

Iterative Poisson Surface Reconstruction (iPSR) for Unoriented Points

Hou Fei, Wang Chiyu, Wang Wencheng, Qin Hong, Qian Chen and He Ying

摘要

泊松表面重建（PSR）由于其效率高、简单性和鲁棒性，自 2013 年发表以来就是一种从三维点云中重建水密表面的广泛运用的技术。然而，现有的 PSR 方法和随后的变体只适用于定向点云输入，即有法向量的点云。本篇论文旨在验证和改进 PSR，迭代式泊松重建（iPSR）可以完全消除输入点云的法向量的要求，并以迭代的方式进行。在每次迭代中，iPSR 将直接从上次迭代中获得的曲面计算法向量来作为输入点云的新的法向量，以生成质量较好的新的曲面。大量的定量实验证明：iPSR 算法即使在随机初始化法向量的条件下，也能在 5-30 次迭代中收敛。

关键词：重建；点云；迭代

1 引言

现有的表面重建技术分为隐式和显示重建的方法，而隐式重建的方法本质上是生成一个距离场和可以用于提取等值面的隐式函数来重建曲面。原则上，隐式重建的方法可以保证表面重建的水密性，但其中生成的物体的表面可能不会通过输入的样本点，但对有噪声的点云输入具有更强的鲁棒性。隐式方法又分为局部重建和全局重建两种方法：局部重建的方法首先拟合一个局部曲面，然后使用贪心算法将局部拟合的曲面逐步扩展至全局。全局重建方法通过优化一个全局函数来同时推断出所有的法向量，最后重建出曲面。显式的表示方法和学习算法依赖于点云、参数化网格这些表示直接参数化几何表面的方式。显式表示通常是轻量级的，需要很少的参数来表示几何形状，但需要离散化，平均化，往往难以表示。

泊松重建技术^[1]自 2013 年发表以来就是一个被广泛使用全局隐式重建的技术，它可以很好的在有向点云上生成水密的物体表面，但是它却对输入点云的法向量有很高的要求。正是因为对输入法向量的高依赖性限制了它的使用场景。本篇论文改进了原本泊松重建需要有向点的条件，也即不需要输入点云的法向量也可以进行表面重建。它只需要随机初始化法向量，而后不断地迭代运用截断式泊松重建的方法重建出表面，然后重新采样点并更新法向量即可。选择本论文不仅可以学习到传统泊松重建的表面重建方法，对三维重建邻域有好的理解，还能学习到前沿的技术进展，学习新的重建方法和重建思想。

2 相关工作

传统的表面重建技术中，SDF（符号距离场）是基于离散化体素对物体或者场景表面的一种表达，而现实中的表面是连续的，是否可以通过深度学习的方法在空间中的连续场来表达物体表面，有了数据驱动训练的方法当然不仅可以让表面重建，也可以做形状补全。近年来，深度学习作为一种新兴的方法，在点定向和表面重建方面显示出了它的运用前景。但本质上，基于深度学习的方法总是依赖于

某些数据集的训练过程，这些数据集是专门为特定类型的模型及其后续的重建任务量身定制的。同时，在计算机图形学的物体表面重建的领域当中，与 2D 图像不同的是，在 3D 中没有既能满足高分辨率的几何形状又满足内存和运行时间开销小的表示方法。因此，许多最先进的基于学习的三维重建方法只能代表非常粗糙的三维几何图形，或被限制在一个有限的领域内。

如何找到一个好的物体表面的表达方式是基于学习的三维重建领域想要解决的问题之一，好的表达方式能够更好地进行表面重建，现有的方法有基于点的表达方式、基于网格的表达方式和基于体素的表达方式：

对于基于点云的表达方式，虽然点云的表达更接近输入数据的点云形式，而且典型工作如 pointnet^[2]，在提特征方面以及做分类分割检测的工作合适，后续如 pointnet++^[3]虽然可以更好地捕捉到点云的全局与局部特征，但这些都没改变此类方法在物体形状的表达方面受限的情况，因为基于点云的学习方法没有描述物体表面的拓扑结构，不适合生成水密的表面。

基于网格的表达方式^[4]的工作虽然能够对物体表面有更加精确的表达，但是这样复杂度也随之变得很高，而且只能用固定的网格拓扑结构来重建表面，可扩展性不强。因此，虽然网格的表示形式比起点云来说，更能精确表达对待重加的物体表面，但因为其局限于固定的拓扑结构，往往效果还没基于点云的表达方式好。

