DOI: 10.1111/cgf.70324

EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Controllable Intrinsic Surface Pattern Generation Using Slime Mold Simulations

Jeffrey Layton[1], Faramarz Samavati[1], and Adam Runions[1]

[1]Department of Computer Science, University of Calgary, Canada

Abstract

Surface-based pattern simulations have proven valuable for texture design and scientific visualization, but existing methods face several limitations. Most simulations either target a narrow range of pattern types (e.g. spots, branching) or support a broad range of patterns at the cost of time-consuming parameter tuning. In either case, local and global control over the character of patterns is desirable, but often not supported. Additionally, transferring 2D simulations to 3D surfaces can introduce distortions, and is sensitive to mesh topology and quality. Finally, colourization further complicates the use of simulations for texturing, often relying on ad hoc mapping of simulation values to colours.

To address these challenges, we introduce a unified framework for generating expressive, controllable patterns that are naturally embedded on curved surfaces. We reformulate Physarum polycephalum slime mold simulations in terms of continuous rates and PDEs, allowing greater consistency across varying space and time discretizations. We introduce agent-based stochastic chemical kinetics to regulate agent turnover, which permits direct control over the uniformity of final patterns. Together, these modifications enable fine-grained control of pattern synthesis using spatially varying parameter maps, directional biases, stimuli, and agent sinks/sources. We demonstrate that our approach allows for the generation of new pattern classes in Physarum slime mold simulations, including stripes, branching, and hierarchical structures. To eliminate distortion and artifacts, we re-purpose intrinsic triangulations proposed for geometry processing to dynamic simulations. Finally, we introduce a simple colourization method to transfer colours from an exemplar image to simulation results.

Notably, while demonstrated through slime mold simulations, our framework generalizes to other patterning models (e.g. reaction-diffusion), thus providing a versatile tool for complex, controllable surface-based pattern synthesis.

CCS Concepts

• ***Computing methodologies*** → ***Modeling and simulation; Simulation types and techniques;*** *Agent / discrete models; Continuous models; Multiscale systems;* ***Artificial life;*** *Simulation tools; Procedural animation; Physical simulation;* ***Texturing;*** *Graphics systems and interfaces;* ***Mesh models;*** *Mesh geometry models;*

1. Introduction

Natural patterns combine structure with irregularity and variation. Zebra stripes are organized yet unique, leaf veins exhibit hierarchical branching that balances regularity with organic complexity, and rivers meander unpredictably yet purposefully. This interplay of order and chaos creates visually rich and complex patterns, but also poses challenges for methods attempting to generate them automatically. For such patterns, simulation-based methods have proven particularly effective, enabling the emergence of complex patterns from simple rules and physical constraints, without relying on curated datasets or overly descriptive methods. Furthermore, many natural patterns are inherently surface-bound, influenced by the geometry on which they emerge, such as the pigmentation on flowers [ROC*21], fish skin [KA95], leopard fur [DFW20], or sea shells [FMP92]. This surface embedding introduces unique challenges for pattern synthesis, requiring specialized approaches to handle curved surface geometry.

To effectively support natural pattern synthesis, we desire simulation methods that can: **(a)** generate a broad range of pattern types [Tur52; Pea93], **(b)** offer intuitive parameter control [STBB14, Sec. 10.2], **(c)** support spatial variation in pattern characteristics [ROC*21], and **(d)** produce global organization from local rules [Tur52; Jon10a], such as anisotropy or hierarchical structure [PL90; LBZ*11]. While all these goals are individually addressed in some existing methods, the literature does not currently offer a single simulation framework that satisfies all simultaneously. Moreover, many existing methods are designed around specific pattern families (e.g., spots, branching), achieving strong results in those cases but lacking the generality to span diverse pattern types.

To generate patterns on curved surfaces, a common strategy in-

volves parametrizing the surface onto a 2D domain for computational simplicity [WK91; FMP92; MMMR22]. However, parameterization introduces metric and curvature distortion, while the cuts used to flatten the surface can introduce noticeable seams. While some correction methods have been proposed [WK91; FMP92; MMMR22], they do not fully address these issues. Alternatively, simulating directly on meshes [Tur91; DGA04; ROC*21] avoids the distortions and discontinuity artifacts caused by parameterization. However, many patterning simulations involve solving PDEs over the patterning domain, which requires high-quality discretizations of the surface to support stable and accurate simulations [SGC21, ch. 4]. Thus, most meshes are not suitable due to poor triangle element quality. Remeshing techniques are often employed in these instances [BKP*10, ch. 6], but introduce a fundamental limitation, as remeshing alters the original surface representation and couples the texturing process to mesh generation, which is undesirable for downstream tasks that rely on the original mesh.

To address both sets of challenges, limitations in controllability and pattern diversity, as well as the geometric issues incurred by curved surfaces, we introduce a surface-based simulation framework. Our framework reformulates and extends Jeff Jones' Physarum slime mold model [Jon10a; Jon10b], selected for its ability to produce a wide range of pattern classes and the relatively interpretable influence of its parameters on emerging patterns. In this model, agents traverse the domain while depositing pheromones, which diffuse and decay to create a coupled dynamic that in turn guides agent behaviour. However, in its original formulation, pheromone dynamics are governed by discrete rules that make the model's behaviour sensitive to the choice of spatial and temporal discretization. Moreover, desirable patterns often arise only transiently, with no mechanism to regulate the uniformity of the final, steady-state, pattern. We extend and reformulate Jones' model by expressing pheromone dynamics as a PDE and redefining parameters as rates and ratios, improving consistency across spatial and temporal discretizations. We introduce an agent-based stochastic chemical kinetics process [Ros10, ch. 4-7, 11] based on particle decay to control the uniformity of the pattern generated at convergence. To further enhance pattern control and diversity, we leverage surface agent pattern stimuli (inspired by [Jon10a]), incorporate spatially varying parameters to locally blend patterns [WK91; ROC*21], and introduce directional agent biases as well as agent sinks and sources.

To support robust simulation on curved surfaces, we employ intrinsic triangulations [SSC19a; SGC21; GSC21], which offer improved mesh quality and numerical stability for PDE computations. Unlike conventional triangulations that depend on vertex positions in 3D space, intrinsic triangulations operate purely in the intrinsic geometry of the surface, preserving the extrinsic surface geometry. We use the intrinsic Delaunay Refinement (iDR) algorithm [SGC21, ch. 4] to produce uniform triangulations for pheromone discretization at specified resolutions. This intrinsic approach substantially improves pattern quality and consistency over prior surface-based slime mold implementations [MMMR22; GSR24]. Crucially, our agents interact solely through pheromone sampling and tracing operations, bypassing the costly geometric operations common in many geometric patterning simulations such as nearest-neighbour or geodesic computations [CLPQ20].

Finally, simulations produce a scalar field over the mesh which must be colourized. This is typically achieved by manually defining a colour map [Tur91; WFM01; ROC*21], which is a time-intensive process. We simplify this process by introducing an image-assisted colourization technique to extract a representative colour map from a target image, using nonlinear dimensionality reduction [TdSL00] on the image's pixel colours. This provides an initial colour map that can be readily tuned to obtain a desired colourization.

Our approach introduces several advances, including an intrinsic triangulation strategy that eliminates parameterization artifacts and supports robust surface-based simulation for both agent dynamics and PDE-based processes, as well as controllability mechanisms and image-assisted colourization for expressive pattern authoring. Combined with our extended slime mold simulation, these contributions establish a robust and versatile framework for procedural texture synthesis on curved surfaces, enabling efficient generation of diverse and controllable texture assets. Beyond aesthetic applications, the framework also highlights the broader utility of simulation-based approaches on intrinsic triangulations, offering a promising foundation for other dynamic simulations on curved surfaces.

2. Background Information

There are two primary approaches to pattern synthesis: data-driven and procedural techniques. Data-driven methods use existing pattern data to synthesize similar patterns, either using hand-engineered methods [Tur01; LH05; LH06; WLKT09; AYD*18; DSJ19; GAD*20] or neural-net approaches [GEB15; RBL*21]. These methods have utility in specific applications, but mapping pixel-space methods onto mesh surfaces is non-trivial without modifying the underlying representation, as their neighbourhood definitions assume a regular lattice structure. While surface parameterization can restore local pixel structure, seams disrupt neighbourhood continuity and alignment. Alternatively, a triangulated discretization embedded directly on the surface, bypassing parameterization issues can be used. Additionally, neural-net based approaches typically require extensive datasets to produce high-quality results.

Procedural methods have been widely used for texture synthesis [Pea85; Per85; Wor96; EMP*02; GAD*20]. However, they often produce simple results, or are augmented with other techniques to achieve greater complexity. A subset of procedural methods can be described as simulation-based due to their physically or biologically inspired formulations. These are particularly adept at capturing the structured yet irregular characteristics and evolving shapes of many natural patterns that other procedural methods struggle to reproduce. These methods are often biologically inspired and grounded in physical phenomena, enabling them to generate patterns with the complex organization of natural patterns. Moreover, simulation parameters often have direct physical or biological meaning, making their effects on emerging patterns easier to interpret and reason about, which is helpful when generating a desired pattern.

Simulation-based approaches can be categorized into two types: field-based and particle-based. Field-based simulations represent

values fixed in space that evolve over time through local spatial interactions. These approaches fall into two main categories: discrete and continuum models. The former defines field evolution using neighbourhood-based rules applied over discrete spatial structures such as grids or graphs [NPW84; Rob01; KSSL07; LM09; YSC*17]. The latter describes evolution using partial differential equations (PDEs) grounded in continuous formulations, with values usually defined as concentrations. When the continuum models are implemented numerically on discrete domains, they retain the mathematical structure and interpretation of continuous systems. However, the domain must be represented using a high quality discretization, such as Delaunay triangulations or regular grids, to avoid patterning artifacts or numerical stability issues.

Reaction-Diffusion (RD) models were among the first methods to compellingly simulate natural pattern formation via PDEs, especially biological pigmentation patterns [Tur52; Tur91; WK91; FMP92; Pea93; WFM01; QW12; DFW20; ROC*21]. Their versatility in generating a wide range of patterns has made RD models one of the most widely adopted simulation techniques. However, different classes of patterns typically require different RD formulations or parameter regimes, leading to a large and fragmented design space [ROC*21]. This, combined with parameters that are often unintuitive and difficult to relate directly to visual outcomes, makes precise control challenging for users. Additionally, many RD implementations operate on regular grids [WK91; FMP92; Pea93; DFW20], rely on specially prepared surface discretizations [ROC*21], or procedurally generate surface approximations suitable for stable computations [Tur91]. This limits their applicability to arbitrary meshes and complicates their use in general surface-based workflows. This is a general issue for many simulation models [STBB14, Sec. 10.2].

Alternatively, particle-based, or agent-based, methods simulate the local behaviours of individual entities that move through space to collectively produce global patterns. Ant colony simulations demonstrate self-organizing emergent behaviours that are useful for both patterning and computational purposes [DAGP90; DBT00; XRW*19]. There are also agent-based Laplacian growth model simulations [WS81; DGA04] that excel at branching patterns. Similar to Laplacian growth models, the space colonization algorithm [RFL*05; RLP07] uses iterative particle insertions to fill a specified region, excelling at branching structures and, with modifications, hierarchical network structures. These approaches often offer more intuitive and spatially controllable outcomes. However, they also often rely on repeated proximity or distance queries. While these operations are simple in Euclidean domains, they are much more difficult on curved surfaces [CLPQ20].

