len_tick;
        pad = '';
        for pi = 1 : len_pad
            pad = strcat('0', pad);
        end
        fig_file = strcat(path, '/', 'particle_concentration_', num2str(pt),
             '_', pad, num2str(tick));
        saveas(gcf, fig_file, 'pdf');

        % delete figure
```

```
            close(h);


        end


    end  % function save_particle_concentration

    %% plot a time series average +/- standard deviation (average curve
        surrounded by +/- gray patches)
    %  a: average            (m time series x n ticks)
    %  s: standard deviation (m time series x n ticks)
    %  c: color map          (m time series x 3 {r,g,b})
    %  b: begin time         (integer)
    %  e: end time           (integer)
    function [] = plot_time_series(a, s, c, b, e)
        domain = b : e;
        gray = [0.9 0.9 0.9];
        hold on;
        for t = 1 : size(a,1)
            patch([domain fliplr(domain)], [a(t,domain) - s(t,domain), fliplr(a(
                t,domain) + s(t,domain))], gray, 'LineStyle', 'none');
            plot(domain, a(t,domain), 'color', c(t,:));
        end
        hold off;
        freezeColors;
        if tick > 1
            xlim([1 tick]);
        end;
    end  % function plot_time_series
```

```
%% create a color map using a basis and a granularity
%  c: color map basis     ([r g b])
%  n: density of gradient (integer)
function [q] = grade(c, n)
    r = linspace(0,c(1),n)';
    g = linspace(0,c(2),n)';
    b = linspace(0,c(3),n)';
    q = horzcat(r,g,b);
end  % function grade

%% create a 3D Gaussian filter using a kernel and standard deviation
%  k: kernel              ([kernel_dim_x kernel_dim_y kernel_dim_z])
%  s: standard deviation ([sigma_x sigma_y sigma_z])
function [g] = gauss3(k, s)
    k = floor(k/2);
    [gx, gy, gz] = ndgrid(-k(1):k(1), -k(2):k(2), -k(3):k(3));
    gx = gx / s(1);
    gy = gy / s(2);
    gz = gz / s(3);
    g = exp(-(gx .* gx + gy .* gy + gz .* gz) .* 0.5);
    g = g / sum(g(:));
end  % function gauss3

%% create a random 3D binary matrix using a gradient function for density
    along any combination of axis directions
%  xd   : x dimension
%  yd   : y dimension
%  zd   : z dimension
```

```
%  f    : gradient function (f(upper)...f(lower) guaranteed to be in the
    range [0,1])
%  d_beg: domain lower bound
%  d_end: domain upper bound
%  dir  : direction of gradient (3-element binary vector)
function [dm] = density_grad(xd, yd, zd, f, d_beg, d_end, dir)
    dm = [];

    domain = linspace(d_beg,d_end,xd);
    target = f(domain);
    dx = [];
    for z_idx = 1 : zd
        zm = [];
        for x_idx = 1 : xd
            zm = horzcat(zm, rand(yd,1) < target(x_idx));
        end
        dx(:,:,z_idx) = zm;
    end

    domain = linspace(d_beg,d_end,yd);
    target = f(domain);
    dy = [];
    for z_idx = 1 : zd
        zm = [];
        for y_idx = 1 : yd
            zm = vertcat(zm, rand(1,xd) < target(y_idx));
        end
        dy(:,:,z_idx) = zm;
    end
```

```
        domain = linspace(d_beg,d_end,zd);
        target = f(domain);
        dz = [];
        for z_idx = 1 : zd
            zm = rand(xd,yd) < target(z_idx);
            dz(:,:,z_idx) = zm;
        end

        if     isequal(dir, [0 0 0])
            dm = rand(xd,yd,zd);
        elseif isequal(dir, [0 0 1])
            dm = dz;
        elseif isequal(dir, [0 1 0])
            dm = dy;
        elseif isequal(dir, [0 1 1])
            dm = dy & dz;
        elseif isequal(dir, [1 0 0])
            dm = dx;
        elseif isequal(dir, [1 0 1])
            dm = dx & dz;
        elseif isequal(dir, [1 1 0])
            dm = dx & dy;
        elseif isequal(dir, [1 1 1])
            dm = dx & dy & dz;
        end
    end

    %% initialize cell types
```

```
    function [] = initialize_occupation()
        initial_occupation_pattern;
        occupation = permute(occupation, [2 1 3]);
    end  % function initialize_occupation

    %% initial occupation pattern
    function [] = initial_occupation_pattern()
        occupation(:,     :,     :    ) = 2;  % initialize with empty cells
        occupation(mid_x, mid_y, mid_z) = 3;  % place an alpha smack in the
            middle
        for i = 1 : 100  % randomly place 100 vessels
            rx = mid_x;
            while rx == mid_x
                rx = floor(rand * x_dim) + 1;
            end
            ry = mid_y;
            while ry == mid_y
                ry = floor(rand * y_dim) + 1;
            end
            rz = mid_z;
            while rz == mid_z
                rz = floor(rand * z_dim) + 1;
            end
            occupation(rx, ry, rz) = 1;
        end
    end  % function initial_occupation_pattern

end  % function simulator
```

C

IMAGE PROCESSING CODE

C.1 INTENSITY-SAMPLE-RAY-BUNDLES

```
function [] = intensity_sample_ray_bundles(im_filename, label, r_center, c_
    center)

batch_mode              = 1;        % in batch mode, figures are invisible and
    no output is displayed
im.raw                  = imread(im_filename);  % raw image filename
r_dim                   = size(im.raw, 1);  % row dimension of the raw image
c_dim                   = size(im.raw, 2);  % column dimension of the raw image
smooth_num              = 100;      % number of times to smooth the gray image
smooth_mask             = [5 5];    % size of mask to smooth the gray image
smooth_std              = 5.0;      % standard deviation of the Gaussian
    function used to smooth the gray image
delta_theta_bundle      = 2*pi;     % angle of bundle: 2*pi (full circle => one
     bundle), pi/4 (1/8 circle => 8 bundles), etc.
delta_theta             = pi/40;    % angle between each extended sample ray:
    pi/40 => 80 rays per full circle, etc.
delta_rho               = 1;        % radial distance between intensity level
    samples (number of pixels)
delta_n                 = 0;        % radius of neighborhood over which to
    average intensity levels at each sample
radius_mean_min_window  = 1000;     % maximum radial distance over which to
    compute the global minimum mean for determining the radius of each bundle
```

```
radius_median_min_window = 1000;     % maximum radial distance over which to
    compute the global minimum median for determining the radius of each bundle
max_rho_sample           = ceil(sqrt(r_dim^2+c_dim^2)/delta_rho);  % maximum
    number of samples that each sample ray can take (for preallocating arrays)
max_theta_sample         = ceil(2*pi/delta_theta);                 % maximum
    number of sample rays that can fit in a circle (for preallocating arrays,
    setting index limits)
max_theta_bundle         = ceil(2*pi/delta_theta_bundle);          % maximum
    number of bundles that can fit in a circle (for setting index limits)
bundle_samples           = ceil(delta_theta_bundle/delta_theta);   % number of
    sample rays that can fit in a bundle (for determining ranges)
center_num               = numel(r_center);  % number of centers to process

% create new results directory
path = strcat('/tmp/analysis/gradient/', label);
mkdir(path);

% open a data file
data_file = strcat(path, '/', 'data.txt');
fid = fopen(data_file, 'w');

fprintf(fid, 'number of bundles per circle = %f\n', max_theta_bundle);
fprintf(fid, 'number of sample rays per circle = %f\n', max_theta_sample);
fprintf(fid, '\n');

% convert RGB image to gray image
im.gray = rgb2gray(im.raw);

% smooth gray image
```

```
G = fspecial('gaussian', smooth_mask, smooth_std);
im.smooth = im.gray;
for i = 1 : smooth_num
    im.smooth = imfilter(im.smooth, G, 'replicate', 'conv');
end

% for each center, measure intensity levels using bundles of sample rays, and
    plot the quantitative results for each bundle
for center_iter = 1 : center_num

    fprintf(fid, 'center %d\n', center_iter);

    % setup
    r_init       = r_center(center_iter);
    c_init       = c_center(center_iter);
    measurements = 256 * ones(max_rho_sample, max_theta_sample);

    % measure
    for theta_iter = 1 : max_theta_sample
        r        = r_init;
        c        = c_init;
        theta    = (theta_iter - 1)*delta_theta;
        rho      = 0;
        rho_iter = 1;
        while r >= 1 && r <= r_dim && c >= 1 && c <= c_dim;
            neighbor_val = [];
            neighbor_num = 1;
            for rr = -delta_n : delta_n
                for cc = -delta_n : delta_n
```

```
                if r+rr >= 1 && r+rr <= r_dim && c+cc >=1 && c+cc <= c_dim
67                  neighbor_val(neighbor_num) = im.smooth(r+rr,c+cc);
                    neighbor_num = neighbor_num + 1;
                end
            end
        end
72      val = mean(neighbor_val);
        measurements(rho_iter,theta_iter) = val;
        if ~batch_mode
            fprintf('%d/%d: (%d,%f) = (%d,%d): %d\n', center_iter, center_
                num, rho, theta/pi, r, c, val);
        end
77      rho = rho + delta_rho;
        [x, y] = pol2cart(theta, rho);
        r = r_init - floor(y);
        c = c_init + floor(x);
        rho_iter = rho_iter + 1;
82  end
end

% compute statistics and plot
for bundle_iter = 1 : max_theta_bundle
87

    fprintf(fid, '\tbundle %d\n', bundle_iter);

    measurements_bundle_range      = (bundle_iter - 1)*bundle_samples + 1 :
        bundle_iter*bundle_samples;
```

```
        measurements_mean              = safe_stats(@mean, measurements(:,
            measurements_bundle_range)); % mean(measurements(:,measurements_
            bundle_range), 2);
92      measurements_std               = safe_stats(@std, measurements(:,
            measurements_bundle_range)); % std(measurements(:,measurements_
            bundle_range), 0, 2);
        measurements_cv                = measurements_std ./ measurements_mean;
        measurements_median            = safe_stats(@median, measurements(:,
            measurements_bundle_range)); % median(measurements(:,measurements_
            bundle_range), 2);
        measurements_median_min_locs   = find_mins(measurements_median, radius_
            median_min_window);
        measurements_radius            = measurements_median_min_locs(1) * delta
            _rho;
97      measurements_range             = 1:measurements_radius;
        radii(center_iter,bundle_iter) = measurements_radius;

        fprintf(fid, '\t\tradius = %d\n', measurements_radius);