基于体素^[5]的表达方式可拓展性也不强，而且精度表达有限，当然有很多学者提出使用八叉树或者体素哈希的方法，但是只是在数据结构层面的改进，自身固有的缺陷（体素网格的分辨率限制）没有解决。

2.1 基于隐式函数的方法

目前不少工作针对基于深度学习的三维重建方法提出了使用如符号距离函数（SDF）来表达点云信息，并基于此表达训练出网络模型，从而达到使用深度学习来进行三维重建的目的。如 Park 等人^[6]提出的 DeepSDF 的曲面隐式表达方式，可以表达所有的形状类型，同时使用与其他用网格来离散化 SDF 的方法不同，他们使用的是一个生成模型来产生这样一个连续的 SDF 场。DeepSDF 很好地适应了点云输入的无序性的特点，不过它无法处理大型的场景输入，很难有一个好的网络训练出来，而且对于有噪点的数据输入，用这样一种网络训练的方法不能很好的重建出物体的表面。

还有一些工作希望能将传统泊松重建中的求解隐式函数的步骤用于网络学习当中，弥补点云输入和隐式函数之间的鸿沟。如 Peng 等人^[7]提出了一种可微的泊松求解器，由于泊松求解器的可微性，输出网格的损失或中间指标网格的损失产生的梯度可以有效地反向传播，以更新定向点云表示。但本论文的方法目前仅限于小型场景，因为内存消耗会随着指示函数网格分辨率的增加成指数级增长。但论文中也提出了以滑动窗口的方式处理场景和空间自适应的数据结构（如八叉树）的方法扩展到更大的场景中的可能。

2.2 基于体素网格的方法

在三维重建邻域汇中由于内存和计算效率的限制，不允许表示任意拓扑的高分辨率几何形状。因此许多最先进的基于学习的三维重建方法只能表示非常粗糙的三维几何图形或仅限于一个受限的领域。Mescheder 等人^[8]引入了一种新的三维几何表示法的，即三维空间的占用概率网络。与现有的方法特别是体素网格的方法相比，体素网格会受到它分辨率的限制，尤其是随着网格分辨率的增加会使

得内存消耗和时间都变得非常大，而占用网络不受三维空间离散化的限制，因此可以用来表示真实的高分辨率网格。本论文的实验表明，这样一种占用网络具有很强的表达能力，可以有效地用于监督学习和无监督学习。本论文提出的占用网络是一个非常有用的工具，可以应用于广泛的三维任务当中。

对于此论文中提出的多层次分辨率的等值面提取的方法，需要注意的是，初始化网格的分辨率十分重要：如果初始化网格的分辨率不够高，那么会因为分辨率太低的问题导致初始化时不能让网格准确的提取到正确的区域，即不能正确识别出物体的边界；而如果初始化网格分辨率过高，那么虽然能够正确识别出物体表面的边界，但是这也会造成多余的计算浪费以及内存消耗。

3 本文方法

3.1 本文方法概述

本篇论文改进了原本泊松重建需要有向点的条件，也即不需要输入点云的法向量也可以进行表面重建。它只需要随机初始化法向量，而后不断地迭代运用截断式泊松重建的方法重建出表面，然后重新采样点并更新法向量即可。如图 1 所示是一个实例的迭代过程：



图 1：迭代过程图

3.2 方法原理

虽然本论文使用的算法在执行过程中对于泊松重建的只是当做黑盒来使用，但出于能够更加深刻地理解本篇论文提出的迭代式泊松重建 iPSR 的目的，还是应该能够对泊松重建有更加深刻的理解。泊松重建提出了一种隐式方法进行表面重建的方法，它首先构造隐式函数为内 1 外 0 的指示函数，对于这样一种指示函数，论文中指出此指示函数的梯度和输入点云的法向量相等，所得到的公式 (1) 如下所示：

$$\Delta \chi = \nabla \cdot V \quad (1)$$

因而可以通过表面有向点的法向量与模型的指示函数的梯度之间的关系来求解出指示函数，最后通过 marching-cube^[9]的方法将表面重建出来。他们之间的关系如图 2 所示。

