Free-form Floor Plan Design using Differentiable Voronoi Diagram

Xuanyu Wu Kenji Tojo Nobuyuki Umetani

The University of Tokyo

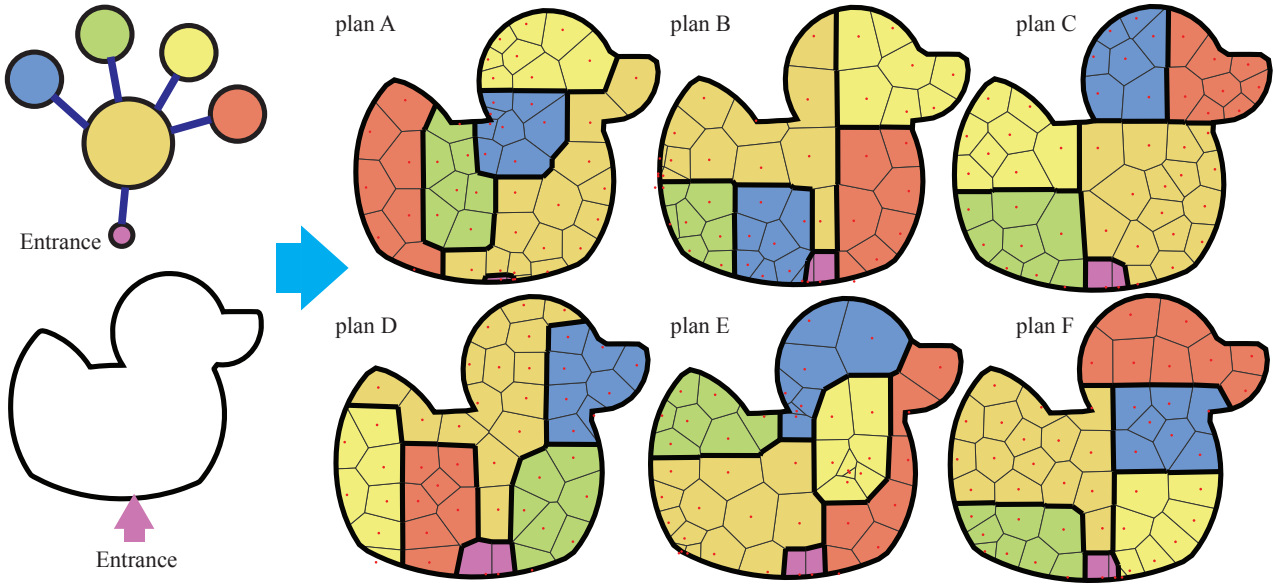

Figure 1: *Our algorithm the input of rooms' connectivity, areas, and boundary shapes (left) and generates floor plans from (right).*

Abstract

Designing floor plans is difficult because various constraints must be satisfied by the layouts of the internal walls. This paper presents a novel shape representation and optimization method for designing floor plans based on the Voronoi diagrams. Our Voronoi diagram implicitly specifies the shape of the room using the distance from the Voronoi sites, thus facilitating the topological changes in the wall layout by moving these sites. Since the differentiation of the explicit wall representation is readily available, our method can incorporate various constraints, such as room areas and room connectivity, into the optimization. We demonstrate that our method can generate various floor plans while allowing users to interactively change the constraints.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling;*** • ***Human-centered computing*** → ***Interaction design;***

1. Introduction

Planning floor layouts is essential in architectural design. In practice, architects prepare various layout plans, such as site plans, section plans, floor plans, and elevation plans, prior to real-world construction. In addition to serving as blueprints for construction, these two-dimensional layouts allow architects to visually examine and convey their unique design concepts. However, manual creation of the floor layout often requires significant time and skills from the architect due to the diversity in the shape of the available space and the constraints that individual rooms must satisfy, such as their sizes and connectivity. These constraints discourage architects from casually exploring their innovative design ideas.

To address this difficulty, researchers have proposed various computational algorithms for generating floor layouts. Previous methods typically represent rooms as rectangular polygons that are optimized to meet the room requirements [MSK10]. However,

DOI: 10.2312/pg.20241296

the explicit polygon representations either require intensive *mesh surgery* procedures to introduce changes in room connectivity or unsuitable for causing large movement due to room collisions, which limits the flexibility of the algorithm to explore diverse room configurations. Recent data-driven methods [HHT*20, NCC*20, NHC*21, SWL*22, SHF23] have employed deep learning to create floor plans. However, these learning-based methods often fail to support unconventional room shapes and new constraints that are not assumed in a specific training dataset. Unfortunately, the existing methods are not suitable for "free design of the ground plans," as advocated by a notable modern architect *Le Corbusier* [Cor24].

In this paper, we present a differentiable room representation for the flexible generation of floor plans, while considering various constraints on the shape and connectivity of the room. Our representation uses the Voronoi diagram to partition the floor into multiple rooms. Specifically, one room is defined using multiple Voronoi cells that are *implicitly* defined based on the distance to the Voronoi sites and support topology changes, which facilitates flexible exploration of room connectivity during optimization. Furthermore, the Voronoi diagram also provides an *explicit* representation of the walls (i.e., partition boundaries), allowing us to handle different geometric constraints, such as wall simplicity and room areas, which are difficult to formulate directly from the Voronoi sites. The explicit wall representations readily provide the gradients with respect to the Voronoi sites, which allows us to optimize the room shapes without using complicated mesh-surgery procedures.

Although the Voronoi diagram represents rooms with changing topologies, the unconstrained Voronoi diagram is excessively flexible to produce plausible floor plans. For instance, a room may be separated into multiple islands or may have unnecessarily complicated wall shapes. To overcome these issues, we develop tailored loss functions that encourage rooms to have connected shapes with simple wall shapes. These additional loss functions are combined with those that promote the user-specified requirements and minimized to generate realistic and useful floor plans.

Our method is efficient enough to provide an interactive user interface for exploring possible floor plans. Our interface allows the user to specify the desired room constraints interactively and instantly preview the optimization results. In addition, by performing the optimization multiple times with different random initializations, the user can efficiently obtain various design options to explore the space of possible floor plans for a given floor outline and room constraints.

In summary, our contributions are:

- A differentiable representation of floor plan using the Voronoi diagram.
- Identification of loss function to make the partition look like a floor plan.
- An interactive user interface to allow the user to easily explore possible floor plans.

2. Related Work

This section reviews related studies in the computer graphics and vision literature to clarify the focus of our work.

