PyFRep: 可微分函数表示的形状建模

Pierre-Alain Fayolle*

Evgenii Maltsev†

摘要

我们提出了一种基于函数表示（FRep）的可微几何建模框架。该框架构建在现代自动微分库之上，使我们可以获得关于空间或形状参数的导数。我们展示了该框架的可能应用：曲率估计用于形状查询、有符号距离函数的计算与逼近以及将参数模型的形状参数拟合到数据中。我们的框架作为开源发布 [1]。

关键词: 形状建模；形状建模编程语言；隐式曲面建模；可微分形状建模；有符号距离函数；参数模型拟合。

1 介绍

通过自动微分计算导数的能力是执行反向传播的基础，而反向传播是深度学习背后的核心机制。它现在开始在科学计算、计算机图形学或 CAD 系统等领域中发挥重要作用。

例如，可微渲染使我们能够解决逆向渲染任务，这包括从图像中恢复场景的信息，如相机位置、光源的强度和位置、模型参数等。另一个例子是基于梯度的优化，在图像修复、图像配准、立体视觉、光流估计等领域广泛使用。在基于 CAD 的形状优化和数值优化任务中，几何敏感性（CAD 表面点

*University of Aizu, fayolle@u-aizu.ac.jp

†Skoltech, evgenii.maltsev@gmail.com

[1] `https://github.com/fayolle/PyFRep`

关于模型设计参数的梯度）非常重要，并且也需要高效准确地计算导数。最后，关于导数的信息在 3D 建模、渲染和网格生成算法中也可能有用。

在这篇论文中，我们描述了一个基于函数表示（FRep）[37] 的可微几何建模框架，并展示了其可能的应用。FRep 允许通过一个连续实函数来定义几何形状。我们将函数几乎处处可微作为要求。我们考虑了该框架的潜在应用，例如评估曲率信息，这在形状查询、符号距离函数计算和近似以及拟合模型参数到数据方面是有用的。我们的框架是用 Python 实现的，并以开源形式发布 [1]。

2 相关工作

程序化形状建模系统 多年来，已经提出了多种程序化形状建模系统。这包括基于隐式曲面建模的系统，如 BlobTree[52, 46]，以及基于函数表示（FRep）的系统，包括 HyperFun[36]、针对 Java 虚拟机的目标变体 [9, 15] 或最近的实现如 LibFive[25, 26]。

OpenSCAD[1] 是一个流行的程序化系统，基于通过边界表示法来表示实体。它似乎在快速原型设计社区中特别受欢迎。

快速原型制作社区中另一个流行的系统是 IceSL[27]。它是一个依赖于隐式表面、有符号距离函数和多边形网格的混合系统。类似的系统还有 OpenFab[51]，侧重于多材料物体建模。

类似于这些系统，我们提出了一种程序化形状建模系统。然而，我们在自动微分库的基础上构建它，因此我们的形状模型对于空间坐标或形状参数是可微的。我们使用这些设施进行形状查询或将参数模型拟合到数据集。

自动微分 自动微分是指用于获取定义为计算机程序（计算图）的函数导数的技术。这些技术通过应用链式法则来自动化导数计算。自动微分有两种不同的模式：前向模式和反向模式。每种模式对应于遍历计算图时应用链式法则的顺序。从左侧（输入）累积导数对应于前向模式，而从右侧（输出）累积导数对应于反向模式。对于基于标量的函数，反向模式更高效；而对于基于向量的函数以及需要更高阶导数的情况，前向模式则更高效。读者可以参考 Griewank 和 Walther 所著的书籍 [18] 或最近的综述 [4] 获取更多细节。

自动微分是高效的，因为它允许以与函数评估相同的（渐近）时间复杂度计算梯度，这不同于例如有限差分的方法。

自动微分作为深度学习框架的一部分，用于实现反向传播算法 [40, 33]，以训练神经网络，这一领域正日益受到关注。流行的框架包括 PyTorch[39]、Tensorflow[2] 或 JAX[8] 等。

可微分系统在图形学中的应用 计算机图形学和计算机视觉领域已经进行了多次尝试来开发可微渲染系统。一个可微渲染器提供图像像素相对于场景参数（相机参数、模型、...）的导数。

第一个可微渲染系统之一，OpenDR[31]，被应用于将人体拟合到图像和深度图像中。随后开发了其他几个用于可微渲染的系统，如 [3, 28, 29, 30]。最近提出了用于着色器设计的可微渲染器 [19, 48]。其他可微系统也被提出用于图像处理和图形中的优化 [10] 或物理模拟 [23]。关于该主题的近期综述，请参阅 [24, 50]。

这些系统处理可微渲染算法。我们处理一个可以程序化表达形状及其导数的系统，其中导数是关于空间或形状参数计算的。随着生成式 AI 的最新进展，我们认为程序化形状建模系统可以发挥重要作用。参见例如 [12, Section 5] 中的讨论。

与本工作更接近的方法是在 [34] 中描述的基于 NURBS 的方法，而我们的系统则基于函数表示。可微几何内核的使用对于 CAD 模型的模拟和优化 [20] 是有用的。

3 背景

本工作的目标是描述一个基于函数表示 [37] 的微分几何核。在本节中，我们提供了关于基于函数表示建模和自动微分的背景信息。

3.1 函数表示

函数表示法 [37] 是一种通过标量函数 $f : \mathbb{R}^3 \to \mathbb{R}$ 来表示实体的方法。实体的表面对应集合 $\{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}) = 0\}$，内部对应 $\{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}) > 0\}$，外部

