合肥工業大學

计算机与信息学院

方向领域综合设计报告

设　计　题　目	基于关系图的住宅建筑空间布局系统设计与实现
学生姓名与学号	余梓俊 2018211991
专 业 与 班 级	计算机科学与技术 18-3 班
指　导　教　师	吴文明
完　成　日　期	2022.1.6

课程目标(Course Objectives, CO)	支撑的毕业要求 (Graduation Requirement，GR)及二级指标点	
CO1 运用计算机科学与技术及其应用领域工程的思想与方法，从需求分析开始，独立完成领域方向系统的设计与开发，兼顾领域方向系统开发的社会评价。	GR6 工程与社会：能够基于计算机工程相关背景知识进行合理分析，评价计算机软硬件开发、系统设计等计算机科学与技术工程实践过程和复杂计算机工程问题解决方案对社会、健康、安全、法律以及文化的影响，并理解应承担的责任。	GR6.2 了解社会发展形势，能分析和评价计算机科学与技术工程实践对社会、健康、安全、法律、文化的影响，以及这些制约因素对项目实施的影响，并理解应承担的责任。
CO2 理解领域方向系统运行应遵守的社会规范，能够在领域方向系统开发实践中自觉遵守工程职业道德和规范，履行责任。	GR8 职业规范：具有人文社会科学素养、社会责任感，能够在计算机科学与技术工程实践中理解并遵守工程职业道德和规范，履行责任。	GR8.2 理解计算机科学与技术工程实践活动有可能涉及公众的安全、健康和福祉，以及环境保护的社会责任，能够在计算机科学与技术工程实践中自觉履行责任。
CO3 理解领域方向系统开发的人员组织方式，理解团队成员及负责人角色。	GR9 个人和团队：能够在多学科背景下的团队中承担个体、团队成员以及负责人的角色。	GR9.2 能够在团队中独立或合作开展工作，能够组织、协调和指挥团队开展工作。
CO4 理解领域方向系统开发相关的工程配置管理方法，掌握领域方向系统项目管理能力。	GR11 项目管理：理解并掌握计算机工程管理原理与经济决策方法，并能在多学科环境中应用。	GR11.3 能在多学科环境下(包括模拟环境)，在设计开发解决方案的过程中，运用工程管理与经济决策方法。

验收成绩

CO1 运用计算机科学与技术及其应用领域工程的思想与方法，从需求分析开始，独立完成领域方向系统的设计与开发，兼顾领域方向系统开发的社会评价。（15 分）	CO2 理解领域方向系统运行应遵守的社会规范，能够在领域方向系统开发实践中自觉遵守工程职业道德和规范，履行责任。（5 分）	CO3 理解领域方向系统开发的人员组织方式，理解团队成员及负责人角色。（5 分）	CO4 理解领域方向系统开发相关的工程配置管理方法，掌握领域方向系统项目管理能力。（5 分）

报告成绩

CO1 运用计算机科学与技术及其应用领域工程的思想与方法，从需求分析开始，独立完成领域方向系统的设计与开发，兼顾领域方向系统开发的社会评价。（55 分）	CO2 理解领域方向系统运行应遵守的社会规范，能够在领域方向系统开发实践中自觉遵守工程职业道德和规范，履行责任。（5 分）	CO3 理解领域方向系统开发的人员组织方式，理解团队成员及负责人角色。（5 分）	CO4 理解领域方向系统开发相关的工程配置管理方法，掌握领域方向系统项目管理能力。（5 分）

总成绩：

指导教师评语（验收/报告）：

指导教师签名：

日期：2022 年 1 月 21 日

目录

1. 需求与任务功能描述

背景描述：

房屋设计是一个昂贵且费时的迭代过程。一个标准的工作流是：

1、绘制“气泡图”以说明房间的数量及他们的类型和连接关系。
2、指定相应的平面图，收集顾客反馈。
3、恢复到气泡图进行细化。
4、迭代上面的步骤。

在预算和时间有限的情况下，建筑师及其客户经常需要在设计质量上做出妥协。因此，在建筑，结构和房地产行业中，对自动平面布置图生成技术的需求巨大，并且潜力巨大。

问题描述：

住宅建筑的空间布局是计算机图形学和计算机视觉中的研究热点。传统的室内空间布局主要依赖人工，自动化程度低。本课题将空间布局中对象间的关系表示为如下所示的关系图，基于深度学习方法和优化技术，研究基于关系图的室内空间布局生成算法，设计并实现基于关系图的住宅建筑空间布局系统。

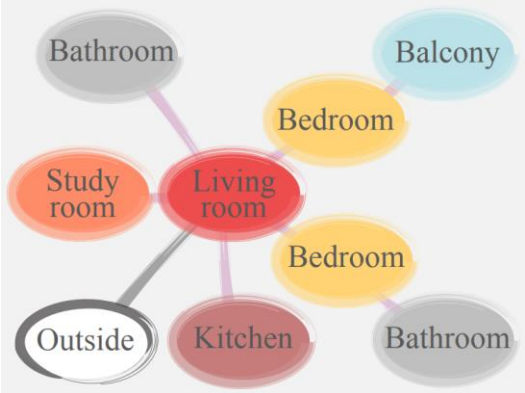

图 1 气泡图

任务：

（1）通过文献调研，理解住宅建筑空间布局问题；
（2）通过文献调研，了解基于关系图的住宅建筑空间的布局方法；
（3） 建立深度学习模型，设计并实现基于关系图的住宅建筑空间布局系统。

2. 设计

2.1 总体设计

站在用户的角度而言，我们的系统需要两个部分：

- 允许用户对房间进行选择，对房间之间的相邻关系做出修改；
- 根据当前的关系图自动生成多个户型图，供用户选择。

2.1.1 前端总体设计

由于需要交互式地修改关系图，对于前端部分，我们选择使用网页来制作我们的用户界面，因为市面上存在着许多优秀的可视化 JavaScript 库。在本次系统中，我们使用 D3.js 作为主要技术手段，来向用户交互式地展示房间关系图。

D3.js [1] 是一个可以基于数据来操作文档的 JavaScript 库。可以帮助你使用 HTML, CSS, SVG 以及 Canvas 来展示数据。D3 遵循现有的 Web 标准，可以不需要其他任何框架独立运行在现代浏览器中，它结合强大的可视化组件来驱动 DOM 操作。

2.1.2 后端总体设计

由于前端采用了网页进行展示，接下来势必要选择 Web 技术作为整体系统框架。

核心算法部分，我们采用 Nauata 等人[2] 提出的 House-GAN++神经网络模型，该模型是对 Nauata 等人[3] 提出的 House-GAN 模型的改进，二者的骨干神经网络均为 Zhang 等人[4] 提出的 Conv-MPN 模型。

Web 服务器部分，由于我们并非实现一个传统的 Web 应用程序，而仅仅是需要一个 Web 服务器来相应前端传来的房屋关系图，进行输出处理后传入神经网络并将网络生成的户型图返回给前端，我们采用了轻量的 Flask 作为 Web 后端框架。Flask [5] 也被称为“microframework”，因为它使用简单的核心，没有默认的数据库、窗体验证工具等等，用 extension 增加其他功能。

2.2 详细设计

2.2.1 前端主要数据结构

房间关系图数据结构：

```
nodes: [
  { id: 0, group: 0 },
  { id: 1, group: 2 },
  { id: 2, group: 2 },
  { id: 3, group: 3 },
  { id: 4, group: 14 },
],
links: [
  { source: 0, target: 1, value: 1 },
  { source: 0, target: 2, value: 1 },
  { source: 0, target: 3, value: 1 },
  { source: 0, target: 4, value: 1 },
  { source: 1, target: 3, value: 1 },
],
```

图 2 前端数据结构样例

该数据与下图对应：

图 3 气泡图样例

其中，nodes 数组存储房间和正门节点，links 数组存储边，source 和 target 字段的值分别为边连接的两个节点的 id。value 字段使用 d3. js 绘制 ForceGraph 时边的权重，对我们而言并不重要，均设为 1。

虽然这是一个无向图，但为了图编辑代码的实现更加方便，以及满足神经网络对输入边的特定要求（见下文），我们约定 link 的 source id 总是小于 target id。