Automatic layout generation Previous methods have generated the layouts of various geometric elements in different application domains, such as virtual scene creation [CKP*24, FRS*12], urban planning [FYY*16, YWVW13], building interior estimation [JFRM17], furniture arrangement [MSL*11, YYT*11], and game design [MVLS14, SLY22, HMVDVI13]. Each domain has a unique set of constraints that must be considered in the layout, such as the adjacency and order of the elements. In this paper, we focus on the generation of architectural floor plans, where the rooms must satisfy varying connectivity and size constraints specified by the architect, as well as other geometric requirements, such as the simplicity of a wall.

Floor plan generation The methods most closely related to ours generated architectural floor plans. One line of this work aims to generate the layout and connectivity of rectangular or cubic rooms using different computational approaches, such as evolutionary computation [RG13a, RG13b, GL17], mixed integer quadratic programming [WFLW18], stochastic optimization [MSK10], and physically-inspired mass-spring models [ESMR20, ZYX23]. In contrast to these approaches, our method can generate free-form room shapes beyond rectangular ones and allows the user to specify a more arbitrary curved outline of the interior space.

Several existing approaches enable more flexible element shapes by using irregular [SLY22] tile shapes to fill the space. To handle more general boundary shapes, other methods have flexibly deformed the templates to better fit the rooms to the interior space [PYW14] or to split a given urban site map into meaningful sections [YWVW13]. However, the flexibility of these template-based methods is still significantly restricted by the shape of the templates and their initial configurations, which limits the output diversity. By contrast, we allow the rooms and their constituent Voronoi cells to move freely in space to effectively produce diverse configurations and room shapes from an unstructured initialization.

Another common approach for floor-plan generation leverages data from existing layouts using neural networks [WFT*19, HHW22, SWL*22]. To produce high-quality outputs, recent learning-based methods have used various advanced techniques, such as generative adversarial networks [WZC*23, NCC*20, NHC*21], transformers [SZZW23], and diffusion models [SHF23]. These methods also consider the room constraints. For example, Graph2Plan [HHT*20] uses a room-connectivity graph and a boundary shape to output a layout that conforms to the input specification. However, these learning-based methods require a large dataset for training, and their output heavily reflects the bias in the training dataset. This limits the shape of the floor boundary specifications and generated rooms, which hinders the generalization of these methods beyond simple room shapes with straight and axis-aligned walls. In contrast, our method can represent various free-form wall and boundary shapes using the differentiable Voronoi diagram, whose shapes can be fluidly optimized through gradient-based optimization.

Voronoi diagram in graphics and vision The Voronoi diagram and its weighted version have been used in a wide array of applications, including fluid simulation [dGWH*15], foam simulation [BDWR12], function approximation [WPLN*20], 3D surface

processing [RAM*21, RGA*21], structural design [ZFD*17], and topology optimization [FXL*23], to name a few. Several recent studies [FXL*23, NLCT24] have proposed methods for differentiating the Voronoi diagrams to address inverse problems involving cellular mechanics. Feng et al. [FXL*23] presented a method for smoothing the boundaries of the Voronoi cells to achieve differentiable transition of data across the cells. However, their method assumes a background grid to compute the differentiation and does not directly provide an explicit representation of the boundary, which our method requires to enforce constraints. Numerov et al. [NLCT24] computed the Voronoi vertices explicitly and differentiated the cell boundary with respect to the Voronoi site positions. Our method also uses an explicit boundary representation and its differentiation. In contrast to this method, we represent a single room using multiple Voronoi cells to generate significantly more complex shapes than when using individual cells. Furthermore, we propose tailored loss functions for the Voronoi vertices to produce partitions that are plausible as architectural floor plans, which we demonstrate in our ablation study.

Hybrid explicit-implicit representations In computer graphics and vision, a hybrid method using both explicit topology and implicit or point-based representations has been applied in several other applications, ranging from surface tension simulation [XRW*22, BBB10] to surface reconstruction [MCR22]. These hybrid representations allow the algorithms to handle constraints on the explicit boundary geometry better while enabling a flexible topology change. In this work, we present a hybrid representation based on the Voronoi diagram for optimizing the partitioning of a floor into multiple rooms. The implicit aspect allows for flexible transition of the room layout, whereas the explicit wall representation facilitates formulation of useful losses in the rooms, such as the area and the alignment of the walls inspired by the 3D surface stylization method [LJ19].

3. Method

3.1. Floor plan representation

The user provides a 2D region Ω, which is the outline of the available space viewed from above. Given the region, a connected and bounded shape in the 2D space $\mathbb{R}^2$, we intend to compute a partition of Ω into multiple rooms that satisfy the specified room constraints.

Voronoi diagram To achieve topological flexibility in the connectivity between rooms, we use the Voronoi diagram to represent the partition. A Voronoi diagram is generated from a set of points, called *sites* $\mathcal{S} = \{\mathbf{s}_1 \dots \mathbf{s}_N\}$. Each site $\mathbf{s}_i$ is associated with a region $\mathcal{C}_i$ called *cell*, which comprises the points whose nearest site is the cell's site, i.e.,

$$\mathcal{C}_i = \{\mathbf{x} \in \Omega \mid ||\mathbf{x} - \mathbf{s}_i|| \leq ||\mathbf{x} - \mathbf{s}_j|| \text{ for all } i \neq j\}. \quad (1)$$

Voronoi vertex The cells take polygonal shapes, whose corner vertices are called Voronoi vertices. We denote the set of all the Voronoi vertices in the diagram as $\mathcal{V} = \{\mathbf{v}_1 \dots \mathbf{v}_M\}$. The Voronoi vertex is either inside the region Ω or on the boundary of Ω.

The Voronoi diagram has aspects of both *implicit* and *explicit* representations. Whereas a Voronoi cell is defined *implicitly* by its distance from the sites, the boundary of the cells can be defined *explicitly* by connecting the Voronoi vertices. The implicit aspect contributes to topological flexibility – the topology of the cells can change through the continuous motion of the site positions. This eliminates the necessity of mesh surgery – a painstaking procedure for editing the topology of explicit shape representations. However, the explicit representation of the boundary of the cells allows us to define constraints that are directly specified in the boundary geometry (e.g., the length of the boundary).