则对应 $\{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}) < 0\}$。复杂固体对象是通过考虑与操作 [43, 7, 37, 17] 结合的原始的对象构建的。

使用隐式曲面或符号距离函数建模固体正变得越来越流行，因为它允许表示瓦片状和重复的图案，即所谓的微结构，在生物医学应用和其他行业中与 3D 打印技术一起使用时非常有用 [38]。

原始元素 示例的基元包括简单的对象，如球体、圆柱体、锥体、长方体、基于骨架的基元、多项式或通过将样条拟合到点云而获得的对象。

操作 应用于原始图形的操作包括仿射变换，如平移、旋转、缩放和剪切。这种表示方式便于使用 R-函数 [47, 37]

- 联合 $S_1 \cup S_2 := f_{S_1} + f_{S_2} + \sqrt{f_{S_1}^2 + f_{S_2}^2}$,
- 交集 $S_1 \cap S_2 := f_{S_1} + f_{S_2} - \sqrt{f_{S_1}^2 + f_{S_2}^2}$,
- 补集 $\overline{S} := -f_S$,
- 差异 $S_1 \setminus S_2 := S_1 \cap \overline{S_2}$,

定义构造型实体几何（CSG）操作，其中 S_i 是与集合 $\{x \in \mathbb{R}^3 : f_{S_i} \geq 0\}$ $(i = 1, 2)$ 对应的实体。布尔运算的替代实现包括 $\min / \max$ 及其变体。也可以很容易地定义诸如混合、拉伸、偏移/壳体、变形等操作 [37]。

基于 SDF 的建模 函数表示法的一个流行子集是仅使用有符号距离函数 (SDF)。一个函数 f 是 SDF 如果 $f(\boldsymbol{x})$ 对应于从 $\boldsymbol{x} \in \mathbb{R}^3$ 到给定表面 ∂S 的有符号（欧几里得）距离。符号用于确定点 $\boldsymbol{x}$ 是否位于由表面 ∂S 界定的实体内部或外部。

SDF 可以通过使用归一化的 CSG 操作 [5, 14] 构造性地构建，也可以通过数值过程构建。它们在材料建模 [6] 或计算物理 [35] 等领域有应用。

3.2 自动微分

我们考虑一个函数 $f : \mathbb{R}^d \to \mathbb{R}$，我们希望计算其导数。我们假设该函数是由一组具有已知导数的基本函数通过复合构成的。请注意，这些基本函数最终

可能是向量值的（即映射 $\mathbb{R}^M \to \mathbb{R}^N$）。计算 f 的导数是通过对链式法则反复应用来完成的。

假设对于 $f = f_3 \circ f_2 \circ f_1$ 我们有

$$y = f(\boldsymbol{x}) = f_3(f_2(f_1(\boldsymbol{x}))), \qquad \boldsymbol{x} \in \mathbb{R}^d, \quad y \in \mathbb{R}.$$

让我们注意

$$\boldsymbol{x}_1 = f_1(\boldsymbol{x}), \qquad \boldsymbol{x}_2 = f_2(\boldsymbol{x}_1).$$

梯度（或雅可比矩阵）为 f 给出如下：

$$f'(\boldsymbol{x}) = \frac{\partial y}{\partial \boldsymbol{x}} = \frac{\partial y}{\partial \boldsymbol{x}_2}\frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{x}_1}\frac{\partial \boldsymbol{x}_1}{\partial \boldsymbol{x}}, \tag{1}$$

和

$$f_3'(\boldsymbol{x}_2) = \frac{\partial y}{\partial \boldsymbol{x}_2}, \quad f_2'(\boldsymbol{x}_1) = \frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{x}_1}, \quad f_1\prime(\boldsymbol{x}) = \frac{\partial \boldsymbol{x}_1}{\partial \boldsymbol{x}},$$

其中 f_1', f_2' 和 f_3' 分别是原函数 f_1, f_2 和 f_3 的雅可比矩阵。

(1) 中的每一项都是一个矩阵。自动微分中的前向模式对应于从右到左执行矩阵乘法，而反向模式则对应于从左到右执行乘法。在使用隐式进行形状建模的情况下，函数 f 是标量值的，因此反向模式更高效。

在实现方面，复合函数 f 对应于原始函数的有向无环图。评估该函数对应于图的前向遍历。为了实现反向模式，必须在前向传递过程中累积中间值。通过逆向遍历图形并计算 $\frac{\partial y}{\partial \boldsymbol{x}_i}$ 以获得中间值 $\boldsymbol{x}_i$ 的梯度。

4 PyFRep 框架

我们用一个标量函数 $f : \mathbb{R}^3 \to \mathbb{R}$ 表示一个固体，该函数几乎处处可微。固体的表面对应于 f 的零水平集，即集合 $\{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}) = 0\}$。我们将 $f > 0$ 作为固体内部的惯例表示。

几乎处处可微的术语用于表示函数除了在一个测度为零的集合之外，在所有地方都是可微的。例如，当 $x = y$ 时，函数 $\min(x, y)$ 才不可微。值得注意的是，PyTorch[39] 在实现函数 min 和 max 时，例如在 $x = y$ 时使用了次梯度 $(0, 1)$。实际上，只需注意防止 NaN 值的传播即可。使用次梯度或次微分是一种可能的策略。另一种可能性是在可用时考虑原始函数和操作的 C^1

