Multi Species Turing Patterns in Three Dimensions

R. C. M. Dur[1]

Supervisor(s): Dr. M. Skrodzki[1]**, Dr. A.B.T. Barbaro**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: R. C. M. Dur
Final project course: CSE3000 Research Project
Thesis committee: Dr. M. Skrodzki, Dr. A.B.T. Barbaro, Dr. J.S. de Pinho Gonçalves

This page was left blank intentionally.

CSE3000 – Research Project – Final Report (2023)
A. Barbaro and M. Skrodzki (Editors)

Short Paper

Multi Species Turing Patterns in Three Dimensions

R. C. M. Dur[1]

[1]EEMCS, Delft University of Technology, The Netherlands

Abstract

In 1952, Alan M. Turing presented a reaction-diffusion model that described formation of skin patterns. The patterns he predicted have later been found in various natural phenomena, such as in skins of fish or even in vegetation around termite hills. His patterns have even been taken to the micro-level. In 1984, David A. Young proposed a discretisation of this model, which enabled computer simulation. Both Turing and Young had only looked at two-dimensional patterns, until Martin Skrodzki and Konrad Polthier took the patterns to the third dimension in 2017. In this paper, these 3D simulations are generalised to produce patterns with more than two substances. We want to see whether Turing-like patterns also emerge there. To increase simulation speeds, a Graphics Processing Unit (GPU) implementation is described for this multi-species extension. Furthermore, we begin to investigate to what extent an order parameter can be defined, to analyse the formation of 3D structures. We found that our multi-species extension produced Turing-like structures that look similar to the ones found by former models. Our GPU simulation provided a significant performance increase. We also found that our order parameter can distinguish between well-mixed, well-segregated, and fully dominated states. However, it is yet unclear whether it can also be used to classify shapes.

1. Introduction

Objects and organisms take on all sorts of shapes and sizes. Animals develop patterns in their skin for camouflage, physical protection and to signal warnings, for example. Aside from skin-patterns, all kinds of organic structures appear in nature, not necessarily limited to two dimensional patterns. These patterns have been studied from a mathematical point of view for a long time.

In 1952, Alan M. Turing published his paper *"The chemical basis of morphogenesis"* [Tur52], where he describes how a simple reaction-diffusion process between chemical substances (morphogens) can accurately describe the general formation of biological shapes and patterns (morphogenesis). Years later, these patterns are referred to as *Turing patterns* [Bar81, Mur81] and they have been found in several settings in nature (see section 2.1). A little over thirty years later (1984), David A. Young presented *"A Local Activator-Inhibitor Model of Vertebrate Skin Patterns"* [You84], where he morphs Turing's reaction-diffusion model into a discrete form that can be simulated on a computer. A detailed explanation of his model is given in section 2.2.

Both papers from Turing and Young are concerned with two-dimensional patterns. In 2017, this model was elevated to three dimensions and the discovery that Turing-like patterns also arise in three dimensions was made [SP17].

Until now, a lot of work has been devoted to Turing-patterns with two types of substances, including the papers mentioned above. There, the considered area or volume is divided into two segments, making the resulting patterns binary. However, nature is much more diverse and rarely shows binary behaviour.

In this paper, the 3D variant of Young's discrete model, presented by [SP17] and [SRZ20], is generalised to allow multi-species simulations that produce Turing patterns with more than two types of cells. The question at hand is: *How can an existing three-dimensional variant of Young's model be generalised to produce multi-species Turing-like structures?* In addition, we investigate parts of the follow-up question: *To what extent can the formation of such structures be quantified through an order-parameter?* As current implementations are too slow for (a lot of) these complex simulations, it was necessary to implement the model on a Graphics Processing Unit (GPU).

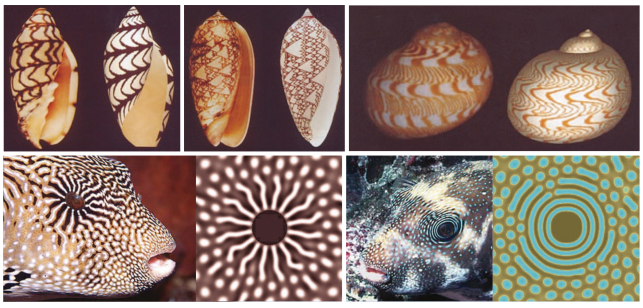

Figure 1: *Biological patterns, reproduced by modified versions of Turing's reaction-diffusion model. Figures taken from [KM10] and reorganised.*

2. Related Work and Background

This section covers related work and technical background to describe the foundation for this work. After a review of the appearance of Turing patterns in nature, Young's activator-inhibitor model is explained in detail. Towards the end, mathematical definitions of a cellular automaton and an order parameter are given. The remainder of this paper will build on these definitions.

2.1. Turing patterns in nature

Various examples of how Turing's reaction-diffusion model could be used to recreate biological patterns have been shown in [KM10] and some of them are shown in Fig. 1. Moreover, the reaction-diffusion theory lies at the basis of other mechanisms that are involved in spatial self-organisation of ecosystems. There is evidence that such mechanisms are related to vegetation patterns generated by, for example, ants and termites [PT17]. In *"The present and future of Turing models in developmental biology"* [Kon22], some more recent developments around Turing's model are discussed from the perspective of an experimental biologist. Experiments around Turing's model are also being brought to the micro-level [XLY*22]. Turing's model is not covered in detail here, as the contributions in this paper are more closely related to Young's discretisation of the model, which is explained in section 2.2.

2.2. Activator-Inhibitor model by D. Young

As a discretisation of Turing's model, Young proposed an activator-inhibitor diffusion model [You84], where pigment cells are spread over a discrete 2D grid with periodic boundary conditions. Each cell can be in either of two states: a differentiated cell (DC) or an undifferentiated cell (UC). All DCs produce two morphogens (chemicals): an activator, which stimulates differentiation of nearby UCs, and an inhibitor, which stimulates dedifferentiation of nearby DCs. Both chemicals diffuse away from their source and get less concentrated at farther distance. Inhibitors have a larger range and are initially less concentrated than activators. This is illustrated in Fig. 2a. Young further introduces the *net activation and inhibition effects*, that are represented by a constant field value. This is illustrated in Fig. 2b. It is therefore possible to represent a morphogen (chemical) by a range R and an influence w. Activators have range R_{act} and influence w_{act}. Inhibitors have range R_{inh} and influence w_{inh}. It typically holds that $R_{\text{inh}} > R_{\text{act}}$ and $w_{\text{inh}} < 0$. The net inhibition field forms a ring around the net activation field.