2.2.2 前端主要函数

2.2.2.1 generate ()

前面给出的关系图样例数据，在送进神经网络生成之前，需要处理成如下的样子：

```
new_nodes = [0, 2, 2, 3, 14, 16, 16, 16, 16]
new_links = [
 [0, 1, 1]
 [0, 1, 2]
 [0, 1, 3]
 [0, 1, 4]
 [0, 1, 5]
 [0, 1, 6]
 [0, 1, 7]
 [0, -1, 8]
 [1, -1, 2]
 [1, 1, 3]
 [1, -1, 4]
 [1, 1, 5]
 [1, -1, 6]
 [1, -1, 7]
 [1, 1, 8]
 [2, -1, 3]
 [2, -1, 4]
 ...
 ...
]
```

其中 new_nodes 的前五个节点即为原 nodes 数组的前五个房间和正门节点，后 4 个 group(type)值为 16 的节点，为中间门节点。因为原 links 数组有 5 条边，意味着除去客厅与正门之间的边，应该还有 4 个中间的房门。

new_links 数组中的元素为一个长度为 3 的数组，下标 0 和下标 2 分别对应原 links 数组中的 source 和 target，如果两个节点之间（包括中间门与房间之间）有边，则下标 1 处的值为 1，否则为-1。此处也对应了前文所说的约定，即 source 值小于 target 值。

注意，原 nodes 数组中的元素包含了房间的 id 值，而原 links 数组中的边，source 和 target 均指代房间 id。但在 new_nodes 和 new_links 中，数组的下标

取代了房间 id。这就要求房间的 id 是从 0 开始连续递增的，不允许间断。后文提到的 reIndex 函数保证了这一特征。

综上给出 generate 函数的流程图：

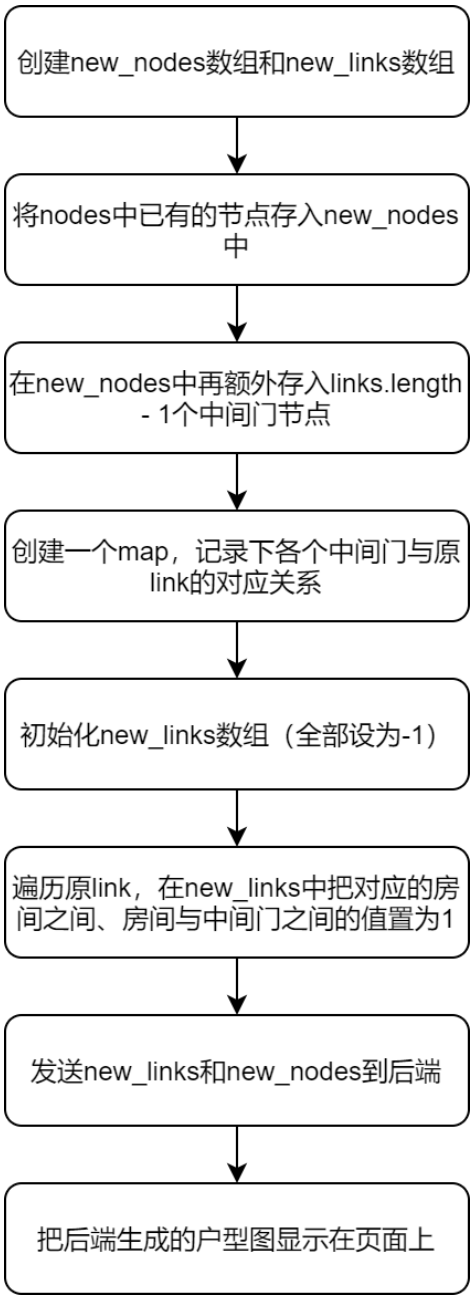

图 4 前端 generate 函数流程图

2.2.2.2 onClick (id)

关系图节点的点击事件的回调函数。

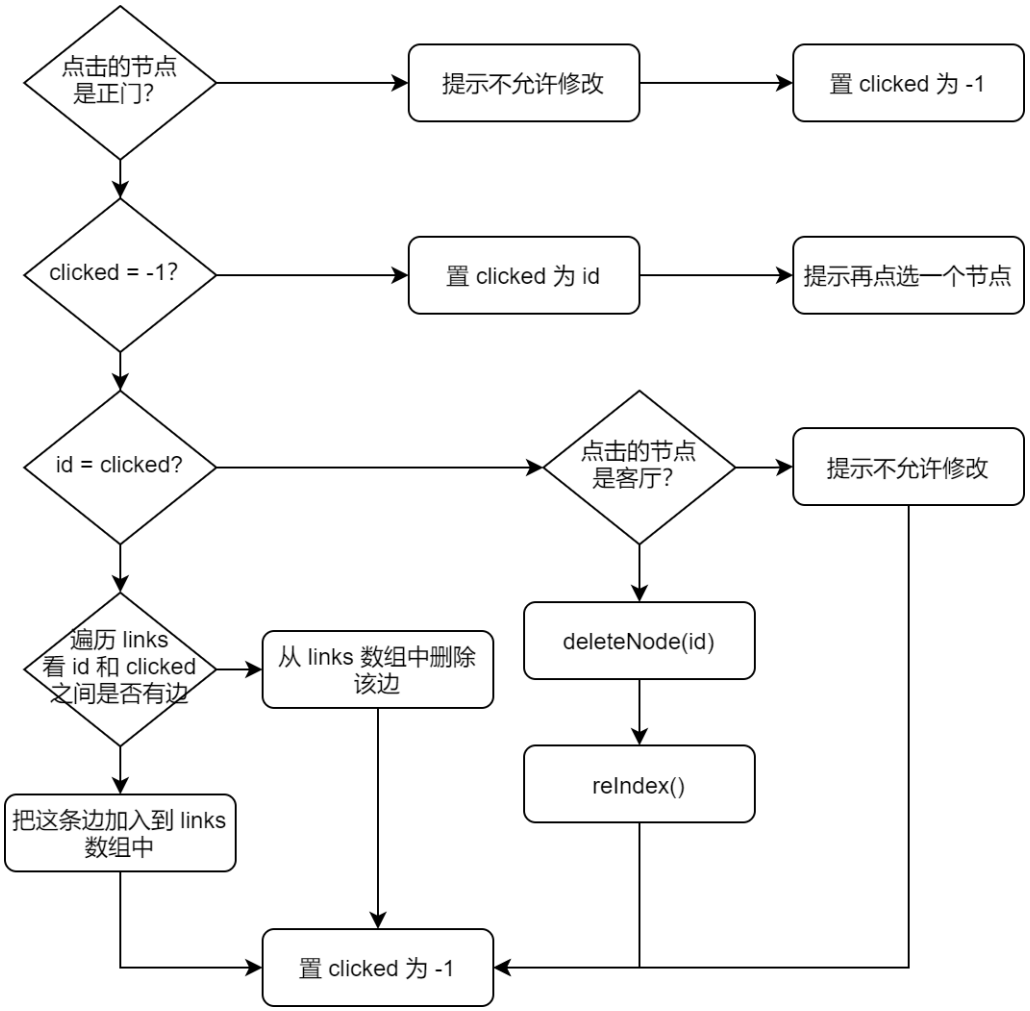

图 5 前端 onClick 函数流程图

2.2.2.3 reIndex ()

1. 令 i=0
2. 遍历 nodes，对于每一个 node
    a. 如果 node.id != i
        i. 遍历 links，对于每一个 link
            1. 如果 link.source = node.id，置 link.source 为 i
            2. 如果 link.target = node.id，置 link.source 为 i
        ii. 置 node.id 为 i
    b. i++

2.2.3 Conv-MPN

Conv-MPN 是图神经网络（GNN）的一种变体，它学习通过交换消息来推断节点的关系。

Conv-MPN 是专门为节点具有显式空间嵌入的情况而设计的，它与标准消息传递神经网络（MPN）有两个主要区别：

1）像 CNN 一样，节点的特征表示为 3D 体一维向量

2）卷积编码消息而不是完全连接的层或矩阵乘法。这种设计允许 Conv-MPN 利用与节点关联的空间信息。

Conv-MPN 背后的基本思想是简单而强大的。标准图神经网络（GNN）将几何信息编码为一维矢量，而不是空间上的 3D 特征。具有 1D 特征向量的 MLP 无法进行有效的几何分析，而与 3D 特征量进行卷积可以实现自然的空间推理。该论文的想法是采用标准的 MPN 体系结构，然后 1）用潜在的 3D 体积替换潜在矢量作为特征表示；2）使用具有卷积的全连接层（或矩阵乘法），用于消息编码。

