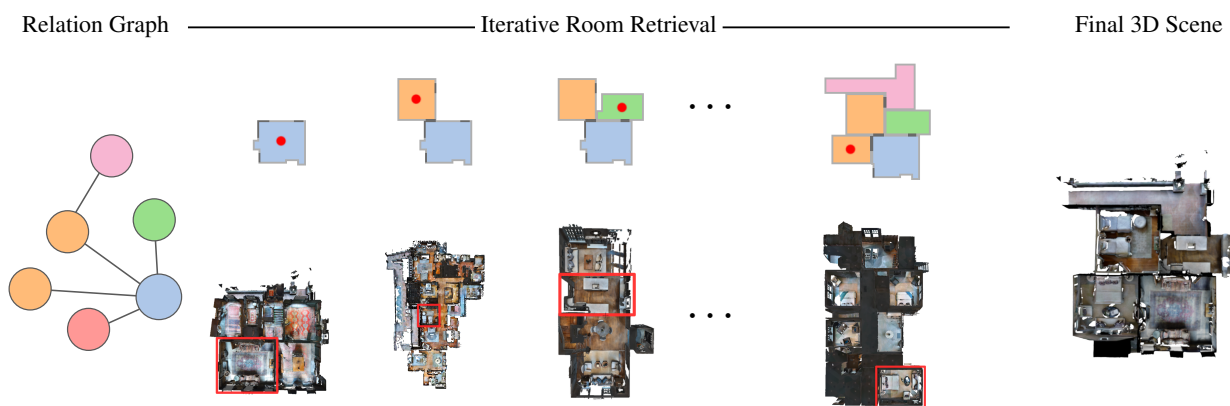# Roominoes:
Generating Novel 3D Floor Plans From Existing 3D Rooms

Kai Wang[1]   Xianghao Xu[1]   Leon Lei[1]   Selena Ling[1]   Natalie Lindsay[1]
Angel X. Chang[2]   Manolis Savva[2]   Daniel Ritchie[1]

[1]Brown University, United States
[2]Simon Fraser University, Canada

Figure 1: *We present Roominoes, a new task for creating house-level 3D environments by piecing together existing 3D rooms. We devise a spectrum of potential strategies to solve this task. In an instance of this task, one of our proposed algorithms creates a combinatorially novel layout by iteratively retrieving and assembling rooms from different 3D floor plans, deforming each room as needed.*

Abstract

Realistic 3D indoor scene datasets have enabled significant recent progress in computer vision, scene understanding, autonomous navigation, and 3D reconstruction. But the scale, diversity, and customizability of existing datasets is limited, and it is time-consuming and expensive to scan and annotate more. Fortunately, combinatorics is on our side: there are enough individual rooms in existing 3D scene datasets, if there was but a way to recombine them into new layouts. In this paper, we propose the task of generating novel 3D floor plans from existing 3D rooms. We identify three sub-tasks of this problem: generation of 2D layout, retrieval of compatible 3D rooms, and deformation of 3D rooms to fit the layout. We then discuss different strategies for solving the problem, and design two representative pipelines: one uses available 2D floor plans to guide selection and deformation of 3D rooms; the other learns to retrieve a set of compatible 3D rooms and combine them into novel layouts. We design a set of metrics that evaluate the generated results with respect to each of the three subtasks and show that different methods trade off performance on these subtasks. Finally, we survey downstream tasks that benefit from generated 3D scenes and discuss strategies in selecting the methods most appropriate for the demands of these tasks.

CCS Concepts
• Computing methodologies → Computer graphics; Neural networks; Probabilistic reasoning;

1 Introduction

Realistic 3D indoor scenes are useful for many tasks. In recent years, datasets of 3D scenes have stimulated advances in computer vision [ZF18, ZSS*18], multimodal scene understanding [AWT*18, CJS*20], 3D reconstruction [JSM*20, AKC*20],

and visual navigation [GDL*17, XZH*18, SKM*19]. In particular, multi-room 3D interiors have enabled work on long-horizon embodied AI tasks involving both manipulation and navigation.

Unfortunately, the scale and diversity of existing 3D indoor scene datasets is a bottleneck for this research agenda: they are not

very large nor configurable, and it is time-consuming and expensive to enlarge them. For example, existing datasets provide mostly 'detached' rooms (ScanNet [DCS*17]), or are limited to a small number of high-quality multi-room residences (Gibson [XZH*18], Matterport3D [CDF*17] and Replica [SWM*19]).

Instead of using 3D reconstructions, one could turn to synthetic scene datasets, such as AI2-THOR [KMG*17], CHALET [YMB*18] and 3D-FRONT [FCG*20]. One could also try to use generative models of furniture layouts to generate new synthetic 3D data [LPX*19, WLW*19, ZWK19]. Regardless of how synthetic scenes are created, there exists a "reality gap" between synthetic scenes and real ones, both in terms of content (e.g. synthetic scenes are not messy enough) and style (e.g. synthetic scenes are not photorealistic enough) [ZSY*17].

In this paper, we propose a new paradigm for creating house-level 3D environments by piecing together existing 3D rooms, a task that we call *Roominoes*. This mix-and-match strategy is similar to room-level 3D scene synthesis that selects from a database of 3D objects and arranges them to create a furnished room. Instead of arranging 3D objects, the Roominoes task requires arranging entire rooms. In other words, the rooms are building blocks to produce a coherent floor plan through a 'retrieve and edit' approach. This approach allows for scalability through combination of existing high-quality 3D rooms, as well as control of the level of complexity (from simple environments with few rooms, to complex environments with many rooms).

To analyze the challenges presented by the Roominoes task, we decompose it into three sub-tasks: (1) generating a 2D floor plan that is compatible with available 3D rooms, (2) retrieving a set of 3D rooms that can fit into the 2D floor plan, and (3) deforming individual 3D rooms so they fit the 2D floor plan. A Roominoes method must address these three sub-tasks. There are a spectrum of strategies for performing them, each with different tradeoffs. One could use existing 2D floor plans to provide plausible layouts; however, fitting available 3D rooms into those layouts may induce undesirably large deformations to the 3D rooms and the objects they contain. Alternatively, one could directly piece together existing 3D rooms without starting with a target layout; this will result in minimal distortion to the 3D rooms but may produce a less-realistic layout. In this paper, we implement and evaluate these two strategies. To measure the quality of the generated 3D environments, we introduce a set of evaluation metrics for measuring both the 2D layout quality and 3D mesh quality (e.g. amount of deformation, navigability). We also discuss the potential usefulness of these generated environments for downstream tasks, and we demonstrate the applicability of generated 3D houses on a visual navigation task.

We find that the Roominoes task is challenging, requiring non-trivial geometry processing to properly deform room architectures while keeping contained objects relatively undeformed. We believe that compositional generation of realistic multi-room 3D houses will become increasingly useful.

2 Background & Related Work

Multi-room 3D environment generation There is limited work that attempts to generate 3D environments with multiple rooms.

Procedural content generation for studying RL is starting to leverage generated 3D environments. However, these are mostly restricted to 3D mazes [BLT*16] or towers with rooms [JKB*19], which do not reflect the complexity of real-world furnished houses. A related line of work addresses procedural generation of floor plan layouts for residences and other buildings [MSK10, BYMW13, LYAM13]. These works, however, focus on generating the 3D architectural structure and do not typically produce realistically furnished 3D interiors.

Modeling by Assembly Our task is related to the long line of research on 3D modeling by retrieving and assembling parts. Early work in this area focused on interactive modeling, using either shape similarity search [FKS*04] or a learned probabilistic graphical model [CKGK11] to retrieve relevant parts for a user to combine. More recent approaches have automated both part retrieval and placement [KCKK12, SSK*17]. ComplementMe [SSK*17] is most similar to one of our approach: it trains a deep metric learning model to retrieve 3D parts that are compatible with a partial query shape. However, floor plans differ from part-based shapes both in data representation and in constraints that must be respected, which motivates us to develop a different formulation.

