2016 IEEE 55th Conference on Decision and Control (CDC)
ARIA Resort & Casino
December 12-14, 2016, Las Vegas, USA

Pattern Synthesis in a 3D Agent-Based Model of Stem Cell Differentiation

Demarcus Briers, Iman Haghighi, Douglas White, Melissa L. Kemp, Calin Belta

***Abstract*— Embryonic stem cells (ESC) are generally regarded as the smallest functional units necessary to reproduce multicellular systems such as tissues and organs. Recent work showed that agent-based models of the stochastic dynamics of locally interacting stem cell agents accurately capture the time-dependent distributions of a variety of spatial patterns. Starting from a 3-dimensional, local interaction model of ESC proliferation and differentiation, in this paper, we developed a pattern classification and parameter optimization approach to maximize the occurrence of desired morphogenic patterns. Our approach uses Particle Swarm Optimization (PSO) and a pattern classification method that exploits a quantitative characterization of pattern formation. Since patterning likely imprints subsequent choices that stem cell aggregates make in lineage specification (e.g. precursors of neurons, lung cells, or muscle cells), our parameter optimization approach can be used to synthesize global differentiation patterns through local cues.**

I. Introduction

There has been an increasing effort in recent years to reproduce the microenvironment of self-renewing biological processes such as embryogenesis [1], [2], tissue development [3], and organoid formation [4] by forcing stem cells to aggregate into 3-dimensional spheroids known as embryoid bodies (EBs). Similar to the self-renewing stem cells residing in its *in vivo* niche, EBs undergo patterned differentiation controlled by local cues [5], [6].

Instead of modeling this patterned differentiation at a global level, the authors of [7], [1] developed an agent-based model that captures proliferation and spatio-temporal patterning of EB differentiation using local interaction rules. They also developed quantitative network metrics to verify their agent-based model captures a distribution of patterns observed experimentally. These pattern classes describe differentiation occurring from the outside-in, the inside-out, in globular (spheroid) subclusters, or in snake-like subclusters.

Even though [1] was able to accurately recapitulate the distribution of pattern trajectories in EBs, their exhaustive search of parameter combinations becomes computationally expensive as the number of parameters increases. Therefore, we employ an alternate approach that quantifies how strongly the spatial behavior of a dynamical system resembles a global pattern. Then, we use this quantifier as the basis for an optimization procedure to synthesize model parameters.

This work was partially supported by the National Science Foundation under grants CBET-0939511 and NSF CNS-1446607.

Demarcus Briers (dbriers@bu.edu) is with the Bioinformatics Program, Iman Haghighi (haghighi@bu.edu), and Calin Belta (cbelta@bu.edu) are with the Division of Systems Engineering at Boston University, Boston, MA, USA. Douglas White (dwhite48@gatech.edu) and Melissa Kemp (melissa.kemp@bme.gatech.edu) are with Georgia Institute of Technology, Atlanta, GA, USA.

There has been an increasing effort in recent years to use formal logics as descriptors of spatial properties. Linear spatial superposition logic (LSSL) was successfully used in [8] to identify self-similar texture. Richer spatial and spatio-temporal logics were introduced in [9], [10] with the capability of describing complex patterns in networked or distributed dynamical systems.

This paper is closely related to [9], [10]. In [9], tree spatial superposition logic (TSSL) is defined and used to enforce the emergence of steady state Turing patterns in a biochemical reaction-diffusion system. In [10], spatial temporal logic (SpaTeL) was introduced to formally express time-varying spatial patterns. Both these logics are equipped with quantitative semantics, which can be used as a robustness score for how a dynamical system satisfies desired specifications. Using these formal quantification tools, in this paper, we develop a heuristic optimization procedure in which we run simulations with interactions at the local level, make quantitative spatio-temporal observations at the global level, and select parameter values with the highest robustness score. We show that simulations produced by parameter values resulting from this framework closely resemble the predefined patterns observed experimentally.

In Sec. II, we describe an agent-based model characterizing state changes in embryoid bodies. This model is a modified version of the model introduced in [7]. Although our model uses the same rules to enforce stem cell differentiation, we utilize more efficient algorithms for growing cell populations, identifying neighbors, and visualizing global behaviors. These modifications decrease the simulation run time and increase the pattern classification accuracy. In Sec. III, the formal methods approach used to synthesize model parameters is presented. Sec. IV presents our final conclusions and a few directions for future work.

II. Stochastic Agent-Based Model

The purpose of performing parameter optimization using an agent-based model of 3D stem cell differentiation is to explore the range of patterns that emerge from local interaction rules. This includes synthesizing experimentally observed patterns or even *de novo* patterns that were not observed experimentally. This enables biologists to better understand the local mechanisms governing morphogenesis (e.g. paracrine signaling) and efficiently test assumptions about these mechanisms.