The initial condition is a uniform mix of differentiated- and undifferentiated cells. This is called a well-mixed state. Each iteration, the cell's states are updated according to these rules:

1. If the sum of all net activation and inhibition effects from neighbouring DCs is larger than zero, this cell becomes a DC itself.
2. If this net effect is smaller than zero, this cell becomes a UC.
3. Otherwise, its state does not change.

With the right parameters, the system will reach a well-segregated state, where the two cell-types separate after convergence. The system therefore experiences a phase transition from a well-mixed to a well-segregated state. Young mentioned the concept of a *cellular automaton* for the first time, when explaining these rules [You84, p.54].

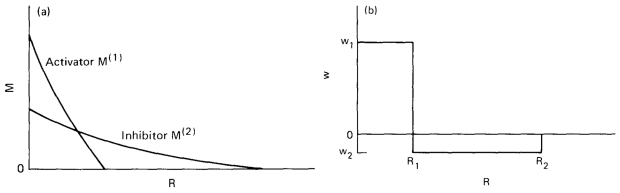

Figure 2: *In (a): diffusion of activator- and inhibitor-morphogens around a DC. M is the concentration of the morphogen and R the distance from the DC. In (b): the net effect of activator (w_1) and inhibitor (w_2), as constant circular regions: positive for the activator and negative for the inhibitor. Figures taken from [You84].*

2.3. Cellular Automata

A Cellular Automaton (CA) typically consists of cells c in a discrete domain D that can be in one of the states in S at a time. Each cell has a finite neighbourhood $\eta(c)$ and cells can transform, between generations, from one state to the other by following a set of rules. CAs became widely-known through John H. Conway's *Game of Life* in the 1970s [Gar70], however they existed long before. The book by Andrew Adamatzky [Ada10] describes a collection of works and extensions to *Game of Life* over the years.

In this paper, CAs are three-dimensional and the cells are organised in an $L \times L \times L$ grid. Cells may therefore also be written using their coordinates: (x, y, z). We may also use $c(t)$ as a function of time, to express the cell's state through iterations. Three dimensional variants of Young's model have already been experimented with in [SP17] and [SRZ20]. In the latter, a parallel implementation on the Central Processing Unit (CPU) is presented, where 24 threads were used [SRZ20, p.14].

2.4. Order Parameter

Later, in section 3.4, an order-parameter will be defined. This parameter is a value between zero and one that indicates the degree of order within the CA. In *"A convection–diffusion model for gang territoriality"* [AB18], by Abdulaziz Alsenafi and Alethea B.T. Barbaro, an agent-based model with two species, that is meant to simulate territorial development of gangs, is studied. Also there, a phase transition between a well-mixed and a well-segregated state occurred. In their paper, an order parameter is defined [AB18, p.768] to analyse this phase transition. A variant of this order parameter, that would be suited for Cellular Automata, was investigated in [Bar23]. In 2021, Abdulaziz Alsenafi and Alethea B.T. Barbaro published *"A Multispecies Cross-Diffusion Model for Territorial Development"* [AB21] where the agent-based order parameter from [AB18] was elevated to the multi-species case [AB21, p.6].

3. Methods

This section provides a detailed explanation of the contributions, mentioned in section 1. It starts with a brief review of the setup and visualisations that are used in experiments, together with an explanation of the steps required for simulation on the GPU. Our multi-species extension of Young's model and the definition of an

order parameter are explained afterwards. Most technologies and software that were used in this research are listed in section 6.

As this paper progresses, multiple simulators will be discussed. To allow for co-existence and comparison of simulators, a server-client framework was setup. The server was programmed in Rust and would mainly be running fast simulations. The client was equipped with a Three.js canvas, to visualise the resulting structures, as well as a useful interface for direct interaction with the CAs that were stored on the server. The main reason for this server-client setup was the combination of fast simulations through a fast and compiled language with widely supported visualisations on the web using Three.js.

3.1. Visualisation

Two visualisation techniques were used: A debugging visualisation and a visualisation that is powered by the Marching Cubes (MC) algorithm. Both are explained here.

3.1.1. Debugging visualisation

Simply showing a coloured cube for each cell in the CA would be too naive: Individual cells would obstruct each other, making it hard to see patterns. Instead, the researcher scrolls through the CA in slices that are rendered as a two dimensional plane of coloured cubes. An example of this visualisation is given in Fig. 3.

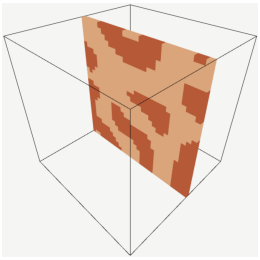

Figure 3: *Debugging visualisation: researchers scroll through the CA to view one intersection at a time.*

3.1.2. Marching Cubes

Following [SP17] and [SRZ20], the Marching Cubes (MC) algorithm was used to make 3D patterns emerge. Showing just one slice at a time is not sufficient for a good representation of these 3D structures. MC creates a border-mesh between two different substances and we then give a different colour to each side to display which substance is located on that side. An example is given in Fig. 4a, where MC is applied on a CA with two cell-types.

A modification is needed to visualise more than two substances. Instead of drawing a border between two substances, we let MC draw a border between one cell-type on one side and any cell *not of that type* on the other. Fig. 4b shows what this looks like.

3.2. GPU simulation

To make sure our GPU simulation was correct, a parallel CPU simulation was implemented, inspired by [SRZ20]. Our CPU implementation worked with an $L \times L \times L$ 3D array of 32-bit unsigned integers, indexed by cell coordinates (x, y, z). The neighbourhood $\eta(c)$ would be computed for every cell c separately, where complexity was brought down by only considering neighbours that lie within the box with sides of size $2R_{\text{inh}} + 1$ around c. Some experiments in this paper use a randomisation seed to make sure all variations have identical initial conditions. This seed is always configured as `ChaCha8Rng::seed_from_u64(1)` from the Rust rand_chacha package.