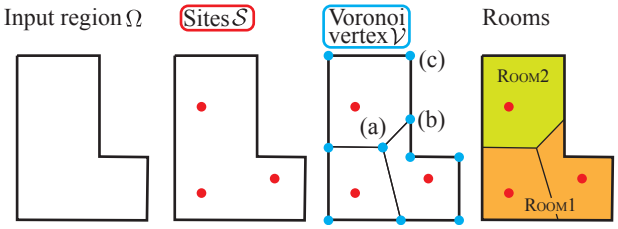

Figure 2: *The Voronoi diagram shows the partition of the input region Ω by a set of points called sites $\mathcal{S}$. The corners of the Voronoi cells are Voronoi vertices $\mathcal{V}$. The Voronoi vertex is either a vertex of Ω (c), on the boundary edge (b), or inside Ω (a). The combination of multiple cells represents a room.*

To compute the deformation of the room boundary from the site motion, classifying the Voronoi vertices into the following three types is convenient (see Figure 2 center-right): (a) the intersection of the three bisectors, (b) the intersection of one bisector and the boundary Ω, and (c) the vertex forming the corner of Ω. Unlike (a) and (b), the position of (c) does not depend on the Voronoi sites and is constant during our optimization. When we compute the Voronoi diagram, we store the symbolic information on each Voronoi vertex, which indicates the type of the Voronoi vertex, the indices of the bisector sites, and the edge or vertex index of Ω. This symbolic information is used in the subsequent differentiation of the Voronoi diagram and the graph search for adjacent cells.

Room representation A single cell in the Voronoi diagram has a simple polygonal shape and cannot represent a complex room shape. Specifically, the shape of each Voronoi cell is limited to a convex polygon intersecting region Ω. In other words, a typical floor plan is complex and thus its room cannot be represented by a single Voronoi cell. To increase the representation capability, we combine multiple cells to represent one room (see Figure 2 rightmost). This is concretely achieved by associating each site with one of the rooms, which is determined at the initialization and then remains constant.

Initialization At the time of initialization, we randomly generate a user-specified number of sites in the input region Ω. To ensure that no two sites are excessively close to each other, we use the Poisson disk sampling with the dart-throwing algorithm. Rooms are randomly assigned to each site. The number of sites associated with each room is set such that it is approximately proportional to the user-specified target room area. This number adjustment is intended to make the density of the sites uniform and to ensure each

room to have roughly equal degree of freedom (i.e., the Voronoi sites) per area. Note that the areas of individual sites are not constrained during the optimization, and the room area of the output plan is governed by the area loss explained later.

3.2. Floor plan optimization

We optimize the shape and connectivity of the rooms by moving the positions of the sites. Each site is associated with a fixed room during optimization. Specifically, the sites are moved to minimize the scalar loss function $\mathcal{L}$ that formulates the desirable properties of the rooms. Since the optimization variables are continuous values of 2D coordinates, they can be easily incorporated into the optimization framework using the gradients of $\mathcal{L}$. During the optimization, we do not strictly constrain the site to be inside the region Ω. Instead, by allowing the sites to move freely outside Ω, we can handle a concave input region Ω.

Differentible Voronoi diagram Several components of our loss function are defined in terms of the Voronoi vertices $\mathcal{V}(\mathcal{S})$, which are determined by the site positions. We differentiate these components with respect to the site positions $\mathcal{S}$ using the chain rule by computing the gradients $\partial\mathcal{V}/\partial\mathcal{S}$. When computing $\partial\mathcal{V}/\partial\mathcal{S}$, we assume that no degeneracy is present and that every Voronoi vertex belongs to at most three cells. In this case, the positions of the Voronoi vertices change linearly with the infinitesimal movement of the sites. Note that the number of the Voronoi vertices may change when the sites move. However, this change does not affect the loss or gradient computation, which is performed independently for each iteration. For the detailed computation of the derivative of the Voronoi vertex with respect to the site position, please refer to [NLCT24].

Loss function We update the coordinates of the sites $\mathcal{S}$ to minimize the loss $\mathcal{L}$. Specifically, our loss $\mathcal{L}$ is defined as the weighted sum of the different losses as

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \mathcal{L}(\mathcal{S}, \mathcal{V}(\mathcal{S})) \tag{2}$$

$$\mathcal{L} = \mathcal{L}_{\text{wall}} + \mathcal{L}_{\text{area}} + \mathcal{L}_{\text{fix}} + \mathcal{L}_{\text{topo}} + \mathcal{L}_{\text{Lloyd}}. \tag{3}$$

The $\mathcal{L}_{\text{wall}}, \mathcal{L}_{\text{area}}, \mathcal{L}_{\text{fix}}, \mathcal{L}_{\text{topo}}, \mathcal{L}_{\text{Lloyd}}$ are the loss functions guiding the movement of the sites $\mathcal{S}$. We use gradient-based optimization to minimize the loss function $\mathcal{L}$ in (2). Each time the sites are moved by the optimization, the Voronoi diagram is updated.

We use $\mathcal{L}_{\text{fix}}$ to fix some of the sites. For instance, the entrance to the house in Figure 1 is specified by the user by fixing the sites related to the entrance there. The $\mathcal{L}_{\text{fix}}$ is defined as the squared sum of the difference between the site and the specified position weighted by $w_{\text{fix}} \in \mathbb{R}^+$. In the following, we explain each term of the loss function, and Section 4.2 demonstrates the effect of each.

Wall loss As the unconstrained Voronoi diagram typically produce undesirable fluctuations in the wall orientations, we design a tailored loss function to regularize the wall complexity. Inspired by the Cubic Stylization [LJ19], we regularize the $L1$ norm of the wall length as

$$\mathcal{L}_{\text{wall}} = w_{\text{wall}} \sum_{(v_i, v_j)\in\mathcal{E}} ||\mathbf{v}_i - \mathbf{v}_j||_{L1}, \tag{4}$$

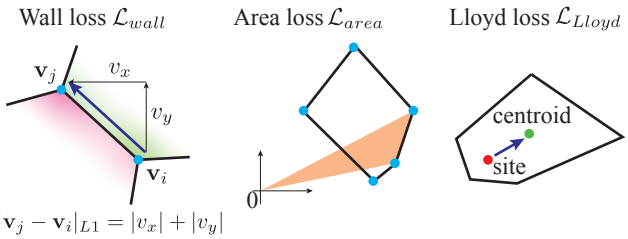

Figure 3: *Computation of loss functions. (Left) to make the walls simple and aligned, we use the L1 norm for the wall. (Center) the area is computed for each cell by adding up the signed area of the triangle made by connecting the edge and the origin of the coordinate. (Right) the density of the site is regulated by moving the site toward the centroid of the cell.*

where $\mathcal{E}$ denotes the set of edges of the Voronoi cells between two adjacent rooms and the $\mathbf{v}_i$ and $\mathbf{v}_j$ denote the Voronoi vertices belonging to the edge (see Figure 3-left). Note that, whereas the original Cubic Stylization [LJ19] uses the $L1$ norm of the surface normal to align the face orientations to cubic faces, we compute the $L1$ norm of the tangent edge vector, which is equivalent to the $L1$ norm of the 2D normal direction. Minimizing this loss has the following two effects: simplifying the wall between two adjacent rooms by penalizing its length, and aligning the boundary to the coordinate axis by making the coordinates of the tangent vector sparse in 2D.

