

Claude's Cat

A three-dimensional iterated function system generated by AI from the prompt “Chat et fougères”, and the question of whether it is a mathematical fractal



Written by **Claude** (Claude Opus 5, Anthropic)
under the supervision of **Dominic Rochon, Ph.D.**, Fundamental Mathematics

Version 2.4.2, 12 October 2026 (build 03eebb9)

Abstract

Claude's Cat is an iterated function system (IFS) of 473 contractive affine maps of $\mathbb{R}^3$ whose attractor has the shape of a cat's face made of Barnsley fern fronds. The system was written by the language model Claude Opus 5 in response to the French prompt "Chat et fougères" (cat and ferns), without the use of any image-generation model. Because the object is defined by equations rather than stored as pixels, it can be examined at any magnification: a zoom by a factor of 1858 into the highlight of one eye returns exactly to the full cat.

We ask whether the object qualifies as a mathematical fractal and address the question by measurement. Two results are central. The work started from a Midjourney image of a fractal shell. Measured by box counting at the same resolution, that image has dimension 1.708, or 1.714 with exactly the box sizes used for the attractor, and the attractor 1.710, so box counting does not distinguish them; the distinction rests on the existence of a generating rule. In addition, the outline of the cat was specified explicitly as parametric curves and does not emerge from the self-similarity. The attractor contains complete copies of itself at arbitrarily small scales. It is therefore a mathematical fractal whose global shape was designed rather than discovered.

1. Introduction

In mathematics a fractal is defined by a construction, not by its appearance. Hutchinson showed that a finite family of contractions of a complete metric space leaves exactly one non-empty compact set invariant [13], and Barnsley used this result to draw natural shapes: his fern is specified by four affine maps [3, 4]. The set is then determined at every scale by those affine maps, including scales at which it has never been computed.

Text-to-image models now produce large numbers of images that resemble fractals. Nothing in such a model guarantees structure below the resolution of the image it outputs: a diffusion model [12, 25] has no generating rule to evaluate at finer scales, only the detail present in its training data. Whether an object has structure below its rendered resolution can be tested directly, by magnifying it.

This document concerns an object produced in a different way. During an extended session, the language model Claude Opus 5 was asked for a fractal based on the prompt "Chat et fougères". Without access to an image model, a GPU or model weights, it constructed the geometry directly: a set of contracting similarities placed along curves describing a cat's face, with Barnsley's four fern maps composed onto every second placement (Figure 1). All 473 maps are listed in Appendix A, so the properties claimed below can be checked independently.

Two results were not anticipated. First, box counting gives nearly the same dimension for the attractor (1.710) and for a Midjourney image (1.708), and the image shows the more regular scaling. Second, the self-similarity of the attractor is resolved over few levels in a single render: two for a map of median ratio and four for the least contracting one, compared with more than twenty for Barnsley's fern. The analysis was revised to take both results into account.

We conclude that ferns, Pollock's drip paintings and AI-generated images share the measurable properties usually associated with fractals without being generated by a

rule, and that mathematical fractals are best regarded as the tools used to model such forms. We therefore propose to distinguish between the two uses of the word "fractal". Section 10 describes an experiment that would extend this work: the matrices were chosen here by a language model reasoning about geometry and inspecting renders, and it remains open what a trained model would find if it chose them by optimisation against a prompt.

Outline. Section 1.1 below separates the claim into five assertions. Sections 2 to 5 establish the existence of the attractor, its self-similarity, its three-dimensionality and its dimension. Section 6 compares the attractor with the Midjourney image, with natural forms and with Pollock's paintings. Sections 7 and 8 describe the role of the AI and the interpretation of the prompt, section 9 the main limitation, section 10 future work, and section 11 the conclusions. A short glossary follows.

1.1 Components of the claim

The statement "this is a mathematical fractal generated by AI from a prompt" combines several assertions of different strength. We examine them separately.

Glossary for the non-specialist

- A **contraction** is a transformation that brings any two points closer together: the distance between the images of two points is always smaller than the distance between the points themselves. For a linear transformation the **largest singular value** is the largest factor by which any distance can be multiplied, so the transformation is a contraction exactly when this value is below 1.
- The **attractor** of a family of contractions is the unique shape that the family maps onto itself: applying all the transformations to it and taking the union of the results gives the same shape.
- The **chaos game** draws the attractor: starting from any point, one repeatedly applies a randomly chosen transformation and plots the result. After a short transient

Assertion	Status	Basis
It is an IFS attractor in the strict sense	Established	Contraction mapping theorem; singular value decomposition of all 473 maps
It is self-similar at fine scales	Established	Deep-zoom test; copies at arbitrarily small scales, 2 to 4 visible per render
Its shape emerges from the self-similarity	Not the case	The outline was specified as parametric curves
It was generated by AI	Yes, in a restricted sense	A language model wrote the geometry; no image model was used
It derives from the prompt	Yes, by interpretation	No text encoder was involved; the prompt was interpreted by the model



Figure 1. *Claude’s Cat*, rendered at 4000×4000 pixels with perspective projection and lighting. The outline is carried by 157 placement maps; the visible strands are the substructure of Barnsley’s fern.

the points lie on the attractor.

- A **decade** is a factor of ten in scale. The number of decades covered by a range of sizes is the base-ten logarithm of the ratio between its largest and smallest value.
- A level of self-similarity is **resolvable** in a given image when the copies of that level are still larger than one pixel. Deeper levels exist in the set but have to be recomputed at a higher magnification to be seen.
- The **box-counting dimension** measures how a shape fills space. The shape is covered with boxes of side ε , and one records how the number of boxes needed grows as ε decreases. Halving ε doubles the count for a line and quadruples it for a filled square; for a fractal the growth rate is intermediate, and the corresponding exponent is the dimension.

2. Existence and uniqueness of the attractor

An IFS is a finite family of contractions of a complete metric space. Here the attractor is the unique non-empty compact set K satisfying

$$K = \bigcup_{i=1}^{473} f_i(K), \quad f_i(x) = A_i x + b_i, \quad x \in \mathbb{R}^3. \quad (1)$$

By Hutchinson’s theorem [13], the map H defined below is a contraction of the space of non-empty compact sets equipped with the Hausdorff metric, which measures the distance between two sets as the amount by which one must be enlarged to contain the other. Its contraction ratio is the largest ratio among the maps:

$$H(S) = \bigcup_i f_i(S), \quad d_H(H(S), H(T)) \leq \left(\max_i \sigma_1(A_i) \right) d_H(S, T). \quad (2)$$

The Banach fixed-point theorem then gives existence and uniqueness of K , provided every map is a contraction. The guarantee is thus theoretical, and the single hypothesis it rests on is a numerical matter: the largest singular value of each of the 473 matrices, obtained by singular value decomposition, must be below 1. It was computed for all of them. Throughout this document $\sigma_1(A) \geq \sigma_2(A) \geq \sigma_3(A) \geq 0$ denote the three singular values of the linear part A , that is, the lengths of the semi-axes of the ellipsoid onto which A maps the unit sphere, ordered from largest to smallest. They give the factors by which A stretches space along three mutually orthogonal directions; σ_1 is therefore the largest factor by which any distance can be multiplied, and their ratios measure how far the map departs from a similarity.

These values refer to the coefficients in double precision, in which the checks were made. Recomputed from published coefficients they hold only to the precision of the publication: from the five decimals of Appendix A, the probabilities sum to 1 within 4×10^{-5} and the similarity identity of the placement maps holds within 3×10^{-6} ;

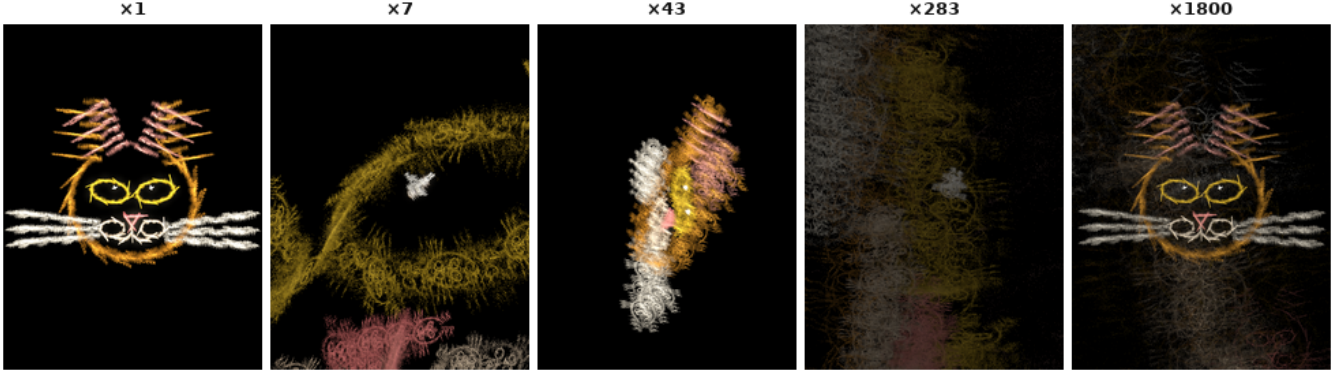


Figure 2. One period of the zoom towards the eye highlight, with magnification indicated. At $\times 1$, the full cat; at $\times 7$, the eye and its highlight; at $\times 43$, a complete miniature cat, mirrored and rotated because the map is an improper similarity; at $\times 1800$, the full cat again. The period is $\times 1858$, as given in section 2: the frame at that magnification is identical to the first, to a mean pixel difference of 0.000. The last panel shown here is the final frame of the 240-frame sequence, one step short of a full period, hence $\times 1800$ rather than $\times 1858$.

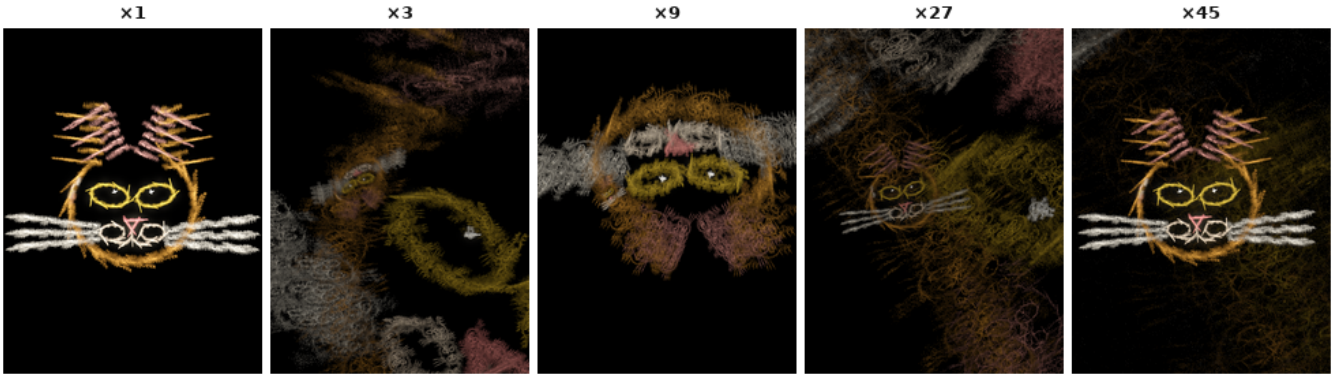


Figure 3. Zoom towards the fixed point of a skull placement map, at $\times 1$, 3, 9, 27 and 45. The period is $\times 47.6$ rather than the $\times 1858$ of Figure 2 because the period is $1/s^2$ and this map contracts by $s = 0.145$, against $s = 0.0232$ for the eye highlight: a map that shrinks less returns to its starting view sooner. The cat reappears in the same way, which shows that the behaviour does not depend on the choice of map.

Check	Result
Largest singular value below 1	473 of 473 maps
Range of σ_1 over all maps	0.00512 to 0.18295
Sum of probabilities	1.0000000000
Placement maps that are exact similarities	157 of 157, to 10^{-14}

from the nine-decimal coefficient file distributed with the sources, within 7×10^{-9} and 4×10^{-10} . In each case the residual is the rounding of the printed values, not a property of the system.

Because the 157 placement maps are similarities, each image $f_i(K)$ is a scaled copy of the whole attractor. All 157 maps have a negative determinant: their orthogonal part is a rotation combined with a reflection, so each copy is also mirrored. A reflection cannot be reached continuously from the identity, so the zoom is built on the composite map $L = f_i \circ f_i$, whose orthogonal part R

is a rotation and whose ratio is s^2 , where s is the ratio of f_i . A zoom towards the common fixed point x^* then returns to its initial view after a magnification of exactly $1/s^2$, and passes through the mirrored copy $f_i(K)$ at $1/s$. The zoom is parameterised by

$$\begin{aligned} g_u(x) &= x^* + s^{-2u} R^{-u}(x - x^*), \\ g_1 &= L^{-1} = f_i^{-2}, \quad g_1 \circ L = \text{id}. \end{aligned} \quad (3)$$

For the eye highlight ($s = 0.0232$, Figure 2) the period is $\times 1858$ and the mirrored copy appears at $\times 43.1$; for the skull map of Figure 3 ($s = 0.145$) the period is $\times 47.6$. Numerically, $g_1 \circ L$ equals the identity to 10^{-12} , and the frames rendered at $u = 0$ and $u = 1$ agree to a relative difference below 10^{-6} , that is, a mean pixel difference of **0.000** on 8-bit images.

3. Self-similarity

A correct formula does not guarantee that a rendering shows structure at depth, so self-similarity was checked

by magnification. Zooming into an ear tuft reveals small copies of the whole cat, including the eyes; at the fixed point of a highlight map, a copy at 2.3 % scale appears inside the white highlight of the eye.

The same test applied to the Midjourney image of a fractal shell gives a different result: at comparable magnification the structure is replaced by featureless blur (Figure 5). The attractor has defined structure at every depth, whereas the diffusion image contains only the detail that was rendered into it.

The residual noise in the deep zooms is Monte Carlo sampling noise, which decreases as the number of chaos-game samples increases. Elton’s ergodic theorem [8] ensures that averages along a chaos-game trajectory converge to integrals with respect to the invariant measure; the topological conditions are refined in [6]. The limit on the left is therefore computational, whereas the limit on the right is a limit on the information in the image.

3.1 At which scales the copies occur

The copies recur in geometric progression rather than at every magnification. Repeated application of one placement map produces copies at the scales s^n , so complete copies exist at arbitrarily small scales, but at a discrete, log-periodic sequence of them. This is why the zoom of Figure 2 closes on itself instead of revealing new structure indefinitely: between two periods one sees parts of the cat, not a smaller complete one.

Composing placement maps gives 157^n exactly similar copies at depth n : **24 600** at depth 2 and **3.9 million** at depth 3. The copies produced by the fern maps are affine rather than similar, and the distortion accumulates: for compositions drawn from the invariant measure, the mean ratio σ_1/σ_2 rises from 1.03 at depth 1 to **2.01** at depth 5 (medians 1.00 and about 1.65). Since the placement maps carry 0.450 of the invariant measure, the fraction of it lying on chains of exact similarities is 45 % at depth 1, 20 % at depth 2 and 9 % at depth 3. The attractor is therefore **self-affine with an exactly self-similar skeleton**, which is the precise form of the statement that the cat is found at smaller and smaller scales.

4. Three-dimensionality

The maps act on $\mathbb{R}^3$, but an IFS of 3×3 matrices may still have a planar attractor. A rotation test shows that this attractor is not planar, and a measurement of its principal axes quantifies how flat it nevertheless is.

4.1 Rotation test

For a planar set, the views at 0° and 180° about a vertical axis are mirror images of each other. On the frames of Figure 4, mirroring the 180° view reduces the mean pixel difference from 15.6 to 11.7, a reduction of about one quarter, so the two views differ substantially. The ratio is used rather than the absolute values because it does not

depend on the exposure of the frames. At 90° and 270° the attractor is seen edge-on and appears as a narrow band with internal structure.

4.2 Flatness

The attractor is three-dimensional but flattened, with the short axis along the viewing direction. The ratio was measured rather than assumed. A sample of 400 000 points was centred and the eigenvalues of its 3×3 covariance matrix computed; the standard deviations along the principal axes are 0.765, 0.654 and 0.142. The third axis, $(-0.001, -0.069, 0.998)$, is almost exactly the depth direction, and the extent of the point set along it is **1.05**, against **4.35** along the first axis. Their ratio, $4.35/1.05 = 4.1$, is the 4:1 quoted here. The ratio of the corresponding standard deviations, $0.765/0.142 = 5.4$, is larger: the figure depends on whether one compares the full extents or the dispersions, and 4:1 is the more conservative of the two.

This flatness follows from the construction: the face was drawn in a plane, lifted onto a dome, and the fronds were given an out-of-plane contraction $s_z = 0.28$, meaning that the component of each frond perpendicular to the plane of the face is multiplied by 0.28 while the two in-plane components keep their own ratios; the fronds are therefore flattened against the dome rather than shrunk uniformly.

Seventy-nine of the 473 maps have rank 2 ($\sigma_3 = 0$ exactly). They are the stem maps of Barnsley’s fern, whose original 2×2 matrix has rank 1, and they map the attractor onto a surface. This is compatible with Hutchinson’s theorem, which requires contraction but not invertibility. These maps carry a probability weight of only 0.0055, but they contribute to the attractor being self-affine rather than self-similar. A set is *self-similar* when its copies are obtained by scaling equally in every direction, so that they have the same shape as the whole; it is *self-affine* when the scaling differs from one direction to another, so that the copies are also stretched.

5. Estimating the fractal dimension

The Hausdorff dimension is not accessible numerically here, since for self-affine sets there is in general no closed-form expression [9, 21]. We therefore estimate the **box-counting dimension**. If $N(\varepsilon)$ boxes of side ε are needed to cover the set, then $N(\varepsilon) \sim \varepsilon^{-D}$, and D is estimated as the slope of $\log N$ against $\log(1/\varepsilon)$ over a range of ε in which this power law holds. The choice of that range is the main source of error; the usual biases, saturation at small scales and too few boxes at large scales, are discussed in [29, 7], and a standard reference is [10].

The estimate below is made on the **planar projection** of the attractor, accumulated as an image. The chaos game was accumulated on a 2000×2000 grid with 360 million samples. A pixel is counted as occupied if its mass exceeds 2×10^{-4} of the maximum; the effect of this

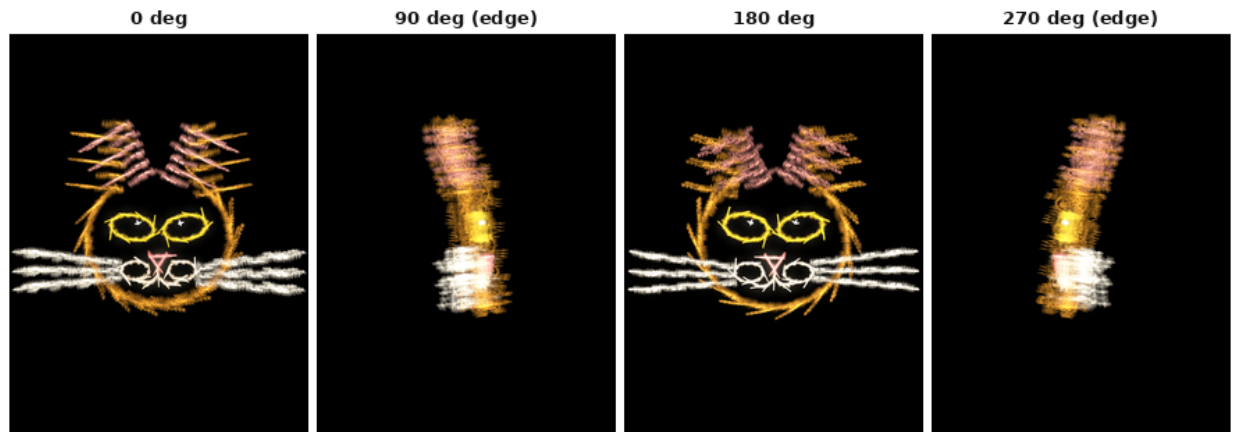


Figure 4. Rotation about the vertical axis. The edge-on views at 90° and 270° show a band of finite thickness, consistent with the thickness of 1.05 for a width of 4.35 measured in section 4.2. Mirroring reduces the difference between the 0° and 180° views only from 15.6 to 11.7.

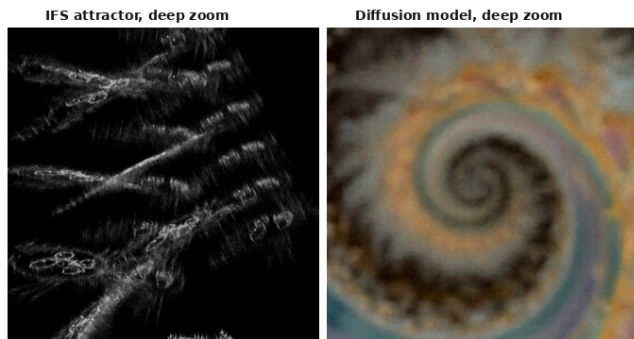


Figure 5. Deep zooms at comparable magnification. Left: an ear tuft of Claude’s Cat, in which fronds and small eyes remain resolved, that is, still separated from one another rather than merged into a single tone. Right: the centre of a Midjourney image of a fractal shell, where the structure has become blur. The left image is limited by the number of samples, the right one by the content of the image.

threshold is examined below. The grid was divided into blocks of $k \times k$ pixels for $k = 1, 2, 4, 5, 8, 10, 16, 20, 25, 40, 50, 80$ and 100 , and a block counted if it contains an occupied pixel:

k (px)	1	2	4	8	16	25	40	50	80	100
boxes N	129 822	35 841	10 269	3085	966	458	209	149	59	46

The fit was restricted to k **between 2 and 40 pixels**. Below 2 pixels the count reflects the sampling density rather than the geometry; above 40 pixels too few boxes remain for a reliable slope. On this range the fit gives $D = 1.710$ with $R^2 = 0.99972$. Here R^2 , the coefficient of determination, is the fraction of the variance of $\log N$ accounted for by the straight line fitted to $\log N$ against $\log(1/k)$: it equals 1 when the points lie exactly on a line and falls below 1 as they scatter. It measures the quality of the fit, not the accuracy of the dimension.