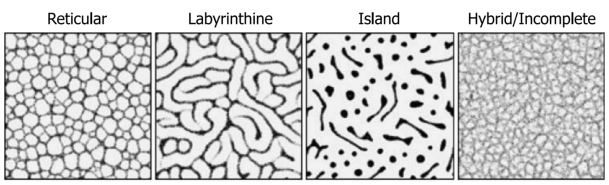

Figure 1: *Several representative patterns generated by Jeff Jones' slime mold simulation [Jon10a].*

Among particle-based models, simulations proposed by Jones, which were inspired by *Physarum polycephalum* (true slime mold), are particularly notable for producing a broad range of natural patterns (Fig. 1) [Jon10a; Jon10b]. We outline Jones' original formulation below, with a comprehensive description provided in Sec. 1 of the Supplementary Document. This model combines both particle-based and field-based aspects with a two layer approach: the agent layer and the pheromone layer. The agent layer consists of particles, defined with a position and heading direction. The particles respond to and deposit pheromones, which in turn is represented as a scalar field subjected to diffusion (box blur) and removal. During a simulation step, each agent samples the pheromones with three offset sensors (left/forward/right) at a fixed distance and lateral angle. Based on the sensed values, the agent (a) continues forward if the centre is largest, (b) turns toward the side that exceeds the centre if only one does, or (c) randomly chooses a side if both exceed the centre, introducing stochasticity into the simulation. The agent then advances one step and deposits pheromones. This establishes a stigmergic feedback loop where stronger trails attract more agents, further reinforcing those trails, while diffusion and decay spread and attenuate pheromones elsewhere. The interactions between the two layers produces complex and natural patterns. The pheromone field is typically visualized directly, with concentrations mapped to colours. Unlike many particle-based techniques, complex geometric queries, such as proximity and distance queries, are not used, as the pheromone layer indirectly controls particle interactions. However, some common natural patterns classes such as branching patterns cannot be easily created. Additionally, the original formulation is highly sensitive to the simulation time-step. For many parameter sets, agents tend to form a small number of irregular aggregations over time, eliminating patterning from large sub-regions of the domain, with no mechanism to regulate agent distribution.

Recent efforts have implemented this patterning simulation within a mesh-based framework, MoMaS [MMMR22]. This framework uses surface parameterization and GPU acceleration to improve performance. However, this approach introduces distortions and biases as a result of the pheromone parameterization and projection-based agent traversal approximation, in addition to the issues inherited from the Jeff Jones' original model [Jon10b; Jon10a]. These errors are reduced using a metric tensor and a by subdividing the traversal step, but still persist (Fig. 12). PhysOM [GSR24] extends the Jones model with diffusion-based anisotropic effects, similar to some reaction–diffusion models [ROC*21], to control pattern orientation within volumetric domains. However, its formulation targets volumetric structures rather than intrinsic surface pattern synthesis.

For surface bound simulations, operating directly on the mesh provides an alternative to parameterization based methods [Tur91; WFM01; DGA04; LBZ*11; AYD*18; DFW20; ROC*21], bypassing parameterization issues and accounting for the surface curvature and geometry. This approach comes at the cost of working on an irregular domain, making computations more difficult and expensive. Additionally, these methods often require remeshing to ensure a high-quality surface domain suitable for simulation. It should be noted that most surface texturing applications assume the mesh is orientable and has a manifold structure [SGC21, pg. 13]. We also

assume all meshes used in our framework are orientable and manifold.

Overall, prior work on pattern synthesis faces persistent challenges when applied to geometric surfaces, including discretization sensitivity, parameterization artifacts, and limited surface-bound controllability. Addressing these issues requires surface-aware formulations that remain robust to mesh structure while supporting expressive and interpretable pattern synthesis.

3. Methodology

This section presents the main components of our framework. We first describe our reformulation of Jeff Jones's slime mold model [Jon10b; Jon10a]. Notably, Jones' formulation is sensitive to changes in the discretization of space and time, as well as the number of agents (see Fig. 1-3 in the Supplementary Document). This sensitivity makes it difficult to obtain equivalent patterns on different meshes when the time-step is adjusted, or agent number is changed. Moreover, over time, agents tend to form a small number of irregular agent aggregations, making it difficult to obtain patterns with a uniform character over the entire domain (see Fig. 4 in the Supplementary Document). To address these limitations, we express agent motion and pheromone dynamics in continuous terms. We also rephrase agent turnover and aspects of pheromone dispersal as stochastic chemical kinetic processes, which allows us to maintain approximately the same distribution of pheromones overtime, even as agent number, surface area, and other pheromone parameters are varied. These changes mitigate discretization sensitivity and enable the generation of patterns with a more uniform, and controllable, character. Our reformulation is required for the controllability extensions we describe that provide fine-grained spatial control over the character, density and anisotropy of patterns (Sec. 3.3), and broaden the range of attainable pattern classes to include striped and branching patterns. It is also required for the direct implementation of our method on surfaces meshes, where irregular triangulation and variable element sizes must be accounted for.

To move our simulations to surfaces, pheromones are stored as vertex concentrations on a manifold triangulation, and agents are stored with the face containing them, with positions and orientations expressed in local coordinates. We adopt intrinsic triangulations [SSC19a; GSC21; SGC21] to support surface-based simulations while avoiding artifacts stemming from parameterization or remeshing.

Finally, we present an image-assisted colourization method that extracts a colour map from an exemplar image.

3.1. Intrinsic Triangulation Surface Simulation

To represent our surface, we use an intrinsic Delaunay triangulation based framework consisting of two spaces: the intrinsic triangulation itself and a collection of intrinsic faces with associated local coordinate systems. These spaces respectively provide domains for pheromone dynamics and agent traversal processes. The integration of these processes in our framework are described in this section and depicted in Fig. 2a.

To simulate continuous pheromone concentrations, we must solve a PDE over the mesh (see Sec. 3.2 for details). This constrains our choice of triangulation, as poor mesh geometry induces artificial anisotropy and nonlocal coupling of concentrations in diffusion simulations. Here, Delaunay triangulations are advantageous, as they preserve locality and isotropy by enforcing well-centred neighbourhoods and well-defined Voronoi (circumcentric) dual geometry [CdGDS13, ch. 6]. While this property can be satisfied via remeshing, this would also alter the extrinsic mesh geometry. To avoid this issue, we use an intrinsic triangulation to construct an appropriate triangulation using intrinsic Delaunay Refinement (iDR) [SSC19a; SGC21; GSC21]. As seen in Fig. 3, intrinsic triangles may not be flat in 3D (d, inset). However, the iDR algorithm ensures each triangle can be isometrically bent into a valid 2D triangle in its own local coordinates. This eliminates distortion between the intrinsic and extrinsic spaces, providing both a consistent surface manifold representation and a compatible local planar space. For our application, we also require that triangulations have sufficient resolution to represent synthesized patterns. We achieve this by restricting the iDR algorithm to ensure no triangle has a circumradius above a specified threshold. This radius defines the resolution of the intrinsic triangulation (compare Fig. 3c and 3d).

Agent dynamics now operate per face, and require appropriate methods to traverse the mesh and deposit pheromones. Mesh traversal is required to update agent positions and determine sensor positions. In either case, this is computed by performing a series of local-local coordinate transformation along internal edges [SGC21, pg. 26-28]. When agents reach a boundary edge, we reflect their trajectory to keep them moving within the domain, preventing clustering at boundaries. In contrast, sensor traces terminate at boundaries to avoid unrealistic pheromone readings outside the surface. Point sampling and point deposition of pheromones use barycentric interpolation to map between face-local positions and vertex-based pheromone values.

Fig. 4 illustrates the importance of maintaining the Delaunay property, highlighting its impact on both pheromone diffusion and agent dynamics. While one could re-mesh instead, this would not preserve the underlying geometry. Although remeshing may be sufficient in this simple case, the intrinsic framework generalizes robustly to more complex cases (e.g., Fig. 13).

3.2. Extended Slime Mold Simulation

We reformulate Jones' slime mold model (Fig. 2) so that pheromone concentrations are governed by physically based diffusion–decay PDEs, representing a continuum process rather than discrete update rules. We also adapt the model to provide time-step independent simulations, improved pheromonal deposition along agent trajectories (Fig. 2c), and agent-based stochastic chemical kinetics to regulate pattern uniformity by integrating agent turnover (Fig. 2d). These extensions support more consistent pattern formation across discretizations, greater interpretability and predictability of parameters (and their effect on patterns), and expanded expressive capabilities for pattern synthesis. The simulation parameters for this reformulation are summarized in Table 1, and can be compared against the parameters of Jones' model (Table 1 in the Supplementary Document).

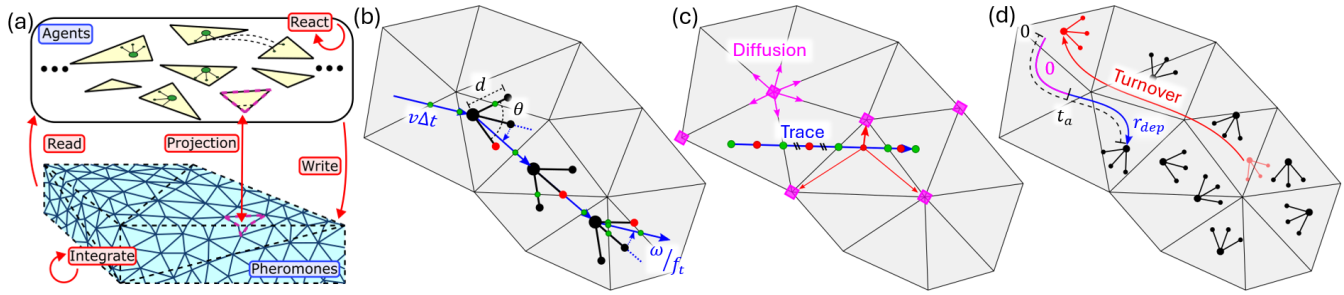

Figure 2: *Diagrams depicting the core elements of our surface-bound slime mold model. (a) The two simulation spaces of our intrinsic triangulation framework (intrinsic triangulation and a collection of intrinsic faces with associated local coordinate systems), and their relations. Each face in the collection stores local-to-local coordinate system transformations for adjacent intrinsic triangles, defining the surface for agent traversal. The projection between spaces is implied by the correspondence between local face coordinates and the intrinsic triangulation on the surface (dotted purple lines). Arrows labelled read, write, react, and project indicate how agents interact with the pheromones across these spaces. (b-d) Diagrams illustrate key components of our simulation on a small region on an example triangulation. (b) Agent traversal. Three traversal steps are shown, where the direction of turning (left, no turn and right) is indicated by the red sensor. The step distance is obtained by integrating the velocity v over Δt, and the turn angle is obtained by integrating the angular velocity ω over $1/f_t$. (c) Line-integrated pheromone deposition and diffusion. Each step is subdivided into per-face segments, with deposits allocated proportional to their length. Pheromone is distributed to face vertices using the barycentric weights of the midpoint (red circle). Pheromones then diffuse between vertices and decay locally. (d) Agent turnover and deposit activation. As agents turnover redistributes older agents across the mesh. Newly activated agents do not deposit pheromones (purple) until reaching an age of t_a (blue).*

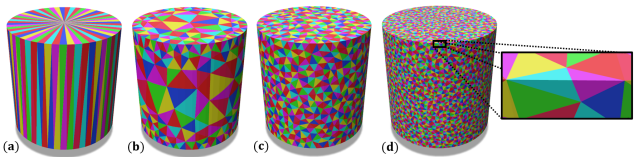

Figure 3: *The intrinsic triangulation used for simulations. (a) An extrinsic mesh consisting of large non-Delaunay triangles. (b) The intrinsic triangulation constructed using unconstrained iDR, with large variation in element size, and coarse elements that cannot represent simulated patterns. (c) Intrinsic Triangulation constructed using the circumcircle constrained iDR to respect a larger (c) and smaller (d) maximum radius.*

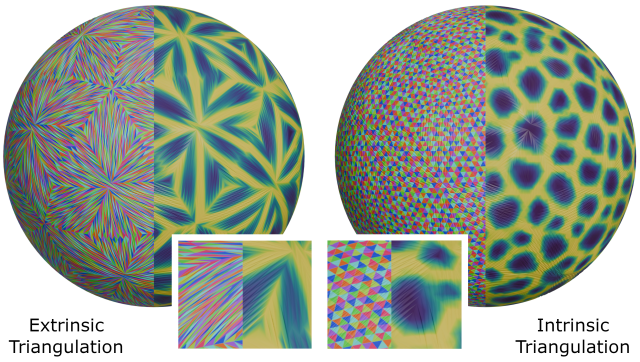

Figure 4: *The effect of mesh quality on simulated patterns, demonstrated using a (Left) spherical mesh with long skinny triangles and (Right) the same mesh following the application of the iDR without vertex insertions (i.e. application of the Delaunay edge flip algorithm) [SSC19a; SGC21; GSC21]. Each panel is split to show both the triangulation and resulting pheromones. The extrinsic model fails to form the target pattern (Left) while the intrinsic triangulation reproduces the pattern without changing the extrinsic geometry (Right).*

To eliminate the model's dependence on time-step (Δt), we converted change-based parameters to rate-based ones. First, agent dynamics are now described using a turn rate (ω), move rate (v), and deposit rate (r_{dep}). The agent geometry, sensor angle (θ) and sensor distance (d), remain unchanged. However, there is a more subtle agent dynamic that impacts the resulting pattern, the frequency of the turning decision. The distance travelled between turns is sometimes a meaningful pattern characteristic rather than a time-step artifact. For example, patterns containing sharp angular changes in features may require a straight trace followed by a sharp turn, rather then continuously turning as the agent traces. To remove time-step dependence, we fix the turn decision frequency (f_t) by tracking the elapsed time since the last decision. Turns are evaluated only when the set period is reached, modelling turning as an impulse event that accumulates the turning magnitude until it is time to evaluate. Additionally, r_{dep} is described as a time-dependent deposition rate, uniformly distributed along the agent's path (Eq. 4). To correct for differences in element areas when updating pheromone concentrations, we must normalize the deposit amount by the element's area, converting pheromone amounts to concentrations (e.g. pixel or, in our case triangle area).