Area loss The area of each room is specified by the user. We minimize the quadratic difference between the current room areas and the user-specified targets.

$$\mathcal{L}_{\text{area}} = w_{\text{area}} \sum_{r=1}^{\#Room} ||A_r(\mathcal{V}) - \bar{A}_r||^2, \tag{5}$$

where $\bar{A}_r$ denotes the target area for the room r (see Figure 3-center).

Lloyd loss To regulate the site density, we design a loss function inspired by the Lloyd's algorithm [LWL*09].

$$\mathcal{L}_{\text{Lloyd}} = w_{\text{Lloyd}} \sum_{i=1}^{N} ||\mathbf{s}_i - \mathbf{c}_i||^2, \tag{6}$$

where $\mathbf{c}_i$ denotes the centroid of the i-th Voronoi cell (see Figure 3-right).

The site may lie outside the user-specified boundary Ω. If the cell of such a site still has a non-empty intersection with Ω, we use the centroid of that intersection to compute the Lloyd loss for this site. This is useful for attracting these exterior sites inside Ω. However, if the site is far away from the Ω, the site does not have a cell intersection. We set the centroid position c_i for such a site to the position $\mathbf{s}_i$ of the site itself, which makes the gradient of $\mathcal{L}_{\text{Lloyd}}$ zero for the site.

Topology loss We design the topology loss such that each room is a single connected region, and the specified connections between

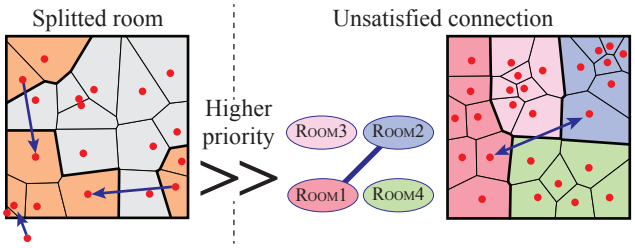

Figure 4: *The positions of target sites for topology loss. (Left) if a room is split into unconnected groups, we select one group and find the nearest sites in that group for all the sites in other groups. (Right) if we find a pair of sites in the rooms with unsatisfactory connections, the target positions of sites are set to each other's position.*

rooms are achieved. We move the site to satisfy the desired topology by setting the goal position $\mathbf{t}_i$ for each site $\mathbf{s}_i$ as

$$\mathcal{L}_{\text{topo}} = w_{\text{topo}} \sum_{i=1}^{N} ||\mathbf{s}_i - \mathbf{t}_i||^2. \tag{7}$$

The goal position t_i can be automatically computed as the nearest site to the site from the same group (see Figure 4-left). For the sites that do not need to move (i.e., the sites whose topology constraints are already satisfied), the goal position is set to the site position to make the gradient zero. Note that, although the goal positions $\mathbf{t}_i$ can depend on the site positions, we treat them as constants in the gradient computation to simplify the effect of the topology loss.

Our optimization begins with the random room assignments to each site. Therefore, in the initial configuration, a room tends to be split into several unconnected groups of sites. For each room, we first group the sites belonging to that room into groups of adjacent sites. If multiple groups are present, that is, a room is split into separated regions, we set the target position of the site $\mathbf{t}_i$ as the nearest site to that group (see Figure 4-left).

If a room is connected in one piece, we examine the connectivity between two different rooms specified by the user. If two rooms that are specified as "connected" in the input are not adjacent, we compute the nearest pair of sites from each room and then set the positions as the target positions for each other.

4. Result

4.1. Implementation detail

Construction of Voronoi diagram For the simplicity of implementation, we construct the Voronoi diagram in a concave region based on the Euclidean distance (i.e., the clipped Voronoi diagram) instead of based on the geodesic distance. Efficient methods to compute the restricted Voronoi diagram exist, such as [YWLL13], which is based on the Delaunay triangulation. For the simplicity of implementation, we computed each Voronoi cell independently. For the construction of the cell $\mathcal{C}_i$, we start from only one site $\mathbf{p}_i$, which produces a cell identical to Ω. Then, we sequentially add the other sites one by one, and for each addition, we cut the cell by the bisector of the new site and $\mathbf{p}_i$ and throw away the parts that are far from $\mathbf{p}$. When the cell is cut by a bisector, we record the symbolic relationship of the newly created Voronoi vertex. This symbolic relationship is used to unify the Voronoi vertices computed independently for each cell and is also used for the differentiation of vertex positions with respect to the site positions.

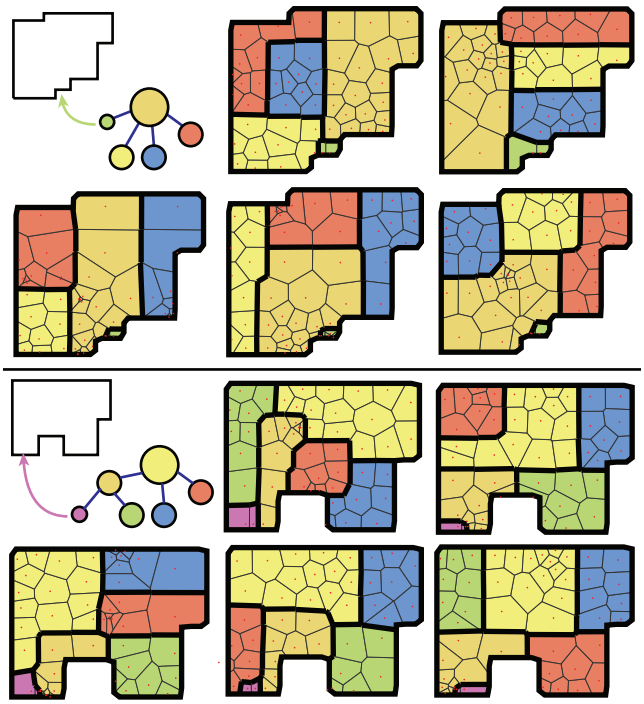

Figure 5: *Examples of the floor plan generation. The top-left corner of each example shows the input region and the bubble diagram (i.e., connectivity between the rooms and the target areas).*

Optimization detail We used the Adam [KB15] optimizer with a learning rate of 1.0e-3. Although we could possibly use the VectorAdam [LSJ22] to further improve the convergence, it is left for future work. The entire program is written using the Rust programming language. We use the framework called candle [Fac23] for automatic differentiation, and the differentiable Voronoi tessellations are implemented by ourselves as a custom differentiable layer. In this study, we set the weights to $w_{\text{wall}} = 0.015, w_{\text{area}} = 1, w_{\text{topo}} = 1$, $w_{\text{fix}} = 100$ and $w_{Lloyd} = 0.1$. Since the physical dimensions of our losses are not equal (e.g., area and length), the effects of these losses are different when the scale is different. The sizes of the models are normalized such that the longest size of the axis-aligned bounding box is one. The code is available at the project website [†].