Since the threshold is arbitrary, the estimate was repeated over two decades of threshold values. The slope varies by about ± 0.07 while the quality of the fit is unchanged, so we report $D = 1.71 \pm 0.07$:

threshold ($\times$ maximum)	10^{-3}	5×10^{-4}	2×10^{-4}	10^{-4}	5×10^{-5}
occupied fraction	0.0233	0.0276	0.0325	0.0355	0.0378
D	1.620	1.664	1.710	1.737	1.754
R^2	0.99965	0.99967	0.99972	0.99971	0.99968

6. Fractal qualities without being a mathematical fractal

The starting point of this work was a Midjourney image of a fractal shell (Figure 6), created by Dominic Rochon with Midjourney V8.2. We measured it with the procedure of section 5.

6.1 Box-counting comparison

Thresholding the luminance of the Midjourney image at its 80th percentile and counting boxes of 2 to 32 pixels (powers of two) gives $D = 1.708$ with $R^2 = 0.99987$, compared with 1.710 for the attractor. With the box sizes of section 5 (2 to 40 pixels) the image gives 1.714. Varying the threshold between the 70th and the 90th percentile moves the estimate between 1.53 and 1.79, a spread comparable to that obtained for the attractor.

threshold (percentile)	p70	p75	p80	p85	p90
occupied fraction	0.299	0.246	0.200	0.147	0.096
D	1.787	1.752	1.708	1.641	1.529
R^2	0.99998	0.99994	0.99987	0.99963	0.99940

This result changes the interpretation of section 5. Avnir et al. examined 96 published reports of fractal



Figure 6. Left: the Midjourney image from which this work started, created by Dominic Rochon with Midjourney V8.2. Right: Barnsley’s fern, rendered from its four maps [3]. The fern is a fractal in the strict sense but resembles a real fern less closely than the diffusion image does.

behaviour in natural systems and found that the reported scaling extended over about 1.5 decades on average; they also showed that apparent scaling over one to two decades can arise from randomness alone [2]. The fits used here cover 1.3 decades for the attractor (2 to 40 pixels) and 1.2 decades for the image (2 to 32 pixels), which is less than that typical value; the local slopes of Figure 7 are shown over 2.1 decades.

Stated without logarithms: the range of scales involved in those reports is usually only about a factor of thirty, meaning that the largest feature measured is some thirty times the smallest. Here we could follow both objects over a factor of 128, and over that whole range the slope of the shell is if anything steadier than that of the cat. The measured dimension is therefore **consistent with fractal structure but does not establish it**, for the attractor as for the image. The fractal nature of the attractor follows from its construction and from Hutchinson’s theorem (section 2).

6.2 Why the image is not a fractal in the strict sense

A mathematical fractal is a set defined by a rule, and its structure is therefore determined at every scale. A digital image is a finite array of values and contains no information below the size of its pixels.

The renders of Claude’s Cat are also finite images. However, each is a sample of a set K that can be computed at any resolution. The zoom of Figure 2 evaluates the maps at scales about three orders of magnitude below that of the original render, and the structure it shows is computed from the 473 maps at that scale. No such

computation is possible for the Midjourney image, since there are no maps f_i to evaluate: magnified, it only blurs (Figure 5).

How closely a render approaches K can be quantified. With the 828 million samples of the 4000-pixel accumulation, each of the 473 maps receives at least 57 600 samples and each of the 223 729 pairs of maps at least 4. At depth 3, however, 44.5 % of the $473^3 \approx 1.06 \times 10^8$ pieces receive no sample at all, although together they carry only 1.6 % of the invariant measure. A render is therefore a finite sample of K , weighted by that measure and projected onto a plane; the parts it omits are numerous but of negligible weight.

6.3 Comparison with natural objects

The same limitation applies to natural objects. The self-similarity of a real fern is bounded above by the size of the frond and below by the size of its cells, which leaves two or three levels of branching, in line with the typical range reported by Avnir et al. [2]. No affine map sends the plant exactly onto a part of itself. By the strict definition a fern is therefore not a fractal, and neither is a coastline: Mandelbrot’s original example concerned statistical self-similarity over a finite range of scales [18, 19].

In this respect the Midjourney image belongs to the same category as the fern. It has a useful and reproducible fractal dimension over a limited range, but no defining rule. Conversely, Barnsley’s fern in Figure 6 is exactly a fractal and only approximately a fern, whereas a photograph of a fern is exactly a fern and only approximately a fractal.

6.4 Pollock’s drip paintings

A similar question was raised about a human artefact. In 1999 Taylor, Micolich and Jonas applied box counting to photographs of Jackson Pollock’s drip paintings and reported fractal dimensions increasing over his career, from about 1.45 in 1948 to about 1.72 in 1952 [27]. In 2006 Jones-Smith and Mathur argued that the scaling range is too short to justify the term and that fractal criteria do not reliably identify authentic works; patterns they constructed without fractal structure passed the same tests [16]. Taylor et al. replied in the same issue, maintaining their analysis while acknowledging that the range of magnification is limited [28]. Both sides accepted the measurements; they disagreed about what a limited scaling range allows one to conclude.

A fern, a Pollock painting and the Midjourney image thus all give a stable fractal dimension over one to two decades, and none is generated by a rule that determines its structure below that range. Claude’s Cat differs from them not by a larger or more regular dimension (Figure 7 shows that it has neither) but because its structure at any scale can be computed from its 473 matrices.

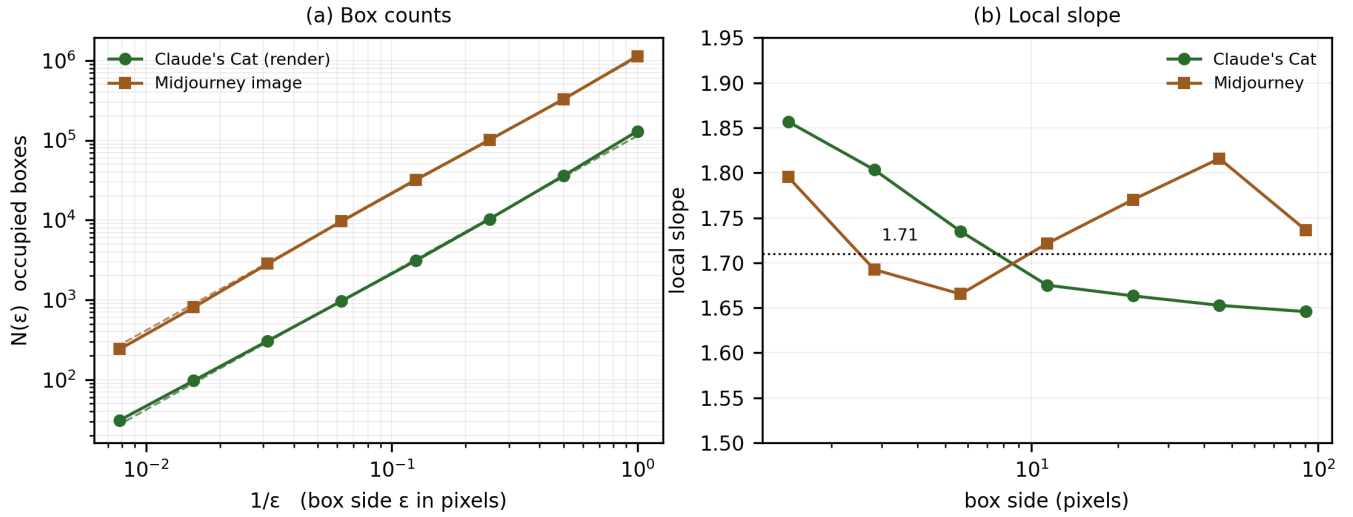


Figure 7. (a) Box counts for the attractor and for the Midjourney image; the two lines are parallel over the measured range. (b) Local slope between successive box sizes. Neither curve decreases at small scales, and the local slope of the diffusion image varies less than that of the attractor.

6.5 Mathematical fractals as approximation tools

The preceding comparisons suggest a positive formulation. The purpose of an IFS is not to be a fern but to approximate one well enough that the approximation can be computed and analysed. In this sense a mathematical fractal is a tool, and the forms it models are its object of study.

This interpretation is consistent with how the theory was developed. Barnsley's collage theorem [3, 4], first stated in [5], is an approximation result. For a compact target set T and an IFS with contraction ratio s , the Hausdorff distance between T and the attractor K is bounded by the distance between T and the union of its images under the maps:

$$d_H(T, K) \leq \frac{1}{1-s} d_H\left(T, \bigcup_i f_i(T)\right). \quad (4)$$

It suffices to choose maps whose images approximately cover the target; the theorem then bounds the error of the attractor. This is how the fronds were placed along the cat's outline (section 8), and it is the principle of fractal image compression [14], in which the maps are fitted block by block to a photograph.

The forms that such tools model can be characterised without reference to a rule, by properties measurable on a finite sample. A form has **fractal qualities** to the extent that it shows:

- **porosity**, with gaps distributed over many scales rather than at a single characteristic size, as quantified by the lacunarity introduced in [19];
- **self-similarity over a range of scales**, in a statistical rather than exact sense;
- a **well-defined non-integer dimension**, that is, a

box-counting exponent that is stable over that range and under changes of threshold (sections 5 and 6);

- **persistent detail**, meaning that the form does not become smooth under magnification within the range available.

The fern, Pollock's paintings and the Midjourney image have all four properties. None of them requires structure at all scales, which is a property of the mathematical model rather than of the modelled form. **Mandelbrot made the same argument in reply to Avnir et al.: the value of fractal geometry lies in what it allows one to compute about irregular forms, and rejecting natural fractality because the scaling range is finite confuses the model with what it models [20].**

Penrose states the general point for physics. He observes that our mathematical models of physical reality are far from complete, yet describe it with a **precision enormously exceeding that of any description that is free of mathematics [22]**. The same holds for fractal models: their value depends on what they allow us to say about the objects they describe, not on whether those objects satisfy the model exactly. No fern is exactly an IFS attractor, just as no planetary orbit is exactly an ellipse.

7. Role of the AI

No trained image-generation model was used. The pipeline that produced the figures contains no diffusion model [12, 25], no GAN, no CLIP guidance [24], no text encoder and no learned weights. These techniques were considered and set aside because the working environment provided neither model weights nor a GPU. The approach that was not taken is score distillation [23] applied to a differentiable parameterisation, as used for vector graphics [15] and CLIP-guided drawing [11]; a

differentiable IFS renderer suitable for this purpose has been published [30].

The language model Claude Opus 5 wrote deterministic geometric and linear algebra code, selected parameters by examining successive renders, and verified the mathematical properties numerically. Its role was that of an author of the construction, not of an image generator.

Component	Origin
Fern coefficients (a, b, c, d, e, f, p)	Barnsley (1988), used unchanged
Face curves, local frames, scale factors by region	Written by Claude Opus 5
Composition of fern maps with placements	Written by Claude Opus 5
About twenty aesthetic parameters	Adjusted by Claude Opus 5 from renders
Camera projection, lighting, colour	Written by Claude Opus 5 (rendering only)
Randomness of the chaos game	Monte Carlo sampling of the invariant measure

8. From prompt to geometry

The prompt “Chat et fougères” was interpreted as a cat drawn with ferns, not as a cat next to ferns. This interpretation was made by the model without a text-embedding model to confirm it. It was implemented as follows.

- 157 contracting similarities are distributed along curves describing the face: the outline of the skull, the ears, the eyes, the nose, the muzzle, the whisker pads and the whiskers.
- At every second position, Barnsley’s four fern maps are composed with the local orthonormal frame of the curve, so that each frond is oriented along the curve or outwards from it.
- The length of the fronds is scaled by region, from 1.00 on the skull to 0.085 on the eye highlights, so that small features are not covered by long fronds.

9. Limitations

The main limitation is that **the shape of the cat does not arise from the fractal structure**. It is given by explicitly drawn curves, and the IFS provides the texture. This can be quantified by the depth of recursion. A copy at level n has a width equal to s^n times that of the whole attractor, where s is the contraction ratio of the map:

$$\frac{\text{width of a level-}n \text{ copy}}{\text{width of the attractor}} = s^n, \quad (5)$$

$$s_{\text{median}} = 0.066, \quad s_{\text{max}} = 0.183.$$

The pixel values below refer to the resolution at which the attractor was accumulated for Figure 1, 1722 pixels across. The 4000-pixel render is upsampled from this

accumulation by a factor of about 1.9 and contains no additional structure.

Level	median map ($s = 0.066$)		least contracting ($s = 0.183$)	
	fraction of width	px	fraction of width	px
1	6.6 %	113	18.3 %	315
2	0.43 %	7.5	3.3 %	58
3	0.03 %	0.49	0.61 %	10.5
4	–	0.03	0.11 %	1.9

In bold, the deepest level whose copies are still wider than one pixel in each column: level 2 for the median map, level 4 for the least contracting one. The figures below them, 0.49 and 0.03 pixels, are the levels that exist in the set but cannot appear in this render.

A level is resolvable when its copies are still wider than one pixel. A map of median ratio therefore gives about **two** resolvable levels and the least contracting map about **four**, compared with more than twenty for Barnsley’s fern, whose main map has ratio 0.85. The deep zooms of section 3 show more levels only because each frame is a new render at a higher effective resolution. Going from 4 maps (the fern) to 98 (an earlier shell) and then to 473 (the cat) **increased the control over the shape and reduced the depth of self-similarity**. With many strongly contracting maps, an IFS tends towards an encoding of a point cloud. Guiding self-similarity towards a prescribed shape is an active research topic in computer graphics [26], and the collage theorem expresses the underlying trade-off [5, 3].

Two further points should be noted. The attractor is self-affine rather than self-similar, since the 316 fern maps inherit the anisotropy of Barnsley’s maps, and no closed-form expression for the Hausdorff dimension of self-affine sets is available in general [9, 21]. The projection, lighting and colour are rendering choices; the attractor K is a subset of $\mathbb{R}^3$ that does not depend on them.

10. Future work: optimisation by a trained model

The two limitations described above are related. The outline had to be specified explicitly (section 9) because nothing in the pipeline could evaluate whether an attractor resembles a cat (section 7). A trained model can provide such an evaluation, so that the shape could be obtained by optimisation instead of being specified.

The required components exist. A diffusion model can compare an image with a text prompt and return a gradient of that comparison with respect to the image [23]; this has been used to optimise vector graphics [15]. An IFS renderer can also be made differentiable [30]. Combining the two gives the following procedure, in which **the optimisation variables are the IFS coefficients and the diffusion model only ever receives rendered images**:

1. **Render.** The current coefficients are rendered into an image by the chaos game.
2. **Evaluate.** The diffusion model compares this image with the prompt and returns the gradient of its score with respect to the pixel values, that is, the direction in which the image should change to match the prompt better.
3. **Back-propagate.** Since the renderer is differentiable, the chain rule transfers this gradient from the pixels to the coefficients.
4. **Update.** The coefficients are updated in that direction, and the procedure is repeated.

The diffusion model is thus **never given the matrices and never modifies them directly**; it evaluates images, and the matrices change as a consequence of those evaluations. Two further ingredients are needed in practice. Contraction should be built into the parameterisation, so that every candidate is a valid IFS. And the task can be amortised by pre-training on randomly generated IFS rendered and labelled automatically, as in fractal pre-training datasets [17, 1], so that a prompt yields matrices in a single step.

Appendix B contains a script implementing this procedure. It has not been run, because the working environment had no GPU and no access to model weights. Its renderer is the one verified numerically for this document, and the script checks the contraction condition on its output before saving it.

The main question is whether the trade-off measured in section 9, between control of the shape and depth of self-similarity, is intrinsic or a consequence of the explicit construction. An optimiser free to choose the contraction ratios might find systems with fewer maps, ratios closer to 1 and deeper self-similarity that still resemble a cat. The collage theorem, which expresses this trade-off (section 9), suggests that the improvement will be modest; a negative result would also be informative.

11. Conclusion

Claude’s Cat is a mathematical fractal. Its 473 maps are verified contractions, so Hutchinson’s theorem gives the existence and uniqueness of the attractor. The 157 placement maps are exact similarities, which allows a zoom that returns exactly to its starting view after a magnification of 1858 and shows complete copies of the cat at intermediate scales.

The measured fractal dimension does not establish this conclusion. At the same resolution, the attractor gives 1.710 and a Midjourney image 1.708, or 1.714 with exactly the same box sizes, and the image shows the more regular scaling. Over the 1.2 to 1.3 decades used for the fits, which is less than is typical of published claims and short enough for randomness to produce similar results [2], box counting does not separate a

mathematical fractal from a form with fractal qualities. The discussion of Pollock’s paintings in *Nature* [27, 16, 28] concerned the same question and was not settled by further measurement.

The difference between the attractor and the other objects considered here is the existence of a generating rule. Ferns, Pollock’s paintings and diffusion images have a stable dimension over a limited range and no information beyond it; the structure of Claude’s Cat at any scale can be computed from Appendix A.

The object was generated by AI in a restricted sense. A language model wrote the geometry, chose parameters by inspecting renders and verified the properties numerically. No image model was used, Barnsley’s coefficients were taken unchanged from the literature, and the prompt was interpreted without a text encoder.

Finally, the shape was designed rather than obtained from the self-similarity. The fractal structure provides the texture, and only two to four levels of recursion are resolved, compared with more than twenty for Barnsley’s fern. Section 10 describes how an optimiser might reduce this limitation.

11.1 Generation by a trained model

Section 7 noted that no trained model was involved here. This was a practical constraint, not a limitation in principle: **a trained model can generate a mathematical fractal in the strict sense.**

The reason is that the optimised quantity is not an image. As described in section 10, the variables are the coefficients of the maps, the diffusion model evaluates rendered images, and its gradient is transferred to the coefficients through the differentiable renderer [23, 15, 30]. Two properties ensure that the result is a mathematical fractal whatever the outcome of the optimisation.

- **Contraction can be imposed by the parameterisation.** If each matrix is written as a rotation, times a diagonal matrix of singular values bounded below 1, times a second rotation, every point of the parameter space is a valid IFS, and Hutchinson’s theorem [13] applies throughout the optimisation.
- **The output is the rule.** The result saved is the list of coefficients, which can be evaluated at any scale after the training images have been discarded. This is the property identified in section 6.2 as distinguishing a mathematical fractal from an image.

The script of Appendix B imposes contraction in this way and checks, before writing its output, that the largest singular value is below 1. If the check succeeds, the saved object is an attractor in the strict sense. The script was not executed, for the reasons given above.

Such a procedure would satisfy the assertions of section 1.1 without restriction: the maps would be chosen by a trained model, from a prompt processed by a text encoder, the result would be a mathematical fractal by

construction, and the shape would be obtained by optimisation. Only the last of these is not achieved by Claude’s Cat. An AI-generated mathematical fractal in the full sense is therefore possible; in this case the obstacle was the absence of a GPU and of model weights.

11.2 Terminology

The discussion above uses the word “fractal” in two different senses. We propose to distinguish them explicitly.

A form is **fractal** if it shows fractal qualities over the range of scales available: porosity over several scales, statistical self-similarity, a stable non-integer dimension, and detail that persists under magnification. This is a property of the form, measurable on a finite sample, and it does not require a generating rule. In this sense ferns, coastlines, Pollock’s paintings and the Midjourney image are fractal.

A **mathematical fractal** is a set defined by a rule, so that its structure is determined at every scale. The rule may take various forms, and the result which guarantees that the set exists varies with it: for an iterated function system it is a fixed-point theorem, whereas the Julia and Mandelbrot sets of complex dynamics are defined by the behaviour of an iteration, and their hypercomplex analogues in the same way. What the notion requires is the rule, not one particular theorem. A mathematical fractal is not a stronger version of the first notion but an object of a different kind, used to model, measure and generate forms of the first kind; for iterated function systems the collage theorem provides the corresponding error bound. Claude’s Cat is a mathematical fractal; a fern is not.

With this distinction the question “is this a fractal?” becomes two separate questions: whether the form has fractal qualities, and which ones; and whether there is a rule that determines its structure at all scales. The Pollock debate [27, 16, 28] can then be read as two sides answering different questions with the same word, and an image produced by a diffusion model is classified with natural forms, which fractal geometry was developed to describe, rather than with the mathematical models themselves.

The experiment proposed in section 10 would relate the two notions directly. Fitting a mathematical fractal to a photograph of a fern would measure how well a set with structure at all scales can approximate a form that has structure over only about two decades, and at which scales the approximation begins to contain structure that the form does not have.

A second terminological question arises from the way this object was produced. It concerns provenance rather than the object itself, and it is recorded here as a suggestion for public communication, not as a claim about established usage. In public discussion “AI-generated” is normally understood to mean an image synthesised by a trained generative model. That is not what happened

here: the maps were reasoned out, and what the process delivered is a rule that anyone can recompute. The supervising author proposes to mark this difference by speaking of **SI**, for *synthetic intelligence*, when a system produces the specification of an object rather than a rendering of it. The term is not in current use, and the rest of this document does not rely on it; it is offered because the distinction it marks is the same one that separates Claude’s Cat from the diffusion image of Figure 6.

Acknowledgements

This work benefited from discussions with the fractal community on social media. The question addressed here, whether an object that looks fractal is a fractal, arose in those discussions and was not the initial aim of the work.

Several points of the argument came from these exchanges: the distinction between a rendered image and the set it samples, the observation that a measured dimension is not sufficient evidence of fractal structure, and the fact that natural forms do not satisfy the strict definition. The terminology proposed in section 11 is an attempt to make explicit a distinction that is already made in practice in that community.

We are particularly grateful to those who offered counter-examples. Section 6 is a direct result of such a counter-example: the measurement expected to separate the attractor from a diffusion image gave 1.710 and 1.708, and the local-slope test did not separate them either.