To make pheromone dynamics independent of both time-step and spatial discretization, we model this process using a standard

diffusion and decay PDE,

$$\frac{\partial u(\mathbf{x})}{\partial t} = r_{dif}\nabla^2 u(\mathbf{x}) - r_{dec}u(\mathbf{x}), \tag{1}$$

where $u(\mathbf{x})$ is the concentration of pheromones at $\mathbf{x}$, r_{dif} is the diffusion rate, and r_{dec} is the decay rate. The diffusion is represented by the Laplacian term, capturing spatial movement between elements, and decay is represented by the linear term, capturing exponential decay. Having our dynamics standardized in a PDE allows simulations via finite element methods, enabling accurate simulations on different discrete representations (e.g. grids or triangles). Our formulation normalizes the factor decay step of Jones' formulation with a time-dependent decay term. This produces smoother pheromone gradients that improve the colourization of synthesized patterns. For our surface triangulations, we use the cotangent Laplacian with a lumped vertex mass matrix [CdGDS13, ch. 6].

The temporal dynamics of pheromones are simulated using Euler integration schemes. In most cases forward Euler (Eq. 2) suffices. In situations where convergence issues arise, each time-step can be subdivided into several integration steps. Alternatively, backward Euler (Eq. 3) integration provides a stable method for larger time-steps, but at an increased computational cost per integration step. The schemes can be expressed in matrix form as:

$$\mathbf{M}\mathbf{u}^{n+1} = [\mathbf{M} - \Delta t(r_{dif}\mathbf{L} + r_{dec}\mathbf{M})]\mathbf{u}^n, \tag{2}$$

$$[\mathbf{M} + \Delta t(r_{dif}\mathbf{L} + r_{dec}\mathbf{M})]\mathbf{u}^{n+1} = \mathbf{M}\mathbf{u}^n, \tag{3}$$

where $\mathbf{M}$ is the Mass matrix, $\mathbf{L}$ is the Laplacian matrix, $\mathbf{u}^n$ is the current concentration values, and $\mathbf{u}^{n+1}$ is the updated concentration values.

Having reformulated agent motion and pheromone diffusion in continuous terms, we next consider pheromone deposition. To make deposition consistent with our continuous formulation, pheromones are uniformly distributed along the path traced by each agent during a time-step, rather than depositing them at the end of the step as in Jones' original formulation and the MoMaS extension. This modification makes deposition independent of the simulation time-step, and allows sufficient pheromone coverage to be achieved with fewer agents. As a result, the number of agents required to synthesize patterns can be reduced, improving the computational efficiency of simulations, particularly on high-resolution spatial discretizations. In a finite element context, we have a set of vertex elements $\mathbf{V}$. Let $\mathbf{x}(s)$, where $s \in [0,1]$, be the arc length parameterized curve representing the trajectory of an agent. For a given position $\mathbf{x}$, let $w_i(\mathbf{x})$ denote the basis function weight attributed to the element $v_i \in V$. The uniformly distributed weight along a line attributed to v_i can then be expressed as a line integral:

$$w_{eq,i} = \int_0^1 w_i(\mathbf{x}(s))\, ds. \tag{4}$$

For each agent step on our surface triangulation, deposits are distributed across the intrinsic faces traversed, with each face receiving a contribution proportional to the length of the segment as shown in Fig. 2c. Assuming linear vertex basis functions, the resulting contribution is equivalent to a point deposit at the midpoint of the segment in each face (as derived in Sec. 3.2 of the Supplementary Document).

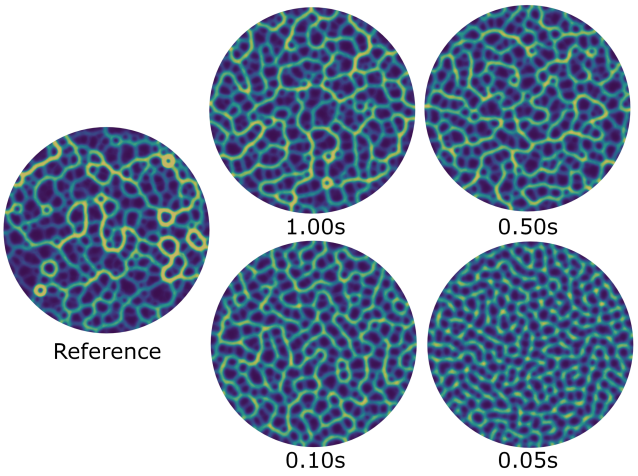

Figure 5: *Effect of varying mean particle lifetime $\bar{\tau}$ on simulation results (from 1.0-0.05s). A constant $U = 0.15926$ is obtained by increasing the agent count as mean particle lifetime decreases. In the reference pattern, agent turnover is disabled and the resulting pattern after 30s (3000 iterations) of simulation time is shown. The deposit activation time is fixed at 0.2s.*

Finally, we address issues with agent migration that cause patterns to converge to sparse non-uniform patterns by introducing agent turnover (Fig. 2d). Specifically, as agents move toward regions where pheromone concentrations are high, there is a tendency for agents to increasingly aggregate along a smaller number of paths as simulations advance. This produces patterns with a non-uniform character (Fig. 5, reference panel and Fig. 4 in the Supplementary Document). As such, we introduced a stochastic chemical kinetics model [Ros10, ch. 4-7, 11] based on particle decay and renewal, a memoryless stochastic process, where each particle has a constant probability of decaying at any given time. This process results in an exponential distribution of particle lifetimes specified by the mean particle lifetime $\bar{\tau}$. This allows better control over the uniformity of the pattern by reseeding agents across the domain once their lifetime expires, preventing uncontrolled accumulation and sustaining global coverage.

Under a global $\bar{\tau}$, it suffices to generate the lifetime directly and compare it to the agent's age. However, to support spatial variation in agent lifetimes (Sec. 3.3), we use the hazard rate function for random variables [Ros10, ch. 11.2.3] which uses an exposure, calculated by normalizing the age (time) accumulation by the local mean lifetime. Each agent maintains an exposure λ, age a, and its exponentially distributed exposure threshold $\lambda_t = -ln(X)$ where X is a uniform random variable in the range $(0,1)$, sampled once upon turnover at the start of a new lifetime. Once $\lambda \geq \lambda_t$, the agent decays and is removed from its current location and placed randomly in the domain. Since we will be using a triangular discretization of space with variable face sizes, we randomly choose a face weighted by the face area and then place the agent randomly inside. To ensure constant time face selection, we employ the Vose Alias Method [Vos91]. However, this introduces other issues, since the agent is uniformly placed, agents will spawn in areas with low

pheromones, depositing pheromones and potentially disrupting the existing pattern. Though this disruption may sometimes be desirable, such as for space filling patterns (similar to Fig. 5), it is useful to be able to explicitly control this influence. To tune this behaviour, a deposit activation time t_a can be specified, so agents will only deposit if $a \geq t_a$.

Jones' original model is sensitive to both domain size, directly controlled by image resolution, and particle count, as demonstrated in Fig. 1-3 of the Supplementary Document. To ensure the pattern character remains consistent when varying domain sizes, particle discretizations, mean particle lifetime, or deposit activation time, we propose the Uniform Steady-state Pheromone Concentration metric, denoted U. This metric approximates the average pheromone concentration once the simulation reaches steady-state. It accounts for the many factors contributing to pheromone abundance, including the number of agents, their deposit rate, pheromone decay, the size of the domain, and the expected number of agents depositing at any given time. The U metric is calculated as,

$$U = \frac{N r_{dep}}{r_{dec} A} e^{-\frac{t_a}{\bar{\tau}}}, \tag{5}$$

where N is the number of agents and A is the simulation domain area (surface area). Using this approach, we can maintain the character of a pattern when changing the simulation space (mesh), varying the number of particles, or tuning activation time by adjusting a compensating parameter. Any parameter present in Eq. 5, aside from agent turnover timing quantities, can be changed to ensure the preservation of the U metric without significantly affecting pattern character.

Symbol	Name	Units
Δt	Time Step	s
r_{dec}	Decay Rate	$1/s$
r_{dif}	Diffuse Rate	m^2/s
N	Number of Agents	$-$
d	Sensor Distance	m
θ	Sensor Angle	rad
v	Move Rate	m/s
ω	Turn Rate	rad/s
r_{dep}	Deposit Rate	mol/s
$\bar{\tau}$	Mean Lifetime	s
t_a	Deposit Activation Time	s
f_t	Turn Decision Frequency	$1/s$

Table 1: *Parameters for our Extended Slime Mold Simulation.*

3.3. Controllability Extensions

Many natural patterns exhibit predictable spatial variation, necessitating methods to locally control the character of simulated patterns. For some pattern classes, such as branching and striped patterns, local control over the orientation of features is required. To provide convenient spatial control over simulated patterns, we employ surface maps that vary over the surface to specify local parameters, and bias agent motion via pheromones and direction fields.

Each controllability extension relies on surface maps that define local values over the surface domain. Maps can be specified either directly on intrinsic elements (vertices and faces) or, when a texture parameterization exists, by sampling textures into the intrinsic elements. Scalar quantities are defined at vertices and interpolated across faces using barycentric coordinates, while tangent vector quantities are defined per face [dGDT15, Part 1].

There are several ways to construct such maps, depending on the application. The simplest is manual authoring through hand-painted textures, which can be further processed with standard image-based methods or directly on the surface geometry to account for curvature. In our framework, we use three surface-based methods for constructing scalar maps: short-time diffusion to blur existing maps in accordance with the surface geometry, surface distance field solvers [MMP87; SSK*05; CWW13; FC24] to generate surface-aware distance fields, and Laplace-based solvers [CdGDS13, ch. 6] to smoothly generate interpolated fields between regions. For vector maps, a straightforward and reliable strategy is to define a scalar map and compute its finite element gradient [dGDT15, p. 13], yielding a tangent vector field. In our examples, these scalar maps are frequently derived from the smoothly interpolated fields produced by the Laplace-based solvers or from surface distance fields defined using the methods described above. This gradient-based approach worked well in most cases and is commonly used to specify directions in biological models of morphogenesis [KCGB11; ROC*21]. These map construction approaches are implemented as intrinsic surface processing tools in our system, with full details provided in Sec. 4 of the Supplementary Document.

Pheromone maps (Stimuli): Specifying maps that alter agent perception or pheromone levels provides an effective way to embed features in patterns. We implement this in two ways. The first, read-stimuli, uses an additional static pheromone layer that attracts or repels agents [Jon10a]. As an indirect influence, it can seamlessly embed features (Fig. 7, sign text) or organize patterns across successive simulations (Fig. 16, thinning reticulate lines). The second, write-stimuli, directly adds or removes pheromones, acting as a direct pheromone influence independent of agents. While write-stimuli can yield artificial features, this approach supports stronger embedding, such as the perimeter dots in Fig. 7.