Indoor Scene Synthesis Our problem statement is similar to that of indoor scene synthesis, which aims to generate room layouts comprised of existing 3D objects, whereas we focus on generating 3D layouts from existing 3D rooms. One family of approaches to this problem attempts to place objects via optimization, typically a variant of Markov Chain Monte Carlo, with respect to learned priors, hand-crafted constraints, or some combination of the two [YYT*11, MSL*11, FRS*12, QZH*18]. Another family of approaches trains deep generative models to output scenes, using either a latent variable model such as Variational Autoencoder [LPX*19] or Generative Adversarial Network [ZYM*20] or using an iterative object-by-object approach [WSCR18, RWL19, WLW*19]. The iterative approaches are most similar to our setting. However, floor plan generation is a considerably different domain than furniture layout, as inter-room relationships require more geometric precision than most inter-object relationships (e.g. walls must align, portals must connect). Hence, a different approach is needed.

Floor Plan Generation There is a large body of existing work on generating 2D floor plans. As part of a larger system for generating 3D residential house models, Merrell et al. introduced a Bayesian network for modeling the room relationship graph along with an optimization-based approach for realizing this graph through precise wall geometry [MSK10]. Liu et al. also use an optimization-based approach as part of a machine-assisted interactive system for designing precast concrete buildings [LYAM13]. More recently, Wu et al. developed a mixed-integer quadratic programming (MIQP) formulation for optimizing building interior layouts [WFLW18]. Finally, the past year has seen the introduction of deep learning methods into this problem space, resulting in learning-based methods for generating floor plans given building outlines [WFT*19], room relationship graphs [NCC*20], or both [HHT*20]. These methods all address generation of 2D layouts with unconstrained room shapes. Very recent work takes a dif-

ferent approach of learning to generate a constraint graph and then solving the optimization problem posed by this graph to produce a final layout [PGK*20]. In contrast to all these approaches, we seek to solve the problem of generating new *3D* layouts, where the room shapes are constrained by an available set of existing 3D rooms.

3 Overview

Generating 3D floor plans using existing 3D rooms is a more constrained problem than typical 2D floor plan generation: one cannot generate rooms of arbitrary shapes, but instead is limited to the shapes available in an existing 3D room database. It would be too restrictive, however, to allow no changes to the 3D room geometry at all, as it is unlikely that an existing set of 3D rooms can be recombined perfectly into a novel floor plan. The likelihood of creating plausible layouts from existing rooms can be greatly increased if we allow deformations to the original rooms. By distorting the outline of the room and moving the positions of connecting doors or spaces (which we call *portals*), we can reuse the available 3D rooms in a more flexible manner and generate higher quality layouts. However, this approach comes at the cost of degrading 3D room quality, which decreases as the amount of distortion increases.

There is no definitive answer on how to trade off between 2D layout plausibility and 3D mesh quality; different applications may warrant different tradeoffs. Thus, we explore a spectrum of strategies which span this tradeoff space. We propose to break down the Roominoes task into three parts: i) generating a 2D floor plan that is compatible with available 3D rooms; ii) retrieving 3D rooms that can fit into the 2D floor plan; and iii) deforming the 3D rooms so they better fit the 2D floor plan. We note that the third step, room deformation, can be solved separately, provided that a correspondence can be found between each retrieved 3D room and its counterpart in the 2D floor plan. Thus, our exploration of possible strategies focuses on approaches for solving the first two steps, taking into account the interaction between them. In the remainder of this section, we first identify strategies for each of these steps and then propose two full pipelines, which we describe further in the following sections.

3.1 Generating 2D Floor Plans

3D interiors follow the structure of an underlying 2D floor plan. Taking into consideration that the rooms in a 2D floor plan must be matched with available 3D rooms, we propose the following strategies for obtaining a 2D layout:

Existing Dataset An obvious option for obtaining 2D floor plans is to draw them from an existing floor plan dataset, such as RPLAN [WFT*19] (80K annotated floor plans), or Li-FULL [LiF16] (millions of floor plan images). These datasets are far larger than their 3D scene counterparts: the Matterport3D [CDF*17] dataset, for example, contains only 184 floors. Using an existing floor plan guarantees the quality of the layout, but has other disadvantages. The first issue is lack of control: if one wants to replace a bedroom with a living room, or to make a balcony larger, it is unlikely that another floor plan in the dataset satisfies such demands. The second issue concerns the availability

Table 1: *Comparing different strategies for generating 2D layouts.*

Strategy	Layout Quality	Control	Deformation
Dataset	High	Low	Large
Generative Model	High	Medium	Large
Naive Portal Stitching	Low	High	None
Smart Portal Stitching	Medium	High	Medium

of 3D rooms which match the shape of the rooms in the 2D layouts. In general, 3D rooms with exact shape matches are not available. To fit the 3D rooms into the layout, large deformation is often necessary, which can impact the resulting 3D mesh quality.

Generative Model Instead of using floor plans drawn directly from a dataset, one could also use one of the many available data-driven generative models of 2D floor plans to generate novel 2D layouts [WFT*19, HHT*20, NCC*20]. This approach will improve the control over the resulting layout, since these methods usually accept some user input specification (e.g. a room relationship graph) and are capable of generating multiple possible layouts given the same input specification. However, this approach does not address the issue with 3D deformation artifacts.

Naive Portal Stitching One can avoid these deformation artifacts entirely by ignoring 2D layout altogether: instead of combining 3D rooms according to a layout, we can instead stitch rooms together by connecting pairs of portals. This approach removes the need to deform 3D rooms, but may produce an incoherent layout and is incompatible with downstream tasks that involve reasoning about the global scene layout.

Smart Portal Stitching Finally, one can attempt to find a "middle ground" trade-off between 2D layout plausibility and amount of 3D deformation by modifying the portal stitching method above: identifying 3D rooms that can likely fit together into a plausible layout without causing too much deformation.

Table 1 summarizes the strengths and weaknesses of these different strategies for obtaining a 2D layout.

3.2 Retrieving Compatible 3D Rooms

Depending on the strategy used to generate the 2D layout, the task of retrieving 3D rooms can be quite different. If the entire 2D layout is known in advance, then the task becomes finding the 3D rooms that best match the rooms in the layout (in terms of shape and/or portal placements). On the other hand, if the layout is determined on-the-fly (as in the two "Portal Stitching" approaches above), then the task becomes more ill-posed. As there is no 'ground truth' target room shape to match, one must instead retrieve rooms that are both geometrically compatible with the rooms already in the layout *and* and lead to a globally plausible layout.

3.3 Deforming 3D Rooms

The final task involves taking the retrieved 3D rooms and deforming them according to the 2D floor plan. The problem can be solved

in two steps: first computing a correspondence between the original room and the target room, and then applying a standard mesh deformation algorithm according to the defined correspondence. When the available 3D rooms come equipped with semantic object annotations, one can also leverage this information to avoid introducing visually objectionable artifacts such as non-rigid bending of semantically-meaningful rigid objects.

3.4 Task Paradigms

While the final task of deforming 3D rooms can be performed independently of the methods used for the first two tasks, these first two tasks are more correlated. We identify two major paradigms for solving them together:

- **2D before 3D:** Generate the 2D floor plan first and use its room shapes as ground truth shape targets for 3D room retrieval (i.e. the "Dataset" strategy in Table 1).
- **2D by 3D:** Build up a layout iteratively by retrieving and connecting 3D rooms (i.e. "Guided Portal Stitching" in Table 1).

In this paper, we propose one representative algorithm for each of the two paradigms. In Section 4, we describe a method that takes an existing 2D floor plan in the dataset, and retrieves 3D rooms similar according to the floor plan. In Section 5, we implement a method that uses deep metric learning to iteratively retrieve compatible 3D rooms and then apply a mixed integer quadratic programming (MIQP) optimization to assemble them into the final layout. Finally, in Section 6, we describes a strategy for deforming the 3D rooms, by first optimizing for a set of dense correspondences between the room outlines, and them applying cage deformation to deform the 3D rooms to the target outline.

4 Generating 2D Layout Before Retrieving 3D Rooms

In this section, we describe an algorithm that utilizes a large collection of available 2D floor plans to guide the retrieval of 3D rooms. Since the 2D layout is given by a floor plan from the dataset, the focus of this algorithm is on minimizing deformation of the retrieved 3D rooms needed to fit into the 2D floor plan. Since it is computationally expensive to perform deformation for all available 3D rooms, we instead develop approximate metrics to assess how much deformation would be required.

4.1 Data Preparation

As our source of 2D floor plans, we use the RPLAN dataset [WFT*19], a collection of 80,000 floor plan images annotated at pixel level. As in prior work, we apply additional filtering to the data: we make all walls the same thickness, retain only the largest portal between each pair of connecting rooms, and remove floor plans containing multi-purpose rooms.