Any remaining errors are the authors’ own.

A note on the name

In fractal geometry an object is usually named after the person who specified it: the Sierpiński triangle, Lévy’s C curve, Barnsley’s fern. By this convention the name used here is justified. The 157 placement maps that give the attractor the shape of a cat were written by Claude Opus 5, as were the composition scheme, the scale factors by region and about twenty parameters chosen from successive renders.

Two qualifications apply. First, Barnsley’s four fern maps produce every visible strand of the object. They account for 316 of the 473 maps and were taken unchanged from *Fractals Everywhere* [3]; the direction and supervision of the work also came from elsewhere. “A cat drawn with Barnsley’s ferns” is therefore a more accurate description than an independent construction.

Second, the convention assumes that the name refers to a person who can be asked about the construction. “Claude” refers to a model of which many instances run concurrently, and the instance that produced these matrices cannot be consulted afterwards. The name therefore indicates which system produced the specification, not authorship in the usual sense. For this reason the docu-

ment refers to “Claude Opus 5” rather than to “Claude” alone.

Declaration on the use of generative AI and AI-assisted technologies

This work was carried out by Dominic Rochon in collaboration with Claude Opus 5 (Anthropic), a large language model used as a reasoning and programming tool under the author’s direction. In accordance with publishers’ policies the model is not listed as an author, since it cannot approve a manuscript, confirm that the work is its own, or answer for its content after publication.

Scope of use. The model derived and wrote the iterated function system described here, including the 157 placement maps, the composition scheme and the scale factors, and chose about twenty free parameters by examining successive renders. It wrote the code for rendering, measurement and figures, carried out the numerical checks reported in sections 2, 4, 5 and 6, and drafted the text, including the argument of section 6 and the terminology of section 11. The four fern maps are Barnsley’s, used unchanged [3].

Human responsibility. The author defined the problem, provided the prompt, directed each stage, reviewed all results and requested the verification that corrected two values reported in earlier drafts. The author takes full responsibility for the content, the reported measurements and the conclusions.

Figures. No generative image model was used to produce any figure. Figures 1, 2, 3 and 4 are deterministic renders computed from the coefficients in Appendix A, and Figure 7 plots the measurements of sections 5 and 6; all can be reproduced from them. An image created by the author with Midjourney V8.2 appears, as an object of study, in Figure 6 and in the right half of Figure 5; its dimension is measured in section 6.

Reproducibility. Appendix A lists the 473 coefficient sets with their singular values, so that every numerical result in this document can be checked independently of the code used to produce it, to the precision of the printed coefficients (section 2).

Appendices

The appendices are separate documents, typeset with margins suited to their content and bound after this one.

Appendix A lists the 473 maps of Claude’s Cat: the nine entries of A , the translation b , the probability p and the largest singular value. Part I contains the 157 placement maps that define the shape of the cat, and Part II the 316 Barnsley fern maps that provide the texture. The coefficients are those used for the renders, after conjugation by the global reflection. For Part I the `sigma1` column equals the similarity ratio, and every

value in both parts is below 1, which is the hypothesis of Hutchinson’s theorem [13].

Appendix B contains the script described in sections 10 and 11: a differentiable IFS renderer driven by score distillation from a fixed diffusion model. It is a specification that has not been executed. It has its own list of references so that it can be read separately.

Citation numbers. The bibliography of this edition is sorted alphabetically. When referring to it from other documents, cite by author and year.

References

- [1] Connor Anderson and Ryan Farrell. Improving fractal pre-training. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022.
- [2] David Avnir, Ofer Biham, Daniel Lidar, and Ofer Malcai. Is the geometry of nature fractal? *Science*, 279(5347):39–40, 1998.
- [3] Michael F. Barnsley. *Fractals Everywhere*. Academic Press, 1988.
- [4] Michael F. Barnsley and Stephen Demko. Iterated function systems and the global construction of fractals. *Proceedings of the Royal Society of London A*, 399(1817):243–275, 1985.
- [5] Michael F. Barnsley, Vincent Ervin, Douglas Hardin, and John Lancaster. Solution of an inverse problem for fractals and other sets. *Proceedings of the National Academy of Sciences USA*, 83:1975–1977, 1986.
- [6] Michael F. Barnsley and Andrew Vince. The chaos game on a general iterated function system. *Ergodic Theory and Dynamical Systems*, 31(4):1073–1079, 2011.
- [7] George Datseris, Inga Kottlarz, Anton P. Braun, and Ulrich Parlitz. Estimating fractal dimensions: A comparative review and open source implementations. *Chaos*, 33(10):102101, 2023.
- [8] John H. Elton. An ergodic theorem for iterated maps. *Ergodic Theory and Dynamical Systems*, 7(4):481–488, 1987.
- [9] Kenneth J. Falconer. The Hausdorff dimension of self-affine fractals. *Mathematical Proceedings of the Cambridge Philosophical Society*, 103(2):339–350, 1988.
- [10] Kenneth J. Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, 2nd edition, 2003.
- [11] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. CLIP-Draw: Exploring text-to-drawing synthesis through language-image encoders. In *Advances in Neural Information Processing Systems*, 2022.
- [12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems* 33, 2020.
- [13] John E. Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.
- [14] Arnaud E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, 1992.

- [15] Ajay Jain, Amber Xie, and Pieter Abbeel. VectorFusion: Text-to-SVG by abstracting pixel-based diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [16] Katherine Jones-Smith and Harsh Mathur. Fractal analysis: revisiting Pollock’s drip paintings. *Nature*, 444:E9–E10, 2006.
- [17] Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, et al. Pre-training without natural images. In *Asian Conference on Computer Vision (ACCV)*, 2020.
- [18] Benoit B. Mandelbrot. How long is the coast of Britain? statistical self-similarity and fractional dimension. *Science*, 156(3775):636–638, 1967.
- [19] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman, 1982.
- [20] Benoit B. Mandelbrot. Is nature fractal? *Science*, 279(5352):783, 1998.
- [21] Curtis T. McMullen. The hausdorff dimension of general Sierpiński carpets. *Nagoya Mathematical Journal*, 96:1–9, 1984.
- [22] Roger Penrose. What is reality? *New Scientist*, November 2006. issue of 18 November 2006.
- [23] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3d using 2d diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.
- [24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.
- [25] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [26] Alexa Schor and Theodore Kim. Into the portal: Directable fractal self-similarity. In *ACM SIGGRAPH 2024 Conference Papers*, 2024.
- [27] Richard P. Taylor, Adam P. Micolich, and David Jonas. Fractal analysis of Pollock’s drip paintings. *Nature*, 399:422, 1999.
- [28] Richard P. Taylor, Adam P. Micolich, and David Jonas. Fractal analysis: revisiting Pollock’s drip paintings (reply). *Nature*, 444:E10–E11, 2006.
- [29] James Theiler. Estimating fractal dimension. *Journal of the Optical Society of America A*, 7(6):1055–1073, 1990.
- [30] Cheng-Hao Tu, Hong-You Chen, David Carlyn, and Wei-Lun Chao. Learning fractals by gradient descent. In *AAAI Conference on Artificial Intelligence*, 2023.

Appendix A — Claude's Cat: Explicit 3×3 Matrices

All 473 contractive affine maps of $\mathbb{R}^3$, from the prompt “Chat et fougères”

Written by Claude (Claude Opus 5, Anthropic) under the supervision of Dominic Rochon, Ph.D., Fundamental Mathematics

Version 2.4.2 — 12 October 2026

Each map is $f(x) = A x + b$, drawn with probability p ; the attractor is $K = \bigcup_i f_i(K)$. Matrices are given to 5 decimals. Column **pos** is the index of the face-curve position the map belongs to.

The reflected frame in which these coefficients are written

The coefficients below are those of the system **after the conjugation** $A \leftarrow R A R$, $b \leftarrow R b$, with $R = \text{diag}(1, -1, 1)$. The matrix R is the reflection that changes the sign of the second coordinate; it is orthogonal and is its own inverse, $R^2 = I$. Conjugating every map by it is a change of coordinates, not a change of system: if K is the attractor of the original maps, the attractor of the conjugated maps is exactly its mirror image $R(K)$.

The reason is practical. The renders in the main document are produced in image coordinates, where the vertical axis runs downwards, which mirrors the figure with respect to the frame in which the maps were first written. Publishing the conjugated coefficients means that **the list below generates the cat as it is shown**, with no further convention to apply.

Nothing structural is affected. Because R is orthogonal, conjugation leaves the singular values of every map unchanged, hence the contraction ratios and the similarity property; it leaves the determinants unchanged as well, since $\det(R A R) = \det(R)^2 \det(A) = \det(A)$. The probabilities are untouched. Only the geometry is mirrored.

What $A^T A = s^2 I$ verifies, and why it matters

A linear map A is a **similarity of ratio s** when it multiplies every length by the same factor s , that is $|A x| = s |x|$ for every vector x . Writing this condition as a product of the vector with itself gives $x^T (A^T A) x = s^2 x^T x$ for every x , which holds precisely when **$A^T A = s^2 I$** . The identity is therefore not one property among others: it is the definition of a similarity, written in a form a computer can test entry by entry. Equivalently, the three singular values of A are all equal to s , so the unit sphere is mapped to a sphere of radius s rather than to an ellipsoid.

This is what distinguishes the two parts of the system. All 157 Part I maps satisfy it, with every entry of $A^T A - s^2 I$ below 10^{-14} in absolute value — the rounding level of double-precision arithmetic, so the identity holds as exactly as a computer can state it. That tolerance refers to the coefficients in double precision: recomputed from the five-decimal values printed below, the identity holds within about 3×10^{-6} , which is the size of the rounding and not a property of the maps. Those maps shrink every direction alike and carry no distortion. The Part II fern maps do not satisfy it: they flatten space in one direction, which is what gives a frond its shape, and it is they that make the attractor self-affine rather than self-similar. The column **sigma1** is the largest singular value; for Part I maps it equals the similarity ratio $s = 0.145 \cdot \mu_{p1}$ exactly, so the table verifies itself.

The two parts are not interchangeable

Part I holds the maps that carry the cat concept: each one is placed on a drawn face curve and its frame determines where and how the structure sits. Remove them and the cat is gone. Part II holds the Barnsley fern maps composed onto every second position; they supply self-similar texture only. Remove them and the cat survives as a bare outline. Note $a_{33} \approx 0$ throughout Part I: the frame's third axis $e_3 = e_1 \times e_2$ is near-horizontal by construction, which does not reduce the rank. In Part II, the 79 stem maps are rank 2 ($\sigma_3 = 0$ exactly), which is legal for an IFS: only contractivity is required.

Part I — Cat concept: 157 placement maps

A = 0.145 · mu_pl · (Fj Rz(tau_j)), b = qj. Probability 0.45/157 = 0.002866 for every map.

Skull oval — 28 maps

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	signal
0	0	+0.00910	+0.11045	-0.09350	-0.00768	-0.09319	-0.11082	+0.14451	-0.01191	+0.00000	-0.71610	-0.69479	+0.24980	0.002866	0.14500
1	1	+0.03078	+0.08723	-0.11166	-0.03716	-0.10530	-0.09250	+0.13674	-0.04825	-0.00000	-0.84020	-0.56976	+0.24299	0.002866	0.14500
2	2	+0.03374	+0.06185	-0.12673	-0.06069	-0.11125	-0.07046	+0.12729	-0.06944	-0.00000	-0.93801	-0.42691	+0.23678	0.002866	0.14500
3	3	+0.02235	+0.03936	-0.13776	-0.06803	-0.11978	-0.04526	+0.12608	-0.07161	-0.00000	-1.00647	-0.27070	+0.23199	0.002866	0.14500
4	4	-0.00074	+0.01785	-0.14390	+0.00593	-0.14377	-0.01786	+0.14488	+0.00597	-0.00000	-1.04344	-0.10602	+0.22924	0.002866	0.14500
5	5	+0.00116	-0.01039	-0.14462	+0.01600	-0.14374	+0.01045	+0.14411	+0.01604	+0.00000	-1.04776	+0.06198	+0.22891	0.002866	0.14500
6	6	-0.00395	-0.03802	-0.13987	-0.01444	-0.13912	+0.03823	+0.14422	-0.01497	+0.00000	-1.01930	+0.22804	+0.23104	0.002866	0.14500
7	7	-0.00112	-0.06410	-0.13006	-0.00227	-0.13004	+0.06411	+0.14498	-0.00253	-0.00000	-0.95895	+0.38696	+0.23536	0.002866	0.14500
8	8	-0.01280	-0.08611	-0.11596	-0.01705	-0.11470	+0.08705	+0.14342	-0.02132	-0.00000	-0.86859	+0.53378	+0.24127	0.002866	0.14500
9	9	+0.01342	-0.10554	+0.09852	+0.01243	-0.09774	+0.10639	+0.14384	+0.01829	-0.00000	-0.75105	+0.66389	+0.24800	0.002866	0.14500
10	10	+0.00524	-0.12169	-0.07866	+0.00338	-0.07859	+0.12181	+0.14487	+0.00623	-0.00000	-0.61001	+0.77323	+0.25472	0.002866	0.14500
11	11	-0.06692	-0.11523	-0.05717	-0.02871	-0.04944	+0.13325	+0.12539	-0.07282	+0.00000	-0.44989	+0.85838	+0.26059	0.002866	0.14500
12	12	+0.01014	-0.14043	-0.03466	-0.00250	-0.03457	+0.14080	+0.14462	+0.01044	-0.00000	-0.27569	+0.91667	+0.26493	0.002866	0.14500
13	13	-0.06987	-0.12652	-0.01161	-0.00561	-0.01016	+0.14453	+0.12693	-0.07010	-0.00000	-0.09287	+0.94628	+0.26723	0.002866	0.14500
14	14	-0.00251	-0.14451	+0.01161	+0.00020	+0.01161	+0.14453	+0.14498	-0.00251	+0.00000	+0.09287	+0.94628	+0.26723	0.002866	0.14500
15	15	-0.05549	-0.12940	+0.03466	-0.01366	-0.03186	+0.14080	+0.13326	+0.05715	+0.00000	+0.27569	+0.91667	+0.26493	0.002866	0.14500
16	16	+0.01014	-0.13287	+0.05717	-0.00435	+0.05701	+0.13325	+0.14458	+0.01103	-0.00000	+0.44989	+0.85838	+0.26059	0.002866	0.14500
17	17	-0.01809	-0.12046	+0.07866	+0.01168	+0.07779	+0.12181	+0.14339	-0.02154	+0.00000	+0.61001	+0.77323	+0.25472	0.002866	0.14500
18	18	-0.03329	-0.10104	+0.09852	+0.03083	-0.09357	+0.10639	+0.13772	-0.04538	-0.00000	+0.75105	+0.66389	+0.24800	0.002866	0.14500
19	19	-0.02003	-0.08472	+0.11596	+0.02669	+0.11285	+0.08705	+0.14111	-0.03337	-0.00000	+0.86859	+0.53378	+0.24127	0.002866	0.14500
20	20	-0.03120	-0.05601	+0.13006	+0.06329	+0.11362	+0.06411	+0.12667	-0.07057	+0.00000	+0.95895	+0.38696	+0.23536	0.002866	0.14500
21	21	-0.01632	-0.03457	-0.13987	+0.05971	+0.12648	+0.03823	+0.13112	-0.06190	-0.00000	+1.01930	+0.22804	+0.23104	0.002866	0.14500
22	22	-0.00061	-0.01043	+0.14462	+0.00848	+0.14437	+0.01045	+0.14475	-0.00850	-0.00000	+1.04776	+0.06198	+0.22891	0.002866	0.14500
23	23	+0.00134	+0.01781	+0.14390	+0.01079	+0.14349	-0.01786	+0.14459	-0.01087	+0.00000	+1.04344	-0.10602	+0.22924	0.002866	0.14500
24	24	+0.00440	+0.04505	-0.13776	+0.01339	+0.13710	-0.04526	+0.14431	-0.01409	-0.00000	-1.00647	-0.27070	+0.23199	0.002866	0.14500
25	25	+0.01808	+0.06810	+0.12673	+0.03252	+0.12249	-0.07046	+0.14014	-0.03721	-0.00000	+0.93801	-0.42691	+0.23678	0.002866	0.14500
26	26	-0.01565	+0.09117	+0.11166	-0.01889	+0.11005	-0.09250	+0.14291	+0.02452	-0.00000	+0.84020	-0.56976	+0.24299	0.002866	0.14500
27	27	-0.01749	+0.10944	-0.09350	-0.01476	+0.09233	-0.11082	+0.14318	+0.02289	+0.00000	+0.71610	-0.69479	+0.24980	0.002866	0.14500

Ear, outer edge — 12 maps

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	signal
28	28	+0.02309	+0.14167	+0.02051	+0.00330	+0.02024	-0.14354	+0.14311	-0.02333	-0.00000	-0.69160	-0.73880	+0.12212	0.002866	0.14500
29	29	+0.02656	+0.14106	+0.02051	+0.00379	+0.02015	-0.14354	+0.14250	-0.02683	-0.00000	-0.66528	-0.92304	+0.09430	0.002866	0.14500
30	30	+0.02280	+0.14172	+0.02051	+0.00326	+0.02025	-0.14354	+0.14316	-0.02304	-0.00000	-0.63896	-1.10728	+0.05298	0.002866	0.14500
31	31	+0.05180	+0.13387	+0.02051	+0.00740	+0.01912	-0.14354	+0.13523	-0.05233	-0.00000	-0.61264	-1.29152	+0.00000	0.002866	0.14500
32	32	+0.07465	+0.12261	+0.02051	+0.01066	+0.01752	-0.14354	+0.12385	-0.07540	-0.00000	-0.58632	-1.47576	+0.00000	0.002866	0.14500
33	33	+0.01952	+0.14221	+0.02051	+0.00279	+0.02032	-0.14354	+0.14365	-0.01972	-0.00000	-0.56000	-1.66000	+0.00000	0.002866	0.14500
40	40	-0.03431	-0.13938	+0.02051	+0.00490	+0.01991	+0.14354	+0.14080	-0.03466	+0.00000	+0.69160	-0.73880	+0.12212	0.002866	0.14500
41	41	-0.05803	-0.13129	+0.02051	+0.00829	+0.01876	+0.14354	+0.13262	-0.05861	+0.00000	+0.66528	-0.92304	+0.09430	0.002866	0.14500
42	42	-0.03209	-0.13991	+0.02051	+0.00458	+0.01999	+0.14354	+0.14133	-0.03241	+0.00000	+0.63896	-1.10728	+0.05298	0.002866	0.14500
43	43	-0.05655	-0.13194	+0.02051	+0.00808	+0.01885	+0.14354	+0.13327	-0.05712	+0.00000	+0.61264	-1.29152	+0.00000	0.002866	0.14500
44	44	-0.05155	-0.13397	+0.02051	+0.00736	+0.01914	+0.14354	+0.13533	-0.05207	+0.00000	+0.58632	-1.47576	+0.00000	0.002866	0.14500
45	45	-0.00262	-0.14352	+0.02051	+0.00037	+0.02050	+0.14354	+0.14498	-0.00265	+0.00000	+0.56000	-1.66000	+0.00000	0.002866	0.14500

Ear, inner edge — 12 maps

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	signal
34	34	+0.03506	+0.12486	-0.06485	-0.01753	-0.06243	-0.12969	+0.13960	-0.03919	+0.00000	-0.53840	-1.61680	+0.00000	0.002866	0.14500
35	35	+0.05339	+0.11819	-0.06485	-0.02670	-0.05910	-0.12969	+0.13214	-0.05969	+0.00000	-0.47072	-1.48144	+0.00000	0.002866	0.14500
36	36	+0.03850	+0.12385	-0.06485	-0.01925	-0.06192	-0.12969	+0.13846	-0.04304	+0.00000	-0.40304	-1.34608	+0.00000	0.002866	0.14500
37	37	+0.00244	+0.12967	-0.06485	-0.00122	-0.06483	-0.12969	+0.14497	-0.00273	+0.00000	-0.33536	-1.21072	+0.06058	0.002866	0.14500
38	38	-0.00160	+0.12968	-0.06485	+0.00080	-0.06484	-0.12969	+0.14499	+0.00179	+0.00000	-0.26768	-1.07536	+0.10144	0.002866	0.14500
39	39	+0.04943	+0.11990	-0.06485	-0.02471	-0.05995	-0.12969	+0.13406	-0.05526	+0.00000	-0.20000	-0.94000	+0.13177	0.002866	0.14500
46	46	-0.06017	-0.11489	-0.06485	-0.03009	-0.05744	+0.12969	+0.12845	-0.06727	-0.00000	+0.53840	-1.61680	+0.00000	0.002866	0.14500
47	47	-0.02643	-0.12697	-0.06485	-0.01321	-0.06349	+0.12969	+0.14196	-0.02955	-0.00000	+0.47072	-1.48144	+0.00000	0.002866	0.14500
48	48	-0.07136	-0.10829	-0.06485	-0.03568	-0.05415	+0.12969	+0.12108	-0.07978	-0.00000	+0.40304	-1.34608	+0.00000	0.002866	0.14500
49	49	-0.00760	-0.12947	-0.06485	-0.00380	-0.06473	+0.12969	+0.14475	-0.00849	-0.00000	+0.33536	-1.21072	+0.06058	0.002866	0.14500
50	50	-0.01172	-0.12916	-0.06485	-0.00586	-0.06458	+0.12969	+0.14441	-0.01310	-0.00000	+0.26768	-1.07536	+0.10144	0.002866	0.14500
51	51	-0.05919	-0.11540	-0.06485	-0.02959	-0.05770	+0.12969	+0.12902	-0.06617	-0.00000	+0.20000	-0.94000	+0.13177	0.002866	0.14500