In this paper we assume local signaling regulates stem cell differentiation in EBs. We also assume differentiation can be described as the binary classification that captures

if a cell has lost the ability to specialize into any cell type (*loss of pluripotency*)[11]. Differentiation can describe any transition down the hierarchy of stem cell specialization [5], but this binary classification, Fig. 1a and Fig. 1c, allows us to investigate early cell fate decisions that likely influence tissue and organ development. Using the *loss of pluripotency* indicator, the protein Oct4, [7] has shown that there is a time-dependent distribution of intermediate patterns (Fig. 1b) before complete differentiation (Fig. 1c) in EBs.

The proposed stochastic agent-based model is a modified version of the approach described in [7], [1], where it is assumed that stem cell differentiation is dictated by basal stochasticity, local negative feed forward signaling, and positive feedback signaling. Our modeling approach uses the same set of local differentiation rules. However, two main modifications were made. First, we use a more efficient approach (KD-trees) to identify a cell's neighborhood. Second, EBs are analyzed as transparent cross-sectional images. This approach is more in line with images that are produced experimentally, such as Fig. 1, where cells within a certain distance from cross-sections of EBs are observable. This visualization approach allows for more accurate pattern classification in images, as demonstrated in Sec. III.

The agent-based model of stem cell differentiation used in this paper is described in detail in the rest of this section.

A. Description of Stem Cell Agents

Consider a network of $N(t)$ locally interacting stem cells in which each cell is labeled by an integer $i \in \{1, \cdots, N(t)\}$, where $N(t)$ is the number of stem cells at time $t \in \{0, \cdots, T\}$ and T is the earliest time at which all cells have differentiated. We represent this network of stem cells as a graph $G(t) = (V(t), E(t))$, where the vertex $i \in V(t)$ represents the ith cell at time t and $(i, j) \in E(t)$ if the corresponding cells i and j are interacting neighbors at time t.

Asynchronous Cell Division: A *primary* cell $k \in \{1, \cdots, N(0)\}$, which is present at the start of a simulation, initially divides at a random time $t_1^k \in [0, \delta]$, where δ is the length of the cell cycle for a single cell. The set of time points that a primary cell $k \in \{1, \cdots, N(0)\}$ or its daughters divide is denoted by $D_k = \{t_1^k, t_2^k, \cdots, t_M^k\}$, where

$$\begin{aligned} 0 \le t_m^k \le T \quad & \forall m \in \{1, \cdots, M\} \\ t_{m+1}^k - t_m^k = \delta \quad & \forall m \in \{1, \cdots, M-1\}, \end{aligned} \tag{1}$$

where m is the mth division, and $M$is the final division of a cell during a simulation. D_k is interpreted as the set of asynchronous division times for the initial population of stem cells. Since experiments have shown that undifferentiated stem cells divide in approximately 19 hours [7], we assume that $\delta = 19$ in embryonic stem cells.

The set of new cells introduced to the system at time t as a result of cell division is denoted by

$$NEW(t) = \{N(t-1) + 1, \cdots, N(t)\}. \tag{2}$$

After each division, one daughter cell replaces the parent cell with the same label and the other daughter cell is placed adjacently in a random direction. We define a mapping $\mathcal{P}: NEW(t) \to V(t-1)$ where $\mathcal{P}(i)$ is the parent cell from which $i \in NEW(t)$ divided.

Cell Neighborhood: Each cell is assumed to be a sphere, with the center located at $l_i(t) \in \mathbb{R}^3$ for $i \in \{1, \cdots, N(t)\}$ and constant radius r. Cell radius is assumed to be constant since changes in the size of a cell over time are negligible [7]. We also define the set of all cell locations at t as

$$L(t) = \{l_1(t), l_2(t), \cdots, l_{N(t)}(t)\}. \tag{3}$$

The Euclidean distance between two cells is denoted by $d(l_i(t), l_j(t))$. Two cells i,j are considered neighbors if they are in contact with one another. Therefore, we construct the set of graph edges $E(t)$ such that

$$(i, j) \in E(t) \subseteq V(t) \times V(t) \iff d(l_i(t), l_j(t)) \le 2r.$$

The neighborhood of cell i is denoted by $e_i(t)$.

$$e_i(t) = \{j \in \{1, \cdots, N(t)\} \mid (i, j) \in E(t)\}. \tag{4}$$

Looping through every pair of vertices in $V(t)$ to construct $E(t)$ is exponential in $|V(t)|$. Instead we use a space partitioning tree, KD-tree, to find neighboring cells. A KD-Tree is a special case of the binary space partitioning tree, which is used to partition points in a k-dimensional space, find nearest neighbors, and perform range queries [12].

