Interactive and Scalable Layout Synthesis with Design Templates

Hameedullah Farooki*, Esra Ataer-Cansizoglu†, Jae-Woo Choi†, and Tomer Weiss*‡
*New Jersey Institute of Technology, Newark, NJ
†Wayfair, Boston, MA
‡tweiss@njit.edu

***Abstract*—The design of virtual and real spaces is a complex task, as is evidenced by the large number of professionals offering their services. Researchers proposed multiple computational methods that aim to alleviate such complexity. Unfortunately, most methods for layout synthesis are not directly applicable to non-professional consumers, because of usability challenges in terms of computation, user input, and scalability. Hence, we propose a novel layout synthesis system based on design templates. Design templates define geometrical rules for creating rooms, according to the room type and furniture function. With such templates, our system allows a customizable user experience, and is computationally fast while remaining scalable. We demonstrate our method with several example layouts, focusing on both small and large rooms.**

***Index Terms*—Interactive layout synthesis, 3D scene modeling, Automatic content creation**

I. Introduction

Arranging furniture is an everyday problem that is nonetheless surprisingly complex. When choosing which furniture to add to a room, we need to examine how it fits with the existing furniture arrangement. Furthermore, furniture arrangements can divide a room into different functional zones, that serve various purposes, such as seating, eating, and socializing. Each such zone has distinct design demands, such as space, type, size, and style of compatible furniture; adjacency to doors and windows; and fit with visual and human flow, among other factors.

Consequently, it is often difficult for non-experts to solve such layout problems, which is confirmed by the existence of a wide variety of professional design services. However, hiring such professionals is not always financially possible, which is driving research into computational design methods. For example, in recent years, there have been several developments in computational interior design, both in academic research [8] and in industry [1].

Our goal is to empower non-professional users when designing their space. To that end, we need to understand user requirements for a layout design system. In a typical workflow, users select furniture and add it to a virtual space incrementally. We focus on improving the user experience when interactively selecting and placing furniture in their space. In order to fulfill such goals, we require that a layout synthesis and design system should adhere to the following conditions (**RQ**):

- **RQ1**: Interactive and responsive to web users.
- **RQ2**: Simple to use.
- **RQ3**: Lightweight computationally.

We will now examine the landscape of existing relevant literature and how they address such requirements. Most early work employs hard-coded rules [4], while recent work is data-driven, utilizing deep learning [9], [10]. Other researchers proposed optimization-based approaches [2], [5], [7]. However, most are not geared towards interactive, seamless layout synthesis [3], and therefore do not satisfy **RQ1**.

Since there are multiple types of rooms, methods for automatically creating layouts require some form of user guidance, either in form of data or manual input. Early work by Tutenel et al [4] allows users to define hard-coded rules for generating layouts. However, users of their method need to encode such rules in advance, and it does not explicitly support sequential layout synthesis as we do. Weiss and Merrell et al. [3], [5], [6] use geometric constraints for synthesizing furniture arrangements. While their approach is interactive, users still need to select furniture and provide explicit layout synthesis constraints, a process which is laborious and difficult for non-professional users, contrary to **RQ2**. Our proposed approach, detailed in Section II, eliminates such need with predefined

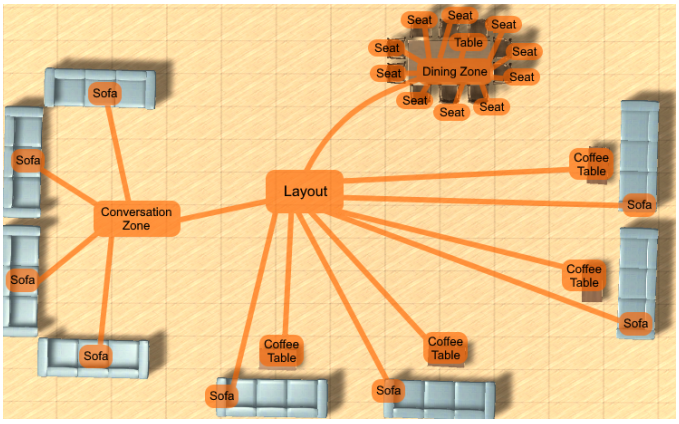


Fig. 1: An example of a layout which contains multiple zones, each of which contain furniture. Such hierarchy defines a tree structure with zone placements as branches and furniture placements as leaves. We utilize such inherit scene structure for providing fast and interactive layout suggestions.