102     if batch_mode
            h = figure('Visible', 'off');
        else
            h = figure(bundle_iter);
        end
107     clf;

        subplot_width = 3;

        % plot all trajectories, overlaid with mean and median
```

```
        subplot(1,subplot_width,1);
        hold on;
        safe_plot(measurements(measurements_range,:));
        plot(measurements_mean(measurements_range), 'b');
        plot(measurements_median(measurements_range), 'r');
        hold off;
        title(['radius=', num2str(measurements_radius)]);

        % plot the mean +/- std
        fprintf(fid, '\t\tmean statistics\n');
        subplot(1,subplot_width,2);
        plot_time_series(measurements_mean', measurements_std', [0 0 1], 1,
            measurements_radius);
        hold on;
        [seg_i, seg_j] = segmented_least_squares(measurements_range,
            measurements_mean(measurements_range)', 200);
        num_segs = numel(seg_i);
        seg_a = zeros(1,num_segs);
        seg_b = zeros(1,num_segs);
        seg_e = zeros(1,num_segs);
        for s = 1 : num_segs
            fprintf(fid, '\t\t\tsegment %d\n', s);
            [seg_a(s), seg_b(s)] = least_squares_fit(measurements_range,
                measurements_mean(measurements_range)', seg_i(s), seg_j(s));
            seg_e(s) = least_squares_error(measurements_range, measurements_mean
                (measurements_range)', seg_i(s), seg_j(s));
            plot(measurements_range(seg_i(s):seg_j(s)), seg_a(s) .* measurements
                _range(seg_i(s):seg_j(s)) + seg_b(s), 'k');
            fprintf(fid, '\t\t\t\tbeg = %d\n', seg_i(s));
```

```
        fprintf(fid, '\t\t\t\tend = %d\n', seg_j(s));
137     fprintf(fid, '\t\t\t\tlen = %d\n', seg_j(s) - seg_i(s) + 1);
        fprintf(fid, '\t\t\t\tslo = %f\n', seg_a(s));
        fprintf(fid, '\t\t\t\terr = %f\n', seg_e(s));
    end
    hold off;
142 title_foo = {};
    for s = 1 : num_segs
        title_foo{numel(title_foo)+1} = ['l=', num2str(seg_j(s) - seg_i(s) +
             1), ', s=', sprintf('%0.2f', seg_a(s)), ', e=', sprintf('%0.2f
            ', seg_e(s))];
    end
    title(title_foo);
147
    % plot the median +/- std
    fprintf(fid, '\t\tmedian statistics\n');
    subplot(1,subplot_width,3);
    plot_time_series(measurements_median', measurements_std', [1 0 0], 1,
        measurements_radius);
152 hold on;
    [seg_i, seg_j] = segmented_least_squares(measurements_range,
        measurements_median(measurements_range)', 200);
    num_segs = numel(seg_i);
    seg_a = zeros(1,num_segs);
    seg_b = zeros(1,num_segs);
157 seg_e = zeros(1,num_segs);
    for s = 1 : numel(seg_i)
        fprintf(fid, '\t\t\tsegment %d\n', s);
```

```
            [seg_a(s), seg_b(s)] = least_squares_fit(measurements_range,
                measurements_median(measurements_range)', seg_i(s), seg_j(s));
            seg_e(s) = least_squares_error(measurements_range, measurements_mean
                (measurements_range)', seg_i(s), seg_j(s));
            plot(measurements_range(seg_i(s):seg_j(s)), seg_a(s) .* measurements
                _range(seg_i(s):seg_j(s)) + seg_b(s), 'k');
            fprintf(fid, '\t\t\t\tbeg = %d\n', seg_i(s));
            fprintf(fid, '\t\t\t\tend = %d\n', seg_j(s));
            fprintf(fid, '\t\t\t\tlen = %d\n', seg_j(s) - seg_i(s) + 1);
            fprintf(fid, '\t\t\t\tslo = %f\n', seg_a(s));
            fprintf(fid, '\t\t\t\terr = %f\n', seg_e(s));
        end
        hold off;
        title_foo = {};
        for s = 1 : num_segs
            title_foo{numel(title_foo)+1} = ['l=', num2str(seg_j(s) - seg_i(s) +
                 1), ', s=', sprintf('%0.2f', seg_a(s)), ', e=', sprintf('%0.2f
                ', seg_e(s))];
        end
        title(title_foo);

        fig_file = strcat(path, '/', 'blur_radius_', num2str(center_iter), '_
            bundle_', num2str(bundle_iter), '.pdf');
        saveas(gcf, fig_file, 'pdf');

    end

end

```

```
% plot the smoothed image, overlaid with centers, bundles, and bundle numbers
if batch_mode
    h = figure('Visible', 'off');
else
    h = figure(bundle_iter);
end
clf;
imshow(im.smooth);
hold on;
for center_iter = 1 : numel(r_center)
    beg_x = [];
    beg_y = [];
    end_x = [];
    end_y = [];
    plot(c_center(center_iter), r_center(center_iter), 'ro');
    text(c_center(center_iter) + 30, r_center(center_iter), num2str(center_iter)
        , 'Color', 'r');
    for bundle_iter = 1 : max_theta_bundle
        theta_range = (bundle_iter - 1)*delta_theta_bundle : 0.01 : bundle_iter*
            delta_theta_bundle;
        [perim_x, perim_y] = pol2cart(theta_range, radii(center_iter,bundle_iter
            ));
        plot(c_center(center_iter) + perim_x, r_center(center_iter) - perim_y, '
            r');
        [label_x, label_y] = pol2cart(theta_range(floor(numel(theta_range)/2)),
            radii(center_iter,bundle_iter) + 30);
        text(c_center(center_iter) + label_x, r_center(center_iter) - label_y,
            num2str(bundle_iter), 'Color', 'r');
        beg_x(bundle_iter) = c_center(center_iter) + perim_x(1);
```

```
        beg_y(bundle_iter) = r_center(center_iter) - perim_y(1);
        end_x(bundle_iter) = c_center(center_iter) + perim_x(numel(perim_x));
        end_y(bundle_iter) = r_center(center_iter) - perim_y(numel(perim_y));
    end
    for bundle_iter = 1 : max_theta_bundle - 1
        plot([end_x(bundle_iter) beg_x(bundle_iter+1)], [end_y(bundle_iter) beg_
            y(bundle_iter+1)], 'r');
    end
    plot([end_x(bundle_iter+1) beg_x(1)], [end_y(bundle_iter+1) beg_y(1)], 'r');
end
hold off;
fig_file = strcat(path, '/', 'blur_radii.pdf');
saveas(gcf, fig_file, 'pdf');

% close the data file
fclose(fid);

return;

    %% plot a time series average +/- standard deviation (average curve
        surrounded by +/- gray patches)
    %  a: average            (m time series x n ticks)
    %  s: standard deviation (m time series x n ticks)
    %  c: color map          (m time series x 3 {r,g,b})
    %  b: begin time         (integer)
    %  e: end time           (integer)
    function [] = plot_time_series(a, s, c, b, e)
        domain = b : e;
        gray = [0.9 0.9 0.9];
```

```
        hold on;
        for t = 1 : size(a,1)
            patch([domain fliplr(domain)], [a(t,domain) - s(t,domain), fliplr(a(
                t,domain) + s(t,domain))], gray, 'LineStyle', 'none');
            plot(domain, a(t,domain), 'color', c(t,:));
        end
        hold off;
    end  % function plot_time_series

    function [l] = find_mins(a, w)
        [m, l] = min(a(1:w));
    end

    function [F] = safe_stats(f, M)
        num_r = size(M,1);
        num_c = size(M,2);
        F = [];
        for r = 1 : num_r
            safe_set = [];
            for c = 1 : num_c
                if M(r,c) ~= 256
                    safe_set(numel(safe_set)+1) = M(r,c);
                end
            end
            F(r,1) = f(safe_set);
        end
    end

    function [] = safe_plot(M)
```

```
        num_r = size(M,1);
        num_c = size(M,2);
        for c = 1 : num_c
            for r = 1 : num_r
                if M(r,c) == 256
                    break;
                end
            end
            safe_r_idx = r - 1;
            plot(1:safe_r_idx, M(1:safe_r_idx, c), 'Color', [0.9 0.9 0.9]);
        end
    end

end
```

C.2 QUAD-TREE

```
function [] = quad_tree(im_filename)

%% load image
im.raw  = imread(im_filename);
im.gray = rgb2gray(im.raw);
r_dim = size(im.raw, 1);
c_dim = size(im.raw, 2);

%% smooth image
smooth_num = 100;
```

```
G = fspecial('gaussian', [5 5], 5.0);
im.smooth = im.gray;
for i = 1 : smooth_num
    im.smooth = imfilter(im.smooth, G, 'replicate', 'same', 'conv');
end

%% quad tree
r_range = 1:r_dim;
c_range = 1:c_dim;
I = im.smooth(r_range,c_range);

fh=figure;
imshow(I);
hold on;
dissect(I, 1, 1, 0, @(x) std(x)/mean(x), 0.02, 1);
hold off;


    function [] = draw_cross(r, c, rs, cs)
        hrs = floor(rs/2);
        hcs = floor(cs/2);
        plot(c+hcs,  r:r+rs, 'r');  % NS
        plot(c:c+cs, r+hrs,  'r');  % EW
    end


    %% recursively dissect a rectangle
    % I: 2-D image matrix
    % r: row coordinate of NW corner of rectangle
    % c: col coordinate of NW corner of rectangle
    % d: depth of recursion (0-based ply)
```

```
    % f: function handle for rectangle pixel evaluation function
41  % t: threshold value in excess which will trigger further dissection
    % x: draw cross predicate
    function [] = dissect(I, r, c, d, f, t, x)
        if 2^d > min(size(I))
            return;
46      end
        rs = floor(size(I,1) / 2^d);
        cs = floor(size(I,2) / 2^d);
        r_range = r : r+rs-1;
        c_range = c : c+cs-1;
51      W = I(r_range,c_range);
        v = double(reshape(W, 1, rs*cs));
        if f(v) > t
            if x
                draw_cross(r, c, rs, cs);
56          end
            hrs = floor(rs/2);
            hcs = floor(cs/2);
            dissect(I, r,     c,     d+1, f, t, x);
            dissect(I, r+hrs, c,     d+1, f, t, x);
61          dissect(I, r,     c+hcs, d+1, f, t, x);
            dissect(I, r+hrs, c+hcs, d+1, f, t, x);
        end
    end