Eye almond — 32 maps

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	sigma1
76	76	-0.00025	-0.01584	+0.07816	+0.00121	+0.07815	+0.01584	+0.07974	-0.00124	+0.00000	-0.10598	-0.12040	+0.37441	0.002866	0.07975
77	77	-0.00840	+0.03332	+0.07197	-0.01759	+0.06978	-0.03436	+0.07733	+0.01949	+0.00000	-0.11582	-0.18682	+0.37125	0.002866	0.07975
78	78	-0.01296	+0.05945	+0.05155	-0.01098	+0.05037	-0.06085	+0.07792	-0.01698	+0.00000	-0.16892	-0.25220	+0.36507	0.002866	0.07975
79	79	+0.00925	+0.07210	+0.03281	+0.00418	-0.03254	-0.07269	+0.07910	-0.01015	-0.00000	-0.25720	-0.30659	+0.35540	0.002866	0.07975
80	80	-0.00630	+0.07791	+0.01584	-0.00128	+0.01579	-0.07816	+0.07949	+0.00643	+0.00000	-0.36722	-0.34171	+0.34222	0.002866	0.07975
81	81	+0.00808	+0.07932	-0.00191	-0.00019	-0.00190	-0.07973	+0.07934	-0.00808	+0.00000	-0.48223	-0.35221	+0.32681	0.002866	0.07975
82	82	+0.00602	+0.07588	-0.02378	-0.00188	-0.02371	-0.07612	+0.07950	-0.00631	+0.00000	-0.58472	-0.33649	+0.31206	0.002866	0.07975
83	83	-0.00598	+0.05938	-0.05290	+0.00530	-0.05264	-0.05968	+0.07935	+0.00799	-0.00000	-0.65909	-0.29695	+0.30179	0.002866	0.07975
84	84	+0.00251	+0.01564	-0.07816	-0.01241	-0.07717	-0.01584	+0.07874	-0.01266	-0.00000	-0.69402	-0.23960	+0.29926	0.002866	0.07975
85	85	+0.00221	-0.03429	-0.07197	+0.00464	-0.07182	+0.03436	+0.07958	+0.00514	+0.00000	-0.68418	-0.17318	+0.30554	0.002866	0.07975
86	86	+0.00293	-0.06078	-0.05155	+0.00248	-0.05149	+0.06085	+0.07966	+0.00384	+0.00000	-0.63108	-0.10780	+0.31887	0.002866	0.07975
87	87	+0.01464	-0.07120	-0.03281	+0.00661	-0.03214	+0.07269	+0.07812	-0.01606	+0.00000	-0.54280	-0.05341	+0.33553	0.002866	0.07975
88	88	-0.02821	-0.07289	-0.01584	-0.00572	-0.01478	+0.07816	+0.07438	-0.02878	+0.00000	-0.43278	-0.01829	+0.35146	0.002866	0.07975
89	89	+0.00483	-0.07958	+0.00191	-0.00012	+0.00191	+0.07973	+0.07960	+0.00483	+0.00000	-0.31777	-0.00779	+0.36381	0.002866	0.07975
90	90	+0.01509	-0.07461	+0.02378	-0.00471	-0.02331	+0.07612	+0.07817	+0.01581	+0.00000	-0.21528	-0.02351	+0.37144	0.002866	0.07975
91	91	+0.01348	-0.05813	+0.05290	-0.01195	+0.05154	+0.05968	+0.07769	+0.01802	+0.00000	-0.14091	-0.06305	+0.37468	0.002866	0.07975
94	94	-0.00372	+0.01540	+0.07816	-0.01836	+0.07597	-0.01584	+0.07752	-0.01874	-0.00000	+0.69402	-0.23960	+0.29926	0.002866	0.07975
95	95	-0.01305	+0.05823	+0.05290	-0.01156	+0.05162	-0.05968	+0.07782	+0.01743	+0.00000	+0.65909	-0.29695	+0.30179	0.002866	0.07975
96	96	+0.02578	+0.07162	+0.02378	+0.00806	+0.02238	-0.07612	+0.07504	-0.02701	+0.00000	+0.58472	-0.33649	+0.31206	0.002866	0.07975
97	97	-0.01827	+0.07761	+0.00191	-0.00044	+0.00186	-0.07973	+0.07763	-0.01827	+0.00000	+0.48223	-0.35221	+0.32681	0.002866	0.07975
98	98	-0.01348	+0.07699	-0.01584	+0.00273	-0.01561	-0.07816	+0.07856	-0.01375	-0.00000	+0.36722	-0.34171	+0.34222	0.002866	0.07975
99	99	-0.00913	+0.07211	-0.03281	+0.00412	-0.03255	-0.07269	+0.07912	+0.01002	+0.00000	+0.25720	-0.30659	+0.35540	0.002866	0.07975
100	100	+0.00690	+0.06046	-0.05155	-0.00585	-0.05122	-0.06085	+0.07924	-0.00905	+0.00000	+0.16892	-0.25220	+0.36507	0.002866	0.07975

101	101	+0.00973	+0.03296	-0.07197	-0.02037	-0.06902	-0.03436	+0.07649	-0.02258	+0.00000	+0.11582	-0.18682	+0.37125	0.002866	0.07975
102	102	+0.00177	-0.01575	-0.07816	+0.00871	-0.07767	+0.01584	+0.07925	+0.00889	+0.00000	+0.10598	-0.12040	+0.37441	0.002866	0.07975
103	103	+0.01167	-0.05852	-0.05290	+0.01034	-0.05188	+0.05968	+0.07821	+0.01559	+0.00000	+0.14091	-0.06305	+0.37468	0.002866	0.07975
104	104	-0.01982	-0.07350	-0.02378	-0.00619	-0.02296	+0.07612	+0.07700	-0.02076	+0.00000	+0.21528	-0.02351	+0.37144	0.002866	0.07975
105	105	-0.00577	-0.07952	-0.00191	-0.00014	-0.00191	+0.07973	+0.07954	-0.00577	+0.00000	+0.31777	-0.00779	+0.36381	0.002866	0.07975
106	106	-0.01089	-0.07740	+0.01584	+0.00221	+0.01569	+0.07816	+0.07897	-0.01111	+0.00000	+0.43278	-0.01829	+0.35146	0.002866	0.07975
107	107	+0.01459	-0.07121	+0.03281	-0.00658	-0.03214	+0.07269	+0.07813	+0.01600	+0.00000	+0.54280	-0.05341	+0.33553	0.002866	0.07975
108	108	-0.02488	-0.05553	+0.05155	+0.02108	+0.04704	+0.06085	+0.07278	-0.03261	-0.00000	+0.63108	-0.10780	+0.31887	0.002866	0.07975
109	109	+0.00215	-0.03430	+0.07197	-0.00451	+0.07182	+0.03436	+0.07959	+0.00500	+0.00000	+0.68418	-0.17318	+0.30554	0.002866	0.07975

Catchlight — 4 maps

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	sigma1
92	92	-0.01441	-0.01692	+0.00667	+0.00432	+0.00507	+0.02222	+0.01766	-0.01504	-0.00000	-0.32500	-0.24000	+0.39440	0.002866	0.02320
93	93	+0.00074	+0.00663	+0.02222	+0.00247	+0.02208	-0.00667	+0.02306	-0.00258	-0.00000	-0.32500	-0.24000	+0.39440	0.002866	0.02320
110	110	-0.00627	-0.02132	+0.00667	+0.00188	+0.00640	+0.02222	+0.02226	-0.00654	-0.00000	+0.32500	-0.24000	+0.39440	0.002866	0.02320
111	111	+0.00427	+0.00512	+0.02222	+0.01425	+0.01705	-0.00667	+0.01780	-0.01488	-0.00000	+0.32500	-0.24000	+0.39440	0.002866	0.02320

Noise triangle — 15 maps

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	sigma1
112	112	+0.00028	-0.06090	+0.00000	+0.00000	+0.00000	+0.06090	+0.06090	+0.00028	+0.00000	-0.09600	+0.26000	+0.40804	0.002866	0.06090
113	113	-0.02854	-0.05380	+0.00000	+0.00000	+0.00000	+0.06090	+0.05380	-0.02854	+0.00000	-0.04800	+0.26000	+0.40912	0.002866	0.06090
114	114	+0.00193	-0.06087	+0.00000	+0.00000	+0.00000	+0.06090	+0.06087	-0.00193	+0.00000	+0.00000	+0.26000	+0.40948	0.002866	0.06090
115	115	-0.00850	-0.06030	+0.00000	+0.00000	+0.00000	+0.06090	+0.06030	-0.00850	+0.00000	+0.04800	+0.26000	+0.40912	0.002866	0.06090
116	116	+0.00919	-0.06020	+0.00000	+0.00000	+0.00000	+0.06090	+0.06020	-0.00919	+0.00000	+0.09600	+0.26000	+0.40804	0.002866	0.06090
117	117	-0.01477	-0.03038	-0.05067	-0.02216	-0.04557	+0.03378	+0.05477	-0.02663	+0.00000	-0.10800	+0.27800	+0.40614	0.002866	0.06090
118	118	+0.00304	-0.03364	-0.05067	+0.00456	-0.05047	+0.03378	+0.06065	-0.00548	+0.00000	-0.08400	+0.31400	+0.40351	0.002866	0.06090
119	119	-0.01476	-0.03039	-0.05067	-0.02214	-0.04558	+0.03378	+0.05478	-0.02661	+0.00000	-0.06000	+0.35000	+0.40029	0.002866	0.06090
120	120	-0.00862	-0.03266	-0.05067	-0.01292	-0.04900	+0.03378	+0.05889	-0.01553	+0.00000	-0.03600	+0.38600	+0.39646	0.002866	0.06090
121	121	-0.00663	-0.03312	-0.05067	-0.00995	-0.04969	+0.03378	+0.05971	-0.01196	+0.00000	-0.01200	+0.42200	+0.39202	0.002866	0.06090
122	122	-0.00596	+0.03325	-0.05067	+0.00894	-0.04988	-0.03378	+0.05994	-0.01074	+0.00000	+0.10800	-0.27800	+0.40614	0.002866	0.06090
123	123	+0.00355	+0.03359	-0.05067	-0.00533	-0.05039	-0.03378	+0.06056	-0.00640	+0.00000	+0.08400	+0.31400	+0.40351	0.002866	0.06090
124	124	+0.01055	+0.03209	-0.05067	-0.01583	-0.04814	-0.03378	+0.05785	-0.01902	+0.00000	+0.06000	+0.35000	+0.40029	0.002866	0.06090
125	125	+0.01094	+0.03196	-0.05067	-0.01641	-0.04794	-0.03378	+0.05762	-0.01972	+0.00000	+0.03600	+0.38600	+0.39646	0.002866	0.06090
126	126	-0.00414	+0.03353	-0.05067	+0.00621	-0.05029	-0.03378	+0.06044	+0.00746	+0.00000	+0.01200	+0.42200	+0.39202	0.002866	0.06090

Muzzle — 30 maps

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	sigma1
127	127	+0.00000	+0.00000	-0.06090	-0.02577	-0.05518	+0.00000	+0.05518	-0.02577	+0.00000	+0.00000	+0.45000	+0.34933	0.002866	0.06090
128	128	+0.00000	+0.00000	-0.06090	-0.02961	-0.05322	+0.00000	+0.05322	-0.02961	+0.00000	+0.00000	+0.53000	+0.33795	0.002866	0.06090
129	129	+0.01090	+0.03562	-0.04818	-0.01410	-0.04607	-0.03725	+0.05823	-0.01783	-0.00000	-0.03135	+0.58815	+0.29178	0.002866	0.06090
130	130	+0.00441	+0.04959	-0.03507	-0.00310	-0.03493	-0.04979	+0.06066	-0.00539	+0.00000	-0.08038	+0.63633	+0.28324	0.002866	0.06090
131	131	+0.00658	+0.05903	-0.01347	-0.00149	-0.01339	-0.05939	+0.06053	-0.00675	+0.00000	-0.13484	+0.66104	+0.27739	0.002866	0.06090
132	132	-0.00408	+0.05992	+0.01009	-0.00068	+0.01007	-0.06006	+0.06076	+0.00413	+0.00000	-0.19471	+0.66226	+0.27451	0.002866	0.06090
133	133	+0.00579	+0.05416	+0.02724	+0.00290	+0.02708	-0.05447	+0.06055	-0.00648	+0.00000	-0.26000	+0.64000	+0.27442	0.002866	0.06090
134	134	-0.01090	-0.03562	-0.04818	-0.01410	-0.04607	+0.03725	+0.05823	-0.01782	+0.00000	+0.03135	+0.58815	+0.29178	0.002866	0.06090
135	135	-0.01037	-0.04870	-0.03507	-0.00731	-0.03430	+0.04979	+0.05956	-0.01269	+0.00000	+0.08038	-0.63633	+0.28324	0.002866	0.06090
136	136	+0.00438	-0.05923	-0.01347	+0.00099	-0.01343	+0.05939	+0.06073	+0.00449	-0.00000	+0.13484	+0.66104	+0.27739	0.002866	0.06090
137	137	-0.00362	-0.05995	+0.01009	+0.00061	+0.01008	-0.06006	+0.06079	-0.00367	+0.00000	+0.19471	+0.66226	+0.27451	0.002866	0.06090
138	138	+0.00935	+0.05366	+0.02724	-0.00467	+0.02683	+0.05447	+0.06000	+0.01045	+0.00000	+0.26000	+0.64000	+0.27442	0.002866	0.06090
139	139	+0.00000	+0.00000	+0.06090	+0.01275	+0.05955	+0.00000	+0.05955	-0.01275	+0.00000	-0.17000	+0.48000	+0.34110	0.002866	0.06090
140	140	+0.00401	+0.04849	+0.03662	+0.00302	+0.03650	-0.04866	+0.06069	-0.00502	+0.00000	-0.21445	+0.40287	+0.34849	0.002866	0.06090
141	141	+0.00325	+0.06044	+0.00674	+0.00036	+0.00673	-0.06053	+0.06081	-0.00327	+0.00000	-0.32701	+0.36182	+0.34422	0.002866	0.06090
142	142	-0.00710	+0.05677	-0.02086	+0.00259	-0.02070	-0.05721	+0.06043	-0.00755	-0.00000	-0.45500	+0.37608	+0.32802	0.002866	0.06090
143	143	+0.01093	-0.02838	-0.05277	-0.01896	-0.04924	-0.03041	+0.05683	-0.02189	+0.00000	-0.53854	+0.43896	+0.30788	0.002866	0.06090
144	144	-0.00559	-0.02989	-0.05277	-0.00970	-0.05187	+0.03041	+0.05986	-0.01119	+0.00000	-0.53854	+0.52104	+0.29580	0.002866	0.06090
145	145	+0.00954	-0.05641	-0.02086	+0.00348	-0.02057	+0.05721	+0.06005	+0.01016	+0.00000	-0.45500	+0.58392	+0.29789	0.002866	0.06090
146	146	-0.02574	-0.05478	+0.00674	+0.00287	+0.00610	-0.06053	+0.05512	-0.02590	-0.00000	-0.32701	+0.59818	+0.31061	0.002866	0.06090
147	147	+0.00517	-0.04838	+0.03662	-0.00389	+0.03642	+0.04866	+0.06056	+0.00647	+0.00000	-0.21445	+0.55713	+0.32684	0.002866	0.06090
148	148	+0.00000	+0.00000	+0.06090	+0.01082	+0.05993	+0.00000	+0.05993	-0.01082	+0.00000	+0.55000	+0.48000	+0.30021	0.002866	0.06090
149	149	-0.00541	+0.04836	+0.03662	-0.00407	+0.03640	-0.04866	+0.06052	-0.00677	+0.00000	+0.50555	+0.40287	+0.31767	0.002866	0.06090
150	150	+0.02700	+0.05417	+0.00674	+0.00301	+0.00603	-0.06053	+0.05450	-0.02717	+0.00000	+0.39299	+0.36182	+0.33730	0.002866	0.06090
151	151	+0.01214	+0.05591	-0.02086	-0.00443	-0.02039	+0.05721	+0.05951	-0.01293	-0.00000	+0.26500	+0.37608	+0.34801	0.002866	0.06090
152	152	+0.01224	+0.02783	-0.05277	-0.02124	-0.04830	-0.03041	+0.05575	-0.02452	+0.00000	+0.18146	+0.43896	+0.34599	0.002866	0.06090
153	153	-0.00038	-0.03041	-0.05277	-0.00066	-0.05276	+0.03041	+0.06090	-0.00077	+0.00000	+0.18146	+0.52104	+0.33451	0.002866	0.06090
154	154	-0.01569	-0.05502	-0.02086	-0.00572	-0.02006	+0.05721	+0.05857	-0.01670	+0.00000	+0.26500	+0.58392	+0.31868	0.002866	0.06090
155	155	+0.00136	-0.06051	+0.00674	-0.00015	+0.00674	+0.06053	+0.06088	+0.00137	-0.00000	+0.39299	+0.59818	+0.30338	0.002866	0.06090
156	156	+0.00925	-0.04777	+0.03662	-0.00696	+0.03596	+0.04866	+0.05979	+0.01158	+0.00000	+0.50555	+0.55713	+0.29511	0.002866	0.06090

Whiskers — 24 maps

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	sigma1
52	52	-0.01227	+0.14327	+0.01862	-0.00159	+0.01855	-0.14380	+0.14447	+0.01237	+0.00000	-0.71020	+0.34760	+0.06000	0.002866	0.14500
53	53	+0.06731	+0.12707	+0.01862	+0.00872	+0.01646	-0.14380	+0.12814	-0.06787	+0.00000	-1.09013	+0.29840	+0.06000	0.002866	0.14500
54	54	+0.04151	+0.13768	+0.01862	+0.00537	+0.01783	-0.14380	+0.13883	-0.04185	+0.00000	-1.47007	+0.24920	+0.06000	0.002866	0.14500
55	55	+0.05902	+0.13113	+0.01862	+0.00764	+0.01698	-0.14380	+0.13223	-0.05951	+0.00000	-1.85000	+0.20000	+0.06000	0.002866	0.14500
56	56	+0.03063	+0.14171	+0.00209	+0.00044	+0.00204	-0.14498	+0.14173	-0.03063	+0.00000	-0.71020	+0.47640	+0.06000	0.002866	0.14500
57	57	-0.00425	+0.14492	+0.00209	-0.00006	+0.00209	-0.14498	+0.14494	+0.00425	+0.00000	-1.09013	+0.47993	+0.06000	0.002866	0.14500
58	58	+0.05197	+0.13595	+0.00209	+0.00075	+0.00195	-0.14498	+0.13536	-0.05198	+0.00000	-1.47007	+0.46547	+0.06000	0.002866	0.14500
59	59	+0.06845	+0.12781	+0.00209	+0.00098	+0.00184	-0.14498	+0.12782	-0.06846	+0.00000	-1.85000	+0.46000	+0.06000	0.002866	0.14500
60	60	+0.03503	+0.14015	-0.01247	-0.00302	-0.01210	-0.14446	+0.14067	-0.03516	+0.00000	-0.71020	+0.60160	+0.06000	0.002866	0.14500
61	61	+0.05481	+0.13366	-0.01247	-0.00473	-0.01154	-0.14446	+0.13416	-0.05501	+0.00000	-1.09013	+0.63440	+0.06000	0.002866	0.14500
62	62	+0.06472	+0.12915	-0.01247	-0.00559	-0.01115	-0.14446	+0.12963	-0.06497	+0.00000	-1.47007	+0.66720	+0.06000	0.002866	0.14500
63	63	+0.01756	+0.14339	-0.01247	-0.00152	-0.01238	-0.14446	+0.14392	-0.01763	+0.00000	-1.85000	+0.70000	+0.06000	0.002866	0.14500
64	64	-0.04510	-0.13654	+0.01862	+0.00584	+0.01768	+0.14380	+0.13768	-0.04548	-0.00000	+0.71020	+0.34760	+0.06000	0.002866	0.14500
65	65	-0.00829	-0.14356	+0.01862	+0.00107	+0.01859	+0.14380	+0.14476	-0.00836	-0.00000	+1.09013	+0.29840	+0.06000	0.002866	0.14500
66	66	-0.05446	-0.13309	+0.01862	+0.00705	+0.01723	+0.14380	+0.13426	-0.05492	-0.00000	+1.47007	+0.24920	+0.06000	0.002866	0.14500
67	67	+0.01884	-0.14256	+0.01862	-0.00244	+0.01846	+0.14380	+0.14375	+0.01900	-0.00000	-1.85000	+0.20000	+0.06000	0.002866	0.14500
68	68	-0.03902	-0.13963	+0.00209	+0.00056	+0.00201	+0.14498	+0.13965	-0.03903	-0.00000	+0.71020	+0.47640	+0.06000	0.002866	0.14500
69	69	-0.06362	-0.13028	+0.00209	+0.00092	+0.00187	+0.14498	+0.13030	-0.06362	-0.00000	+1.09013	+0.47993	+0.06000	0.002866	0.14500
70	70	-0.01270	-0.14443	+0.00209	+0.00018	+0.00208	+0.14498	+0.14444	-0.01270	-0.00000	+1.47007	+0.46547	+0.06000	0.002866	0.14500

71	71	+0.01749	-0.14393	+0.00209	-0.00025	+0.00207	+0.14498	+0.14394	+0.01749	-0.00000		+1.85000	+0.46000	+0.06000		0.002866	0.14500
72	72	-0.03150	-0.14099	-0.01247	-0.00272	-0.01217	+0.14446	+0.14151	-0.03162	+0.00000		+0.71020	+0.60160	+0.06000		0.002866	0.14500
73	73	+0.02018	-0.14305	-0.01247	+0.00174	-0.01235	+0.14446	+0.14358	+0.02025	+0.00000		+1.09013	+0.63440	+0.06000		0.002866	0.14500
74	74	-0.02560	-0.14218	-0.01247	-0.00221	-0.01227	+0.14446	+0.14270	-0.02570	+0.00000		+1.47007	+0.66720	+0.06000		0.002866	0.14500
75	75	-0.03300	-0.14064	-0.01247	-0.00285	-0.01214	+0.14446	+0.14117	-0.03312	+0.00000		+1.85000	+0.70000	+0.06000		0.002866	0.14500

Part II — Self-similar texture: 316 Barnsley fern maps

Four maps (stem, frond, left and right leaflet) composed onto every second position.