B. Structural Constraints in 3D

EB's are an *in vitro* system used to study stem cell differentiation in 3D [13], [2]. To accurately capture the mechanical forces of an EB, a mass-spring model is employed. Like the model in [7], 1000 cells is the size of the initial population. The radius, circularity, average connection length, average connection number, and density of the simulated EB's show these simulations resemble *in vitro* EBs [7]. This resulting structure represents the set of primary cells at the initial time ($t = 0$).

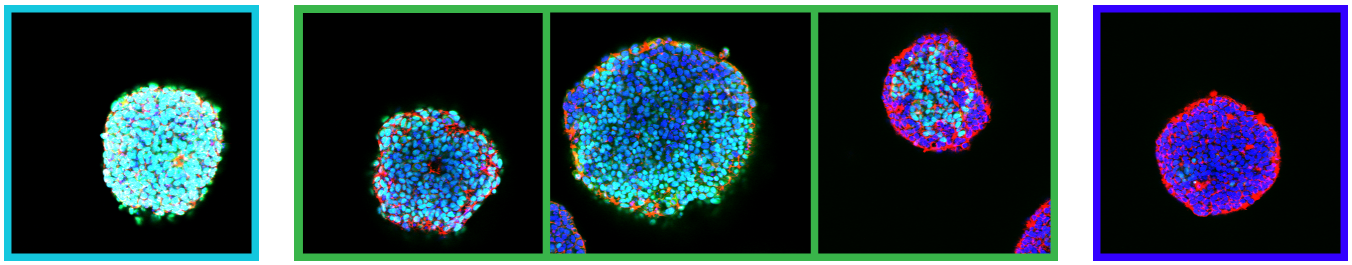
Collisions and Interactions: Cells are represented as rigid spheres connected by springs. The rigid spheres model the incompressibility of the cell nucleus, and the springs represent the malleability of the cytoplasm. We enforce rigid sphere behavior for every pair of interacting cells.

$$d(l_i(t), l_j(t)) \ge 2r, \quad \forall (i, j) \in E(t). \tag{5}$$

Equation (5) can be violated as a result of population growth. When cell i divides, a new cell j is randomly placed in the network such that $d(l_i(t), l_j(t)) = 2r$. This can cause cell j to overlap with some of the cells surrounding i. If the constraint in Equation (5) is violated, then distance corrections F_{ij} for the pair of cells (i, j) are determined using the equation:

$$F_{ij} = -\mathcal{X} k_c, \tag{6}$$

where k_c is a constant that represents the strength of the interaction, and $\mathcal{X}$ represents the amount of displacement between the cytoplasm of two cells ($\mathcal{X} = d(l_i(t), l_j(t)) - 2r$). This interaction system is solved iteratively until all pairs of cells satisfy Equation 5.



(a) All Pluripotent (b) Transitioning Patterns (c) All Differentiated

Fig. 1: Confocal microscopy multichannel images of EBs stained with DAPI (nuclear stain, blue), phalloidin (red), and Oct4 (cyan) shown at a depth of 25 um for EBs. (a) Early stem cell populations are primarily pluripotent and uniformly express Oct4 (cyan). (b) Over multiple days, a distribution of differentiation patterns are observed before every cell is differentiated into an Oct4-negative status. These patterns are labeled as inside-out (left),globular (center),and outside-in(right). (c) After 7 days most stem cells have differentiated into specialized cell lineages (no Oct4, loss of cyan).

C. Stochastic Rules for State Changes

We chose to model stem cell differentiation as a 3-state process with differentiation controlled by cell division [14]. The state of cell i at time $t \in \{0, \cdots, T\}$ is denoted by $x_i(t) \in \{\mathcal{U}, \mathcal{T}, \mathcal{D}\}$, where $x_i(t) = \mathcal{U}$ if cell i is undifferentiated, $x_i(t) = \mathcal{T}$ if cell i is transitioning, and $x_i(t) = \mathcal{D}$ if a cell i is differentiated at time t.

At $t = 0$, every cell is initialized in state $\mathcal{U}$. At each time step of the simulation, three *stochastic rules* regulate the probability that a cell will differentiate. Biologically, these stochastic rules represent basal stochasticity (random rule), the influence of undifferentiated neighbors to cause a cell to change state (negative feed forward), and the influence of differentiated neighbors to cause a cell to change state (positive feedback). These rules are derived in [7], [1] and were motivated by [15].

