Modeling irrelevant cell types in the organization of human iPSC-derived mini-kidneys

Cassandra Elisabeth Rogers: 260504163

I. Introduction

CURRENTLY, human kidney transplantation relies on fully-developed donor kidneys. However, the advent of induced pluripotent stem cell (iPSC) technology developed by Takahashi and Yamanaka in 2006 has led to the research aimed at creating functional kidneys that can replace *in vivo* animal testing of renal drug toxicity, and furthermore, alleviate the reliance on kidney donors for clinical transplantation [1]. iPSCs are unique in that they can be created using human adult somatic cells, such as skin cells, and induced to maintain a pluripotent state analogous to embryonic stem cells. Similar to ES cells, iPSCs can be induced to differentiate into other somatic cell types.

The Davies lab have successfully differentiated iPSCs into renal cells in an effort to create human kidneys, but have found that the renal cells do not self-organize properly and thus fail to develop into kidneys [2]. The proposed hypothesis is that the differentiation is imperfect; there may be irrelevant cell types contaminating the cultures. These contaminating cells would likely physically obstruct properly differentiated renal cells from sorting into organized cell populations during kidney construction. This obstruction of cell migration and self-organization presents a problem in kidney engineering; in order to produce functional human iPSC-derived miniature kidneys for drug toxicity testing, and eventually mature kidneys for transplantation, it is critical to understand the nature of the self-organization problem [2].

Two approaches are taken to test whether contaminating cells can interrupt self-patterning cell systems. The first approach deliberately contaminates a two-dimensional *in vitro* self-patterning system previously developed by the lab [2]. This system uses synthetic genetic modules to fluorescently colour and visualize the locations of different cell types after the system is allowed to self-organize. Comparison of these two approaches will determine whether the computer simulation is an adequate model of the *in vitro* self-patterning culture, and furthermore, will allow future researchers to model their self-patterning systems without the labour- and time-intensive efforts of cell culture.

The second approach is to develop a computer model of the system. This model uses the idea that different cell types adhere to each other and themselves with different binding energies [2]. As cells are allowed to migrate in the virtual culture as the program progresses, the system begins to pattern. The computer model allows for additional cell types, with different binding energies, to be added in order to examine the effects on pattern formation.

Cellular automata are virtual grids of ”cells” with specific properties. Cell movement, appearance, and disappearance are governed by a set of rules. Cellular automata represent useful models for biological systems, but also have a wide array of computational and mathematical applications. In 1952, Alan Turing proposed mathematical cellular automaton system for cell sorting during morphogenesis [3]. This system used ”morphogen” concentrations, rather than cell adhesions and binding

Advisors: Jamie A. Davies and Elise Cachat, University of Edinburgh Centre for Integrative Physiology, Edinburgh, Scotland.

energies, to model cell sorting during development. However, we can build a system that relies on rules based on direct cell–cell interactions [4]. Such a system is would be useful for modeling the effect of ”irrelevant” cell types on cell pattern formation, because additional ”cell types,” defined by their adhesive properties, can be added to a known adhesive patterning simulation. The simulation can then be compared to simulations run without irrelevant cell types.

This paper will focus on the computational approach, although the biological approach will be discussed in order to draw parallels between the two representations of the adhesive cell sorting system.

II. Methodology

Biological Methods

A self-patterning system has already been created and used by the Davies lab to allow human embryonic kidney cells expressing the tetracycline repressor protein (T-REx-293 cells) to grow and self-organize into spontaneous patterns in two-dimensional cultures [2]. This system includes tetracycline-inducible cells over-expressing Cdh1 (an adhesive surface protein) labelled with green fluorescent protein (GFP) and Cdh3-expressing cells labelled with mCherry (a red fluorescent marker). Both cadherins Cdh1 and Cdh3 have previously been shown to be involved in cell sorting in mammalian cells [5]. This existing system was modified to include a wild type T-REx-293 cell line, which did not express either Cdh1 or Cdh3, nor did it express a fluorescent marker. This cell line represents the contaminating, irrelevant cell line.

Varying proportions of Cdh1- and Cdh3-expressing cells and wild type cells were allowed to form monolayers in flasks over 48 hours. After 48 hours, they were imaged using a fluorescent inverted microscope and examined for pattern formation. Cultures of only Cdh1- and Cdh3-expressing cells, as well as cultures of only wild type cells, were imaged as controls.