Parameter maps: Spatial variation of parameters is a classic approach for achieving fine-grained control of simulation models, including texture synthesis on meshes (e.g. reaction-diffusion flower pigmentation [ROC*21]). To implement this, we store per-vertex values for affected parameters. In examples where smooth pattern transitions are desired, these maps are constructed using the surface-based scalar map methods described above to produce smooth gradients over the mesh, enabling continuous transitions between regions with different pattern characteristics. Accounting for spatial-variation in the parameters controlling pheromone dynamics requires us to replace the scalars in the PDE from Eq. 2 and 3 with matrices capturing element (vertex) level parameter values. Thus our forward and backward Euler integration become:

$$\mathbf{M}\mathbf{u}^{n+1} = [\mathbf{M} - \Delta t(\mathbf{R}_{dif}\mathbf{L} + \mathbf{R}_{dec}\mathbf{M})]\mathbf{u}^n, \tag{6}$$

$$[\mathbf{M} + \Delta t(\mathbf{R}_{dif}\mathbf{L} + \mathbf{R}_{dec}\mathbf{M})]\mathbf{u}^{n+1} = \mathbf{M}\mathbf{u}^n, \tag{7}$$

where $\mathbf{R}_{dif}$ and $\mathbf{R}_{dec}$ represent diffusion and decay coefficient matrices, respectively.

For agent dynamics, spatially varying parameters can be interpolated at the agent's position to determine their behaviour during a given iteration. Since we use the hazard rate function to determine when an agent decays, spatial differences in mean lifetime are incorporated directly into the exposure increment. This keeps the decay condition independent of the agent's position. However, because deposition is determined by the local activation time, initializing all agents with an age and exposure of 0 causes regions with shorter activation times to begin depositing earlier, creating a transient pheromone imbalance. Agents respond to this imbalance and drift toward shorter-activation regions, introducing a pattern bias that is not present at steady state. To remove this artifact, we initialize agent age and exposure to match the expected steady-state distribution. When doing so, care must be taken to avoid the bias introduced by the tendency for observed lifetimes to appear longer than expected (a.k.a. the inspection paradox). Exponential decay is memoryless, so an agent that has already survived to a given exposure behaves like a newly initialized agent with respect to future decay. We therefore compute λ_t by adding an independently sampled remaining exposure to the randomly generated elapsed exposure [Ros10, ch. 7.7]. Finally, we introduce a spawn-weight probability map to allow for customized agent seeding (Fig. 14, 16, 19). This is accomplished by multiplying the face element weight (area) by the spawn-weight probability values.

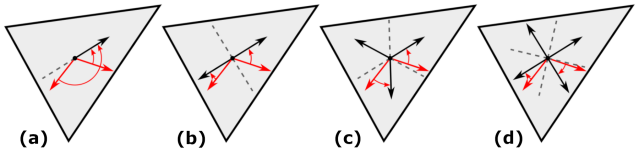

Figure 6: *A face with a bias direction with increasing symmetries. The black arrows represent a symmetric direction and the red arrows represent example agent directions and the direction they will rotate towards. (a) symmetry 1, (b) symmetry 2, (c) symmetry 3, (d) symmetry 4.*

Directional effects: To introduce directional effects, such as anisotropy and n-fold symmetries, into patterns we bias the direction of agent motion. This bias is defined using a vector field over the mesh faces, which determines the direction and strength (determined by the magnitude of the vector) of the bias. During each step, the agent rotates toward the target direction at a rate proportional to the angle between the current and target direction multiplied by the strength of the bias. To account for n-fold symmetries in synthesized patterns we use n rotated copies of our bias vector, as depicted in Fig. 6. In this case, the agent rotates towards the closest copy (smallest angle). When $n = 1$ or 2, the directional bias increases pattern anisotropy. See Sec. 6 in the Supplementary Document for additional evaluations of the patterns caused by various directional bias weights and symmetries.

The above extensions can be utilized together to create a seamless pattern that follows our specification, as exemplified by Fig. 7. The write-stimulus embeds dots directly in the pattern that diffuse pheromones. The read-stimuli in the middle of the sign attracts agents to the text, seamlessly embedding this artificial constraint in the pattern without the artificial edges that would appear if the write-stimuli was used. The bias gradient elongates the pattern anisotropically along the perimeter, but this influence is excluded from the interior of the sign by the bias weights, creating a more isotropic pattern in this region. Additionally, the bias region seamlessly blends with the isotropic region in the middle.

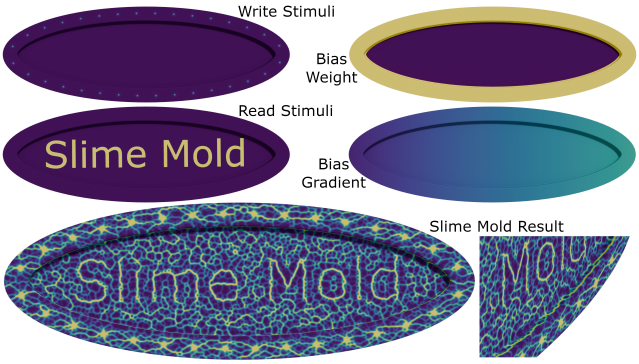

Figure 7: *Slime mold sign created using a combination of controllability extensions. The sign is generated using four maps: a write stimuli with regular dots along the border; a read stimuli with the sign's text; and two maps to specify an anisotropic bias on the signs border (a bias weight map and a bias gradient to specify the direction of anisotropy).*

We can use agent turnover to control patterns by introducing non-uniform turnover, or by defining sinks and sources. For particle decay, we can leverage parameter maps to locally define mean lifetime, allowing users to specify regions of high and low decay. At the extreme (mean-lifetime of 0), a region where all particles decay instantly is produced, creating an agent sink. Combining these regions with anisotropic biases pointing toward the sink promotes branching patterns emanating from the sink. Conversely, we can locally specify spawning probabilities to create sources by setting probabilities in other regions to zero.

By utilizing the mean lifetime and spawn weight probabilities, we can define agent sinks and sources. Sinks arise where lifetimes vanish, while sources are regions of high spawning probability. These mechanisms allow us to go beyond controlling pattern character, enabling new classes of patterns such as branching patterns rooted at sinks or radiating from sources. Fig. 8 illustrates how we can use sinks and weighted spawning to generate branching patterns. The unidirectional bias is pointed toward the centre with the agents spawning inline with the direction of the bias. With the agents aligned to the bias direction upon turnover, we can see very limited cycles, resulting in nearly pure branching patterns (unlike Fig. 15). By varying the bias weight, we can see that this parameter controls the perpendicular (to the bias direction) density of the branches, while the agent turn rate and sensor angle controls the dichotomous branching angle. Notice how the empty regions are filled with terminating branches, where agents are spawned before

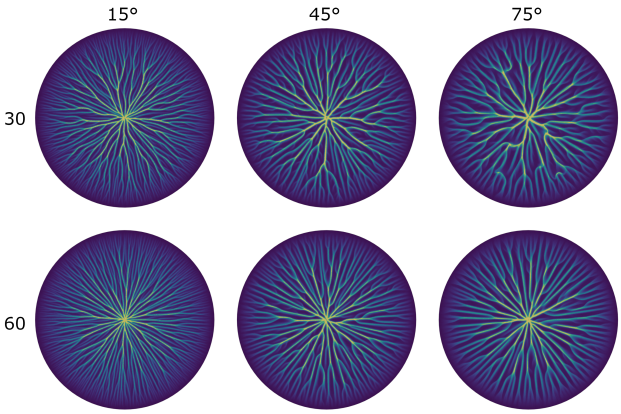

Figure 8: *Branching patterns generated by combining the slime mold simulation with controllability extensions. The patterns emerge from a unidirectional bias towards a sink at the centre. Bias strength varies from 30 (top row) to 60 (bottom row). Sensor angle and turn rate are increased from left-to-right (15-75 degrees). All simulations produce dichotomous branching patterns.*

merging with the main branching lines that extend to the perimeter of the circle.

3.4. Image-Assisted Colourization

To produce realistic textures, we map pheromone concentrations to colours using an exemplar-derived colour map. To simplify the reproduction of observed patterns, we developed a method to assist in the generation of colour-maps from exemplar images. Given an image, the method finds a 1D parameterization of the image pixels in the RGB colour space. To construct our parameterization, we first map the pixel colours to 1D using non-linear dimensionality reduction (ISOMAPs [TdSL00]). Using these 1D values, we construct a parameterization by fitting a cubic spline for each colour channel and then regularly sample the resulting curves to generate a colour map. If needed, the extracted colour map can then be interactively fine-tuned to refine the colour mapping. As our method assumes that the mapping from concentrations to colours is continuous; it can fail when this mapping should be discontinuous or cannot be explained by a single variable. Nonetheless, our semi-automated method simplifies the colourization process compared to the manually defined colour-maps used in many simulation based approaches (e.g. flower pigmentation [ROC*21], leopard spots [DFW20]), producing plausible results in many cases (e.g. Fig. 14, 16-18, 20).

4. Implementation

We implemented our methods in C++, using Polyscope [Sha*19] for program visualization and Geometry Central [SC*19] for our intrinsic triangulation data structure. We used the integer coordinates [GSC21] implementation of the intrinsic triangulation data structure, as we found it to be more robust compared to the sign-post [SSC19a], which sometimes failed for high resolution (small maximum circumcircle) triangulations. The full implementation can be found at `https://csgit.ucalgary.ca/jeffrey.layton/intrinsic-surface-pattern-generation.git` and is detailed further in Sec. 4 of the Supplement Document.

All experiments were run on a workstation with an AMD Ryzen 9 5950X (16 cores, 32 threads), 128 GB RAM, and an NVIDIA GeForce RTX 3080 Ti (driver 581.29). Both our implementation and the MoMaS baseline were compiled using MSBuild with the RelWithDebInfo configuration. Our code was written in C++23, while the MoMaS baseline was written in C++17 and uses CUDA 13.0. For comparisons with MoMaS, we used the published implementation [MMMR22].

5. Evaluation

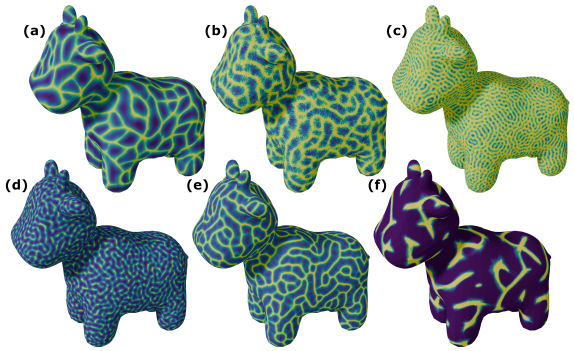

Figure 9: *Six different slime mold simulations results using the extended slime mold simulation method on the Spot model [CPS13]. (a) Shallow angle simulation with* 1*s mean lifetime and* 3*s activation time. (b) Small move and turn rate with* 1*s mean lifetime and activation time. (c) Chemo-repulsion scheme (negative sensor angle), similar to that described in [Jon10a]. (d) Small move and turn rate with* 0.1*s mean lifetime and activation time. (e) Disabled activation time. (f) Large sensor distance with* 0.2*s mean lifetime and* 0.4*s activation time.*

Our method can generate a broad range of patterns on surfaces (Fig. 9). Moreover, the particle stochastic chemical kinetics process we employ allows for explicit control over the uniformity of patterns. Our method is formulated to provide a continuous representation that remains well-behaved across different temporal, spatial, and particle discretizations. We demonstrate the effect of varying intrinsic triangulation resolution, particle discretization, and time-step discretization while maintaining a consistent pattern character (Fig. 10), which is not possible using Jones' original method [Jon10b] or approaches built upon it [MMMR22; GSR24], as seen in Fig. 1-3 in the Supplementary Document.