Skull oval — 56 maps (14 positions × 4)

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33		b1	b2	b3		p	sigma1
157	0	+0.00000	+0.02588	-0.03882	+0.00000	-0.02183	-0.04601	+0.00000	-0.00609	+0.00000		-0.73866	-0.67575	+0.25512		0.000070	0.06020
158	0	+0.03121	+0.13630	-0.03882	-0.02633	-0.11500	-0.04601	+0.17834	-0.04084	+0.00000		-0.99743	-0.45743	+0.31606		0.005222	0.18295
159	0	-0.03138	+0.04315	-0.03882	+0.02647	-0.03640	-0.04601	+0.05108	+0.04664	+0.00000		-0.99743	-0.45743	+0.31606		0.000835	0.07325
160	0	-0.04642	+0.03066	-0.03882	+0.03916	-0.02587	-0.04601	-0.02184	-0.06839	+0.00000		-0.80982	-0.61571	+0.27188		0.000835	0.08152
161	2	+0.00000	+0.01645	-0.05261	+0.00000	-0.02959	-0.02925	+0.00000	-0.00609	-0.00000		-0.95236	-0.40110	+0.24210		0.000070	0.06020
162	2	+0.01984	+0.08666	-0.05261	-0.03569	-0.15587	-0.02925	+0.17834	-0.04084	-0.00000		-1.11687	-0.10520	+0.30304		0.005222	0.18295
163	2	-0.01995	+0.02743	-0.05261	-0.03588	-0.04934	-0.02925	+0.05108	+0.04664	-0.00000		-1.11687	-0.10520	+0.30304		0.000835	0.07325
164	2	-0.02951	+0.01949	-0.05261	+0.05308	-0.03506	-0.02925	-0.02184	-0.06839	-0.00000		-0.99760	-0.31973	+0.25886		0.000835	0.08152
165	4	+0.00000	+0.00417	-0.05974	+0.00000	-0.03360	-0.00742	+0.00000	-0.00609	-0.00000		-1.04708	-0.07672	+0.23455		0.000070	0.06020
166	4	+0.00503	+0.02197	-0.05974	-0.04053	-0.17698	-0.00742	+0.17834	-0.04084	-0.00000		-1.08878	+0.25926	+0.29549		0.005222	0.18295
167	4	-0.00506	+0.00695	-0.05974	+0.04074	-0.05602	-0.00742	+0.05108	+0.04664	-0.00000		-1.08878	+0.25926	+0.29549		0.000835	0.07325
168	4	-0.00748	+0.00494	-0.05974	+0.06027	-0.03981	-0.00742	-0.02184	-0.06839	-0.00000		-1.05855	+0.01568	+0.25131		0.000835	0.08152
169	6	+0.00000	-0.00893	-0.05807	+0.00000	-0.03266	+0.01587	+0.00000	-0.00609	+0.00000		-1.01152	+0.25652	+0.23636		0.000070	0.06020
170	6	-0.01077	-0.04701	-0.05807	-0.03939	-0.17203	+0.01587	+0.17834	-0.04084	+0.00000		-0.92226	+0.58310	+0.29730		0.005222	0.18295
171	6	+0.01082	-0.01488	-0.05807	+0.03960	-0.05446	+0.01587	+0.05108	+0.04664	+0.00000		-0.92226	+0.58310	+0.29730		0.000835	0.07325
172	6	+0.01601	-0.01058	-0.05807	+0.05858	-0.03870	+0.01587	-0.02184	-0.06839	+0.00000		-0.98697	+0.34633	+0.25312		0.000835	0.08152
173	8	+0.00000	-0.02033	-0.04814	+0.00000	-0.02708	+0.03614	+0.00000	-0.00609	-0.00000		-0.85086	+0.55739	+0.24658		0.000070	0.06020
174	8	-0.02452	-0.10707	-0.04814	-0.03266	-0.14262	+0.03614	+0.17834	-0.04084	-0.00000		-0.64760	+0.82814	+0.30752		0.005222	0.18295
175	8	+0.02465	-0.03389	-0.04814	+0.03283	-0.04515	+0.03614	+0.05108	+0.04664	-0.00000		-0.64760	+0.82814	+0.30752		0.000835	0.07325
176	8	+0.03646	-0.02409	-0.04814	+0.04857	-0.03208	+0.03614	-0.02184	-0.06839	-0.00000		-0.79496	+0.63185	+0.26334		0.000835	0.08152
177	10	+0.00000	-0.02844	-0.03266	+0.00000	-0.01837	+0.05057	+0.00000	-0.00609	+0.00000		-0.58521	+0.78925	+0.26003		0.000070	0.06020
178	10	-0.03431	-0.14981	-0.03266	-0.02216	-0.09675	+0.05057	+0.17834	-0.04084	+0.00000		-0.30080	+0.97292	+0.32097		0.005222	0.18295
179	10	+0.03448	-0.04742	-0.03266	+0.02227	-0.03063	+0.05057	+0.05108	+0.04664	+0.00000		-0.30080	+0.97292	+0.32097		0.000835	0.07325
180	10	+0.05102	-0.03370	-0.03266	+0.03295	-0.02176	+0.05057	-0.02184	-0.06839	+0.00000		-0.50700	+0.83976	+0.27679		0.000835	0.08152
181	12	+0.00000	-0.03287	-0.01439	+0.00000	-0.00809	+0.05845	+0.00000	-0.00609	-0.00000		-0.24702	+0.92373	+0.27024		0.000070	0.06020
182	12	-0.03965	-0.17317	-0.01439	-0.00976	-0.04263	+0.05845	+0.17834	-0.04084	-0.00000		+0.08172	+1.00466	+0.33118		0.005222	0.18295
183	12	+0.03986	-0.05482	-0.01439	+0.00981	-0.01350	+0.05845	+0.05108	+0.04664	-0.00000		+0.08172	+1.00466	+0.33118		0.000835	0.07325
184	12	+0.05897	-0.03896	-0.01439	+0.01452	-0.00959	+0.05845	-0.02184	-0.06839	-0.00000		-0.15662	+0.94598	+0.28700		0.000835	0.08152
185	14	+0.00000	-0.03375	+0.00482	+0.00000	-0.00271	+0.06001	+0.00000	-0.00609	+0.00000		+0.12230	+0.94391	+0.27254		0.000070	0.06020
186	14	-0.04071	-0.17776	+0.00482	+0.00327	-0.01428	+0.06001	+0.17834	-0.04084	+0.00000		+0.45977	+0.91680	+0.33348		0.005222	0.18295
187	14	+0.04092	-0.05627	+0.00482	-0.00329	-0.00452	+0.06001	+0.05108	+0.04664	+0.00000		+0.45977	+0.91680	+0.33348		0.000835	0.07325
188	14	+0.06053	-0.03999	+0.00482	-0.00486	-0.00321	+0.06001	-0.02184	-0.06839	+0.00000		+0.21510	+0.93646	+0.28930		0.000835	0.08152
189	16	+0.00000	-0.03111	+0.02374	+0.00000	-0.01335	+0.05532	+0.00000	-0.00609	+0.00000		+0.47702	+0.84674	+0.26591		0.000070	0.06020
190	16	-0.03753	-0.16389	+0.02374	+0.01610	-0.07031	+0.05532	+0.17834	-0.04084	+0.00000		+0.78816	+0.71325	+0.32685		0.005222	0.18295
191	16	+0.03772	-0.05188	+0.02374	-0.01619	-0.02226	+0.05532	+0.05108	+0.04664	+0.00000		+0.78816	+0.71325	+0.32685		0.000835	0.07325
192	16	+0.05581	-0.03687	+0.02374	-0.02394	+0.01582	+0.05532	-0.02184	-0.06839	+0.00000		+0.56259	+0.81003	+0.28266		0.000835	0.08152
193	18	+0.00000	-0.02484	+0.04090	+0.00000	-0.02300	+0.04417	+0.00000	-0.00609	+0.00000		+0.77271	+0.64383	+0.25332		0.000070	0.06020
194	18	-0.02996	-0.13085	+0.04090	+0.02775	-0.12117	+0.04417	+0.17834	-0.04084	+0.00000		+1.02112	+0.41379	+0.31426		0.005222	0.18295
195	18	+0.03012	-0.04142	+0.04090	-0.02789	-0.03836	+0.04417	+0.05108	+0.04664	+0.00000		+1.02112	+0.41379	+0.31426		0.000835	0.07325
196	18	+0.04456	-0.02944	+0.04090	-0.04126	-0.02726	+0.04417	-0.02184	-0.06839	+0.00000		+0.84102	+0.58057	+0.27008		0.000835	0.08152
197	20	+0.00000	-0.01497	+0.05400	+0.00000	-0.03037	+0.02662	+0.00000	-0.00609	+0.00000		+0.97200	+0.36048	+0.24067		0.000070	0.06020
198	20	-0.01806	-0.07885	+0.05400	-0.03663	-0.15996	+0.02662	+0.17834	-0.04084	+0.00000		+1.12169	+0.05681	+0.30161		0.005222	0.18295
199	20	+0.01815	-0.02496	+0.05400	-0.03682	+0.05064	+0.02662	+0.05108	+0.04664	+0.00000		+1.12169	+0.05681	+0.30161		0.000835	0.07325
200	20	+0.02685	-0.01774	+0.05400	-0.05447	-0.03598	+0.02662	-0.02184	-0.06839	+0.00000		+1.01317	+0.27697	+0.25743		0.000835	0.08152
201	22	+0.00000	-0.00244	+0.06004	+0.00000	-0.03377	+0.00434	+0.00000	-0.00609	+0.00000		+1.04989	+0.03253	+0.23422		0.000070	0.06020
202	22	-0.00294	-0.01285	+0.06004	+0.04073	-0.17787	+0.00434	+0.17834	-0.04084	+0.00000		+1.07429	-0.30515	+0.29516		0.005222	0.18295
203	22	+0.00296	-0.00407	+0.06004	-0.04094	-0.05631	+0.00434	+0.05108	+0.04664	+0.00000		+1.07429	-0.30515	+0.29516		0.000835	0.07325
204	22	+0.00438	-0.00289	+0.06004	-0.06057	-0.04001	+0.00434	-0.02184	-0.06839	+0.00000		+1.05660	-0.06033	+0.25098		0.000835	0.08152
205	24	+0.00000	-0.01057	+0.05719	+0.00000	-0.03216	-0.01879	+0.00000	-0.00609	+0.00000		+0.99725	-0.29875	+0.23730		0.000070	0.06020
206	24	+0.01275	-0.05567	+0.05719	+0.03880	-0.16943	-0.01879	+0.17834	-0.04084	+0.00000		+0.89158	-0.62039	+0.29824		0.005222	0.18295
207	24	-0.01281	-0.01762	+0.05719	-0.03900	-0.05363	-0.01879	+0.05108	+0.04664	+0.00000		+0.89158	-0.62039	+0.29824		0.000835	0.07325
208	24	-0.01896	-0.01252	+0.05719	-0.05769	-0.03811	-0.01879	-0.02184	-0.06839	+0.00000		+0.96819	-0.38720	+0.25406		0.000835	0.08152
209	26	+0.00000	-0.02160	+0.04636	+0.00000	-0.02607	-0.03840	+0.00000	-0.00609	-0.00000		+0.82136	-0.59250	+0.24831		0.000070	0.06020
210	26	+0.02605	-0.11377	+0.04636	+0.03145	-0.13733	-0.03840	+0.17834	-0.04084	-0.00000		+0.60538	-0.85322	+0.30925		0.005222	0.18295
211	26	-0.02619	+0.03601	+0.04636	-0.03161	-0.04347	-0.03840	+0.05108	+0.04664	-0.00000		+0.60538	-0.85322	+0.30925		0.000835	0.07325
212	26	-0.03874	-0.02559	+0.04636	-0.04677	-0.03090	-0.03840	-0.02184	-0.06839	-0.00000		+0.76197	-0.66420	+0.26506		0.000835	0.08152

244	42	+0.06218	-0.03180	+0.00851	-0.00888	+0.00454	+0.05959	-0.01483	-0.07249	+0.00000	+0.75711	-1.12416	+0.08878	0.000835	0.08152
245	44	+0.00000	-0.03262	+0.00851	+0.00000	+0.00466	+0.05959	+0.00000	-0.00988	+0.00000	+0.61477	-1.47982	+0.00862	0.000070	0.06020
246	44	-0.06014	-0.17084	+0.00851	+0.00859	+0.02441	+0.05959	+0.17257	-0.06075	+0.00000	+0.94095	-1.52642	+0.10747	0.005222	0.18295
247	44	+0.03466	-0.06075	+0.00851	-0.00495	+0.00868	+0.05959	+0.05540	-0.03995	+0.00000	+0.94095	-1.52642	+0.10747	0.000835	0.07325
248	44	+0.06218	-0.03180	+0.00851	-0.00888	+0.00454	+0.05959	-0.01483	-0.07249	+0.00000	+0.70447	-1.49264	+0.03580	0.000835	0.08152

Ear, inner edge — 24 maps (6 positions × 4)

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	sigma1
225	34	+0.00000	+0.02947	-0.02692	+0.00000	-0.01474	-0.05384	+0.00000	-0.00988	+0.00000	-0.56410	-1.60395	+0.00862	0.000070	0.06020
226	34	+0.05434	+0.15435	-0.02692	-0.02717	-0.07718	-0.05384	+0.17257	-0.06075	+0.00000	-0.85881	-1.45660	+0.10747	0.005222	0.18295
227	34	-0.03131	+0.05489	-0.02692	+0.01566	-0.02744	-0.05384	+0.05540	+0.03995	+0.00000	-0.85881	-1.45660	+0.10747	0.000835	0.07325
228	34	-0.05618	-0.02873	-0.02692	+0.02809	-0.01437	-0.05384	-0.01483	-0.07249	+0.00000	-0.64515	-1.56343	+0.03580	0.000835	0.08152
229	36	+0.00000	+0.02947	-0.02692	+0.00000	-0.01474	-0.05384	+0.00000	-0.00988	+0.00000	-0.42874	-1.33323	+0.00862	0.000070	0.06020
230	36	+0.05434	+0.15435	-0.02692	-0.02717	-0.07718	-0.05384	+0.17257	-0.06075	+0.00000	-0.72345	-1.18588	+0.10747	0.005222	0.18295
231	36	-0.03131	+0.05489	-0.02692	+0.01566	-0.02744	-0.05384	+0.05540	+0.03995	+0.00000	-0.72345	-1.18588	+0.10747	0.000835	0.07325
232	36	-0.05618	-0.02873	-0.02692	+0.02809	-0.01437	-0.05384	-0.01483	-0.07249	+0.00000	-0.50979	-1.29271	+0.03580	0.000835	0.08152
233	38	+0.00000	+0.02947	-0.02692	+0.00000	-0.01474	-0.05384	+0.00000	-0.00988	+0.00000	-0.29338	-1.06251	+0.11006	0.000070	0.06020
234	38	+0.05434	+0.15435	-0.02692	-0.02717	-0.07718	-0.05384	+0.17257	-0.06075	+0.00000	-0.58809	-0.91516	+0.20891	0.005222	0.18295
235	38	-0.03131	+0.05489	-0.02692	+0.01566	-0.02744	-0.05384	+0.05540	+0.03995	+0.00000	-0.58809	-0.91516	+0.20891	0.000835	0.07325
236	38	-0.05618	-0.02873	-0.02692	+0.02809	-0.01437	-0.05384	-0.01483	-0.07249	+0.00000	-0.37443	-1.02199	+0.13725	0.000835	0.08152
249	46	+0.00000	+0.02947	-0.02692	+0.00000	-0.01474	-0.05384	+0.00000	-0.00988	-0.00000	+0.56410	-1.60395	+0.00862	0.000070	0.06020
250	46	+0.05434	+0.15435	-0.02692	-0.02717	-0.07718	+0.05384	+0.17257	-0.06075	-0.00000	+0.85881	-1.45660	+0.10747	0.005222	0.18295
251	46	+0.03131	+0.05489	-0.02692	+0.01566	-0.02744	+0.05384	+0.05540	+0.03995	-0.00000	+0.85881	-1.45660	+0.10747	0.000835	0.07325
252	46	+0.05618	-0.02873	-0.02692	+0.02809	-0.01437	+0.05384	-0.01483	-0.07249	-0.00000	+0.64515	-1.56343	+0.03580	0.000835	0.08152
253	48	+0.00000	+0.02947	-0.02692	+0.00000	-0.01474	+0.05384	+0.00000	-0.00988	-0.00000	-0.42874	-1.33323	+0.00862	0.000070	0.06020
254	48	+0.05434	+0.15435	-0.02692	-0.02717	-0.07718	+0.05384	+0.17257	-0.06075	-0.00000	+0.72345	-1.18588	+0.10747	0.005222	0.18295
255	48	+0.03131	+0.05489	-0.02692	+0.01566	-0.02744	+0.05384	+0.05540	+0.03995	-0.00000	+0.72345	-1.18588	+0.10747	0.000835	0.07325
256	48	+0.05618	-0.02873	-0.02692	+0.02809	-0.01437	+0.05384	-0.01483	-0.07249	-0.00000	+0.50979	-1.29271	+0.03580	0.000835	0.08152
257	50	+0.00000	+0.02947	-0.02692	+0.00000	-0.01474	+0.05384	+0.00000	-0.00988	-0.00000	-0.29338	-1.06251	+0.11006	0.000070	0.06020
258	50	+0.05434	+0.15435	-0.02692	-0.02717	-0.07718	+0.05384	+0.17257	-0.06075	-0.00000	-0.58809	-0.91516	+0.20891	0.005222	0.18295
259	50	+0.03131	+0.05489	-0.02692	+0.01566	-0.02744	+0.05384	+0.05540	+0.03995	-0.00000	-0.58809	-0.91516	+0.20891	0.000835	0.07325
260	50	+0.05618	-0.02873	-0.02692	+0.02809	-0.01437	+0.05384	-0.01483	-0.07249	-0.00000	+0.37443	-1.02199	+0.13725	0.000835	0.08152

Eye almond — 64 maps (16 positions × 4)

