FlowLenia

March 2025

1 Lenia

Consider a 2D array of points that

$$\mathcal{L} = \{(x, y) \mid x, y \in \mathbb{Z}\} \tag{1}$$

Single Channel Lenia each cell (x, y) holds a single activation in $[0, 1]$. Define the state at a time t as $A^t(x, y)$.
Multi Channel Lenia similar but each cell has multiple $A_i^t(x, y)$ i.e. to introduce colour.
In Lenia, each activation is kept within $[0, 1]$ and can hold a real value within this range. Generally after updates are computed, clipping is done such that the next state is kept within the bounds $[0, 1]$.
In traditional cellular automata, each cell considers only a small discrete set of nearby cells (like a 3×3 grid) when updating. Instead of checking a fixed discrete neighbourhood, Lenia uses a continuous weighting function (kernel) that depends on the distance from the centre cell.

Convolution is defined as (discrete as our grid is discrete):

$$(K * A^t)(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{L}} K(\mathbf{x} - \mathbf{y}) A^t(\mathbf{y}). \tag{2}$$

It basically measured the contribution of every other cell's activation $A^t(\mathbf{y})$ scaled by the Kernel which is a function of the vector separating $\mathbf{x}$ and $\mathbf{y}$.

In practice, Lenia's kernel is a very small value outside a radius of R. This makes the sum much easier to compute as we only sum over the $\mathbf{y}$ values with

$$\|\mathbf{x} - \mathbf{y}\| \leq R. \tag{3}$$

A common Lenia kernel looks like a sum of Gaussians in the radial coordinate r:

$$K(r) = \sum_{j=1}^{k} b_j \exp\left[-\frac{(r - a_j)^2}{2w_j^2}\right], \quad \text{with } r = \frac{\|\mathbf{x}\|}{R}. \tag{4}$$

- a_j shifts the ring
- b_j controls how strong the ring is compared to other rings
- w_j controls how thick the ring is (a bigger w_j means a more gradual peak)

By placing multiple Gaussians in the radial profile, Lenia can create its interesting structures. For instance, it might give a strong positive weight at some radius and weaker at another.

Note: Often $\sum_x K(x)$ is normalised to 1 - turning K into a weighted average.
I am interested to know how we set the parameters for the kernel - worth looking into further.
Okay, now that we can compute all of these kernels for each coordinate in $\mathcal{L}$ at a timestep t, we introduce a growth function G. The intuition behind this is to effectively map the Kernel $[0, 1]$ to a value $[-1, 1]$ to basically see how much the cell's activation should increase or decrease at that location.

A common choice is:

$$G(\alpha) = 2\exp\left(-\frac{(\alpha - \mu)^2}{2\sigma^2}\right) - 1. \tag{5}$$

- $\alpha \in [0, 1]$ is the convolution result $(K * A^t)(x)$
- $\mu \in (0, 1)$ is the centre of the Gaussian
- $\sigma > 0$ controls the width (how narrow or broad the bump is)

If the cell's local average $= \mu$, we see positive growth. If it is too high or low, we see negative growth. Each cell likes to sit at activations μ. This form of local homeostasis is what sllows for self-organisation.

1.1 Putting it all together - Lenia

1. Initialise with 2D grid, growth function G, kernel K and time step.
2. Compute convolution: For each cell x, compute

$$C_t(x) = (K * A_t)(x) = \sum_{y \in L} K(x - y) A_t(y), \tag{6}$$

   but in practice, sum only over y where $\|x - y\| \leq R$.

3. Apply the growth function, For each cell x, compute

$$U_t(x) = G(C_t(x)). \tag{7}$$

4. Update Activations: For each cell x, do

$$A_{t+\Delta t}(x) = \left(A_t(x) + \Delta t \cdot U_t(x)\right)_0^1. \tag{8}$$

5. Repeat from step 2 using the newly computed $A_{t+\Delta t}$.

2 Motivation for Flow Lenia

Despite Lenia's ability to generate complex creatures, we find exploding or vanishing patterns. Most random parameter configurations lead to patterns that either disappear or diffuse to fill the whole grid. Discovering interesting SLPs often requires nontrivial optimisation and a careful existential check that ensures a pattern stays localised and alive. Also in Lenia, the entire 2D lattice is updated with the same kernel-growth rules, preventing different rule sets from coexisting or interacting within the same simulation.