If the triangulation resolution is too coarse, features of the pattern below the resolution of the triangulation cannot be generated. This can be seen in Fig. 10b, where the overall character of the pattern only emerges for circumradii under 0.2, and minor strands only emerge for radii under 0.1. Once the resolution is sufficiently dense,

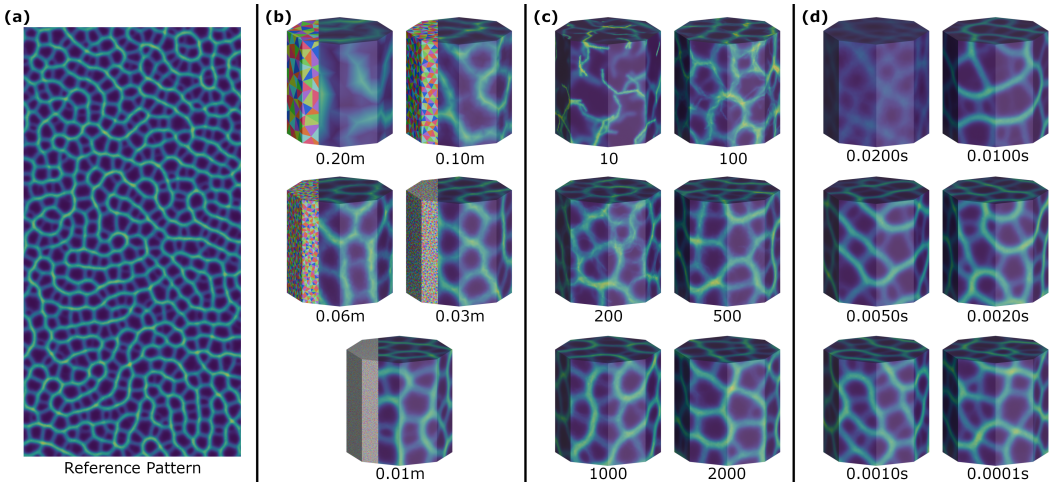

Figure 10: *Analysis of pattern character as simulation discretization is varied. The pattern in panel (a) serves as a reference for simulations performed on an octagonal prism where (b) triangle resolution (c) agent number and (d) time-step are varied. In (b) the maximum circumradius is varied from* 0.20*m to* 0.01*m. A portion of the intrinsic triangulation is shown for each simulation. In (c) the number of agents is varied from* 10 *to* 2000*. To ensure a consistent U metric, the deposit rate is adjusted as described in Sec. 3.2. In (d) the simulation time-step is varied from* 0.02*s to* 0.0001*s. Parameters reported in Table 2 in the Supplementary Document.*

decreasing the triangle size further improves the quality, with negligible effects below 0.03, indicating a convergence to a limit pattern. Evaluating the runtime data for these simulations, the simulation time was found to increase linearly with mean intrinsic triangle density.

If the agent count is too low, pattern quality breaks down, with individual agents diverging into a small number of disorganized paths. This can be seen in the first panel in Fig. 10c. Increasing the agent count smooths and coalesces the agent paths across the entire pattern, again indicating a convergence to a limit pattern (Fig. 10c, the change in pattern character is minimal between 500, 1000, and 2000 agents). For this parameter set, 1000 agents are sufficient for a high quality result. As expected, the time complexity of the simulation was found to be linear to the number of agents.

Compared to triangulation resolution and agent discretization, the relationship between the character of patterns and time-step size is more complex. If the time-step is equal to or exceeds the order of magnitude of the three time parameters (turn decision period, mean particle lifetime, or deposit activation time), then the agent dynamics may be aliased, leading to a pattern that differs substantially from the pattern created by smaller time-steps. This can be seen in the first two examples of Fig. 10d, where the time-steps of 0.02*s* and 0.01*s* equals or exceeds the order of magnitude of the time parameters. Additionally, large time-steps may cause explicit Euler integration of pheromone concentrations to introduce oscillations. Once the time-step is small enough, as seen in the last three simulations, the pattern convergences to a limit pattern and produces high quality results. At large time-step values, the relationship between the time-step and the simulation time is linear. However, this relationship is complex at smaller time steps, as it also depends on the triangulation resolution and the move rate. The relationship between time step and average compute time per step is shown in

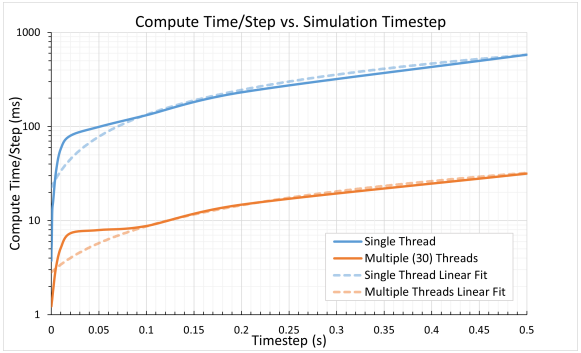

Figure 11: *Log plot of the average computation time per step, in ms, as a function of the time step, in s. Simulations performed using a single thread and 30 threads are plotted in blue and orange, respectively. For each, we fit a linear curve to the data so we can compare our log plotted result to a linear relationship.*

Fig. 11 and is non-linear for small time-steps where the compute time per step sharply rises and then levels off. This indicates the sharp increase is likely due to edge traversals where change in coordinates is required. However, extremely small time steps mean more total steps, resulting in larger simulation times. A balance depending on the pattern for per step time, pattern quality, and total time must be considered depending on the target pattern.

Pattern synthesis on surfaces often introduces distortion and seam artifacts. To test for these artifacts, Fig. 12 illustrates the patterning results on a mesh with multiple topological holes, UV seams, and varying local distortion between the parameterization and the surface. Our simulation produces coherent results across the

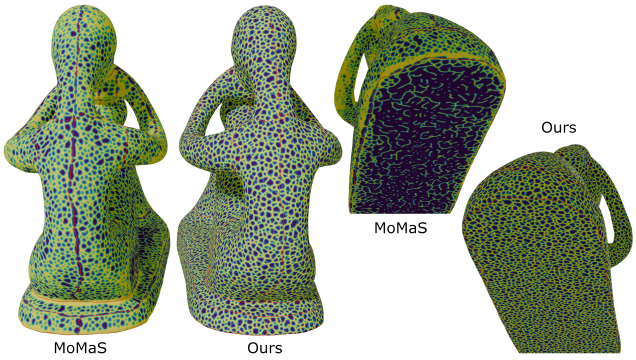

Figure 12: *Comparison of surface pattern synthesis using our approach vs MoMaS on the Fertility mesh [UU09]. In each, the UV seams are visualized using thin red lines.*

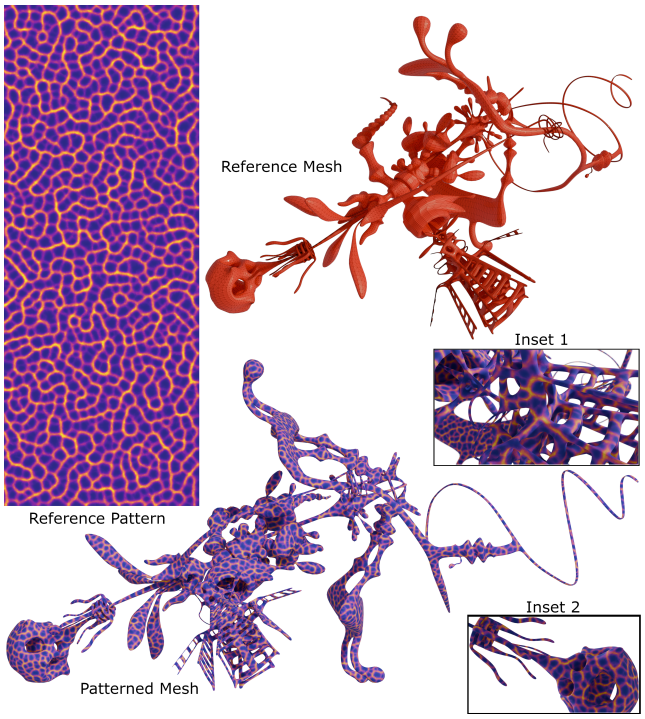

Figure 13: *Stress test results using the Yeah Right [Cra] mesh, seen in the top right. This surface exhibits large discrepancies in element size and the scale of mesh features, as well as having a genus of 131, which is extremely challenging to handle using parameterized or extrinsic projection methods. We synthesized the reference pattern shown in the top left on this mesh (bottom). The two inset images each focus on regions with a large number of topological holes and high variance in mesh scale.*

entire surface, free of seam-related artifacts with consistent pattern characteristics. By contrast, the MoMaS simulation [MMMR22] exhibits visible artifacts around seams and generates patterns with significantly different pattern characteristics. Performance between the two methods is comparable. For the example in Fig. 12, the MoMaS simulation required 300 iterations with 3 million agents (14.7 seconds), while our method converged in 102 iterations with 350 thousand agents (17.6 seconds). Although our run involved fewer agents, the overall computational cost remains in the same range, confirming that our approach is competitive while eliminating artifacts. In general, the results presented in this paper were generated in tens of seconds (Fig. 12) and at most several minutes (Fig. 19; 107*s* at 28% real-time speed with 30 threads). Additionally, many simulations could reduce the simulation time further, usually by a factor of 2–3, by optimizing for convergence with early stopping once the pattern stabilizes.

To further evaluate potential geometric artifacts, we applied our method to a complex model specifically designed to be difficult to parameterize (Fig. 13). This test assessed whether our intrinsic triangulation framework and simulation could remain stable and produce viable patterns. Overall, the framework remained stable and reproduced the expected reference pattern, except in regions where the mesh scale was so fine that the target structure of the pattern could not be fully realized. Even in these cases, the resulting patterns merged coherently with neighbouring regions of the mesh.

6. Results

In this section, we show how our proposed system can be employed to control texture synthesis and generate a broad range of natural patterns. We also illustrate how our slime mold based framework can account for patterns often associated with other simulation based approaches. Finally, we demonstrate that the non slime mold based components of our framework are relatively general, and can be used to improve other surface-based texture synthesis approaches.

In Fig. 14, we use our framework to synthesize the pigmentation patterns of the blue ring octopus, leveraging our colour map parameterization method. We apply two surface maps to modulate parameters, ensuring that rings only form in appropriate regions and that their scale changes as we progress from the head into the tentacles. This produces spot patterns with smooth diffusive falloffs. Using this smooth pheromone falloff, we can reparameterize the colour map so that the blue pigmentation is expressed as a ring around the peak of the spot instead of the centre. Using the exact same parameters and simulation configuration, we generated several distinct patterns with the same character, as a result of the inherent stochasticity of the slime mold simulation (the agent turn decisions and the stochastic chemical kinetics).

The structure of sea fans varies fluidly from branching to network patterns, with a spectrum of intermediate forms observed in different species. Our method is able to capture this aspect of sea fan diversity, as demonstrated in Fig. 15. The branching patterns fill the surface space, but unlike in Fig. 8, at larger sensor angles, cycles appear in the pattern, a common characteristic of specific sea fan species. Additionally, all but the 90° example shows plausible sea fan pattern. For such large turning rates, global connectivity starts to break down, although the individual pieces still produce interesting patterns.

