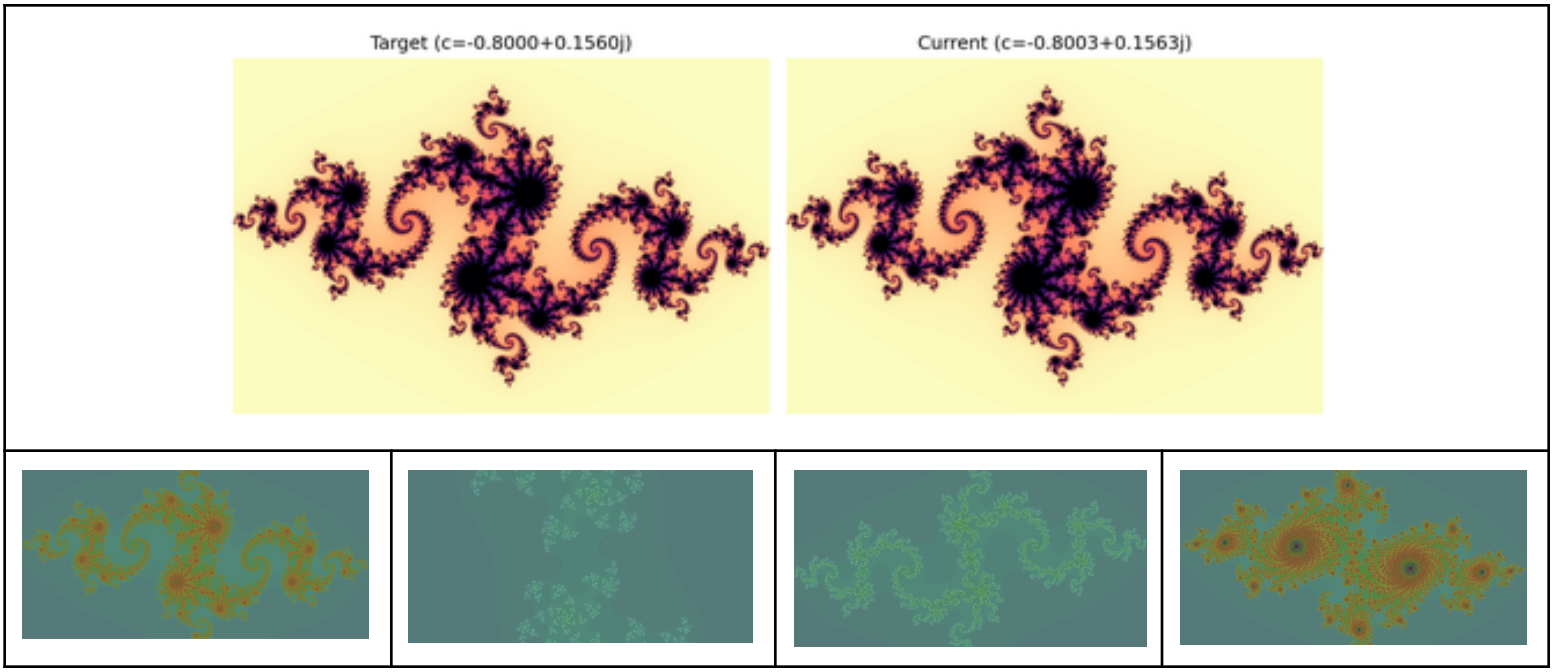
Inverse IFS Julia Sets: Differentiating Through Discontinuities

Benjamin Ahlbrand



Overview: Escape-time fractals such as IFS Julia sets are a demanding test for differentiable rendering. Each pixel is generated by iterating a nonlinear map with parameters θ until an orbit escapes or a maximum iteration is reached. As θ changes, most pixels do not move until a trajectory crosses the escape threshold one step earlier or later, abruptly switching bands or flipping from “inside” to “outside.” The image is therefore almost piecewise constant in θ with sharp, fractal discontinuities, and standard reverse-mode automatic differentiation (AD) mostly sees zero or misleading gradients, making inverse parameter recovery fragile.

Goal: Recover the parameters of an IFS Julia set from a target image using gradient-based optimization, **without** smoothing away the hard control-flow decisions that define the fractal.

Method: We adopt the discontinuity-aware sampling approach from *“Automatic Sampling for Discontinuities in Differentiable Shaders”* and implement it in Slang using [Disc].

