2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

LEARNING INTERIOR DESIGN RULES FROM USER-CREATED LAYOUT EXAMPLES

Christoph Sydora and Eleni Stroula
Department of Computing Science, University of Alberta, Canada
{csydora, stroulia}@ualberta.ca

Abstract

When creating layouts using 3D interior design tools, users are typically guided by implicit design rules that aim to improve function and comfort. This tacit knowledge is extremely valuable for automated layout generation tools, such that automatically generated layouts exhibit similar function and comfort characteristics as user-designed ones. This paper evaluates a novel rule-learning method that, given user-designed layouts, learns rules that characterize the input layouts. Our comparison of the perceived quality of user- and computer-generated layouts indicates that the method works well in some room scenarios and better with fewer, more consistent-looking input layouts.

Introduction

Interior layout design is a difficult task. Large spaces with many furnishing objects can be laid out in a wide variety of ways, which can vary in their aesthetics and comfort. Rules, often developed through a deep understanding of factors such as human behaviour, ergonomics, aesthetics, and safety, are a way of evaluating the quality of a layout. Simply put, a layout that more closely aligns with a set of expert design rules should perform its function better than a layout that does not.

Computer-Aided Design (CAD) tools help designers create 3D renderings of layouts to plan and visualize design options, such that they can be manually or automatically evaluated against rules. Modern CAD tools have begun to support generative design tools that automatically design and optimize layouts based on rules. Thus, CAD tools provide designers with a means to (a) evaluate the rule-compliance of their layout design prior to implementation, and (b) optimize these layouts towards higher rule compliance and quality.

To perform these automated rule-compliance-based tasks, rules must be well defined and explicitly represented in a computer-processable format. Early approaches proposed the manual encoding of rules in general programming languages (Merrell et al. (2011); Yu et al. (2011)), with more recent approaches encoding them in Domain Specific Languages (DSLs) (Sydora and Stroulia (2020)). These rely on users to be familiar with the programming language of choice which requires training time, especially for general programming languages.

As large layout datasets have emerged, data-driven approaches have mitigated the challenge of programming design knowledge and patterns. The majority of these methods, however, are created solely for automatic layout generation and have not been designed to explicitly support rule explainability and rule modification.

One proposed data-driven method that addresses these shortcomings is rule code learning in Sydora and Stroulia (2024). Our approach learns a programmable rule code, specifically for geometric rules relating to furnishing layout and arrangement, represented in RuleDSL, an interior rule DSL. RuleDSL is human-readable and machine-executable at the same time, thus providing a more explainable rule representation. Furthermore, special-purpose RuleDSL editors allow for code review and modification. We evaluated our rule-learning method on a synthetic layout dataset, automatically generated based on expert rules defined in RuleDSL. The findings of this first evaluation study demonstrated that the rule-learning method was effective in learning the expert rules that guided the generation of the method's input layouts.

In this work, we describe a follow-up study to Sydora and Stroulia (2024), designed to evaluate this method's effectiveness in capturing the implicit knowledge embedded in our collected user-created layouts. Implicit knowledge is often relied upon to identify and create perceived quality layout configurations and is the basis for evaluating the output layout quality of data-driven approaches. Our evaluation assumes that our layout creators follow implicit knowledge-guided geometric design patterns. We hypothesize that our rule-learning method can capture these implicit patterns and formatted rules in the machine-readable RuleDSL. Therefore, this work is motivated by the following research hypothesis:

Implicit interior design knowledge can be learnt and expressed in the RuleDSL format

The hypothesis is tested by comparing two sets of layout datasets: (i) a human-created layout set and (ii) a computer-generated layout set, automatically produced by the rules learnt from the human-created layout set. The comparison is based on the perceived quality of the layouts, collected through evaluation scores from a user survey. If the evaluation scores of the user- and computer-generated layouts for the same room scenarios are not significantly different, then the hypothesis holds and the method can capture the implicit rules in the RuleDSL.

This paper first outlines related work on design knowledge

extraction and why the rule-language approach has its advantages. Next, a description of RuleDSL and the evaluated rule learning method is given at a high level. The method for user layout data collection is described, followed by the process of creating the computer-generated layout dataset from the user-created layouts. The user study used to collect layout evaluation scores is then detailed. Finally, we report and discuss the results of the evaluation survey and conclude with our contributions and potential future directions.