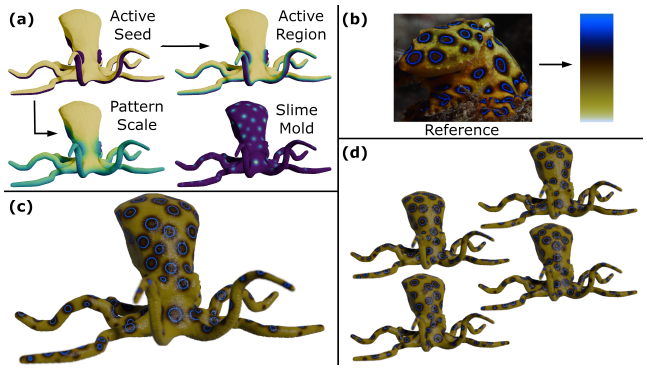

Figure 14: *A blue ring octopus example. (a) Using a seed map, we diffuse it two times, each at different intervals. With a small interval, we generate the active region map that modulates spawn and deposit weights. With a larger interval, we generate the pattern scale map which modulates sensor distance, move rate, diffuse rate, mean particle lifetime, and deposit activation time. The resulting simulation output is then colourized (c) with a reparameterized colour map (b) from the reference image [Zer15] to ensure the blue pigment is expressed as rings. Running multiple simulations using the same parameters produces different patterns with the same character (d). The octopus mesh was adapted from a model provided by Ahmed Sobuj under CC BY 4.0 [Sob24].*

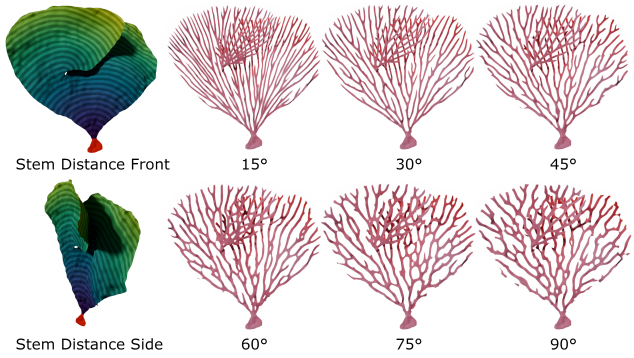

Figure 15: *Simulated sea fan patterns showing a transition from branching to networked patterns. These simulations employ a unidirectional bias along the surface, toward the stem. Bias directions were generated using the gradient of the geodesic distance from the stem, computed as described in [CWW13] (Leftmost panels). This bias is applied with a uniform strength of* $40^\circ/s$*. Agents spawn with random orientations and the stem is filled with pheromones using a write-stimuli. The sensor angle and turn rate are varied by regular increments of* 15° *and indicated below each panel. The model's transparency is modulated by the pattern to visualize a 3D branching structure and colourized by the depth of the fragment to better distinguish overlapping regions.*

Giraffe patterns are a classic example of natural pigmentation, with the reticulated giraffe distinguished by its narrow, net-like coat (Fig. 16). This case demonstrates how simulation outputs can be reused as inputs to guide more complex effects. While such patterns could be produced in a single slime mold simulation by greatly increasing the number of agents and reducing their movement rate, this would be computationally expensive. Instead, we adopt an iterative procedure that leverages spawn-weight and read-stimuli mechanisms to achieve thin-line convergence more efficiently. We begin by generating a thick-lined reticulate pattern with only $10,000$ agents, applying a quad-directional bias along the neck to promote alignment and adjusting the scale on the head, tail, and legs to match observed coat characteristics. This result is thresholded and reused as both a spawn-weight map and a small read-stimuli bias (0.02) to promote thinner lines. In the next iteration, we increase the population to $50,000$ while reducing movement rate and sensor distance to keep costs low, and repeat the process once more to obtain the final reticulate pattern. To colourize the interior of the cells, we compute a distance-transform map from this final result and combine it with the simulation output to produce the render.

Flower petal pigmentation provides an example of natural patterns that benefit from synthesis directly on the surface, with the most successful approach relying on reaction diffusion [ROC*21]. As shown in Fig. 17, our approach can generate comparable results. Two features of our result stand out. First, the boundary of the petal contains partially clipped spots, as also seen in the reference image. Second, the inner part of the petal shows stronger striped pigmentation compared to the reaction–diffusion result.

The ginkgo leaf venation pattern (Fig. 18) provides an interesting challenge for patterning methods, as it combines tightly packed, nearly parallel veins with strict termination at the margin. Such patterns are difficult to capture with methods such as the space colonization algorithm [RFL*05; Run08]. To recreate this branching geometry, we applied a unidirectional gradient away from the leaf base, which specified the agent's spawn orientation direction and modulated mean particle lifetime to prevent overcrowding near the base. Distance fields from the geodesics-in-heat method proved inaccurate along the margin and produced veins which quickly collapsed together; replacing it with fast marching [SSK*05] resolved this issue. A continuous spawn-weight map along the entire boundary generated blurred margin veins, so instead we adopted a stippled boundary map. This choice was inspired by biological observations of leaf development, where discrete sources of a vein inducing signal (auxin) arise at regular intervals along the margin [RTP17]. Finally, while agents normally reflect at boundary edges, here we terminated them to emulate them merging with the margin veins. Compared to space colonization, our approach produces veins that are more compact, exhibiting shallow branching angles and dichotomous branching.

Some patterns require information derived from the extrinsic geometry of the mesh, which is often difficult to incorporate into intrinsic triangulations, particularly when using the integer coordinate data structure [GSC21] that lacks tangent vector correspondence. To enable pattern generation guided by extrinsic cues, we focus on an example incorporating elevation-related information into the intrinsic framework (Fig. 19). Specifically, to emulate a river drainage network, we require a local proxy for elevation. We construct this proxy as a normalized height map, defined by the signed

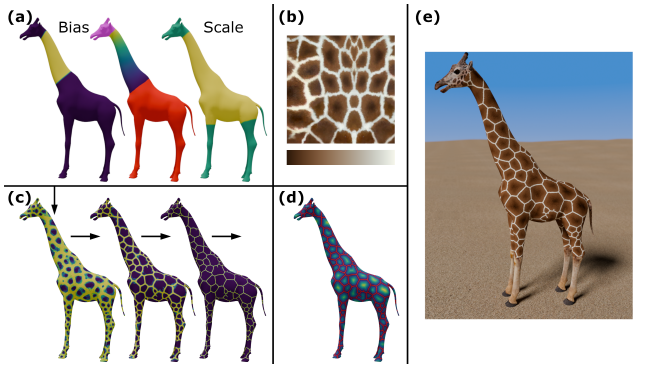

Figure 16: *A reticulated giraffe pigmentation. (a) Inputs to the first slime mold simulation: neck bias weight and gradient (for direction) and pattern scale. (c) Three step slime mold iteration process with progressively thinner lines. The first simulation uses the input maps (a) and the other two use the previous (thresholded) as as spawn weight and read-stimuli while reducing the move rate. (d) We compute the distance transform using the geodesic-in-heat method [CWW13] for proper cell colourization. Extract the colour map from the exemplar image (b) and apply it to the sum of the third iteration result (c) and the distance transform result (d) to produce the final render (e). The head, tail, and lower half of the legs are unpatterned in real giraffes, so we render these areas using the reference texture. Model and texture created by Baryshev Maksim for non-commercial purposes [Bar].*

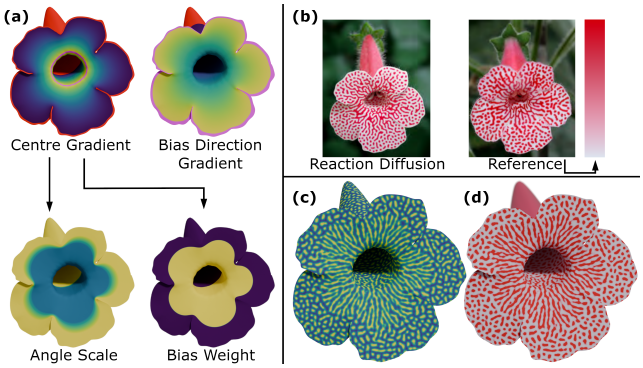

Figure 17: *Kohleria flower pigmentation example. (a) We generate the slime mold maps using two Laplace solves with fixed values prescribed at selected regions (1 in pink, 0 in red). The bias weights and an angle scale map are generated from the centre gradient, and the bias direction is generated from the direction gradient. The resulting slime-mold simulation is shown in (c), and in (d) we have colourized our result using the reference image provided in (b). The images in (b) were adapted, with permission, from Ringham et al. [ROC*21] and show a Koleria flower pattern produced using reaction-diffusion, as well as a comparable image of a real flower.*

distance between the original surface and a volume-preserving smoothed version. This approach captures elevation variation influenced by geometric properties such as mean curvature, allowing extrinsic geometry to meaningfully affect intrinsic pattern design. From the height map we derive slope and flatness maps, which provide directional biases that guide agents to form a drainage network shaped by elevation. Applying a geodesics-in-heat distance transform [CWW13] to the drainage network and using it to modulate height yields terrain-like patterns directly on the surface. This approach illustrates how extrinsic metrics can be systematically incorporated into intrinsic simulations and extended beyond height to other surface-derived quantities.

Finally, we note that the approaches we propose can also improve the pattern synthesis capabilities of non-slime mold based approaches. This is illustrated by the giant puffer fish example in Fig. 20, where we use our framework in conjunction with the Gray–Scott reaction-diffusion simulation [GS84; GS83] to synthesize a texture for the puffer mesh. While the base mesh could not support surface-based pattern simulation, intrinsic Delaunay triangulation produces a plausible texture. Moreover, it improves on the result obtained from extrinsic refinement via linear subdivision, which has minor directionally biased artifacts, slower convergence, or oscillating numerical instabilities around very non-Delaunay or small elements such as the tail and eyes. Our image-assisted colourization is also immediately applicable to reaction-diffusion patterns, allowing us to colourize the mesh using a colour map extracted from the reference image.

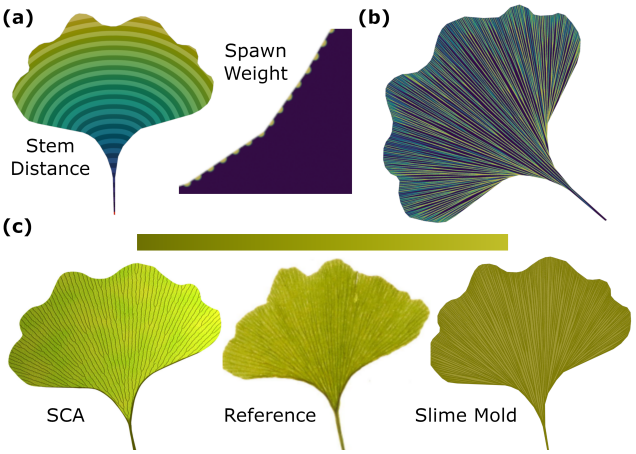

Figure 18: *Ginkgo leaf venation pattern. (a) Distance gradient from the stem computed with fast marching [SSK*05], used both as a unidirectional bias and to modulate particle lifetime. Stippled perimeter weights define agent spawning locations, aligned with the bias (inset of the margin). (b) Slime mold simulation with agents terminating at the margin to emulate the merging of veins. (c) Final render using a colour map extracted from the reference image, compared with results from the space colonization algorithm [RFL*05; Run08] and a reference ginkgo leaf.*

7. Discussion

Our results demonstrate our method's ability to generate a broad and diverse range of natural patterns, including branching, reticulate, spot, stripe, and hybrid structures, directly on curved surfaces. This diversity is achieved through three key advances: a contin-

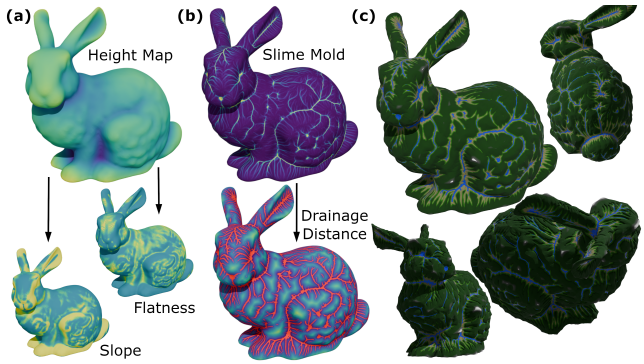

Figure 19: *River-like terrain pattern generated from extrinsic quantities within our intrinsic framework, demonstrated on the Stanford bunny [TL94]. (a) Height map computed as the signed distance between the original mesh and a volume-preserving smoothed version, which is used to calculate local slope and flatness maps. The slope map controls agent move rate, causing steeper regions to produce faster, straighter rivers. The flatness map controls turn rate, causing flatter regions to produce more meandering rivers. (b) Slime mold simulation seeded from the height map with globally uniform bias weight to maintain flow direction. The drainage distance map from geodesics-in-heat [CWW13] is used to modulate height and sharpen elevation. (c) Final renders showing the resulting river-like terrain patterns on the surface.*

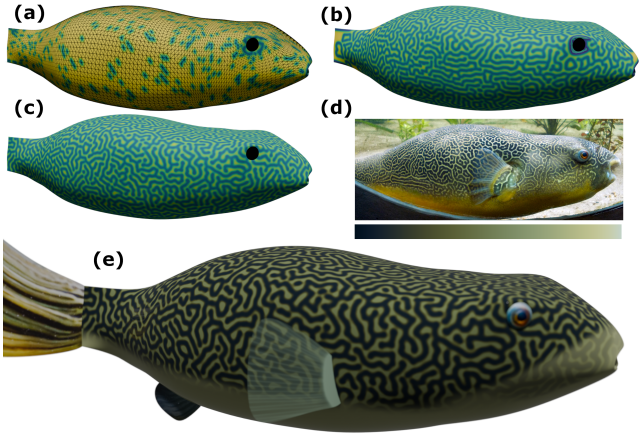

Figure 20: *Gray–Scott reaction-diffusion [GS84; GS83] giant puffer fish example utilizing our intrinsic framework and image assisted colourization. The Gray–Scott model is simulated on three different versions of the mesh: (a) extrinsic mesh, (b) linearly subdivided extrinsic mesh, and (c) intrinsic triangulation. Extracting the colour map from the reference image [Cha12] (d), we colourize the mesh to produce the final result in (e).*

uum reformulation of the slime mold simulation, the use of intrinsic triangulations for robust surface representation, and a stochastic chemical kinetics process that regulates agent turnover and ensures uniform coverage of the simulation domain. Unlike deterministic models, our method's inherent stochasticity produces similar yet visually distinct patterns between runs, even when using the same parameters. These variations, as shown in Fig. 14, maintain the pattern's overall characteristics while providing visually rich alternatives. Together, these elements allow pattern characteristics to remain stable across changes in mesh resolution, agent count, and time-step size.

