基于深度学习的三维重建结果的可视化

陈刚 2021214038

黎思宇 2021312592

沈冠霖 2021312593

问题背景

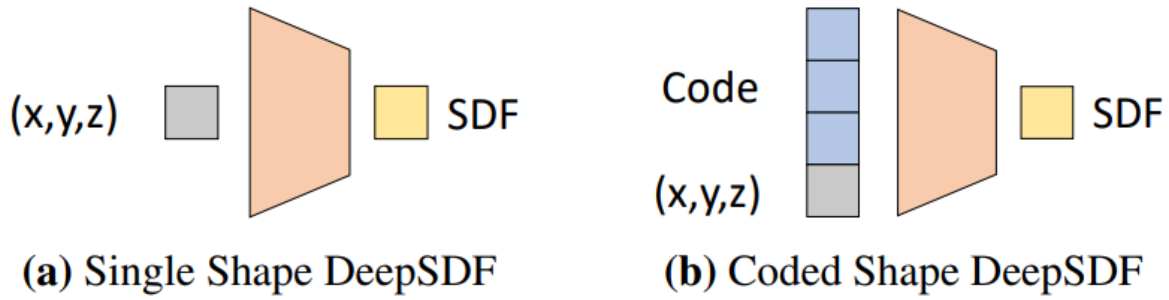
问题驱动及解决思路

背景

近些年，使用深度学习方法重建物体三维结构引起了广泛关注，在虚拟现实、增强现实、游戏等领域有巨大的应用潜力。基于深度学习的三维重建方法大多数采用编码-解码器结构，自 DeepSDF[1] 以来也有许多工作采用自解码结构。这些方法都会在中间步骤得到物体的潜在编码（latent code)，后续可以通过解码器网络重建出最终的三维结构。仅仅使用测试集上的全局定量指标，例如 Chamfer Distance、IoU 等，虽然能一定程度反映重建好坏，但是并不一定能代表视觉效果。同时，较大的数据集给全面分析带来了挑战。因此，我们的项目致力于设计一个系统，它能够更加直观展示三维重建的结果，能对 latent space 进行可视化，结合定量指标以及视觉结果对比，从而进行全面细致的分析，发现算法的问题，提高算法表现。

问题解决方案

我们使用 DeepSDF 网络学习物体表面的有向距离场，对 DeepSDF 网络得到的结果进行可视化。DeepSDF 方法将深度隐式场作为物体的三维表示，研究经典的 DeepSDF 算法的结果有利于提高我们对隐式方法的理解。DeepSDF 的基本网络结构如下所示：



(a) Single Shape DeepSDF　　(b) Coded Shape DeepSDF

对于三维空间中的点 P(x,y,z) 以及物体的 latent code 表示 C，算法尝试预测该点到物体表面的有向距离长度，最后使用 Marching Cubes 算法即可得到物体的显式表面。DeepSDF 中物体的潜在编码向量没有使用编码器，而是使用网络不断反向优化得到，即采用了自解码结构。测试时候需要使用少量 (x,y,z) 和对应的 SDF 值优化得到测试样例的 latent code。

我们使用 DeepSDF 测试集的 latent code 以及重建网格作为可视化的数据。我们对 latent code 使用 t-sne 展示分布，使用格点化算法重新布局，避免样例间的遮挡问题等。t-sne 图中嵌入了重建的定量结果（Chamfer Distance），并使用 heatmap 展示数据的偏离程度。通过点击 t-sne 中二维点，还可以展示预测结果和真实网格的对比。我们进行了 latent code 的连续插值实验，展示了物体的变形过程。