Computational Methods

The cellular automaton was developed in Java and in Processing, a Java-like language used for visual programming. The program has three critical user-defined variables: an array of all cell types, C; the proportion of each cell type $Prop(C_i)$ and the binding energies of each cell to each other cell type, a two-dimensional, symmetric array $BindE$.

The algorithm first sets up a $N \times N$ grid of randomly distributed cells with cell types specified in C (Algorithm 1). It does this by calculating the cumulative proportion of cells and choosing a random float between 0.0 and 1.0 to decide the identity of each index in $Grid$.

A function is set up to calculate the energy of a cell $ChosenCell$ in relation to the cells surrounding it (Algorithm 2). This energy is calculated using the set of previously defined binding energies $BindE$, a two-dimensional array containing each cell’s binding energy to every other cell.

The algorithm then carries out a set of cellular automaton rules that determine when and how a cell can move within the grid on a particular iteration (Algorithm 3). On each iteration, a cell ch is chosen randomly from the $N \times N$ grid. If ch is surrounded by cells of a different type, a series of rules is carried out to allow for a possible swap in $Grid$. If none of the surrounding cells are different, then no swap is initiated and $Grid$ does not change.

Algorithm 1 Set Up Grid

function INITIAL–DISTRIBUTION(C)
  Let $Grid$ be a $N \times N$ grid of cells
  for each cell $Grid_{i,j}$ in $Grid$ **do**
    choose a random `float` $0.0 \leq d \leq 1.0$
    for each cell type C_k in C **do**
      if $\sum_{j=1}^{j=k-1} Prop(C_j) \leq d \leq \sum_{j=1}^{j=k} Prop(C_j)$ **then**
        Place C_k in $Grid_{i,j}$
      end if
    end for
  end for
  return $Grid$
end function

Algorithm 2 Calculate Conformation Energy

function CALCULATE–ENERGY($Grid$, $BindE$, $ChosenCell$, x, y)
  Initialize $TotalEnergy$
  for each cell $Surr_i$ in the 8 cells surrounding the cell at $Grid_{x,y}$ **do**
    $TotalEnergy = TotalEnergy + BindE_{Surr_i, ChosenCell}$
  end for
  return $TotalEnergy$
end function

Algorithm 3 Allow Cell To Move

function MOVE–CELL
  Let ch with coordinates x, y be a cell randomly chosen from $Grid$
  Let $DiffNeighbours$ be the number of cells adjacent to ch that are a different cell type from ch
  if $DiffNeighbours > 0$ **then**
    Let $ChosenEnergy$ = CALCULATE–ENERGY($Grid$, $BindE$, ch, x, y)
    for each different neighbour $Surr_i$ with coordinates $xSurr_i, ySurr_i$ **do**
      Let $CurrentEnergy$ = $ChosenEnergy$ + CALCULATE–ENERGY($Grid$, $BindE$, $Surr_i$, $xSurr_i$, $ySurr_i$)
      Store $CurrentEnergy$ into an array $NoSwapEnergies_i$
      Let $SwappedEnergy$ = CALCULATE–ENERGY($Grid$, $BindE$, $Surr_i$, x, y) + CALCULATE–ENERGY($Grid$, $BindE$, ch, $xSurr_i$, $ySurr_i$)
      Store $SwappedEnergy$ into an array $SwapEnergies_i$
      Let $EnergyDifferences_i = SwappedEnergy_i - NoSwapEnergies_i$
    end for
    Let $Offset$ be a random number such that $0 \leq Offset < 8$
    for i in $DiffNeighbours$ **do**
      $index = i + Offset$
      if $index > 8$ **then**
        $index = index - 8$
      end if
      Let $BestEnergyChange = \min(EnergyDifferences)$
      Let $BestSwap = \min_{index}(EnergyDifferences)$
      if there exists more than one $BestEnergyChange$ **then**
        Choose a random $BestSwap$ from the equivalent $BestEnergyChanges$
      end if
      if $BestEnergyChange \leq 0$ **then**
        Swap $BestSwap$ with ch in $Grid$
      end if
    end for
  end if
end function
Repeat function *ad infinitum*