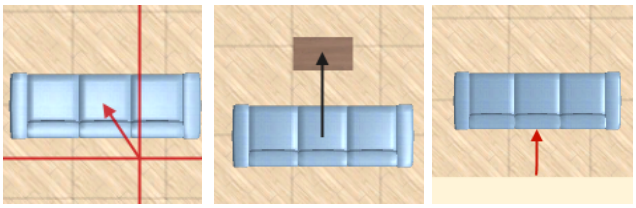
blueprints, which we call *design templates*.

Recent deep-learning work in layout synthesis [9], [10] heavily relies on datasets, some of which are not publicly accessible, and therefore are difficult to reproduce. Furthermore, such data-driven methods may only synthesize furniture pieces and scene variations that are related to the input datasets. Additionally, such methods typically rely on GPU hardware for utilizing deep learning or parallel computation [3], [9], [10]. Per **RQ3**, our approach needs to be lightweight, and should be able to run on a user's web browser.

Since previous work does not meet **RQ1**—**RQ3**, we propose a novel system for interactive room design, focusing on furniture placement. Specifically, we focus on assisting users in placing and selecting furniture.

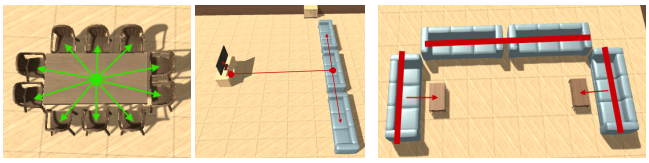
II. Method

We modularize a layout synthesis problem into several components, which include *furniture items* to be placed, and *functional zones*, which are areas within a layout that are associated with specific furniture. For example, a conversation area is associated with furniture such as accent chairs, loveseats, and sofas. Accepting these as input, our system outputs *furniture placement* suggestions (Fig. 1). Below we describe our layout synthesis system in detail.



(a) (b) (c)

Fig. 2: (a) Origin placement. The sofa's center is placed at the center of the layout plus the placement rule's offset. (b) Side placement. The sofa is the anchor and the coffee table's center is placed at the specified offset from the sofa's center. (c) Wall placement. The sofa is automatically adjusted so that its edge would be touching the wall, and then the sofa is further moved by the placement rule's offset.



(a) (b) (c)

Fig. 3: (a) a Table zone, which employs origin and side placement rules, with an input of a single table and chair. (b) Base zone, created with origin and side placement rules. The table forms the basis for the layout. The TV is placed on top of the table. (c) Conversation zone, which encompasses an abstract rectangle-shaped area, in which we visualize three rectangle edges. The input furniture items (four sofas and two coffee tables) were distributed amongst the edges, resulting in this layout.

A. Functional Zones and Design Templates

We define functional zones as spatial areas containing groups of furniture items (Fig. 1). Zones are categorized into the following types: Conversation, Dining, Bedroom, Office, and Television. Additionally, we include a "default" zone, which can be used for furniture placement outside of individual zones. We also include an "any" zone, which can be used for placement rules that apply regardless of the zone. Creating more custom zones is easily accomplished with our system (Figure 5b). Each zone is associated with design templates and furniture. Given a design template, and a list of furniture, the output is a sequence of possible layouts. We define several *design templates*:

Base: This template (Fig. 3b) supports zone types such as a Bedroom, Office, and Television (Algorithm 1). In brief, we first generate a list of potential furniture items that have an associated origin placement rule applicable to the zone, called *origin items*. Then, for each origin item, we generate a list of side items by filtering the input furniture items, excluding the origin item, and match the resulting furniture with open side item placements. Finally, we add all possible layouts that could be generated using such origin items and side items.