Optimization As shown in Figures 1 and 5, our optimization framework produces reasonable floor plans for various concave and curved boundary shapes. These results are automatically computed from 300 optimization iterations without interactive editing. This iteration number was sufficient even for the highly concave shape,

† https://github.com/nobuyuki83/floor_plan

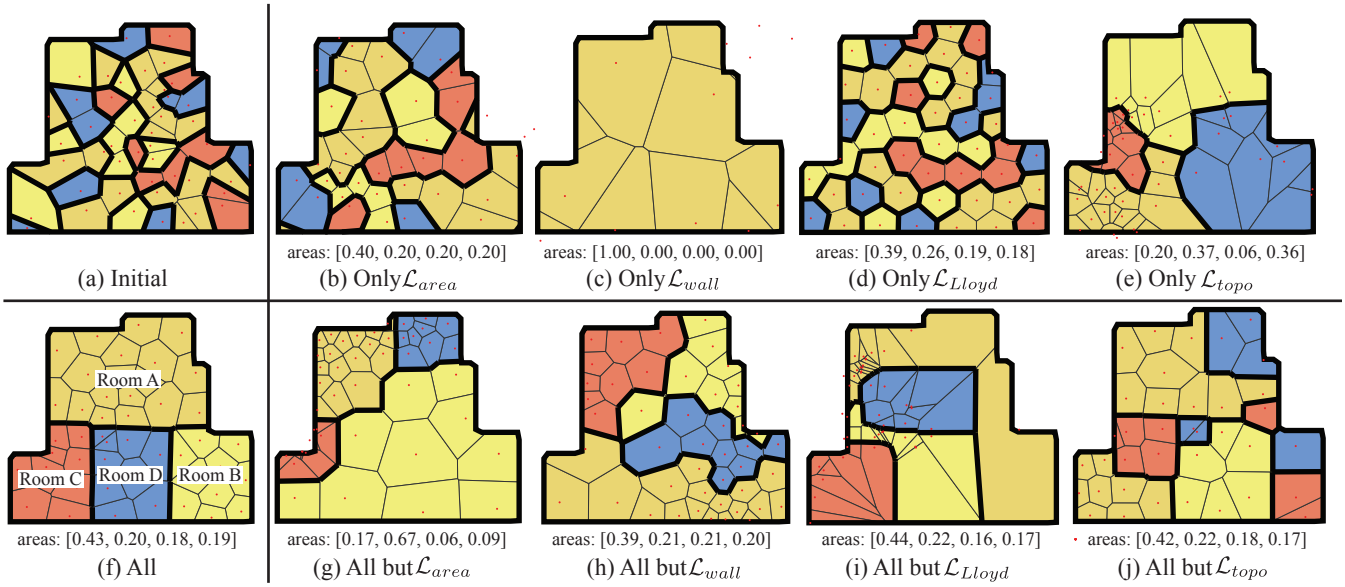

Figure 6: *Ablation study. (f) four rooms are involved (rooms A, B, C, and D). Room A has to be connected to the other three rooms. The target areas for rooms A, B C, and D are 0.4, 0.2, 0.2, and 0.2 respectively. Starting from the initial tessellation (a), top row shows the results when we optimized for just one loss: (b) area loss $\mathcal{L}_{\text{area}}$, (c) the wall loss $\mathcal{L}_{\text{wal}}$, (d) the Lloyd loss $\mathcal{L}_{\text{Lloyd}}$, and (e) the topology loss $\mathcal{L}_{\text{topo}}$. Compared to the result with all the loss combined (f), the bottom row shows the result when just one loss is removed; (g) area loss $\mathcal{L}_{\text{area}}$, (h) the wall loss $\mathcal{L}_{\text{wal}}$, (i) the Lloyd loss $\mathcal{L}_{\text{Lloyd}}$, and (j) the topology loss $\mathcal{L}_{\text{topo}}$. We report the list of resulting rooms' areas for rooms A, B, C, and D at the bottom of each optimization result.*

such as the one shown in Figure 7. We use 40 to 50 sites for the optimization. Whereas the number of sites controls the complexity of the room shape, we observe that the overall room configuration was not sensitive to this number if sufficiently many sites are present. In our current naive implementation, the speed of the optimization is quadratic in the number of sites as we compute the shape of the cells independently from the other cells. However, the computation was sufficiently light to perform the optimization during the interaction. For example, in Figure 6, where 49 sites are present, each iteration of the optimization takes 2.0 milliseconds on average measured on a MacBook Pro 2019 model with 2.6GHz Intel Core i7.

Convergence Figure 7 shows the tessellation of the room during the optimization and convergence of each loss. Note that, in contrast to other losses, the wall loss $\mathcal{L}_{wall}$ does not converge toward zero, this is because this loss is designed to be non-zero if walls exist. For the concave region, some sites must travel outside the given region Ω to satisfy the topology constraints as the topology of the initial condition is given randomly. Typically, the topology loss rapidly converges to zero when the topology constraints are satisfied.

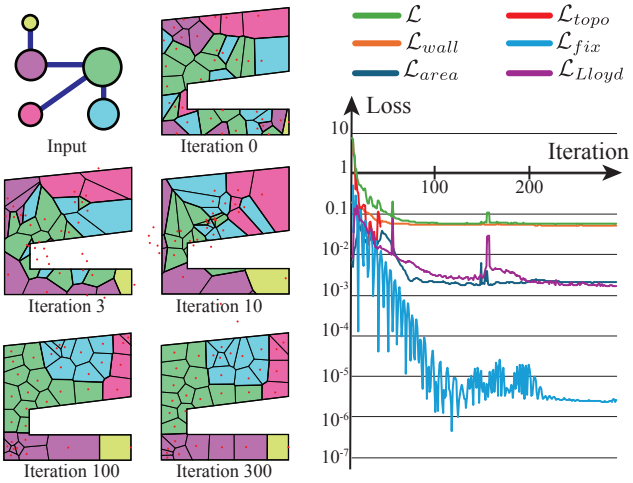

Figure 7: *Convergence of the optimization. (Left) During the optimization, the sites travel outside the given region to satisfy the topology constraints. (Right) The convergence of the losses.*

4.2. Ablation study

We optimize the sum of the loss functions (e.g., $\mathcal{L}_{\text{area}}$, $\mathcal{L}_{\text{wall}}$, $\mathcal{L}_{\text{Lloyd}}$, $\mathcal{L}_{\text{topo}}$). To demonstrate the effect of each loss function, we compute the optimization using each loss function alone. In addition, we also compute the optimization results that excludes each loss function from the summation (see Figure 6). We confirmed that only the summation of all the loss functions resulted in a reasonable optimization result. For example, when we optimize only for the wall loss $\mathcal{L}_{\text{wall}}$, most of the sites escape outside the input region such that no walls are present inside the region. If we optimize only for the Loyd loss $\mathcal{L}_{\text{Lloyd}}$, the diagram ends up with the centroidal Voronoi tessellation. Figure 6 demonstrates that all the loss func-

tions have the intended effects. For example, without the area loss $\mathcal{L}_{\text{area}}$, the area is not regulated and without the wall loss $\mathcal{L}_{\text{wall}}$, the walls are not straight and are cluttered.