阶或更高阶变体，例如 [37] 中的 (5)。实际上，根据我们的实验以及我们所考虑的应用，在某些点上不可微并没有造成任何问题。

与固体对应的函数 f 由计算机程序表示。使用 Python 的一个子集作为编程语言。通过反向模式的自动微分计算 f 的导数。在当前实现中我们依赖于 PyTorch[39]，尽管任何其他的自动微分库 [2, 8] 都可以作为后端使用。

通常，实体对象不是从零开始构建的，而是通过将几何运算应用于更简单的基元来定义的，参见第 3.1 节。我们已经实现了所有在基于隐式建模软件包中常见的基元和操作。一个简单固体及其对应函数的例子如图 1 所示。对应的实体显示在右侧。

```
def model(x):
    sp1 = sphere(x,center=(0,0,0),r=1)
    b1 = block(x,
               vertex=(-0.75,-0.75,-0.75),
               dx=1.5,dy=1.5,dz=1.5)
    t1 = intersection(sp1,b1)
    c1 = cylX(x,center=(0,0,0),r=0.5)
    c2 = cylY(x,center=(0,0,0),r=0.5)
    c3 = cylZ(x,center=(0,0,0),r=0.5)
    t2 = difference(t1,c1)
    t3 = difference(t2,c2)
    t4 = difference(t3,c3)
    return t4
```

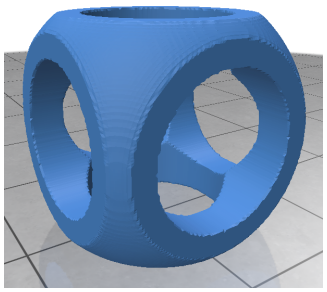

图 1: 定义简单形状的程序示例（左）及其对应的实体（右）。

所有原始操作和运算都用纯 Python 实现，并与执行导数计算的部分解耦。通过调用一个表示实心物体的 Python 程序 $f(.)$，并将变量 $\boldsymbol{x}$ 设置为需要计算导数，可以得到跟踪反向模式计算出的导数的计算图。然后通过反向模式自动微分获得这些导数。这种方法使得系统易于与不同的后端（例如 PyTorch[39]、Tensorflow[2] 或 JAX[8]）一起使用，因为原始操作和运算的核心部分是用纯 Python 编写的。

有符号距离函数 FRep 的一个流行子集是有符号距离函数（SDF）。如果 f 给出从 $\boldsymbol{x} \in \mathbb{R}^3$ 到给定曲面 ∂S 的有符号（欧几里得）距离，则 $f(\boldsymbol{x})$ 是一个 SDF。我们实现了所有通常的 SDF 原语和操作 [14, 41]。目前，SDF 原始形状和操作与非 SDF 原始形状和操作分别存在于两个不同的模块中。当然，可以通过混合使用 SDF 和非 SDF 原始形状和操作来建模实体，但总体函数的距离属性将丢失。因此，如果 f 是从 SDF 操作和基元以及非 SDF 操作和基元的混合构建而成的函数，那么 $f(\boldsymbol{x})$ 不再对应于点 $\boldsymbol{x}$ 到表面 ∂S 的欧几里得距离。SDF 允许更有效地实现渲染算法，例如使用球体追踪的光线行进 [22]，并为异质对象建模提供了一个建模参数 [6]。另一方面，它具有更为有限的一组建模范式和操作。

周期函数 函数表示和基于 SDF 的建模在近年来越来越流行，用于建模平铺和重复图案。这用于建模微观结构，在 3D 打印等领域已被证明是有用的 [38]。我们的系统实现了不同的操作符以重复（单元格）模式，例如三角波函数和锯齿波函数。它们可以与简单的原语（如球体或环面）结合生成重复图案，如图 [38] 所示。尽管锯齿波或三角波函数在某些点不可微分，但也可以用其截断的傅里叶级数来替换它们。我们还提供了几种三重周期极小曲面（如旋节线和利迪诺伊德）的实现，这些曲面正变得非常流行用于建模生物结构等 [21]。

参数建模 在为计算机辅助设计（CAD）应用建模形状时，会将常规的几何体（如长方体、球体、圆柱体等）与布尔运算（并集、交集、差集）和其他操作（扫掠、倒角等）结合使用。这些基本几何体和操作可以进行参数化，从而产生参数化建模 [13]。例如，一个球体可以通过其中心点和半径来参数化（参见图 1 中模型中对 $sphere()$ 的调用）。改变这些参数可以探索建模空间中的不同形状。例如，图 2 展示了通过改变圆柱体的半径从图 1 中的模型获得的不同形状。如果 f 是定义给定实体的函数，我们用 $f(\cdot; \boldsymbol{p})$ 表示对形状参数 $\boldsymbol{p} \in \mathbb{R}^n$ 的依赖。对应于一组给定参数 $\boldsymbol{p}$ 的表面由 $\{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}; \boldsymbol{p}) = 0\}$ 给出。对于给定的 $\boldsymbol{x}$，可以将 $f(\boldsymbol{x}; \boldsymbol{p})$ 解释为 $\boldsymbol{p}$ 的函数，并计算相对于这些参数的导数。这些计算类似于空间导数的情况。这使我们能够将一个参数模型拟合到给定的数据集上，例如，在对象表面上采样的点集合 $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$。这