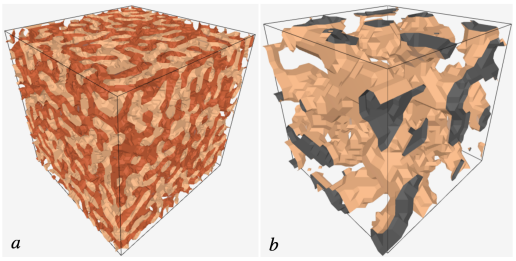

Figure 4: *Using the Marching Cubes (MC) algorithm to visualise the border between cell-types in a CA. This is also done in [SP17] and [SRZ20]. In (a): MC draws the border between two substances. In (b): MC draws the border between, on the one side, the cell-type that was selected by the researcher (here: light orange) and, on the other side, any other cell-types (in grey).*

To bring the simulation to the GPU, some fundamentals had to be changed. Firstly, each cell is represented by an eight-bit unsigned integer, instead of 32 bits, to prevent unnecessary data-transfer between system- and GPU-memory. Secondly, methods were implemented to translate between a 3D-array, as used in the CPU simulation, and a flattened array for use in GPU memory. Lastly, instead of computing neighbourhoods for every cell separately, two arrays of *relative neighbourhoods* are created once for the entire simulation. They are called relative neighbourhoods, because they store $(\Delta x, \Delta y, \Delta z)$ coordinate-deltas, so that the actual neighbour's location can be computed with $(x + \Delta x, y + \Delta y, z + \Delta z)$.

There is a number of steps involved in running an iteration on the GPU. A detailed explanation of these steps is given in the following section 3.2.1. To determine how many iterations are needed to reach convergence, a convergence-checker was implemented, that compared the cell-states in the new generation to the old one. If less than 1% of the cells had changed while going to the next iteration, the CA is said to have converged. All GPU implementations were done using Apple's Metal API (see section 6).

3.2.1. Steps involved in an iteration on the GPU

Step one in the GPU computation is organising the data in a way that it can be inserted into a buffer. A buffer is a container for bytes of data and they are used to transfer data to the GPU's dedicated memory. This GPU implementation requires ten buffers:

1. Two 8-bit unsigned integer arrays: one for the current CA generation and one for the next generation.
2. One 32-bit integer array that contains the size of the automaton and the number of entries in both $\delta\eta_{\text{act}}$ and $\delta\eta_{\text{inh}}$ (further referred to as *the size-buffer*).

3. One 32-bit float array that contains the influences of each chemical (further referred to as *the influences-buffer*).
4. Six 32-bit integer arrays that contain the relative neighbourhoods $\delta\eta_{\text{act}}$ and $\delta\eta_{\text{inh}}$. Both of these require three arrays: one for Δx-coordinates, one for Δy-coordinates and one for Δz-coordinates. (Further referred to as *the neighbourhood-buffers*).

The second step was to instruct the Metal API where the buffers could be found and in what order they should be placed. This is a small step, but an important one as the order of buffers matters. This will be elaborated on in the next step.

The third step was to write a *shader* file. Here, the shader files are written in the *metal-language*. This file specifies the *kernel* function that is called for each cell individually. Its signature contains all buffers and their types in the same order as they were provided to Metal. This function also takes a 'gid' argument, which is used by Metal to specify which cell to work with. This *gid* represents the cell's index in the flattened array that was mentioned in section 3.2. The body of this function contains the calculations that must be performed to determine the new state for this cell.

The last step is to specify the configuration of threads that should be deployed for this computation. In this case, the total number of threads that should be deployed is equal to the number of cells in the CA. Since the GPU does not necessarily have as many threads, Metal divides these threads into thread-groups. The size of each thread-group can be specified by using the maximal number of available threads. This number depends on the machine that is used and can be requested through the Metal API. After these steps, the simulation could be started.

To make an indication of the performance-gain that resulted from stepping to the GPU, the following benchmark was setup. On a CA with $L = 30$ and $K = 1$, with chemical configuration $R_{0,\text{act}} = 3.2, w_{0,\text{act}} = 1, R_{0,\text{inh}} = 6$ and $w_{0,\text{inh}} = -0.2$, 100 iterations are run and the total duration was recorded. This was repeated ten times. The results are discussed in section 4.1.

3.3. Extending to multi-species

Here, Young's activator-inhibitor model will be generalised to K-species and we use a CA to define the extension. In this model, the cells can attain one of $K+1$ types: K differentiated cell-types, corresponding to the K species, and one undifferentiated cell-type. In all following equations, cell-states 0 through $K-1$ indicate a differentiated cell-type, corresponding to the species, and cell-state K indicates an undifferentiated cell. $K = 1$ corresponds exactly to Young's activator-inhibitor model.

In this model, each species j, like the DCs in Young's model, produces two chemicals: an activator and an inhibitor. The activator of species j has range $R_{j,\text{act}}$ and influence $w_{j,\text{act}}$. Likewise, the inhibitor range $R_{j,\text{inh}}$ and influence $w_{j,\text{inh}}$. Again it holds that $\forall j : R_{j,\text{inh}} > R_{j,\text{act}} \wedge w_{j,\text{inh}} < 0$. Each cell is now associated with different neighbourhoods for each species, because the activator- and inhibitor-chemicals from different species have different ranges. We identify two neighbourhoods per species per cell: $\eta_{j,\text{act}}(c)$ is the set of all neighbours of cell c that lie within range $R_{j,\text{act}}$ from c. Similarly, $\eta_{j,\text{inh}}(c)$ is the set of all neighbours of cell c that lie within range $R_{j,\text{inh}}$ and outside of range $R_{j,\text{act}}$ from c.

The simulation process is similar to that of Young's. For each cell, the net effect of activator and inhibitor is calculated for each species *separately*. This will result in a set of K effect-values per cell. The state of this cell in the next generation is then calculated with the following rules, formalised in a series of equations. The neighbourhoods $\eta_{j,\text{act}}(c)$ and $\eta_{j,\text{inh}}(c)$ are defined in Eq. (1) and (2), where $\mathrm{d}(c,c')$ denotes the euclidean distance between cells c and c'. The net activator-inhibitor effect $W_j(c)$ for species j on cell c is computed with Eq. (3). The cell's state in the next generation then follows from Eq. (4).

$$\eta_{j,\text{act}}(c) = \{c' : \mathrm{d}(c,c') \leq R_{j,\text{act}}\} \setminus \{c\} \tag{1}$$

$$\eta_{j,\text{inh}}(c) = \{c' : R_{j,\text{act}} < \mathrm{d}(c,c') \leq R_{j,\text{inh}}\} \tag{2}$$

$$W_j(c) = \sum_{\tilde{c} \in \eta_{j,\text{act}}(c)} w_{j,\text{act}} + \sum_{\tilde{c} \in \eta_{j,\text{inh}}(c)} w_{j,\text{inh}} \tag{3}$$

$$c(t+1) = \begin{cases} \arg\max_j \{W_j(c)\} & \max\{W_j(c)\} \geq 0 \\ K & \max\{W_j(c)\} < 0 \\ c(t) & \max\{W_j(c)\} = 0 \end{cases} \tag{4}$$