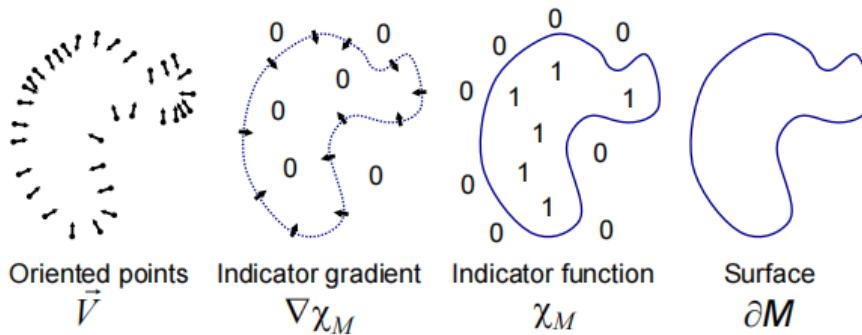


图 2：指示函数和法向量关系图

本论文的方法是首先运用随机初始化的方式，对输入的无向点云中的每个点随机初始化出点的法向量，随后运用泊松重建的方法，对这样一个经过随机初始化得到有向点云进行泊松重建。随机初始

化获得的法向量进行重建得到的表面会有很多本不应该相互连接，却互相连接的小区域。如图 3 中的例子所示：

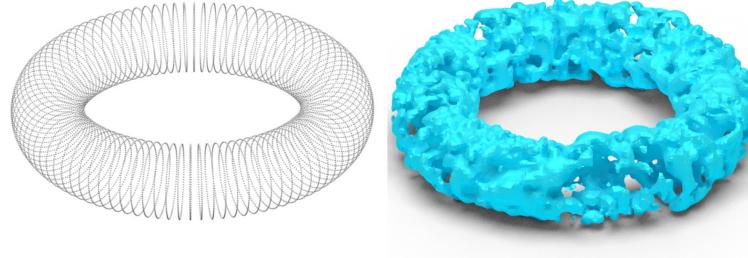


图 3：随机初始化结果例图

若一条射线从重建得到的等值面和物体内部射出（要求方向垂直于物体表面，其实也就是沿着采样点的法线），那它将交于 $2n+1$ 个点，同时奇数交点上的法向朝向物体内部，而偶数交点的法向朝向物体外部。通过平均 $2n+1$ 个点我们就可以获得朝向物体内部的法向（对于泊松重建，我们需要的采样点的法线方向应该是朝向物体内部的）。

论文中还定义了一些奇数和偶数层。简单来说就是在交点的局部区域上，定义其等值面为一个层，并使用这个层上重建出来的三角网格的法向来更新 k 个临近的采样点的。对于奇偶层，奇数层能够产生正确的朝向物体内部的法向，而偶数层也会在后续迭代过程中变为奇数层。同时随机初始化能够保证生成的大多数层为奇数层，这样一种方式能保证总体的法向迭代方向是正确。寻找 k 临近的二维例子如图 4 所示：

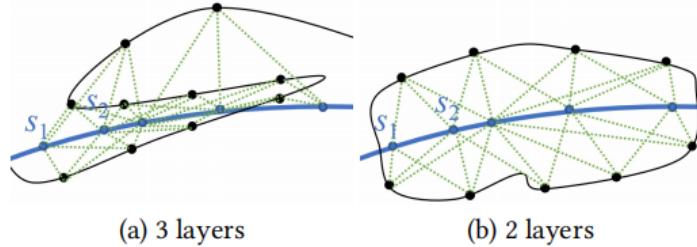


图 4：算法二维示意图

3.3 算法伪代码

本篇论文的算法中，最重要的是它更新输入点云的法向量的方式：通过三角网格中每个三角面片的中心找到与其最近的 k 个输入点云（建立 k 个点和三角网格中心的联系）。通过面积加权的方式更新每个输入点云的法向量（即输入点云仅用和它有联系的三角面片的法向量来进行面积加权平均的方向更新法向量）。三角网格的法向通过自身网格顶点即可叉乘获得。再利用新生成的法向量进行新一轮的迭代（更新后的有向点云再次使用泊松重建的方式重建出曲面），经过一定的迭代次数或者仅当百分比 0.1 的变化量最大的法向量的平均变化程度小于阈值时再停止迭代。

本篇论文提出的算法的关键伪代码如下：

Algorithm 1 The core part of iPSR.

Input: unoriented points $\{p_1, p_2, \dots, p_m\}$
Output: A watertight surface approximating points

construct a kd-tree for S for nearest sample searching;

initialize normal for each sample $s_i \in S$ randomly.