Rule 1: (Basal Differentiation) The probability that a cell i changes its state from $\mathcal{U}$ to $\mathcal{T}$, regardless of the state of its neighbors, is given by:

$$p_1^i = \alpha, \tag{7}$$

where $\alpha \in (0, 1)$ is a constant.

Rule 2: (Local Negative Feedforward) The probability that undifferentiated neighboring cells cause a cell i to change its state from $\mathcal{U}$ to $\mathcal{T}$ is given by:

$$p_2^i = \frac{1}{1 + (U_{norm,i}/k_1)^{n_1}}, \tag{8}$$

where k_1 and n_1 are tuning parameters and $U_{norm,i}$ is defined as the percentage of neighboring cells of cell i that are undifferentiated or transitioning.

Rule 3: (Local Positive Feedback) The probability that differentiated neighboring cells cause a cell i to change its state from $\mathcal{U}$ to $\mathcal{T}$ is given by:

$$p_3^i = \frac{D_{norm,i}^{n_2}}{k_2^{n_2} + D_{norm,i}^{n_2}}, \tag{9}$$

where k_2 and n_2 are tuning parameters and $D_{norm,i}$ is defined as the percentage of neighboring cells of cell i that are differentiated.

Three independent random variables $\{r_1^i, r_2^i, r_3^i\}$ uniformly distributed between 0 and 1 are assigned to each cell i, $i \in \{1, \cdots, N(t)\}$. For a cell i in state $x_i(t) = \mathcal{U}$, the state of the cell is set to $\mathcal{T}$ until the next cell division if

$$r_1^i < p_1^i \vee r_2^i < p_2^i \vee r_3^i < p_3^i \tag{10}$$

evaluates to true. A cell at state $x_i(t) = \mathcal{T}$ will change its state to $\mathcal{D}$ when the next cell division occurs. This mimics the notion that cell division cycles control stem cell differentiation [14]. It important to realize that Equations 8 and 9 resemble Hill equations, and their respective tuning parameters influence how many neighboring cells in a specific state will cause a cell to differentiate. For all cells that are differentiated ($x_i = \mathcal{D}$), the division time δ is approximately 51 hours [7].

D. Visualization

The model described above was implemented in Python as a simulation tool that receives a list of parameters

$$\Pi = (\alpha, k_1, n_1, k_2, n_2) \in \Omega, \tag{11}$$

where

$$\Omega = \omega_\alpha \times \omega_{k_1} \times \omega_{n_1} \times \omega_{k_2} \times \omega_{n_2}, \tag{12}$$

and $\omega_j \subseteq \mathbb{R}$ is an allowed interval for parameter j. For a given set of parameter values Π, the simulator produces a sequence of images $Y_t(\Pi)$, where $t \in \{0, 1, \cdots, T\}$ denotes the time point in hours. Fig. 2 illustrates the output produced by a simulation. The simulator produces red spheres for cells in undifferentiated or transitioning state ($x_i(t) \neq \mathcal{D}$) and black spheres for differentiated cells ($x_i(t) = \mathcal{D}$).

Although cells were located in a 3 dimensional space, we visualized differentiation at each time point t as a transparent cross-sectional image. Transparent 2D images allowed us to accurately classify patterns in images without the computational complexity required to represent 3D images. In cross-sectional experimental images (Fig. 1), undifferentiated cells appear cyan and differentiated cells appear blue. The choice of red and black instead of cyan and blue was made in order to create a more significant contrast between the

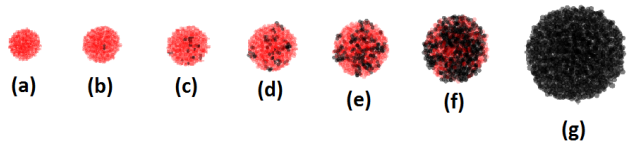


Fig. 2: Sample images from a simulation for the parameter set $\Pi = (0.01, 0.1, 25, 0.3, 25)$ at time (a) $t = 0$ (b) $t = 5$ (c) $t = 10$ (d) $t = 15$ (e) $t = 20$ (f) $t = 30$ (g) $t = 60$. Cells are colored red when in the undifferentiated state, and black upon differentiation.

RGB concentrations of differentiated and undifferentiated cells, making it easier to learn pattern classifiers.

III. Optimizing 3D Spatio-temporal Patterns

In this section, we explore the parameter set $\Pi \in \Omega$, Eq. (11), in the agent-based model of Sec. II with the purpose of determining valuations such that the simulations of the model produce images that best resemble specific predetermined patterns. Specifically, we are interested in inside-out, outside-in, and globular patterns similar to Fig. 1b.