As our source of 3D room data, we use the Matterport3D dataset [CDF*17], a collection of 184 floor plans containing 2056 rooms. For each 3D room, we build a 2D representation equivalent to those in the 2D floor plan dataset by extracting its outline (given by human annotations) and using raycasting to identify open regions on its walls, which we consider as portals. Since all of our

2D rooms are rectilinear, we also convert the non-rectilinear architecture of 3D rooms by computing their rectilinear bounding cages (i.e. the union of axis-aligned bounding boxes of all wall segments). We discard rooms that are not closed as well as rooms where a portal falls on a non-rectilinear part of the wall. We also discard room types not present in RPLAN. This leaves us with a final retrieval database of 1036 rooms. Finally, we apply a four way rotational augmentation of this dataset.

4.2 Retrieving Compatible 3D Rooms

Given a room from the 2D floor plan, we first filter out a subset of the 3D rooms that are compatible: those that have the same type label and contain same number of portals. For each 3D room in the filtered subset, we compute the following matching score between the source 3D room R_S and the target 2D room R_T, each containing k portals $P_S^0 \ldots P_S^{k-1}$ and $P_T^0 \ldots P_T^{k-1}$:

$$\lambda_a c_{\text{area}} + \lambda_o c_{\text{outline}} + \lambda_p c_{\text{portal}}$$

where

$$c_{\text{area}} = \frac{\max(A(R_S), A(R_T))}{\min(A(R_S), A(R_T))} - 1$$

penalizes large difference in room areas $A(R_S), A(R_T)$, which will lead to large uniform scaling in the deformation process.

$$c_{\text{outline}} = d_{\text{chamfer}}(O(R_S), O(R_T))$$

computes the two-way chamfer distance between a 250 point sample of the room outlines $O(R_S), O(R_T)$, with the rooms normalized to have unit area and centered. This penalizes room outline differences which lead to nonrigid deformations.

$$c_{\text{portal}} = \min_{j=0}^{k-1} \sum_{i=0}^{k-1} c_{\text{match}}(P_S^i, P_T^{(i+j)\bmod k})$$
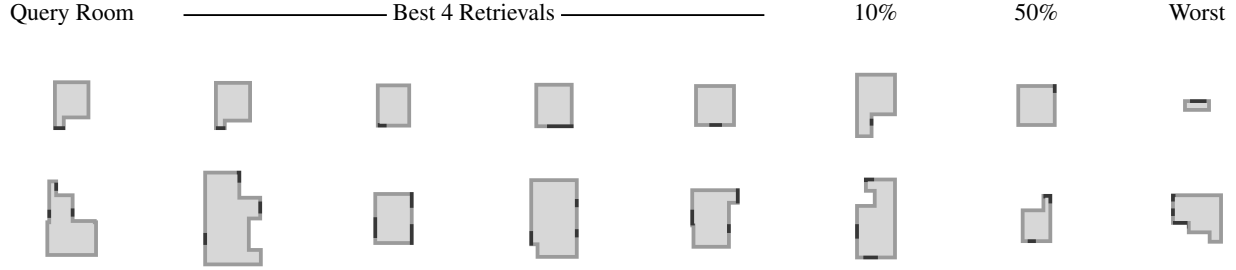
tries all possible k ways of pairing the k portals. There are only k ways because the order of the portals along the outline cannot be changed, and pairing any two portals uniquely determines the pairing of the rest. The portal matching cost c_{match} is:

$$\begin{aligned} c_{\text{match}}(P_1, P_2) = {} & (M(P_1) - M(P_2))^2 + \\ & \lambda_l(|P_1| - |P_2|)^2 + \\ & \lambda_d \mathbb{1}_{\{d(P_1) \neq d(P_2)\}} \end{aligned}$$
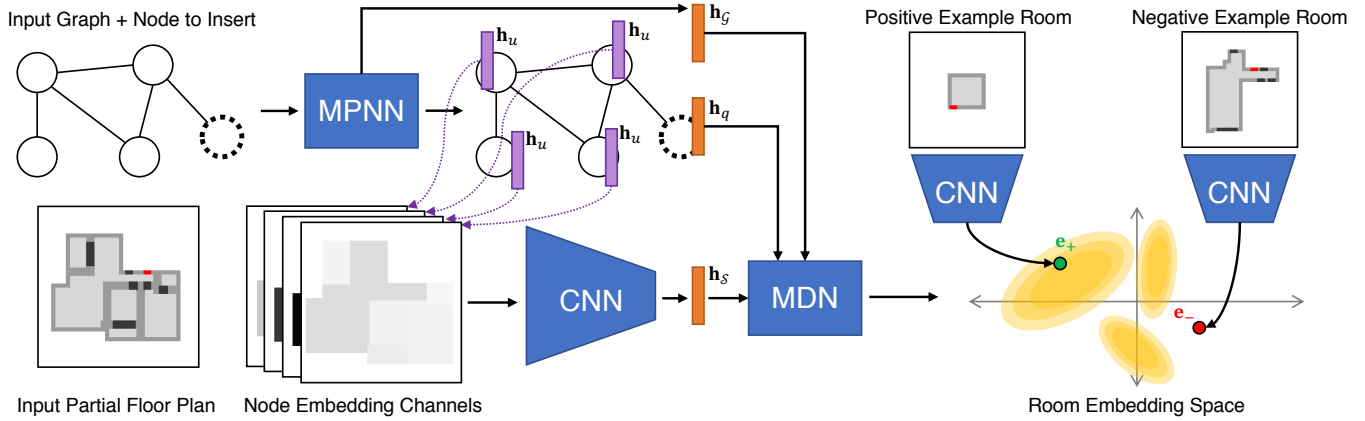
where $M(P)$ denotes the position of the midpoint of the portal in the 1D parametrized, unit length, outline of the room, $|P|$ denotes the length of the portal in the same 1D parametrization, and $\mathbb{1}$ is the indicator function that takes the value of 1 if the front-facing directions of the portals $d(P_1), d(P_2)$ are different. This penalizes matching portals that might lead to large deformation: we don't want to slide the portal along the wall by too much, scale it too much, or put it in the wrong orientation. We use $\lambda_a = 1, \lambda_o = 0.5, \lambda_p = 10, \lambda_l = 0.5, \lambda_d = 0.05$ for all experiments.

Figure 2 shows examples of rooms retrieved using this score. The score allows retrieval of rooms that are similar regarding size, shape and portal locations. However, due to the limited availability of 3D rooms, it becomes increasingly unlikely to get an exact match as the number of portals and the complexity of the room shape increase.

Query Room ——————— Best 4 Retrievals ——————— 10% 50% Worst

Figure 2: *Given a 2D room in a floor plan, we find similar 3D rooms that can be deformed into the 2D room with minimal changes with regard to room size, shape, and portal locations. In the top row, the top retrievals are compatible with respect to all criteria, whereas subsequent retrievals start to different in terms of shape, portal location, and finally size. The bottom row shows that the difficulty of finding a compatible 3D room increases substantially when the room shape becomes more non-rectangular, and when the number of portals is greater than* 1.

Figure 3: *The architecture of our neural network-based room retrieval module. The* embedding network *maps top down views of rooms into an embedding vector space (Right). Then, given an input floor plan relationship graph (with the node corresponding to the room to be inserted marked) and a top-down view of the current partial floor plan, the* retrieval network *predicts a Gaussian mixture probability distribution over this embedding space. The two networks are trained jointly such that positive example rooms (obtained by removing random rooms from ground-truth floor plans) have higher probability than negative example rooms (random distractors).*

5 Generating 2D Layout By Retrieving 3D Rooms

While guiding 3D retrieval with existing 2D floor plans ensures the quality of the resulting scene's 2D layouts, it can lead to large deformations of 3D rooms. To reduce the amount of deformation, it is possible to instead build the layout 'on the fly': retrieving 3D rooms one by one and iteratively assembling them into a plausible 2D layout, changing the rooms slightly as needed. In this section, we describe one pipeline that follows this paradigm. We extract a graph abstraction of an existing 2D floor plan, where each node encodes the type of the room and and each unlabeled edge represents a portal connection. Conditioned on the graph, we iteratively retrieve compatible 3D rooms and use an optimization procedure to assemble them into a partial floor plan, repeat these steps until the layout is complete.