在标准的 MPN 中，特征向量的更新方法是利用多层感知器（MLP）对消息进行编码并与当前特征混合：

$$f_v \leftarrow \mathrm{MLP}\left(f_v; \sum_{w \in \mathbf{N}(v)} \mathrm{MLP}(f_v; f_w)\right)$$

Conv-MPN 则是改用 CNN 代替 MLP 来形成特征更新规则。不过尽管 Conv-MPN 可以简单地用 CNN 代替 MLP 来形成功能更新规则，但是这将需要两个 CNN 模块，因此需要更多的 GPU 内存。这其中很关键的一点是，节点的特征是分布在整个 volume 上的，因此简单的池化可以将所有信息保留在消息中而不会发生冲突。更准确地说，我们无需为每对节点编码消息，而是将特征跨所有相邻节点合并以编码消息，然后由 CNN 更新特征向量：

$$f_v \leftarrow \mathrm{CNN}\left[f_v; \underset{w \in \mathbf{N}(v)}{\mathrm{Pool}} f_w\right]$$

2.2.3 House-GAN

该论文提出了一种新的图约束生成对抗网络，其生成器和判别器建立在关系架构上。主要思想是将约束编码进关系网络的图结构中。作者对 117000 多张真实平面图进行了定性和定量的评估。结果显示，所提出的方法优于现有方法和基准。

相对于现实的设计过程，作者简化了问题设置：

1、一个结点属性不包含房间大小。

2、房间形状是长方形。

3、边缘属性（即房间邻接）不能反映门的存在。

这是解决问题的第一步，也是将来工作可以扩展的点。作者再次年的 House-GAN++ 模型中便对 2、3 两点进行了改进。

2.2.3.1 Metrics

作者按照房间数将样本分为五组。为了测试生成器的能力，做了 5 折交叉验证：当在一个组中生成布局时，作者利用除这一组中的样本训练一个模型是的一个方法不会简单地被记忆。

在测试时，随机选择房屋布局并生成 X 个样本。X = 10 用于测量真实性和多样性，X = 1 用于测量兼容性，这种评估方法的运算成本非常大。

1、真实性：四个等级，更好、更差、同样好、同样差。

2、多样性：通过 FID 分数和栅格化的布局图像进行测量的。

通过以下方法对布局进行栅格化：

1）将背景设置为白色；

2）按区域的降序对房间进行排序；

3）根据房间类型（例如，卧室为橙色）为每个房间涂上颜色，如图 3 所示。

3、气泡图的兼容性：输入气泡图和从输出布局构造的气泡图之间的图形编辑距离。

2.2.3.2 Models

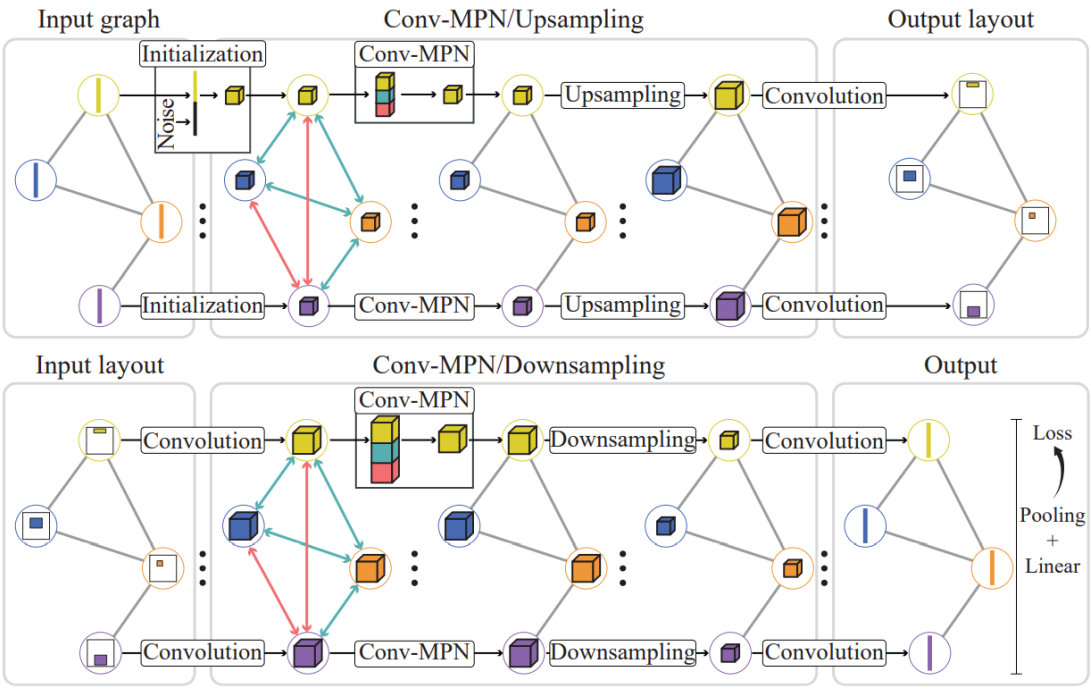

Fig. 4. Relational house layout generator (top) and discriminator (bottom). Conv-MPN is our backbone architecture [26]. The input graph constraint is encoded into the graph structure of their relational networks.

图 6 House-GAN 模型示意图

生成器：

给定一个气泡图，形成 Conv-MPN，其关系图结构与气泡图相同。为每个房间生成一个节点，并使用从正态分布中采样的 128 维噪声向量进行初始化，并与 10 维房间类型向量连接，r 是房间索引。结果是 138 维的向量 gr。

Conv-MPN 将特征存储成一个 3D 张量到输出设计空间中。应用了一个共享的线性层把 gr 扩展为了一个 8*8*16 的特征量：$\mathbf{g}_r^{l=1}$。1=1 表示该特征适用于第一个 Conv-MPN 模块，该模块将被上采样两次，以在稍后成为一个（32×32×16）的特征量 $\mathbf{g}_r^{l=3}$。

Conv-MPN 模块通过卷积消息传递来更新房间的特征量的图：

1）在图中连接的各个房间之间级联一个汇总池特征；

2）在未连接的房间之间级联一个汇总池特征；

3）应用 CNN：

$$\mathbf{g}_r^l \leftarrow \mathrm{CNN}\left[\mathbf{g}_r^l \quad ; \quad \underset{s\in\mathbf{N}(r)}{\mathrm{Pool}}\,\mathbf{g}_s^l \quad ; \quad \underset{s\in\overline{\mathbf{N}}(r)}{\mathrm{Pool}}\,\mathbf{g}_s^l\right]$$

不带上划线的 N(r)表示连接的房间，带上划线的表示不连接的房间。使用转置卷积（内核= 4，步幅= 2，填充= 1）将特征上采样 2 倍，同时保持通道数。生成器具有两轮 Conv-MPN 和上采样，使最终特征量的大小为（32×32×16）。

共享的三层 CNN 将特征量转换为大小为（32×32×1）的房间分割掩码。分割掩码的图将在训练过程中传递给判别器。在测试时，房间掩码（tanh 函数的输出范围为[-1，1]）的阈值设置为 0.0，为每个房间拟合最紧密的轴对齐矩形，以生成房屋布局。

判别器：

判别器以相反的顺序执行上面的操作，输入是来自生成器（在矩形拟合之前）的结果或实际的平面图（前景为 1.0，背景为-1.0）的房间分割掩码。

分割掩码的大小为 32×32×1。为了关联房间类型信息，采用 10 维房间类型向量，应用线性层以扩展到 8192 维，然后重塑为（32×32×8 ）张量，该张量连接到分割掩码。 共享的三层 CNN 会将特征转换为大小（32×32×16），然后进行两轮 Conv-MPN 和下采样。下采样时每次通过卷积层（内核= 3，步幅= 2，填充= 1）减少 2 倍。最后，使用三层 CNN 将房间特征转换为 128 维向量。最后对所有房间向量求和池化，并添加单个线性层以输出标量 d，分类出生成的样本中的真实样本。

作者使用 WGAN-GP 损失，将梯度罚分设置为 10。作者按照 Gulrajani 等人的建议计算梯度罚分：在真实样本和生成的样本之间，以像素方式线性均匀地内插空间分割掩码，同时固定关系图结构。共享的三层 CNN 将特征量转换为大小为（32×32×1）的房间分割掩码。分割掩码的图将在训练过程中传递给判别器。在测试时，房间掩码（tanh 函数的输出范围为[-1，1]）的阈值设置为 0.0，为每个房间拟合最紧密的轴对齐矩形，以生成房屋布局。

Table 2. House-GAN architectural specification. "s" and "p" denote stride and padding. "x", "z" and "t" denote the room mask, noise vector, and room type vector. "conv_mpn" layers have the same architecture in all occurrences. Convolution kernels and layer dimensions are specified as $(N_{in} \times N_{out} \times W \times H)$ and $(W \times H \times C)$.