In contrast to prior approaches, such as the original formulation by Jeff Jones [Jon10a; Jon10b] and the MoMaS surface extension [MMMR22], which rely on image-space or hybrid representations, our framework operates fully in the intrinsic geometry of the surface, enabling artifact-free pattern formation. Compared to MoMaS, our approach avoids artifacts stemming from parameterization and geometric bias by working directly in the intrinsic domain. While MoMaS utilized GPU acceleration to support agent counts in the millions, our framework achieves comparable pattern quality with significantly fewer agents and without GPU-specific optimizations. As a result, our method not only replicates the types of patterns produced by these earlier techniques, but also improves upon them in the context of surface-based procedural texture synthesis.

The use of intrinsic triangulations for surface simulations produces high-quality patterns, even on extremely low-quality or complicated meshes where parameterization, projective, or extrinsic methods are not viable. This applies not only to the slime mold simulation (Fig. 4) but also to other simulation-based approaches like reaction–diffusion (Fig. 20). A mesh is often created with a specific use in mind (e.g., low-poly for games or to support animation), and altering its extrinsic geometry can interfere with the intended use of the mesh. By using intrinsic representations, the simulated pattern can be transferred to a texture map and applied directly to the original extrinsic triangulation, allowing the mesh to be reused in downstream tasks without modification.

A key limitation of prior Physarium slime mold simulations was the lack of convergence across pattern classes. In those methods, output patterns were often selected at a specific time that balanced uniformity and structure, rather than emerging as a stable final state. In contrast, our stochastic chemical kinetics process ensures steady-state convergence by continually renewing agents, preserving uniform coverage without the need for manual tuning of simulation duration. This behaviour is illustrated in Fig. 5, where varying the mean particle lifetime controls the uniformity and global coverage. By eliminating the need to target transient frames, our method removes an important barrier to the use of slime mold simulations in controllable surface texture synthesis.

Our controllability extensions to the slime mold simulation substantially expand the range and complexity of achievable patterns. As demonstrated in the evaluation and results, these mechanisms enable targeted authoring of specific pattern features, including scale variation, structural transitions, directional biases, and more. In many cases, the imposed constraints can be artificial, such as the readable text formed in the signage example (Fig. 7), yet the system respects these constraints while still producing coherent patterns. Notably, the anisotropic effects produced by directional biases unlock additional pattern classes for Physarum slime mold

simulations, such as stripes (Fig. 17) and diverse branching structures (Fig. 8, 15, 18, 19). Notably, pattern anisotropy has also been controlled using anisotropic diffusion [GSR24; ROC*21]to achieve similar visual effects to the directional biasing of agent motion proposed here when $n = 2$ (bidirectional biasing). An aspect of our formulation, which is not supported by anisotropic diffusion, is the ability to specify unidirectional biases, which promote branching behaviour. Similarly, the ability to support n-fold symmetries enables the synthesis of natural patterns with cross-hatched, triangular or hexagonal features (Fig. 5-6 in the Supplementary Document). The ability to produce smooth transitions between network-like and branching structures further highlights the procedural flexibility of our approach.

Manual tuning of colour maps to colourize simulation outputs is difficult and unintuitive. Each example that used our image-assisted colourization approach produced a usable colour map that reduced the tuning space to a 1D parameterization value. We were able to reparameterize each example with relative ease to match the concentration map of the given simulation, resulting in high-quality pigmentation patterns when compared to the reference images.

8. Conclusion

We presented a controllable, intrinsically defined s lime mold framework for procedural pattern synthesis directly on curved surfaces. By reformulating the original model as a continuum process, introducing stochastic chemical kinetics for agent turnover, and operating entirely in the intrinsic geometry of the surface, our method produces diverse natural patterns without parameterization or projection artifacts. These patterns remain consistent across changes in triangulation resolution, agent count, and time-step, while preserving stochastic variation between runs.

The use of intrinsic triangulations enables robust simulation on low-quality or complex meshes without modifying their extrinsic geometry, allowing synthesized patterns to be transferred back to the original surface representation for downstream use. In addition, our controllability mechanisms and directional bias extensions expand the expressive range of slime mold simulations, enabling targeted authoring of directional and structured patterns. Finally, the image-assisted colourization pipeline reduces manual tuning by mapping simulation outputs to exemplar imagery through a one-dimensional parameterization, supporting both scientific visualization and creative applications.

Several directions remain for future exploration. While the current CPU implementation performs competitively to MoMaS due to reduced agent counts, GPU acceleration could further improve performance and enable larger-scale simulations. A particularly promising complementary direction for future work is the use of multi-resolution intrinsic triangulations [LZBJ21; LGC*23], where agent dynamics operate on a coarser mesh to reduce traversal cost, while pheromone diffusion and deposition are maintained on a finer representation. This would require efficient interpolation between intrinsic resolutions.

Further extensions include improving parameter inference and colourization by adapting the exemplar-based parameterization to better match simulation outputs, and exploring automated estimation of simulation parameters from reference images [SS23]. Implementing or utilizing more expressive tools for designing intrinsic vector fields could also enhance control over anisotropic pattern formation [HZ06; FMFG07; RLL*07; SSC19b]. Finally, extending the framework to dynamic or growing surfaces [WFM01; QW12] presents an open challenge, as maintaining valid intrinsic triangulations under extrinsic deformation remains an unresolved problem.

9. Acknowledgements

This work would not be possible without the C++ libraries of Polyscope [Sha*19] and Geometry Central [SC*19]. We also gratefully acknowledge the support of the National Sciences and Engineering Research Council of Canada (Discovery Grants RGPIN-2021-02795 to AR and RGPIN-2024-06195 to FS).

References

[AYD*18] AKL, ADIB, YAACOUB, CHARLES, DONIAS, MARC, et al. "A survey of exemplar-based texture synthesis methods". *Computer Vision and Image Understanding* 172 (2018), 12–24 2, 3.

[Bar] BARYSHEV, MAKSIM. *Giraffe (3D model)*. Done3D. non-commercial use license. URL: https://done3d.com/giraffe/ 13.

[BKP*10] BOTSCH, MARIO, KOBBELT, LEIF, PAULY, MARK, et al. *Polygon Mesh Processing*. A K Peters/CRC Press, 2010 2.

[CdGDS13] CRANE, KEENAN, de GOES, FERNANDO, DESBRUN, MATHIEU, and SCHRÖDER, PETER. "Digital geometry processing with discrete exterior calculus". *ACM SIGGRAPH 2013 Courses*. ACM, 2013 4, 6, 7.

[Cha12] CHAP, CHISWICK. *Elaborate Skin Pattern of Giant Freshwater Puffer Fish, Tetraodon mbu*. Wikimedia Commons. Own work. Licensed under CC BY-SA 3.0. 2012. URL: https://commons.wikimedia.org/wiki/File:Giant_Puffer_fish_skin_pattern.JPG 14.

[CLPQ20] CRANE, KEENAN, LIVESU, MARCO, PUPPO, ENRICO, and QIN, YIPENG. "A survey of algorithms for geodesic paths and distances". *arXiv* (2020). arXiv: 2007.10430 2, 3.

[CPS13] CRANE, KEENAN, PINKALL, ULRICH, and SCHRÖDER, PETER. "Robust fairing via conformal curvature flow". *ACM Trans. Graph.* 32.4 (2013), 1–10 9.

[Cra] CRANE, KEENAN. *Yeah Right*. Released to the Public Domain. Carnegie Mellon University: Keenan Crane Page. URL: https://www.cs.cmu.edu/~kmcrane/Projects/ModelRepository/ (visited on 09/14/2025) 11.

[CWW13] CRANE, KEENAN, WEISCHEDEL, CLARISSE, and WARDETZKY, MAX. "Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow". *ACM Trans. Graph.* 32.5 (2013), 152:1–152:11 7, 12–14.

[DAGP90] DENEUBOURG, J-L, ARON, S, GOSS, S, and PASTEELS, J M. "The self-organizing exploratory pattern of the argentine ant". *J. Insect Behav.* 3.2 (1990), 159–168 3.

[DBT00] DORIGO, MARCO, BONABEAU, ERIC, and THERAULAZ, GUY. "Ant algorithms and stigmergy". *Future Generation Computer Systems* 16.8 (2000), 851–871 3.

[DFW20] DE GOMENSORO MALHEIROS, MARCELO, FENSTERSEIFER, HENRIQUE, and WALTER, MARCELO. "The leopard never changes its spots: realistic pigmentation pattern formation by coupling tissue growth with reaction-diffusion". *ACM Trans. Graph.* 39.4 (2020), 63:1–63:14 1, 3, 9.

[DGA04] DESBENOIT, BRETT, GALIN, ERIC, and AKKOUCHE, SAMIR. "Simulating and modeling lichen growth". *Comput. Graph. Forum* 23.3 (2004), 341–350 2, 3.

[dGDT15] DO GOES, FERNANDO, DESBRUN, MATHIEU, and TONG, YIYING. "Vector field processing on triangle meshes". *SIGGRAPH Asia 2015 Courses*. ACM, 2015 7.

[DSJ19] DAVISON, TIMOTHY, SAMAVATI, FARAMARZ, and JACOB, CHRISTIAN. "Interactive example-palettes for discrete element texture synthesis". *Computers & Graphics* 78 (2019), 23–36 2.

[EMP*02] EBERT, DAVID S., MUSGRAVE, F. KENTON, PEACHEY, DARWYN, et al. *Texturing & Modeling: A Procedural Approach, 3rd Edition*. Morgan Kaufmann Publishers, 2002 2.

[FC24] FENG, NICOLE and CRANE, KEENAN. "A Heat Method for Generalized Signed Distance". *ACM Trans. Graph.* 43.4 (2024), 92:1–92:19 7.

[FMFG07] FISHER, MATTHEW, MÜLLER, PATRICK, FUHRMANN, HELMUT, and GROSS, MARKUS. "Design of Tangent Vector Fields". *ACM Trans. Graph.* 26.3 (2007), 56:1–56:9 15.

[FMP92] FOWLER, DEBORAH R., MEINHARDT, HANS, and PRUSINKIEWICZ, PRZEMYSLAW. "Modeling seashells". *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1992, 379–387 1–3.

[GAD*20] GUEHL, P, ALLÈGRE, R, DISCHLER, J-M, et al. "Semi-procedural textures using point process texture basis functions". *Comput. Graph. Forum* 39.4 (2020), 159–171 2.

[GEB15] GATYS, LEON, ECKER, ALEXANDER S, and BETHGE, MATTHIAS. "Texture Synthesis Using Convolutional Neural Networks". *Advances in Neural Information Processing Systems*. Ed. by CORTES, C., LAWRENCE, N., LEE, D., et al. Vol. 28. Curran Associates, Inc., 2015 2.

[GS83] GRAY, P. and SCOTT, S. K. "Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Isolas and other forms of multistability". *Chemical Engineering Science* 38.1 (1983), 29–43 13, 14.