注意到，虽然我们采用 DeepSDF 方法得到实验数据，我们的系统也可以用于其他使用 latent code 解码的三维重建方法。

分工

陈刚：网络输入数据处理、网络模型训练、网络编码插值及输出、网络输出三维模型渲染。

沈冠霖：网页后端数据处理，包括 t-sne 算法、heatmap 图及 LAP 格点化。

黎思宇：网页前端实现、可视效果渲染及交互。

实现

数据生成

三维形状数据集

我们使用 ShapeNetV2 的五个类别进行实验，包括飞机、桌子、沙发、椅子和台灯。每个类别有 800 个训练样本和 200 个测试样本。我们使用多类别混合训练的设置，以便 latent code 空间的统一。注意到 DeepSDF 论文中仅给出了单类别训练的结果，我们的实验也能探索 DeepSDF 多类别混合训练设置下的表现。我们使用了总计 4000 个训练样本 和 1000 个测试样本。我们基本保持 DeepSDF 源码中给定的超参数设置，训练 2000 个 epoch。可视化的数据采用这 1000 个测试样例的 latent code （256 维）以及重建网格。

编码插值形状生成

潜在编码插值实验中，每个类别我们选取了一个代表形状，在给定目标编码和源编码后，通过线性插值 w * code1 + (1 - w) * code2 得到插值结果，我们采取 0.01 的步长进行插值，路径上共有 99 个中间插值节点。我们使用解码网络进行重建，使用 Marching Cubes 提取网格。

网格渲染

选取合适的角度旋转物体后，我们使用 mayavi.mlab 渲染出所有的重建结果的图像。

可视化

降维

我们使用 t-sne 算法进行降维，使用开源库 sklearn。

格点化

我们使用开源库 LapJv[2] 进行格点化操作。

该方法求解一个凸优化问题：

$$minimize_{\delta_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\delta_{ij}$$

$$s.t. \sum_{i=1}^{N} \delta_{ij} = 1, \forall j \in \{1,\dots,N\}$$

$$s.t. \sum_{i=1}^{N} \delta_{ij} = 1, \forall i \in \{1,\dots,N\}$$

$$\delta_{ij} \in \{0,1\}, \forall i,j$$

其中，w_{ij}代表第i个数据点的$tsne$坐标x_i和第j个格点的坐标y_j的距离，$w_{ij} = ||x_i - y_j||_2$

$\delta_{ij} = 1$代表第i个数据点x_i被分配到了第j个格点y_j

我们使用 40*40 的格点进行格点化，由于只有 1000 个数据点，因此我们在平面上随机生成了 600 个数据点让总数据等于 40*40。

热力图

我们生成了两种热力图，一种是 t-sne 坐标的热力图，一种是格点坐标的热力图。

计算 t-sne 坐标的热力图时，我们用高斯核函数计算格点和每个数据点 t-sne 坐标的距离，然后取平均，作为这个格点的密度值。我们取分辨率为 200*200。

计算格点坐标的热力图时，我们先用高斯核函数计算每个数据点和其他数据点 t-sne 坐标的距离，取平均作为该数据点的密度值，然后对于每个格点，我们让其密度值等于其中数据点的密度值即可。

无论是哪种热力图，我们都是对每一类分别计算的。

插值路径动画

我们使用权重坐标 $x_w = (1-w)x_0 + wx_1, y_w = (1-w)y_0 + wy_1$表示插值后的坐标点，我们使用 d3 控制 svg 中的 path 元素实现了路径动画效果，而插值图片则直接使用帧序列动画来完成。