Algorithm 4 Quadrant Count

1: **function** QUADRANT–COUNT($Grid$, N, $CellType$)
2: Let $NoCellsOfType$ be the number of cells in $Grid$ of type $CellType$
3: $NoQuadrants = 2\frac{N^2}{NoCellsOfType}$
4: $Mean = \frac{NoCellsOfType}{NoQuadrants}$
5: Let c_q be the number of cells in quadrant q
6: $Variance = \frac{\sum c_q^2 - \frac{(\sum c_q)^2}{n}}{n-1}$
7: $VTMR = \frac{Variance}{Mean}$
8: **return** $VTMR$
9: **end function**
10: **function** FIND–P– VALUE($Grid$, N, $CellType$)
11: $VTMR$ = QUADRANT–COUNT($Grid$, N, $CellType$)
12: **for** i in (0,500) **do**
13: Generate a randomly distributed $N \times N$ grid $RandGrid$ with the number of cells of type $CellType$ corresponding to $Prop(CellType)$
14: $NewVTMR$ = QUADRANT–COUNT($RandGrid$, N, $CellType$)
15: **if** $NewVTMR > VTMR$ **then**
16: $NoGreater = NoGreater + 1$
17: **end if**
18: $p = \frac{NoGreater}{500}$
19: **return** p
20: **end for**
21: **end function**

The algorithm uses the CALCULATE–ENERGY function to calculate the energy of the randomly chosen cell ch. This energy is added to the energy of each surrounding cell $Surr_i$ and stored into an array representing energies of no swap.

Then, the algorithm calculates the energy of a swap by using the CALCULATE–ENERGY function with swapped cells in the coordinates of $Grid$. The energy differences for each index of the $SwappedEnergy$ and $NoSwapEnergies$ are calculated and stored in a separate array.

To prevent bias that comes from examining the surrounding cells in the same order each time, we define an $Offset$ parameter (Algorithm 3, Line 13) that starts the examination at a different index each time. The best energy change is actually the *minimum* energy change, because we are dealing with Hamiltonians; therefore, the most favourable energy conformation achieves the lowest energy. The swap is only considered favourable if this energy change is less than 0. In this case, the swap is initiated and $Grid$ is updated (Algorithm 3, Line 25). We also take into account the situation in which the best possible energy can be achieved by different swaps by choosing a random best swap. The MOVE–CELL function is repeated *ad infinitum*, until the user terminates the program or sets a defined termination point.

The last algorithm uses quadrant count analysis to determine whether a particular type of cell $CellType$ is clustered in the $Grid$ [6] (Algorithm 4). It then compares the Variance-To-Mean-Ratio $VTMR$ of the quadrant count of $Grid$ to the quadrant count of many randomly distributed grids. the p-value is the proportion of these random grids with a higher $VTMR$ than the $Grid$. If $p > 0.05$, then the original grid is randomly distributed. If $p < 0.05$, then the original grid is distributed significantly differently from a randomly distributed grid and has formed clusters of the type of cell $CellType$. QUADRANT–COUNT can be called on every iteration, or every 10 iterations, of the program (for example). This allows the p-value to be tracked as the program progresses. To this end, the cycle at which the grid no longer resembles a random distribution can be determined.

III. Results

Biological Results

The correct seeding proportions in order to achieve approximately equal proportions at 48 hours were determined to be 37.5% Cdh1, 37.5% Cdh3 and 25% WT (Fig. 1, left).

In the absence of quantitative analysis in the form of Kolmogorov-Smirnov statistics, it is not possible to definitively say whether the addition of wild type T-REx-293 cells hindered the self-organization of Cdh1- and Cdh3-expressing cells over time. However, basic analysis of the images allowed some determinations to be made about the behaviour of cells when irrelevant, wild type cells are introduced. For example in a mixture seeded with 37.5% Cdh1, 37.5% Cdh3, and 25% wild type, red Cdh3-expressing cells appear to form large clusters with rounded boundaries, compared to green Cdh1-expressing cells, which form clusters with rougher edges interrupted with wild type (black) cells (Fig. 1, right). By contrast, when wild type cells are not present in a mixture of 50% Cdh1- and 50% Cdh3-expressing cells, green Cdh1-expressing cells form the largest clusters, mostly surrounding islands of smaller red clusters (Fig. 1, left).