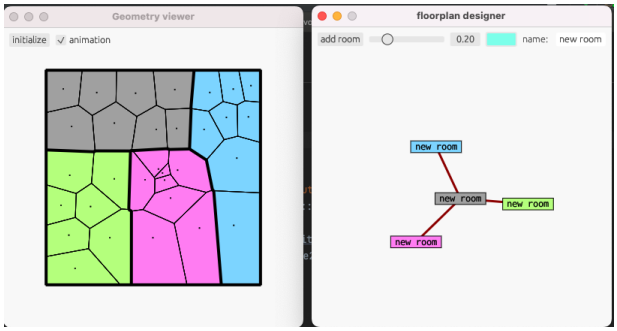

Figure 8: *Screenshot of our design interface. (Left) The convergence of the floor plan is animated. (Right) The user can specify the number of rooms, their connections, and their areas.*

4.3. Interactive design interface

Our optimization is sufficiently fast to allow the user to edit the area constraints interactively during the optimization. Hence, we design a simple user interface for the floor plan design (see Figure 8). The user inputs the boundary geometry as a polygon, and then specifies the number of rooms and their connectivity.

When the user presses the "initialize" button, the Voronoi diagram with random site locations is generated. Then, the animation visualizes the convergence of the design into a valid floor plan. When a room is selected, the user can interactively change its area. Please refer to the accompanying video for the animation.

5. Conclusion

Our floor plan representation using the Voronoi diagram facilitates topological changes in the room topology, yet enables the specification of the functionality based on the explicit boundary representation. However, this study has a few limitations we need to mention.

Limitation As shown in Section 4.2, simultaneously minimizing all the losses (e.g., wall length loss $\mathcal{L}_{\text{wall}}$, the area loss $\mathcal{L}_{\text{area}}$, and the Lloyd loss $\mathcal{L}_{\text{Lloyd}}$) by minimizing the sum of these losses is not always possible. Hence, the optimized floor plan sometimes contains small defects – the room areas deviate slightly from the target areas, the walls of the rooms are not always straight, and they do not always intersect at the right angle. Although a small deviation from the target area is not usually critical in practice and our interactive user interface allows the user to adjust the room size during optimization, developing new loss functions that automatically generate better optimization results remains a topic for future work.

This is infrequent, but there is a case in which the topology loss $\mathcal{L}_{\text{topo}}$ does not converge to zero. As shown in Figure 10, we identified a failure mode in which the four vertices oscillate to satisfy the two locally contradictory topology constraints. When two sites approach to connect the separated rooms, the connection between

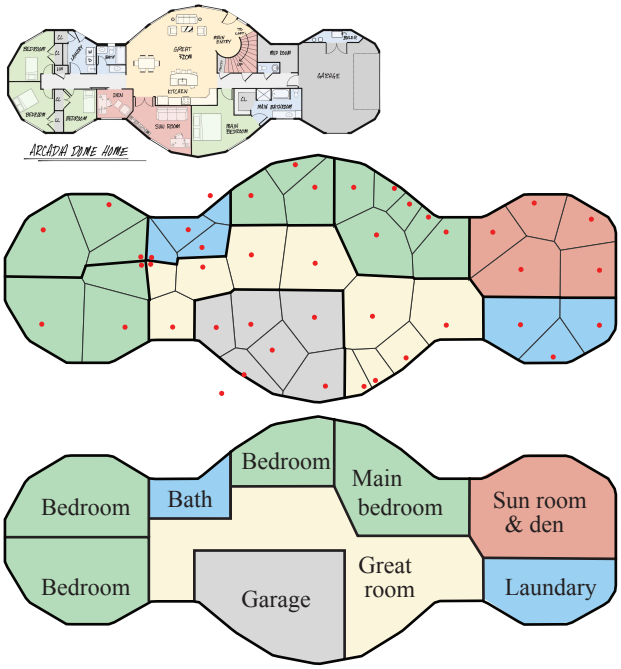

Figure 9: *Complex example. From the boundary of the floor plan of an actual architecture (Top), we optimize the floor plan with 9 rooms with the topological constraint that the "Great room" is connected to all the other rooms (Middle). We manually straighten the walls to produce realistic floor plan (Bottom). Top image credit: Dave South via Monolithic Dome Institute (CC BY-SA 4.0 license)*

the other rooms breaks, and vice versa. Note that this oscillation is usually naturally resolved as the balance between the sites changes.

The combination of the wall and area losses $\mathcal{L}_{\text{wall}}$ and $\mathcal{L}_{\text{area}}$ typically produce rooms with small aspect ratios. However, our optimization sometimes produces narrow regions surrounded by walls (see Figure 6-center). Furthermore, our topology loss only considers the topological connection between rooms, and thus sometimes two rooms are connected by an extremely narrow wall (see Figure 6-right). Although it was rare for our optimization to produce unnatural results, designing new losses to penalize these poor floor plans is a topic for future work. For instance, we plan to add another loss that penalizes the short shared wall length between adjacent rooms. As shown in Figure 9, we currently need to manually adjust the wall configuration to produce practical floor plan.

Future work Currently, our method does not generate doors between the connected rooms. We would like to optimize the position of the door by incorporating new loss functions that account for the usability of the floor plan using motion planning techniques. Similarly, we would like to consider various more complicated functionalities such as natural lighting and natural air ventilation.

It would also be interesting to extend the room connectivity constraints to rooms that must not be adjacent. Although the current topology loss is intended to pull separated islands, we plan to de-

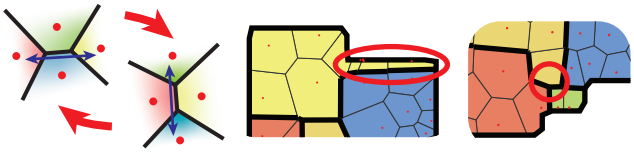

Figure 10: *Failure cases. (Left) Oscillation occurs when optimizing two locally contradicting connections (i.e., green and blue rooms, and yellow and red rooms) simultaneously at the shared corner of four rooms. Sometimes the optimization produces rooms with a narrow region (center) and excessively small boundaries between two connected rooms (right).*

sign a loss that repels partitions to achieve this effect. A similar effect could also be achieved by regularizing the length of the wall that partitions two rooms that must not be adjacent.

