A Appendix

A.1 Notation

conv C	The *convex hull* of a finite set C.
bd X	Boundary set of X, the set difference of the closure of X with its interior.
dist $(\mathbf{x}, C)$	The (Euclidean) distance between a point x and a set C, i.e., $\inf\{\|\|\mathbf{x}-\mathbf{y}\|\|_2 \mid \mathbf{y} \in C\}$.
len f	Length of a fiber f, which is the norm of the difference of its endpoints $\|\|\mathbf{x}_s - \mathbf{x}_e\|\|_2$ and generally a fixed positive quantity $\ell > 0$.

A.2 Proofs

In this section we provide a (simple) proof which shows correctness of the Fiber Monte Carlo estimator (and its derivatives) in the two-dimensional case. We use the following lemma in our proof of correctness:

Lemma 1

$$\int_{y=-1-\ell}^{y=1} \int_{t=y}^{t=y+\ell} \mathbb{I}[t \in [-1,1]] f(t) dt\ dy = \ell \int_{t=-1}^{t=1} f(t) dt. \tag{21}$$

The justification is:

$$\begin{aligned}\int_{y=-1-\ell}^{y=1} \int_{t=y}^{t=y+\ell} \mathbb{I}[t \in [-1,1]] f(t) dt\ dy &= \int_{t=-1}^{t=1} \int_{y=t}^{y=t+\ell} f(t) dy\ dt \\ &= \ell \int_{t=-1}^{t=1} f(t) dt.\end{aligned}$$

The first equality holds if f is absolutely integrable (i.e., interchanging of integrals/limits is valid), and the fact that t is restricted to the closed set $[-1, 1]$ via the indicator function. The second step follows since the resultant integrand has no dependence on the innermost variable of integration, so the contribution is simply a constant value ℓ.

Correctness
For simplicity consider an absolutely integrable function $g_\theta : \mathbb{R}^2 \to \mathbb{R}$ supported in the square domain $[-1, -1] \times [1, 1]$, and say we sample a fiber of length $\ell > 0$ with vertical direction. We will prove that the line integral of g_θ over the fiber is an unbiased estimate of

$$I(\theta) = \int_{x=-1}^{x=1} \int_{y=-1}^{y=1} g_\theta(x, y) dy\ dx.$$

Concretely, the fiber has endpoints (x_0, y_0) and $(x_0, y_0 + \ell)$, where $x_0 \sim$ Unif$[-1, 1]$ and $y_0 \sim$ Unif$[-1 - \ell, 1]$. The line integral over the fiber is then:

$$\hat{I}_1 = \frac{1}{\ell} \int_{t=y_0}^{t=y_0+\ell} \mathbb{I}[t \in [-1,1]] g_\theta(x_0, t) dt.$$

The expectation of this estimate over (x_0, y_0) is:

$$\begin{aligned}\mathbb{E}[\hat{I}_1] &= \frac{1}{\ell} \int_{x=-1}^{x=1} \int_{y=-1-\ell}^{y=1} \int_{t=y}^{t=y+\ell} \mathbb{I}[t \in [-1,1]]\ g_\theta(x,t)\ dt\ dy\ dx & (22) \\ &= \frac{1}{\ell} \int_{x=-1}^{x=1} \ell \int_{t=-1}^{t=1} g_\theta(x,t)\ dt\ dx & (23) \\ &= \int_{x=-1}^{x=1} \int_{y=-1}^{y=1} g_\theta(x,y)\ dy\ dx & (24) \\ &= I(\theta), & (25)\end{aligned}$$

where equation 23 follows from Lemma 1, and equation 24 simply substitutes y for t to correspond with the original quantity. This proves correctness of the estimator.

Correctness of Derivatives

Correctness of the Fiber Monte Carlo derivative estimate follows straightforwardly from Lemma 1. First note that we cannot simply interchange differentiation and integration in the quantity

$$\frac{\partial}{\partial\theta}\int_{y=-1}^{y=1} f_\theta(y)dy, \tag{26}$$

since f_θ contains a discontinuity with respect to θ by assumption. Put another way, interchanging differentiation and integration is valid when interchanging integrals and limits is valid; a condition to establish this validity is that the integrand is continuously differentiable in θ. Despite the fact that we cannot straightforwardly interchange integration and differentiation here, we know the following equality holds from Lemma 1:

$$\frac{\partial}{\partial\theta}\int_{y=-1}^{y=1} f_\theta(y)dy = \frac{\partial}{\partial\theta}\frac{1}{\ell}\int_{t=y}^{t=y+\ell} \mathbb{I}[t\in[-1,1]]f_\theta(t). \tag{27}$$

The integral on the right hand side is almost everywhere differentiable. Assuming we can analytically compute the integral (e.g., via quadrature), we can use automatic differentiation to compute derivatives. Given Lemma 1, we can write derivatives of the estimator as,

$$\frac{\partial}{\partial\theta}\hat{I}(\theta) = \frac{1}{\ell}\frac{\partial}{\partial\theta}\left(\int_{t=y}^{t=y+\ell} \mathbb{I}[t\in[-1,1]]f_\theta(t)dt\right)dy. \tag{28}$$

A.3 Technical Details

A.3.1 Initialization of Scalar Fields

In applications like image stylization, we explicitly parameterized the scalar fields corresponding to rendering primitives so that at any initialization of the parameters, there were fibers intersecting the zero sublevel set of the field. Otherwise, there would be no gradient signal using this parameterization. In practice so long as the scalar field has mean value equal to zero and some zero crossings in the sampling domain, one can push useful derivatives through it during optimization.