Because of the apparent effects of wild type cells on cluster formation, the next step was to determine whether wild type cells tended to preferably border either Cdh1- or Cdh3-expressing cells. However, there did not appear to be a difference in the amount of red or green cells bordering wild type populations, only a difference in the shape of those border regions.

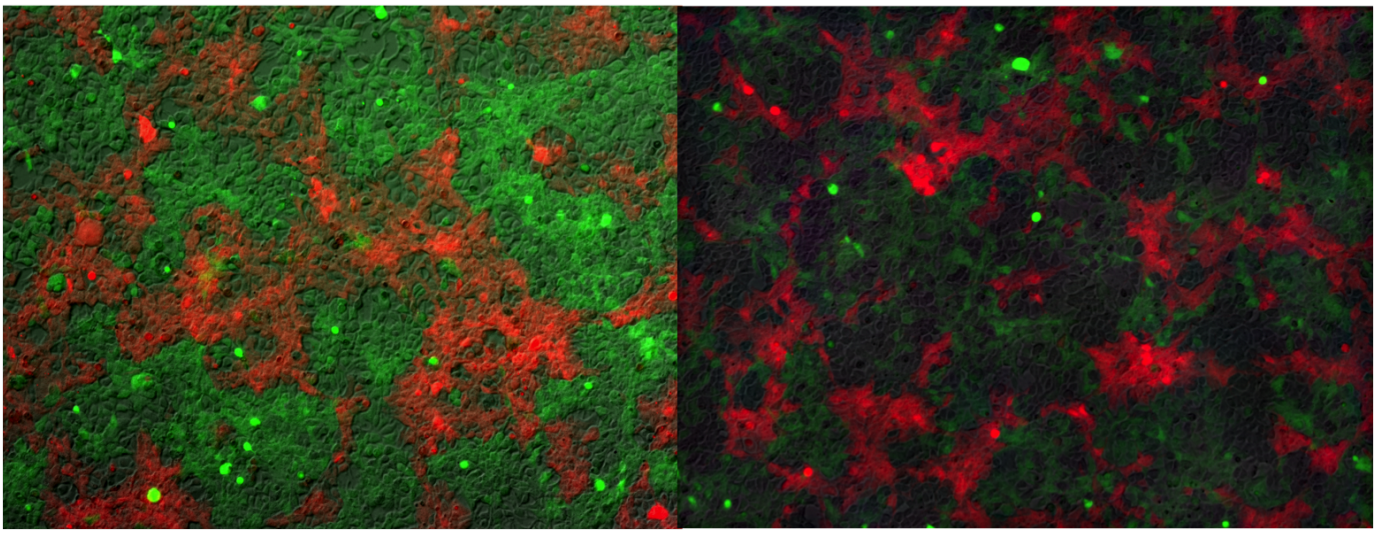

Fig. 1. The leftmost image shows a 50% Cdh1 (green), 50% Cdh3 (red) mix of cells after 48 hours. The rightmost image shows a 37.5% Cdh1 (green), 37.5% Cdh3 (red), and 25% wild type (black) mix of cells after 48 hours. Wild type cells grow faster than Cdh-expressing cells because they do not express a fluorescent protein, and therefore had to be plated at a lower density than Cdh-expressing cells.

Computational Results

INITIAL–DISTRIBUTION must visit each index in the $N \times N$ grid and place a cell from the array possible cell types, C, of length nc.The computational complexity of this algorithm is $O(n^2)$.

Let MOVE–CELL be repeated M times, where M is the indefinite number of cycles the user allows the program to run. Then, for each cycle, INITIAL–DISTRIBUTION must examine each different neighbour surrounding a randomly chosen cell, at most 8 neighbours. For each of these different neighbours, the energy of the configuration must be calculated for the central

cell and the cell with which it will be swapped. CALCULATE–ENERGY(v)isits each of the surrounding cells, or 8 cells. Then, the minimum of $EnergyDifferences$ must be calculated, where $EnergyDifferences$ contains at most 8 differences. These operations are $O(8) \sim O(1)$. The grid is not updated for every possible swap; it is only updated when a swap is considered energetically favourable compared to the previous conformation. Furthermore, the energy calculations are necessary only for the comparison of the new possible conformation to the old one.