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	sigma
309	76	+0.00000	-0.00245	+0.02124	+0.00000	+0.01208	+0.00431	+0.00000	-0.00123	+0.00000	-0.10385	-0.13093	+0.37548	0.000070	0.02167
310	76	-0.00191	-0.01294	+0.02124	+0.00944	+0.06386	+0.00431	+0.06516	-0.00963	+0.00000	-0.07936	-0.25170	+0.38781	0.005222	0.06586
311	76	+0.00321	-0.00376	+0.02124	-0.01585	+0.01857	+0.00431	+0.01717	+0.01833	+0.00000	-0.07936	-0.25170	+0.38781	0.000835	0.02637
312	76	+0.00421	-0.00324	+0.02124	-0.02076	+0.01600	+0.00431	-0.00955	-0.02341	+0.00000	-0.09711	-0.16414	+0.37887	0.000835	0.02935
313	78	+0.00000	+0.00940	+0.01401	+0.00000	+0.00797	-0.01654	+0.00000	-0.00123	+0.00000	-0.17712	-0.25915	+0.36615	0.000070	0.02167
314	78	+0.00735	+0.04971	+0.01401	+0.00622	+0.04212	-0.01654	+0.06516	-0.00963	+0.00000	-0.27114	-0.33880	+0.37847	0.005222	0.06586
315	78	-0.01234	+0.01446	+0.01401	-0.01045	+0.01225	-0.01654	+0.01717	+0.01833	+0.00000	-0.27114	-0.33880	+0.37847	0.000835	0.02637
316	78	-0.01616	+0.01246	+0.01401	-0.01369	+0.01055	-0.01654	-0.00955	-0.02341	+0.00000	-0.20297	-0.28105	+0.36954	0.000835	0.02935
317	80	+0.00000	+0.01208	+0.00431	+0.00000	+0.00245	-0.02124	+0.00000	-0.00123	+0.00000	-0.37775	-0.34385	+0.34329	0.000070	0.02167
318	80	+0.00944	+0.06386	+0.00431	+0.00191	-0.01294	-0.02124	+0.06516	-0.00963	+0.00000	-0.49852	-0.36833	+0.35562	0.005222	0.06586
319	80	-0.01585	+0.01857	+0.00431	-0.00321	+0.00376	-0.02124	+0.01717	+0.01833	+0.00000	-0.49852	-0.36833	+0.35562	0.000835	0.02637
320	80	-0.02076	+0.01600	+0.00431	-0.00421	+0.00324	-0.02124	-0.00955	-0.02341	+0.00000	-0.41096	-0.35058	+0.34668	0.000835	0.02935
321	82	+0.00000	+0.01176	-0.00646	+0.00000	-0.00368	-0.02069	+0.00000	-0.00123	+0.00000	-0.59498	-0.33329	+0.31314	0.000070	0.02167
322	82	+0.00919	+0.06219	-0.00646	-0.00287	-0.01943	-0.02069	+0.06516	-0.00963	+0.00000	-0.71260	-0.29654	+0.32546	0.005222	0.06586
323	82	-0.01544	+0.01808	-0.00646	+0.00482	-0.00565	-0.02069	+0.01717	+0.01833	+0.00000	-0.71260	-0.29654	+0.32546	0.000835	0.02637
324	82	-0.02022	+0.01558	-0.00646	-0.00632	-0.00487	-0.02069	-0.00955	-0.02341	+0.00000	-0.62733	-0.32318	+0.31653	0.000835	0.02935
325	84	+0.00000	+0.00245	-0.02124	+0.00000	-0.01208	-0.00431	+0.00000	-0.00123	-0.00000	-0.69615	-0.22907	+0.30033	0.000070	0.02167
326	84	+0.00191	-0.01294	-0.02124	-0.00944	-0.06386	-0.00431	+0.06516	-0.00963	-0.00000	-0.72064	-0.10830	+0.31266	0.005222	0.06586
327	84	-0.00321	+0.00376	-0.02124	+0.01585	-0.01857	-0.00431	+0.01717	+0.01833	-0.00000	-0.72064	-0.10830	+0.31266	0.000835	0.02637
328	84	-0.00421	+0.00324	-0.02124	+0.02076	-0.01600	-0.00431	-0.00955	-0.02341	-0.00000	-0.70289	-0.19586	+0.30372	0.000835	0.02935
329	86	+0.00000	-0.00940	-0.01401	+0.00000	-0.00797	+0.01654	+0.00000	-0.00123	+0.00000	-0.62288	-0.10085	+0.31995	0.000070	0.02167
330	86	-0.00735	-0.04971	-0.01401	-0.00622	-0.04212	+0.01654	+0.06516	-0.00963	+0.00000	-0.52886	-0.02120	+0.33227	0.005222	0.06586
331	86	+0.01234	-0.01446	-0.01401	+0.01045	-0.01225	+0.01654	+0.01717	+0.01833	+0.00000	-0.52886	-0.02120	+0.33227	0.000835	0.02637
332	86	+0.01616	-0.01246	-0.01401	-0.01369	-0.01055	-0.01654	-0.00955	-0.02341	+0.00000	-0.59703	-0.07895	+0.32334	0.000835	0.02935
333	88	+0.00000	-0.01208	-0.00431	+0.00000	-0.00245	-0.02124	+0.00000	-0.00123	+0.00000	-0.42225	-0.01615	+0.35254	0.000070	0.02167
334	88	-0.00944	-0.06386	-0.00431	-0.00191	-0.01294	+0.02124	+0.06516	-0.00963	+0.00000	-0.30148	+0.00833	+0.36486	0.005222	0.06586
335	88	+0.01585	-0.01857	-0.00431	+0.00321	-0.00376	+0.02124	+0.01717	+0.01833	+0.00000	-0.30148	+0.00833	+0.36486	0.000835	0.02637
336	88	+0.02076	-0.01600	-0.00431	+0.00421	-0.00324	+0.02124	-0.00955	-0.02341	+0.00000	-0.38904	-0.00942	+0.35593	0.000835	0.02935
337	90	+0.00000	-0.01176	-0.00646	+0.00000	-0.00368	-0.02069	+0.00000	-0.00123	+0.00000	-0.20502	-0.02671	+0.37252	0.000070	0.02167
338	90	-0.00919	-0.06219	-0.00646	+0.00287	+0.01943	-0.02069	+0.06516	-0.00963	+0.00000	-0.08740	-0.06346	+0.38484	0.005222	0.06586
339	90	+0.01544	-0.01808	-0.00646	-0.00482	-0.00565	-0.02069	+0.01717	+0.01833	+0.00000	-0.08740	-0.06346	+0.38484	0.000835	0.02637
340	90	-0.02022	-0.01558	-0.00646	-0.00632	-0.00487	-0.02069	-0.00955	-0.02341	+0.00000	-0.17267	-0.03682	+0.37591	0.000835	0.02935
345	94	+0.00000	+0.00245	-0.02124	+0.00000	-0.01208	-0.00431	+0.00000	-0.00123	-0.00000	+0.69188	-0.25013	+0.30033	0.000070	0.02167
346	94	+0.00191	-0.01294	-0.02124	+0.00944	+0.06386	-0.00431	+0.06516	-0.00963	-0.00000	+0.66740	-0.37090	+0.31266	0.005222	0.06586
347	94	-0.00321	+0.00376	-0.02124	-0.01585	+0.01857	-0.00431	+0.01717	+0.01833	-0.00000	+0.66740	-0.37090	+0.31266	0.000835	0.02637
348	94	-0.00421	+0.00324	-0.02124	-0.02076	+0.01600	-0.00431	-0.00955	-0.02341	-0.00000	+0.68515	-0.28334	+0.30372	0.000835	0.02935
349	96	+0.00000	+0.01176	-0.00646	+0.00000	+0.00368	-0.02069	+0.00000	-0.00123	+0.00000	+0.57447	-0.33970	+0.31314	0.000070	0.02167
350	96	+0.00919	+0.06219	-0.00646	-0.00287	-0.01943	-0.02069	+0.06516	-0.00963	+0.00000	+0.45685	-0.37645	+0.32546	0.005222	0.06586
351	96	-0.01544	+0.01808	+0.00646	-0.00482	+0.00565	-0.02069	+0.01717	+0.01833	+0.00000	+0.45685	-0.37645	+0.32546	0.000835	0.02637
352	96	-0.02022	+0.01558	-0.00646	-0.00632	+0.00487	-0.02069	-0.00955	-0.02341	+0.00000	+0.54212	-0.34980	+0.31653	0.000835	0.02935
353	98	+0.00000	+0.01208	-0.00431	+0.00000	-0.00245	-0.02124	+0.00000	-0.00123	+0.00000	+0.35669	-0.33958	+0.34329	0.000070	0.02167
354	98	+0.00944	+0.06386	-0.00431	-0.00191	-0.01294	-0.02124	+0.06516	-0.00963	-0.00000	+0.23592	-0.31509	+0.35562	0.005222	0.06586
355	98	-0.01585	+0.01857	-0.00431	+0.00321	-0.00376	-0.02124	+0.01717	+0.01833	-0.00000	+0.23592	-0.31509	+0.35562	0.000835	0.02637
356	98	-0.02076	+0.01600	-0.00431	+0.00421	-0.00324	-0.02124	-0.00955	-0.02341	-0.00000	+0.32348	-0.33284	+0.34668	0.000835	0.02935
357	100	+0.00000	+0.00940	-0.01401	+0.00000	-0.00797	-0.01654	+0.00000	-0.00123	+0.00000	+0.16072	-0.24526	+0.36615	0.000070	0.02167
358	100	+0.00735	+0.04971	-0.01401	-0.00622	-0.04212	-0.01654	+0.06516	-0.00963	+0.00000	+0.06670	-0.16560	+0.37847	0.005222	0.06586
359	100	-0.01234	+0.01446	-0.01401	+0.01045	-0.01225	-0.01654	+0.01717	+0.01833	+0.00000	+0.06670	-0.16560	+0.37847	0.000835	0.02637
360	100	-0.01616	+0.01246	-0.01401	-0.01369	-0.01055	-0.01654	-0.00955	-0.02341	+0.00000	+0.13486	-0.22335	+0.36954	0.000835	0.02935
361	102	+0.00000	-0.00245	-0.02124	+0.00000	-0.01208	-0.00431	+0.00000	-0.00123	+0.00000	+0.10812	-0.10987	+0.37548	0.000070	0.02167
362	102	-0.00191	-0.01294	-0.02124	-0.00944	-0.06386	-0.00431	+0.06516	-0.00963	+0.00000	+0.13260	+0.01090	+0.38781	0.005222	0.06586
363	102	+0.00321	-0.00376	-0.02124	+0.01585	-0.01857	+0.00431	+0.01717	+0.01833	+0.00000	+0.13260	+0.01090	+0.38781	0.000835	0.02637
364	102	+0.00421	-0.00324	-0.02124	+0.02076	-0.01600	-0.00431	-0.00955	-0.02341	+0.00000	+0.11485	-0.07666	+0.37887	0.000835	0.02935
365	104	+0.00000	-0.01176	-0.00646	+0.00000	-0.00368	-0.02069	+0.00000	-0.00123	+0.00000	+0.22553	-0.02030	+0.37252	0.000070	0.02167
366	104	-0.00919	+0.06219	-0.00646	-0.00287	-0.01943	-0.02069	+0.06516	-0.00963	+0.00000	+0.34315	-0.01645	+0.38484	0.005222	0.06586

367	104	+0.01544	-0.01808	-0.00646	+0.00482	-0.00565	+0.02069	+0.01717	+0.01833	+0.00000		+0.34315	+0.01645	+0.38484		0.000835	0.02637
368	104	+0.02022	-0.01558	-0.00646	+0.00632	-0.00487	+0.02069	-0.00955	-0.02341	+0.00000		+0.25788	-0.01020	+0.37591		0.000835	0.02935
369	106	+0.00000	-0.01208	+0.00431	+0.00000	+0.00245	+0.02124	+0.00000	-0.00123	+0.00000		+0.44331	-0.02042	+0.35254		0.000070	0.02167
370	106	-0.00944	-0.06386	+0.00431	+0.00191	+0.01294	+0.02124	+0.06516	-0.00963	+0.00000		+0.56408	-0.04491	+0.36486		0.005222	0.06586
371	106	+0.01585	-0.01857	+0.00431	-0.00321	+0.00376	+0.02124	+0.01717	+0.01833	+0.00000		+0.56408	-0.04491	+0.36486		0.000835	0.02637
372	106	+0.02076	-0.01600	+0.00431	-0.00421	+0.00324	+0.02124	-0.00955	-0.02341	+0.00000		+0.47652	-0.02716	+0.35593		0.000835	0.02935
373	108	+0.00000	-0.00940	+0.01401	+0.00000	+0.00797	+0.01654	+0.00000	-0.00123	+0.00000		+0.63928	-0.11474	+0.31995		0.000070	0.02167
374	108	-0.00735	-0.04971	+0.01401	+0.00622	+0.04212	+0.01654	+0.06516	-0.00963	-0.00000		+0.73330	-0.19440	+0.33227		0.005222	0.06586
375	108	+0.01234	-0.01446	+0.01401	-0.01045	+0.01225	+0.01654	+0.01717	+0.01833	-0.00000		+0.73330	-0.19440	+0.33227		0.000835	0.02637
376	108	+0.01616	-0.01246	+0.01401	-0.01369	+0.01055	+0.01654	-0.00955	-0.02341	-0.00000		+0.66514	-0.13665	+0.32334		0.000835	0.02935

Catchlight — 8 maps (2 positions × 4)

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33		b1	b2	b3		p	sigma1
341	92	+0.00000	-0.00262	+0.00147	+0.00000	+0.00078	+0.00490	+0.00000	-0.00105	-0.00000		-0.32272	-0.24068	+0.39532		0.000070	0.00512
342	92	-0.00598	-0.01364	+0.00147	+0.00179	+0.00409	+0.00490	+0.01424	-0.00624	-0.00000		-0.29657	-0.24853	+0.40578		0.005222	0.01555
343	92	+0.00251	-0.00522	+0.00147	-0.00075	+0.00157	+0.00490	+0.00492	+0.00300	-0.00000		-0.29657	-0.24853	+0.40578		0.000835	0.00623
344	92	+0.00519	-0.00217	+0.00147	-0.00156	+0.00065	+0.00490	-0.00086	-0.00635	-0.00000		-0.31553	-0.24284	+0.39819		0.000835	0.00693
377	110	+0.00000	-0.00262	+0.00147	+0.00000	+0.00078	+0.00490	+0.00000	-0.00105	-0.00000		+0.32728	-0.24068	+0.39532		0.000070	0.00512
378	110	-0.00598	-0.01364	+0.00147	+0.00179	+0.00409	+0.00490	+0.01424	-0.00624	-0.00000		+0.35343	-0.24853	+0.40578		0.005222	0.01555
379	110	+0.00251	-0.00522	+0.00147	-0.00075	+0.00157	+0.00490	+0.00492	+0.00300	-0.00000		+0.35343	-0.24853	+0.40578		0.000835	0.00623
380	110	+0.00519	-0.00217	+0.00147	-0.00156	+0.00065	+0.00490	-0.00086	-0.00635	-0.00000		+0.33447	-0.24284	+0.39819		0.000835	0.00693

Nose triangle — 32 maps (8 positions × 4)

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33		b1	b2	b3		p	sigma1
381	112	+0.00000	-0.00885	+0.00000	+0.00000	+0.00000	+0.01565	+0.00000	-0.00133	+0.00000		-0.08829	+0.26000	+0.40920		0.000070	0.01565
382	112	-0.00926	-0.04666	+0.00000	+0.00000	+0.00000	+0.01565	+0.04666	-0.00926	+0.00000		+0.00016	+0.26000	+0.42246		0.005222	0.04757
383	112	+0.01106	-0.01432	+0.00000	+0.00000	+0.00000	+0.01565	+0.01296	+0.01255	+0.00000		+0.00016	+0.26000	+0.42246		0.000835	0.01905
384	112	+0.01562	-0.01095	+0.00000	+0.00000	+0.00000	+0.01565	-0.00614	-0.01747	+0.00000		-0.06396	+0.26000	+0.41285		0.000835	0.02119
385	114	+0.00000	-0.00885	+0.00000	+0.00000	+0.00000	+0.01565	+0.00000	-0.00133	+0.00000		+0.00771	+0.26000	+0.41064		0.000070	0.01565
386	114	-0.00926	-0.04666	+0.00000	+0.00000	+0.00000	+0.01565	+0.04666	-0.00926	+0.00000		+0.09616	+0.26000	+0.42391		0.005222	0.04757
387	114	+0.01106	-0.01432	+0.00000	+0.00000	+0.00000	+0.01565	+0.01296	+0.01255	+0.00000		+0.09616	+0.26000	+0.42391		0.000835	0.01905
388	114	+0.01562	-0.01095	+0.00000	+0.00000	+0.00000	+0.01565	-0.00614	-0.01747	+0.00000		+0.03204	+0.26000	+0.41429		0.000835	0.02119
389	116	+0.00000	-0.00885	+0.00000	+0.00000	+0.00000	+0.01565	+0.00000	-0.00133	+0.00000		+0.10371	+0.26000	+0.40920		0.000070	0.01565
390	116	-0.00926	-0.04666	+0.00000	+0.00000	+0.00000	+0.01565	+0.04666	-0.00926	+0.00000		+0.19216	+0.26000	+0.42246		0.005222	0.04757
391	116	+0.01106	-0.01432	+0.00000	+0.00000	+0.00000	+0.01565	+0.01296	+0.01255	+0.00000		+0.19216	+0.26000	+0.42246		0.000835	0.01905
392	116	+0.01562	-0.01095	+0.00000	+0.00000	+0.00000	+0.01565	-0.00614	-0.01747	+0.00000		+0.12804	+0.26000	+0.41285		0.000835	0.02119
393	118	+0.00000	-0.00491	-0.01302	+0.00000	-0.00736	+0.00868	+0.00000	-0.00133	+0.00000		-0.07972	+0.32042	+0.40467		0.000070	0.01565
394	118	-0.00514	-0.02588	-0.01302	-0.00770	-0.03882	+0.00868	+0.04666	-0.00926	+0.00000		-0.03066	+0.39401	+0.41794		0.005222	0.04757
395	118	+0.00613	-0.00794	-0.01302	+0.00920	-0.01191	+0.00868	+0.01296	+0.01255	+0.00000		-0.03066	+0.39401	+0.41794		0.000835	0.01905
396	118	+0.00866	-0.00607	-0.01302	+0.01299	-0.00911	+0.00868	-0.00614	-0.01747	+0.00000		-0.06623	+0.34066	+0.40832		0.000835	0.02119
397	120	+0.00000	-0.00491	-0.01302	+0.00000	-0.00736	+0.00868	+0.00000	-0.00133	+0.00000		-0.03172	+0.39242	+0.39762		0.000070	0.01565
398	120	-0.00514	-0.02588	-0.01302	-0.00770	-0.03882	+0.00868	+0.04666	-0.00926	+0.00000		+0.01734	+0.46601	+0.41088		0.005222	0.04757
399	120	+0.00613	-0.00794	-0.01302	+0.00920	-0.01191	+0.00868	+0.01296	+0.01255	+0.00000		+0.01734	+0.46601	+0.41088		0.000835	0.01905
400	120	+0.00866	-0.00607	-0.01302	+0.01299	-0.00911	+0.00868	-0.00614	-0.01747	+0.00000		-0.01823	+0.41266	+0.40126		0.000835	0.02119
401	122	+0.00000	-0.00491	-0.01302	+0.00000	-0.00736	+0.00868	+0.00000	-0.00133	+0.00000		+0.10372	+0.28442	+0.40729		0.000070	0.01565
402	122	+0.00514	-0.02588	-0.01302	-0.00770	-0.03882	+0.00868	+0.04666	-0.00926	+0.00000		+0.05466	+0.35801	+0.42056		0.005222	0.04757
403	122	-0.00613	-0.00794	-0.01302	+0.00920	-0.01191	+0.00868	+0.01296	+0.01255	+0.00000		+0.05466	+0.35801	+0.42056		0.000835	0.01905
404	122	-0.00866	-0.00607	-0.01302	+0.01299	-0.00911	+0.00868	-0.00614	-0.01747	+0.00000		+0.09023	+0.30466	+0.41094		0.000835	0.02119
405	124	+0.00000	+0.00491	-0.01302	+0.00000	-0.00736	+0.00868	+0.00000	-0.00133	+0.00000		+0.05572	+0.35642	+0.40144		0.000070	0.01565
406	124	+0.00514	-0.02588	-0.01302	-0.00770	-0.03882	+0.00868	+0.04666	-0.00926	+0.00000		+0.00666	+0.43001	+0.41471		0.005222	0.04757
407	124	-0.00613	-0.00794	-0.01302	+0.00920	-0.01191	+0.00868	+0.01296	+0.01255	+0.00000		+0.00666	+0.43001	+0.41471		0.000835	0.01905
408	124	-0.00866	-0.00607	-0.01302	+0.01299	-0.00911	+0.00868	-0.00614	-0.01747	+0.00000		+0.04223	+0.37666	+0.40509		0.000835	0.02119
409	126	+0.00000	-0.00491	-0.01302	+0.00000	-0.00736	+0.00868	+0.00000	-0.00133	+0.00000		+0.00772	+0.42842	+0.39317		0.000070	0.01565
410	126	+0.00514	-0.02588	-0.01302	-0.00770	-0.03882	+0.00868	+0.04666	-0.00926	+0.00000		-0.04134	+0.50201	+0.40644		0.005222	0.04757
411	126	-0.00613	-0.00794	-0.01302	+0.00920	-0.01191	+0.00868	+0.01296	+0.01255	+0.00000		-0.04134	+0.50201	+0.40644		0.000835	0.01905
412	126	-0.00866	-0.00607	-0.01302	+0.01299	-0.00911	+0.00868	-0.00614	-0.01747	+0.00000		-0.00577	+0.44866	+0.39362		0.000835	0.02119

Muzzle — 60 maps (15 positions × 4)

#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33		b1	b2	b3		p	sigma1
413	128	+0.00000	+0.00000	-0.01806	+0.00000	-0.01001	+0.00000	+0.00000	-0.00250	+0.00000		+0.00000	+0.53873	+0.34014		0.000070	0.01806
414	128	+0.00000	+0.00000	-0.01806	-0.01580	-0.05256	+0.00000	+0.05256	-0.01580	+0.00000		+0.00000	+0.63885	+0.36517		0.005222	0.05489
415	128	+0.00000	+0.00000	-0.01806	+0.01126	-0.01783	+0.00000	+0.01611	+0.01283	+0.00000		+0.00000	+0.63885	+0.36517		0.000835	0.02198
416	128	+0.00000	+0.00000	-0.01806	+0.01862	-0.01064	+0.00000	-0.00532	-0.02128	+0.00000		+0.00000	+0.56626	+0.34702		0.000835	0.02446
417	130	+0.00000	+0.00834	-0.01040	+0.00000	-0.00588	-0.01477	+0.00000	-0.00153	+0.00000		-0.08766	+0.64146	+0.28458		0.000070	0.01806
418	130	+0.00874	+0.04401	-0.01040	-0.00615	-0.03100	-0.01477	+0.05384	-0.01068	+0.00000		-0.17110	+0.70023	+0.29989		0.005222	0.05489
419	130	-0.01043	+0.01351	-0.01040	+0.00735	-0.00951	-0.01477	+0.01496	+0.01448	+0.00000		-0.17110	+0.70023	+0.29989		0.000835	0.02198
420	130	-0.01473	+0.01033	-0.01040	+0.01038	-0.00727	-0.01477	-0.00708	-0.02016	+0.00000		-0.11061	+0.65762	+0.28879		0.000835	0.02446
421	132	+0.00000	+0.01006	-0.00299	+0.00000	+0.00169	-0.01781	+0.00000	-0.00153	+0.00000		-0.20349	+0.66078	+0.27585		0.000070	0.01806
422	132	+0.01054	+0.05309	+0.00299	+0.00177	+0.00892	-0.01781	+0.05384	-0.01068	+0.00000		-0.30413	+0.64387	+0.29116		0.005222	0.05489
423	132	-0.01258	+0.01629	+0.00299	-0.00211	+0.00274	-0.01781	+0.01496	+0.01448	+0.00000		-0.30413	+0.64387	+0.29116		0.000835	0.02198
424	132	-0.01777	+0.01245	+0.00299	-0.00299	+0.00209	-0.01781	-0.00708	-0.02016	+0.00000		-0.23116	+0.65613	+0.28006		0.000835	0.02446
425	134	+0.00000	-0.00624	-0.01429	-0.00000	-0.00807	+0.01105	+0.00000	-0.00153	+0.00000		+0.03679	+0.59519	+0.29311		0.000070	0.01806
426	134	-0.00653	-0.03293	-0.01429	-0.00845	-0.04259	+0.01105	+0.05384	-0.01068	+0.00000		+0.09921	+0.67594	+0.30842		0.005222	0.05489
427	134	+0.00780	-0.01010	-0.01429	+0.01009	-0.01307	+0.01105	+0.01496	+0.01448	+0.00000		+0.09921	+0.67594	+0.30842		0.000835	0.02198
428	134	+0.01102	-0.00772	-0.01429	+0.01426	-0.00999	+0.01105	-0.00708	-0.02016	+0.00000		+0.05396	+0.61740	+0.29732		0.000835	0.02446
429	136	+0.00000	-0.00995	-0.00399	+0.00000	-0.00226	+0.01761	+0.00000	-0.00153	-0.00000		+0.14352	+0.66301	+0.27872		0.000070	0.01806
430	136	-0.01042	-0.05250	-0.00399	-0.00236	-0.01191	+0.01761	+0.05384	-0.01068	-0.00000		+0.24305	+0.68558	+0.29403		0.005222	0.05489
431	136	+0.01244	-0.01611	-0.00399	+0.00282	-0.00365	+0.01761	+0.01496	+0.01448	-0.00000		+0.24305	+0.68558	+0.29403		0.000835	0.02198
432	136	+0.01757	-0.01232	-0.00399	+0.00399	-0.00279	+0.01761	-0.00708	-0.02016	-0.00000		+0.17089	+0.66921	+0.28293		0.000835	0.02446
433	138	+0.00000	-0.00913	+0.00808	+0.00000	+0.00456	+0.01615	+0.00000	-0.00153	-0.00000		+0.26796	+0.63602	+0.27575		0.000070	0.01806
434	138	-0.00956	-0.04815	+0.00808	+0.00478	+0.02408	+0.01615	+0.05384	-0.01068	-0.00000		+0.35924	+0.59038	+0.29106		0.005222	0.05489
435	138	+0.01141	-0.01478	+0.00808	-0.00571	+0.00739	+0.01615	+0.01496	+0.01448	-0.00000		+0.35924	+0.59038	+0.29106		0.000835	0.02198
436	138	+0.01612	-0.01138	+0.00808	-0.00806	+0.00565	+0.01615	-0.00708	-0.02016	-0.00000		+0.29306	+0.62347	+0.27996		0.000835	0.02446
437	140	+0.00000	+0.00819	+0.01086	+0.00000	+0.00616	-0.01443	+0.00000	-0.00123	+0.00000		-0.22159	+0.39749	+0.34956		0.000070	0.01806
438	140	+0.00727	+0.04325	+0.01086	+0.00547	+0.03255	-0.01443	+0.05413	-0.00909	+0.00000		-0.30346	+0.33587	+0.36186		0.005222	0.05489
439	140	+0.01054	+0.01285	+0.01086	+0.00793	+0.00967	-0.01443	+0.01458	+0.01496	+0.00000		-0.30346	+0.33587	+0.36186		0.000835	0.02198
440	140	-0.01422	+0.01056	+0.01086	-0.01071	+0.00795	-0.01443	-0.00761	-0.01978	+0.00000		-0.24410	+0.38055	+0.35295		0.000835	0.02446
441	142	+0.00000	+0.00963	-0.00619	+0.00000	-0.00351	-0.01697	+0.00000	-0.00123	-0.00000		-0.46340	+0.37914	+0.32909		0.000070	0.01806
442	142	+0.00854	+0.05085	-0.00619	-0.00312	-0.01854	-0.01697	+0.05413	-0.00909	-0.00000		+0.55966	+0.41424	+0.34139		0.005222	0.05489