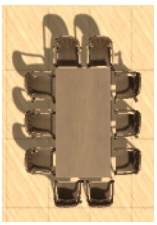


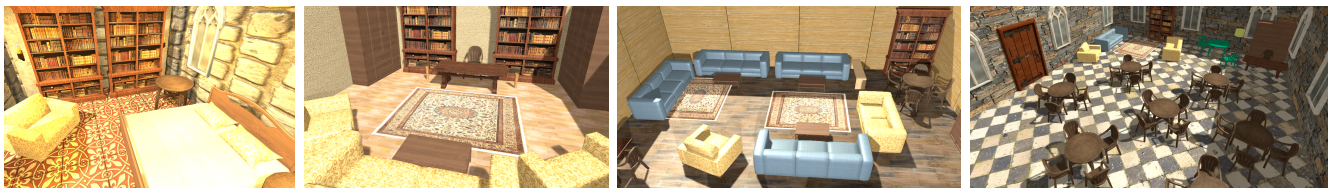
Table: Dining and similar areas can be modeled with such template (Algorithm 2). While mostly similar to the previous template, here we do not take into account take into account the number of side items. Hence, if the input is one table and one chair, the output would be the number of chairs the table has space for (Fig. 3a).

Conversation: Social interaction zones, such as a lounge area or a living room (Fig. 3c), can be designed with such template. Our system generates placement suggestions by adding furniture to the edges of an abstract rectangular shape that encompasses the conversation area, preferring smaller edges (Algorithm 3).

B. Placements

We define two placement types: furniture and zone placements (Fig. 6). Zone placements are spatial regions within a layout that serve a certain function, such as socializing or working. Furniture placements are candidate positions of such furniture, which may be numerous depending on the available space and *placement rules*. Furniture may either be placed independently or placed within zones (Fig. 1).

As an abstraction of the individual rules for placing furniture items in a layout, we introduce *placement rules*. Placement rules take an existing layout, along with an additional furniture, as input, and output a collection of possible layouts with the input furniture item added to the original layout. Thus, there are only furniture-based placement rules at present, but zone-based placement rules that automatically populate rooms



(a) Bedroom (b) Office (c) Living Room (d) Restaurant

Fig. 4: Our system allows synthesis of various room types, ranging from a bedroom, office, living room to a restaurant.

with zones may be added in the future. We define several placement rules. Each rule is associated with a placement algorithm that validates whether there is sufficient space in the scene for the furniture item and determines where to place it. The current placement rules are:

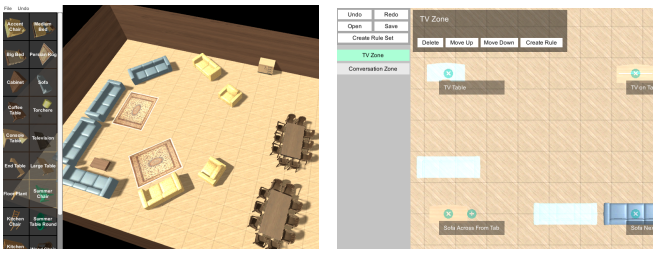
Origin: We consider a furniture item in a layout relative to its center, adding an offset position and rotation (Fig. 2a). The item is then added at the same offset if there is sufficient space.

Side: Given a list of furniture items and a list of side positions, our method places furniture next to an existing anchor item (Fig. 2b). We iterate through each potential furniture placement. In case of an anchor placement, we generate placements next to such anchors, providing they do not collide with existing furniture. This rule is powerful since it supports a wide range of use cases by allowing to align suggested positions relative to different furniture items.

Wall: Several furniture items need to be placed against a wall (Fig. 2c). We attempt to generate placements for each wall of the room, as close to the corner as possible, ignoring walls with no space. Additionally, we include an offset to control how far the placement is from the wall, and its orientation relative to it.