Architecture	Layer	Specification	Output Size
House layout generator	$concat(z, t)$	N/A	1×138
	$linear\_reshape_1$	138×1024	$8 \times 8 \times 16$
	$conv\_mpn_1$	$\begin{bmatrix} 16 \times 16 \times 3 \times 3, (s = 1, p = 1) \\ 16 \times 16 \times 3 \times 3, (s = 1, p = 1) \\ 16 \times 16 \times 3 \times 3, (s = 1, p = 1) \end{bmatrix}$	$8 \times 8 \times 16$
	$upsample_1$	$16 \times 16 \times 4 \times 4, (s = 2, p = 1)$	$16 \times 16 \times 16$
	$conv\_mpn_2$	-	$16 \times 16 \times 16$
	$upsample_2$	$16 \times 16 \times 4 \times 4, (s = 2, p = 1)$	$32 \times 32 \times 16$
	$conv\_leaky\_relu_1$	$16 \times 256 \times 3 \times 3, (s = 1, p = 1)$	$32 \times 32 \times 256$
	$conv\_leaky\_relu_2$	$256 \times 128 \times 3 \times 3, (s = 1, p = 1)$	$32 \times 32 \times 128$
	$conv\_tanh_1$	$128 \times 1 \times 3 \times 3, (s = 1, p = 1)$	$32 \times 32 \times 1$
House layout discriminator	$linear\_reshape_1(t)$	10×8192	$32 \times 32 \times 8$
	$concat(t, x)$	N/A	$32 \times 32 \times 9$
	$conv\_leaky\_relu_1$	$9 \times 16 \times 3 \times 3, (s = 1, p = 1)$	$32 \times 32 \times 16$
	$conv\_leaky\_relu_2$	$16 \times 16 \times 3 \times 3, (s = 1, p = 1)$	$32 \times 32 \times 16$
	$conv\_leaky\_relu_3$	$16 \times 16 \times 3 \times 3, (s = 1, p = 1)$	$32 \times 32 \times 16$
	$conv\_mpn_1$	-	$32 \times 32 \times 16$
	$downsample_1$	$16 \times 16 \times 3 \times 3, (s = 2, p = 1)$	$16 \times 16 \times 16$
	$conv\_mpn_2$	-	$16 \times 16 \times 16$
	$downsample_2$	$16 \times 16 \times 3 \times 3, (s = 2, p = 1)$	$8 \times 8 \times 16$
	$conv\_leaky\_relu_1$	$16 \times 256 \times 3 \times 3, (s = 2, p = 1)$	$4 \times 4 \times 256$
	$conv\_leaky\_relu_2$	$256 \times 128 \times 3 \times 3, (s = 2, p = 1)$	$2 \times 2 \times 128$
	$conv\_leaky\_relu_3$	$128 \times 128 \times 3 \times 3, (s = 2, p = 1)$	$1 \times 1 \times 128$
	$pool\_reshape\_linear_1$	128×1	1

图 7 House-GAN 模型详细定义

2.2.4 House-GAN++

Nauata 等人在其前作 House-GAN 的基础上，提出了更优秀的网络，网络的输入同样为气泡图，通过不断地迭代优化，生成一个平面户型图。作者在 House-GAN 的基础上作了如下几个重要的扩展：

1）处理非矩形房间形状；

2）生成中间门和正门；

3）使用功能图而不是邻接图。

该论文的技术贡献有三个方面:架构、训练算法和测试时元优化算法。首先，该网络是一个图约束关系型 GAN 和一个 conditional GAN 的集成，上一步生成的图成为下一个输入约束，实现迭代布局优化。其次，作者发现，一个计算上负担得起的非迭代训练过程，被称为 component-wise GT-conditioning，在学习迭代生成器时是有效的。最后，作者的框架发现了通过元优化进一步优化选择指标的可能性，例如通过控制何时使用哪些约束来进行贝叶斯优化。

和 House-GAN 相同，该网络的主干仍然是卷积消息传递网络 Conv-MPN，其关系图结构由气泡图定义。模型中有三个关键的区别:

1）除了节点之外的边承载了门生成的特性；

2）每个节点/边缘采用 2D 分割掩码作为具有相关新损失的附加输入约束；

3）Conv-MPN 特征池被重构以允许节点和边缘之间的特征交换。

以下分别对这三点进行阐述。

1）边特征

House-GAN 使用的是 10 维的独热向量，其编码房间类型并初始化节点特征向量。在 House-GAN++中，向量被扩展到 12 维，包括 10 种房间类型和 2 种门类型。使用通用类型向量，除了下段所述的卷积消息传递中的池机制不同外，从边生成门与从节点生成房间是一样的。

2）Mask Condition

House-GAN 用一个噪声向量和一个房间类型初始化每个节点，并将其转换为 8×8×16 的特征 volume。House-GAN++的关系生成器为每个节点/边获取一个额

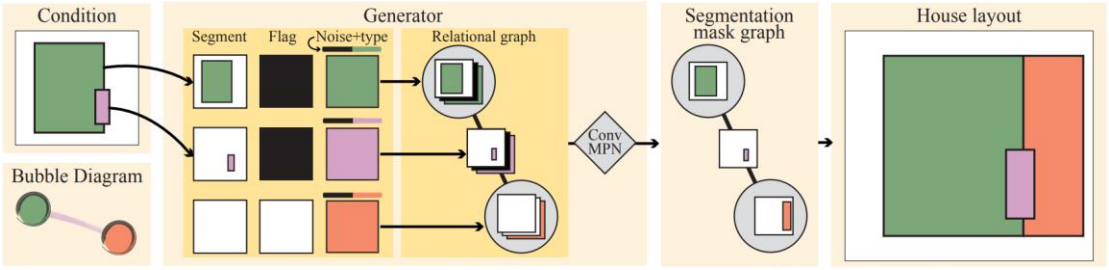

Figure 3. Our architecture is built on top of a relational GAN from the state-of-the-art system [24]. An additional 2D segmentation mask for each room/door can be specified as an input condition, enabling iterative design refinement.

图 7 House-GAN++ 模型示意图

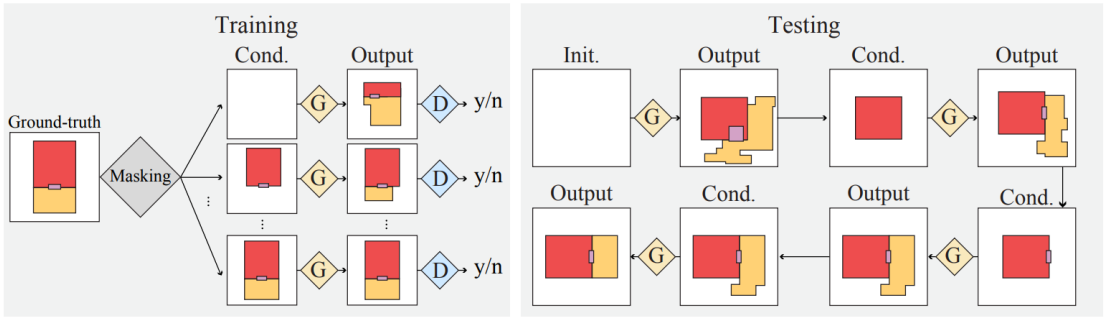

Figure 4. During training, we specify a GT segmentation mask for each room/door with a 50% chance. The generator learns the task of inpainting missing components when many GT masks are given, or the task of generating a complete design when few masks are given. During testing, previously generated layouts are passed to the generator as potential input constraints, enabling iterative design refinement.