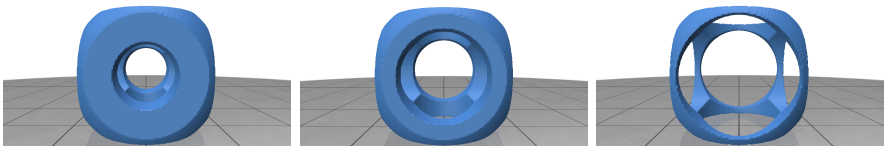

图 2: 通过改变圆柱的半径获得的不同形状（从左到右，$r = 0.35, 0.5, 0.65$）。

可以通过最小化损失函数

$$E(\boldsymbol{p}) = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i; \boldsymbol{p})^2 \tag{2}$$

来实现，使用随机梯度下降（SGD）[44]。注意，对于空间变量 $\boldsymbol{x}$ 或形状参数 $\boldsymbol{p}$ 的导数计算是不同的事情。即使一个函数在给定点上相对于空间变量不可微，它可能仍然在该点相对于其形状参数可微。

导数计算 如上所述，核心库由用纯 Python 编写的几何原语和操作构成。可以计算关于空间参数（上述的 $\boldsymbol{x}$）或形状参数（上述的 $\boldsymbol{p}$）的导数。我们依赖 PyTorch 为给定函数创建计算图并通过反向模式自动微分来计算导数。这是通过将相应的变量封装在一个需要计算导数的 Torch 张量中来完成的。例如，参见图中的函数 3 以计算 $\nabla f(\boldsymbol{x})$，即函数 f 关于其参数 x 的梯度。计

```
def compute_grad(f, x):
    if not torch.is_tensor(x):
        x = torch.tensor(x)
    x.requires_grad = True
    y = f(x)
    g = autograd.grad(y, [x],
                      grad_outputs=torch.ones_like(y),
                      create_graph=True)[0]
    return g
```

图 3: $\nabla f(\boldsymbol{x})$ 的计算。

算模型的梯度对其空间参数或形状参数（用于形状优化）都是有用的。对于后者，通常考虑损失函数（如 2）关于形状参数 $\boldsymbol{p}$ 的梯度（空间参数 $\boldsymbol{x}$ 是固定的）。我们还提供了计算散度、拉普拉斯算子（梯度的散度）、p-拉普拉斯算子和曲率（主曲率、平均曲率和高斯曲率）的函数。这些运算符主要用于空间导数。例如，图 4 显示了计算向量场散度的代码。梯度和散度的计算是

```
def compute_div(v, x):
    div = 0.0
    if not torch.is_tensor(x):
        x = torch.tensor(x)
    x.requires_grad = True
    y = v(x)
    for i in range(y.shape[-1]):
        div += autograd.grad(y[..., i],
                             x,
                             grad_outputs=torch.ones_like(y[..., i]),
                             create_graph=True)[0][..., i:i + 1]
    return div
```

图 4: 向量场的散度计算 $\boldsymbol{y} = \boldsymbol{v}(\boldsymbol{x})$

计算不同曲面曲率、拉普拉斯算子（Δf）以及传统几何建模中常用微分算子所必需的唯一要素。

附加功能 框架的其余部分包含了用于执行标准网格和点云文件格式输入/输出的通用功能，实现了 Marching Cubes 算法 [32] 以渲染给定模型的零水平集（表面），计算模型不同曲率的功能，不同的归一化方案以及一个重新初始化算法，该算法用于计算给定函数零水平集的符号距离函数。最后，我们提出了根据不同约束条件拟合形状参数的不同算法实现，这对于逆几何建模非常有用。

5 应用示例

本节描述了几种可能的应用：表面曲率的自动计算；用于距离近似的归一化方案；通过启发式方法和梯度下降的组合进行重距和形状拟合。

5.1 曲率计算

曲面的主曲率是高阶微分曲面量，用于测量给定点处曲面在给定方向上的弯曲程度。它们对于形状分析很有用。在我们的框架中，曲面 ∂S 是从给定函数 $\partial S = \{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}) = 0\}$ 的零水平集获得的。计算这些曲率涉及 f 的二阶导数，这些导数是通过自动微分从 f 的程序规范中得出的。与大多数现有方法不同，我们不依赖任何近似，例如用多项式或样条拟合来逼近曲面。

平均曲率 对于曲面点 $\boldsymbol{x}$，平均曲率由下式给出

$$H = -\frac{1}{2}\operatorname{div}(\boldsymbol{n}), \quad \text{where} \quad \boldsymbol{n} = \frac{\nabla f}{|\nabla f|} \tag{3}$$

其中 div 是散度算子，$\boldsymbol{n}$ 是该点 $\boldsymbol{x}$ 处的（内向）单位法向量，从定义函数 f 的归一化梯度获得。

高斯曲率 我们使用了高斯曲率的表达式给出在 [16, Theorem 4.1]

$$K = \frac{\nabla f \cdot \text{Hessian}^*(f) \cdot \nabla f^T}{|\nabla f|^4} \tag{4}$$

其中 Hessian(f) 是矩阵 f 的海森矩阵，M^* 表示矩阵 M 的伴随矩阵，而 M^T 表示矩阵 M 的转置。

主曲率 最后，主曲率 $\kappa_{\min}$ 和 $\kappa_{\max}$ 可以通过平均曲率和高斯曲率计算得到

$$H = \frac{1}{2}(\kappa_{\min} + \kappa_{\max}) \qquad K = \kappa_{\min}\,\kappa_{\max}.$$

即我们有

$$\kappa_{\min} = H - \sqrt{H^2 - K} \qquad \kappa_{\max} = H + \sqrt{H^2 - K}.$$