C. Placement Rule Editor

To facilitate placement rule configuration, we developed an interactive rule editor (Fig. 5b). The placement rule editor allows users to group placement rules into placement rule sets. Placement rule sets are a collection of placement rules that users may edit in grid view. With placement rule sets, users can easily create rules for entire furniture layouts.



(a) Layout Editor (b) Placement rule editor

Fig. 5: (a) Left: A list of furniture items that can be added to the scene. Right: Interactive 3D view of the furniture layout. (b) Left: Dashboard for basic editing functions. Beneath it, there is a list of placement rule sets. Selecting one opens the view that takes the majority of the screen on the right. Right: the placement rules visualization.

D. Layout Design Visualization

To visualize the output of our approach, we created a layout editor (Fig. 5). The layout editor allows users to add furniture items and zones individually. When placing a furniture item or a zone, our system allows the user to cycle through multiple placement options to choose an optimal layout. Additionally, we allow users to manually drag around and rotate existing placements to their preferences.

III. Results

We tested our approach on several example scenarios. Experiments were conducted on an AMD Ryzen 5 2600 CPU, using C#. We utilized the Unity Engine for rendering and user interaction. Below we describe several experiments in detail (Fig. 4). All resulting layouts were created interactively.

Bedroom: We include a bed with tables next to it, a chair for reading, several shelves, and a cabinet. We utilized a bedroom zone, along with two placement rules: (i) an origin placement rule to place the bed at the center of the zone, and (ii) a side placement rule to place the tables at the sides of the bed.

Restaurant: The restaurant scene has a variety of features, including a serving area, a conversation zone, small tables, and large tables. Conversation and dining zones were used in this scene. The conversation zone used an origin placement rule for the sofa and a side placement rule for the accent chair. The various seating arrangements used origin placement rules for the tables and side placement rules for the chairs.

Living Room: We include several conversation zones, a TV zone, and accessory items. The TV zone used an origin placement rule for the console table, along with a side placement rule for the TV itself. The conversation zones used origin placement rules for the sofas and loveseats, as well as side placement rules for sofas, loveseats, and accent chairs to be placed next to sofas. The accent chair side placement rules includes a rotational offset.

Office: We include a desk, chair, bookshelves, cabinets, guest seating, and a few accessories. The layout includes an office zone an office zone, which has an origin placement rule for the desk and a side placement rule for the chair.

IV. Discussion and Future Work

We demonstrated that our system is effective in synthesizing layout arrangements using design templates. While current

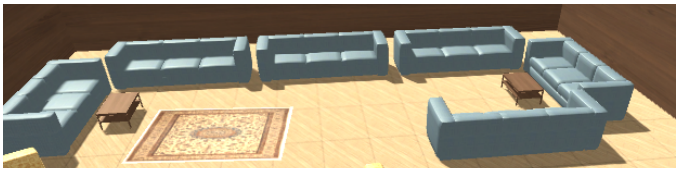


Fig. 6: A large conversation seating layout.

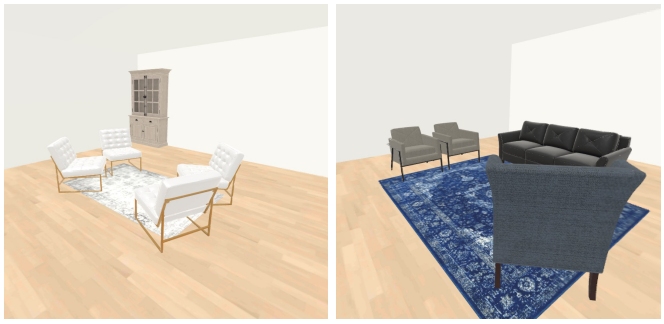


Fig. 7: Rooms captured in Virtual Reality.

results focus on interior spaces, our approach is extendible to other domains and spaces, by editing and creating new design templates with our editor (Section II-C). Unlike previous work, our method allows interactive layout design at scale, generating multiple layout variations. Moreover, since our approach is interactive, users can quickly modify layout suggestions.

Our system can also be utilized to bootstrap other layout synthesis work. For example, to encode an initialization for a graph of geometric constraints [5], [6], or graph relationships for a deep neural network layout synthesis approach [9], [10].