图 8 House-GAN++ 训练过程

外的 64×64×2 的 condition image。第一个通道提供分割掩码，希望生成器学会保持不变。当指定分割掩码时，第二个通道对每个像素都变成 1，否则为 0。使用一个 3 层 CNN 将 condition image 转换为 8×8×16，并将其连接到原始特征。当指定分割掩码时，我们在 condition image 和生成的掩码之间应用 L1 Loss。

3）Conv-MPN 池化

Conv-MPN 的消息传递通过关系图的连通性来定义：

$$\mathbf{g}_r \leftarrow \mathrm{CNN}\left[\mathbf{g}_r \quad ; \quad \underset{s\in\mathcal{N}(r)}{\mathrm{Pool}}\ \mathbf{g}_s \quad ; \quad \underset{s\in\overline{\mathcal{N}}(r)}{\mathrm{Pool}}\ \mathbf{g}_s\right]$$

式的两个 N(r)分别表示 r 的邻接点及其补。N(r)被重新定义，以允许房间和门之间交换特征：首先，一个房间的邻接点包含与它相连的房间和中间门；其次，一个中间门是它的两个房间的邻接点。

模型的其余部分与 House-GAN 相同，但是上采样进行了三次，而不是两次，最终的分割掩码大小为 64×64。

3. 项目结构

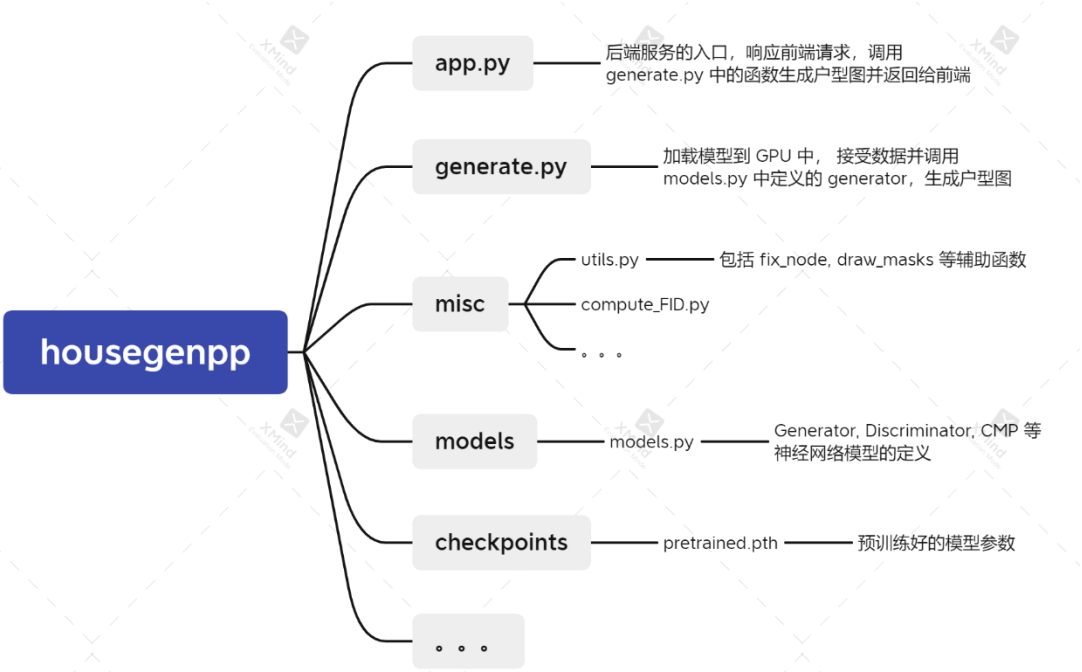

图 9 后端项目结构

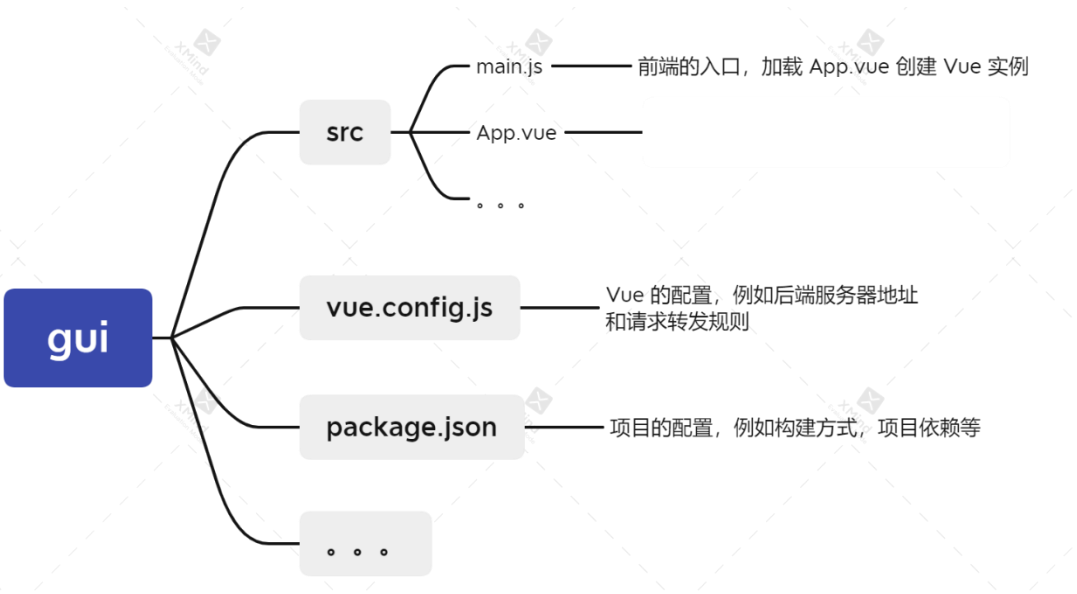

图 10 前端项目结构

4. 用户手册

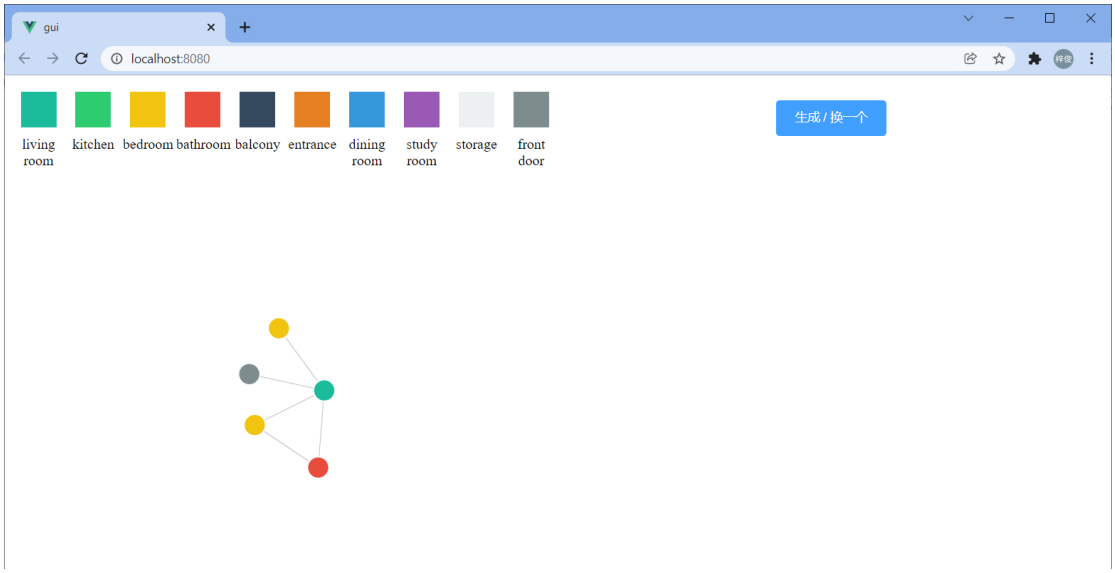

图 11 用户手册示例图 1

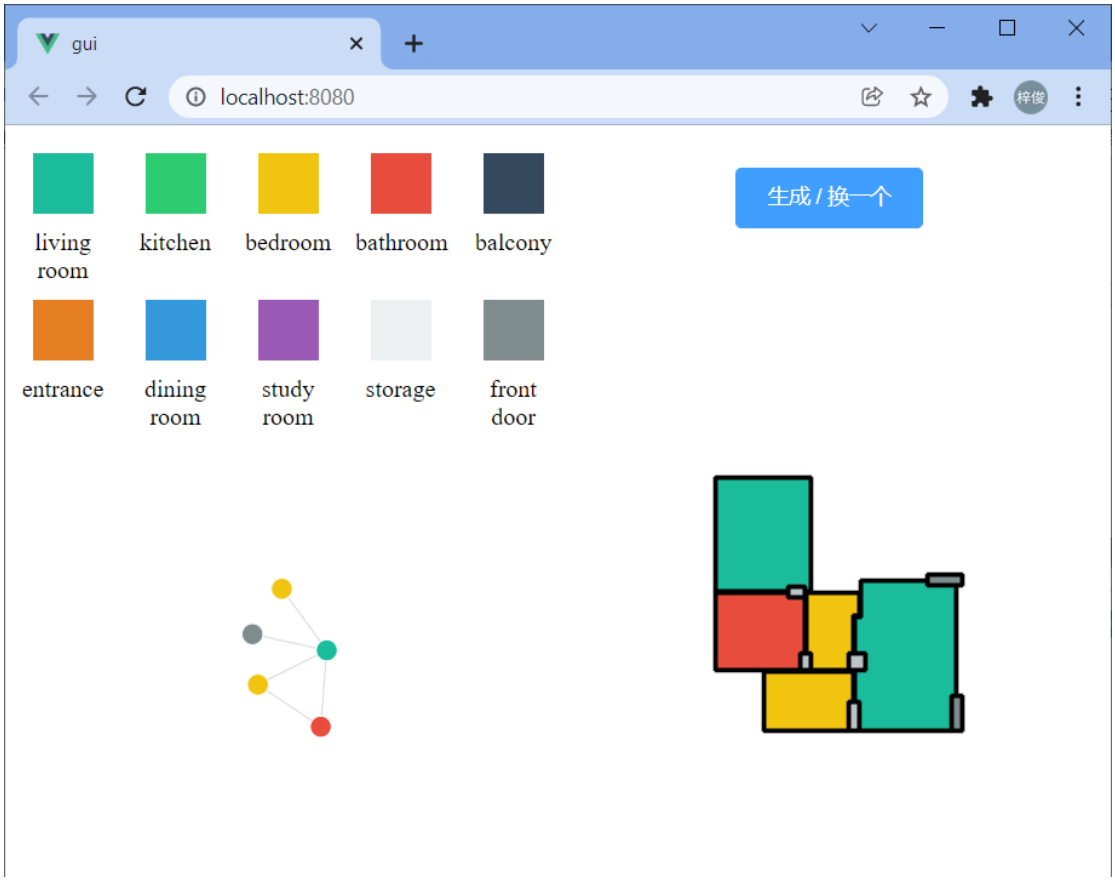

图 12 用户手册示例图 2

1）基本用法

打开系统，可以看到预先默认设好的房间气泡图，点击“生成/换一个按钮”，即可看到生成的户型图。

2）增加房间

系统的左上方展示了所有可以选择的房间类型，点击相应色块，即可添加一个对应类型的房间。

3）增加、删除房间之间的关联关系