- The escape condition in the renderer is annotated with [Disc].
- The Slang code is parsed in Python (via TreeSitter), where we build a control-flow graph and apply the [Disc] program transformation around that branch.
- The transformed code uses Monte Carlo “segment snapping” to sample across the decision boundary, evaluate both sides of the branch, and estimate how small changes in θ would flip control flow.
- Compiled with SlangTorch, this becomes a differentiable PyTorch module, and we optimize θ with Adam on an image-space loss to the target.

The result is a **control-flow–sensitive, piecewise gradient** that explicitly captures how moving the escape boundary changes the image.

Results: Naive AD through the escape-time renderer produces gradients that vanish away from the fractal boundary and cause optimization runs to plateau at high error or become unstable when the escape indicator is softened; reconstructions typically miss fine boundary structure.

With [Disc] discontinuity-aware sampling:

- Gradients reflect how θ shifts the escape-time boundary.
- Optimization converges more reliably to low loss.

Inverse IFS Julia Sets: Differentiating through Discontinuities

Benjamin Ahlbrand
https://github.com/bmahlbrand/auto_disc_sample

Introduction

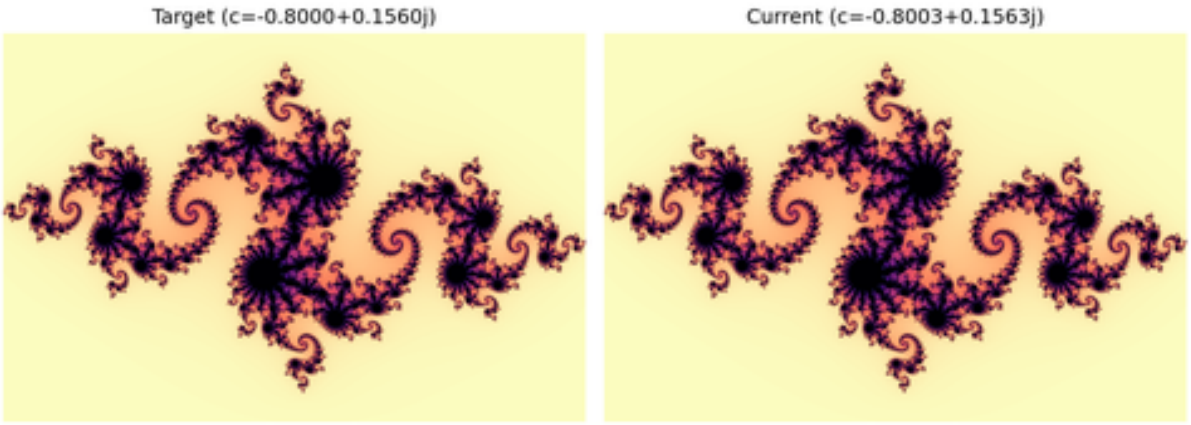
We aim to optimize parameters of shape programs containing hard branches. Existing AD and image-space losses often fail because gradients vanish or point in misleading directions when small parameter changes cause discontinuous structural changes. We adopt the discontinuity-aware sampling transformation from X and implement it in Slang using [Disc], enabling piecewise gradient estimation across decision boundaries. Concretely, given a target IFS Julia set image, we want to recover its underlying parameters via gradient-based optimization.

Differentiable optimization is a rich area which enables many applications. Of particular interest is differentiable rendering with an eye towards solving inverse problems in graphics to optimize shapes or add details to them. A classic limitation of this line of work is dealing with discontinuities, namely, how to differentiate through discontinuities. The way around this is by handling the dirac delta (elaborated on here) in some way, some recent work focuses on integrating the dirac delta to provide a piece-wise derivative. Specifically there is recent work described in “Automatic Sampling for Discontinuities in Differentiable Shaders” which enabled differentiating through hard decision boundaries in CSG tasks, voronoi partition towards targets, etc. In this work we demonstrate a new application for differentiably optimizing parameters of a well-known IFS Julia set fractal (a subset of Mandelbrot), a task which is fragile and difficult to optimize due to the inherent chaotic geometry of fractals with traditional approaches, such as with naive auto-diff and only image space gradients.