5.1 Room Retrieval

Each iteration of the algorithm must retrieve a 3D room corresponding to a node in the input layout graph. The retrieved room needs to be compatible not only with the shapes of the other rooms retrieved so far, but also with the rooms retrieved later on, in order

to form a coherent floor plan. Instead of trying to manually select features that lead to such compatibility, we formulate the retrieval problem as a deep metric learning problem: we learn an embedding space of possible room shapes, where rooms which are compatible with the same layouts should be grouped together. This is similar in spirit to the approach taken by Sung et al. for learning to assemble part-based shapes [SSK*17]. Our neural architecture has two sub-networks: an *embedding network* and a *retrieval network*. The embedding network maps individual 3D rooms into a high-dimensional vector space; the retrieval network maps an input partial floor plan, as well as a relation graph of the entire floor plan, into a probability distribution over this space. The two networks are trained jointly using a contrastive learning framework, i.e. rooms that are known to fit well with a layout in the training set should map to higher-probability points than other rooms. Figure 3 shows the architecture of these networks and their interaction; the rest of this section describes them in more detail.

Embedding Network The embedding network maps a 2D room r into an embedding vector $\mathbf{e}_r$. For this, we pass an image-based representation of the room through a convolutional neural network

(CNN). The image representation uses separate channels to represent the rooms, walls, portals, as well as the specific portal to which we want to connect the next room.

Retrieval Network The retrieval network takes the current partial floor plan $\mathcal{S}$, the target floor plan relation graph $\mathcal{G}$, and the node q corresponding to the room to be inserted, and predicts a conditional probability distribution $P(\mathbf{e}_r \mid q, \mathcal{G}, \mathcal{S})$ over room embeddings $\mathbf{e}_r$. Similarly to the work of Sung et al. [SSK*17], we model this distribution as a mixture of Gaussians over a room shape embedding space. We use a mixture density network [Bis94] to predict this distribution.

Since the features of $\mathcal{G}$ and $\mathcal{S}$ are highly correlated, we use a single neural network to learn from both simultaneously. We first use a message passing neural network [GSR*17] to extract information from the relationship graph $\mathcal{G}$. After T rounds of message propagation, we predict a per-node descriptor $\mathbf{h}_u$, as well as a descriptor of the entire graph $\mathbf{h}_\mathcal{G}$. To extract information from the partial floor plan, we use a CNN conditioned on an image-based representation of the floor plan to predict an image feature $\mathbf{h}_\mathcal{S}$. The image-based representation is the same as that used by the embedding network, with an additional set of $|\mathbf{h}_u|$ channels. These channels encode the corresponding node embedding $\mathbf{h}_u$ for each room in the layout. Finally, we concatenate the image feature $\mathbf{h}_\mathcal{S}$, the graph descriptor $\mathbf{h}_\mathcal{G}$, as well as the descriptor for the room to be retrieved $\mathbf{h}_q$, and feed it to a mixture density network, which predicts the parameters of the Gaussian mixture, where the k-th Gaussian is parameterized by a weight ϕ_k, a mean μ_k and a standard deviation σ_k.

Joint Training We train the embedding and retrieval networks jointly using a triplet loss. Given a room embedding $\mathbf{e}_r$, we define the log likelihood that the embedding is sampled from the mixture distribution predicted by the retrieval network:

$$\ell^{q,\mathcal{G},\mathcal{S}}(\mathbf{e}_r) = \log \sum_{k=1}^{N} \phi_k(q,\mathcal{G},\mathcal{S}) \cdot \mathcal{N}(\mathbf{e}_r \mid \mu_k(q,\mathcal{G},\mathcal{S}), \sigma_k(q,\mathcal{G},\mathcal{S})^2)$$

Using this embedding loss, we define a contrastive loss [CSSB10] between an embedding of a positive example room $\mathbf{e}_+$ and embedding of a negative example room $\mathbf{e}_-$:

$$\mathcal{L}^{q,\mathcal{G},\mathcal{S}}(\mathbf{e}_+, \mathbf{e}_-) = \max\{m + \ell^{q,\mathcal{G},\mathcal{S}}(\mathbf{e}_-) - \ell^{q,\mathcal{G},\mathcal{S}}(\mathbf{e}_+), 0\}$$

where $m = 10$ is a constant margin. In other words, this loss encourages positive example rooms to have a higher predicted likelihood than negative example rooms. Positive example rooms are created by removing a room from one of the training layouts. Negative examples are created by sampling rooms from other floor plans. For rooms with more than one portal, we randomly select a subset of portals to show in the query portal mask.

We use $T = 5$ rounds of propagation for the graph neural network. We choose $|\mathbf{h}_u| = 64$ as the dimension of the per node embedding and $\mathbf{h}_\mathcal{G} = 128$ as the dimension of the graph embedding. The CNN contains seven layers, without batch normalization, which we find to be detrimental to the training process. We use $|\mathbf{h}_\mathcal{S}| = 512$ as the dimension of predicted image feature. For the retrieval network, we predict a mixture of 20 Gaussians over an 128 dimensional embedding space. We use a margin of 10 for the triplet contrastive loss. We train the neural networks using the Adam [KB15] optimizer, and with a batch size of 16. At each batch, negative examples are sampled from 16 randomly selected floor plans. We start with uniformly selecting from these examples, gradually favoring harder examples as the training proceeds.

Figure 4 shows examples of rooms predicted by the retrieval network. The network learns to retrieve rooms of the correct shape, correct type, and the correct number of portals. Note that the networks are not trained to recognize the possibility of using a room after rotations. To allow for this, we apply a four way rotational augmentation to all the candidate 3D rooms instead.

5.2 Placing Retrieved Rooms into the Floor Plan

We use the trained embedding and retrieval networks to iteratively retrieve and insert rooms into the layout. We determine the location of the new room by matching the position of the portal to the connecting rooms. When multiple such portals exist, we randomly select one for initialization.
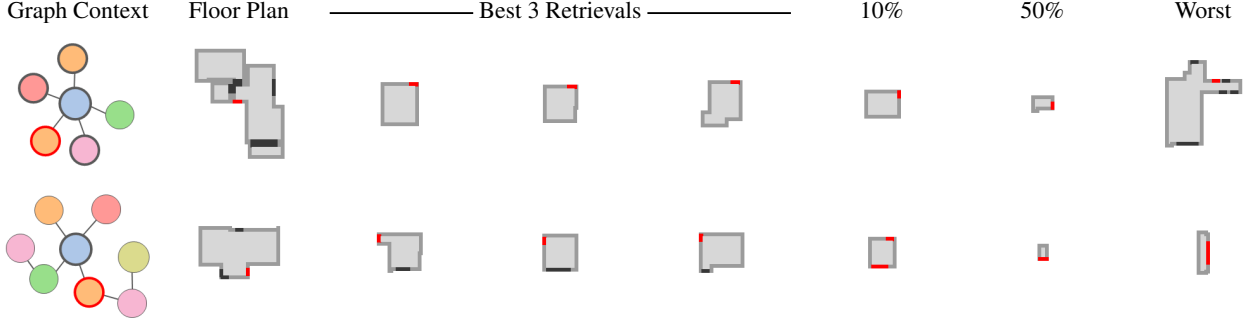
However, it is unlikely that the retrieved room will fit perfectly into the current partial floor plan. We use beam search to increase the chance that we find better fits. To resolve remaining inconsistencies, we apply an optimization procedure after each step of room retrieval. Inspired by prior work [WFLW18, PGK*20], we adopt a mixed integer quadratic programming (MIQP) based procedure for layout optimization. The optimization procedure has two goals. The first is to ensure that the partial floor plan is valid, i.e. each room has a positive area, no overlaps exist between different rooms, and all paired portals align with each other. Figure 5 shows examples of this process. The second goal is to avoid obvious semantic issues in the floor plan. We tackle one such issue, interior voids in the floor plan, by encouraging the rooms to be adjacent to each other, though additional heuristics could be employed to improve the quality of the 2D floor plans further. Figure 6 shows the effect of encouraging room adjacency. These goals must be achieved while minimizing deformation of the room outlines and the portal locations. The precise formulation of the objective function and constraints can be found in the supplemental material.
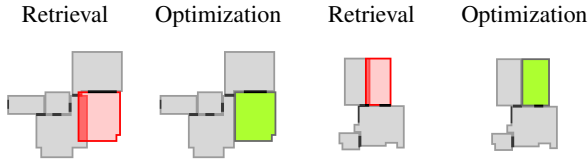
6 Deforming Retrieved 3D Rooms

Having generated a 2D layout and retrieved 3D rooms, the final task involves deforming the individual room geometries to fit the layout. This is a nontrivial problem: the 3D room must be warped to fit the new 2D outline, but we must take care to avoid introducing visually-objectionable artifacts, most prominently non-rigid distortion to semantically-meaningful objects within the room.