点击气泡图中的任一节点，将弹出提示“请再点击一个节点”，并且此时“生成”按钮变灰，不可点击。再点击另外一个节点，如果这两个节点之间没有边，则会给他加上边，否则即为删除该边。操作结束后，“生成”按钮重新变为可用状态。

4）删除房间

点击气泡图中的任一节点，将弹出提示“请再点击一个节点”，此时再点击一次该节点，即可将该房间节点及相关联的边删除。

5）其他事项

客厅与正门之间的边不允许被删除，客厅与正门本身不允许被删除。若进行这些操作，系统将会提示这是不允许的。

如果气泡图的边之间有重叠，可以按住并拖动房间节点。

5. 系统功能测试

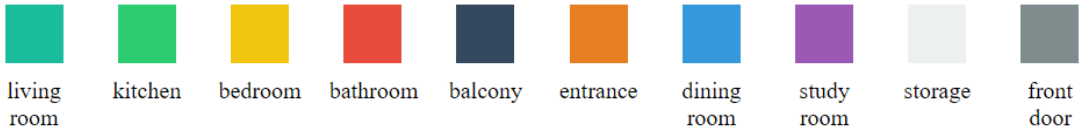

图 13 测试数据说明

测试数据 1：

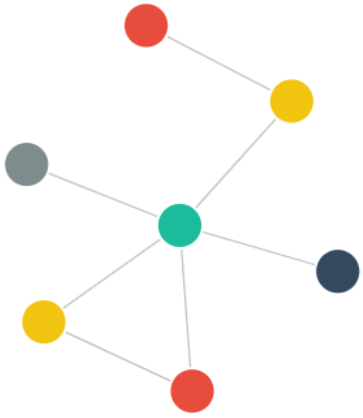

图 14 测试数据 1

运行结果 1：

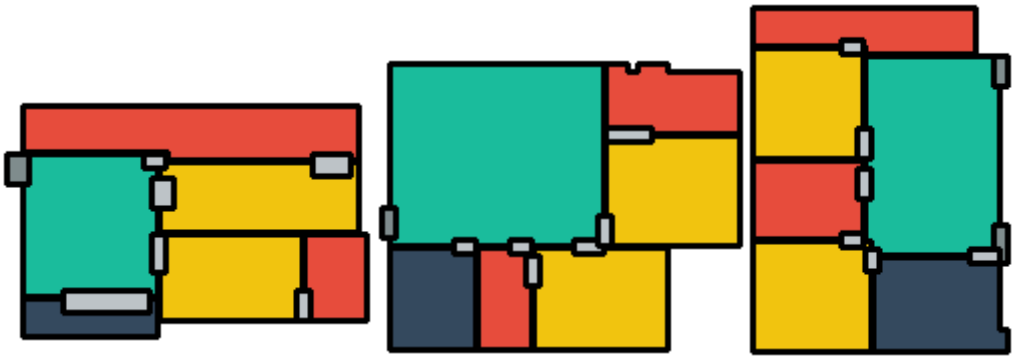

图 15 测试结果 1

测试数据 2：

图 16 测试数据 2

运行结果 2：

图 17 测试结果 2

6．存在的问题

1）气泡图交互的问题

对气泡图的修改并不是直接操作 SVG 中的元素，而是对原始的数据数组进行修改，修改完后调用绘图函数重绘。这就导致每次修改气泡图后，气泡会闪动一下，并且用户所作的拖拽操作被清空，节点回到初始的位置。虽然功能上不会出现问题，但在交互上不够友好。

2）应当保留生成的历史记录

用户点击生成按钮后，旧户型图被新图替换，再看到新图后，用户可能会更倾向于旧图，但 GAN 几乎不可能再次生成相同的户型图。因此应当保留每一次生成的图片，并允许用户前后翻动查看。

3）生成的户型图作用有限

我们希望该系统能嵌入到设计师的设计流程中，即设计师使用该系统生成初步的户型图后，对其进行修改并展示给户主。但当前我们的系统仅仅返回了一张 PNG 格式的图片，设计师无法对此进行直接修改。

一个可能改进的方案是，在神经网络生成房间掩码后，不要使用 Matplotlib 画出 PNG 图片返回给前端，而是利用其他图形库画出 SVG 矢量图返回给前端，设计师则可以在系统中下载矢量图或者将矢量图进一步转换为 CAD 等设计软件所需的格式。