Since *in silico* simulations of the model are very time-consuming, (approximately one hour on average on a machine with a 2.4GHz core i7 CPU and 8 GB RAM), it is impractical to explore the parameter space with a brute-force approach. Instead, we implement the formal methods approach presented in [9] and [10] to solve this problem. In this approach, desirable patterns are formally specified by logical formulas. These formulas are then used to analyze the behavior of a dynamical system and synthesize desired spatio-temporal behaviors.

A. Quad-tree Representation of Spatial Behaviors

Consider an RGB representation of an $m \times n$ image as the matrix A where the element $a_{ij} = \langle a_{ij}^{(r)}, a_{ij}^{(g)}, a_{ij}^{(b)} \rangle$ is the normalized RGB values for the pixel located on the ith row and jth column of the image. Thus,

$$0 \leq a_{ij}^{(c)} \leq 1 \text{ for } c \in \{r, g, b\}.$$

Given a matrix A, $A[i_s, i_e; j_s, j_e]$ is used to denote the submatrix created by selecting rows with indices from i_s to i_e and columns from j_s to j_e. Following [9], we represent the matrix A as a quad-tree defined as follows, A quad-tree $Q = (\mathcal{V}, R)$ is a quaternary tree [16] representation of matrix A where each vertex $v \in \mathcal{V}$ represents a submatrix of A and the relation $R \subset \mathcal{V} \times \mathcal{V}$ defines four children for each vertex that is not a leaf.

Fig. 3 demonstrates how a quad-tree is built from a matrix. In this figure, we label each edge in the quad-tree with the direction of the sub-matrix represented by the child: north west (NW), north east (NE), south west (SW), and south east (SE). In Fig. 3(b), v_0 represents the complete matrix A. v_1 represents $A[1, \lfloor m/2 \rfloor; 1, \lfloor n/2 \rfloor]$, where m is the total number of rows and n is the total number of columns in A. v_5 represents $A[1, \lfloor m/4 \rfloor; 1, \lfloor n/4 \rfloor]$, etc. The construction of

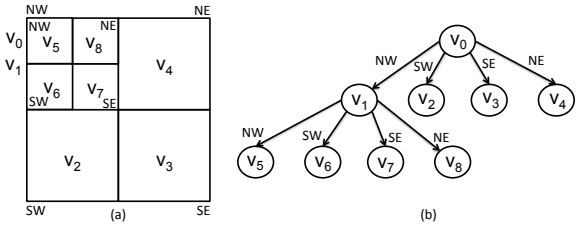


Fig. 3: Quad-tree representation (b) of a matrix (a)

a quad-tree in this paper slightly differs from [9]. In [9], the assumption is made that A has a size of $2^k \times 2^k$ so that each submatrix can be divided into four equal-sized partitions. Here, we have relaxed this requirement by allowing non-equal submatrices to be children of a node. Furthermore, [9] defines a leaf as a vertex of the quad-tree for which all the elements of a submatrix have the same values. While this approach works perfectly for the 32×32 network that is studied in that paper, it can be problematic for larger images since the number of vertices in a quad-tree grows exponentially as more levels are added to it. In this paper, we construct quad-trees with a fixed depth d, regardless of the size and other characteristics of A. A *representation function* $\mu^{(c)}(v) : \mathcal{V} \to [0, b] \times [0, b]$ for sub-matrix $A[i_s, i_e; j_s, j_e]$ represented by vertex $v \in \mathcal{V}$ of the quad-tree $Q = (\mathcal{V}, R)$ is a function that provides the mean value and variance for the concentration of RGB colors in a particular region of the space represented by the vertex v. where $c \in \{r, g, b\}$ is an RGB color. The function $\mu^{(c)}$ provides the mean value and variance for the concentration of RGB colors in a particular region of the space represented by the vertex v.

We use quad-trees to analyze spatial patterning in images. In the rest of this section, we assume that any given image derived from the simulator of Sec. II ($Y_t(\Pi)$) is translated into a corresponding quad-tree ($Q_t(\Pi)$).

B. Formal Specification of Global Patterns

In [9], the logic, called tree spatial superposition logic (TSSL), introduced that is capable of formally specifying patterns in a network of locally interacting agents. They show that this logic is sophisticated enough to describe complicated patterns such as Turing patterns in biochemical reaction-diffusion systems. We use this logic to express inside-out, outside-in, and globular patterns in the model of Sec. II. First, we present a brief introduction to TSSL. Refer to [9] for a thorough explanation of this logic, definitions of syntax and semantics, and its properties. A TSSL formula is recursively constructed using the following:

- Linear predicates over valuations for the representation function. e.g., $\mu_1^{(r)} > \lambda_1$, $\mu_1^{(b)} < \lambda_2$.
- Boolean operators: e.g., $\neg\phi$, $\phi_1 \wedge \phi_2$, and $\phi_1 \vee \phi_2$.
- Spatial operators. e.g., $\exists_B \bigcirc \phi$, $\forall_B \bigcirc \phi$, where B is a nonempty subset of the set of directions $\{NW, NE, SW, SE\}$.