66  end
```

C.3 PLY-STATS-QUAD-TREE

```
function [] = ply_stats_quad_tree(im_filename, label)

%% setup

batch_mode  = 1;  % in batch mode, figures are invisible and no output is
    displayed
im.raw      = imread(im_filename);  % raw image
r_dim       = size(im.raw, 1);  % row dimension of the raw image
c_dim       = size(im.raw, 2);  % column dimension of the raw image
smooth_num  = 100;    % number of times to smooth the gray image
smooth_mask = [5 5];  % size of mask to smooth the gray image
smooth_std  = 5.0;    % standard deviation of the Gaussian function used to
    smooth the gray image

%% preprocessing

% create new results directory
base_path = '/tmp/analysis/quadtree';
full_path = strcat(base_path, '/', label);
if ~exist(base_path, 'dir')
    mkdir(base_path);
end
if ~exist(full_path, 'dir')
    mkdir(full_path);
end

```

```
% open global and local data files
loc_data_file = strcat(full_path, '/', 'data.txt');
loc_fid       = fopen(loc_data_file, 'w');


% convert RGB image to gray image
im.gry = rgb2gray(im.raw);


% smooth gray image
G = fspecial('gaussian', smooth_mask, smooth_std);
im.smo = im.gry;
for i = 1 : smooth_num
    im.smo = imfilter(im.smo, G, 'replicate', 'conv');
end


%% quad tree
r_range = 1:r_dim;
c_range = 1:c_dim;
I = im.smo(r_range,c_range);


%% ply-by-ply quadtree
%    ply 1 is whole image
%    ply 2 is 1/4 of whole image
%    ply q is (1/4)^(q-1) of whole image
%    for each trial value of CV in [0.01 : 0.01 : 0.10] report statistics for
    each ply's set of windows
ply.n      = zeros(1,10);
ply.sum    = zeros(1,10);
ply.mean   = zeros(1,10);
ply.median = zeros(1,10);
```

```
ply.std    = zeros(1,10);
ply.cv     = zeros(1,10);
ply.hist   = zeros(12,10);
for t_iter = 1 : 10
    t = t_iter * 0.01;
    fprintf('Processing t=%f\n', t);
    ply.data = [];
    dissect(I, 1, 1, 0, @(x) std(x)/mean(x), t, 0);
    ply.n(t_iter) = numel(ply.data);
    ply.sum(t_iter) = sum(ply.data);
    ply.mean(t_iter) = mean(ply.data);
    ply.median(t_iter) = median(ply.data);
    ply.std(t_iter) = std(ply.data);
    ply.cv(t_iter) = ply.std / ply.mean;
    for p_iter = 1 : 12
        ply.hist(t_iter,p_iter) = sum(ply.data == p_iter);
    end
end

%% write data

head = strcat(label, '\t');

for t_iter = 1 : 10
    glo_data_file = strcat(base_path, '/', 'data_', num2str(t_iter), '.txt');
    glo_fid = fopen(glo_data_file, 'a');
    fprintf(glo_fid, head);
    fprintf(glo_fid, '%d\t', ply.hist(t_iter,:));
    fprintf(glo_fid, '\n');
```

```
    fclose(glo_fid);
end

fprintf(loc_fid, head);
fprintf(loc_fid, '%d\t', ply.n);
fprintf(loc_fid, '\n');
fprintf(loc_fid, head);
fprintf(loc_fid, '%d\t', ply.sum);
fprintf(loc_fid, '\n');
fprintf(loc_fid, head);
fprintf(loc_fid, '%f\t', ply.mean);
fprintf(loc_fid, '\n');
fprintf(loc_fid, head);
fprintf(loc_fid, '%d\t', ply.median);
fprintf(loc_fid, '\n');
fprintf(loc_fid, head);
fprintf(loc_fid, '%f\t', ply.std);
fprintf(loc_fid, '\n');
fprintf(loc_fid, head);
fprintf(loc_fid, '%f\t', ply.cv);
fprintf(loc_fid, '\n');
for t_iter = 1 : 10
    fprintf(loc_fid, head);
    fprintf(loc_fid, '%d\t', ply.hist(t_iter,:));
    fprintf(loc_fid, '\n');
end

% close the local data file
fclose(loc_fid);
```

```
return;


    function [] = draw_cross(r, c, rs, cs)
        hrs = floor(rs/2);
        hcs = floor(cs/2);
        plot(c+hcs,  r:r+rs, 'r');  % NS
        plot(c:c+cs, r+hrs,  'r');  % EW
    end


    %% recursively dissect a rectangle
    % I: 2-D image matrix
    % r: row coordinate of NW corner of rectangle
    % c: col coordinate of NW corner of rectangle
    % d: depth of recursion (0-based ply)
    % f: function handle for rectangle pixel evaluation function
    % t: threshold value in excess which will trigger further dissection
    % x: draw cross predicate
    function [] = dissect(I, r, c, d, f, t, x)
        if 2^d > min(size(I))
            ply.data(numel(ply.data)+1) = d;
            return;
        end
        rs = floor(size(I,1) / 2^d);
        cs = floor(size(I,2) / 2^d);
        r_range = r : r+rs-1;
        c_range = c : c+cs-1;
        W = I(r_range,c_range);
        v = double(reshape(W, 1, rs*cs));
```

```
        if f(v) > t
            if x
                draw_cross(r, c, rs, cs);
            end
            hrs = floor(rs/2);
            hcs = floor(cs/2);
            dissect(I, r,     c,     d+1, f, t, x);
            dissect(I, r+hrs, c,     d+1, f, t, x);
            dissect(I, r,     c+hcs, d+1, f, t, x);
            dissect(I, r+hrs, c+hcs, d+1, f, t, x);
        else
            ply.data(numel(ply.data)+1) = d;
        end
    end

end
```

C.4 EPC-SIGNATURE

```
function [] = epc_signature(im_filename, label)

%% setup

batch_mode = 1;      % in batch mode, figures are invisible and no output is
    displayed
seg_cost   = 50000;  % cost to add a segment in the segmented least squares
    algorithm
```

```
im.raw     = imread(im_filename);  % raw image
r_dim      = size(im.raw, 1);  % row dimension of the raw image
c_dim      = size(im.raw, 2);  % column dimension of the raw image
gray_lim   = 255;         % maximum intensity level in an 8-bit image
x          = 1:gray_lim;  % the intensity range in an 8-bit image

%% preprocessing

% create new results directory
base_path = '/tmp/analysis/epc';
full_path = strcat(base_path, '/', label);
if ~exist(base_path, 'dir')
    mkdir(base_path);
end
if ~exist(full_path, 'dir')
    mkdir(full_path);
end

% open global and local data files
glo_data_file = strcat(base_path, '/', 'data.txt');
glo_fid       = fopen(glo_data_file, 'a');
loc_data_file = strcat(full_path, '/', 'data.txt');
loc_fid       = fopen(loc_data_file, 'w');

% convert RGB image to gray image
im.gry = rgb2gray(im.raw);

%% compute EPC curve for gray image

```

```
fprintf(glo_fid, '%s\t', label);
fprintf(loc_fid, '%s\t', label);
chi = zeros(1,gray_lim);
for i = 1 : gray_lim
    im.bin = im2bw(im.gry, i/gray_lim);
    [chi(i), l] = imEuler2d(im.bin);
    fprintf(glo_fid, '%d\t', chi(i));
    fprintf(loc_fid, '%d\t', chi(i));
end

%% compute segmented least-squares fit
[seg_i, seg_j] = segmented_least_squares(x, chi, seg_cost);
num_segs = numel(seg_i);
seg_a = zeros(1,num_segs);
seg_b = zeros(1,num_segs);
seg_e = zeros(1,num_segs);
compression_factor = gray_lim/(2*num_segs);
fprintf(glo_fid, '%d\t%f\t', num_segs, compression_factor);
fprintf(loc_fid, '%d\t%f\t', num_segs, compression_factor);
for s = 1 : num_segs
    [seg_a(s), seg_b(s)] = least_squares_fit(x, chi, seg_i(s), seg_j(s));
    seg_e(s) = least_squares_error(x, chi, seg_i(s), seg_j(s));
end
tot_err  = sum(seg_e);
norm_err = tot_err / gray_lim;
fprintf(glo_fid, '%f\t%f\t', tot_err, norm_err);
fprintf(loc_fid, '%f\t%f\t', tot_err, norm_err);
for s = 1 : num_segs
    fprintf(glo_fid, '%f\t%f', seg_a(s), seg_b(s));
```

```
    fprintf(loc_fid, '%f\t%f', seg_a(s), seg_b(s));
    if s ~= num_segs
        fprintf(glo_fid, '\t');
        fprintf(loc_fid, '\t');
    end
end
fprintf(glo_fid, '\n');
fprintf(loc_fid, '\n');

%% plot EPC curve for gray image

if batch_mode
    h = figure('Visible', 'off');
else
    h = figure;
end
clf;
hold on;
plot(x, chi, 'b');
for s = 1 : numel(seg_i)
    plot(x(seg_i(s):seg_j(s)), seg_a(s) .* x(seg_i(s):seg_j(s)) + seg_b(s), 'r')
        ;
end
hold off;
fig_file = strcat(full_path, '/', 'epc_signature.pdf');
saveas(gcf, fig_file, 'pdf');

% close the data files
fclose(glo_fid);
```

```
fclose(loc_fid);

return;

end
```

C.5 MULTITHRESHOLD-SEGMENTATION

```
function [] = multithreshold_segmentation(im_filename, label)

%% setup

batch_mode  = 1;  % in batch mode, figures are invisible and no output is
    displayed
im.raw      = imread(im_filename);  % raw image
r_dim       = size(im.raw, 1);  % row dimension of the raw image
c_dim       = size(im.raw, 2);  % column dimension of the raw image
smooth_num  = 100;    % number of times to smooth the gray image
smooth_mask = [5 5];  % size of mask to smooth the gray image
smooth_std  = 5.0;    % standard deviation of the Gaussian function used to
    smooth the gray image

%% preprocessing

% create new results directory
base_path = '/tmp/analysis/segment';
full_path = strcat(base_path, '/', label);
```