Our room-shape representation is based on the Voronoi diagram, hence the walls are flat. We would like to achieve smoothly curved interior walls by considering a distance measure in addition to the Euclidean distance or by considering the weighted Voronoi diagram.

The design of a multilayer floor plan is a promising avenue for future studies. To generate a floor plan for a multi-floor building, we can combine multiple 2D floor plans while the locations of the inter-floor connections (e.g., stairs or elevators) are constraints. We are also interested in a more flexible design, including (maisonettes and lofts) using the 3D Voronoi diagram with various 3D constraints.

Acknowledgement

We would like to thank anonymous reviewers for reviewing the submission. This work was supported by JSPS KAKENHI Grant Numbers JP21K11910, 23KJ0699.

References

[BBB10] BROCHU T., BATTY C., BRIDSON R.: Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. 29*, 4 (jul 2010). URL: https://doi.org/10.1145/1778765.1778784, doi:10.1145/1778765.1778784. 3

[BDWR12] BUSARYEV O., DEY T. K., WANG H., REN Z.: Animating bubble interactions in a liquid foam. *ACM Trans. Graph. 31*, 4 (jul 2012). URL: https://doi.org/10.1145/2185520.2185559, doi:10.1145/2185520.2185559. 2

[CKP*24] COGO E., KRUPALIJA E., PRAZINA I., BEĆIROVIĆ Š., OKANOVIĆ V., RIZVIĆ S., MULAHASANOVIĆ R. T.: A survey of procedural modelling methods for layout generation of virtual scenes. *Computer Graphics Forum 43*, 1 (2024), e14989. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14989, doi:https://doi.org/10.1111/cgf.14989. 2

[Cor24] CORBUSIER L.: *Vers une architecture*. Collection de "L'esprit nouveau". G. Crès et Cie, 1924. URL: https://books.google.co.jp/books?id=FbogAAAAMAAJ. 2

[dGWH*15] DE GOES F., WALLEZ C., HUANG J., PAVLOV D., DESBRUN M.: Power particles: an incompressible fluid solver based on power diagrams. *ACM Trans. Graph. 34*, 4 (jul 2015). URL: https://doi.org/10.1145/2766901, doi:10.1145/2766901. 2

[ESMR20] EGOR G., SVEN S., MARTIN D., REINHARD K.: Computer-aided approach to public buildings floor plan generation. magnetizing floor plan generator. *Procedia Manufacturing 44* (2020), 132–139. The 1st International Conference on Optimization-Driven Architectural Design (OPTARCH 2019). URL: https://www.sciencedirect.com/science/article/pii/S2351978920308003, doi:https://doi.org/10.1016/j.promfg.2020.02.214. 2

[Fac23] FACE H.: candle. https://github.com/huggingface/candle, 2023. 5

[FRS*12] FISHER M., RITCHIE D., SAVVA M., FUNKHOUSER T., HANRAHAN P.: Example-based synthesis of 3d object arrangements. *ACM Trans. Graph. 31*, 6 (nov 2012). URL: https://doi.org/10.1145/2366145.2366154, doi:10.1145/2366145.2366154. 2

[FXL*23] FENG F., XIONG S., LIU Z., XIAN Z., ZHOU Y., KOBAYASHI H., KAWAMOTO A., NOMURA T., ZHU B.: Cellular topology optimization on differentiable voronoi diagrams. *International Journal for Numerical Methods in Engineering 124*, 1 (2023), 282–304. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.7121, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.7121, doi:https://doi.org/10.1002/nme.7121. 3

[FYY*16] FENG T., YU L.-F., YEUNG S.-K., YIN K., ZHOU K.: Crowd-driven mid-scale layout design. *ACM Trans. Graph. 35*, 4 (jul 2016). URL: https://doi.org/10.1145/2897824.2925894, doi:10.1145/2897824.2925894. 2

[GL17] GUO Z., LI B.: Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system. *Frontiers of Architectural Research 6* (01 2017). doi:10.1016/j.foar.2016.11.003. 2

[HHT*20] HU R., HUANG Z., TANG Y., VAN KAICK O., ZHANG H., HUANG H.: Graph2plan: Learning floorplan generation from layout graphs. *ACM Trans. Graph. 39*, 4 (Aug. 2020). URL: https://doi.org/10.1145/3386569.3392391, doi:10.1145/3386569.3392391. 2

[HHW22] HE F., HUANG Y., WANG H.: iplan: Interactive and procedural layout planning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Los Alamitos, CA, USA, jun 2022), IEEE Computer Society, pp. 7783–7792. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.00764, doi:10.1109/CVPR52688.2022.00764. 2

[HMVDVI13] HENDRIKX M., MEIJER S., VAN DER VELDEN J., IOSUP A.: Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl. 9*, 1 (feb 2013). URL: https://doi.org/10.1145/2422956.2422957, doi:10.1145/2422956.2422957. 2

[JFRM17] JULIAN F. ROSSER G. S., MORLEY J. G.: Data-driven estimation of building interior plans. *International Journal of Geographical Information Science 31*, 8 (2017), 1652–1674. URL: https://doi.org/10.1080/13658816.2017.1313980, arXiv:https://doi.org/10.1080/13658816.2017.1313980, doi:10.1080/13658816.2017.1313980. 2

[KB15] KINGMA D., BA J.: Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)* (San Diega, CA, USA, 2015). 5

[LJ19] LIU H.-T. D., JACOBSON A.: Cubic stylization. *ACM Transactions on Graphics* (2019). 3, 4

[LSJ22] LING S., SHARP N., JACOBSON A.: VectorAdam for rotation equivariant geometry optimization. In *Advances in Neural Information Processing Systems* (2022), Oh A. H., Agarwal A., Belgrave D., Cho K., (Eds.). URL: https://openreview.net/forum?id=df1g_KeEjQ. 5

[LWL*09] LIU Y., WANG W., LÉVY B., SUN F., YAN D.-M., LU

L., YANG C.: On centroidal voronoi tessellation-energy smoothness and fast computation. *ACM Trans. Graph. 28*, 4 (sep 2009). URL: https://doi.org/10.1145/1559755.1559758, doi:10.1145/1559755.1559758. 4

[MCR22] MEHTA I., CHANDRAKER M., RAMAMOORTHI R.: A level set theory for neural implicit evolution under explicit flows. In *European Conference on Computer Vision* (2022), Springer, pp. 711–729. 3