The spatial operators $\exists_B \bigcirc$ and $\forall_B \bigcirc$ are read as *there exists in directions B next* and *for all directions B next*,

Pattern	Positive learning images	Negative learning images	Positive testing images	Negative testing images
Outside-in	6000	22000	2000	6000
Inside-out	6000	22000	2000	6000
Globular	8000	20000	2000	6000

TABLE I: Number of samples generated to learn and test TSSL classifiers

respectively. $\exists_B \bigcirc \phi$ is interpreted as follows: For at least one of the nodes located in the next level of the quad-tree labeled with one of the directions in B, ϕ must be satisfied. $\forall_B \bigcirc \phi$ specifies that for all such nodes ϕ must be satisfied.

TSSL formulas can be viewed as pattern classifiers. Although TSSL is capable of describing complicated spatial behaviors, it is difficult in general to write a formula that describes a complex pattern. In [9], the authors propose to use machine learning techniques in order to find such a formula from a given set of positive and negative examples.

Assume a set of positive images ($\mathcal{Y}_+$), illustrating a desirable pattern, and a set of negative images ($\mathcal{Y}_-$) are available. We can create set $\mathcal{L}$ from these images as

$$\mathcal{L} = \{(Q_y, +) \mid y \in \mathcal{Y}_+\} \cup \{(Q_y, -) \mid y \in \mathcal{Y}_-\},$$

where Q_y is the quad-tree generated from image y. The set $\mathcal{L}$ is separated into a learning set $\mathcal{L}_L$ (used to train a classifier) and a testing set $\mathcal{L}_T$ (used to test the classifier obtained from $\mathcal{L}_L$) such that $\mathcal{L} = \mathcal{L}_L \cup \mathcal{L}_T$. A rules-based learner called RIPPER [17] is used learn a set of classification rules from $\mathcal{L}_L$ and these rules are transformed into a TSSL formula [9].

We applied this framework to obtain classifiers for three classes of patterns: inside-out, outside-in, and globular. First, a tool was developed that creates learning and testing sets for these patterns by generating random images corresponding to different patterns as well as images without any particular behavior. The sizes of the learning and testing sets for each of the patterns are presented in Table I.

Three learning sets $\mathcal{L}_L^{\text{inside-out}}$, $\mathcal{L}_L^{\text{outside-in}}$, and $\mathcal{L}_L^{\text{globular}}$ were created by constructing quad-trees with a depth $d = 5$ from these images. Next, RIPPER was employed to learn a TSSL classifier for each pattern: $\Phi_{\text{inside-out}}$, $\Phi_{\text{outside-in}}$, and Φ_{globular}. The algorithm terminated in 42 minutes for $\Phi_{\text{outside-in}}$, 74 minutes for $\Phi_{\text{inside-out}}$, and 161 minutes for Φ_{globular}, using an iMac with 2.8 GHz Intel core i5 CPU and 32 GB RAM. The classification rate for the testing sets in Table I are 99% for outside-in. 98% for inside-out, and 96% for the globular pattern. $\Phi_{\text{inside-out}}$, $\Phi_{\text{outside-in}}$, and Φ_{globular} will be used later to automatically check whether an image generated by the simulator of Sec. II demonstrates one of the three specified patterns.

C. Quantification

A formal recursive definition for the qualitative semantics of TSSL is presented in [9]. These semantics can be used to assign a true (satisfied) or false (violated) label to a TSSL specification with respect to an unlabeled quad-tree. To provide information about how strongly an image satisfies or violates the given property, TSSL is also equipped with a recursive quantitative semantics definition which assigns a real value to a TSSL formula ϕ with respect to vertex $v \in \mathcal{V}$ of quad-tree $Q = (\mathcal{V}, R)$; denoted by $\rho(\phi, v)$. It is proven in [9] that TSSL quantitative semantics are sound. In other words, a quad-tree Q satisfies a formula ϕ ($Q \models \phi$) if $\rho(\phi, v_0) > 0$ where v_0 is the root of Q, and Q violates ϕ ($Q \not\models \phi$) if $\rho(\phi, v_0) < 0$. Therefore, the problem of checking whether an image contains a pattern expressed as a TSSL formula reduces to computing its quantitative valuation. Moreover, the absolute value of $\rho(\phi, v_0)$ can be viewed as a measure of how strongly ϕ is satisfied (or violated) by Q. Hence, the quantitative valuation of a formula with respect to a quad-tree is called its *robustness*.