附录——源程序

App.vue

```
<template>
  <div id="app">
    <el-row style="width: 100%">
      <el-col :span="12" style="height: 100%">
        <div id="draw" style="height: 100%">
          <div style="display: flex; justify-content: space-between; flex-wrap:
wrap;">
            <div
              style="flex-shrink: 0; flex-grow: 0"
              v-for="(value, name) in ROOM_CLASS"
              :key="name"
            >
              <div
                :style="{
                  height: '40px',
                  width: '40px',
                  margin: '10px',
                  'background-color': ID_COLOR[value],
                  cursor: 'pointer',
                }"
                @click="addNewRoom(value - 1)"
              ></div>
              <div style="text-align: center; width: 60px">{{ name }}</div>
            </div>
          </div>
        </div>
      </el-col>
      <el-col :span="12" style="height: 100%">
        <div id="result" style="height: 100%; text-align: center; display: flex;
flex-direction: column; justify-content: space-between; align-items: center;">
          <el-button
            style="margin: 20px"
            type="primary"
            :loading="loading"
            :disabled="clicked !== -1"
            @click="generate()"
            >生成 / 换一个</el-button
          >
          <img style="margin-top: 120px;" :src="imgurl" alt="" />
        </div>
      </el-col>
    </el-row>
  </div>
</template>

<script>
// import HelloWorld from './components/HelloWorld.vue'

import axios from "axios";
import * as d3 from "d3";
```

```
export default {
  name: "App",
  data: function () {
    return {
      loading: false,
      clicked: -1,
      imgurl: "",
      ID_COLOR: {
        1: "#1ABC9C",
        2: "#2ECC71",
        3: "#F1C40F",
        4: "#E74C3C",
        5: "#34495E",
        6: "#E67E22",
        7: "#3498DB",
        8: "#9B59B6",
        10: "#ECF0F1",
        15: "#7F8C8D",
        17: "#BDC3C7"
      },
      ROOM_CLASS: {
        "living room": 1,
        kitchen: 2,
        bedroom: 3,
        bathroom: 4,
        balcony: 5,
        entrance: 6,
        "dining room": 7,
        "study room": 8,
        storage: 10,
        "front door": 15 /*, "unknown": 16, "interior_door": 17*/,
      },
      nodes: [
        { id: 0, group: 0 },
        { id: 1, group: 2 },
        { id: 2, group: 2 },
        { id: 3, group: 3 },
        { id: 4, group: 14 },
      ],
      links: [
        { source: 0, target: 1, value: 1 },
        { source: 0, target: 2, value: 1 },
        { source: 0, target: 3, value: 1 },
        { source: 0, target: 4, value: 1 },
        { source: 1, target: 3, value: 1 },
      ],
    };
  },

  components: {},

  mounted: function () {
    this.ForceGraphWrapper();
  },

  methods: {
    addNewRoom(group) {
      if (group === 14 || group === 0) {
```

```
      this.$message.warning({
        message: "前门和客厅不允许修改",
        duration: 1000,
      });
      return;
    }

    this.nodes.push({
      id: this.nodes[this.nodes.length - 1].id + 1,
      group: group,
    });
    document.getElementsByTagName("svg")[0].remove();
    this.ForceGraphWrapper();
  },

  onClick(id) {
    const node = this.nodes.find((node) => node.id === id);
    if (node.group === 14) {
      this.$message.warning({
        message: "前门和客厅不允许修改",
        duration: 1000,
      });
      this.clicked = -1;
      return;
    }

    if (this.clicked == -1) {
      this.$message({ message: "请再点击另一个节点", duration: 1000 });
      this.clicked = id;
      return;
    }

    if (id === this.clicked) {
      // 两次都是点同一个节点，等于删除节点

      if (node.group === 0) {
        this.$message.warning({
          message: "前门和客厅不允许修改",
          duration: 1000,
        });
      } else {
        this.deleteNode(id);
        this.reIndex();
        document.getElementsByTagName("svg")[0].remove();
        this.ForceGraphWrapper();
      }
    } else {
      // 两次点击不同的节点，等于删除或加上这两个节点间的边

      for (let i = 0; i < this.links.length; ++i) {
        if (
          (this.links[i].source === id &&
            this.links[i].target === this.clicked) ||
          (this.links[i].source === this.clicked &&
            this.links[i].target === id)
        ) {
          this.links.splice(i, 1);
          document.getElementsByTagName("svg")[0].remove();
```

```
        this.ForceGraphWrapper();
        this.clicked = -1;
        return;
      }
    }

    if (id < this.clicked)
      this.links.push({ source: id, target: this.clicked, value: 1 });
    else this.links.push({ source: this.clicked, target: id, value: 1 });
    document.getElementsByTagName("svg")[0].remove();
    this.ForceGraphWrapper();
  }

  this.clicked = -1;
},

deleteNode(id) {
  // const node = this.nodes.find((node) => node.id === id);
  // if (node.group === 14 || node.group === 1) {
  //   this.$message.warning({ message: "房门不允许修改", duration: 1000 });
  //   return;
  // }

  this.nodes.splice(
    this.nodes.findIndex((node) => node.id === id),
    1
  );

  const links = [];
  for (const link of this.links) {
    if (link.source !== id && link.target !== id) links.push(link);
  }
  this.links = links;
},

reIndex() {
  let i = 0;
  for (const node of this.nodes) {
    if (node.id !== i) {
      for (const link of this.links) {
        if (link.source === node.id) link.source = i;
        else if (link.target === node.id) link.target = i;
      }
      node.id = i;
    }
    ++i;
  }
},

generate() {
  this.loading = true;

  const nodes = [];
  const edges = [];
  const frontDoorIndex = this.nodes.findIndex((node) => node.group === 14);
  const numInteriorDoor = this.links.length - 1;

  for (const node of this.nodes) nodes.push(node.group);
```

```
for (let i = 0; i < numInteriorDoor; ++i) nodes.push(16);

for (let i = 0; i < nodes.length; ++i) {
  for (let j = i + 1; j < nodes.length; ++j) {
    edges.push([i, -1, j]);
  }
}

// map 保存 link 对应的 Interior Door 的下标
const map = new Map();
let i = this.nodes.length;
for (const link of this.links) {
  if (link.source === frontDoorIndex || link.target === frontDoorIndex)
    continue;
  map.set(link, i);
  ++i;
}
for (const link of this.links) {
  const edge = edges.find(
    edge => edge[0] === link.source && edge[2] === link.target
  );
  edge[1] = 1;
  if (
    link.source !== frontDoorIndex &&
    link.target !== frontDoorIndex
  ) {
    const interiorDoorIndex = map.get(link);
    const edge1 = edges.find(
      edge => edge[0] === link.source && edge[2] === interiorDoorIndex
    );
    edge1[1] = 1;
    const edge2 = edges.find(
      edge => edge[0] === link.target && edge[2] === interiorDoorIndex
    );
    edge2[1] = 1;
  }
}

console.log(nodes);
console.log(edges);

axios({
  url: "/api/generate",
  method: "post",
  data: {
    nodes,
    edges,
  },
  responseType: "blob",
})
  .then((res) => {
    this.imgurl = window.URL.createObjectURL(
      new window.Blob([res.data], { type: "image/png" })
    );
  })
  .finally(() => {
    this.loading = false;
  });
```

```
    },

    ForceGraphWrapper() {
      return this.ForceGraph(
        "#draw",
        {
          nodes: this.nodes,
          links: this.links,
        },
        {
          nodeId: (d) => d.id,
          nodeGroup: (d) => d.group,
          nodeTitle: (d) => `${d.id}\n${d.group}`,
          linkStrokeWidth: (l) => Math.sqrt(l.value),
          width: 720,
          height: 600,
          nodeRadius: 14,
          distance: 100,
          // invalidation // a promise to stop the simulation when the cell is re-
run
        }
      );
    },

    // Copyright 2021 Observable, Inc.
    // Released under the ISC license.
    // https://observablehq.com/@d3/force-directed-graph
    ForceGraph(
      elem,
      {
        nodes, // an iterable of node objects (typically [{id}, …])
        links, // an iterable of link objects (typically [{source, target}, …])
      },
      {
        distance = 50,
        nodeId = (d) => d.id, // given d in nodes, returns a unique identifier
(string)
        nodeGroup, // given d in nodes, returns an (ordinal) value for color
        nodeGroups, // an array of ordinal values representing the node groups
        nodeTitle, // given d in nodes, a title string
        nodeFill = "currentColor", // node stroke fill (if not using a group color
encoding)
        nodeStroke = "#fff", // node stroke color
        nodeStrokeWidth = 1.5, // node stroke width, in pixels
        nodeStrokeOpacity = 1, // node stroke opacity
        nodeRadius = 5, // node radius, in pixels
        nodeStrength,
        linkSource = ({ source }) => source, // given d in links, returns a node
identifier string
        linkTarget = ({ target }) => target, // given d in links, returns a node
identifier string
        linkStroke = "#999", // link stroke color
        linkStrokeOpacity = 0.6, // link stroke opacity
        linkStrokeWidth = 1.5, // given d in links, returns a stroke width in
pixels
        linkStrokeLinecap = "round", // link stroke linecap
        linkStrength,
```

```
        colors = d3.schemeTableau10, // an array of color strings, for the node
groups
        width = 640, // outer width, in pixels
        height = 400, // outer height, in pixels
        invalidation, // when this promise resolves, stop the simulation
      } = {}
    ) {
      // Compute values.
      const N = d3.map(nodes, nodeId).map(intern);
      const LS = d3.map(links, linkSource).map(intern);
      const LT = d3.map(links, linkTarget).map(intern);
      if (nodeTitle === undefined) nodeTitle = (_, i) => N[i];
      const T = nodeTitle == null ? null : d3.map(nodes, nodeTitle);
      const G = nodeGroup == null ? null : d3.map(nodes, nodeGroup).map(intern);
      const W =
        typeof linkStrokeWidth !== "function"
          ? null
          : d3.map(links, linkStrokeWidth);

      // Replace the input nodes and links with mutable objects for the simulation.
      nodes = d3.map(nodes, (_, i) => ({ id: N[i] }));
      links = d3.map(links, (_, i) => ({ source: LS[i], target: LT[i] }));

      // Compute default domains.
      if (G && nodeGroups === undefined) nodeGroups = d3.sort(G);

      // Construct the scales.
      const color =
        nodeGroup == null ? null : d3.scaleOrdinal(nodeGroups, colors);

      // Construct the forces.
      const forceNode = d3.forceManyBody();
      const forceLink = d3
        .forceLink(links)
        .id(({ index: i }) => N[i])
        .distance(distance);
      if (nodeStrength !== undefined) forceNode.strength(nodeStrength);
      if (linkStrength !== undefined) forceLink.strength(linkStrength);

      const simulation = d3
        .forceSimulation(nodes)
        .force("link", forceLink)
        .force("charge", forceNode)
        .force("center", d3.forceCenter())
        .on("tick", ticked);

      const svg =
        // d3.create("svg")
        d3
          .select(elem)
          .append("svg")
          .attr("width", width)
          .attr("height", height)
          .attr("viewBox", [-width / 2, -height / 2, width, height])
          .attr("style", "max-width: 100%; height: auto; height: intrinsic;");

      const link = svg
        .append("g")
```

```
    .attr("stroke", linkStroke)
    .attr("stroke-opacity", linkStrokeOpacity)
    .attr(
      "stroke-width",
      typeof linkStrokeWidth !== "function" ? linkStrokeWidth : null
    )
    .attr("stroke-linecap", linkStrokeLinecap)
    .selectAll("line")
    .data(links)
    .join("line");

  const node = svg
    .append("g")
    .attr("fill", nodeFill)
    .attr("stroke", nodeStroke)
    .attr("stroke-opacity", nodeStrokeOpacity)
    .attr("stroke-width", nodeStrokeWidth)
    .selectAll("circle")
    .data(nodes)
    .join("circle")
    .attr("r", nodeRadius)
    .call(drag(simulation));

  if (W) link.attr("stroke-width", ({ index: i }) => W[i]);
  // if (G) node.attr("fill", ({index: i}) => color(G[i]));
  if (G) node.attr("fill", ({ index: i }) => this.ID_COLOR[G[i] + 1]);
  if (G) node.on("click", (_, { id: i }) => this.onClick(i));
  if (T) node.append("title").text(({ index: i }) => T[i]);
  if (invalidation != null) invalidation.then(() => simulation.stop());

  function intern(value) {
    return value !== null && typeof value === "object"
      ? value.valueOf()
      : value;
  }

  function ticked() {
    link
      .attr("x1", (d) => d.source.x)
      .attr("y1", (d) => d.source.y)
      .attr("x2", (d) => d.target.x)
      .attr("y2", (d) => d.target.y);

    node.attr("cx", (d) => d.x).attr("cy", (d) => d.y);
  }

  function drag(simulation) {
    function dragstarted(event) {
      if (!event.active) simulation.alphaTarget(0.3).restart();
      event.subject.fx = event.subject.x;
      event.subject.fy = event.subject.y;
    }

    function dragged(event) {
      event.subject.fx = event.x;
      event.subject.fy = event.y;
    }
```

```
        function dragended(event) {
          if (!event.active) simulation.alphaTarget(0);
          event.subject.fx = null;
          event.subject.fy = null;
        }

        return d3
          .drag()
          .on("start", dragstarted)
          .on("drag", dragged)
          .on("end", dragended);
      }

      console.log(nodes);
      console.log(links);
    },
  },
};
</script>

<style>
</style>
```