Deforming a 3D room to fit an optimized floor plan layout amounts to solving an analogy problem: given the source 3D room mesh $\mathcal{M}_S$ and its 2D outline polygon $\mathcal{P}_S$, as well as the target 2D outline polygon $\mathcal{P}_T$ produced by the optimizer, what is the corresponding target 3D room mesh $\mathcal{M}_T$?
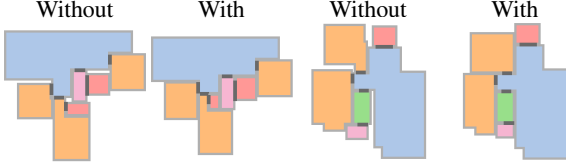
In this section, we describe a two-step solution. First, we warp the source outline $\mathcal{P}_S$ onto the target outline $\mathcal{P}_T$ by optimizing for a dense correspondence between them. Then, we use the warped outline as a control cage for cage-based deformation of the source mesh $\mathcal{M}_S$ to produce the target mesh $\mathcal{M}_T$.

Graph Context   Floor Plan   ——— Best 3 Retrievals ———   10%   50%   Worst

Figure 4: *Our learned retrieval network retrieves rooms to add to an in-progress layout. The three highest-probability retrievals are good matches and permit the rest of the graph to be completed later. In the top row, the top 3 retrievals are all medium-sized rooms that can be easily inserted into the available corner given their portal placements. In the bottom row, the query node in the graph specifies that the new room should have two portals, and all the top 3 retrieval results do. The quality of retrieval falls off further down the probability-ordered list of rooms in our validation set, exhibiting errors such as incorrect room types, incorrect shapes, and incorrect numbers of portals.*

Retrieval   Optimization   Retrieval   Optimization

Figure 5: *Examples of a 2D floor plan before and after our optimization-based snapping. From left to right: fixing a room slightly too large to fit in a small corner; sliding a pair of portals to make the new room match better.*

Without   With   Without   With

Figure 6: *Two examples illustrating the effect of rewarding the layout optimizer for maximizing adjacencies between different rooms.*

Finding a correspondence between outlines Our first step in deforming the 3D room mesh to fit the new target room outline $\mathcal{P}_T$ is to first deform its 2D outline $\mathcal{P}_S$ to the target outline. As both outlines are closed, one-dimensional, piecewise linear curves, this problem reduces to finding a correspondence between $\mathcal{P}_S$ and $\mathcal{P}_T$. We solve this problem by sampling a finite set of N 1D points $u_S^0 \ldots u_S^{N-1}$ along the source outline $\mathcal{P}_S$ (including all of its corner points) and then optimizing for a corresponding set of 1D points $u_T^0 \ldots u_T^{N-1}$ on $\mathcal{P}_T$, by minimizing the following objective:

$$\sum_{i=0}^{N-1} \frac{\lambda_e}{N} c_{\text{elasticity}}(i) + \frac{\lambda_n}{N} c_{\text{normal}}(i) - r_{\text{corner}}(i)$$

where

$$c_{\text{elasticity}}(i) = \left((u_T^{(i+1)\bmod N} - u_T^i) - \frac{|P_T|}{N} \right)^2$$

is the elasticity regularization term that encourages the output points u_T^i to be spread out evenly on the target outline,
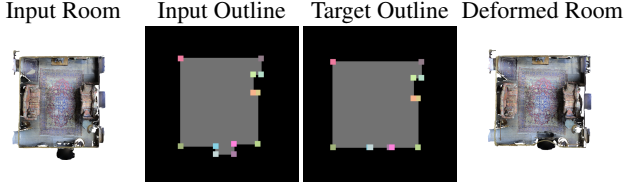
$$c_{\text{normal}}(i) = \theta_i \left(x_T^{(i+1)\bmod N} - x_T^i \right)^2 + (1 - \theta_i) \left(y_T^{(i+1)\bmod N} - y_T^i \right)^2$$

is the normal matching term that encourages originally vertical segments to stay vertical ($\theta_i = 1$) and originally horizontal segments to stay horizontal ($\theta_i = 0$), where (x_T^i, y_T^i) is the 2D position of u_T^i along $\mathcal{P}_T$, and $r_{\text{corner}}(i) = \sigma_S^i \sigma_T^i$ is the term that encourages source corner points to correspond to target corner points ($\sigma^i \in \{0, 1\}$ is a constant indicating whether the point i is a corner in the source/target).

In addition, we impose the constraint that $u_T^i < u_T^{i+1}$ for all points i, which enforces that the sequence of points remains monotonic. To make sure that the portals fall on the right segment of the outline after deformation, we impose the additional constraint that for each pairs of portals p_S^i and p_T^i, the endpoints of the original portal μ_S^{ia} and μ_S^{ib} falls within the corner points μ_T^{ia} and μ_T^{ib} that contain the target portal. For both the source and target outlines, we define $u = 0$ to occur at the upper-left-most corner of the outline.

Outline-driven deformation Given the warped outline produced by the correspondence step, we drive a cage-based deformation of the room mesh. That is, $\mathcal{P}_S(u_S^0 \ldots u_S^{N-1})$ is treated as the initial polygonal control cage for the mesh, and $\mathcal{P}_T(u_T^0 \ldots u_T^{N-1})$ provides the new positions of the cage vertices. We then interpolate the positions of all interior mesh vertices using mean value coordinates (MVC) [Flo03]. Finally, we deform each wall where a portal falls on to make sure the portals fall on the right position. Figure 7 shows an example of these procedures applied to a 3D room.

Handling rigid objects The deformation scheme described above non-rigidly distorts the room geometry. Since room retrieval and layout optimization try to retrieve and place rooms that fit into a layout together without changing their shape much, this distortion is typically small and unobjectionable. However, when objects within a room undergo such deformations, it can result in semantically-implausible bending artifacts. To prevent such arti-

Input Room   Input Outline   Target Outline   Deformed Room

Figure 7: *Example of a 3D room deformed to fit an optimized 2D outline. From left to right: the initial 3D room mesh; the 2D outline for the input mesh (with a subset of sample points $u_S^0 \ldots u_S^{N-1}$ colored); the target 2D outline for the deformation (with corresponding points $u_T^0 \ldots u_T^{N-1}$ colored); the final deformed room geometry.*

Input Room   w/out Rigid Objects   w/ Rigid Objects

Figure 8: *Examining the effect of treating objects as rigid in our 3D room deformation procedure. From left to right: a view of the initial 3D room mesh; the room deformed without treating objects as rigid; the room deformed while treating objects as rigid. Enforcing object rigidity prevents semantically implausible bending artifacts.*

facts, we cut all labeled objects other than those with labels floor, ceiling, wall or curtain out of the mesh, deform the rest of the mesh using the scheme above, and then insert the objects back into the deformed room mesh. To determine object insertion positions, we move their original centroid position according to the MVC deformation. In some cases, this results in placing the object in a position that intersects with other objects or the room geometry. If this occurs, we push the object away from the collision until the intersection is resolved. If doing so results in another intersection before the first is resolved, we uniformly scale the object down until the collision is resolved. Figure 8 shows deformation results with and without this special logic for handling rigid objects.

7 Evaluation Strategies for 3D Layout Generation

In this section, we propose a set of metrics for evaluating the effectiveness of methods that solve the Roominoes task. First, we identify the properties of the output that are of interest to downstream tasks. We then define metrics that estimate the quality of the results, without having to spend many machine hours to actually test the generated data on downstream tasks. Finally, we demonstrate that 3D scenes generated by Roominoes can be useful for downstream tasks.