TSSL quantitative valuation is an effective tool to quantify emergence of a pattern in an individual image. However, notice that the simulator of Sec. II produces a time sequence of images, and patterns can emerge at any time point in general. Consequently, we need to quantify the emergence of global behaviors in time-varying quad-trees. We use a temporal extension of TSSL, called spatial temporal logic (SpaTeL) for this purpose. SpaTeL was first introduced in [10] to study time-varying spatial patterns in networked systems. Refer to [10] for a complete explanation and formal definitions for SpaTeL. In this paper, we focus on SpaTeL formulas that are used to synthesize patterns at an unknown time. A SpaTeL formula is formed by nesting TSSL formulas inside temporal operators. Consider a time sequence of quad-trees Q_t, where $t \in \{0, 1, \cdots, T\}$, with the following specification: the TSSL formula Φ_{pattern} must eventually be satisfied within the interval $t \in \{t_1, \cdots, t_2\}$. This specification can be formalized by the following SpaTeL formula:

$$\Psi_{\text{pattern}} = F_{[t_1, t_2)} \phi_{\text{pattern}}. \tag{13}$$

SpaTeL is also equipped with quantitative semantics. The quantitative valuation for Ψ_{pattern} with respect to the sequence of quad-trees Q_t is denoted by $\rho_t(\Psi_{\text{pattern}}, v_0(t))$.

$$\rho_t(\Psi_{\text{pattern}}, v_0(t)) = \max_{t \in \{t_1, \cdots, t_2\}} \rho(\phi_{\text{pattern}}, v_0(t)), \tag{14}$$

where $v_0(t)$ is the root of Q_t.

It is proven that SpaTeL quantitative semantics are sound (has a positive valuation for satisfying sequences of quad-trees and a negative valuation for violating ones) [10]. Similar to TSSL, SpaTeL's quantitative valuation (robustness) is shown to be a good measure of how strongly a time varying quad-tree (a sequence of images) satisfies or violates a spatio-temporal specification [10]. Note that in Equation 13 $\rho_t(\Psi_{\text{pattern}}, v_0(t)) > 0$ if the TSSL formula ϕ_{pattern} is satisfied by at least one of the quad-trees in $\{Q_t \mid t_1 \leq t < t_2\}$. Therefore, we can use SpaTeL's quantitative valuation for Ψ_{pattern} to quantify the emergence of a pattern (e.g., outside-in, inside-out, or globular) in a sequence of images simulated from the model of Sec. II.

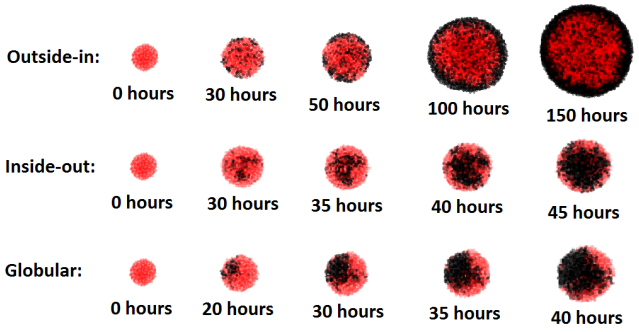


Fig. 4: Sample images of simulations derived from the optimized parameter values: $(k_1, k_2) = (0.08, 0.92)$ for outside-in, $(k_1, k_2) = (0.33, 0.09)$ for inside-out, and $(k_1, k_2) = (0.50, 0.11)$ for globular.

D. Parameter Synthesis

At this point, we are able to quantify how strongly a sample trace $Q_t(\Pi)$ from the simulator of Sec. II satisfies or violates emergence of a pattern specified by the SpaTeL formula (13). This metric can serve as the fitness function in an optimization process over the parameter space from Equation (12). The goal is to determine the parameterization Π_{pattern} that maximizes this metric:

$$\Pi_{\text{pattern}} = \arg\max_{\Pi \in \Omega} \rho_t(\Psi_{\text{pattern}}, v_0(t)) \tag{15}$$

Many optimization methods can be used to solve (15). Inspired by [9] and [10], we employed particle swarm optimization (PSO) [18]. PSO is a heuristic solution to unconstrained optimization problems that is capable of solving problems with irregular search spaces and does not require the fitness function to be differentiable.