示例 Schwarz D 极小曲面由

$$\sin(x)\sin(y)\sin(z) + \sin(x)\cos(y)\cos(z)$$
$$+ \cos(x)\sin(y)\cos(z) + \cos(x)\cos(y)\sin(z) = 0.$$

给出。见图 5。该曲面（零水平集）通过 Marching Cubes 算法计算得到。然

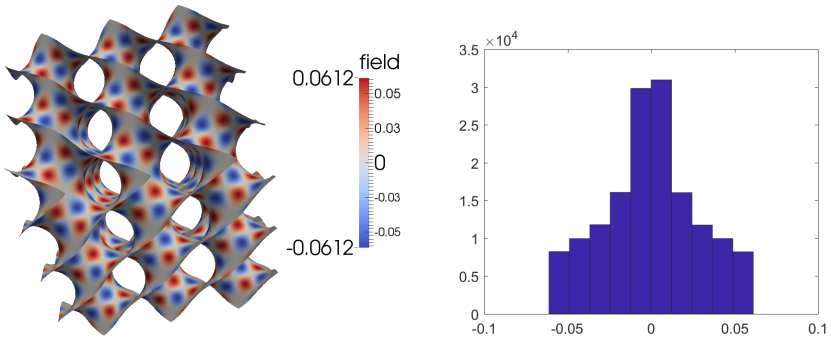

图 5: 左：极小曲面上的平均曲率（Schwarz D 极小曲面）。右：平均曲率值的分布。

后在从 Marching Cubes 算法获得的三角网格的每个顶点处，根据 (3) 计算平均曲率。对于极小曲面，我们知道 $H = 0$。我们可以通过实验验证图 5 中计算得到的平均曲率值接近于零。这也通过查看图 5 右侧的平均曲率值分布得到了证实。

5.2 规范化

标准化是一种用于计算从一个点到隐式定义的表面（二维情况下的曲线）[45, 49] 的大致距离的技术。假设边界 ∂S 是函数 $f(\boldsymbol{x})$ 的零水平集，则距离 $d(\boldsymbol{x})$ 可以通过 [45] 来估计

$$\omega_1[f](\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{\sqrt{f(\boldsymbol{x})^2 + |\nabla f(\boldsymbol{x})|^2}}. \tag{5}$$

归一化 $\omega_1[f]$ 满足边界条件 $\omega_1[f](\boldsymbol{x}) = 0$ 和 $\partial\omega_1[f]/\partial\boldsymbol{n} = 1\partial S$

对顺序 k 的归一化是通过递归获得的：

$$\omega_k[f](\boldsymbol{x}) = \omega_{k-1}[f](\boldsymbol{x}) - \frac{1}{k!}\omega_1[f]^k \frac{\partial^k \omega_{k-1}[f]}{\partial \boldsymbol{n}^k}, \tag{6}$$

并满足边界条件

$$\begin{aligned} &\omega_k[f] = 0 \quad \text{and} \quad \partial\omega_k[f]/\partial\boldsymbol{n} = 1 \quad \text{on} \quad \partial S, \\ &\partial^l\omega_k[f]/\partial\boldsymbol{n}^l = 0 \quad \text{on} \quad \partial S, \quad l = 2, 3, \ldots, k. \end{aligned} \tag{7}$$

一种替代的标准化方案由如下给出 [49]

$$\delta_1[f](\boldsymbol{x}) = f(\boldsymbol{x})/|\nabla f(x)|. \tag{8}$$

所有这些规范化方案都基于模型 $f(\cdot)$ 的空间导数的计算。尽管 $\omega_1[f]$ 和 $\delta_1[f]$ 满足适当的边界条件，并且在表面附近提供了距离函数的良好近似，但它们远离边界的性能要准确得多。椭圆的归一化如图 6 所示。

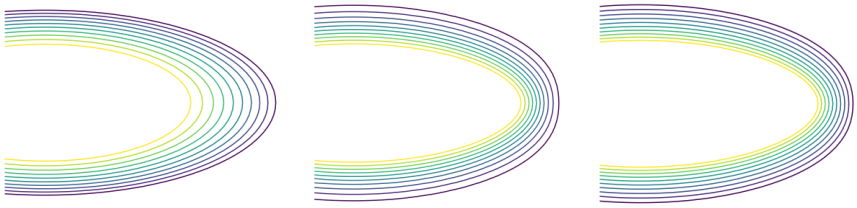

图 6: 左：长轴和短轴分别为 5 和 2 的椭圆的等值线（从-0.5 到 0.5）。中：Rvachev 标准化 w_1。右：Taubin 标准化 δ_1

5.3 距离函数估计

我们考虑重新初始化问题，也称为重距离化，对于隐式曲面而言。问题是这样的：给定 $f : \mathbb{R}^3 \to \mathbb{R}$，其零水平集定义了一个曲面 ∂S，我们想要计算符号距离函数（SDF）$d : \mathbb{R}^3 \to \mathbb{R}$，使其与 f 在 ∂S 上一致。函数 f 是按照第 4 节中描述的方法构建的，并且在 ∂S 的定义中隐含出现。

为了计算 SDF $d(\cdot)$，我们使用了在 [11] 中介绍的方法：它将 $d(\boldsymbol{x})$ 的试探解设为函数 $\text{sign}(f(\boldsymbol{x}))h(\boldsymbol{x};\boldsymbol{\theta})$，其中 $h(\cdot;\boldsymbol{\theta})$ 是一个全连接的深度神经网络，

sign(·) 是符号函数。将深度神经网络的最后一层乘以 $\operatorname{sign}(f(\boldsymbol{x}))$ 保证了 f 的零水平集得以保持。参数 $h(\cdot;\boldsymbol{\theta})$ 是通过最小化损失函数