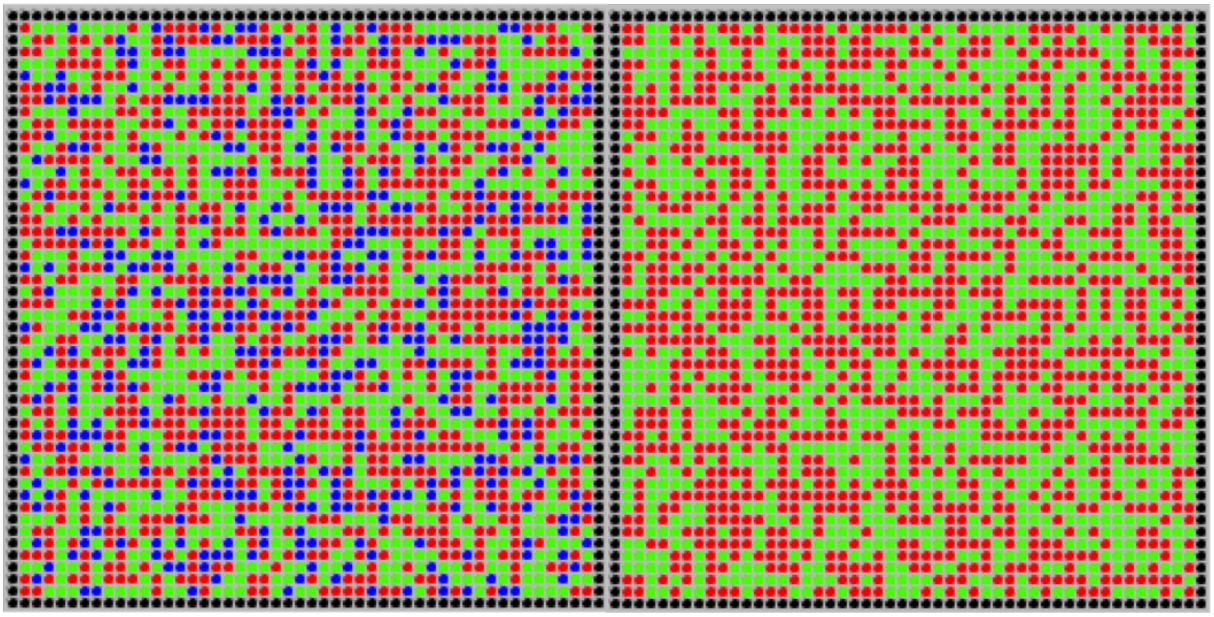

Fig. 2. These images demonstrate possible initial random distributions of the cellular automaton with n=3 cell types (left) and n=2 cell types. The left hand image was seeded with proportions 40% green, 40% red, and 20% blue. The right hand image was seeded with proportions 50% red and 50% green.