Table 2: *Comparing different methods for 2D floor plan generation.*

Method	FID ⇓	FD[R] ⇓	% fool ⇑
Rectangles	63.49	450.74	1.34
Dataset	6.84	1.29	53.54
Generative Model	9.36	2.72	46.86
Smart Portal Stitching	18.61	86.05	28.72
Smart Portal Stitching (no net)	22.40	90.20	25.00

7.1 Evaluating 2D Layout Quality

The quality of a generated floor plan's 2D layout is important for tasks that require reasoning about the global structure of a scene, e.g. indoor navigation. We propose the following metrics to evaluate 2D layout quality:
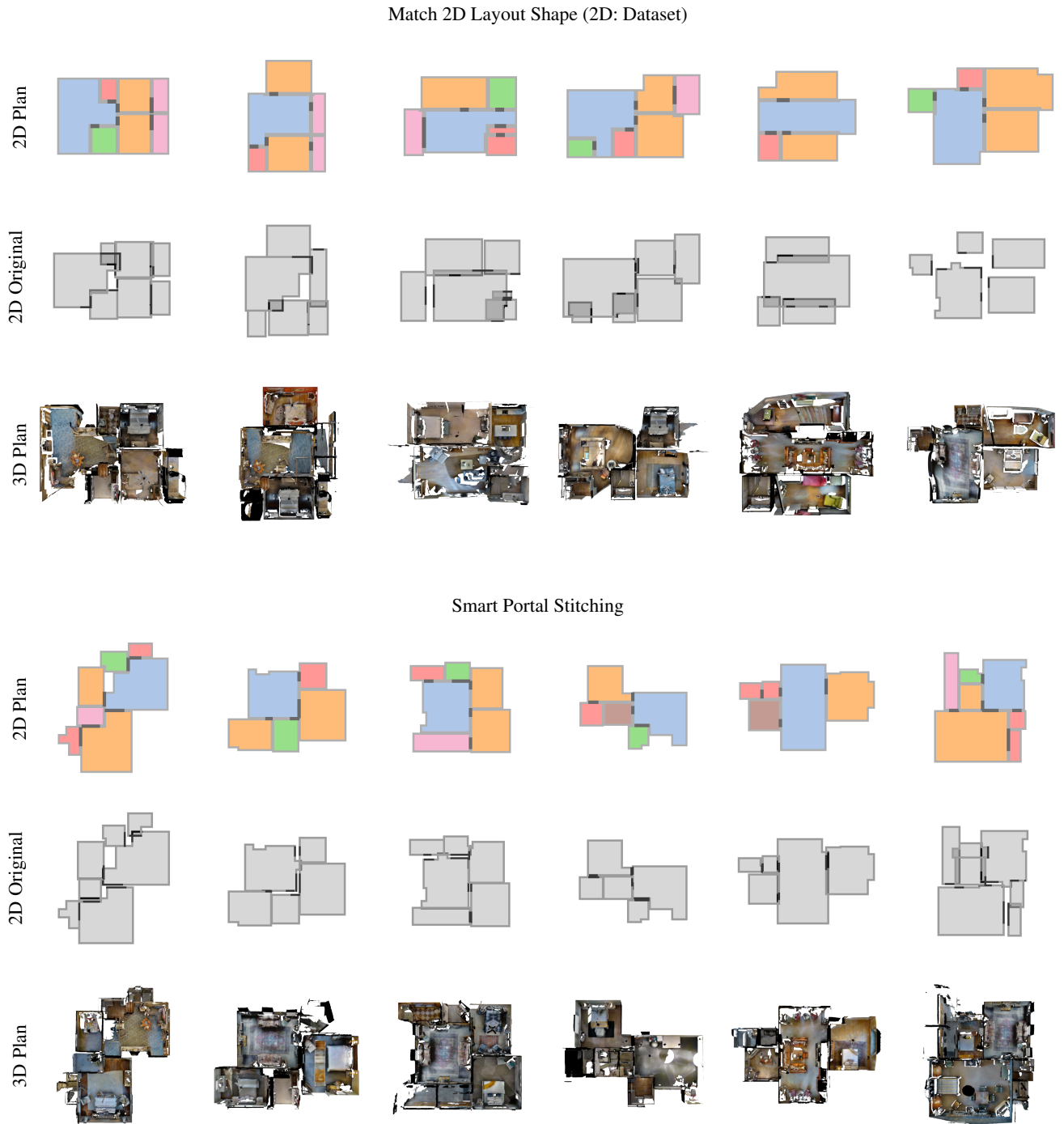
- **Fréchet Distance (FD)**: a measure of distributional similarity between the generated floor plans and those in a held-out test set [HRU*17]. We evaluate this distance in two different feature spaces: a pre-trained Inception image classifier, i.e. the standard Fréchet Inception Distance (**FID**), and the CNN component of the retrieval network described in Section 5 (**FD[R]**). The first is more general, whereas the second is domain-specific.
- **Classifier Fool Percentage**: the percentage of held-out test layouts classified as "generated" by a classifier trained to distinguish between generated 2D layouts and layouts from the training dataset. A value of 50% would suggest that the two sets of layouts are indistinguishable from each other.

We compare the following sources of 2D floor plans:

- **Rectangles:** Randomly place N rectangles, where N equals the number of nodes in the graph. This simple baseline establishes a lower bound on performance.
- **Dataset:** Floor plans drawn from a dataset (RPLAN [WFT*19]) that are not used for training the following models.
- **Generative Model:** Generate floor plan given the polygonal outline of the building, using a recent method [WFT*19].
- **Smart Portal Stitching**: Build a 2D layout while retrieving 3D rooms using the algorithm described in Section 5.
- **Smart Portal Stitching (no net)**: Use the algorithm from Section 5, but do not use the learned retrieval network; instead, retrieve random rooms with correct type and portal number.

Table 2 shows the results of this experiment. As expected, sourcing 2D layouts from a floor plan dataset or a data-diven 2D floor plan generative model produces the highest quality layouts. The smart portal stitching methods sacrifice layout plausibility, although the version with the learned retrieval network performs better. Figure 9 shows examples of dataset layouts vs. those produced by smart portal stitching. The layouts generated by the smart portal stitching method have less consistent outlines and sometimes contain implausible features, such as interior holes. In extreme cases, the layout quality can be too low to be useful, as visualized in Figure 10a and 10b.

Match 2D Layout Shape (2D: Dataset)

Smart Portal Stitching

Figure 9: *Novel 3D house meshes generated by the two representative strategies, visualized along with the 2D floor plan, and an overlay of the original room shapes and portal location. The Match 2D Layout Shape strategy obtains better layouts by directly using existing 2D floor plans. However, the 3D rooms do not match the floor plan well, leading to large deformations and visual artifacts. In contrast, the Smart Portal Stitching strategy requires minimal deformation to the retrieved 3D room outlines and portals, but at the cost of lower layout quality.*

Table 3: *Comparing different methods for retrieving 3D rooms in terms of their effect on final mesh quality. Lower is better.*

Method	Area	Outline	Portal	Mesh A	Mesh E
Match 2D Layout Shape	16.31	7.776	2.758	4.47	3.19
Smart Portal Stitching	10.45	0.026	0.008	2.42	1.83
Smart Portal Stitching (no net)	19.32	0.460	0.752	3.47	2.30

Table 4: *Evaluating how well a pretrained DDPPO [WKM*20] agent navigates scenes generated from different sources. Higher is better.*

Scene Source	Success Rate	SPL
Match 2D Layout Shape	0.783	0.608
Smart Portal Stitching	0.759	0.618
MP3D Scenes	0.876	0.785

Table 5: *Navigation agent performance for agents trained on scenes generated using our approach and original Matterport3D dataset scenes. Higher values are better. Evaluation is done on the Gibson [XZH*18] dataset, in scenes unseen at training time.*

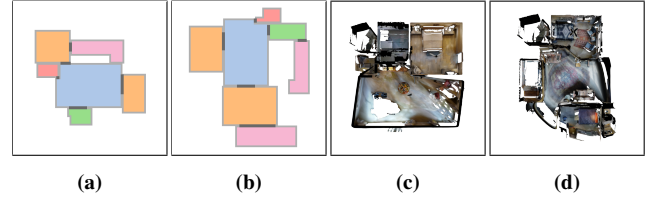| Scene Source	Success Rate	SPL
Smart Portal Stitching	0.753	0.484
MP3D Scenes	0.818	0.628

7.2 Evaluating 3D Mesh Quality

Large deformations of 3D room meshes will degrade perceived visual quality of the output scenes, which is important to most downstream tasks. We propose the following metrics to evaluate mesh quality in the presence of deformations:

- **Area Change (Area):** Percent change in the area of the room's 2D outline after deformation.
- **Outline Change (Outline):** Average distance between corresponding points on the outlines
- **Portal Change (Portal):** Average distance between the midpoint of the portal after the deformation and the final portal location (obtained through deforming the wall segment)
- **Mesh Statistics**: For each mesh, we discretize the area of each triangle and the length of each edge into 64 bins, and compute the Wasserstein distance of the triangle area distribution (**Mesh A**) and edge length distribution (**Mesh E**).