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim D}\left(|\nabla_{\boldsymbol{x}} d(\boldsymbol{x};\boldsymbol{\theta})| - 1\right)^2, \tag{9}$$

获得的，其中 $\mathbb{E}$ 是期望，而 D 是在给定计算域上的均匀分布。自动微分用于计算空间导数以及关于参数 $\boldsymbol{\theta}$ 的导数。

图 7 显示了一个隐式建模的国际象棋兵，使用简单的基元（多项式）和布尔运算。左侧的图像说明了表面 $\{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x}) = 0\}$ 以及切片上 f 的填充轮廓图。右侧的图像显示了没有表面的 f 的填充轮廓图，以进一步说明物体内部的场行为。

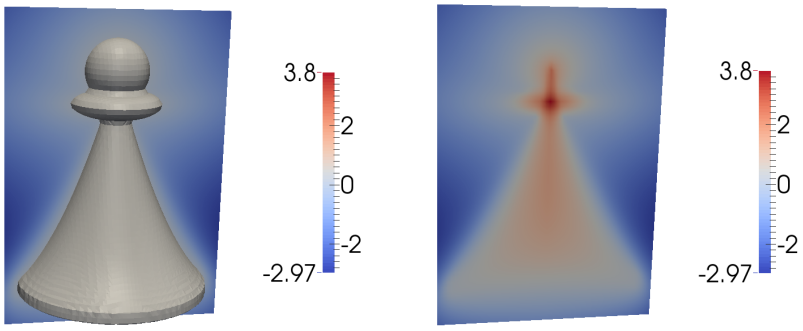

图 7: 左：f 的零水平集和切片上的填充等高线图。右：没有表面的切片上的填充等高线图。

图 8 显示了距离函数 $d(\boldsymbol{x})$ 的结果。比较填充等高线图与图 7，可以看出它对表面的距离函数提供了更精确的近似值。

5.4 参数形状拟合

我们展示了所提出的框架如何用于将参数模型拟合到数据，并在逆几何建模中使用 [13]。设 $f(\boldsymbol{x};\boldsymbol{p})$ 是一个参数模型，其中 $\boldsymbol{p}$ 控制对象的形状。对于给定的一组参数 $\boldsymbol{p}$，对应的实体由集合 $\{\boldsymbol{x} \in \mathbb{R}^3 : f(\boldsymbol{x};\boldsymbol{p}) \geq 0\}$ 给出。

令 $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^3$ 为输入点云，即在对象表面获取的三维点集。目标是找到参数 $\boldsymbol{p}$，使得与 f 对应的形状拟合输入点云。每个参数模型都有一

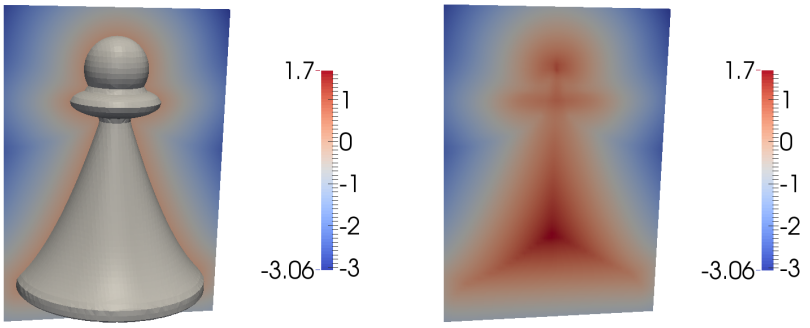

图 8: 左：$d(\boldsymbol{x}) = \text{sign}(f(\boldsymbol{x}))h(\boldsymbol{x}; \widetilde{\boldsymbol{\theta}})$ 的零水平集和切片上的填充轮廓图。右：没有表面的切片上的填充轮廓图。

组控制模型外观的形状参数。例如，图 10 中所示模型的参数对应于杆的频率和缩放比例。

为了将模型参数拟合到给定的数据集，我们定义拟合误差为

$$E(\boldsymbol{p}) = \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}_i; \boldsymbol{p})^2 \tag{10}$$

其中 $\boldsymbol{p}$ 是模型参数，$\boldsymbol{x}_i \in \mathbb{R}^3$ 是来自输入点云（或给定三角网格的顶点）的点。需要最小化 $E(\boldsymbol{p})$ 以获得 $\boldsymbol{p}$。

最小化是通过启发式方法和随机梯度下降的组合来完成的。我们使用正则进化的一种变体 [42] 作为启发式方法，以寻找接近 (10) 全局最小值的参数。正则进化是一种进化算法，在这种算法中，每次迭代都通过锦标赛选择选择一个父代，然后对其进行变异并重新插入种群。每次迭代都会移除种群中最老的一个个体。我们然后使用随机梯度下降（SGD）[44] 来改进解

$$\boldsymbol{p}_{i+1} \leftarrow \boldsymbol{p}_i - \nabla_{\boldsymbol{p}}\widetilde{E}(\boldsymbol{p}_i), \quad \text{where} \quad \widetilde{E} = \frac{1}{|I|}\sum_{i\in I} f(\boldsymbol{x}_i; \boldsymbol{p})^2,$$

其中 I 是 $\{1, \ldots, N\}$ 的一个随机子集，初始参数 p_0 通过正则化演化获得，并且模型关于形状参数 $\boldsymbol{p}$ 的导数通过自动微分计算。