This procedure was performed for the three patterns inside-out, outside-in, and globular. Inspired by earlier analysis of similar models in [7] and [1], we chose to fix the values for α, n_1, and n_2 at 0.005, 25, and 25, respectively. The PSO search algorithm was performed on parameters $(k_1, k_2) \in [0, 1] \times [0, 1]$. The optimized parameter values are presented in Fig. 4. The computation was distributed on a cluster with 8 processors at 2.1GHz and the running time was about 30 hours for the outside-in pattern, 11 hours for the inside-out pattern, and 16 hours for globular. Sample simulations derived from these parameters are demonstrated in Fig. 4. As illustrated in this figure, the simulated traces resulting from these parameters qualitatively resemble the patterns of Fig. 1b at the transitioning stage.

IV. Future Work

Directions for future work include exploring how context-independent parameters, such as cell division rates, affect the emergence of patterns. We also plan to study stem cell aggregate shapes other than spheres (e.g., cones, rods). More complex and biologically interesting patterns will be studied by incorporating diffusion among locally interacting cells in the model. Also, a more effective fitness function for the optimization procedure may be developed by combining TSSL/SpaTeL robustness with other metrics such as [7], [1]. Finally, we will define oct-trees as 3D extensions of quad-trees and use them to learn formal classifiers for patterns that are only observable in a 3D space.

Acknowledgments

The authors would like to acknowledge Cristian-Ioan Vasile from Boston University for providing insight about the model efficiency and implementation.

References

[1] D. E. White, J. B. Sylvester, T. J. Levario, H. Lu, J. Todd Streelman, T. C. McDevitt, and M. L. Kemp, "Quantitative multivariate analysis of dynamic multicellular morphogenic trajectories," *Integr. Biol.*, 2015.

[2] M. Kinney, T. Hookway, Y. Wang, and T. McDevitt, "Engineering three-dimensional stem cell morphogenesis for the development of tissue models and scalable regenerative therapeutics," *Annals of Biomedical Engineering*, vol. 42, no. 2, pp. 352–367, 2014.

[3] R. Darabi, K. Gehlbach, R. M. Bachoo, S. Kamath, M. Osawa, K. E. Kamm, M. Kyba, and R. C. R. Perlingeiro, "Functional skeletal muscle regeneration from differentiating embryonic stem cells," *Nature Methods*, vol. 14, no. 2, pp. 134–143, Feb 2008.

[4] A. Shkumatov, K. Baek, and H. Kong, "Matrix rigidity-modulated cardiovascular organoid formation from embryoid bodies," *PLoS ONE*, vol. 9, no. 4, p. e94764, 04 2014.

[5] S. J. Morrison and A. C. Spradling, "Stem cells and niches: Mechanisms that promote stem cell maintenance throughout life," *Cell*, vol. 132, no. 4, pp. 598 – 611, 2008.

[6] D. T. Scadden, "The stem-cell niche as an entity of action," vol. 441, no. June, pp. 1075–1079, 2006.

[7] D. E. White, M. a. Kinney, T. C. McDevitt, and M. L. Kemp, "Spatial pattern dynamics of 3D stem cell loss of pluripotency via rules-based computational modeling." *PLoS computational biology*, vol. 9, no. 3, p. e1002952, jan 2013.

[8] R. Grosu, S. A. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci, "Learning and detecting emergent behavior in networks of cardiac myocytes," *Communications of the ACM*, vol. 52, no. 3, pp. 97–105, 2009.

[9] E. Bartocci, E. A. Gol, I. Haghighi, and C. Belta, "A formal methods approach to pattern recognition and synthesis in reaction diffusion networks," *IEEE Transactions on Control of Network Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[10] I. Haghighi, A. Jones, Z. Kong, E. Bartocci, R. Grosu, and C. Belta, "Spatel: a novel spatial-temporal logic and its applications to networked systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 189–198.

[11] A. Livigni and J. M. Brickman, "Previews Oct4 : The Final Frontier , Differentiation Defining Pluripotency," *Developmental Cell*, vol. 25, no. 6, pp. 547–548, 2013.

[12] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[13] J. Cha, H. Bae, N. Sadr, S. Manoucheri, F. Edalat, K. Kim, S. Kim, I. Kwon, Y.-S. Hwang, and A. Khademhosseini, "Embryoid body size-mediated differential endodermal and mesodermal differentiation using polyethylene glycol (peg) microwell array," *Macromolecular Research*, vol. 23, no. 3, pp. 245–255, 2015.

[14] M. D. Johnston, C. M. Edwards, W. F. Bodmer, P. K. Maini, and S. J. Chapman, "Mathematical modeling of cell population dynamics in the colonic crypt and in colorectal cancer," *PNAS*, pp. 24–29, 2007.

[15] Z. Sun and N. L. Komarova, "Stochastic modeling of stem-cell dynamics with control," *Mathematical Biosciences*, vol. 240, no. 2, pp. 231–240, 2012.

[16] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[17] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 115–123.

[18] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 760–766.