Flow Lenia adds the conservation of mass. If mass cannot be created or annihilated, patterns must remain cohesive and finite in extent (or break into smaller lumps) rather than vanish or spread indefinitely. It also enables a novel way to embed parameters locally, allowing multi-species interactions.

3 Mass Conservation

In Flow Lenia, we continue to store:

$$A^t(\mathbf{x}) \in \mathbb{R}^C_{\geq 0} \tag{9}$$

However, these activations are now treated as concentrations of matter that obey:

$$\sum_{\mathbf{x}\in\mathbb{Z}^2} A^t_c(\mathbf{x}) = \sum_{\mathbf{x}\in\mathbb{Z}^2} A^{t+\Delta t}_c(\mathbf{x}), \quad \text{for each channel } c. \tag{10}$$

Note: This is summed over space, not channels, so the entire space conserves its quantity. We can think of each channel as representing some conserved quantity like energy, mass ect..

Flow Lenia also introduces an affinity map U^t, which is similar to the growth function in Lenia but acts more as an attraction field in how it is applied to the states. It steers the flow of matter rather than directly modifying activations:

$$U^t_j(\mathbf{x}) = \sum_i h_i \left[G_i(K_i * A^t_{c_i,0})\right](\mathbf{x}). \tag{11}$$

We define a velocity field $F^t_i(\mathbf{x})$ for each channel i and cell $\mathbf{x}$:

$$\mathbf{F}^t_i(\mathbf{x}) = (1-\alpha^t(\mathbf{x}))\nabla U^t_i(\mathbf{x}) - \alpha^t(\mathbf{x})\nabla A^t_\Sigma(\mathbf{x}). \tag{12}$$

Here, the affinity gradient steers mass towards attractive areas, while the repulsion term prevents excessive clumping (second term). We determine the mixing by

$$\alpha^t(\mathbf{x}) = \left[\left(\frac{A^t_\Sigma(\mathbf{x})}{\theta_A}\right)^n\right]^1_0. \tag{13}$$

- $\theta_A > 0$ is a chosen parameter
- $n > 1$ (often $n = 2$) makes the function saturate faster: once the local mass $A^t_\Sigma(\mathbf{x})$ exceeds θ_A, $\alpha^t(\mathbf{x})$ quickly approaches 1, forcing strong repulsion
- $[\cdot]^1_0$ clamps the value between 0 and 1.

If $A^t_\Sigma(\mathbf{x}) \ll \theta_A$, then $\alpha^t(\mathbf{x}) \approx 0$. The region is not too dense, so matter follows ∇U^t. If $A^t_\Sigma(\mathbf{x}) \gg \theta_A$, then $\alpha^t(\mathbf{x}) \approx 1$. The region is too dense, so the flow uses the negative gradient of A^t_Σ, pushing matter away.

This ensures matter does not clump infinitely into a single cell; as soon as a cell becomes too dense, the repulsion effect dominates.

4 Transport

To determine the new positions of mass for a given timestep, we compute the destination center:

$$x_i'' = x + \Delta t \mathbf{F}_i^t(\mathbf{x}). \tag{14}$$

Instead of placing all the mass at x_i'', Flow Lenia disperses it into a small region (e.g., a square of sides $2s$) to model diffusion-like behavior:

$$D(z; x_i'', s) = \begin{cases} \frac{1}{(2s)^2}, & \text{if } \|z - x_i''\|_\infty \leq s, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Each destination patch overlaps certain discrete cells in the grid. For a cell p, define $\Omega(p)$ as the unit square domain of that cell. The fraction of mass from x that lands in p is given by:

$$I_i(x, p) = \int_{\Omega(p)} D(z; x_i'', s) dz. \tag{16}$$

Summing over all source cells x, the new activation at p is:

$$A_i^{t+\Delta t}(p) = \sum_x A_i^t(x) I_i(x, p). \tag{17}$$

Since D integrates to 1, total mass is exactly conserved—every bit of mass from each cell x is distributed among the grid cells covered by the diffusion region around x_i''.