We had 3 high level goals with this work exploring the space of differentiable program optimization:

1. To evaluate / understand trade offs by using the segment snapping mechanism proposed by *Automatic Sampling for Discontinuities in Differentiable Shaders*[1] paper
2. To evaluate whether this would help get “base sculpts” for 3D geometry to manipulate and / or enable addition of surface surface details / refinement via inverse CSG
3. Bridge the gap between CSCI1951Q course material and shape program analysis (ie optimizing DSLs) while learning more about the advanced features built-into Slang

We interpret discontinuity-aware differentiation as a control-flow sensitive extension of auto-diff. Instead of differentiating only along the executed CFG path, the [Disc] transformation estimates how outputs change under infinitesimal perturbations that alter control-flow decisions.



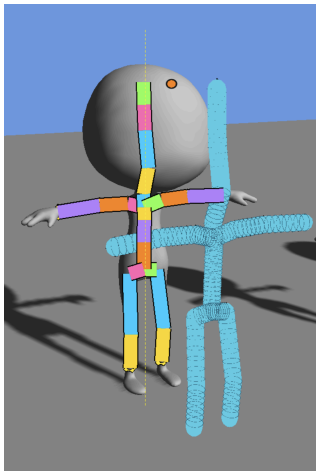
To recap, in an IFS Julia-style renderer each pixel is generated by repeatedly applying a nonlinear map to a complex value. Starting from an initial point, we iterate this map with some parameter vector theta until either the orbit escapes beyond a fixed radius or we hit a maximum number of iterations. The final image is therefore driven by an “escape-time” indicator: did the orbit escape, and if so, on which iteration? As a function of theta, this mapping is highly non-smooth. Tiny changes in theta can make a trajectory cross the escape threshold one step earlier or later, abruptly flipping a pixel from “inside” to “outside” the set or moving it to a different color band. At high resolution, these branch flips accumulate along a fractal boundary, so the rendered image becomes essentially piecewise constant with sharp discontinuities that follow the geometry of the set. Standard reverse-mode automatic differentiation only follows the branch decisions actually taken during a forward pass, so it either sees zero gradient almost everywhere (because the control flow does not change under small perturbations) or misses the Dirac-delta-like contribution when a branch flips. As a result, naive image-space losses combined with plain AD tend to produce gradients that either vanish or push parameters in unhelpful directions, making inverse parameter recovery fragile.

Implementation

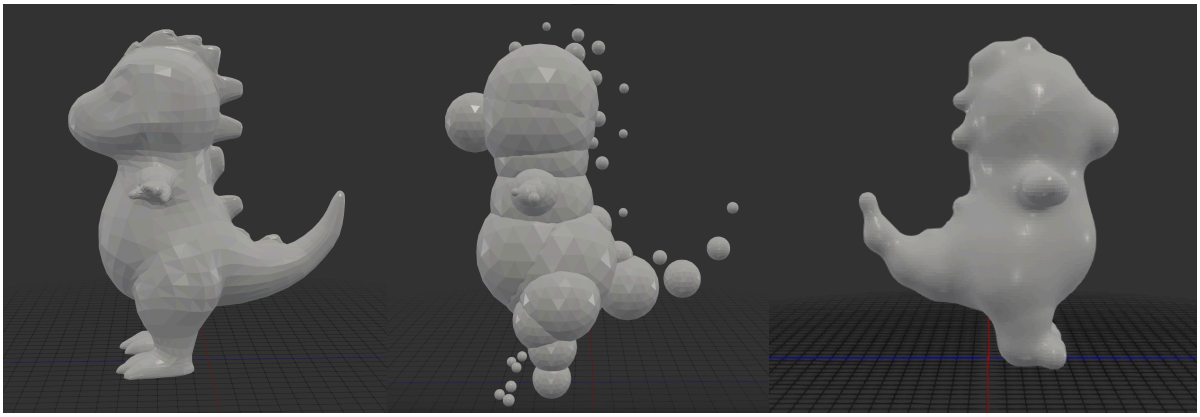
The system that we’ve worked on to enable CFG dependency analysis scaffolded atop of an automatic differentiation, PyTorch. This is a more generalized vectorized differentiable programming framework (Jax is a newer one which aims to give you more flexibility) that leverage GPUs in order to efficiently calculate gradients via the chain rule in order to optimize complex auto-diff graphs with respect to some objective function. This is often as simple as an L2 loss (mean-squared error) between a ground truth artifact and a generated output from some differentiable process. These systems are not capable of performing differentiation through discontinuities to handle control flow branches - traditionally what’s been anecdotally observed is a tendency to swap if else branches for lerps or other clever smooth approximations of the branch which are approximations and thus soft / fuzzy gradients, not the exact hard decision boundary.