app.py

```
from flask import Flask, request, send_file

import numpy as np
from generate import generate as gen

app = Flask(__name__)

@app.route("/generate", methods=['POST'])
def generate():
    real_nodes = request.json['nodes']
    nds = np.eye(18)[real_nodes].tolist()
    eds = request.json['edges']
    gen(nds, eds)
    return send_file('./dump/fp_final_generate.png', mimetype='image/png')
```

generate.py

```
import numpy as np
import torch
from models.models import Generator
from misc.utils import _init_input, draw_masks, draw_graph
from torchvision.utils import save_image

model = Generator()
model.load_state_dict(torch.load('./checkpoints/pretrained.pth'), strict=True)
model = model.eval()
```

```
if torch.cuda.is_available():
    model.cuda()

Tensor = torch.cuda.FloatTensor if torch.cuda.is_available() else torch.FloatTensor

# run inference
def _infer(graph, model, prev_state=None):
    # configure input to the network
    z, given_masks_in, given_nds, given_eds = _init_input(graph, prev_state)
    # run inference model
    with torch.no_grad():
        masks = model(z.to('cuda'), given_masks_in.to('cuda'),
given_nds.to('cuda'), given_eds.to('cuda'))
        masks = masks.detach().cpu().numpy()
    return masks

def generate(nds, eds):
    nds = torch.tensor(nds)
    eds = torch.tensor(eds)
    print(nds)
    print(eds)

    real_nodes = np.where(nds.detach().cpu() == 1)[-1]
    graph = [nds, eds]
    true_graph_obj, graph_im = draw_graph([real_nodes, eds.detach().cpu().numpy()])
    graph_im.save('./dump/graph_generate.png')  # save graph

    # add room types incrementally
    _types = sorted(list(set(real_nodes)))
    selected_types = [_types[:k + 1] for k in range(10)]
    _round = 0

    # initialize layout
    state = {'masks': None, 'fixed_nodes': []}
    masks = _infer(graph, model, state)
    im0 = draw_masks(masks.copy(), real_nodes)
    im0 = torch.tensor(np.array(im0).transpose((2, 0, 1))) / 255.0
    # save_image(im0, './dump/fp_init_generate.png', nrow=1, normalize=False) #
visualize init image

    # generate per room type
    for _iter, _types in enumerate(selected_types):
        _fixed_nds = np.concatenate([np.where(real_nodes == _t)[0] for _t in
_types]) \
            if len(_types) > 0 else np.array([])
        state = {'masks': masks, 'fixed_nodes': _fixed_nds}
        masks = _infer(graph, model, state)

    # save final floorplans
    imk = draw_masks(masks.copy(), real_nodes)
    imk = torch.tensor(np.array(imk).transpose((2, 0, 1))) / 255.0
    save_image(imk, './dump/fp_final_generate.png', nrow=1, normalize=False)
```

剩余代码见压缩包。