443	142	-0.01239	+0.01511	-0.00619	+0.00452	-0.00551	-0.01697	+0.01458	+0.01496	-0.00000	-0.55966	+0.41424	+0.34139	0.000835	0.02198
444	142	-0.01673	+0.01242	-0.00619	+0.00610	-0.00453	-0.01697	-0.00761	-0.01978	-0.00000	-0.48987	+0.38879	+0.33247	0.000835	0.02446
445	144	+0.00000	-0.00512	-0.01565	+0.00000	-0.00888	+0.00902	+0.00000	-0.00123	+0.00000	-0.53408	+0.52878	+0.29687	0.000070	0.01806
446	144	-0.00454	-0.02703	-0.01565	-0.00788	-0.04690	+0.00902	+0.05413	-0.00909	+0.00000	-0.48292	+0.61756	+0.30917	0.005222	0.05489
447	144	+0.00659	-0.00803	-0.01565	+0.01143	-0.01394	+0.00902	+0.01458	+0.01496	+0.00000	-0.48292	+0.61756	+0.30917	0.000835	0.02198
448	144	+0.00889	-0.00660	-0.01565	+0.01543	-0.01145	+0.00902	-0.00761	-0.01978	+0.00000	-0.52001	+0.55320	+0.30025	0.000835	0.02446
449	146	+0.00000	-0.01018	+0.00200	+0.00000	+0.00113	+0.01795	+0.00000	-0.00123	-0.00000	-0.31813	+0.59719	+0.31168	0.000070	0.01806
450	146	-0.00904	-0.05379	+0.00200	+0.00101	+0.00599	+0.01795	+0.05413	-0.00909	-0.00000	-0.21629	+0.58585	+0.32398	0.005222	0.05489
451	146	+0.01311	-0.01599	+0.00200	-0.00146	+0.00178	+0.01795	+0.01458	+0.01496	-0.00000	-0.21629	+0.58585	+0.32398	0.000835	0.02198
452	146	+0.01769	-0.01314	+0.00200	-0.00197	+0.00146	+0.01795	-0.00761	-0.01978	-0.00000	-0.29012	+0.59407	+0.31506	0.000835	0.02446
453	148	+0.00000	+0.00000	+0.01806	+0.00000	+0.01025	+0.00000	+0.00000	-0.00123	+0.00000	+0.55000	+0.47106	+0.30128	0.000070	0.01806
454	148	+0.00000	+0.00000	+0.01806	+0.00909	+0.05413	+0.00000	+0.05413	-0.00909	+0.00000	+0.55000	+0.36860	+0.31358	0.005222	0.05489
455	148	+0.00000	+0.00000	+0.01806	-0.01319	+0.01609	+0.00000	+0.01458	+0.01496	+0.00000	+0.55000	+0.36860	+0.31358	0.000835	0.02198
456	148	+0.00000	+0.00000	+0.01806	-0.01780	+0.01322	+0.00000	-0.00761	-0.01978	+0.00000	+0.55000	+0.44289	+0.30466	0.000835	0.02446
457	150	+0.00000	+0.01018	+0.00200	+0.00000	+0.00113	-0.01795	+0.00000	-0.00123	+0.00000	+0.38411	+0.36083	+0.33837	0.000070	0.01806
458	150	+0.00904	-0.05379	+0.00200	+0.00101	+0.00599	-0.01795	+0.05413	-0.00909	+0.00000	+0.28228	+0.34949	+0.35067	0.005222	0.05489
459	150	-0.01311	-0.01599	+0.00200	-0.00146	+0.00178	-0.01795	+0.01458	+0.01496	+0.00000	+0.28228	+0.34949	+0.35067	0.000835	0.02198
460	150	-0.01769	-0.01314	+0.00200	-0.00197	+0.00146	-0.01795	-0.00761	-0.01978	+0.00000	+0.35611	+0.35772	+0.34175	0.000835	0.02446
461	152	+0.00000	+0.00512	-0.01565	+0.00000	-0.00888	-0.00902	+0.00000	-0.00123	+0.00000	+0.17700	+0.44670	+0.34706	0.000070	0.01806
462	152	+0.00454	-0.02703	-0.01565	-0.00788	-0.04690	-0.00902	+0.05413	-0.00909	+0.00000	+0.12584	+0.53548	+0.35936	0.005222	0.05489
463	152	-0.00659	+0.00803	-0.01565	+0.01143	-0.01394	-0.00902	+0.01458	+0.01496	+0.00000	+0.12584	+0.53548	+0.35936	0.000835	0.02198
464	152	-0.00889	-0.00660	-0.01565	+0.01543	-0.01145	-0.00902	-0.00761	-0.01978	+0.00000	+0.16293	+0.47111	+0.35044	0.000835	0.02446
465	154	+0.00000	-0.00963	-0.00619	+0.00000	-0.00351	+0.01697	+0.00000	-0.00123	+0.00000	+0.27340	+0.58698	+0.31975	0.000070	0.01806
466	154	-0.00854	-0.05085	-0.00619	-0.00312	-0.01854	+0.01697	+0.05413	-0.00909	+0.00000	+0.36966	+0.62209	+0.33205	0.005222	0.05489
467	154	+0.01239	-0.01511	-0.00619	+0.00452	-0.00551	+0.01697	+0.01458	+0.01496	+0.00000	+0.36966	+0.62209	+0.33205	0.000835	0.02198
468	154	+0.01673	-0.01242	-0.00619	+0.00610	-0.00453	+0.01697	-0.00761	-0.01978	+0.00000	+0.29987	+0.59664	+0.32313	0.000835	0.02446
469	156	+0.00000	-0.00819	+0.01086	+0.00000	+0.00616	+0.01443	+0.00000	-0.00123	+0.00000	+0.51269	+0.55176	+0.29618	0.000070	0.01806
470	156	-0.00727	-0.04325	+0.01086	+0.00547	+0.03255	+0.01443	+0.05413	-0.00909	+0.00000	+0.59455	+0.49014	+0.30847	0.005222	0.05489
471	156	+0.01054	-0.01285	+0.01086	-0.00793	+0.00967	+0.01443	+0.01458	+0.01496	+0.00000	+0.59455	+0.49014	+0.30847	0.000835	0.02198
472	156	+0.01422	-0.01056	+0.01086	-0.01071	+0.00795	+0.01443	-0.00761	-0.01978	+0.00000	+0.53520	+0.53482	+0.29956	0.000835	0.02446

Whiskers — 48 maps (12 positions × 4)

	#	pos	a11	a12	a13	a21	a22	a23	a31	a32	a33	b1	b2	b3	p	sigma1
261	52	+0.00000	+0.03358	+0.00773	+0.00000	+0.00435	-0.05970	+0.00000	-0.00609	+0.00000	-0.73948	+0.34381	+0.06531	0.000070	0.06020	
262	52	+0.04050	+0.17686	+0.00773	+0.00524	+0.02290	-0.05970	+0.17834	-0.04084	+0.00000	-1.07524	+0.30033	+0.12626	0.005222	0.18295	
263	52	-0.04071	-0.05599	+0.00773	-0.00527	+0.00725	-0.05970	+0.05108	+0.04664	+0.00000	-1.07524	+0.30033	+0.12626	0.000835	0.07325	
264	52	-0.06023	-0.03979	+0.00773	-0.00780	+0.00515	-0.05970	-0.02184	-0.06839	+0.00000	-0.83181	+0.33185	+0.08207	0.000835	0.08152	
265	54	+0.00000	+0.03358	+0.00773	+0.00000	+0.00435	-0.05970	+0.00000	-0.00609	+0.00000	-1.49935	+0.24541	+0.06531	0.000070	0.06020	
266	54	+0.04050	+0.17686	+0.00773	+0.00524	+0.02290	-0.05970	+0.17834	-0.04084	+0.00000	-1.83510	+0.20193	+0.12626	0.005222	0.18295	
267	54	-0.04071	-0.05599	+0.00773	-0.00527	+0.00725	-0.05970	+0.05108	+0.04664	+0.00000	-1.83510	+0.20193	+0.12626	0.000835	0.07325	
268	54	-0.06023	-0.03979	+0.00773	-0.00780	+0.00515	-0.05970	-0.02184	-0.06839	+0.00000	-1.59168	+0.23345	+0.08207	0.000835	0.08152	
269	56	+0.00000	+0.03385	+0.00087	+0.00000	+0.00049	-0.06019	+0.00000	-0.00609	+0.00000	-0.73972	+0.47598	+0.06531	0.000070	0.06020	
270	56	+0.04083	+0.17832	+0.00087	+0.00059	+0.00257	-0.06019	+0.17834	-0.04084	+0.00000	-1.07825	+0.47110	+0.12626	0.005222	0.18295	
271	56	-0.04105	+0.05645	+0.00087	-0.00059	+0.00081	-0.06019	+0.05108	+0.04664	+0.00000	-1.07825	+0.47110	+0.12626	0.000835	0.07325	
272	56	-0.06072	+0.04012	+0.00087	-0.00087	+0.00058	-0.06019	-0.02184	-0.06839	+0.00000	-0.83282	+0.47464	+0.08207	0.000835	0.08152	
273	58	+0.00000	+0.03385	+0.00087	+0.00000	+0.00049	-0.06019	+0.00000	-0.00609	+0.00000	-1.49959	+0.46504	+0.06531	0.000070	0.06020	
274	58	+0.04083	+0.17832	+0.00087	+0.00059	+0.00257	-0.06019	+0.17834	-0.04084	+0.00000	-1.83811	+0.46017	+0.12626	0.005222	0.18295	
275	58	-0.04105	+0.05645	+0.00087	-0.00059	+0.00081	-0.06019	+0.05108	+0.04664	+0.00000	-1.83811	+0.46017	+0.12626	0.000835	0.07325	
276	58	-0.06072	+0.04012	+0.00087	-0.00087	+0.00058	-0.06019	-0.02184	-0.06839	+0.00000	-1.59268	+0.46370	+0.08207	0.000835	0.08152	
277	60	+0.00000	+0.03373	-0.00518	+0.00000	-0.00291	-0.05998	+0.00000	-0.00609	+0.00000	-0.73962	+0.60414	+0.06531	0.000070	0.06020	
278	60	+0.04069	+0.17768	-0.00518	-0.00351	-0.01534	-0.05998	+0.17834	-0.04084	+0.00000	-1.07692	+0.63326	+0.12626	0.005222	0.18295	
279	60	-0.04090	+0.05625	-0.00518	+0.00353	-0.00486	-0.05998	+0.05108	+0.04664	+0.00000	-1.07692	+0.63326	+0.12626	0.000835	0.07325	
280	60	-0.06050	+0.03997	-0.00518	+0.00522	-0.00345	-0.05998	-0.02184	-0.06839	+0.00000	-0.83237	+0.61215	+0.08207	0.000835	0.08152	
281	62	+0.00000	+0.03373	-0.00518	+0.00000	-0.00291	-0.05998	+0.00000	-0.00609	+0.00000	-1.49948	+0.66974	+0.06531	0.000070	0.06020	
282	62	+0.04069	+0.17768	-0.00518	-0.00351	-0.01534	-0.05998	+0.17834	-0.04084	+0.00000	-1.83679	+0.69886	+0.12626	0.005222	0.18295	
283	62	-0.04090	+0.05625	-0.00518	+0.00353	-0.00486	-0.05998	+0.05108	+0.04664	+0.00000	-1.83679	+0.69886	+0.12626	0.000835	0.07325	
284	62	-0.06050	+0.03997	-0.00518	+0.00522	-0.00345	-0.05998	-0.02184	-0.06839	+0.00000	-1.59224	+0.67775	+0.08207	0.000835	0.08152	
285	64	+0.00000	+0.03358	+0.00773	+0.00000	+0.00435	+0.05970	+0.00000	-0.00609	+0.00000	+0.73948	+0.34381	+0.06531	0.000070	0.06020	
286	64	-0.04050	+0.17686	+0.00773	+0.00524	+0.02290	+0.05970	+0.17834	-0.04084	-0.00000	+1.07524	+0.30033	+0.12626	0.005222	0.18295	
287	64	-0.04071	-0.05599	+0.00773	-0.00527	+0.00725	+0.05970	+0.05108	+0.04664	-0.00000	+1.07524	+0.30033	+0.12626	0.000835	0.07325	
288	64	+0.06023	-0.03979	+0.00773	-0.00780	+0.00515	+0.05970	-0.02184	-0.06839	-0.00000	+0.83181	+0.33185	+0.08207	0.000835	0.08152	
289	66	+0.00000	-0.03358	+0.00773	+0.00000	+0.00435	+0.05970	+0.00000	-0.00609	-0.00000	+1.49935	+0.24541	+0.06531	0.000070	0.06020	
290	66	-0.04050	+0.17686	+0.00773	+0.00524	+0.02290	+0.05970	+0.17834	-0.04084	-0.00000	+1.83510	+0.20193	+0.12626	0.005222	0.18295	
291	66	+0.04071	-0.05599	+0.00773	-0.00527	+0.00725	+0.05970	+0.05108	+0.04664	-0.00000	+1.83510	+0.20193	+0.12626	0.000835	0.07325	
292	66	+0.06023	-0.03979	+0.00773	-0.00780	+0.00515	+0.05970	-0.02184	-0.06839	-0.00000	+1.59168	+0.23345	+0.08207	0.000835	0.08152	
293	68	+0.00000	+0.03385	+0.00087	+0.00000	+0.00049	-0.06019	+0.00000	-0.00609	-0.00000	+0.73972	+0.47598	+0.06531	0.000070	0.06020	
294	68	-0.04083	+0.17832	+0.00087	+0.00059	+0.00257	-0.06019	+0.17834	-0.04084	-0.00000	+1.07825	+0.47110	+0.12626	0.005222	0.18295	
295	68	+0.04105	+0.05645	+0.00087	-0.00059	+0.00081	-0.06019	+0.05108	+0.04664	-0.00000	+1.07825	+0.47110	+0.12626	0.000835	0.07325	
296	68	-0.06072	+0.04012	+0.00087	-0.00087	+0.00058	-0.06019	-0.02184	-0.06839	-0.00000	+0.83282	+0.47464	+0.08207	0.000835	0.08152	
297	70	+0.00000	-0.03385	+0.00087	+0.00000	+0.00049	-0.06019	+0.00000	-0.00609	-0.00000	+1.49959	+0.46504	+0.06531	0.000070	0.06020	
298	70	-0.04083	-0.17832	+0.00087	+0.00059	+0.00257	-0.06019	+0.17834	-0.04084	-0.00000	+1.83811	+0.46017	+0.12626	0.005222	0.18295	
299	70	+0.04105	+0.05645	+0.00087	-0.00059	+0.00081	-0.06019	+0.05108	+0.04664	-0.00000	+1.83811	+0.46017	+0.12626	0.000835	0.07325	
300	70	+0.06072	-0.04012	+0.00087	-0.00087	+0.00058	-0.06019	-0.02184	-0.06839	-0.00000	+1.59268	+0.46370	+0.08207	0.000835	0.08152	
301	72	+0.00000	-0.03373	-0.00518	+0.00000	-0.00291	-0.05998	+0.00000	-0.00609	+0.00000	-0.73962	+0.60414	+0.06531	0.000070	0.06020	
302	72	-0.04069	-0.17768	-0.00518	-0.00351	-0.01534	-0.05998	+0.17834	-0.04084	+0.00000	+1.07692	+0.63326	+0.12626	0.005222	0.18295	
303	72	+0.04090	-0.05625	-0.00518	+0.00353	-0.00486	+0.05998	+0.05108	+0.04664	+0.00000	+1.07692	+0.63326	+0.12626	0.000835	0.07325	
304	72	-0.06050	-0.03997	-0.00518	+0.00522	-0.00345	+0.05998	-0.02184	-0.06839	+0.00000	+0.83237	+0.61215	+0.08207	0.000835	0.08152	
305	74	+0.00000	-0.03373	-0.00518	+0.00000	-0.00291	-0.05998	+0.00000	-0.00609	+0.00000	-1.49948	+0.66974	+0.06531	0.000070	0.06020	
306	74	-0.04069	-0.17768	-0.00518	-0.00351	-0.01534	-0.05998	+0.17834	-0.04084	+0.00000	+1.83679	+0.69886	+0.12626	0.005222	0.18295	
307	74	+0.04090	-0.05625	-0.00518	+0.00353	-0.00486	+0.05998	+0.05108	+0.04664	+0.00000	+1.83679	+0.69886	+0.12626	0.000835	0.07325	
308	74	-0.06050	-0.03997	-0.00518	+0.00522	-0.00345	-0.05998	-0.02184	-0.06839	+0.00000	-1.59224	+0.67775	+0.08207	0.000835	0.08152	

Appendix B. Claude's Cat: Generating a Fractal with a Trained Model

claudes_cat_sds.py: score distillation from a frozen diffusion model onto the coefficients of an IFS

Written by Claude (Claude Opus 5, Anthropic) under the supervision of Dominic Rochon, Ph.D., Fundamental Mathematics · Version 2.4.2, 12 October 2026

Purpose

This script accompanies section 11.1 of the main document, which states that a trained model can generate a mathematical fractal in the strict sense. The optimisation variables are the coefficients of N contractive affine maps of $\mathbb{R}^3$. At each step the maps are rendered into an image, a diffusion model evaluates the image against a text prompt, and the resulting gradient is transferred to the coefficients through the differentiable renderer. The diffusion model receives only images; the output of the script is a list of matrices, which defines an iterated function system.

The script does not reproduce Claude's Cat. That object was constructed map by map (Appendix A), whereas the script searches for a shape by optimisation. The two approaches are not expected to give the same attractor; comparing their depth of self-similarity is the experiment proposed in section 10.

Contents

Part	Contents	Role
1. IFS3D	The N maps as trainable parameters: two rotations, three singular values passed through a sigmoid scaled by $s_{\max}$, a translation and a weight.	The largest singular value is below $s_{\max} < 1$ for all parameter values, so every candidate is a contraction and Hutchinson's theorem [B1] applies. No penalty term or projection is needed.
2. render, splat	Chaos game with the random index sequence fixed during a step, pinhole camera, bilinear splatting, Gaussian blur and logarithmic tone mapping.	With the indices fixed, the image depends continuously on the coefficients, so gradients reach A and b. The blur widens the support of these gradients; without it the loss is nearly flat.
3. SDS, CLIPObjective	Score distillation with a fixed Stable Diffusion U-Net [B2], or, as a cheaper alternative, cosine distance to a CLIP text embedding over random crops.	Provides the evaluation of an attractor against the prompt, which the explicit construction of Appendix A did not have.
4. main	Adam optimiser, gradient clipping, an entropy term, annealing of the contraction bound and of the blur, then two assertions and a JSON export.	The output is written only if $\sigma_{\max} < 1$ and the probabilities sum to 1.

Expected results

Fitting an IFS to a target is the inverse problem left open by the collage theorem [B3], and the loss is non-convex with many poor local minima. A successful run should produce a coherent branching structure whose overall arrangement corresponds to the prompt, not a detailed illustration. Several random seeds should be tried and the best result kept; many runs are likely to be unusable.