```
if ~exist(base_path, 'dir')
    mkdir(base_path);
end
if ~exist(full_path, 'dir')
    mkdir(full_path);
end

% open global and local data files
glo_data_file = strcat(base_path, '/', 'data.txt');
glo_fid       = fopen(glo_data_file, 'a');
loc_data_file = strcat(full_path, '/', 'data.txt');
loc_fid       = fopen(loc_data_file, 'w');

% convert RGB image to gray image
im.gry = rgb2gray(im.raw);

% smooth gray image
G = fspecial('gaussian', smooth_mask, smooth_std);
im.smo = im.gry;
for i = 1 : smooth_num
    im.smo = imfilter(im.smo, G, 'replicate', 'conv');
end

%% segment gray image

thresh = multithresh(im.gry, 2);
im.seg = imquantize(im.gry, thresh);
im.hyp = im.seg == 1;
im.via = im.seg == 2;
```

```
im.nec = im.seg == 3;
num_pix = r_dim * c_dim;
num_hyp = sum(sum(im.hyp));
num_via = sum(sum(im.via));
num_nec = sum(sum(im.nec));
fprintf(glo_fid, '%s\t%d\t%d\t%d\t%d\t%d\t%d\t', label, num_pix, thresh(1),
    thresh(2), num_hyp, num_via, num_nec);
fprintf(loc_fid, '%s\t%d\t%d\t%d\t%d\t%d\t%d\t', label, num_pix, thresh(1),
    thresh(2), num_hyp, num_via, num_nec);

%% plot gray and segmented-gray images side-by-side

if batch_mode
    h = figure('Visible', 'off');
else
    h = figure;
end
im.rgb = label2rgb(im.seg);
imshowpair(im.gry, im.rgb, 'montage');
fig_file = strcat(full_path, '/', 'segmented_gray.pdf');
saveas(gcf, fig_file, 'pdf');

%% segment smooth image

thresh = multithresh(im.smo, 2);
im.seg = imquantize(im.smo, thresh);
im.hyp = im.seg == 1;
im.via = im.seg == 2;
im.nec = im.seg == 3;
```

```
num_pix = r_dim * c_dim;
num_hyp = sum(sum(im.hyp));
num_via = sum(sum(im.via));
num_nec = sum(sum(im.nec));
fprintf(glo_fid, '%d\t%d\t%d\t%d\t%d\n', thresh(1), thresh(2), num_hyp, num_via,
    num_nec);
fprintf(loc_fid, '%d\t%d\t%d\t%d\t%d\n', thresh(1), thresh(2), num_hyp, num_via,
    num_nec);

%% plot smooth and segmented-smooth images side-by-side

if batch_mode
    h = figure('Visible', 'off');
else
    h = figure;
end
im.rgb = label2rgb(im.seg);
imshowpair(im.smo, im.rgb, 'montage');
fig_file = strcat(full_path, '/', 'segmented_smooth.pdf');
saveas(gcf, fig_file, 'pdf');

% close the data files
fclose(glo_fid);
fclose(loc_fid);

return;

end
```

D

STATISTICAL MODEL CHECKING AND ADAPTIVE SAMPLING CODE

D.1 BAYESIAN STATISTICAL MODEL CHECKING

```
function [h, n, x] = bayesian_statistical_model_checking(phi, theta, T, g)

% phi   in BLTL (or modal logic)
%       [useless here; this code uses a random indicator variable to simulate
    binary outcome of M satisfying phi]
% theta in (0,1)
% T     > 1
% g     prior density g for unknown parameter p
%       [useless here; this code uses a beta distribution parameterized by alpha
     and beta]

p     = 0.99;  % the unknown actual probability of M satisfying phi
alpha = 1;     % first parameter of Beta distribution
beta  = 1;     % second parameter of Beta distribution

n     = 0;     % number of Bernoulli trials performed so far
x     = 0;     % number of successful Bernoulli trials performed so far

while 1
    n = n + 1;
```

```
    sigma = rand < p;  % perform a Bernoulli trial based on a simple random
        variable, based on p
    if sigma
        x = x + 1;
    end
    fprintf('n=%d, x=%d\n', n, x);
    B = bayes_factor(n, x);
    if B > T
        h = 0;
        return;
    elseif B < 1/T
        h = 1;
        return;
    end
end

    function [b] = bayes_factor(n, x)
        F = betacdf(theta, n + alpha, n - x + beta);
        b = (1 / F) - 1;
        return;
    end

end
```

D.2 MC-BOOST

```
function [] = monte_carlo_boost(d, m, s, w)
```

```
% d = number of dimensions
% m = number of samples
% s = sigma  of Gaussian reward/penalty function
% w = weight of Gaussian reward/penalty function

t      = zeros(m,d);         % coordinates of 'true'  samples
f      = zeros(m,d);         % coordinates of 'false' samples
nt     = 0;                  % number of true  samples so far
nf     = 0;                  % number of false samples so far
x      = 0:0.001:1;          % dimension axis
cum_x  = ones(d,numel(x));   % cumulative (unnormalized) PDFs
cum_a  = ones(d,1);          % cumulative PDF areas
norm_x = zeros(d,numel(x));  % normalized PDFs

% initialize the normalized PDFs
for j = 1 : d
    norm_x(j,:) = cum_x(j,:)/cum_a(j);
end

% adaptively sample points according to evolving normalized PDFs
for i = 1 : m

    % sample d coordinates according to d normalized PDFs
    for j = 1 : d
        q(j) = sample(x,norm_x(j,:),1,1);
    end

    % render a result according to a decision process
```

```
    r = decision(q);

    % if the result is true...
    if r

        % record the true sample in the appropriate place
        nt = nt + 1;
        t(nt,:) = q;

        % place a Gaussian reward upon each axis, centered at each
        % corresponding coordinate, having inputted sigma
        for j = 1 : d
            cum_x(j,:) = cum_x(j,:) + w * normpdf(x,q(j),s);
            cum_a(j)   = cum_a(j)   + w * (normcdf(1,q(j),s) - normcdf(0,q(j),s)
                );
        end

    else

        % record the false sample in the appropriate place
        nf = nf + 1;
        f(nf,:) = q;

    end

    % normalize the evolved PDFs
    for j = 1 : d
        norm_x(j,:) = cum_x(j,:)/cum_a(j);
    end
```

```
end

% plot the results
figure(1);
clf;

subplot(1,2,1);
hold on;
scatter(t(1:nt,1), t(1:nt,2), 50, 'r', 'fill');
scatter(f(1:nf,1), f(1:nf,2), 50, 'b', 'fill');
hold off;
xlim([0 1]);
ylim([0 1]);
axis square;

subplot(1,2,2);
hold on;
plot(x, norm_x(1,:));
plot(norm_x(2,:), x);
hold off;
xlim([0 1]);
ylim([0 1]);
axis square;

    % decision process
    function [r] = decision(q)
        if (q(1) > 0.3 && q(1) < 0.7 && q(2) > 0.1 && q(2) < 0.4) | (sqrt((0.8-q
            (1))^2 + (0.8-q(2))^2) < 0.1)
```

```
            r = 1;
        else
            r = 0;
        end
    end

    % sample an arbitrary distribution, defined by y along x,
    % returning an r-by-c matrix of sample
    function [rx] = sample(x,y,r,c)
        max_y = max(y);
        rx = zeros(r*c,1);
        for sn = 1 : r*c
            while 1
                sx = rand;
                sy = rand * max_y;
                if sy <= interp1(x,y,sx)
                    break;
                end
            end
            rx(sn) = sx;
        end
        rx = reshape(rx,r,c);
    end

end
```

D.3 MC-WALK

```
function [] = monte_carlo_walk(d, m, p, n)

% d = number of dimensions
% m = number of samples
% p = diffusion rate
% n = number of random walks

t      = zeros(m,d);         % coordinates of 'true'  samples
f      = zeros(m,d);         % coordinates of 'false' samples
nt     = 0;                  % number of true  samples so far
nf     = 0;                  % number of false samples so far
x      = 0:0.001:1;          % dimension axis
cum_x  = ones(d,numel(x));   % cumulative (unnormalized) PDFs
cum_a  = ones(d,1);          % cumulative PDF areas
norm_x = zeros(d,numel(x));  % normalized PDFs

% initialize the normalized PDFs
for j = 1 : d
    norm_x(j,:) = cum_x(j,:)/cum_a(j);
end

% adaptively sample points according to evolving normalized PDFs
for i = 1 : m

    % sample d coordinates according to d normalized PDFs
    for j = 1 : d
```

```
    q(j) = sample(x,norm_x(j,:),1,1);
end

% render a result according to a decision process
r = decision(q);

% if the result is true...
if r

    % record the true sample in the appropriate place
    nt = nt + 1;
    t(nt,:) = q;

    for k = 1 : n

        % walk q
        q = walk(q);

        % render a result according to a decision process
        r = decision(q);

        % if the result is true...
        if r

            % record the true sample in the appropriate place
            nt = nt + 1;
            t(nt,:) = q;

        else
```

```
                % record the false sample in the appropriate place
                nf = nf + 1;
                f(nf,:) = q;

            end

        end

    else

        % record the false sample in the appropriate place
        nf = nf + 1;
        f(nf,:) = q;

    end

end

% plot the results
figure(1);
clf;
hold on;
scatter(t(1:nt,1), t(1:nt,2), 5, 'r', 'fill');
scatter(f(1:nf,1), f(1:nf,2), 5, 'b', 'fill');
hold off;
xlim([0 1]);
ylim([0 1]);
axis square;
```

```
% decision process
function [r] = decision(q)
    if (q(1) > 0.3 && q(1) < 0.7 && q(2) > 0.1 && q(2) < 0.4) | (sqrt((0.8-q
        (1))^2 + (0.8-q(2))^2) < 0.1)
        r = 1;
    else
        r = 0;
    end
end

% sample an arbitrary distribution, defined by y along x,
% returning an r-by-c matrix of sample
function [rx] = sample(x,y,r,c)
    max_y = max(y);
    rx = zeros(r*c,1);
    for sn = 1 : r*c
        while 1
            sx = rand;
            sy = rand * max_y;
            if sy <= interp1(x,y,sx)
                break;
            end
        end
        rx(sn) = sx;
    end
    rx = reshape(rx,r,c);
end
```