while not convergence or exceeding maximum number of iterations **do**

- compute indicator function χ by applying screened PSR with parameter to S ;
- extract the iso-surface \mathcal{F} with iso-value $\frac{1}{n} \sum_{j=1}^n \chi(s_j)$ by marching cubes;
- for** each sample $s_j \in S$ **do**
 - $s_j.\text{face list} \leftarrow \emptyset$
- end**
- for** each triangular face $f_j \in \mathcal{F}$ **do**
 - $\vec{n}_j \leftarrow f_j$'s inward normal, i.e., towards interior of the component f_j belonging to;
 - $a_j \leftarrow$ the area of triangle f_j ;
 - use kd-tree to find top- k samples in S that are closest to f_j ;
 - add f_j to the associated face list of each of the k samples;
- end**
- for** each sample $s_j \in S$ **do**
 - $\vec{n}(s_j) \leftarrow \sum_{i=1}^{m_j} a_{j_i} \vec{n}_{j_i}$;
 - normalize $\vec{n}(s_j)$;
- end**
- $d \leftarrow$ the average of top 0.1 % normal difference of samples between the previous and the current iterations;
- if** $d < \delta$ **then**
 - convergence $\leftarrow true$
- end**

end

apply screened PSR to the sample points s_j with predicted normals $\vec{n}(s_j)$ and return the iso-surface \mathcal{F} extracted from the computed indicator function.

一些其他的初始化方式：除了随机初始化法向量的方法之外，还有基于采样点的可见性的法向量的方法。它的过程是：建立一个单位立方体（输入点云放入单位立方体当中）和同心的边长为 3 的立方体，立方体上有 26 个视点，假设每个视点往其中的一个点看，设置一定的遮挡因子，对可视的方向进行平均作为点的初始法线方向。通过这样的初始化能在输入点云的点较多的情况下加快迭代的速度。

4 复现细节

对于这样一种随机初始化点云以后迭代式的更新法向的重建算法，希望能够探究一下它为什么能够达到很好的效果。虽然原论文中并未给出此算法的数值分析，但也可以通过一些对比实验的方式来探究原理。迭代算法中最重要的部分就是如何更新它的法向，因此我另一通过对比不同的更新输入点云法向量的方式做一个对比。

针对原论文中通过重建出来的三角网格的中心来寻找 k 临近个点并建立联系，后续在依据这些联系来面积加权式的更新法向量的方法。可进行的对比实验有：1、不用面积加权平均的方式但使用 k 临近的算法（Direct-average），就是说直接对不同面片上的法向量单位化后求平均即可，2、使用

距离加权平均和 k 临近的算法 (distance-weighted)，对于单位面片中心点求与其 k 临近个输入采样点求其距离，使用距离的倒数作为权重来更新法向量，3、使用范围临近的面积加权平均更新法向量 (Radius-search)，即先求三角面片中心距离最近的点，再去所有中心中所求距离的最大值作为范围临近中球的半径，以此来采样更新法向量。

在这里对 Radius-search 的方式作一个解释：它依据重建出来的三角网格的中心点寻找半径为 r 的大小内的输入采样点，并建立联系，随后进行面积加权的方式更新法向量。若范围内没有相应的点，则不作更新。

4.1 与已有开源代码对比

针对上述提到的不同更新输入点的法向量的代码，在原有开源代码的基础上，增设不同的法向量更新方式的代码。下面以伪代码的形式列出这三种方式与原论文中的算法的不同的部分：

对于不使用面积加权平均方式的算法伪代码不同的部分如下所示：

Algorithm 2 Direct-average.

Other part is the same as iPSR.

for each triangular face $f_j \in \mathcal{F}$ **do**

$\vec{n}_j \leftarrow f_j$'s inward normal, i.e., towards interior of the component f_j belonging to;

$\vec{n}_j \leftarrow$ normalize \vec{n}_j ;

use kd-tree to find top- k samples in S that are closest to f_j ;

add f_j to the associated face list of each of the k samples;

end

for each sample $s_j \in S$ **do**

$\vec{n}(s_j) \leftarrow \sum_{i=1}^{m_j} \vec{n}_{j_i}$;

normalize $\vec{n}(s_j)$;

end

在具体的 C++ 代码实现方面，这样一种更新方式对代码的改动不会特别多，只需要每次依据三角面片中的向量叉乘出来的法向进行单位化即可。

使用距离加权平均方式 (distance-weighted) 更新输入点云的法向量的方法与源代码的不同的部分如下所示：

Algorithm 3 Distance-weighted.