我们以一个通过重复杆件构建的参数化微结构为例（参见图 10）。模型的参数控制杆件的频率及其尺寸。给定由 $40K$ 个点组成的点云作为输入，

我们将参数化模型拟合到输入的点云上。我们使用 $10K$ 次正则化进化迭代，种群大小为 100，样本大小为 10，随后进行 100 次 SGD 迭代。图示 9 和 10

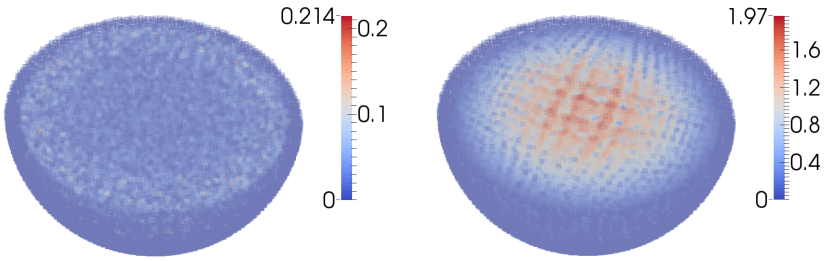

图 9: 拟合模型的点绝对误差。左：使用了正则化进化+SGD。右：仅使用 SGD。请注意，仅使用 SGD（右侧）拟合的模型出现了较大的误差。

显示了上述方法获得的结果。由 SGD 单独拟合的模型在输入点云的每个点上的绝对值误差（$|f(\boldsymbol{x}_i; \tilde{\boldsymbol{p}})|$ 对应于拟合参数 $\tilde{\boldsymbol{p}}$）如图 9 右图所示，而由上述正则化演化与 SGD 组合方法拟合的模型如图 9 左图所示。请注意，仅使用 SGD 拟合的模型相较于通过正则化演化和 SGD 拟合的模型产生了更大的误差。

由仅 SGD 拟合的模型对应的表面（中间图像）和正则化演化+SGD 对应的表面（左图像）如图 10 所示。最右边的图像同时显示了这两个表面，其中正则化演化+SGD 的结果以黄色显示，而仅 SGD 的结果以蓝色显示。

6 结论

我们描述了一个基于函数表示的可微几何建模框架。该系统建立在自动微分库的基础上，允许轻松计算并获得几何对象相对于空间以及形状参数的导数。我们展示了该框架的一些可能应用：曲率计算、归一化、符号距离估计和参数模型拟合。代码是开源的，我们希望它能够进一步扩展并在几何流水线中使用。

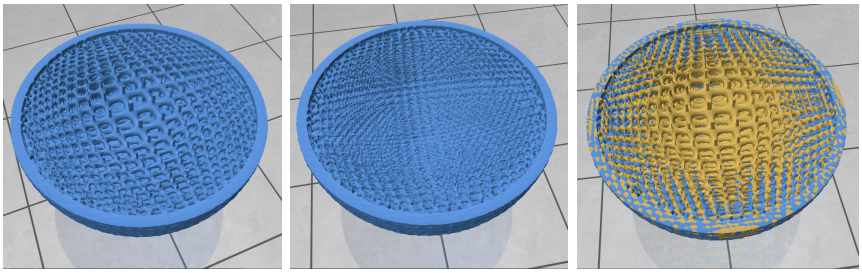

图 10: 通过拟合参数模型重构的对象。左：使用正则化演化+SGD。中：仅使用 SGD。右：两个模型一起展示。黄色：正则化演化+SGD；蓝色：仅使用 SGD。

参考文献

[1] OpenSCAD. `https://openscad.org/`.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Luke Anderson, Tzu-Mao Li, Jaakko Lehtinen, and Frédo Durand. Aether: An embedded domain specific sampling language for monte carlo rendering. *ACM Transactions on Graphics (TOG)*, 36(4):1–16, 2017.

[4] Atılım Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.*, 18(1):5595 – 5637, January 2017.

[5] Arpan Biswas and Vadim Shapiro. Approximate distance fields with non-vanishing gradients. *Graphical Models*, 66(3):133–159, 2004.

[6] Arpan Biswas, Vadim Shapiro, and Igor Tsukanov. Heterogeneous material modeling with distance fields. *Computer Aided Geometric Design*, 21(3):215–242, 2004.

[7] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *SIGGRAPH Comput. Graph.*, 24(2):109 – 116, February 1990.

[8] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, and et al. JAX: composable transformations of Python+NumPy programs. `http://github.com/google/jax`, 2018.

[9] Richard Cartwright, Valery Adzhiev, Alexander A Pasko, Yuichiro Goto, and Tosiyasu L Kunii. Web-based shape modeling with hyperfun. *IEEE Computer Graphics and Applications*, 25(2):60–69, 2005.

[10] Zachary DeVito, Michael Mara, Michael Zollhöfer, Gilbert Bernstein, Jonathan Ragan-Kelley, Christian Theobalt, Pat Hanrahan, Matthew Fisher, and Matthias Niessner. Opt: A domain specific language for non-linear least squares optimization in graphics and imaging. *ACM Transactions on Graphics (TOG)*, 36(5):1–27, 2017.

[11] Pierre-Alain Fayolle. Signed distance function computation from an implicit surface. *ArXiv preprint arXiv:2104.08057v1 [cs.GR]*, 2021.

[12] Pierre-Alain Fayolle and Markus Friedrich. A survey of methods for converting unstructured data to csg models. *Computer-Aided Design*, 168:103655, 2024.