[GS84] GRAY, P. and SCOTT, S. K. "Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Oscillations and instabilities in the system A + 2B $\rightarrow$ 3B; B $\rightarrow$ C". *Chemical Engineering Science* 39.6 (1984), 1087–1097 13, 14.

[GSC21] GILLESPIE, MARK, SHARP, NICHOLAS, and CRANE, KEENAN. "Integer coordinates for intrinsic geometry processing". *ACM Trans. Graph.* 40.6 (2021), 252:1–252:13 2, 4, 5, 9, 12.

[GSR24] GARNIER, DAVID-HENRI, SCHMIDT, M. P., and ROHMER, DAMIEN. "PhysOM: Physarum polycephalum Oriented Microstructures". *Comput. Graph. Forum* 43.6 (2024), e15075 2, 3, 9, 15.

[HZ06] HERTZMANN, AARON and ZORIN, DENIS. "Guiding Curves for Texture Placement". *ACM Trans. Graph.* 25.3 (2006), 927–932 15.

[Jon10a] JONES, JEFF. "Characteristics of pattern formation and evolution in approximations of Physarum transport networks". *Artif. Life* 16.2 (2010), 127–153 1–4, 7, 9, 14.

[Jon10b] JONES, JEFF. "The emergence and dynamical evolution of complex transport networks from simple low-level behaviours". *Int. Journal of Unconventional Computing* (2010) 2–4, 9, 14.

[KA95] KONDO, SHIGERU and ASAI, RIHITO. "A reaction–diffusion wave on the skin of the marine angelfish Pomacanthus". *Nature* 376.6543 (1995), 765–768 1.

[KCGB11] KENNAWAY, RICHARD, COEN, ENRICO, GREEN, AMELIA, and BANGHAM, ANDREW. "Generation of Diverse Biological Forms through Combinatorial Interactions between Tissue Polarity and Growth". *PLOS Computational Biology* 7.6 (2011), 1–22 7.

[KSSL07] KIM, THEODORE, SEWALL, JASON, SUD, AVNEESH, and LIN, MING C. "Fast simulation of Laplacian growth". *IEEE Comput. Graph. Appl.* 27.2 (2007), 68–76 3.

[LBZ*11] LI, YUANYUAN, BAO, FAN, ZHANG, EUGENE, et al. "Geometry Synthesis on Surfaces Using Field-Guided Shape Grammars". *IEEE Trans. Vis. Comput. Graph.* 17.2 (2011), 231–243 1, 3.

[LGC*23] LIU, HSUEH-TI DEREK, GILLESPIE, MARK, CHISLETT, BENJAMIN, et al. "Surface Simplification using Intrinsic Error Metrics". *ACM Trans. Graph.* 42.4 (2023), 118:1–118:17 15.

[LH05] LEFEBVRE, SYLVAIN and HOPPE, HUGUES. "Parallel controllable texture synthesis". *ACM Trans. Graph.* 24.3 (2005), 777–786 2.

[LH06] LEFEBVRE, SYLVAIN and HOPPE, HUGUES. "Appearance-space texture synthesis". *ACM Trans. Graph.* 25.3 (2006), 541–548 2.

[LM09] LONG, JEREMY and MOULD, DAVID. "Dendritic stylization". *The Visual Computer* 25.3 (2009), 241–253 3.

[LZBJ21] LIU, HSUEH-TI DEREK, ZHANG, JIAYI ERIS, BEN-CHEN, MIRELA, and JACOBSON, ALEC. "Surface Multigrid via Intrinsic Prolongation". *ACM Trans. Graph.* 40.4 (2021), 80:1–80:13 15.

[MMMR22] MAGGIOLI, F., MARIN, R., MELZI, S., and RODOLÀ, E. "MoMaS: Mold Manifold Simulation for real-time procedural texturing". *Comput. Graph. Forum* 41.7 (2022), 519–527 2, 3, 9, 11, 14.

[MMP87] MITCHELL, JOSEPH S. B., MOUNT, DAVID M., and PAPADIMITRIOU, CHRISTOS H. "The Discrete Geodesic Problem". *SIAM Journal on Computing* 16.4 (1987), 647–668 7.

[NPW84] NIEMEYER, L., PIETRONERO, L., and WIESMANN, H. J. "Fractal Dimension of Dielectric Breakdown". *Phys. Rev. Lett.* 52.12 (1984), 1033–1036 3.

[Pea85] PEACHEY, DARWYN R. "Solid texturing of complex surfaces". *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1985, 279–286 2.

[Pea93] PEARSON, JOHN E. "Complex Patterns in a Simple System". *Science* 261.5118 (1993), 189–192 1, 3.

[Per85] PERLIN, KEN. "An image synthesizer". *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1985, 287–296 2.

[PL90] PRUSINKIEWICZ, P. and LINDENMAYER, ARISTID. *The algorithmic beauty of plants*. Springer-Verlag, 1990 1.

[QW12] QUEIROZ, FABIANE and WALTER, MARCELO. "Texture synthesis of contrasting natural patterns". *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*. IEEE, 2012 3, 15.

[RBL*21] ROMBACH, ROBIN, BLATTMANN, ANDREAS, LORENZ, DOMINIK, et al. "High-Resolution Image Synthesis with Latent Diffusion Models". *arXiv* (2021). arXiv: 2112.10752 2.

[RFL*05] RUNIONS, ADAM, FUHRER, MARTIN, LANE, BRENDAN, et al. "Modeling and visualization of leaf venation patterns". *ACM Trans. Graph.* 24.3 (2005), 702–711 3, 12, 13.

[RLL*07] RAY, NICO, LI, WEI, LÉVY, BRUNO, et al. "Rotational Symmetry Field Design on Surfaces". *ACM Trans. Graph.* 26.3 (2007), 10:1–10:7 15.

[RLP07] RUNIONS, ADAM, LANE, BRENDAN, and PRUSINKIEWICZ, PRZEMYSLAW. "Modeling trees with a space colonization algorithm". *Natural Phenomena* (2007), 63–70 3.

[Rob01] ROBERTS, JONATHAN C. "Sticky Pixels: Evolutionary Growth by Random Drop Ballistic Aggregation". *Eurographics UK 2001 Conference Proceedings*. Eurographics Association, 2001, 149–155 3.

[ROC*21] RINGHAM, LEE, OWENS, ANDREW, CIESLAK, MIKOLAJ, et al. "Modeling flower pigmentation patterns". *ACM Trans. Graph.* 40.6 (2021), 233:1–233:14 1–3, 7, 9, 12, 13, 15.

[Ros10] ROSS, SHELDON M. *Introduction to Probability Models*. 10th ed. Elsevier Inc., 2010 2, 6, 8.

[RTP17] RUNIONS, ADAM, TSIANTIS, MILTOS, and PRUSINKIEWICZ, PRZEMYSLAW. "A common developmental program can produce diverse leaf shapes". *New Phytologist* 216.2 (2017), 401–418 12.

[Run08] RUNIONS, ADAM. "Modeling Biological Patterns using the Space Colonization Algorithm". MA thesis. University of Calgary, 2008 12, 13.

[SC*19] SHARP, NICHOLAS, CRANE, KEENAN, et al. *GeometryCentral: A modern C++ library of data structures and algorithms for geometry processing*. 2019. URL: https://geometry-central.net 9, 15.

[SGC21] SHARP, NICHOLAS, GILLESPIE, MARK, and CRANE, KEENAN. "Geometry processing with intrinsic triangulations". *ACM SIGGRAPH 2021 Courses*. ACM, 2021 2–5.

[Sha*19] SHARP, NICHOLAS et al. *Polyscope*. 2019. URL: https://polyscope.run 9, 15.

[Sob24] SOBUJ, AHMED. *Octopus Base Mesh*. Sketchfab. Downloaded under CC Attribution license. 2024. URL: https://sketchfab.com/3d-models/octopus-base-mesh-4b00684b91124694a2b4caa0b7dff44f 12.

[SS23] SCHNÖRR, DAVID and SCHNÖRR, CHRISTOPH. "Learning system parameters from turing patterns". *Machine Learning* 112.9 (2023), 3151–3190 15.

[SSC19a] SHARP, NICHOLAS, SOLIMAN, YOUSUF, and CRANE, KEENAN. "Navigating intrinsic triangulations". *ACM Trans. Graph.* 38.4 (2019), 1–16 2, 4, 5, 9.

[SSC19b] SHARP, NICHOLAS, SOLIMAN, YOUSUF, and CRANE, KEENAN. "The Vector Heat Method". *ACM Trans. Graph.* 38.3 (2019), 24:1–24:19 15.

[SSK*05] SURAZHSKY, VITALY, SURAZHSKY, TATIANA, KIRSANOV, DANIL, et al. "Fast exact and approximate geodesics on meshes". *ACM Trans. Graph.* 24.3 (2005), 553–560 7, 12, 13.

[STBB14] SMELIK, RUBEN M., TUTENEL, TIM, BIDARRA, RAFAEL, and BENES, BEDRICH. "A Survey on Procedural Modeling for Virtual Worlds". *Comput. Graph. Forum* 33.6 (2014), 31–50 1, 3.

[TdSL00] TENENBAUM, J B, de SILVA, V, and LANGFORD, J C. "A global geometric framework for nonlinear dimensionality reduction". *Science* 290.5500 (2000), 2319–2323 2, 9.

[TL94] TURK, GREG and LEVOY, MARC. *The Stanford Bunny*. Stanford 3D Scanning Repository. Computer Graphics Laboratory, Stanford University. 1994. URL: http://graphics.stanford.edu/data/3Dscanrep/ 14.

[Tur01] TURK, GREG. "Texture synthesis on surfaces". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 2001, 347–354 2.

[Tur52] TURING, A M. "The chemical basis of morphogenesis". *Philos. Trans. R. Soc. Lond.* 237.641 (1952), 37–72 1, 3.

[Tur91] TURK, GREG. "Generating textures on arbitrary surfaces using reaction-diffusion". *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. ACM, 1991 2, 3.

[UU09] UU. *Fertility (AIM@SHAPE Shape Repository)*. AIM@SHAPE Shape Repository, Utrecht University. Model provided courtesy of UU by the AIM@SHAPE Shape Repository; license: http://segeval.cs.princeton.edu/public/AIM.txt. 2009. URL: https://sketchfab.com/3d-models/fertility-19823b77d3ed4fa0ab301dddd0c5b2cb 11.

[Vos91] VOSE, M. D. "A linear algorithm for generating random numbers with a given distribution". *IEEE Trans. Softw. Eng.* 17.9 (1991), 972–975 6.

[WFM01] WALTER, MARCELO, FOURNIER, ALAIN, and MENEVAUX, DANIEL. "Integrating shape and pattern in mammalian models". *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001 2, 3, 15.

[WK91] WITKIN, ANDREW and KASS, MICHAEL. "Reaction-diffusion textures". *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1991, 299–308 2, 3.

[WLKT09] WIE, LI-YI, LEFEBVRE, SYLVAIN, KWATRA, VIVEK, and TURK, GREG. "State of the Art in Example-based Texture Synthesis". *Eurographics 2009 - State of the Art Reports*. Ed. by PAULY, M. and GREINER, G. Eurographics Association, 2009 2.

[Wor96] WORLEY, STEVEN. "A cellular texture basis function". *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1996, 291–294 2.

[WS81] WITTEN, T. A. and SANDER, L. M. "Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon". *Phys. Rev. Lett.* 47.19 (1981), 1400–1403 3.

[XRW*19] XIANG, WEI, REN, JIAPING, WANG, KUAN, et al. "Biologically inspired ant colony simulation". *Computer Animation and Virtual Worlds* 30.5 (2019), 1–16 3.

[YSC*17] YUN, JEONGSU, SON, MYUNGBAE, CHOI, BYUNGYOON, et al. "Physically inspired, interactive lightning generation". *Computer Animation and Virtual Worlds* 28.3-4 (2017), e1760 3.

[Zer15] ZERPE, RICKARD. *Greater blue-ringed octopus (Hapalochlaena lunulata)*. Flickr / Wikimedia Commons. Taken December 31, 2014. Licensed under CC Attribution-ShareAlike 2.0 Generic. 2015. URL: https://www.flickr.com/photos/krokodiver/16219454856/ 12.