Other part is the same as iPSR.

for each triangular face $f_j \in \mathcal{F}$ **do**

$\vec{n}_j \leftarrow f_j$'s inward normal, i.e., towards interior of the component f_j belonging to;

$\vec{n}_j \leftarrow$ normalize \vec{n}_j ;

use kd-tree to find top- k samples in S that are closest to f_j ;

add f_j to the associated face list of each of the k samples;

end

for each sample $s_j \in S$ **do**

$d_{j_i} \leftarrow$ compute the distance between sample s_i and face f_j ;

$\vec{n}(s_j) \leftarrow \sum_{i=1}^{m_j} \frac{1}{d_{j_i}} \vec{n}_{j_i}$;

normalize $\vec{n}(s_j)$;

end

这里使用的距离加权平均使用的权重是采样点与面片中心之间距离的倒数的值，也就是说距离越大权重占比越少，从某种程度上来说，距离采样点位置越远的面片中心点理应有更小的权重影响。在实现代码过程中，使用 kd-tree 求出距离最近的采样点，还要注意使用 OpenMP 并行计算三角形面片的中心和输入点云的之间的距离，这样在程序运行时才不会过于慢。

对于不使用范围临近采样的算法，它的伪代码与原算法伪代码的不同的部分如下所示：

Algorithm 4 Radius-search.

Other part is the same as iPSR.

```

initial  $d_{max} = +\infty$ 
for each triangular face  $f_j \in \mathcal{F}$  do
     $\vec{n}_j \leftarrow f_j$ 's inward normal, i.e., towards interior of the component  $f_j$  belonging to;
     $\vec{n}_j \leftarrow \text{normalize } \vec{n}_j;$ 
    use kd-tree to find top-1 samples in  $S$  that are closest to  $f_j$ ;
     $d_t \leftarrow \text{compute the distance between closest point and } f_j$ 
    if  $d_t < d_{max}$  then
        |  $d_{max} \leftarrow d_t$ 
    end
end
for each triangular face  $f_j \in \mathcal{F}$  do
    use kd-tree to find sample inside the radius( $d_{max}$ )
    add  $f_j$  to the associated face list of each of the samples;
end
for each sample  $s_j \in S$  do
     $\vec{n}(s_j) \leftarrow \sum_{i=1}^{m_j} a_{ji} \vec{n}_{j_i};$ 
    normalize  $\vec{n}(s_j);$ 
end
```

在代码实现方面：主要是通过使用 kd-tree 来进行范围临近的采样，指定采样的半径即可将点采样出来。需要指出的是，对于在指定半径范围之内没用对应采样点的情况的处理为：不对齐进行更新，沿用上一次使用的随机初始化后的结果来进行下一步计算。

4.2 实验环境搭建

本论文的代码实验环境搭建并不难，只需将代码 clone 下来之后，通过 Makefile 或者在 windows 环境下使用 Visual Studio 进行项目编译即可。但是相比于在 windows 环境下：使用 Linux 系统运行此代码效率更高。因此接下来的实验都在 ubuntu 的环境下进行。编译完成代码之后，只需在终端输入相应的输入点云 ply 文件路径、输出点云的 ply 文件路径，并且指定迭代次数之后即可运行（用我此次补充的代码还可以指定不同的模型进行采样）。

4.3 界面分析与使用说明

在本次实验中，我在 windows 环境下搭建了对应于本论文代码的 UI 界面，界面 UI 如下图 5 所示：

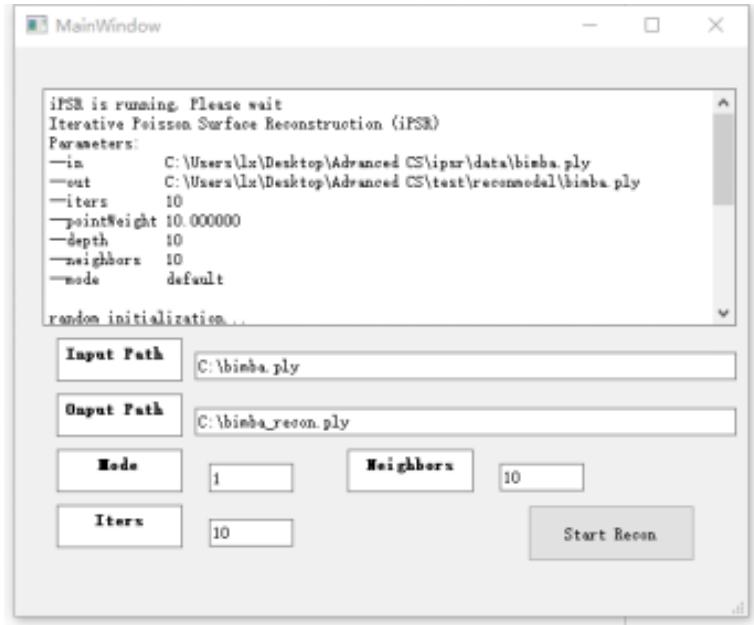


图 5: 操作界面示意

本次设计的 UI 的使用方式如下表 1 所示:

表 1: 使用说明

输入	作用	使用说明
Input Path	输入文件的路径	必须是绝对路径, 同时要求格式是.ply 的无法向量的文件
Output Path	输出文件的路径	与 Input-path 中的要求一致
Mode	选择更新法向量的方式	1-default 2-without_area-weighted 3-distance-weighted 4-radius-search
Iters	迭代的最大次数	即迭代次数达到最大迭代次数时必停止 (默认为 10)
Neighbors	K 临近算法中的 k 值	默认 10, 使用 Radius-search 的方法时无需设置
Start Recon	开始重建	设置完上诉参数之后才可点击重建

5 实验结果分析

为测试提出的不同的更新法向量的方式和原论文中的方法在效果上的不同之处, 代码实验部分通过对比不同的采样更新方式在各种不同的模型之间的效果来进行分析。

首先是对于能够很好表达模型 (没有非常扭曲或者纤细的结构) 输入的点云是在模型表面上均匀采样的情况, 对于比 Bimba 和 Stanford bunny 的点云输入数据, 他们没有太大的区别, Stanford bunny 的不同模式运行结果如图 6 所示:

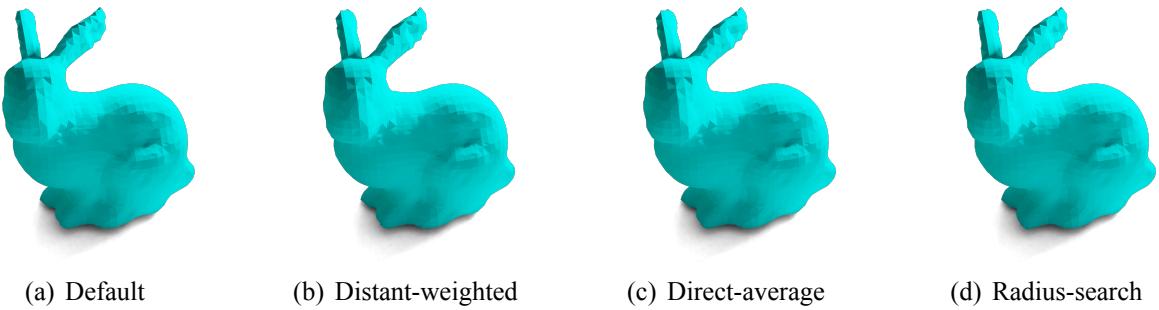


图 6: Stanford Bunny

然而上述代表的是一些输入点云较好的采样表明以及模型本身比较简单的情况, 而对于具有很多

纤细结构且模型非常复杂的情况，使用论文中的方法一般可以获得较好的结果，而对于所提出的其他方法来说，效果差强人意，如图 7 所示：

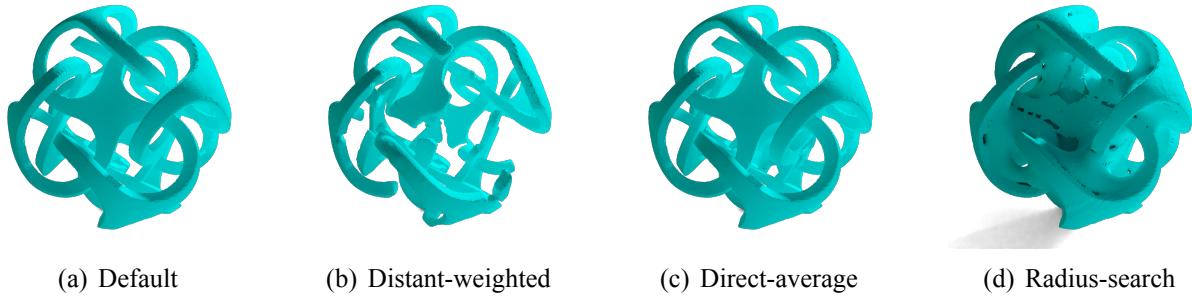


图 7: MetatronPointclouds

由图中结果可以看出使用原论文的算法效果挺好，而使用距离加权平均的方式生成的模型有很多地方没链接上，范围临近采样的方法将本不应该连接起来的部分连在了一起，不过上述结果只是在迭代次数为 10 次时的不同方式下的结果。考虑可能是因为迭代次数不够的问题，在此模型增加了 30 次迭代运算之后的结果。结果如图 8 所示：