3.3.1. Adaptations to the GPU simulation

Some adaptations had to be made to the GPU implementation, to allow for simulation of this K-species model. The number of buffers remained the same and they all have the same roll. However, their structure slightly changed:

1. Where the size-buffer originally contained the automaton-size and the number of neighbours for each chemical, it now also contains the number of species K immediately after the automaton-size, as well as the sizes of all neighbourhoods $\delta\eta_{j,\text{act}}$ and $\delta\eta_{j,\text{inh}}$.
2. The influences-buffer stores the influences of the chemicals for every species in sequence.
3. The six neighbourhood-buffers still represent the $(\Delta x, \Delta y, \Delta z)$ coordinates for both chemicals, but now contain these neighbourhoods $\delta\eta_{j,\text{act}}$ and $\delta\eta_{j,\text{inh}}$ for every species j in sequence. This is illustrated in Fig. 5.

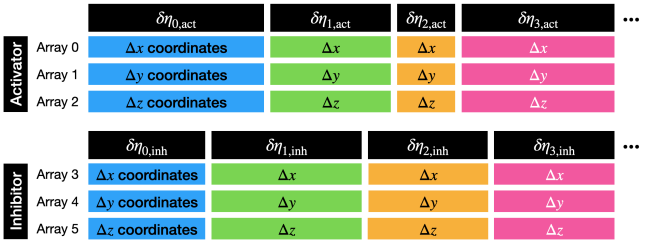

Figure 5: *Six 32-bit integer arrays that contain the relative neighbourhoods $\delta\eta_{j,\text{act}}$ and $\delta\eta_{j,\text{inh}}$ for every species j in sequence. The size of each neighbourhood is stored in the size-buffer.*

3.4. An order parameter

To quantify the phase transition that the automaton undergoes, as more iterations are run, an order parameter is defined. The param-

eter has the form of a vector $\vec{E} = [\varepsilon_0, \varepsilon_1, ...\varepsilon_{K-1}, \varepsilon_{\text{undif}}]$ and it contains one entry for each species, as well as one for the undifferentiated cell-type. The order parameter is shown in Eq. 5 and it should be noted that $\varepsilon_K \equiv \varepsilon_{\text{undif}}$. $\mathcal{N}(c)$ is defined as the direct neighbourhood of c, consisting of its six direct neighbours. As mentioned in section 2, this model has periodic boundary conditions.

$$\varepsilon_j(t) = \frac{1}{6L^3}\left|\sum_{c\in D}\sum_{\tilde{c}\in\mathcal{N}(c)}\left(\sigma_j(c,t)\cdot\sigma_j(\tilde{c},t)\right)\right|$$
$$\sigma_j(c,t) = \begin{cases} 1 & c(t) = j \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

This order parameter is modeled after the order parameter in [Bar23]. Inspiration was taken from the multi-species order parameter in [AB21], however it was argued that a vector $\vec{E}$ of order-parameters, instead of a single number, better suited the nature of a cellular automaton.

3.5. Weighted Volume Difference

As explained in section 2.2, the effect of the chemicals can be influenced by either changing their range or influence. Enlarging either variable increases the dominance of that chemical. Here, the Weighted Volume Difference (WVD_j) is defined in Eq. 6. It captures the net dominance of the activator- and inhibitor-chemicals for a species j, by multiplying volume in which either chemical is active by the influence of that chemical and summing them. Remember that the inhibitor influence $w_{j,\text{inh}}$ is always negative.

$$\text{WVD}_j = \frac{4}{3}\pi\left(R^3_{j,\text{act}}w_{j,\text{act}} + (R^3_{j,\text{inh}} - R^3_{j,\text{act}})w_{j,\text{inh}}\right) \tag{6}$$

If the WVD_j is smaller than zero, it means that the inhibitor of species j is stronger than its activator. Larger than zero means the opposite and precisely zero means the chemicals are *balanced*.

4. Results and discussion

Throughout this section, various results are shown and discussed. After a quick speed comparison between the CPU- and GPU-simulations, various multi-species Turing patterns are shown. The behaviour of the order parameter is investigated and its connection to the development of different types of shapes is discussed.

4.1. GPU simulation

The GPU simulation, used in this paper, produced the same output as the parallel CPU simulation. To assess this, the cells in the CA were compared one by one in various settings and after various numbers of iterations. Such comparison was not possible for the multi-species extension, because that was solely implemented on the GPU.

Table 1 shows the results of the benchmark described in section 3.2.1, from which we see that our GPU simulation was significantly faster than our CPU simulation. On average, the GPU simulated over 200x faster.

	CPU	GPU
Average time	275,236s	1,233s
Standard deviation	18,573s	0,047s

Table 1: *Time to run 100 iterations on a CA with $L = 30$ and $K = 1$, averaged over ten repetitions.*

j	0	1	2	3
$R_{j,\text{act}}$	6	4	4.3	3.2
$w_{j,\text{act}}$	1	1	1	1
$R_{j,\text{inh}}$	10	7	8	6
$w_{j,\text{inh}}$	-0.3	-0.3	-0.22	variable

Table 2: *Chemical configuration for CA with $K = 4$.*

4.2. Multi-species Turing patterns

Consider a CA with $L = 100$ and $K = 2$. Both species have identical chemical-configuration: $R_{01,\text{act}} = 3.2$, $w_{01,\text{act}} = 1.0$, $R_{01,\text{inh}} = 6.0$ and $w_{01,\text{inh}} = -0.2$. Twenty iterations were run, after which the CA converged. 42.86% of the cells were captured by species 0, 42.88% by species 1 and 14.26% are undifferentiated cells.