We use these metrics to compare the following 3D room retrieval methods:

- **Match 2D Layout Shape:** Retrieve rooms whose shape best matches the shape of their corresponding room in a given input 2D floor plan (i.e. the algorithm from Section 4).
- **Smart Portal Stitching:** Same as above.
- **Smart Portal Stitching (no net):** Same as above.

Table 3 shows the results of this experiment. The smart portal stitching strategy requires significantly fewer changes to the retrieval 3D rooms, with respect to the size, outline shape, portal

| (a) | (b) | (c) | (d) |

Figure 10: *Failure cases of different algorithms. (a, b): the Smart Portal Stitching strategy generates a low quality 2D layout that contains large holes or non-smooth outlines; (c, d): the Match and Deform strategy fails to find reasonably similar living rooms, leading to large deformations in the final mesh.*

locations, and mesh properties. Using a neural network further amplifies this effect. On the other hand, due to the limited availability of 3D rooms, the match and deform strategy struggles to find 3D rooms that match exactly, and thus has to modify each room much more. Figure 9 illustrates the differences in terms of amount of change required and impact on the 3D mesh quality. In extreme cases, the retrieved rooms are so incompatible that leads to unacceptable mesh quality post deformation. Figure 10c and 10d shows two such examples.

7.3 Evaluating on Downstream Tasks

To demonstrate that the 3D house scenes we generate through Roominoes can be useful for downstream tasks, we use them for two small-scale indoor visual navigation experiments. For the first experiment, we use a DDPPO agent [WKM*20] trained on the Gibson [XZH*18] dataset in the Habitat simulation environment [SKM*19]. We task the agent with performing a series of point goal navigation tasks in scenes generated by our method, as well as in original Matterport3D scenes. To evaluate how well the agent navigates these scenes, we use two standard metrics from the visual navigation literature: Success Rate and Success Weighted by Path Length (SPL) [ACC*18]. The pre-trained agent can navigate reasonably well in the scenes generated by both methods. Additional qualitative examples of agents navigating in the generated scenes are in the supplementary material. For the second experiment, we train two agents with proximal policy optimization [SWD*17] with the settings used in [SKM*19], one on 104 floor plans generated by the Smart Portal Stitching strategy, and the other on 184 floor plans from Matterport3D. Both agents are trained with 30 million steps. We then evaluate the performance of the agents on the Gibson [XZH*18] validation set as used in the Habitat challenge [SKM*19]. Table 5 summarizes the results. Agents trained on our data achieve comparable success rate to the agent trained on the original Matterport3D data, though with relatively lower SPL. These experiments suggest that data generated by our method is of comparable quality to original Matterport3D data with respect to indoor visual navigation. A full-scale evaluation of the impact of data augmentation with our data is computationally challenging, as it has been shown that navigation agents continue to learn from data after billions of steps [WKM*20]. The supplemental material reports on an experiment intended to approximate this setting; we leave rigorous, full-scale evaluation to future work. Regardless, it has been shown that additional training

data is beneficial to the performance of agents, even if the additional data is of poor quality [WKM*20]. Thus, we believe that the scenes we generate, which we have shown to be of reasonable quality, can be used to improve the performance of navigation agents further.

7.4 Timing

It takes on average about 30 seconds for the Match 2D Layout Shape strategy to complete a 2D layout. The Smart Portal Stitching strategy, in contrast, requires about 10 minutes per floor plan. Virtually all of the time is spent on 2D layout optimization, where per-room-insertion optimization time varies considerably depending upon layout complexity (from 0.1 to 60 seconds, which is our hard time limit). We note that the time spent on optimization can be reduced from 10 minutes down to 1 minute without beam search. Computing outline correspondence takes about 1 minute per floor plan with 250 point samples. This can be reduced drastically if done with fewer point samples. Finally, deforming the 3D mesh takes 3 minutes per floor plan, primarily due to the large number of vertices we have to handle.

8 Conclusion

In this paper, we presented Roominoes, a new task for creating house-level 3D environments by mixing-and-matching existing 3D rooms. We examined the possible solutions to three sub-tasks, and implemented two representative algorithms for solving the task. As discussed in Section 7, we found Roominoes to be challenging, primarily due to the need to trade-off between 2D layout quality and 3D mesh quality. There are several potential directions one can take to address such challenges. First, the 2D floor plan quality for the "2D by 3D" class of methods can be improved by designing additional semantic features capturing what constitutes a good floor plan. For example, one could attempt to reduce the number of corners of the generated 2D floor plans to make them more regular, or fine-tune the shape of individual rooms to make the floor plan more similar to real ones. The 3D room deformation procedure can also be improved. Our current method does not fare well in the case of large deformations, particularly when portals need to slide significantly along walls. Future work might instead perform additional mesh surgery e.g. cutting and pasting the portal regions to a different part of the wall, instead of deforming the entire wall, or copying and pasting some existing regions, instead of applying large deformations to those regions. Doing so leads to the need to synthesize new textures, which is a challenging problem on its own. Currently, there is no guarantee that the layout of the 3D room post-deformation is still realistic. Thus, it would be beneficial to augment the control points with semantic layout rules learned from available 2D layout data. This can be extended further to take into account relations between individual rooms. It is possible to experiment with additional 3D room datasets such as ScanNet [DCS*17] or Gibson [XZH*18], in order to increase the chance of retrieving rooms that fit the target floor plan well. Finally, a more thorough evaluation in downstream tasks such as visual navigation, as well as other tasks that benefit from 3D scene datasets can reveal what properties of the generated scenes are most beneficial in such tasks.

Beyond addressing these limitations, there are several broader

avenues for future work. For instance, one can extend Roominoes to support on-demand data generation, creating scenes tailored to address scenarios where methods for downstream tasks are underperforming. It is also worth doing more investigation on strategies that bridge the gap between different datasets: in this work, we simply combined one dataset of 2D floor plans with another dataset of 3D rooms. While the performance is acceptable, a more careful investigation addressing differences between datasets can potentially improve the generalization performance of the algorithms.

Finally, we would be excited to see Roominoes applied at scale to more of the applications that originally motivated its creation: computer vision, scene understanding, 3D reconstruction, and embodied AI. There are a number of interesting questions for future work. How do we find the right trade-off between 2D and 3D quality depending on the type of downstream tasks? Does training models for these applications on massive sets of Roominoes-generated 3D floor plans improve their generalization performance? And is it possible to reduce the number of training scenes needed if those scenes are tailored to the task(s) the method must address? These are some exciting questions that would be fruitful to explore with Roominoes.

References

[ACC*18] ANDERSON P., CHANG A., CHAPLOT D. S., DOSOVITSKIY A., GUPTA S., KOLTUN V., KOSECKA J., MALIK J., MOTTAGHI R., SAVVA M., ZAMIR A. R.: On evaluation of embodied navigation agents, 2018. arXiv:1807.06757. 10

[AKC*20] AVETISYAN A., KHANOVA T., CHOY C., DASH D., DAI A., NIESSNER M.: SceneCAD: Predicting object alignments and layouts in RGB-D scans. In *Eur. Conf. Comput. Vis.* (2020). 1

[AWT*18] ANDERSON P., WU Q., TENEY D., BRUCE J., JOHNSON M., SÜNDERHAUF N., REID I., GOULD S., VAN DEN HENGEL A.: Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 3674–3683. 1

[Bis94] BISHOP C. M.: *Mixture density networks*. Aston University, 1994. 6

[BLT*16] BEATTIE C., LEIBO J. Z., TEPLYASHIN D., WARD T., WAINWRIGHT M., KÜTTLER H., LEFRANCQ A., GREEN S., VALDÉS V., SADIK A., ET AL.: Deepmind lab, 2016. arXiv:1612.03801. 2

[BYMW13] BAO F., YAN D.-M., MITRA N. J., WONKA P.: Generating and exploring good building layouts. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–10. 2

[CDF*17] CHANG A., DAI A., FUNKHOUSER T., HALBER M., NIESSNER M., SAVVA M., SONG S., ZENG A., ZHANG Y.: Matterport3D: Learning from RGB-D data in indoor environments. In *3DV* (2017). 2, 3, 4