```
    % walk the walk
    function [wq] = walk(q)
        wq = q;
        for wi = 1 : d
            wq(wi) = wq(wi) + randn() * sqrt(2*p);
        end
    end

end
```

D.4 MEAN-VARIANCE THRESHOLDING

```
function [sample_mean, sample_std, sample_cv, n] = mean_variance_thresholding(cv
    _thresh, trial_lower_thresh, trial_upper_thresh)

% cv_thresh          = CV lower threshold at which to stop
% trial_lower_thresh = lower bound on the number of trials
% trial_upper_thresh = upper bound on the number of trials

n         = 0;
sample_cv = Inf;
sample    = zeros(1, trial_upper_thresh);

while n < trial_upper_thresh
    if sample_cv < cv_thresh && n > trial_lower_thresh
        return;
    end
```

```
    n = n + 1;
    sample(n)   = 0.5 + 0.1 * randn();
    sample_mean = mean(sample(1:n));
    sample_std  = std(sample(1:n));
    sample_cv   = abs(sample_std / sample_mean);
    fprintf('%d: mean=%f, std=%f, cv=%f\n', n, sample_mean, sample_std, sample_
        cv);
end

end
```

D.5 MC-BRANCH-AND-BOUND

```
function [] = branch_and_bound()

%% load image
im.gray = create_image(1000);
r_dim = size(im.gray, 1);
c_dim = size(im.gray, 2);


%% smooth image
smooth_num = 100;
G = fspecial('gaussian', [5 5], 5.0);
im.smooth = im.gray;
for i = 1 : smooth_num
    im.smooth = imfilter(im.smooth, G, 'replicate', 'same', 'conv');
```

```
end


%% quadtree implementing branch-and-bound
T = 0;
r_range = 1:r_dim;
c_range = 1:c_dim;
I = im.smooth(r_range,c_range);

fh=figure;
imshow(I);
hold on;
dissect(I, 1, 1, 0, 6, 128, 1);
hold off;


    function [] = draw_cross(r, c, rs, cs)
        hrs = floor(rs/2);
        hcs = floor(cs/2);
        plot(c+hcs,  r:r+rs, 'r');  % NS
        plot(c:c+cs, r+hrs,  'r');  % EW
    end


    %% recursively dissect a rectangle
    % I: 2-D image matrix
    % r: row coordinate of NW corner of rectangle
    % c: col coordinate of NW corner of rectangle
    % d: depth of recursion (0-based ply)
    % n: depth cutoff (0-based ply)
    % s: number of samples to take inside of rectangle
    % x: draw cross predicate
```

```
function [] = dissect(I, r, c, d, n, s, x)
    if 2^d > min(size(I))
        return;
    end
    rs = floor(size(I,1) / 2^d);
    cs = floor(size(I,2) / 2^d);
    r_range = r : r+rs-1;
    c_range = c : c+cs-1;
    W = I(r_range,c_range);
    [min_sample, max_sample] = sample(W, r, r+rs-1, c, c+cs-1, floor(s / 2^d
        ));
    if min_sample > T
        T = min_sample;
    end
    if max_sample > 0 && max_sample >= T && d < n
        if x
            draw_cross(r, c, rs, cs);
        end
        hrs = floor(rs/2);
        hcs = floor(cs/2);
        dissect(I, r,     c,     d+1, n, s, x);
        dissect(I, r+hrs, c,     d+1, n, s, x);
        dissect(I, r,     c+hcs, d+1, n, s, x);
        dissect(I, r+hrs, c+hcs, d+1, n, s, x);
    end
end

% decision process
function [r] = decision(q)
```

```
        if (q(1) > 0.3 && q(1) < 0.7 && q(2) > 0.1 && q(2) < 0.4) | (sqrt((0.8-q
            (1))^2 + (0.8-q(2))^2) < 0.1)
            r = 1;
        else
            r = 0;
        end
    end

    % create a binary image based in the decision process
    function [i] = create_image(p)
        i = zeros(p, p);
        for r = 1 : p
            for c = 1 : p
                y = p - r;
                x = c;
                nx = x / p;
                ny = y / p;
                i(r,c) = 255 * decision([nx ny]);
            end
        end
    end

    function [min_sampled_w, max_sampled_w] = sample(w, r_beg, r_end, c_beg, c_
        end, num_samples)
        min_sampled_w = Inf;
        max_sampled_w = 0;
        for sample_idx = 1 : num_samples
            sample_r_idx = floor((r_end - r_beg + 1) * rand()) + 1;
            sample_c_idx = floor((c_end - c_beg + 1) * rand()) + 1;
```

```
            sample = w(sample_r_idx, sample_c_idx);
            if sample < min_sampled_w
                min_sampled_w = sample;
            end
            if sample > max_sampled_w
                max_sampled_w = sample;
            end
        end
    end