Finally, a version of our system is publicly accessible via *Room Planner 3D* (RP3D)[1], and integrated with Wayfair's Virtual Reality room design experience (Figure 7). In the future, we plan to add style-aware suggestions to our system, in which suggested furniture arrangements are harmonious in terms of materials, color, and 3D style. We believe that extending our framework in such direction would be beneficial to many users.

Algorithm 1 Base Template

```
Require Furniture items
procedure ArrangeFurniture(items)
    originItems ← getOriginItems(items)
    for all originItem ∈ originItems do
        sideItems ← getSideItems(items, originItem)
        layouts ← getPermutations(originItem, sideItems)
        for all layout ∈ layouts do
            yield layout
        end for
    end for
end procedure
```

Algorithm 2 Table Area Template

```
Require Furniture items
procedure ArrangeFurniture(items)
    originItems ← getOriginItems(items)
    for all originItem ∈ originItems do
        layout ← getLayout(originItem)
        sideItems ← getUniqueSideItems(items, originItems)
        while #sideItems > 0 do
            for all sideItem ∈ sideItems do
                clonedLayout ← layout
                if not addToLayout(clonedLayout, sideItem) then
                    remove sideItem
                    continue
                end if
                layout ← clonedLayout
            end for
        end while
        if #layout == 0 then
            continue
        end if
        for all angle ∈ {0, 90, 180, 270} do
            yield layout rotated by angle
        end for
    end for
end procedure
```

Algorithm 3 Conversation Template

```
Require Furniture items
procedure ArrangeFurniture(items)
    n ← max(#items, 4)
    for all i ∈ {1,...,n} do
        shapes ← getConversationAreaShapes(i)
        for all activeEdges ∈ shapes do
            if not assignItems(items, activeEdges) then
                continue
            end if
            yield layout generated by placing furniture in activeEdges
        end for
    end for
end procedure
```

References

[1] E. Ataer-Cansizoglu, H. Liu, T. Weiss, A. Mitra, D. Dholakia, J.-W. Choi, and D. Wulin. Room style estimation for style-aware recommendation. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 267–2673. IEEE, 2019.

[2] P. Kán and H. Kaufmann. Automatic furniture arrangement using greedy cost minimization. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 491–498. IEEE, 2018.

[3] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun. Interactive furniture layout using interior design guidelines. In *ACM Trans. Graphics*, volume 30, page 87, 2011.

[4] T. Tutenel, R. Bidarra, R. M. Smelik, and K. J. De Kraker. Rule-based layout solving and its application to procedural interior generation. In *CASA workshop on 3D advanced media in gaming and simulation*, 2009.

[5] T. Weiss, A. Litteneker, N. Duncan, M. Nakada, C. Jiang, L.-F. Yu, and D. Terzopoulos. Fast and scalable position-based layout synthesis. *IEEE Trans. on Visualization and Computer Graphics*, 2018.

[6] T. Weiss, M. Nakada, and D. Terzopoulos. Automated layout synthesis and visualization from images of interior or exterior spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 41–47, 2017.

[7] L.-F. Yu, S. K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. Osher. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graphics*, 30(4):86, 2011.

[8] S.-H. Zhang, S.-K. Zhang, Y. Liang, and P. Hall. A survey of 3d indoor scene synthesis. *Journal of Computer Science and Technology*, 34(3):594–608, 2019.

[9] Z. Zhang, Z. Yang, C. Ma, L. Luo, A. Huth, E. Vouga, and Q. Huang. Deep generative modeling for scene synthesis via hybrid representations. *arXiv preprint arXiv:1808.02084*, 2018.

[10] Y. Zhou, Z. While, and E. Kalogerakis. Scenegraphnet: Neural message passing for 3d indoor scene augmentation. In *International Conference on Computer Vision (ICCV)*, pages 7384–7392, 2019.

[1] Accessible at http://www.wayfair.com/roomplanner3d, currently supporting the conversation area, as of October 2020.