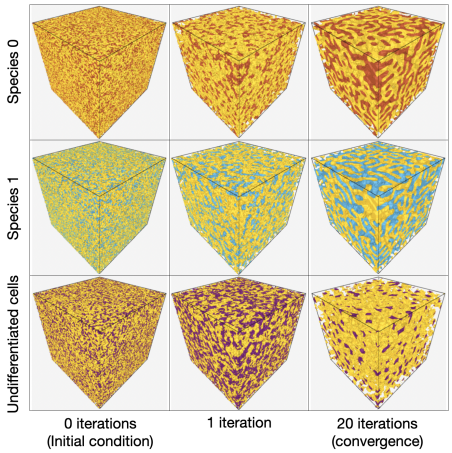

Figure 6: *Temporal evolution of a CA with $L = 100$ and $K = 2$. Both species have identical chemical-configuration: $R_{01,\text{act}} = 3.2, w_{01,\text{act}} = 1, R_{01,\text{inh}} = 6$ and $w_{01,\text{inh}} = -0.2$. Convergence occurred after 20 iterations. Yellow indicates the side of the Marching Cubes border where the respective species is not present.*

It is possible to change the type of structures that emerge, by altering the chemical configuration of just one species. The experiment remains the same, except $w_{0,\text{inh}}$ is varied as indicated in Fig. 7. Also in that figure, the percentage of volume captured by both species is displayed. Remaining cells, not captured by either species, get the undifferentiated type.

To push this experiment further, we now consider a CA with $L = 100$ and $K = 4$. The chemical configuration for all species can be found in Table 2. These were chosen to let species develop different shapes. $w_{3,\text{inh}}$ is varied as indicated alongside the results of this experiment in Fig. 8.

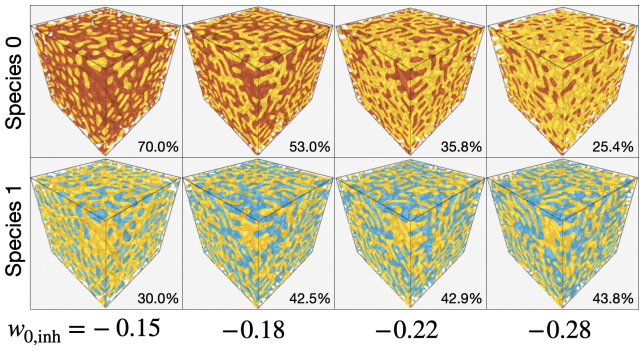

Figure 7: *Converged states of a CA with $L = 100$ and $K = 2$ for varying values of $w_{0,\text{inh}}$. Other variables: $R_{01,\text{act}} = 3.2, w_{01,\text{act}} = 1, R_{01,\text{inh}} = 6$ and $w_{1,\text{inh}} = -0.2$. Undifferentiated cells are not shown. Percentage of volume captured by the species is indicated in the bottom-right. Initial conditions are identical between variations and convergence was reached after a varying number of iterations. Yellow indicates the side of the Marching Cubes border where the respective species is not present.*

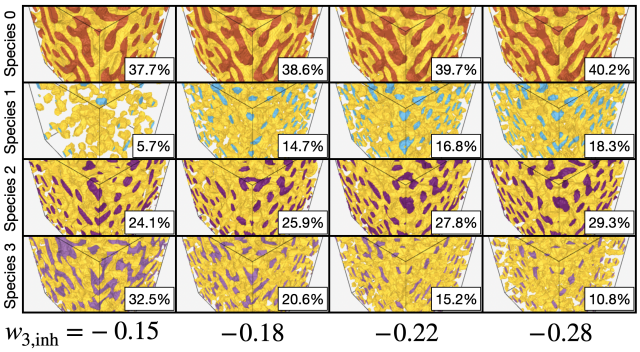

Figure 8: *Converged states of a CA with $L = 100$ and $K = 4$ for varying values of $w_{3,\text{inh}}$. Undifferentiated cells are not shown. Percentage of volume captured by the species is indicated in the bottom-right. Initial conditions are identical between variations and convergence was reached after a varying number of iterations. Yellow indicates the side of the Marching Cubes border where the respective species is not present.*

4.3. Discussing multi-species Turing patterns

It is evident from Figures 6, 7 and 8 that the described configurations reach convergence into a well-segregated state. This was confirmed with the convergence-checker.

By looking more closely at Fig. 7 we can see that changing the parameters of species 0 has a small effect on the structures that form in species 1, with the exception of the transition from $w_{0,\text{inh}} = -0.15$ to -0.18. It seems that, by capturing 70% of the domain, species 0 *'pushed away'* species 1, and when species 0 was made less dominant, species 1 took that opportunity to grow and capture 42.5%. After this, making species 0 less dominant did not have such large effect on species 1 any more. Similar behaviour is seen in Fig. 8, where changing the parameters of species 3 has little effect on the formation of structures in other species, except for species 1 in the transition from $w_{0,\text{inh}} = -0.15$ to -0.18.

If the species are balanced enough, like in the experiments above, the pattern formation of a certain species j is much more dependent on the parameters of that species j itself than on other species' parameters. However, species can be made such dominant that they have the tendency to drive other species away and possibly capture 100% of the domain. The latter came forward in an experiment that is not discussed in detail here.

4.4. Temporal evolution of the order parameter

To analyse the evolution of the order parameter over time, the same setup is used as before, where we have a CA with $L = 100$ and $K = 2$ with chemical configuration $R_{01,\text{act}} = 3.2, w_{01,\text{act}} = 1, R_{01,\text{inh}} = 6$ and $w_{01,\text{inh}} = -0.2$. The influence $w_{0,\text{inh}}$ is now varied from -0.10 to -0.30 in several steps. In Fig. 9, the order parameter ε_0 is plotted for every variation of $w_{0,\text{inh}}$ and for every iteration.

Fig. 9 shows how the order parameter behaves over time for different values of $w_{0,\text{inh}}$. For $w_{0,\text{inh}} = -0.1$, which results in a CA

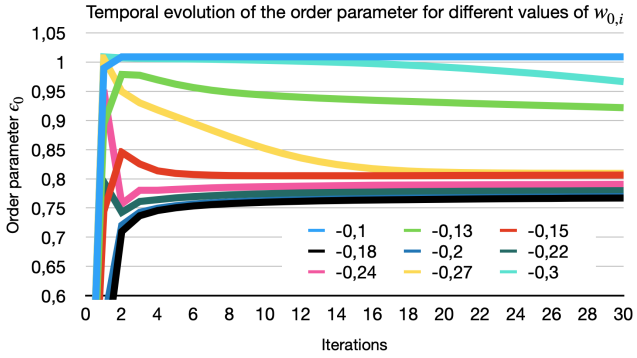

Figure 9: *Order parameter ε_0, plotted for every iteration and for various values of $w_{0,\text{inh}}$.*

that is entirely filled by species 0, the order parameter remains 1 (or slightly above). For all other values of $w_{0,\text{inh}}$, the CA would eventually converge to a well-segregated state and the order parameter attains a lower value. Furthermore, the parameter follows a different path over time for the different values of $w_{0,\text{inh}}$. Right after applying the initial state, when the CA is well-mixed, the order parameter attains a value close to zero, which is indicated in Fig. 9 by the steep slope after one iteration. This shows that the order parameter is able to distinguish between a well-mixed state ($\varepsilon \approx 0$), well-segregated states ($0 < \varepsilon < 1$) and states that are fully occupied by one species ($\varepsilon \approx 1$).