end
```

BIBLIOGRAPHY

[1] C. Athena Aktipis. Know when to walk away: contingent movement and the evolution of cooperation. *Journal of Theoretical Biology*, 231:249–260, 2004.

[2] C. Athena Aktipis. Is cooperation viable in mobile organisms? Simple walk away rule favors the evolution of cooperation in groups. *Evolution and Human Behavior*, 32:263–276, 2011.

[3] Sergey Astanin and Luigi Preziosi. Mathematical modeling of the Warburg effect in tumor cords. *Journal of Theoretical Biology*, 258:578–590, 14 Feb 2009.

[4] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, revised edition, 2006.

[5] Robert Axelrod and Douglas Dion. The further evolution of cooperation. *Science*, 242(4884):1385–1390, 1988.

[6] Robert Axelrod and William D. Hamilton. The evolution of cooperation. *Science*, 211(1390):1390–1396, 1981.

[7] Robert Axelrod, David E. Axelrod, and Kenneth J. Pienta. Evolution of cooperation among tumor cells. *PNAS*, 103(36):13474–13479, 2006.

[8] L.A. Bach, S.M. Bentzen, J. Alsner, and F.B. Christiansen. An evolutionary game-model of tumor cell interactions: possible relevance to gene therapy. *European Journal of Cancer*, 37:2116–2120, 2001.

[9] L.A. Bach, D.J.T. Sumpter, J. Alsner, and V. Loeschcke. Spatial evolutionary games of interaction among generic cancer cells. *Journal of Theoretical Medicine*, 5(1):47–58, 2003.

[10] Ezio Bartocci, Flavio Corradini, Emilia Entcheva, Radu Grosu, and Scott A. Smolka. CellExcite: an efficient simulation environment for excitable cells. In *Italian Society of Bioinformatics (BITS): Annual Meeting 2007*, pages 1–13. BMC Bioinformatics 2008, 9(Suppl 2):S3, 26–28 Apr 2007. http://www.biomedcentral.com/bmcbioinformatics/supplements/9/S2.

[11] D. Basanta, H. Hatzikirou, and A. Deutsch. Studying the emergence of invasiveness in tumors using game theory. *European Physical Journal B*, 63:393–397, 2008.

[12] D. Basanta, M. Simon, H. Hatzikirou, and A. Deutsch. Evolutionary game theory elucidates the role of glycolysis in glioma progression and invasion. *Cell Proliferation*, 41:980–987, 2008.

[13] Amy L. Bauer, Trachette L. Jackson, and Yi Jiang. A cell-based model exhibiting branching and anastomosis during tumor-induced angiogenesis. *Biophysical Journal*, 92(9):3105–3121, 1 May 2007. doi:10.1529/biophysj.106.101501.

[14] A.E. Beaton and J.W. Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974.

[15] Richard Ernest Bellman. *Dynamic Programming*. Princeton University Press, 1957. ISBN 978-0-691-07951-6.

[16] Thi Bui and Craig B. Thompson. Cancer's sweet tooth. *Cancer Cell*, pages 419–420, Jun 2006.

[17] Rob A. Cairns and Tak W. Mak. Oncogenic isocitrate dehydrogenase mutations: mechanisms, models, and clinical opportunities. *Cancer Discovery*, OF:1–12, Jul 2013. doi:10.1158/2159-8290.cd-13-0083.

[18] Rob A. Cairns, Isaac S. Harris, and Tak W. Mak. Regulation of cancer cell metabolism. *Nature Reviews Cancer*, 11:85–95, Feb 2011. doi:10.1038/nrc2981.

[19] Jason R. Cantor and David M. Sabatini. Cancer cell metabolism: one hallmark, many faces. *Cancer Discovery*, pages OF1–OF18, Oct 2012. doi:10.1158/2159-8290.CD-12-0345.

[20] Laura I. Cárdenas-Navia, Daniel Mace, Rachel A. Richardson, David F. Wilson, Siqing Shan, and Mark W. Dewhirst. The pervasive presence of fluctuating oxygenation in tumors. *Cancer Research*, 68(14):5812–5819, 15 Jul 2008. doi:10.1158/0008-5472.can-07-6387.

[21] *Oxygen sensing in cancer and metabolism*, 8 Jul 2013. Cell. http://view6.workcast.net/?pak=3998850097719807.

[22] Paolo Cirri and Paola Chiarugi. Cancer associated fibroblasts: the dark side of the coin. *American Journal of Cancer Research*, 1(4):482–497, 2011.

[23] Chris Cleveland, David Liao, and Robert Austin. Physics of cancer progression: a game theory perspective. *AIP Advances* 2, 011201:1–10, 22 Mar 2012. doi:10.1063/1.3699043.

[24] Vittorio Cristini and John Lowengrub, editors. *Multiscale Modeling of Cancer: An Integrated Experimental and Mathematical Modeling Approach*. Cambridge University Press, 2010.

[25] Chi V. Dang. MYC, microRNAs and glutamine addiction in cancers. *Cell Cycle*, 8(20):3243–3256, 15 Oct 2009.

[26] Chi V. Dang. Rethinking the Warburg effect with Myc micromanaging glutamine metabolism. *Cancer Research*, 70(3):859–862, 1 Feb 2010. doi:10.1158/0008-5472.can-09-3556.

[27] Chi V. Dang. Links between metabolism and cancer. *Genes & Development*, 26: 877–890, 2012. doi:10.1101/gad.189365.112.

[28] Ralph J. DeBerardinis. Is cancer a disease of abnormal cellular metabolism?: new angles on an old idea. *Genetics in Medicine*, 10(11):767–777, Nov 2008. doi:10.1097/gim.0b013e31818b0d9b.

[29] Ralph J. DeBerardinis and Craig B. Thompson. Cellular metabolism and disease: what do metabolic outliers teach us? *Cell*, 148:1132–1144, 16 Mar 2012. doi:10.1016/j.cell.2012.02.032.

[30] Ralph J. DeBerardinis, Julian J. Lum, Georgia Hatzivassiliou, and Craig Thompson. The biology of cancer: metabolic reprogramming fuels cell growth and proliferation. *Cell Metabolism*, 7:11–20, Jan 2008. doi:10.1016/j.cmet.2007.10.002.

[31] Ralph J. DeBerardinis, Nabil Sayed, Dara Ditsworth, and Craig B. Thompson. Brick by brick: metabolism and tumor cell growth. *Current Opinion in Genetic Development*, 18(1):54–61, Feb 2008.

[32] Thomas S. Deisboeck and Georgios S. Stamatakos, editors. *Multiscale Cancer Modeling*. CRC Press, 2011.

[33] D. Dey and J.O. Berger. On truncation of shrinkage estimators in simultaneous estimation of normal means. *Journal of the American Statistics Association*, 78 (384):865–869, 1983.

[34] Jan R. Dörr, Maja Milanovic, Dido Lenze, Gunnar Dittmar, Georg Lenz, and Clemens A. Schmitt. Mechanism and predictive power of paracrine, secondary senescence. Unpublished.

[35] Bradley Efron. Large-scale simultaneous hypothesis testing: the choice of a null hypothesis. *Journal of the American Statistics Association*, 88(465):96–104, 2004.

[36] Bradley Efron. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. Cambridge University Press, 2010.

[37] James J. Elser, John D. Nagy, and Yang Kuang. Biological stoichiometry: an ecological perspective on tumor dynamics. *BioScience*, 53:1112–1120, 2003.

[38] James J. Elser, Marcia M. Kyle, Marilyn S. Smith, and John D. Nagy. Biological stoichiometry in human cancer. *PLoS ONE*, 2(10:e1028):1–7, Oct 2007. doi:10.1371/journal.pone.0001028.

[39] Heinz W. Engl, Christoph Flamm, Philipp K ugler, James Lu, Stefan M uller, and Peter Schuster. Inverse problems in systems biology. *Inverse Problems*, 25 (12):123014, Dec 2009. doi:10.1088/0266-5611/25/12/123014.

[40] Joseph Farrell. Meaning and credibility in cheap-talk games. *Games and Economic Behavior*, 5:514–531, 1993.

[41] Olivier Feron. Pyruvate into lactate and back: from the Warburg effect to symbiotic energy fuels exchange in cancer cells. *Radiotherapy and Oncology*, 92:329–333, 13 Jul 2009. doi:10.1016/j.radonc.2009.06.025.

[42] Tania Fiaschi and Paola Chiarugi. Oxidative stress, tumor microenvironment, and metabolic reprogramming: a diabolical liaison. *International Journal of Cell Biology*, 2012(762825):1–8, 2012. doi:10.1155/2012/762825.

[43] Brian P. Fiske and Matthew G. Vander Heiden. Seeing the Warburg effect in the developing retina. *Nature Cell Biology*, 14(8):790–791, Aug 2012.

[44] Daniela Gaglio, Christian M. Metallo, Paulo A. Gameiro, Karsten Hiller, Lara Sala Danna, Chiara Balestrieri, Lilia Alberghina, Gregory Stephanopoulos, and Ferdinando Chiaradonna. Oncogenic K-Ras decouples glucose and glutamine metabolism to support cancer cell growth. *Molecular Systems Biology*, 7(523):1–15, 2011. doi:10.1038/msb.2011.56.

[45] R.A. Gatenby, T.L. Vincent, and R.J. Gillies. Evolutionary dynamics in carcinogenesis. *Mathematical Models and Methods in Applied Sciences*, 15:1619, 2005.

[46] Robert A. Gatenby and Robert J. Gillies. Why do cancers have aerobic glycolysis? *Nature Reviews Cancer*, 4:891–899, Nov 2004. doi:10.1038/nrc1478.

[47] Robert A. Gatenby and Robert J. Gillies. A microenvironmental model of carcinogenesis. *Nature Reviews Cancer*, 8(1):56–61, Jan 2008. doi:10.1038/nrc2255.

[48] Robert A. Gatenby and Philip K. Maini. Cancer summed up. *Nature*, 421:321, 2003.

[49] Robert A. Gatenby and Thomas L. Vincent. Application of quantitative models from population biology and evolutionary game theory to tumor therapeutic strategies. *Molecular Cancer Therapeutics*, 2(9):919–927, 2003.

[50] Robert A. Gatenby and Thomas L. Vincent. An evolutionary model of carcinogenesis. *Cancer Research*, 63:6212–6220, 2003.

[51] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 6(6):721–741, 1984. doi:10.1109/TPAMI.1984.4767596.

[52] A.J. Giaccia and E. Schipani. Role of carcinoma-associated fibroblasts and hypoxia in tumor progression. *Current Topics in Microbiology and Immunology*, 345: 31–45, 2010. doi:10.1007/82_2010_73.

[53] Herbert Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2 edition, 2009.

[54] Haijun Gong, Paolo Zuliani, Anvesh Komuravelli, James Faeder, and Edmund M. Clarke. Computational modeling and verification of signaling pathways in cancer. In K. Horimoto, M. Nakatsu, and N. Popov, editors, *International Conference on Algebraic and Numeric Biology 2011 (LNCS 6479)*, pages 117–135. Springer-Verlag Berlin Heidelberg, 31 Jul–2 Aug 2010. http://www.risc.jku.at/conferences/anb2010/.

[55] Haijun Gong, Paolo Zuliani, Anvesh Komuravelli, James Faeder, and Edmund M. Clarke. Analysis and verification of the HMGB1 signaling pathway. In *Asia Pacific Bioinformatics (APBioNet) International Conference on Bioinformatics 2010 (InCoB2010)*, pages 1–13. BMC Bioinformatics 2010, 11(Suppl 7):S10, 26–28 Sep 2010. http://incob.apbionet.org/incob10/.

[56] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 3 edition, 2008.

[57] Douglas R. Green. *Means to an End: Apoptosis and Other Cell Death Mechanisms*. Cold Spring Harbor Laboratory Press, 2011.

[58] Radu Grosu, Scott A. Smolka, Flavio Corradini, Anita Wasilewska, Emilia Entcheva, and Ezio Bartocci. Learning and detecting emergent behavior in networks of cardiac myocytes. *Communications of the ACM*, 52(3):97–105, Mar 2009. doi:10.1145/1467247.1467271.