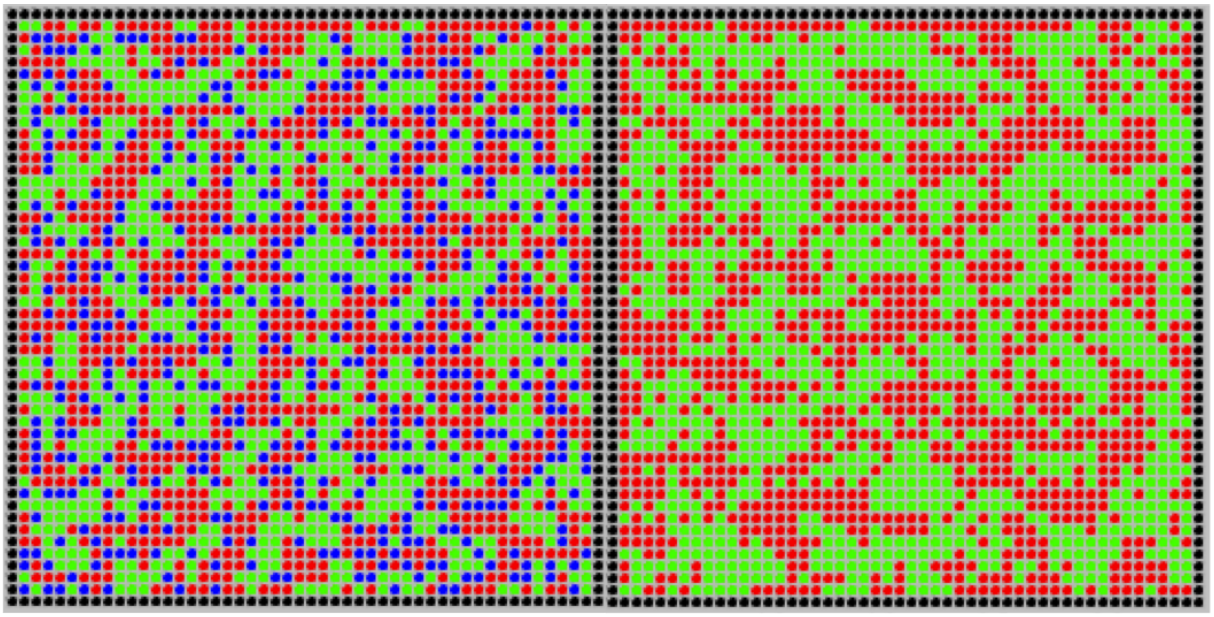

Fig. 3. These images demonstrate the cellular automaton after approximate 3500 cycles. The starting proportions are the same as those for Fig. 2. The binding energies were defined by a matrix M for which $M_{g,g} = -2.0$, $M_{g,r} = -0.5$, $M_{r,r} = -1.0$, $M_{g,b} = -2.5$, $M_{r,b} = -2.5$ and $M_{b,b} = -2.0$. The extreme binding energies for *blue* cells to other cells were chosen because these values are more likely to interfere with red and green cell sorting. The binding energies for the right hand image were the same, but without the bonds to *red* cells.

It is therefore not necessary to calculate the energy of the entire grid, which would involve calculating N^2 energies, which involves $\sim N^8$ bonds to adjacent cells.

In total, the setup, energy calculation, and movement of cells in the cellular automaton is $O(N^2)$.

The quadrant count analysis must find the number of cells of type $CellType$ in a $N \times N$ grid, which is $O(N^2)$. It must also generate a randomly distributed $N \times N$ grid k times, where $k = 500$ in this example. Therefore, the computational complexity of QUADRANT–COUNT is $O(N^2) + O(k \cdot N^2) \sim O(k \cdot N^2)$.

The simulation was tested for $n = 3$ and $n = 2$ cell types and allowed to run for 3500 cycles (approximate 3 minutes) (Figs. 2 and 3). Based on the changes that are visible in the grid when the program is run, the cells appear to be sorting correctly according the the adhesive sorting rules set out by MOVE–CELL(.) After 3500 cycles, there are clear clusters of the two cell types with the highest proportions (red and green cells, each with a 40% starting proportion) and interspersed blue cells. The binding energies chosen for blue cells to other cell types were purposefully chosen to be extreme. This is because blue cells' extreme binding energies are more likely to interfere with red and green cell binding if they are higher than red and green cell binding energies.

In order to quantify transition of the grid from random to non-random, p-values were calculated according to the protocol defined in QUADRANT–COUNT and FIND– P–VALUE. However, these protocol do not appear to be accurate as of yet: the data indicate that the p-values do not change over time, remaining approximately the same even at 100,000 cycles, at which point the system is clearly clustered based on visual inspection. It would appear that the error could lie within the calculation of the Variance-to-Mean Ratio or with the choice of quadrant size.

Experimentation with different quadrant sizes yielded the same problem. However, the protocol for quadrant count analysis was originally designed for systems in which points or cells are clustered irregularly in empty space [6]. The cellular automaton system described here represents a regular grid with no empty space, and the protocol has been adapted to find clusters of only one cell type in the grid.

It is likely that the choice of quadrant size was not the issue because of the irregularity of the Variance-to-Mean Ratio and p-value calculations. For each program test (starting with a different random grid), the Variance-to-Mean Ratios were widely different, and the p-values were as high as 0.9 or as low as 0.05. Without fail, however, the p-values within any one test of the program did not differ by any large from cycle to cycle.