[CJS*20] CHEN C., JAIN U., SCHISSLER C., GARI S. V. A., AL-HALAH Z., ITHAPU V. K., ROBINSON P., GRAUMAN K.: SoundSpaces: Audio-visual navigation in 3D environments. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2020). 1

[CKGK11] CHAUDHURI S., KALOGERAKIS E., GUIBAS L., KOLTUN V.: Probabilistic reasoning for assembly-based 3D modeling. *ACM transactions on graphics (TOG) 30*, 4 (2011), 1–10. 2

[CSSB10] CHECHIK G., SHARMA V., SHALIT U., BENGIO S.: Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research 11*, Mar (2010), 1109–1135. 6

[DCS*17] DAI A., CHANG A. X., SAVVA M., HALBER M., FUNKHOUSER T., NIESSNER M.: ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2017). 2, 11

[FCG*20] FU H., CAI B., GAO L., ZHANG L., WANG J., LI C., ZENG Q., SUN C., JIA R., ZHAO B., ZHANG H.: 3D-FRONT: 3D furnished rooms with layouts and semantics, 2020. arXiv:2011.09127. 2

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM transactions on graphics (TOG) 23*, 3 (2004), 652–663. 2

[Flo03] FLOATER M. S.: Mean value coordinates. *Comput. Aided Geom. Des. 20*, 1 (Mar. 2003), 19–27. 7

[FRS*12] FISHER M., RITCHIE D., SAVVA M., FUNKHOUSER T., HANRAHAN P.: Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 1–11. 2

[GDL*17] GUPTA S., DAVIDSON J., LEVINE S., SUKTHANKAR R., MALIK J.: Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2616–2625. 1

[GSR*17] GILMER J., SCHOENHOLZ S. S., RILEY P. F., VINYALS O., DAHL G. E.: Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning* (2017). 6

[HHT*20] HU R., HUANG Z., TANG Y., VAN KAICK O., ZHANG H., HUANG H.: Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 39*, 4 (2020). 2, 3

[HRU*17] HEUSEL M., RAMSAUER H., UNTERTHINER T., NESSLER B., HOCHREITER S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017), pp. 6629–6640. 8

[JKB*19] JULIANI A., KHALIFA A., BERGES V.-P., HARPER J., TENG E., HENRY H., CRESPI A., TOGELIUS J., LANGE D.: Obstacle tower: A generalization challenge in vision, control, and planning. In *International Joint Conference on Artificial Intelligence (IJCAI)* (2019). 2

[JSM*20] JIANG C., SUD A., MAKADIA A., HUANG J., NIESSNER M., FUNKHOUSER T.: Local implicit grid representations for 3D scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2020). 4

[KB15] KINGMA D. P., BA J.: Adam: A Method for Stochastic Optimization. In *Int. Conf. Learn. Represent.* (2015). 6

[KCKK12] KALOGERAKIS E., CHAUDHURI S., KOLLER D., KOLTUN V.: A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 1–11. 2

[KMG*17] KOLVE E., MOTTAGHI R., GORDON D., ZHU Y., GUPTA A., FARHADI A.: AI2-THOR: an interactive 3D environment for visual AI, 2017. arXiv:1712.05474. 2

[LiF16] Lifull home's dataset. https://www.nii.ac.jp/dsc/idr/en/lifull/, 2016. Accessed: 2016-11-2. 3

[LPX*19] LI M., PATIL A. G., XU K., CHAUDHURI S., KHAN O., SHAMIR A., TU C., CHEN B., COHEN-OR D., ZHANG H.: Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG) 38*, 2 (2019), 1–16. 2

[LYAM13] LIU H., YANG Y.-L., ALHALAWANI S., MITRA N. J.: Constraint-aware interior layout exploration for pre-cast concrete-based buildings. *The Visual Computer 29*, 6-8 (2013), 663–673. 2

[MSK10] MERRELL P., SCHKUFZA E., KOLTUN V.: Computer-generated residential building layouts. *ACM Transactions on Graphics (TOG) 29*, 6 (2010). 2

[MSL*11] MERRELL P., SCHKUFZA E., LI Z., AGRAWALA M., KOLTUN V.: Interactive furniture layout using interior design guidelines. *ACM transactions on graphics (TOG) 30*, 4 (2011), 1–10. 2

[NCC*20] NAUATA N., CHANG K.-H., CHENG C.-Y., MORI G., FURUKAWA Y.: House-GAN: Relational generative adversarial networks for graph-constrained house layout generation. In *European Conference on Computer Vision* (2020), pp. 162–177. 2, 3

[PGK*20] PARA W., GUERRERO P., KELLY T., GUIBAS L., WONKA P.: Generative layout modeling using constraint graphs, 2020. arXiv:2011.13417. 3, 6

[QZH*18] QI S., ZHU Y., HUANG S., JIANG C., ZHU S.-C.: Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 5899–5908. 2

[RWL19] RITCHIE D., WANG K., LIN Y.-A.: Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 6182–6190. 2

[SKM*19] SAVVA M., KADIAN A., MAKSYMETS O., ZHAO Y., WIJMANS E., JAIN B., STRAUB J., LIU J., KOLTUN V., MALIK J., PARIKH D., BATRA D.: Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2019). 1, 10

[SSK*17] SUNG M., SU H., KIM V. G., CHAUDHURI S., GUIBAS L.: ComplementMe: Weakly-supervised component suggestions for 3D modeling. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 226. 2, 5, 6

[SWD*17] SCHULMAN J., WOLSKI F., DHARIWAL P., RADFORD A., KLIMOV O.: Proximal policy optimization algorithms, 2017. arXiv:1707.06347. 10

[SWM*19] STRAUB J., WHELAN T., MA L., CHEN Y., WIJMANS E., GREEN S., ENGEL J. J., MUR-ARTAL R., REN C., VERMA S., ET AL.: The Replica dataset: A digital replica of indoor spaces, 2019. arXiv:1906.05797. 2

[WFLW18] WU W., FAN L., LIU L., WONKA P.: MIQP-based layout design for building interiors. *Computer Graphics Forum 37*, 2 (2018), 511–521. 2, 6

[WFT*19] WU W., FU X.-M., TANG R., WANG Y., QI Y.-H., LIU L.: Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 234. 2, 3, 4, 8

[WKM*20] WIJMANS E., KADIAN A., MORCOS A., LEE S., ESSA I., PARIKH D., SAVVA M., BATRA D.: DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. In *International Conference on Learning Representations (ICLR)* (2020). 10, 11

[WLW*19] WANG K., LIN Y.-A., WEISSMANN B., SAVVA M., CHANG A. X., RITCHIE D.: PlanIt: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG) 38*, 4 (2019), 132. 2

[WSCR18] WANG K., SAVVA M., CHANG A. X., RITCHIE D.: Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–14. 2

[XZH*18] XIA F., ZAMIR A. R., HE Z., SAX A., MALIK J., SAVARESE S.: Gibson Env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 9068–9079. 1, 2, 10, 11

[YMB*18] YAN C., MISRA D. K., BENNETT A., WALSMAN A., BISK Y., ARTZI Y.: CHALET: Cornell house agent learning environment, 2018. arXiv:1801.07357. 2

[YYT*11] YU L. F., YEUNG S. K., TANG C. K., TERZOPOULOS D., CHAN T. F., OSHER S. J.: Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH 30*, 4 (2011). 2

[ZF18] ZHANG Y., FUNKHOUSER T.: Deep depth completion of a single RGB-D image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 175–185. 1

[ZSS*18] ZAMIR A. R., SAX A., SHEN W., GUIBAS L. J., MALIK J., SAVARESE S.: Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 3712–3722. 1

[ZSY*17] ZHANG Y., SONG S., YUMER E., SAVVA M., LEE J.-Y., JIN H., FUNKHOUSER T.: Physically-based rendering for indoor scene understanding using convolutional neural networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). 2

[ZWK19] ZHOU Y., WHILE Z., KALOGERAKIS E.: SceneGraphNet: Neural message passing for 3D indoor scene augmentation. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 7384–7392. 2

[ZYM*20] ZHANG Z., YANG Z., MA C., LUO L., HUTH A., VOUGA E., HUANG Q.: Deep generative modeling for scene synthesis via hybrid representations. *ACM Transactions on Graphics 39*, 2 (2020). 2