The preliminary experiments for this project showed the difficulty of choosing the objective. An **L2 fit** [B4] scores a candidate by the sum, over all pixels, of the squared difference between its value and that of the target. The name refers to the **ℓ^2 norm**, the ordinary Euclidean length: if the two images are regarded as vectors holding one number per pixel, the length of their difference is the square root of that sum, so minimising the L2 fit means making the difference vector as short as possible in the Euclidean sense. It is the default measure of disagreement between two images, and it treats every pixel independently and symmetrically — which is exactly where the difficulty lies.

On this target the L2 fit was outperformed by an entirely black image. The target was a line drawing and therefore mostly empty. An empty render matches the empty regions and is penalised only on the thin lines it misses, whereas a render that attempts the shape is penalised both for missing lines and for ink placed in the wrong position. The objective was replaced by one that measures the fraction of the target that is covered, and useful results still required several restarts. This does not mean that L2 is a poor choice in general, but on a sparse target it favours an empty image. The weakness of squared error as a measure of image similarity is discussed in detail in [B5].

Typical failure modes:

Symptom	Likely cause	Remedy
The mass concentrates on a few points	One map has captured the measure	Increase the entropy weight (0.01 in the code) or decrease --lr
A faint uniform haze without structure	Contraction bound too high or blur too wide	Lower --s-hi towards 0.8; lower --sigma-hi

Sharp texture without meaning	The objective is being exploited	With CLIP, increase n_aug; with SDS, reduce the guidance scale (100)
No change during optimisation	Gradients do not reach the coefficients	Increase --sigma-hi; check that --points is large enough for the image not to be mostly empty

Usage

A GPU is needed in practice for the SDS objective, which evaluates a U-Net at each step. About 8 GB of video memory is sufficient at the default resolution. The CLIP objective is about ten times cheaper and runs on a laptop, but gives a weaker evaluation. Installation and examples:

```
python3 -m venv .venv && source .venv/bin/activate
pip install torch diffusers transformers accelerate
pip install open_clip_torch # only for --objective clip

python claudes_cat_sds.py --prompt "a cat drawn in fern fronds" --maps 24
python claudes_cat_sds.py --prompt "a coiled nautilus of ferns" \
--objective clip --iters 800 --seed 3
```

Approximate timings on a recent GPU: about one third of a second per SDS iteration, or about ten minutes for the default 1500 iterations; a few minutes for the CLIP objective. On Apple silicon, use --objective clip, and fall back to the CPU if the MPS backend does not support index_add or multinomial.

Option	Default	Effect
--prompt	(required)	Text against which the diffusion model evaluates the images
--objective	sds	sds for the diffusion model, clip for the cheaper alternative
--maps	24	Number of maps. Fewer maps allow deeper self-similarity; more maps allow more control of the shape (see section 9 of the main document)
--iters	1500	Number of optimisation steps
--s-lo / --s-hi	0.55 / 0.90	Contraction bound, increased from the first to the second value; the upper value limits the depth of recursion
--points / --chain	120000 / 14	Points per render and length of the chaos-game chain; increase the chain if the measure has not converged
--seed	0	Random seed; results vary strongly with it
--out	fractal.json	Output file for the coefficients

Output

The output is a JSON file with the same convention as Appendix A: $f_i(x) = A_i x + b_i$, with a probability and the singular values of each map, so that it can be read by the renderer used for the figures of the main document. The file is written only if both assertions succeed, and it records sigma_max_overall so that the check can be repeated. If this value is below 1, the file defines an attractor in the strict sense, independently of the intermediate images.

Status

The script has not been executed. The environment in which this document was prepared had no GPU and no network access to download model weights, which is also why the construction of Appendix A was carried out explicitly. The renderer at the core of the script has been checked in NumPy: forward splatting of the Hutchinson operator conserves mass exactly (1.000000) and agrees with the chaos game with a correlation of 0.998, including for maps with a singular value as small as 0.07. The contraction parameterisation and the final check are exact by construction and are tested at run time. The score-distillation part should be regarded as a specification to be run, not as a reported result.

References for this appendix

- [B1] J. E. Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981. doi:10.1512/iumj.1981.30.30055
- [B2] B. Poole, A. Jain, J. T. Barron and B. Mildenhall. DreamFusion: text-to-3D using 2D diffusion. *ICLR*, 2023. (The source of score distillation sampling.)
- [B3] M. F. Barnsley, V. Ervin, D. Hardin and J. Lancaster. Solution of an inverse problem for fractals and other sets. *Proceedings of the National Academy of Sciences USA*, 83:1975–1977, 1986. (The collage theorem, and the inverse problem it leaves open.)
- [B4] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning*. MIT Press, 2016, ch. 5. (Squared-error loss, written L2: the sum over pixels of the squared difference between a candidate and its target, that is the squared L^2 norm of the difference image.)
- [B5] Z. Wang and A. C. Bovik. Mean squared error: love it or leave it? A new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009. (Why squared error, despite being the default choice for fifty years, corresponds poorly to how similar two images actually look.)

These references are numbered separately so that this appendix can be read on its own. Hutchinson, Poole et al. and Barnsley et al. also appear in the main document's reference list, but under different numbers, since that list is sorted alphabetically. Cite by author and year when referring across documents.

```
#!/usr/bin/env python3
"""
claudes_cat_sds.py -- Generating a mathematical fractal from a text prompt
using a trained diffusion model.

Purpose
-----
This script supports the claim made in the conclusion of "Claude's Cat": that an
AI-generated mathematical fractal is achievable in every sense of the term --
not only with the AI writing the geometry, but with a *trained* model choosing
the maps.

The object optimised here is never an image. It is a list of N contractive
affine maps of  $\mathbb{R}^3$ . The image is only the means by which a diffusion model is
allowed to express an opinion about those maps. Whatever the optimiser
converges to is therefore an iterated function system, and its attractor is a
mathematical fractal in the strict sense, because:

* contractivity is structural, not a penalty (see IFS3D.matrices below), so
  every candidate visited during optimisation satisfies the single hypothesis
  of Hutchinson's theorem;
* the output is the coefficient list itself, which can be evaluated at any
  depth, long after the 512x512 image used to train it has been discarded.

That is the difference from text-to-image generation: a diffusion model here
produces a *rule*, not pixels.

Method
-----
Score Distillation Sampling (Poole et al., DreamFusion, ICLR 2023) provides a
gradient from a frozen diffusion model to any differentiable parameterisation.
The differentiable parameterisation used here is a chaos-game renderer:


$$x_{k+1} = A_{i_k} x_k + b_{i_k}$$


With the index sequence  $\{i_k\}$  fixed for a given step, the map coefficients
enter continuously, so pathwise gradients reach A and b directly. Points are
projected with a pinhole camera and splatted bilinearly, which is differentiable
with respect to position. Probabilities receive a gradient through a mass
reweighting  $p/\text{detach}(p)$ .

Requirements
-----
    pip install torch diffusers transformers accelerate
    (optional, for the cheaper CLIP objective) pip install open_clip_torch

A GPU is effectively required: SDS evaluates a U-Net every step.

Status
-----
NOT EXECUTED as written. The environment in which "Claude's Cat" was produced
had neither PyTorch, a GPU, nor network access to fetch model weights -- which
is precisely why that object was hand-designed. What *has* been verified
numerically, in NumPy, is the renderer at the heart of this script: forward
splating of the Hutchinson operator conserves mass exactly (1.000000) and
agrees with the chaos game to a correlation of 0.998, including for maps with a
singular value as small as 0.07. The contractivity parameterisation and the
final SVD audit are likewise exact by construction and are asserted at runtime.
Treat the SDS plumbing as a specification to be run, not as a reported result.

Usage
-----
    python claudes_cat_sds.py --prompt "a cat drawn in fern fronds" --maps 24
    python claudes_cat_sds.py --prompt "a coiled nautilus of ferns" --objective clip
"""

from __future__ import annotations

import argparse
import json
import math

import torch
import torch.nn as nn
import torch.nn.functional as F

# -----
# 1. The object being optimised: N contractive affine maps of  $\mathbb{R}^3$ 
# -----
class IFS3D(nn.Module):
    """Contractivity is structural.
```

Each matrix is built as $A = U \text{diag}(s) V$ with U, V rotations and $0 < s_k < s_{\max} < 1$, so the largest singular value is $\max(s) < s_{\max}$ for every parameter value the optimiser can reach. No projection step, no penalty term, and no possibility of visiting a divergent system.

```
def __init__(self, n_maps: int, s_max: float = 0.85, seed: int = 0):
    super().__init__()
    g = torch.Generator().manual_seed(seed)
    self.n, self.s_max = n_maps, s_max
    # three angles per rotation, three singular values, translation, weight
    self.rotU = nn.Parameter(torch.rand(n_maps, 3, generator=g) * 2 * math.pi)
    self.rotV = nn.Parameter(torch.rand(n_maps, 3, generator=g) * 2 * math.pi)
    self.raw_s = nn.Parameter(torch.randn(n_maps, 3, generator=g) * 0.3)
    self.b = nn.Parameter((torch.rand(n_maps, 3, generator=g) * 2 - 1) * 0.6)
    self.logit_p = nn.Parameter(torch.zeros(n_maps))
    # fixed reflection signs: restores improper similarities without ever
    # making A singular
    self.register_buffer(
        "flip", torch.where(torch.rand(n_maps, generator=g) < 0.5, -1.0, 1.0))
```

```
@staticmethod
def _rot(a):
    # (n,3) Euler -> (n,3,3)
    ca, sa = torch.cos(a), torch.sin(a)
    z, o = torch.zeros_like(ca[:, 0]), torch.ones_like(ca[:, 0])
    Rx = torch.stack([torch.stack([o, z, z], -1),
                      torch.stack([z, ca[:, 0], -sa[:, 0]], -1),
                      torch.stack([z, sa[:, 0], ca[:, 0]], -1)], -2)
    Ry = torch.stack([torch.stack([ca[:, 1], z, sa[:, 1]], -1),
                      torch.stack([z, o, z], -1),
                      torch.stack([-sa[:, 1], z, ca[:, 1]], -1)], -2)
    Rz = torch.stack([torch.stack([ca[:, 2], -sa[:, 2], z], -1),
                      torch.stack([sa[:, 2], ca[:, 2], z], -1),
                      torch.stack([z, z, o], -1)], -2)
    return Rx @ Ry @ Rz
```

```
def matrices(self):
    s = self.s_max * torch.sigmoid(self.raw_s) # (n,3) in (0,s_max)
    s = s * torch.stack([torch.ones_like(s[:, 0]),
                        torch.ones_like(s[:, 1]), self.flip], -1)
    D = torch.diag_embed(s)
    A = self._rot(self.rotU) @ D @ self._rot(self.rotV)
    return A, self.b, torch.softmax(self.logit_p, 0)
```

```
@torch.no_grad()
def audit(self):
    """The claim this whole script exists to support, checked numerically."""
    A, _, p = self.matrices()
    sv = torch.linalg.svdvals(A)
    return float(sv[:, 0].max()), float(p.sum())
```

```
@torch.no_grad()
def to_json(self, path, prompt):
    A, b, p = self.matrices()
    sv = torch.linalg.svdvals(A)
    json.dump({
        "prompt": prompt,
        "n_maps": self.n,
        "convention": "f_i(x) = A_i x + b_i, x in R^3",
        "sigma_max_overall": float(sv[:, 0].max()),
        "maps": [{"A": A[i].tolist(), "b": b[i].tolist(),
                    "p": float(p[i]), "sigma": sv[i].tolist()}
                 for i in range(self.n)],
    }, open(path, "w"), indent=1)
```

```
# -----
# 2. Differentiable renderer: chaos game + pinhole camera + bilinear splat
# -----
```

```
def splat(uv, w, res):
    """Bilinear scatter. Differentiable in uv (through the weights) and in w.
```

Forward splatting rather than a backward gather: it conserves mass exactly and tolerates rank-deficient maps. Verified in NumPy against the chaos game.

```
"""
fu = (uv[:, 0] + 1) * 0.5 * res - 0.5
fv = (uv[:, 1] + 1) * 0.5 * res - 0.5
u0, v0 = torch.floor(fu).detach(), torch.floor(fv).detach()
du, dv = fu - u0, fv - v0
u0, v0 = u0.long(), v0.long()
out = torch.zeros(res * res, device=uv.device, dtype=uv.dtype)
for dj, wj in ((0, 1 - du), (1, du)):
    for di, wi in ((0, 1 - dv), (1, dv)):
```

```

        jj, ii = u0 + dj, v0 + di
        inside = (jj >= 0) & (jj < res) & (ii >= 0) & (ii < res)
        idx = ii.clamp(0, res - 1) * res + jj.clamp(0, res - 1)
        out = out.index_add(0, idx, w * wi * wj * inside)
    return out.view(1, 1, res, res)

def render(ifs, res, n_points, chain, cam, sigma, generator):
    A, b, p = ifs.matrices()
    N = A.shape[0]
    # index sequence fixed for this step: the maps then enter continuously,
    # so gradients reach A and b by the pathwise route
    idx = torch.multinomial(p.detach(), n_points * chain, replacement=True,
                           generator=generator).view(n_points, chain)
    x = (torch.rand(n_points, 3, generator=generator, device=A.device) * 2 - 1)
    for k in range(chain):
        i = idx[:, k]
        x = torch.einsum("pij,pj->pi", A[i], x) + b[i]
    # probabilities get a gradient through a mass ratio that is 1 in value
    w = p[idx[:, -1]] / p.detach()[idx[:, -1]]

    yaw, pitch, dist, focal, scale = cam
    c, s = math.cos(yaw), math.sin(yaw)
    cp, sp = math.cos(pitch), math.sin(pitch)
    R = torch.tensor([[c, s * sp, s * cp], [0, cp, -sp], [-s, c * sp, c * cp]],
                     device=A.device, dtype=A.dtype)
    xr = (x / scale) @ R.T
    den = (xr[:, 2] + dist).clamp(min=0.3)
    uv = torch.stack([focal * xr[:, 0] / den, focal * xr[:, 1] / den], -1)

    img = splat(uv, w, res)
    img = gaussian_blur(img, sigma)
    img = torch.loglp(60.0 * img / (img.amax().detach() + 1e-9))
    img = (img / (img.amax().detach() + 1e-9)).clamp(0, 1)
    return img.repeat(1, 3, 1, 1) # SD expects 3 channels

def gaussian_blur(img, sigma):
    """Widens the one-pixel support of the splat gradient. Without it the loss
    landscape is almost flat and the optimiser cannot move the maps."""
    if sigma <= 0:
        return img
    r = max(1, int(3 * sigma))
    t = torch.arange(-r, r + 1, device=img.device, dtype=img.dtype)
    k = torch.exp(-t ** 2 / (2 * sigma ** 2))
    k = k / k.sum()
    img = F.conv2d(F.pad(img, (r, r, 0, 0), mode="replicate"), k.view(1, 1, 1, -1))
    return F.conv2d(F.pad(img, (0, 0, r, r), mode="replicate"), k.view(1, 1, -1, 1))

# -----
# 3. The trained model supplies the objective
# -----

class SDS:
    """Score Distillation Sampling against a frozen latent diffusion model.

    The U-Net is never back-propagated through: the SDS gradient is
    w(t)(eps_pred - eps), injected at the latent and carried onward to the IFS
    parameters by the renderer.
    """

    def __init__(self, prompt, device, model="stabilityai/stable-diffusion-2-1-base",
                 guidance=100.0, t_range=(0.02, 0.98)):
        from diffusers import AutoencoderKL, UNet2DConditionModel, DDPMScheduler
        from transformers import CLIPTextModel, CLIPTokenizer
        self.device, self.guidance = device, guidance
        self.vae = AutoencoderKL.from_pretrained(model, subfolder="vae").to(device).eval()
        self.unet = UNet2DConditionModel.from_pretrained(model, subfolder="unet").to(device).eval()
        tok = CLIPTokenizer.from_pretrained(model, subfolder="tokenizer")
        enc = CLIPTextModel.from_pretrained(model, subfolder="text_encoder").to(device).eval()
        self.sched = DDPMScheduler.from_pretrained(model, subfolder="scheduler")
        self.alphas = self.sched.alphas_cumprod.to(device)
        for m in (self.vae, self.unet, enc):
            m.requires_grad_(False)
        with torch.no_grad():
            def emb(txt):
                ids = tok([txt], padding="max_length", truncation=True,
                          max_length=tok.model_max_length, return_tensors="pt")
                return enc(ids.input_ids.to(device))[0]
            self.cond = torch.cat([emb(""), emb(prompt)])
        self.t_min = int(t_range[0] * self.sched.config.num_train_timesteps)
        self.t_max = int(t_range[1] * self.sched.config.num_train_timesteps)

```

```

def __call__(self, img):
    lat = self.vae.encode(img * 2 - 1).latent_dist.sample() * 0.18215
    t = torch.randint(self.t_min, self.t_max, (1,), device=self.device)
    noise = torch.randn_like(lat)
    noisy = self.sched.add_noise(lat, noise, t)
    with torch.no_grad():
        out = self.unet(torch.cat([noisy] * 2), t,
                          encoder_hidden_states=self.cond).sample
        eps_u, eps_c = out.chunk(2)
        eps = eps_u + self.guidance * (eps_c - eps_u)
        grad = (1 - self.alphas[t]) * (eps - noise)
        grad = torch.nan_to_num(grad)
        # surrogate whose gradient equals the SDS gradient
        return 0.5 * F.mse_loss(lat, (lat - grad).detach(), reduction="sum")

class CLIPObjective:
    """Cheaper alternative: cosine distance to the prompt embedding, with
    random crops (without them CLIP is satisfied by adversarial texture)."""

    MEAN = (0.48145466, 0.4578275, 0.40821073)
    STD = (0.26862954, 0.26130258, 0.27577711)

    def __init__(self, prompt, device, name="ViT-B-32",
                  pretrained="laion2b_s34b_b79k", n_aug=8):
        import open_clip
        self.model, _, _ = open_clip.create_model_and_transforms(name, pretrained=pretrained)
        self.model = self.model.to(device).eval().requires_grad_(False)
        tok = open_clip.get_tokenizer(name)
        with torch.no_grad():
            self.tf = F.normalize(self.model.encode_text(tok([prompt])).to(device), dim=-1)
        self.device, self.n_aug = device, n_aug
        self.mean = torch.tensor(self.MEAN, device=device).view(1, 3, 1, 1)
        self.std = torch.tensor(self.STD, device=device).view(1, 3, 1, 1)

    def __call__(self, img):
        res, views = img.shape[-1], []
        for _ in range(self.n_aug):
            sz = int(res * (0.65 + 0.35 * torch.rand(()).item()))
            i, j = (torch.randint(0, res - sz + 1, (2,))).tolist()
            v = img[..., i:i + sz, j:j + sz]
            if torch.rand(1) < 0.5:
                v = torch.flip(v, [-1])
            views.append(F.interpolate(v, 224, mode="bilinear", align_corners=False))
        batch = (torch.cat(views) - self.mean) / self.std
        feats = F.normalize(self.model.encode_image(batch), dim=-1)
        return 1 - (feats @ self.tf.T).mean()

# -----
# 4. Optimisation
# -----
def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--prompt", required=True)
    ap.add_argument("--objective", choices=["sds", "clip"], default="sds")
    ap.add_argument("--maps", type=int, default=24)
    ap.add_argument("--iters", type=int, default=1500)
    ap.add_argument("--res", type=int, default=512)
    ap.add_argument("--points", type=int, default=120_000)
    ap.add_argument("--chain", type=int, default=14)
    ap.add_argument("--lr", type=float, default=0.02)
    ap.add_argument("--s-lo", dest="s_lo", type=float, default=0.55)
    ap.add_argument("--s-hi", dest="s_hi", type=float, default=0.90)
    ap.add_argument("--sigma-hi", dest="sig_hi", type=float, default=3.0)
    ap.add_argument("--sigma-lo", dest="sig_lo", type=float, default=0.8)
    ap.add_argument("--out", default="fractal.json")
    ap.add_argument("--seed", type=int, default=0)
    a = ap.parse_args()

    dev = "cuda" if torch.cuda.is_available() else "cpu"
    gen = torch.Generator(device=dev).manual_seed(a.seed)
    ifs = IF3D(a.maps, s_max=a.s_hi, seed=a.seed).to(dev)
    obj = SDS(a.prompt, dev) if a.objective == "sds" else CLIPObjective(a.prompt, dev)
    opt = torch.optim.Adam(ifs.parameters(), lr=a.lr)
    cam = (0.0, 0.12, 4.5, 2.4, 2.2)

    for it in range(a.iters):
        f = it / max(1, a.iters - 1)
        # annealing both the contraction bound and the blur is what makes the
        # landscape navigable: coarse structure first, detail afterwards
        ifs.s_max = a.s_lo + (a.s_hi - a.s_lo) * f
        sigma = a.sig_hi + (a.sig_lo - a.sig_hi) * f

```

```

img = render(ifs, a.res, a.points, a.chain, cam, sigma, gen)
loss = obj(img)
# entropy keeps one map from monopolising the measure, the commonest
# collapse mode
_, _, p = ifs.matrices()
loss = loss - 0.01 * (-(p * p.clamp_min(1e-9).log()).sum())

opt.zero_grad(set_to_none=True)
loss.backward()
torch.nn.utils.clip_grad_norm_(ifs.parameters(), 1.0)
opt.step()

if it % 50 == 0:
    smax, psum = ifs.audit()
    print(f"it {it:5d}  loss {loss.item():+.4f}  "
          f"sigma_max {smax:.4f}  sum p {psum:.6f}", flush=True)

ifs.s_max = a.s_hi
smax, psum = ifs.audit()
# the result is an IFS or it is nothing: assert the hypothesis of
# Hutchinson's theorem before writing the file
assert smax < 1.0, f"non-contractive map survived: sigma_max = {smax}"
assert abs(psum - 1.0) < 1e-5, f"probabilities do not sum to 1: {psum}"
ifs.to_json(a.out, a.prompt)
print(f"\nwrote {a.out}: {a.maps} maps, sigma_max = {smax:.5f} < 1")
print("The attractor of these maps is a mathematical fractal, whatever the "
      "image looked like.")

if __name__ == "__main__":
    main()

```