IV. DISCUSSION AND CONCLUSIONS

It is possible that the cellular automaton described here is a useful model of an adhesive cell sorting system contaminated by an irrelevant cell type. The extreme binding energies chosen for the extraneous cell type (blue cells) represent interference by an irrelevant cell type. Cell conformations with extreme binding energies are more likely to be chosen as the favourable conformation because the binding energies are so low.

However it is impossible to say with certainty whether the cellular automaton with the third, irrelevant cell type actually fails to form statistically significant clusters (compared to a random distribution) in the same number of iterations as does the cellular automaton with only two cell types. In order to do this, we would need to compare the distribution of cells after each

iteration i of the program to many randomly distributed grid in order to determine what percentage of randomly distributed grids are as sorted as the cellular automaton at iteration i. QUADRANT–COUNT and FIND–P–VALUE attempt to do this but apparently fail at producing accurate results.

Ultimately, in order to determine if this cellular automaton is biologically useful, we need a quantitative way to compare the computation distribution to the *in vitro* distribution. However, it is extremely difficult to calculate these statistics on biological samples. There are two main reasons for this. The first is that it requires manually counting and cells on two-dimensional plates using a specialized fixation and microscope. Counting and labelling cells is not always straightforward, because cells tend to overlap or have unclear peripheries. The second reason is that calculating the statistics must be done using a randomly generated distribution of cells that is irregular: cells do not grow in the neat grid we represented with the cellular automaton. However, if this is done, we can compare the results of the cellular automaton to *in vitro* results and determine whether the former accurately represents the latter.

The computational model will never perfectly represent a biological system. In particular, this model does not take into account a major phenomenon seen in *in vitro* cell culture. The cellular automaton does not take into account growth. In a biological sample, cells are seeded in specific proportions (see fig. 1) at very low densities. This is because they are allowed to grow and sort over 48 hours, at which point they reach confluence, or 100% coverage of the flask. Past 48 hours, cells begin to die from overcrowding. The computational model, on the other hand, can be allowed to run infinitely long without any cell death. The barriers of the model do not expand: the grid will always remain of size $N \times N$. Furthermore, the virtual cells are confluent at the start of the program. Rather than growing and sorting, they only sort.

In the future, it will be interesting to compare the computational results to the biological results and extend the capacities of the cellular automaton. For example, we could incorporate cell growth and death into the model, allowing the boundaries of the grid to expand. We could also explore three-dimensional computational models, should we find that the computational model represents the biological one. It would be extremely difficult to compare a three-dimensional computational model to a three-dimensional cell culture, as labeling cells in three-dimensional culture is near impossible.

If this model does represent the biological system, it can be applied in human kidney tissue engineering to determine the proportion of irrelevant cell types that can be allowed in iPSC-derived renogenic cell cultures such that cell sorting is not significantly disrupted and embryonic kidney development can proceed.

V. ACKNOWLEDGMENTS

I would like to thank Jamie A. Davies at the University of Edinburgh Centre for Integrative Physiology, as well as Elise Cachat, Melanie Lawrence, Joanna Sharman, and Chris Mills. I would also like to thank the Dr. Hadwen Trust for their scholarship generous funding of my research.

The research described in this paper was started in May 2015 at the University of Edinburgh. All biological methodology was carried out during the summer. The computational model was started but not finished during that time, and the majority of the code was written this semester. Parts of the cellular automaton are inspired by code previously written by Jamie Davies.

REFERENCES

[1] Takahashi K., Yamanaka S. (2006). Induction of pluripotent stem cells from mouse embryonic and adult fibroblast cultures by defined factors. *Cell*. 126(4): 663–76.

[2] Cachat E., Liu W., Martin KC., Yuan X., Yin H., Hohenstein P., Davies JA. (2015). An engineered system for *de novo* 2- and 3-dimensional pattern formation by mammalian cells. *Manuscript submitted for publication*.

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

[4] Deutsch A., Dormann S. (2005). Cellular automaton modeling of biological pattern formation: characterization, applications, and analysis. Boston: Birkhauser (eBook): 143–159.

[5] Nose A., Nagafuchi A., Takeichi M. (1988). Expressed recombinant cadherins mediate cell sorting in model systems. *Cell*. 23;54(7):993–1001.

[6] Tariq E. (2004). Introduction to Point Pattern Analysis. *School of Mines and Technology, South Dakota*.