In section 4.2, it was established that altering $w_{0,\text{inh}}$, in this configuration, changes the resulting shapes and thus it seems that the order parameter reflects these changes by attaining different values (over time) for different $w_{0,\text{inh}}$. However, this is an initial observation and further research is required before one can be certain of this connection.

4.5. Chemical Parameter Space

To further investigate the connection between the order parameter and the resulting 3D structures, we turn back to section 3.5, where the Weighted Volume Difference (WVD_j) is defined. It is expected that, when two configurations have different values for WVD_j, they also develop different structures. We investigate whether the order parameter captures this.

Consider a CA with $L = 30$ and $K = 1$ and take that $w_{0,\text{act}} = 1, R_{0,\text{inh}} = 6$ and $w_{0,\text{inh}} = -0.2$. We now vary $R_{0,\text{act}}$ between 2.8 and 3.7 with steps of 0.001 and plot the order parameter ε_0 against WVD_0 after 50 iterations. The initial condition was kept random but unseeded, meaning every variation of $R_{0,\text{act}}$ had a different initial state. The result of this experiment can be found in Fig. 10. This experiment was later repeated with a seeded random initialisation, giving identical initial states for every variation. The result of that experiment can be found in Fig. 11. In addition, Fig. 11 includes images of the structures that appeared for the different values of $R_{0,\text{act}}$. From this point onward, all experiments are run with a randomisation seed that ensures identical initial conditions for every CA of the same size and number of species.

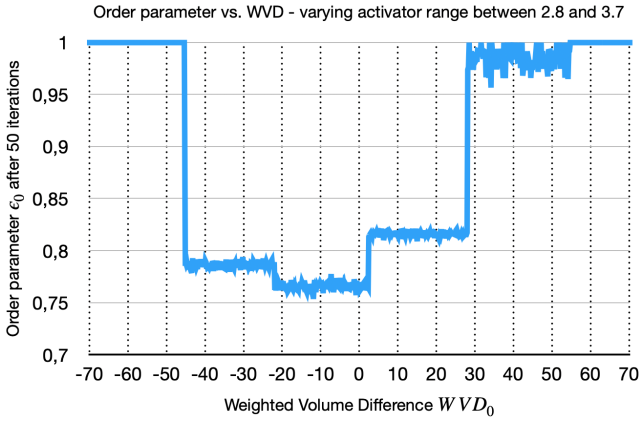

Figure 10: *Order parameter ε_0 versus WVD_0 as $R_{0,\text{act}}$ is varied between 2.8 and 3.7 with steps of 0.001. Initial condition is random but unseeded, meaning every variation of $R_{0,\text{act}}$ had a different initial condition.*

Both Fig. 10 and Fig. 11 show a step-response in the order parameter as the WVD_0 varies between -70 and 70. The reason for this step-response is that, as $R_{0,\text{act}}$ is increased with steps of 0.001, the WVD_0 immediately reacts because of its continuous nature. However, the CA does not react until $R_{0,\text{act}}$ has grown enough to include additional neighbours in $\eta_{0,\text{act}}(c)$ for its cells c. This is due to the discrete nature of the CA's domain. This was confirmed when looking at the structures that appeared for different values of $R_{0,\text{act}}$: Fig. 11 includes an image of the resulting structures for each of the steps in the order parameter. When creating these images, it did not matter which value for $R_{0,\text{act}}$ was chosen, as long as the WVD_0 lied between two values where the order parameter remained on the same step-level.

Continuing this reasoning, one would expect no such step-response to appear when, instead of the range $R_{0,\text{act}}$, the influence

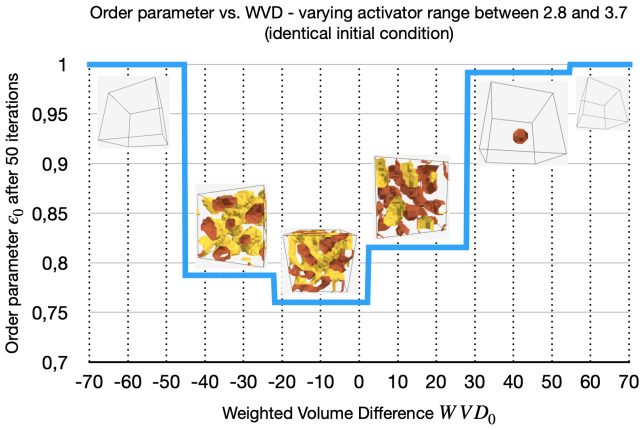

Figure 11: *Order parameter ε_0 versus WVD_0 as $R_{0,\text{act}}$ is varied between 2.8 and 3.7 with steps of 0.001. Initial condition is random and seeded, meaning every variation of $R_{0,\text{act}}$ had identical initial condition.*

$w_{0,\text{act}}$ was varied. Different values of $w_{0,\text{act}}$ do not rely on in- or exclusion of neighbours before the structure changes. This also follows from Fig. 7. To show this, the experiment was repeated and $R_{0,\text{act}}$ was locked at $R_{0,\text{act}} = 3$ and $w_{0,\text{act}}$ was varied between 0.75 and 2.05. The resulting plot can be found in Fig. 12.

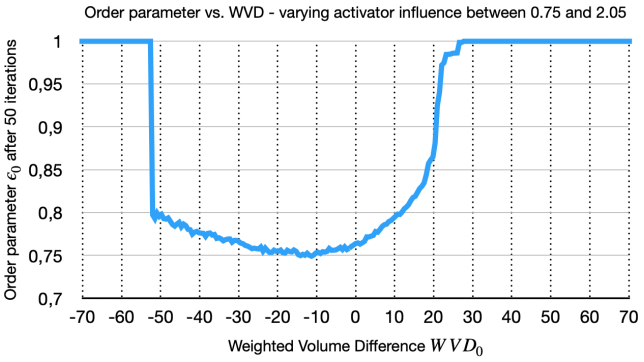

Figure 12: *Order parameter ε_0 versus WVD_0 as $w_{0,\text{act}}$ is varied between 0.75 and 2.05.*

Before discussing what these results mean, we first look at a multi-species variation of the experiment in Fig. 11. This time, $K = 3$ and the species have identical chemical configuration: $R_{12,\text{act}} = 3, w_{012,\text{act}} = 1, R_{012,\text{inh}} = 6$ and $w_{012,\text{inh}} = -0.2$. The activator range $R_{0,\text{act}}$ of species 0 is now varied between 2.8 and 3.7. In Fig. 13, all three order parameters $\varepsilon_0, \varepsilon_1$ and ε_2 are plotted against WVD_0.