[59] R. Guderlei, S. Klenk, J. Mayer, V. Schmidt, and E. Spodarev. Algorithms for the computation of the Minkowski functionals of deterministic and random polyconvex sets. *Image and Vision Computing*, 25(4):464–474, Apr 2007. doi:10.1016/j.imavis.2006.07.019.

[60] Shakti Gupta, Eric Aslakson, Brian M. Gurbaxani, and Suzanne D. Vernon. Inclusion of the glucocorticoid receptor in a hypothalamic pituitary adrenal axis model reveals bistability. *Theoretical Biology in Medical Modeling*, 4(8), 14 Feb 2007. doi:10.1186/1742-4682-4-8.

[61] Douglas Hanahan and Robert A. Weinberg. The hallmarks of cancer. *Cell*, 100 (1):57–70, 7 Jan 2000. doi:10.1016/S0092-8674(00)81683-9.

[62] Douglas Hanahan and Robert A. Weinberg. Hallmarks of cancer: the next generation. *Cell*, 144(5):646–674, 4 Mar 2011. doi:10.1016/j.cell.2011.02.013.

[63] Haralambos Hatzikirou and Andreas Deutsch. *Cellular automata as microscopic models of cell migration in heterogeneous environments*, volume 81 of *Multiscale Modeling of Developmental Systems*, chapter Current Topics in Developmental Biology, pages 401–434. Academic Press. ISBN:978-0-12-374253-7.

[64] Matthew G. Vander Heiden, Lewis C. Cantley, and Craig B. Thompson. Understanding the Warburg effect: the metabolic requirements of cell proliferation. *Science*, 324:1029–1033, 22 May 2009. doi:10.1126/science.1160809.

[65] Michael Höckel, Karlheinz Schlenger, Billur Aral, Margarite Mitze, Uwe Schäffer, and Peter Vaupel. Association between tumor hypoxia and malignant progression in advanced cancer of the uterine cervix. *Cancer Research*, 56:4509–4515, 1 Oct 1996.

[66] Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

[67] Frank C. Hoppensteadt. *Mathematical Methods for Analysis of a Complex Disease*, volume 22 of *Courant Lecture Notes*. American Mathematical Society and the Courant Institute of Mathematical Sciences, 2011.

[68] Peggy P. Hsu and David M. Sabatini. Cancer cell metabolism: Warburg and beyond. *Cell*, 134:703–707, 5 Sep 2008. doi:10.1016/j.cell.2008.08.021.

[69] Jie Hu, Jason W. Locasale, Jason H. Bielas, Jacintha O'Sullivan, Kieran Sheahan, Lewis C. Cantley, Matthew G. Vander Heiden, and Dennis Vitkup. Heterogeneity of tumor-induced gene expression changes in the human metabolic network. *Nature Biotechnology*, AOP:1–8, 21 Apr 2013. doi:10.1038/nbt.2530.

[70] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, 2 edition, 2004.

[71] Stephan Hutterer, Gerald Zauner, Marlene Huml, Rene Silye, and Kurt Schilcher. Data mining techniques for AFM-based tumor classification. In *2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 105–111, 9-12 May 2012. doi:10.1109/cibcb.2012.6217218.

[72] Mohit Jain, Roland Nilsson, Sonia Sharma, Nikhil Madhusudhan, Toshimori Kitami, Amanda L. Souza, Ran Kafri, Marc W. Kirschner, Clary B. Clish,

and Vamsi K. Mootha. Metabolite profiling identifies a key role for glycine in rapid cancer cell proliferation. *Science*, 336:1040–1044, 25 May 2012. doi:10.1126/science.1218595.

[73] Y. Jamali, M. Azimi, and M.R.K. Mofrad. A sub-cellular viscoelastic model for cell population mechanics. *PLoS ONE*, 5(8:e12097), 2010. doi:10.1371/journal.pone.0012097.

[74] W. James and C. Stein. Estimation with quadratic loss. In *Proceedings of the Berkeley Symposium on Mathematical Statistical Probability*, pages 316–379, 1961.

[75] H. Jeffreys. *Theory of Probability*. Clarendon Press, Oxford, 1961.

[76] Junhwan Jeon, Vito Quaranta, and Peter T. Cummings. An off-lattice hybrid discrete-continuum model of tumor growth and invasion. *Biophysical Journal*, 98(1):37–47, 6 Jan 2010. doi:10.1016/j.bpj.2009.10.002.

[77] Sumit K. Jha, Edmund M. Clarke, Christopher J. Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A Bayesian approach to model checking biological systems. In P. Degano and R. Gorrieri, editors, *Computational Methods in Systems Biology 2009 (LNBI 5688)*, pages 218–234. Springer-Verlag Berlin Heidelberg, 31 Aug–1 Sep 2009. http://www.cmsb09.cs.unibo.it/.

[78] Russell G. Jones and Craig B. Thompson. Tumor suppressors and cell metabolism: a recipe for cancer growth. *Genes & Development*, 23:537–548, 2009. doi:10.1101/gad.1756509.

[79] Caroline Jose, Nadège Bellance, and Rodrigue Rossignol. Choosing between glycolysis and oxidative phosphorylation: a tumor's dilemma. *Biochimica et Biophysica Acta*, 1807:552–561, 2011. doi:10.1016/j.bbabio.2010.10.012.

[80] William G. Kaelin Jr. and Craig B. Thompson. Clues from cell metabolism. *Nature*, 465:562–564, 3 Jun 2010.

[81] Jurre J. Kamphorst, Justin R. Cross, Jing Fan, Elisa de Stanchina, Robin Mathew, Eileen P. White, Craig B. Thompson, and Joshua D. Rabinowitz. Hypoxic and Ras-transformed cells support growth by scavenging unsaturated fatty acids from lysophospholipids. *Proceedings of the National Academy of Sciences*, 110(22): 8882–8887, 28 May 2013. doi:10.1073/pnas.1307237110.

[82] Irina Kareva. Prisoner's dilemma in cancer metabolism. *PLoS ONE*, 6 (12:e28576):1–9, Dec 2011. doi:10.1371/journal.pone.0028576.

[83] Irina Kareva. Biological stoichiometry in tumor micro-environments. *PLoS ONE*, 8(1:e51844), 2013. doi:10.1371/journal.pone.0051844.

[84] Kelley M. Kennedy and Mark W. Dewhirst. Tumor metabolism of lactate: the influence and therapeutic potential for MCT and CD147 regulation. *Future Oncology*, 6(1):127–148, 2010. doi:10.2217/FON.09.145.

[85] B. Kerr, M.A. Riley, M.W. Feldman, and B.J.M Bohannan. Local dispersal promotes biodiversity in areal-life game of rock-paper-scissors. *Nature*, 418:171–174, 2002.

[86] Jon Kleinberg and Éva Tardos. *Algorithm Design*, chapter 6, pages 261–266. Addison Wesley, 2 edition, 2006.

[87] Willem H. Koppenol, Patricia L. Bounds, and Chi V. Dang. Otto Warburg's contributions to current concepts of cancer metabolism. *Nature Reviews Cancer*, 11:325–337, May 2011. doi:10.1038/nrc3038.

[88] Guido Kroemer and Jacques Pouyssegur. Tumor cell metabolism: cancer's Achilles' heel. *Cancer Cell*, 13:472–482, Jun 2008. doi:10.1016/j.ccr.2008.05.005.

[89] Kenneth A. Krohn, Jeanne M. Link, and Ralph P. Mason. Molecular imaging of hypoxia. *Journal of Nuclear Medicine*, 49(6):129S–148S, Jun 2008. doi:10.2967/jnumed.107.045914.

[90] A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. doi:10.2307/1910129.

[91] Mitchell A. Lazar and Morris J. Birnbaum. De-meaning of metabolism. *Science*, 336:1651–1652, 29 Jun 2012. doi:10.1126/science.1221834.

[92] Axel Legay and Benoît Delahaye. Statistical model checking: an overview. *arXiv*, 2010. arXiv:1005.1327 [cs.LO].

[93] David Legland, Kiên Kiêu, and Marie-Françoise Devaux. Computation of Minkowski measures on 2D and 3D binary images. *Image Analysis & Stereology*, 26:83–92, 2007.

[94] J.C. Leloup and A. Goldbeter. A model for circadian rhythms in Drosophila incorporating the formation of a complex between the PER and TIM proteins. *Journal of Biological Rhythms*, 13:70–87, 1998.

[95] David Lewis. *Convention. A Philosophical Study*. Harvard University Press, 1969.

[96] Jason W. Locasale. The consequences of enhanced cell-autonomous glucose metabolism. *Trends in Endocrinology and Metabolism*, 23(11):545–551, Nov 2012. doi:10.1016/j.tem.2012.07.005.

[97] Jason W. Locasale. Serine, glysine and one-carbon units: cancer metabolism in full circle. *Nature Reviews Cancer*, AOP:1–12, 4 Jul 2013. doi:10.1038/nrc3557.

[98] Jason W. Locasale and Lewis C. Cantley. Altered metabolism in cancer. *BMC Systems Biology*, 8(88):1–3, 2010.

[99] Alfred J. Lotka. Undamped oscillations derived from the law of mass action. *Journal of the American Chemical Society*, 42(8):1595–1599, 1920. doi:10.1021/ja01453a010.

[100] J. Lu, H.W. Engl, R. Machné, and P. Schuster. Inverse bifurcation analysis of a model for mammalian G1/S regulatory module BIRD '07. *Lecture Notes in Bioinformatics*, 4414:168–184, 2007. Berlin: Springer.

[101] Sophia Y. Lunt and Matthew G. Vander Heiden. Aerobic glycolysis: meeting the metabolic requirements of cell proliferation. *Annual Review of Cell Developmental Biology*, 27:441–464, 2011. doi:10.1146/annurev-cellbio-092910-154237.

[102] Costas A. Lyssiotis and Lewis C. Cantley. SIRT6 puts cancer metabolism in the driver's seat. *Cell*, 151(6):1155–1199, 7 Dec 2012. doi:10.1016/j.cell.2012.11.020.

[103] Yuri Mansury, Mark Diggory, and Thomas S. Deisboeck. Evolutionary game theory in an agent-based brain tumor: exploring the "genotype-phenotype" link. *Journal of Theoretical Biology*, 238:146–156, 2006.

[104] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953. doi:10.1063/1.1699114.

[105] Nicholas Metropolis and Stanislav Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[106] Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*, chapter 4, pages 101–105. Springer, 2 edition, 2004.

[107] K. Michielsen and H. De Raedt. Integral-geometry morphological image analysis. *Physics Reports*, 347:461–538, 2001.

[108] Donald M. Miller, Shelia D. Thomas, Ashraful Islam, David Muench, and Kara Sedoris. c-Myc and cancer metabolism. *Clinical Cancer Research*, 18(20):5546–5553, 15 Oct 2012. doi:10.1158/1078-0432.ccr-12-0977.

[109] Klaus Mosegaard and Albert Tarantola. Monte Carlo sampling of solutions to inverse problems. *Journal of Geophysical Research*, 100(B7):12431–12447, 1995.

[110] John Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:48–49, 1950.

[111] Nicholas Navin, Alexander Krasnitz, Linda Rodgers, Kerry Cook, Jennifer Meth, Jude Kendall, Michael Riggs, Yvonne Eberling, Jennifer Troge, Vladimir Grubor, Dan Levy, Pär Lundin, Susanne Månér, Anders Zetterberg, James Hicks, and Michael Wigler. Inferring tumor progression from genomic heterogeneity. *Genome Research*, 20(1):68–80, Jan 2010. doi:10.1101/gr.099622.109.

[112] *Cancer metabolomics: elucidating the biochemical programs that support cancer initiation and progression*, 3 Feb 2012. New York Academy of Sciences. http://www.nyas.org/CancerMetabolomics.

[113] Martin A. Nowak. *Evolutionary Dynamics: Exploring the Equations of Life*. Belknap Press of Harvard University Press, 2006.

[114] Martin A. Nowak. Five rules for the evolution of cooperation. *Science*, 314:1560–1563, 2006.

[115] Martin A. Nowak and Robert M. May. Evolutionary games and spatial chaos. *Nature*, 359:826–829, 1992.