前端渲染

在后端进行数据处理后，会将计算得到的降维后二维坐标、格点化坐标、热力图的值、插值路

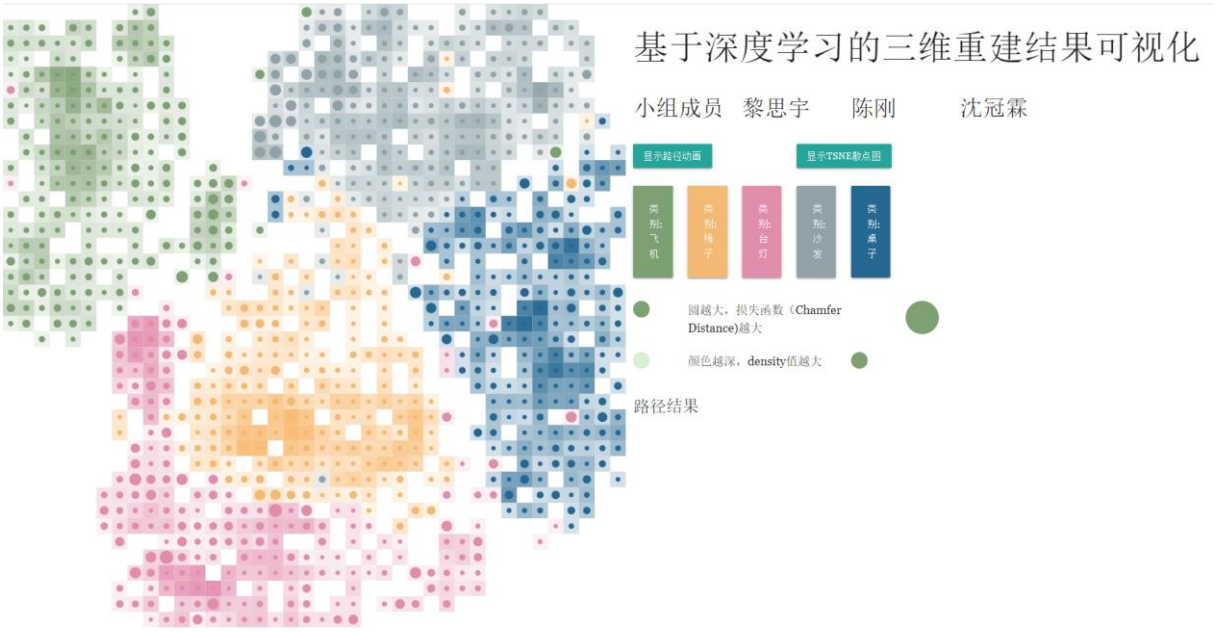
径动画参数制作成 JSON 格式中间数据、发送给前端。

我们使用 html+css+js 来进行前端渲染，在读取数据后，我们使用 d3.js[3] 进行前端数据可视化，我们使用了 Materilialize[4] 的 CSS 样式渲染 UI。

实验结果及结论

可视化界面

我们的可视化界面如下图所示



可以看到，我们的可视化界面由两部分组成，左边的可视化结果以及右边的菜单界面

可视化界面中：

1. 不同颜色的点代表不同类别的形状样本。
2. 点越大，代表对应样本学习的损失函数越大（即是重建情况越糟糕）。
3. 框中颜色越深，代表对应热力图中该点对应的值越深（即颜色越浅代表样本越是离群点）。