S gives a measure of how much diffusion occurs. See figure 3 below!

4.1 Putting it all together - Flow Lenia (not multi-species) - see Figure 1 below!

1. For each cell compute the convolution

2. For each cell x, compute:

$$U_i^t(x) = h_i G_i(S_i^t(x)). \tag{18}$$

   If multiple (K, G) pairs feed channel i, sum them up.

3. Compute the flow field.

$$F_i^t(x) = (1 - \alpha^t(x))\nabla U_i^t(x) - \alpha^t(x)\nabla A_\Sigma^t(x), \tag{19}$$

4. For each cell x, compute the destination center:

$$x_i'' = x + \Delta t \cdot F_i^t(x). \tag{20}$$

Spread the mass $A_i^t(x)$ uniformly in a box of size $2s$ around x_i'':

$$D(z; x_i'', s) = \begin{cases} \frac{1}{(2s)^2}, & \text{if } \|z - x_i''\|_\infty \leq s, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

For each cell p, integrate the portion of that box that intersects $\Omega(p)$. Denote this intersection fraction by $I_i(x, p)$.

5. Sum the contributions to find the new activation at p.

$$A_i^{t+\Delta t}(p) = \sum_{x \in L} A_i^t(x) I_i(x, p). \tag{22}$$

This final sum is the new activation of channel i at cell p.

6. Repeat for next time step

5 Local Parameter Embeddings - Multi-Species

Each cell can carry its own parameter rules e.g. h or kernel parameters, etc. Denote a parameter map by:

$$P^t : \mathcal{L} \to \Theta, \tag{23}$$

An intuitive way to see this is each blob of mass can carry a distinct genetic parameter set. As the blob flows around, it brings its parameters with it. If two distinct blobs collide and merge, you now have coexisting parameters in the same region and we must decide how to reconcile them.

This also opens ideas like competition and symbiosis.

As multiple sources may deposit mass to the same target cell, we define mixing rules for the parameters: **Weighted Sum**

$$P^{t+\Delta t}(\mathbf{p}) = \frac{\sum_{\mathbf{x}} \left[A^t(\mathbf{x}) I(\mathbf{x}, \mathbf{p})\right] P^t(\mathbf{x})}{\sum_{\mathbf{x}} A^t(\mathbf{x}) I(\mathbf{x}, \mathbf{p})}, \tag{24}$$

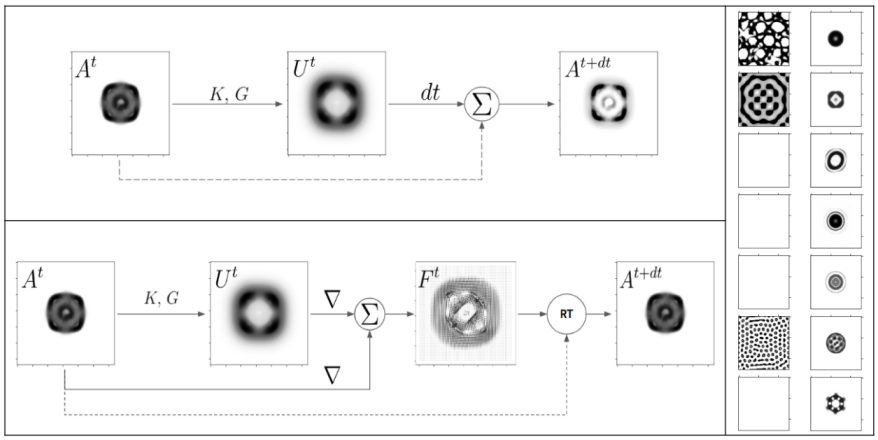

Figure 2: Top left : Lenia update rule. The growth U^t is computed with kernels K and growth functions G (equation 3) defined by a specific parameter configuration sampled in Lenia's parameter space. A small portion of the growth is then added to activations A^t to give the next state A^{t+dt} (equation 4). Bottom left : Flow Lenia update rule. Affinity map U^t is computed as in Lenia. The flow F^t is given by combining the affinity map and concentration (i.e activations) gradients (equation 5). Finally, the next state is obtained by "moving" matter in the CA space according to the flow F^t using reintegration tracking (RT) (equations 6 and 7). Right : Patterns generated from randomly sampled parameters sets (no cherry picking) in Lenia (left) and Flow Lenia (right). Parameters used for both systems are the same (i.e each row corresponds to one parameters set).