4.5.1. Discussing the behaviour of the order parameter

Figures 10, 11, 12 and 13 show that the order parameter reacts to changes in the structures that develop within a certain CA, when either only the range or only the influence of a chemical is changed. However, they do not show a direct connection or correlation between the order parameter and the Weighted Volume Difference.

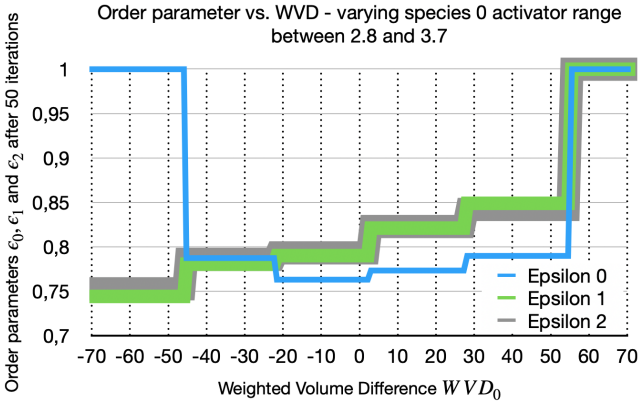

Figure 13: *Order parameters* ϵ_0, ϵ_1 *and* ϵ_2 *versus* WVD_0. *Species have identical chemical configuration:* $R_{12,\text{act}} = 3, w_{012,\text{act}} = 1, R_{012,\text{inh}} = 6$ *and* $w_{012,\text{inh}} = -0.2$. $R_{0,\text{act}}$ *is varied between* 2.8 *and* 3.7.

Further research must be done to establish such connection or correlation. At this point, it is therefore not yet justifiable to conclude that this order parameter can be used to classify developed structures. All we have shown is that this order parameter reacts to *changes in the structures* that appear.

Looking more closely at Fig. 13, it can be seen that the order parameters ϵ_1 and ϵ_2 of species 1 and 2 react to changes in the chemical-configuration of species 0. The parameters for species 1 and 2 remained unchanged during the entire experiment, yet still their order parameters changed.

From a mathematical standpoint, this behaviour could be explained as follows: As the chemical configuration of species 0 changes, that species will develop different shapes within the CA. We have seen this in Fig. 7. One could imagine that this also influences the number of cells that species 0 occupies after convergence. The CA therefore has more or less volume for other species to occupy, resulting in a different order parameter value for those species as well. The latter does not necessarily mean that other species develop *different shapes*, just that they occupy a different (possibly smaller or larger) set of cells within the CA.

It is harder to explain from another point of view, however. Namely if the desired behaviour of the order parameter was to quantify the type of shapes that develop. As Fig. 7 also shows, the type of patterns that develop in species 1 seems to not depend on the chemical configuration of species 0. Species 1 seemingly continues showing pipes, where species 0 clearly develops different shapes for different values of $w_{0,\text{inh}}$. Yet, Fig. 13 shows that the order parameter of species 1 would certainly have changed.

It is important to mention that this discussion relies on initial observations and that further research is required before a meaningful connection between the order parameter and the types of structures can be made. For now, this order parameter can solely be used to distinguish between a well-mixed state, well-segregated state and a state where one species dominates the domain. We saw this in section 4.4.

5. Conclusion

The main objective of this research was to discover how an existing 3D variant of Young's model could be generalised to produce multi-species Turing patterns. The generalisation, that is presented in this paper, indeed shows convergence to multi-species Turing patterns and the developed structures are seemingly similar to those found in [SP17] and [SRZ20]. A big part of this work was the step towards a GPU simulation, which showed to be significantly faster than our parallel CPU implementation. The second objective was to discover to what extent the formation of multi-species Turing patterns could be quantified through an order parameter. After looking at the temporal evolution of the order parameter for varying parameter-values, it was concluded that the order parameter is able to distinguish well-mixed states from well-segregated ones or states where one species fully dominates the domain. However, it is yet uncertain what role the order parameter plays in classifying different patterns.

Recommendations for further research include:

1. Classification of shapes, as they arise from our multi-species extension, can be performed to discover any differences with original (3D) Turing patterns.
2. Additional research must be done to discover the role of this order parameter in classifying the patterns that developed. Perhaps this can be connected to point 1.
3. When patterns can be classified and quantitatively distinguished, additional experiments can be done to discover what influence species have on each other in different settings.
4. Investigate how the shape of the kernel influences the formation of patterns. And what happens if species have different kernel shapes?
5. Investigate how the initial condition affects the formation of patterns (among different species).

6. Technologies

Most technologies and software that were used during this research are listed below. To implement the GPU simulation, documentation from *Metal - Apple* was used, alongside examples from the *Rust bindings for Metal* package for Rust.

Metal - Apple	https://developer.apple.com/metal/
Rust bindings for Metal	https://crates.io/crates/metal
Rust	https://www.rust-lang.org
Actix	https://actix.rs
Svelte	https://svelte.dev
Three.js	https://threejs.org
WebGL	https://www.khronos.org/api/webgl
TypeScript	https://www.typescriptlang.org
WebAssembly	https://webassembly.org, https://www.w3.org/TR/wasm-core-2, https://web.dev/webassembly-threads/

The codebase, used for this research, is available on GitHub.

7. Responsible Research

This section contains a few words on the responsibility and reproducibility of this work.

7.1. Connection to nature

In the introduction (section 1) and while discussing the background of this work (section 2), several connections of Turing patterns with the world around us have been mentioned. They are related to patterns on animal-skin, spatial self-organisation of ecosystems, experiments on the micro-scale, and much more. It should be noted that this paper does not contribute directly, in any way, to the (type of) Turing patterns that we see in the world around us. By no means should the results, presented in this paper, be directly associated with patterns in nature or in, for example, medical research. This paper is meant to investigate a new type of computer simulation of Turing patterns, that may form predictions for patterns that we may later discover in nature. Until then, all remains a simulation.