[116] S. Osinsky, L. Bubnovskaya, I. Ganusevich, A. Kovelskaya, L. Gumenyuk, G. Olijnichenko, and S. Merentsev. Hypoxia, tumour-associated macrophages, microvessel density, VEGF and matrix metalloproteinases in human gastric cancer: interaction and impact on survival. *Clinical and Translational Oncology*, 13 (2):133–138, Feb 2011. doi:10.1007/s12094-011-0630-0.

[117] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[118] Stephanos Pavlides, Diana Whitaker-Menezes, Remedios Castello-Cros, Neal Flomenberg, Agnieszka K. Witkiewicz, Philippa G. Frank, Mathew C. Casimiro, Chenguang Wang, Paolo Fortina, Sankar Addya, Richard G. Pestell, Ubaldo E. Martinez-Outschoorn, Federica Sotgia, and Micheal P. Lisanti. The reverse Wwarburg effect. *Cell Cycle*, 8(23):3984–4001, 1 Dec 2009.

[119] Michael J. Plank and Matthew J. Simpson. Models of collective cell behavior with crowding effects: comparing lattice-based and lattice-free approaches. *Journal of the Royal Society Interface*, 9:2983–2996, 13 Jun 2012. doi:10.1098/rsif.2012.0319.

[120] C. Räth, R. Monetti, J. Bauer, I. Sidorenko, D. Uüller, M. Matsuura, E-M. Lochmüller, P. Zysset, and F. Eckstein. Strength through structure: visualization and local assessment of the trabecular bone structure. *New Journal of Physics*, 10:1–18, 1 Dec 2008. doi:10.1088/1367-2630/10/12/125010.

[121] Maurice Reimann, Clemens A. Schmitt, and Soyoung Lee. Non-cell-autonomous tumor suppression: oncogene-provoked apoptosis promotes tumor cell senescence via stromal crosstalk. *Journal of Molecular Medicine*, 89: 869–875, 19 May 2011. doi:10.1007/s00109-011-0770-2.

[122] Zachary J. Reitman and Hai Yan. Isocitrate dehydrogenase 1 and 2 mutations in cancer: alterations at a crossroads of cellular metabolism. *Journal of the National Cancer Institute*, 102(13):932–941, 7 Jul 2010. doi:10.1093/jnci/djq197.

[123] Osbaldo Resendis-Antonio, Alberto Checa, and Sergio Encarnación. Modeling core metabolism in cancer cells: surveying the topology underlying the Warburg effect. *PLoS ONE*, 5(8:e12383):1–11, Aug 2010. doi:10.1371/journal.pone.0012383.

[124] R.S. Root-Bernstein and M.I. Bernstein. A simple stochastic model of development and carcinogenesis. *Anticancer Research*, 19(6(A)):4869–4876, 1999. PMID:10697600.

[125] Waldir L. Roque, Antonio Carlos A. de Souza, and Denis X. Berbieri. The Euler-poincaré characteristic applied to identify low bone density from vertebral tomographic images. *Brazilian Journal of Rheumatology*, 49(2):146–152, 2009.

[126] Michael K. Samoszuk, James Walter, and Eugene Mechetner. Improved immunohistochemical method for detecting hypoxia gradients in mouse tissues and tumors. *Journal of Histochemistry & Cytochemistry*, 52(6):837–839, 2004. doi:10.1369/jhc.4b6248.2004.

[127] Carlos Sebastián, Bernadette M.M. Zwaans, Dafne M. Silberman, Melissa Gymrek, Alon Goren, Lei Zhong, Oren Ram, Jessica Truelove, Alexander R. Guimaraes, Debra Toiber, Claudia Cosentino, Joel K. Greenson, Alasdair I. MacDonald, Liana McGlynn, Fraser Maxwell, Joanne Edwards, Sofia Giacosa, Ernesto Guccione, Ralph Weissleder, Bradley E. Bernstein, Aviv Regev, Paul G. Shields, David B. Lombard, and Raul Mostoslavsky. The histone deacetylase

SIRT6 is a tumor suppressor that controls cancer metabolism. *Cell*, 151(6):1185–1199, 7 Dec 2012. doi:10.1016/j.cell.2012.10.047.

[128] Gregg L. Semenza. Tumor metabolism: cancer cells give and take lactate. *Journal of Clinical Investigation*, 118(12):3835–3837, Dec 2008. doi:10.1172/JCI37373.

[129] Gregg L. Semenza. HIF-1: upstream and downstream of cancer metabolism. *Current Opinions in Genetics and Development*, 20:51–56, 2010. doi:10.1016/j.gde.2009.10.009.

[130] Gregg L. Semenza. Defining the role of hypoxia-induced factor 1 in cancer biology and therapeutics. *Oncogene*, 29:625–634, 2010. doi:10.1038/onc.2009.441.

[131] Ashish Sen and Muni Srivastava. *Regression Analysis: Theory, Methods, and Applications*, chapter 12, pages 253–262. Springer, 1990.

[132] M. Celeste Simon, editor. *Diverse Effects of Hypoxia on Tumor Progression*, volume 345. Springer, 2010. ISBN:978-3-642-13329-9.

[133] Brian Skyrms. Signals: Evolution, learning, and information. *Oxford University Press*, 2010.

[134] J. Maynard Smith and J.R. Price. The logic of animal conflict. *Nature*, 246:15–18, 1973. doi:10.1038/246015a0.

[135] John Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[136] Pierre Sonveaux, Frédérique Végran, Thies Schroeder, Melanie C. Wergin, Julien Verrax, Zahid N. Rabbani, Christophe J. De Saedeleer, Kelley M. Kennedy, Caroline Diepart, Bénédicte F. Jordan, Michael J. Kelley, Bernard

Gallez, Miriam L. Wahl, Olivier Feron, and Mark W. Dewhirst. Targeting lactate-fueled respiration selectively kills hypoxic tumor cells in mice. *Journal of Clinical Investigation*, 118(12):3930–3942, Dec 2008. doi:10.1172/JCI36843.

[137] Federica Sotgia, Ubaldo E. Martinez-Outschoorn, Stephanos Pavlides, Anthony Howell, Richard G. Pestell, and Micheal P. Lisanti. Understanding the Warburg effect and the prognostic value of stromal caveolin-1 as a marker of a lethal tumor microenvironment. *Breast Cancer Research*, 13(213):1–13, 2011.

[138] C. Stein. Estimation of the parameters of a multivariate normal distribution: I. estimation of the means. *Annals of Statistics*, 9:1135–1151, 1981.

[139] Andrew Sundstrom, Silvio Cirrone, Salvatore Paxia, Carlin Hsueh, Rachel Kjolby, James K. Gimzewski, Jason Reed, and Bud Mishra. Image analysis and length estimation of biomolecules using AFM. *IEEE Transactions on Information Technology in Biomedicine*, 16(6):1200–1207, Nov 2012. doi:10.1109/titb.2012.2206819.

[140] Masaru Tomita and Kenjiro Kami. Systems biology, metabolomics, and cancer metabolism. *Science*, 336:990–991, 25 May 2012. doi:10.1126/science.1223066.

[141] I.P.M. Tomlinson. Game-theory models of interactions between tumour cells. *European Journal of Cancer*, 33(9):1495–1500, 1997.

[142] I.P.M. Tomlinson and W.F. Bodmer. Modelling the consequences of interactions between tumour cells. *British Journal of Cancer*, 75(2):157–160, 1997.

[143] Alan M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B*, 327:37–72, 1952.

[144] Mahesh A. Varia, Dennise P. Calkins-Adams, Lillian H. Rinker, Andrew S. Kennedy, Debra B. Novotny, Wesley C. Fowler Jr., and James A. Raleigh. Pimonidazole: a novel hypoxia marker for complementary study of tumor hypoxia and cell proliferation in cervical carcinoma. *Gynecologic Oncology*, 71 (GO985163):270–277, 1998.

[145] Peter Vaupel and Arnulf Mayer. Hypoxia in cancer: significance and impact on clinical outcome. *Cancer Metastasis Review*, 26:225–239, 2007. doi:10.1007/s10555-007-9055-1.

[146] Alexei Vazquez, Jiangxia Liu, Yi Zhou, and Zoltán N. Oltvai. Catabolic efficiency of aerobic glycolysis: the Wwarburg effect revisited. *BMC Systems Biology*, 4(58):1–9, 2010.

[147] Vito Volterra. Fluctuations in the abundance of a species considered mathematically. *Nature*, 118:558–560, 1926. doi:10.1038/118558a0.

[148] John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[149] Ching-Wei Wang, Dean Fennell, Ian Paul, Kienan Savage, and Peter Hamilton. Robust automated tumour segmentation on histological and immunohistochemical tissue images. *PLoS ONE*, 6(2:e15818), 2011. doi:10.1371/journal.pone.0015818.

[150] O. Warburg. On the origins of cancer cells. *Science*, 123:309–314, 1956.

[151] O. Warburg, K Posener, and E. Negelein. Ueber den stoffwechsel der tumoren. *Biochem. Z.*, 152(3):319–344, 1924.

[152] Patrick S. Ward and Craig B. Thompson. Metabolic reprogramming: a cancer hallmark even Warburg did not anticipate. *Cancer Cell*, 21:297–308, 20 Mar 2012. doi:10.1016/j.ccr.2012.02.014.

[153] Robert A. Weinberg. *The Biology of Cancer*. Garland Science, 2006.

[154] Jung whan Kim and Chi V. Dang. Cancer's molecular sweet tooth and the Warburg effect. *Cancer Research*, 66(18):8927–8930, 15 Sep 2006. doi:10.1158/0008-5472.can-06-1501.

[155] Chris Wijns, Fabio Boschetti, and Louis Moresi. Inverse modelling in geology by interactive evolutionary computation. *Journal of Structural Geology*, 25:1615–1621, 2003.

[156] David R. Wise and Craig B. Thompson. Glutamine addiction: a new therapeutic target in cancer. *Trends in Biochemical Sciences*, 35(8):427–433, 2010. doi:10.1016/j.tibs.2010.05.003.

[157] David R. Wise, Ralph J. DeBerardinis, Anthony Mancuso, Nabil Sayed, Xiao-Yong Zhang, Harla K. Pfeiffer, Ilana Nissim, Evgueni Daikhin, Marc Yudkoff, Steven B. McMahon, and Craig B. Thompson. Myc regulates a transcriptional program that stimulates mitochondrial glutaminolysis and leads to glutamine addiction. *Proceedings of the National Academy of Sciences*, 105(48):18782–18787, 2 Dec 2008. doi:10.1073/pnas.0810199105.

[158] Dominik Wodarz and Natalia L. Komarova. *Computational Biology of Cancer: Lecture Notes and Mathematical Modeling*. World Scientific, 2005.

[159] Joao B. Xavier, Cristian Picioreanu, and Mark C. M. van Loosdrecht. A framework for multidimensional modeling of activity and structure of multispecies

biofilms. *Environmental Microbiology*, 7(8):1085–1103, 2005. doi:10.1111/j.1462-2920.2005.00787.x.

[160] Haoqiang Ying, Alec C. Kimmelman, Costas A. Lyssiotis, Sujun Hua, Gerald C. Chu, Eliot Fletcher-Sananikone, Jason W. Locasale, Jaekyoung Son, Heilei Zhang, Jonathan L. Coloff, Haiyan Yan, Wei Wang, Shujuan Chen, Adrea Viale, Hongwu Zheng, Ji hye Paik, Carol Lim, Alexander R. Guimaraes, Eric S. Martin, Jeffrey Chang, Aram F. Hezel, Samuel R. Perry, Jian Hu, Boyi Gan, Yonghong Xiao, John M. Asara, Ralph Weissleder, Y. Alan Wang, Lynda Chin, Lewis C. Cantley, and Ronald A. DePinho. Oncogenic Kras maintains pancreatic tumors through regulation of anabolic glucose metabolism. *Cell*, 149:656–670, 27 Apr 2012. doi:10.1016/j.cell.2012.01.058.

[161] Jin Zhang, Esther Nuebel, George Q. Daley, Carla M. Koehler, and Michael A. Teitell. Metabolic regulation in pluripotent stem cells during reprogramming and self-renewal. *Cell Stem Cell*, 11:589–595, 2 Nov 2012. doi:10.1016/j.stem.2012.10.005.

[162] Paolo Zuliani, André Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to Stateflow/Simulink verification. In *International Conference on Hybrid Systems: Computation and Control 2010*, pages 243–252, 12–15 Apr 2010. http://hscc10.it.uu.se/.