Figure 1: Lenia vs. FlowLenia update step

If one species strongly dominates the mass flow into p, that species' parameters become almost exclusively chosen.

Over time, you can see entire regions get colonised by the parameter sets of whichever species is most successful at amassing mass.

I wonder if you can also change these rules based on where in the grid you are—some regions prefer mass weighting while others use softmax:

Softmax

$$\Pr\left[P^{t+\Delta t}(\mathbf{p}) = P^t(\mathbf{x})\right] = \frac{\exp\left(A^t(\mathbf{x})I(\mathbf{x},\mathbf{p})\right)}{\sum_{\mathbf{y}} \exp\left(A^t(\mathbf{y})I(\mathbf{y},\mathbf{p})\right)}. \tag{25}$$

Other possible rules are: Sample each element of θ (like each kernel ring parameter) from one or another source, creating more complex recombinations. Occasionally add random perturbations to the chosen parameter, enabling evolutionary dynamics.

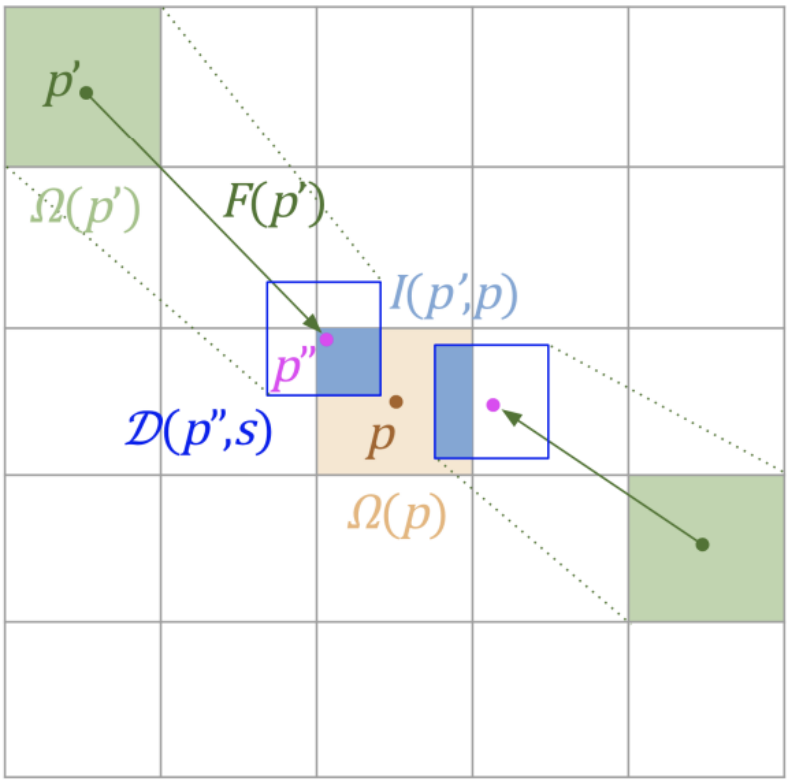

Figure 3: Calculation of incoming matter to cell $p \in \mathcal{L}$ through reintegration tracking (Moroz, 2020). Mass contained in cell at location $p' \in \mathcal{L}$ is moved to a square distribution $\mathcal{D}$ centered on $p'' = p' + dt \cdot F^t(p')$. The proportion of mass from p' arriving in p is then given by the integral of $\mathcal{D}$ on the cell domain of p, $\Omega(p)$, denoted as $I(p', p)$.

Figure 2: Matter flow into new cell and mixing