We leverage source code privately shared by the author of “Automatic Sampling for Discontinuities in Differentiable Shaders”. This roughly follows the following high level architecture – parse a Slang shader with a decision boundary (if else branch) annotated by [Disc] to an AST in Python via TreeSitter, build a control flow graph to have the relationships

between branches - Belhe et al's system requires branch indexing to support piecewise sampling. Essentially perform a monte carlo sampling of segments - to drive evaluating lines across F(x) and G(x) [if and else] - the discontinuities between branching statements (integral over dirac delta) through importance sampling. You transform the Slang+[Disc] AST according to the piecewise constant strategy they're using to make it differentiable and thus compatible with autodiff - you output the transformation as a new Slang (around this [Disc] tag) program. You then take this compiled Slang shader, and make a differentiable module via SlangTorch and drop into a "traditional" PyTorch optimization loop with Adam (adaptive momentum) to minimize the error between the IFS fractal evaluated at given parameters and the optimized parameters.



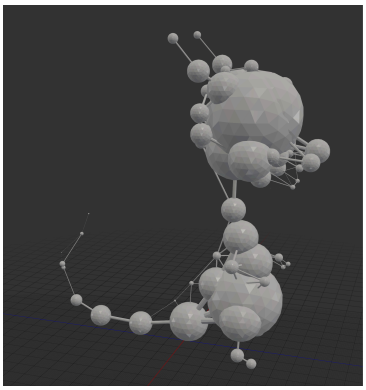
Building on someone else's unpolished unreleased research code presented its own set of challenges, but largely the program transformation code was relatively painless to use after a week or so of experimenting (and getting dependencies reproducible in a pyproject.toml file). Towards the end of getting some smooth base meshes to add details to, we attempted implementing Generalized Cylindrical Decomposition[3] [GCD] on predefined skeletons: given a 3D mesh, recover a set of tapered cylinders that approximate the input. The skeletons were exported from sampling a user-drawn curve over the mesh. One core issue in this pipeline is that the sampling mechanism does not reject overlapping segments,



which confounds the optimization and leads to low-quality fits even when the optimization technically converges. We then tried to discover the skeleton itself via optimization, following the GCD formulation. In practice, reliably estimating rotational symmetry axes (ROSA) of the mesh point cloud was the main failure mode; the difficulty of robustly detecting these symmetries dominated any benefit from discontinuity-aware sampling. We ultimately dropped the automatic sampling pipeline for this 3D setting and instead built a separate approach that recovers a set of spheres (e.g., 50 primitives - middle) that best approximate the mesh (left) and then wraps a soft skin over them in the spirit of *Dynamic Skeletonization Via Variational Medial Axis Sampling*[2]. The far right reconstruction in our example above is not yet a perfectly smooth base mesh—the spheres still bulge—but it is the raw output of a soft-min union followed by marching cubes. For