A.3.2 Zero-Crossings and Implicit Differentiation

Our implicit differentiation formulation operates under the tacit assumption that the function does not vary ‘too much’ on the length-scale of any given fiber. This is required to guarantee convergence of the bisection method in computing the intersection point of a fiber whose endpoints evaluate to different signs: we need to ensure the function has no more than one zero-crossing along the fiber.

In practice, this condition is not problematic: in many applications of interest the function is piecewise linear. Moreover, even if it were not, the length of the fibers is a degree of freedom which can be chosen to avoid the circumstance of multiple zero-crossings.

In this section we explore a pedagogical case, using functions sampled from Gaussian processes as a surrogate: not meant as a practical field guide but a conceptual picture of what can go wrong if the fiber length is too long (which we make precise shortly).

We consider a Gaussian process with mean function m and kernel k over real scalar field functions $g(x) \sim \mathcal{GP}(m(x), k(x,x'))$. Assume a zero mean Gaussian process with unit marginals and standard Gaussian kernel:

$$k(x,x') = \exp(-({}^{1}/{}_{(2L^2)})||x-x'||_2^2).$$

Then it follows that the expected number of zeros over a fiber of length ℓ is $\mathbb{E}N = \frac{\ell}{\pi L}$ (Ylvisaker, 1965). This result dates back to (Rice, 1944) who derives it from a different (but equivalent) set of assumptions. Suppose we want to bound the probability of more than one zero crossings over ℓ by 0.01, then we choose $\ell = (2\pi L)(.01)$ by Markov's inequality. In this simple case, the interpretation of the bound is: provided the fiber length to is less than 6% of a given characteristic lengthscale of variation associated with the functions, on average fewer than 1% of the fibers we sample result in the bisection method failing to converge.

In the general case, we of course lack an explicit model for g. The point of the argument is to certify the intuitive idea that the fiber length should be chosen strictly smaller than some characteristic 'lengthscale of variation' of the function, we use the simplified case in order to make this notion precise.

There is a straightforward analogy to simple Monte Carlo. In simple Monte Carlo, the variance of the estimator is proportional to the variance of the estimand: analogously, the more 'variable' the implicit function, the shorter (and thus, more) fibers are needed to achieve the same estimate variance (all else equal).

A.4 Experimental Details

A.4.1 Image Stylization

We used a single Nvidia RTX 3080Ti GPU for differentiable rendering, with the wall clock time to optimize against a single image ranging from 5-15 minutes, depending on the number of rendering primitives used. Rendering with triangles, we used 75 triangles, initialized by sampling their vertices uniformly at random from the space spanned by the image, and then computing the triangles with the Delauney triangulation of the vertices (during optimization we disallowed updates that would cause any vertices to migrate outside of the image). Rendering with negated, shifted, Gaussian densities, we used 50 implicit functions, whose means were sampled uniformly over the image space, with precisions sampled uniformly from a fixed range (determined visually so that the primitives were large enough at initialization). Rendering with SIREN networks, we used 20 networks with a frequency $\omega = 10$. Each network had 3 hidden layers, each of size 8. For all rendering experiments we optimized using Adam with a step size of .001.

A.4.2 Topology Optimization

The most common relaxation pixelizes the domain, relaxes the problem to contain "intermediate" amounts of material in each pixel, and runs an optimization algorithm over the relaxed densities of each pixel to find the optimal topology. To ensure physicality, the method penalizes intermediate values. This is called SIMP (Andreassen et al., 2011), and although by far the most common approach, can often lead to mesh dependence ("checkerboarding") and highly nonphysical materials. To counteract this, various filtering and regularization techniques have been designed to stabilize the resulting designs. Despite these limitations, SIMP-type approaches are widely considered the industry standard in topology optimization.

An attractive alternative approach to topology optimization involves representing the domain as a level-set of a higher dimensional function. This approach has garnered much research (van Dijk et al., 2013) as it promises better continuity and mesh independence, but a problem it has is that it involves differentiating through integrals with parametric discontinuities (usually used to compute potential energy over the domain defined by a level-set). The common way to compute this involves using a smoothed Heaviside-function in the integral, but with fiber sampling we can do it exactly.

Although with the level-set approach any type of implicit surface is in theory possible (e.g., neural implicit fields), to demonstrate fiber sampling and avoid additional difficulties of using neural fields for topology optimization we stick to the most common parameterization: a grid of localized basis functions forming a piecewise linear function over the background domain, as in Belytschko et al. (2003).

A.4.3 Amortized Convex Hulls

The hypernetwork produces a pointwise embedding for each input point (using a single MLP), aggregates those embeddings dimension-wise with a symmetric function (e.g., maximum, sum, average), and then produces the implicit parameters as output by transforming this global embedding (using another MLP).

For the ModelNet40 experiment we sampled 40 normalized halfspaces whose direction was chosen uniformly over the unit cube. We preprocessed the data by translating the mean of the point clouds to the origin, and scaling so that the point cloud fit within the unit 2-norm ball, each point cloud was comprised of 1000 samples drawn uniformly over the surface of the original plant mesh. The hypernetwork used a pointwise MLP with three hidden layers of size 200, dimension-wise maximum as the symmetric aggregation, and an output MLP with three hidden layers of size 200. We trained on a single Nvidia RTX 3080Ti GPU using Adam with step size .001: training took 2 hours wall clock time.

A.4.4 Reproducing Results

We will release our generic utilities via a public Python package. In the effort of minimizing the difficulty of reproducing the results detailed in this work, we will also publish an associated Dockerfile that can be used to build a cross-platform container image to reproduce any of the figures displayed in this paper.