7.2. Apple Metal

During this research, it was decided to use Apple's Metal API to implement a CA simulation on the GPU. This decision was made for the following two reasons:

1. It was believed that using Metal directly, so not through an abstraction layer, would yield the best simulation performance on the Mac computer that was used for this research. Apple invested greatly in optimising graphics performance on the Mac through its Metal API.
2. The use of Metal means that the implementation, built for this research, can be accelerated through use of Apple's M-series ARM chips with little to no changes to the code.

The use of Metal does, however, have an impact on reproducibility. This implementation can be solely used on the macOS operating system. The web-segment of this implementation has been written with full reproducibility in mind. A lot of devices that run a browser will be able to view visualisations and instruct an external server to run simulations. The server is the only part requiring macOS. Still, creating a GPU implementation that works on multiple operating-systems can still be created by (partially) following the guidelines in this paper. An example of a Rust-library that can be used to create cross-platform GPU implementations is *wgpu*, a publicly available Rust library.

7.3. General reproducibility

During the writing of all the code, necessary to conduct this research, reproducibility has been taken into account to a significant extent. Others should be well able to not only run the program and create custom experiments, but also alter the implementation to produce different results. This paper includes steps that were required to adapt the GPU simulation to the K-species case and should be reproducible by someone who is familiar with the different libraries. Most of the libraries that were used are widely known and well documented. Apple provides a very extensive documentation on the use of Metal and the library that was used to utilise Metal in the Rust programming language is also well documented, including examples.

The codebase that is used for this research is available on GitHub. This can be used as an example or to run custom experiments with directly.

References

[AB18] ALSENAFI A., BARBARO A. B. T.: A convection–diffusion model for gang territoriality. *Physica A: Statistical Mechanics and its Applications 510* (2018), 765–786. Cited By :7. URL: www.scopus.com. 2

[AB21] ALSENAFI A., BARBARO A. B. T.: A multispecies cross-diffusion model for territorial development. *Mathematics 9*, 12 (2021). Cited By :2. URL: www.scopus.com. 2, 5

[Ada10] ADAMATZKY A.: *Game of life cellular automata*. 2010. Cited by: 109; All Open Access, Green Open Access. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84895224604&doi=10.1007%2f978-1-84996-217-9&partnerID=40&md5=30c60dae13f059f1b1d8a404a8b5fb33, doi:10.1007/978-1-84996-217-9. 2

[Bar81] BARD J. B.: A model for generating aspects of zebra and other mammalian coat patterns. *Journal of Theoretical Biology 93*, 2 (1981), 363 – 385. Cited by: 103. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0019785186&doi=10.1016%2f0022-5193%2881%2990109-0&partnerID=40&md5=da5f50501fc15d7bf0d85d9a71a2fc27, doi:10.1016/0022-5193(81)90109-0. 1

[Bar23] BARAJAS D. P.: *Implementation and Evaluation of an Order Parameter for the Reaction-Diffusion Model in a Cellular Automaton*. Tech. rep., Delft University of Technology, 2023. in preparation. 2, 5

[Gar70] GARDNER M.: The fantastic combinations of john conway's new solitaire game "life". *Scientific American 223* (Oct. 1970), 120–123. URL: https://doi.org/10.1038/scientificamerican1070-120, doi:10.1038/scientificamerican1070-120. 2

[KM10] KONDO S., MIURA T.: Reaction-diffusion model as a framework for understanding biological pattern formation. *Science 329*, 5999 (2010), 1616 – 1620. Cited by: 1033. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-77957347061&doi=10.1126%2fscience.1179047&partnerID=40&md5=baa5bcd9c577056b670af5d2a7e21d90, doi:10.1126/science.1179047. 1, 2

[Kon22] KONDO S.: The present and future of Turing models in developmental biology. *Development 149*, 24 (12 2022). dev200974. URL: https://doi.org/10.1242/dev.200974, arXiv:https://journals.biologists.com/dev/article-pdf/149/24/dev200974/2387786/dev200974.pdf, doi:10.1242/dev.200974. 2

[Mur81] MURRAY J.: A pre-pattern formation mechanism for animal coat markings. *Journal of Theoretical Biology 88*, 1 (1981), 161 – 199. Cited by: 266. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0019382427&doi=10.1016%2f0022-5193%2881%2990334-9&partnerID=40&md5=9cc3d6ec1691c0c8983c02b56f99704d, doi:10.1016/0022-5193(81)90334-9. 1

[PT17] PRINGLE R. M., TARNITA C. E.: Spatial self-organization of ecosystems: Integrating multiple mechanisms of regular-pattern formation. *Annual Review of Entomology 62* (2017), 359 – 377. Cited by: 56. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85011307863&doi=10.1146%2fannurev-ento-031616-035413&partnerID=40&md5=9a5eb8a5bf7c90f0e06c71669b7ac511, doi:10.1146/annurev-ento-031616-035413. 2

[SP17] SKRODZKI M., POLTHIER K.: Turing-like patterns revisited: A peek into the third dimension., 2017. 1, 2, 3, 8

[SRZ20] SKRODZKI M., REITEBUCH U., ZIMMERMANN E.: Experimental visually-guided investigation of sub-structures in three-dimensional turing-like patterns, 2020. arXiv:2006.16676. 1, 2, 3, 8

[Tur52] TURING A.: The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 237*, 641 (Aug. 1952), 37–72. URL: https://doi.org/10.1098/rstb.1952.0012, doi:10.1098/rstb.1952.0012. 1

[XLY*22] XIANG Z., LI J., YOU P., HAN L., QIU M., CHEN G., HE Y., LIANG S., XIANG B., SU Y., AN H., LI S.: Turing patterns with high-resolution formed without chemical reaction in thin-film solution of organic semiconductors. *Nature Communications 13*, 1 (2022), 7422. URL: https://doi.org/10.1038/s41467-022-35162-z, doi:10.1038/s41467-022-35162-z. 2

[You84] YOUNG D. A.: A local activator-inhibitor model of vertebrate skin patterns. *Mathematical Biosciences 72*, 1 (1984), 51 – 58. Cited by: 79. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-0021738836&doi=10.1016%2f0025-5564%2884%2990060-9&partnerID=40&md5=29c42276760ac2689c87a88697db3b68, doi:10.1016/0025-5564(84)90060-9. 1, 2