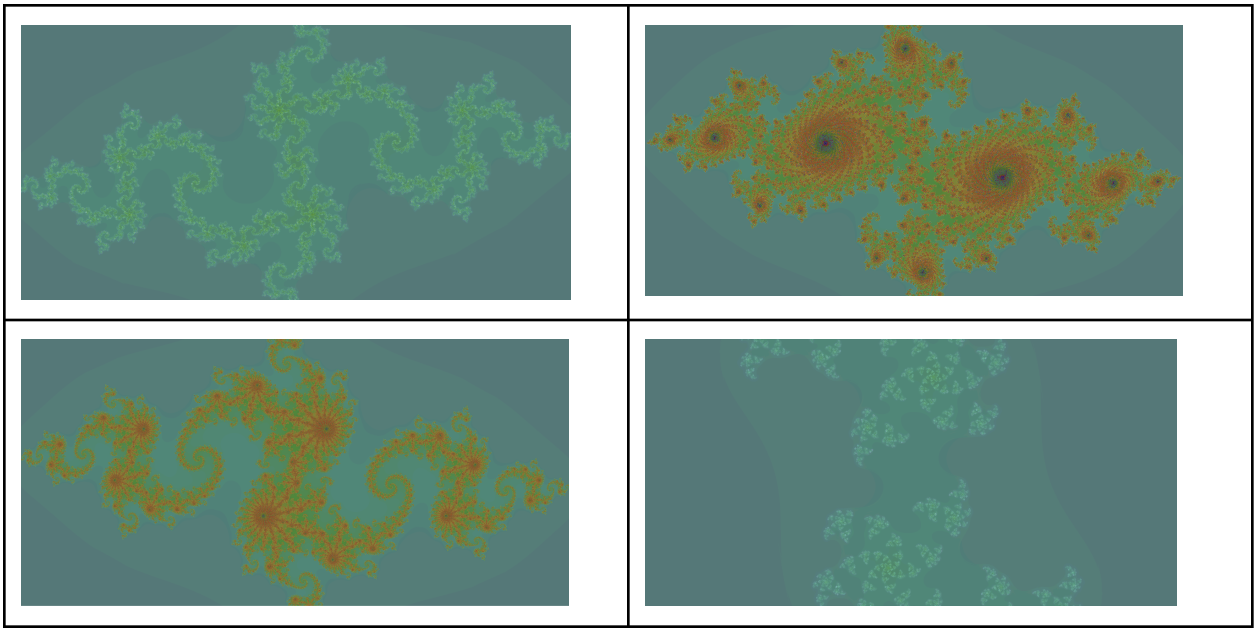
this formulation of coarse 3D reconstruction, hard decision boundaries and the [Disc] machinery were not the most effective or accessible tools.

This 3D detour, while useful for stress-testing the tooling, suggested that discontinuity-aware sampling is most valuable when branch-induced discontinuities are the central difficulty rather than one of several secondary issues (such as overlapping segments or symmetry detection). We therefore pivoted our main application to a domain where sharp decision boundaries are intrinsic to the definition of the object: IFS Julia set fractals. In the remainder of this work, we focus on using the [Disc] transformation to obtain informative gradients for inverse parameter optimization of these fractals, use case not covered in the original “Automatic Sampling for Discontinuities in Differentiable Shaders” paper.



a

Evaluation



Why focus on IFS fractals? As discussed above, the escape-time construction makes them a compact but challenging stress test for gradient-based inverse optimization – small parameter changes often have no effect until a branch flips, at which point the image can change abruptly. Empirically, this shows up very clearly. A naive baseline that differentiates through the shader with standard automatic differentiation and optimizes an image-space MSE loss systematically fails to recover nontrivial targets. In our experiments, such runs either plateau at high error (because gradients vanish away from the boundary), become numerically unstable when we soften the indicator, or wander in parameter space without reliably improving the reconstruction. This behavior makes IFS fractal parameter recovery a good candidate for testing whether discontinuity-aware sampling actually yields more informative, piecewise gradients in practice.

We tag the escape condition as the discontinuity to differentiate through. Admittedly we have no comparisons to other methods aside from image space AD, but the implementation contains naive autodiff with no interesting results since the gradients flatline and fail to find meaningful directions for the cases we tried (shown in the figures above), where using the [Disc] gradients to drive the optimization resulted in convergence across a scale curriculum, where we progressively introduce more iterations / complexity to the fractal (a toy problem to be sure).

Bibliography

1. Yash Belhe, Ishit Mehta, Wesley Chang, Iliyan Georgiev, Michael Gharbi, Ravi Ramamoorthi, and Tzu-Mao Li. 2025. Automatic Sampling for Discontinuities in Differentiable Shaders. ACM Trans. Graph. 44, 6, Article 209 (December 2025), 19 pages. https://doi.org/10.1145/3763291
2. Qijia Huang, Pierre Kraemer, Sylvain Thery, and Dominique Bechmann. 2024. Dynamic Skeletonization via Variational Medial Axis Sampling. In SIGGRAPH Asia 2024 Conference Papers (SA '24). Association for Computing Machinery, New York, NY, USA, Article 66, 1–11. https://doi.org/10.1145/3680528.3687678
3. Yang Zhou, Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, and Daniel Cohen-Or. 2015. Generalized cylinder decomposition. ACM Trans. Graph. 34, 6, Article 171 (November 2015), 14 pages. https://doi.org/10.1145/2816795.2818074
4. Derivative AT a Discontinuity