交互：点击样本点

使用鼠标点击样本对应的点（或者格点图中对应的框），交互结果如下图所示

可以看到提示框从点击位置处产生，在提示框中，左边是网络重建结果，右边是形状的原始数据样本。

交互：路径动画

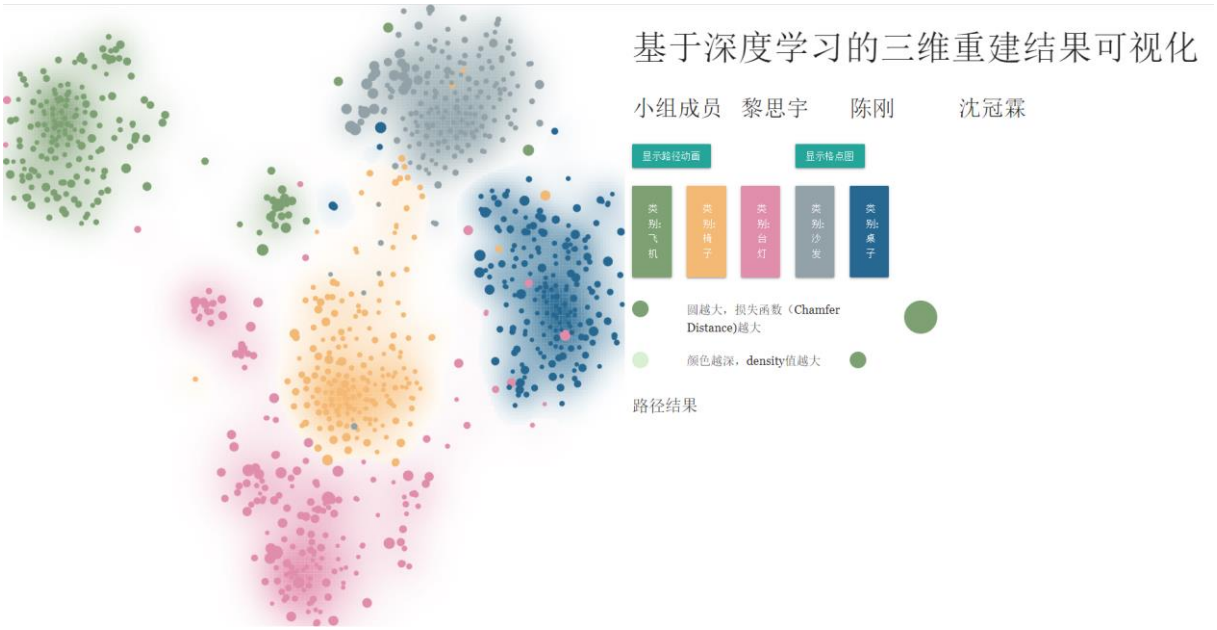
点击”显示路径动画“按钮，网页中就会开始循环放映一条预先计算好的路径动画。

可以看到，路径动画由两部分构成：

1. 可视化界面中的红线表示路径在特征值空间的移动情况