[13] Pierre-Alain Fayolle, Alexander Pasko, Elena Kartasheva, Christophe Rosenberger, and Christian Toinard. *Automation of the Volumetric Models Construction*, pages 214–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[14] Pierre-Alain Fayolle, Alexander Pasko, Benjamin Schmitt, and Nikolay Mirenkov. Constructive heterogeneous object modeling using signed approximate real distance functions. *Journal of Computing and Information Science in Engineering*, 6(3):221–229, 11 2005.

[15] Pierre-Alain Fayolle, Benjamin Schmitt, Yuichiro Goto, and Alexander Pasko. Web-based constructive shape modeling using real distance functions. *IEICE TRANSACTIONS on Information and Systems*, 88(5):828–835, 2005.

[16] Ron Goldman. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7):632–658, 2005.

[17] Abel Gomes, Irina Voiculescu, Joaquim Jorge, Brian Wyvill, and Callum Galbraith. *Implicit Curves and Surfaces: Mathematics, Data Structures and Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[18] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, USA, second edition, 2008.

[19] Yu Guo, Milos Hasan, Lingqi Yan, and Shuang Zhao. A bayesian inference framework for procedural material parameter estimation. *Computer Graphics Forum*, 39(7):255–266, 2020.

[20] Christian Hafner, Christian Schumacher, Espen Knoop, Thomas Auzinger, Bernd Bickel, and Moritz Bächer. X-cad: optimizing cad models with extended finite elements. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[21] Lu Han and Shunai Che. An overview of materials with triply periodic minimal surfaces and related geometry: From biological structures to self-assembled systems. *Advanced Materials*, 30(17):1:705–708, 2018.

[22] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[23] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *ArXiv preprint arXiv:1910.00935*, 2019.

[24] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *Arxiv prepring arxiv:2006.12057 [cs.CV]*, 2020.

[25] Matt Keeter. LibFive. `https://github.com/libfive/libfive`.

[26] Matt Keeter. Massively parallel rendering of complex closed-form implicit surfaces. *ACM Trans. Graph.*, 39(4), July 2020.

[27] Sylvain Lefebvre. IceSL: A GPU accelerated CSG modeler and slicer. In *18th European Forum on Additive Manufacturing (AEFA' 13)*. AEFA, 2013.

[28] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

[29] Tzu-Mao Li, Michaël Gharbi, Andrew Adams, Frédo Durand, and Jonathan Ragan-Kelley. Differentiable programming for image processing and deep learning in halide. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[30] Tzu-Mao Li, Michal Lukác, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[31] Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014.

[32] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[33] Dougal Maclaurin. *Modeling, inference and optimization with composable differentiable procedures*. PhD thesis, Harvard, 2016.

[34] Orest Mykhaskiv, Mladen Banović, Salvatore Auriemma, Pavanakumar Mohanamuraly, Andrea Walther, Herve Legrand, and Jens-Dominik Müller. Nurbs-based and parametric-based shape optimization with differentiated cad kernel. *Computer-Aided Design and Applications*, 15(6):916–926, 2018.

[35] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006.

[36] Alexander Pasko, Valery Adzhiev, Richard Cartwright, Eric Fausett, Anatoly Ossipov, and Vladimir Savchenko. Hyperfun project: A framework for collaborative multidimensional f-rep modeling. In *Eurographics/ACM SIGGRAPH Workshop Implicit Surfaces' 99*, pages 59–69, 1999.

[37] Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The visual computer*, 11(8):429–446, 1995.

[38] Alexander Pasko, Oleg Fryazinov, Turlif Vilbrandt, Pierre-Alain Fayolle, and Valery Adzhiev. Procedural function-based modelling of volumetric microstructures. *Graphical Models*, 73(5):165–181, 2011.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, and et al. Pytorch: An imperative style, high-performance deep learning library. *ArXiv preprint arXiv:1912.01703*, 2019.

[40] Barak A Pearlmutter and Jeffrey Mark Siskind. Reverse-mode ad in a functional framework: Lambda the ultimate backpropagator. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 30(2):1–36, 2008.

[41] Inigo Quilez. Modeling with distance functions. `http://iquilezles.org/www/articles/distfunctions/distfunctions.htm`.

[42] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789. AAAI, 2019.

[43] A. Ricci. A Constructive Geometry for Computer Graphics. *The Computer Journal*, 16(2):157–160, May 1973.

[44] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[45] V. L. Rvachev. *Methods of Logic Algebra in Mathematical Physics*. Naukova Dumka, 1974. In Russian.

[46] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, page 43, New York, NY, USA, 2007. ACM, Association for Computing Machinery.

[47] Vadim Shapiro. Semi-analytic geometry. *Acta Numerica 2007: Volume 16*, 16:239–303, 2007.

[48] Liang Shi, Beichen Li, Milos Hasan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. Match: differentiable material graphs for procedural material capture. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020.

[49] Gabriel Taubin. Distance approximations for rasterizing implicit curves. *ACM Transactions on Graphics*, 13:3–42, 1994.

[50] A. Tewari, O. Fried, J. Thies, V. Sitzmann, and et al. State of the art on neural rendering. *Computer Graphics Forum*, 39(2):701–727, 2020.

[51] Kiril Vidimce, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. Openfab: A programmable pipeline for multi-material fabrication. *ACM Transactions on Graphics*, 32, July 2013.

[52] Brian Wyvill, Eric Galin, and Andrew Guy. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158, 1999.