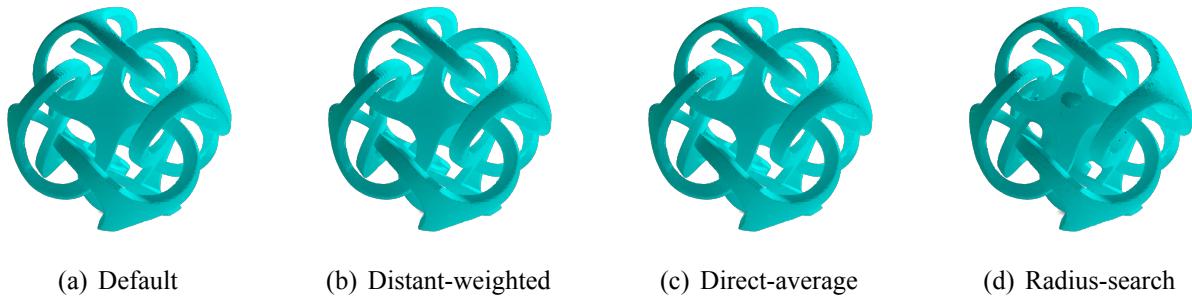


图 8: MetatronPointclouds after 30 iterations

经过 30 次迭代之后，直接平均更新法向量的方式也和原论文中的方法一样有很好的效果，因此可以推测：使用直接平均更新法向量的方式可以捕捉到相邻法向量的信息且不会像范围临近那样捕捉到多余的信息而使得本不该闭合的地方闭合，但是收敛速度慢。从结果上来看距离加权平均的方式也是相同的情况，而范围临近的采样方式依然没达到好的效果。

同样的，在 PentagonalIcositetrahedron 模型下的实验结果也和上述模型一样，10 次算法迭代的结果差强人意，30 次迭代之后就好很多，结果如图 9 所示：

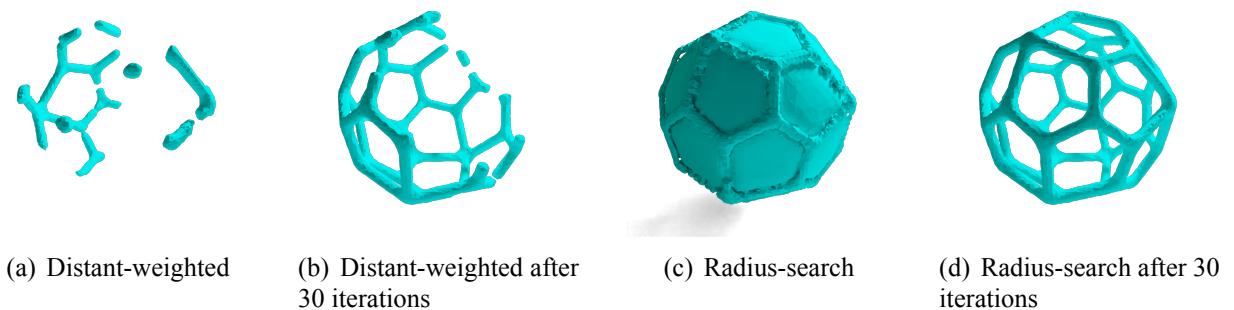


图 9: PentagonalIcositetrahedron

但也有范围临近表现的更好的模型的情况：如在 woodfish 上，使用范围临近采样的方式能够更好地使用局部点云的法向量，如图 10 所示：鱼尾巴的地方，鱼尾巴的地方使用范围临近采样的方法不会产生缺块，而使用其他采样更新的方式，即使在 30 次迭代之后也会在这个地方产生缺块。因此可以

分析得到：虽然范围临近采样的方式能够更多地捕捉到周围法向量的信息，当时常常会过分地使用过多的点的法向量，这样收敛的速度就会受到影响。要经过更多的迭代次数之后才能有较好的结果。