2. 右下角的帧动画表示路径最前面的点对应特征值的重建形状

交互：切换格点图/散点图

我们提供了进行原始 t-sne 散点图和格点化后的格点图之间的切换，点击”显示 t-sne 散点图/格点图 “ 后，左侧可视化结果会重新绘制

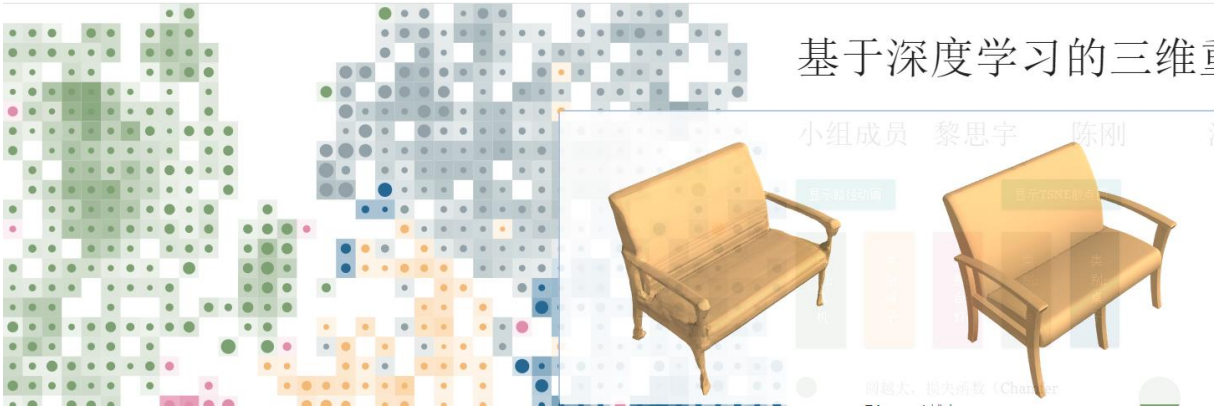


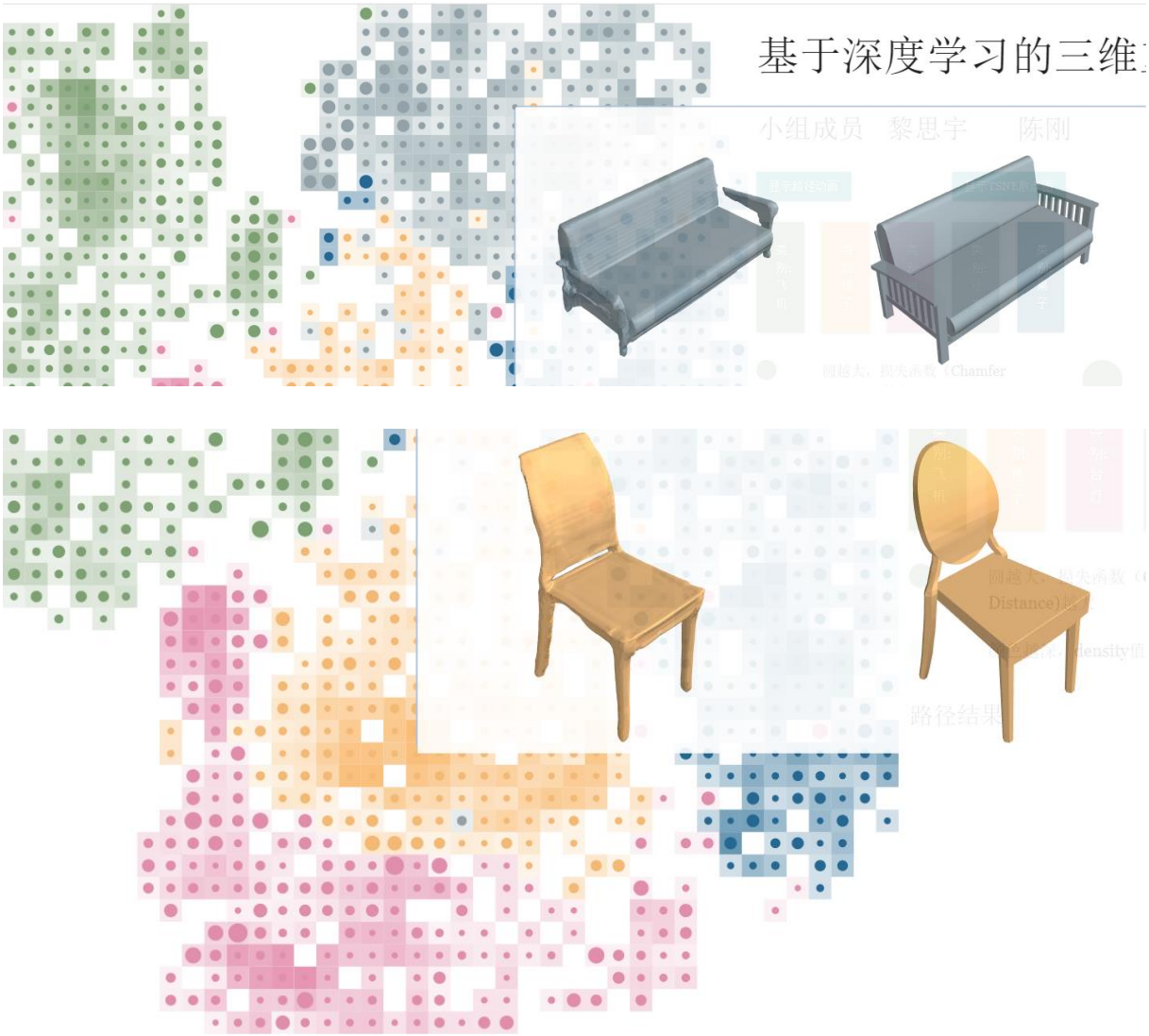
散点图如上图所示，他的信息表示和格点图完全一样，并且他也具有点击样本点/路径动画/切换格点图的交互功能。