[MSK10] MERRELL P., SCHKUFZA E., KOLTUN V.: Computer-generated residential building layouts. *ACM Trans. Graph. 29*, 6 (Dec. 2010). URL: https://doi.org/10.1145/1882261.1866203, doi:10.1145/1882261.1866203. 1, 2

[MSL*11] MERRELL P., SCHKUFZA E., LI Z., AGRAWALA M., KOLTUN V.: Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. 30*, 4 (July 2011). URL: https://doi.org/10.1145/2010324.1964982, doi:10.1145/2010324.1964982. 2

[MVLS14] MA C., VINING N., LEFEBVRE S., SHEFFER A.: Game level layout from design specification. *Computer Graphics Forum 33*, 2 (2014), 95–104. 2

[NCC*20] NAUATA N., CHANG K., CHENG C., MORI G., FURUKAWA Y.: House-gan: Relational generative adversarial networks for graph-constrained house layout generation. *CoRR abs/2003.06988* (2020). URL: https://arxiv.org/abs/2003.06988, arXiv:2003.06988. 2

[NHC*21] NAUATA N., HOSSEINI S., CHANG K.-H., CHU H., CHENG C.-Y., FURUKAWA Y.: House-gan++: Generative adversarial layout refinement networks, 2021. arXiv:2103.02574. 2

[NLCT24] NUMEROW L., LI Y., COROS S., THOMASZEWSKI B.: Differentiable voronoi diagrams for simulation of cell-based mechanical systems. *ACM Transactions on Graphics (TOG) 43*, 4 (2024). 3, 4

[PYW14] PENG C.-H., YANG Y.-L., WONKA P.: Computing layouts with deformable templates. *ACM Trans. Graph. 33*, 4 (jul 2014). URL: https://doi.org/10.1145/2601097.2601164, doi:10.1145/2601097.2601164. 2

[RAM*21] RAKOTOSAONA M., AIGERMAN N., MITRA N. J., OVSJANIKOV M., GUERRERO P.: Differentiable surface triangulation. *CoRR abs/2109.10695* (2021). URL: https://arxiv.org/abs/2109.10695, arXiv:2109.10695. 3

[RGA*21] RAKOTOSAONA M.-J., GUERRERO P., AIGERMAN N., MITRA N. J., OVSJANIKOV M.: Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021), pp. 22–31. 3

[RG13a] RODRIGUES E., GASPAR A. R., ÁLVARO GOMES: An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 1: Methodology. *Computer-Aided Design 45*, 5 (2013), 887–897. URL: https://www.sciencedirect.com/science/article/pii/S0010448513000031, doi:https://doi.org/10.1016/j.cad.2013.01.001. 2

[RG13b] RODRIGUES E., GASPAR A. R., ÁLVARO GOMES: An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 2: Validation and performance tests. *Computer-Aided Design 45*, 5 (2013), 898–910. URL: https://www.sciencedirect.com/science/article/pii/S0010448513000055, doi:https://doi.org/10.1016/j.cad.2013.01.003. 2

[SHF23] SHABANI M. A., HOSSEINI S., FURUKAWA Y.: Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2023), pp. 5466–5475. 2

[SLY22] SFIKAS K., LIAPIS A., YANNAKAKIS G. N.: A general-purpose expressive algorithm for room-based environments. In *Proceedings of the 17th International Conference on the Foundations of Digital Games* (New York, NY, USA, 2022), FDG '22, Association for Computing Machinery. URL: https://doi.org/10.1145/3555858.3563262, doi:10.1145/3555858.3563262. 2

[SWL*22] SUN J., WU W., LIU L., MIN W., ZHANG G., ZHENG L.: Wallplan: Synthesizing floorplans by learning to generate wall graphs. *ACM Trans. Graph. 41*, 4 (July 2022). URL: https://doi.org/10.1145/3528223.3530135, doi:10.1145/3528223.3530135. 2

[SZZW23] SUN J., ZHENG L., ZHANG G., WU W.: BubbleFormer: Bubble Diagram Generation via Dual Transformer Models. *Computer Graphics Forum* (2023). doi:10.1111/cgf.14984. 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. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13380, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13380, doi:https://doi.org/10.1111/cgf.13380. 2

[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 Trans. Graph. 38*, 6 (nov 2019). URL: https://doi.org/10.1145/3355089.3356556, doi:10.1145/3355089.3356556. 2

[WPLN*20] WILLIAMS F., PARENT-LEVESQUE J., NOWROUZEZAHRAI D., PANOZZO D., YI K. M., TAGLIASACCHI A.: Voronoinet: General functional approximators with local support. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (June 2020). 2

[WZC*23] WANG S., ZENG W., CHEN X., YE Y., QIAO Y., FU C.: Actfloor-gan: Activity-guided adversarial networks for human-centric floorplan design. *IEEE Transactions on Visualization and Computer Graphics 29*, 03 (mar 2023), 1610–1624. doi:10.1109/TVCG.2021.3126478. 2

[XRW*22] XING J., RUAN L., WANG B., ZHU B., CHEN B.: Position-based surface tension flow. *ACM Trans. Graph. 41*, 6 (nov 2022). URL: https://doi.org/10.1145/3550454.3555476, doi:10.1145/3550454.3555476. 3

[YWLL13] YAN D.-M., WANG W., LÉVY B., LIU Y.: Efficient computation of clipped voronoi diagram for mesh generation. *Computer-Aided Design 45*, 4 (2013), 843–852. 5

[YWVW13] YANG Y.-L., WANG J., VOUGA E., WONKA P.: Urban pattern: layout design by hierarchical domain splitting. *ACM Trans. Graph. 32*, 6 (nov 2013). URL: https://doi.org/10.1145/2508363.2508405, doi:10.1145/2508363.2508405. 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 Trans. Graph. 30*, 4 (July 2011). URL: https://doi.org/10.1145/2010324.1964981, doi:10.1145/2010324.1964981. 2

[ZFD*17] ZHANG X., FANG G., DAI C., VERLINDEN J., WU J., WHITING E., WANG C. C.: Thermal-comfort design of personalized casts. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2017), UIST '17, Association for Computing Machinery, p. 243–254. URL: https://doi.org/10.1145/3126594.3126600, doi:10.1145/3126594.3126600. 3

[ZYX23] ZHONG X., YU F., XU B.: A human–machine collaborative building spatial layout workflow based on spatial adjacency simulation. In *Hybrid Intelligence* (Germany, Apr. 2023), Yuan P., Chai H., Yan C., Li K., Sun T., (Eds.), Computational Design and Robotic Fabrication, Springer, pp. 14–24. International Conference on Computational Design and Robotic Fabrication, CDRF ; Conference date: 27-06-2022 Through 05-07-2022. URL: https://digitalfutures.international/conference/, doi:10.1007/978-981-19-8637-6_2. 2