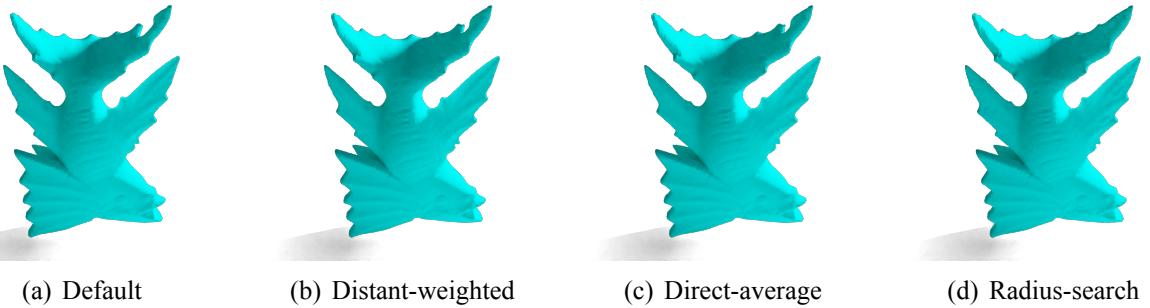


图 10: Woodfish

6 总结与展望

本篇论文扩展了经典的泊松曲面重建的方法，消除了它对点取向的要求。提出了一种简单而有效的定向策略，并证明了即使使用随机初始化的点的法向量，用本论文提出的迭代的泊松重建的方法也可以得到视觉上水密、光滑的表面。本论文提出的迭代泊松重建的方法继承了泊松重建的可伸缩性和鲁棒性特性，对于稀疏和密集的输入点云都很有效。而论文提出的方法在输入点云是采样率很低的薄型结构上的效果会不好，解决的方式只能依靠增加采样率来缓解。同时，此方法只能用于生成封闭的曲面结果，对于开放的曲面模型，必须通过后处理来消除模型中生成的不必要的填充。且由于泊松重建会将曲面顺滑过头的特性，使得本论文的方法不能生成具有锐利特征的曲面结构。

在我提出的不同的采样、加权更新的方式之中：在输入点云是较好的采样且基本上能够覆盖表面的低属模型之中，各个方法在不同的迭代次数之后都能得到一个较好的效果，不过使用范围临近采样的方式在这种情况下收敛的更快，而使用距离加权平均的方式收敛的慢。用距离加权平均以及 k 临近的算法在高属模型之中取得效果不好，原因是采用距离加权平均的方式不能很好的改变输入点云的法向，使其纠正为想要的法向量方向，因而它迭代收敛的速度比其他方法都要慢。而使用范围临近采样的方式虽然迭代收敛的快，但因为它过分的使用过多的点来更新输入点的法向量，因而在高属模型中常常不能生成想要的形状。而反观论文中的方法：它在大多数模型中都能取到较好的效果。当然因为范围临近充分采样了周围的点，因此在部分模型下它的首先速度要比原论文中的方法好。

iPSR 的迭代式更新法向量的思路非常的创新，而且效果非常的好。应用的场景也十分地广泛。但是由于每次迭代更新完法向量之后都需要运用 Marching-Cube^[9] 的算法来生成物体的表面以进一步进行法向量的更新，因而当输入的点云十分的多的时候就会使得算法运行的特别慢。因此我认为可以有的改进思路为：可以在每次迭代法向量然后使用泊松重建的算法时不使用 Marching-Cube 的算法来将表面重建出来，而是直接使用泊松重建得到的指示函数，由这个指示函数直接计算梯度，并以此来更新点的法向量。通过这样的一种方式可以使得 iPSR 的算法执行效率更加地快速。不过这就要与本论文中使用的从生成出来的曲面上的三角网格中心的法向量来更新采样点法向量的方式有所不同：需要从泊松重建构建的八叉树网格当中用网格顶点代替 iPSR 中的三角网格中心点，这样才能适配直接从指示函数中更新法向量的方式。

最后论文中并没有给出严格的数学推导与分析，且没有证明这样一种迭代的算法最后一定会收敛，当然这也是后续论文可以探究的地方。

参考文献

- [1] KAZHDAN M, HOPPE H. Screened poisson surface reconstruction[J]. ACM Transactions on Graphics (ToG), 2013, 32(3): 1-13.
- [2] QI C R, SU H, MO K, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 652-660.
- [3] QI C R, YI L, SU H, et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space [J]. Advances in neural information processing systems, 2017, 30.
- [4] KANAZAWA A, TULSIANI S, EFROS A A, et al. Learning category-specific mesh reconstruction from image collections[C]//Proceedings of the European Conference on Computer Vision (ECCV). 2018: 371-386.
- [5] RIEGLER G, OSMAN ULUSOY A, GEIGER A. Octnet: Learning deep 3d representations at high resolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 3577-3586.
- [6] PARK J J, FLORENCE P, STRAUB J, et al. Deepsdf: Learning continuous signed distance functions for shape representation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 165-174.
- [7] PENG S, JIANG C, LIAO Y, et al. Shape as points: A differentiable poisson solver[J]. Advances in Neural Information Processing Systems, 2021, 34: 13032-13044.
- [8] MESCHEDER L, OECHSLE M, NIEMEYER M, et al. Occupancy networks: Learning 3d reconstruction in function space[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 4460-4470.
- [9] LORENSEN W E, CLINE H E. Marching cubes: A high resolution 3D surface construction algorithm [J]. ACM siggraph computer graphics, 1987, 21(4): 163-169.