结果分析

离群点

我们可以根据可视化结果便利地找到数据集中的离群点。





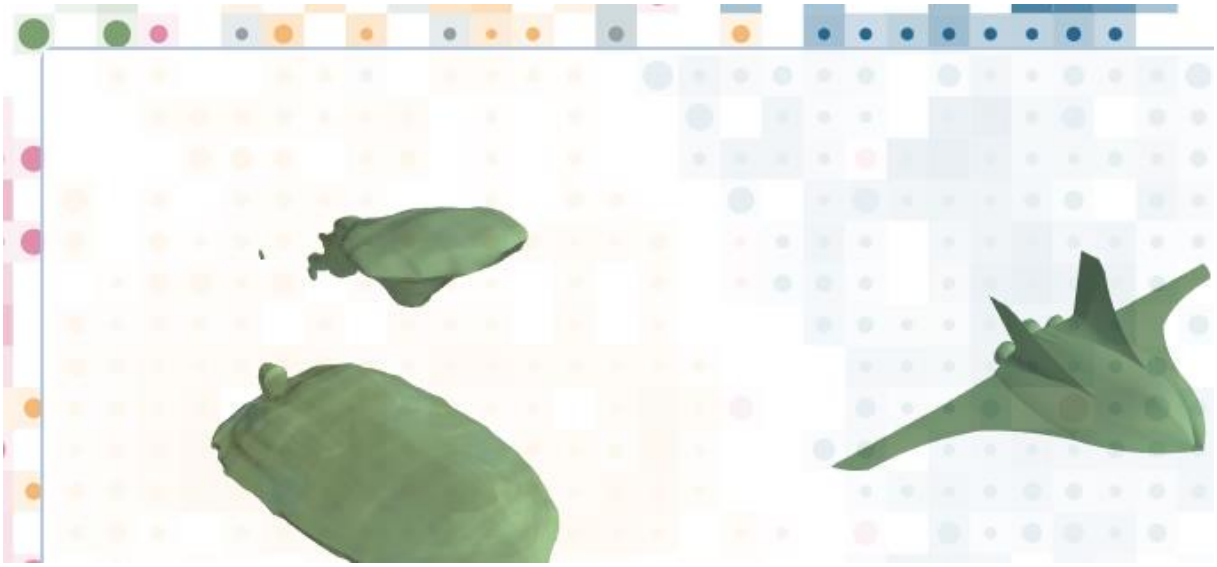
以上图为例子，椅子 1 在特征空间中更靠近沙发而不是椅子。通过观察我们可以发现，椅子 1 确实长得像沙发。我们认为这是因为语言具有多义性，而制作数据集的人可能不是同一拨人，其对”沙发“、”椅子“等词语的认知有一定偏离

另外，我们也发现离群点的产生原因还有：

1. 大部分形状是单一的，但有些形状不是单一的，在其中出现了其他物体
2. 同类形状差异过大：例如台灯从长条形到圆饼形都有分布

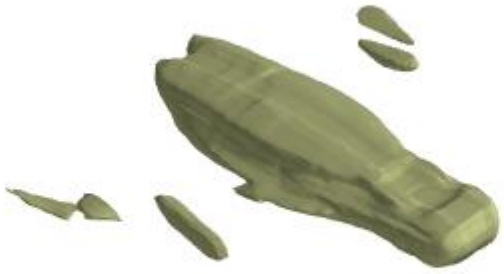
重建质量

通过观察可视化结果中的点的大小，我们可以一目了然地了解重建结果地好坏。



上图为点击一个小圆后的结果，下图为点击大圆后的结果，不出所料，下图重建质量显著低于上图。

插值结果



通过以上插值结果，我们能直观感受到 DeepSDF 编码空间的意义，它确实对应上了形状的某种特种。

结论

综上所述，我们得出以下结论：

1. 可视化技术确实帮助我们更好地认识数据分布.

2. 可视化技术更好地帮我们认识了编码空间的意义.
3. 可视化技术帮助我们更直观看出重建结果的全貌，这是平均损失函数远远不能做到的。

参考文献

[1] Park, Jeong Joon, et al. "Deepsdf: Learning continuous signed distance functions for shape representation." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

[2] Vadim Markovtsev. PyLapjv. https://github.com/src-d/lapjv

[3] Mike Bostock. D3.js. https://d3js.org/

[4] https://materializecss.com/auto-init.html