Interior Design Knowledge Extraction

Some of the earliest work on automatically generating (or synthesizing) layouts began in 2011. In the work of Merrell et al. (2011), layouts were generated by creating a layout evaluation (or cost) function. These functions were created programmatically by manually interpreting rules from interior design text into mathematical formulas which in turn were written into general-purpose programming languages. This challenge here is that the layout evaluation functions first had to be interpreted from text, which is often ambiguous and not described with computer formulation at its core. This requires design expertise in understanding the true intent of the described rule and also the implications of that rule should it conflict with other rules. The approach also requires expertise in computer programming as the rules have to then be written in machine-readable format which, using general-purpose code, can take years to master.

Using a few examples, Yu et al. (2011) included priors into their cost function to include statistical likelihood object relationships into their generation. An example of an early data-driven approach for object arrangement is Fisher et al. (2012). In this work, they use a probabilistic model trained based on Bayesian networks and Gaussian mixtures together with user-created scenes.

Over time, larger datasets emerged including SUNCG (Song et al. (2017)) and more recently 3D-FRONT (Fu et al. (2021)). This paved the way for deeper learning methods that could identify patterns automatically between furnishing objects. Thus, design knowledge (or priors) could be learnt from the examples and used to directly create layouts that appear or share similar qualities to the layouts in the dataset. Approaches in this realm include Wang et al. (2018); Ritchie et al. (2019); Wang et al. (2019) who utilize Convolutional Neural Networks on image representations of a layout, Zhang et al. (2020) who use a Neural Network of scene objects and their features, and Wei et al. (2023) which is a diffusion model inspired transformer-based iterative method for rearranging "messy" rooms.

Most recently, image generators have been able to create highly detailed images of buildings and interiors. These generators are either completely text-driven, such as the most well-known generator DALL-E 3 (Betker et al. (2023)), or can have some control using edges or input "skeletons" of the desired image, as in Zhang et al. (2023). In general, while producing impressive visuals of layouts, these image generation methods do not appear to be able to enforce desired geometrical relations within the layout. Each of these methods falls short of achieving our goal, which is to learn geometric arrangement rules from the examples provided in the dataset. This is because we differentiate priors, which are based on likelihood approximation functions, and rules, which are comprised of ranges of allowable values. Why we care about rules as opposed to priors is because rules allow us to more strictly enforce what is acceptable in a scene; priors, on the other hand, are plausibility estimations meant to mimic the dataset used to create them. It is theoretically possible for rules to be created based on priors by encapsulating the bounds of the priors; however, to our knowledge, this has not been investigated in previous research.

Background

The research in this paper is a practical evaluation of an existing rule Domain Specific Language (DSL) approach to rule extraction described in Sydora and Stroulia (2024). The data-driven approach at a high level extracts observations of interesting geometric relationships (Distance, Facing, Alignment) between pairs of object types (Walls, Tables, Chairs, etc.) from a set of layout examples and formulates them into machine-readable rule code. The motivation is that the generated machine-readable rule code can be read, understood, and modified; should the values being checked against need to be changed for example. This provides a starting point for creating rule code that can be used to check and evaluate a new design idea.

RuleDSL

To enhance readability, the generated code is described at a high-level, task-specific DSL; in our case RuleDSL. RuleDSL (Sydora and Stroulia (2020)) is a programming language for describing geometrical interior design rules between building elements (Wall, Floor, Door, etc.) and furnishings (Couch, Table, Shelf, etc.). The structure of a rule in RuleDSL breaks down into three components: (i) Object type filters (Existential Clauses) which define the types relevant to the rules and the existence relationships (one-to-one, one-to-many, etc.). For instance, a rule between one table and many chairs would say: *ANY t ∈ Tables; All c ∈ Chairs*. (ii) Property and Relationship Checks which define the checks on particular property and relationship functions. For instance, if specifying the distance between the tables and chairs, the relation check could state: *Distance(t,c)<0.5m*. (iii) Finally, the Logical Expression combines the property and relation checks and nested logical expressions using logical operands *AND* and *OR*. For instance: *Distance(t,c)<0.5m AND Facing(c,t)<0DEG*.

Together, this simple example looks like:

```
ANY t ∈ Table; ALL c ∈ Chair;
Distance(t,c)<0.5m AND Facing(c,t)<0DEG
```

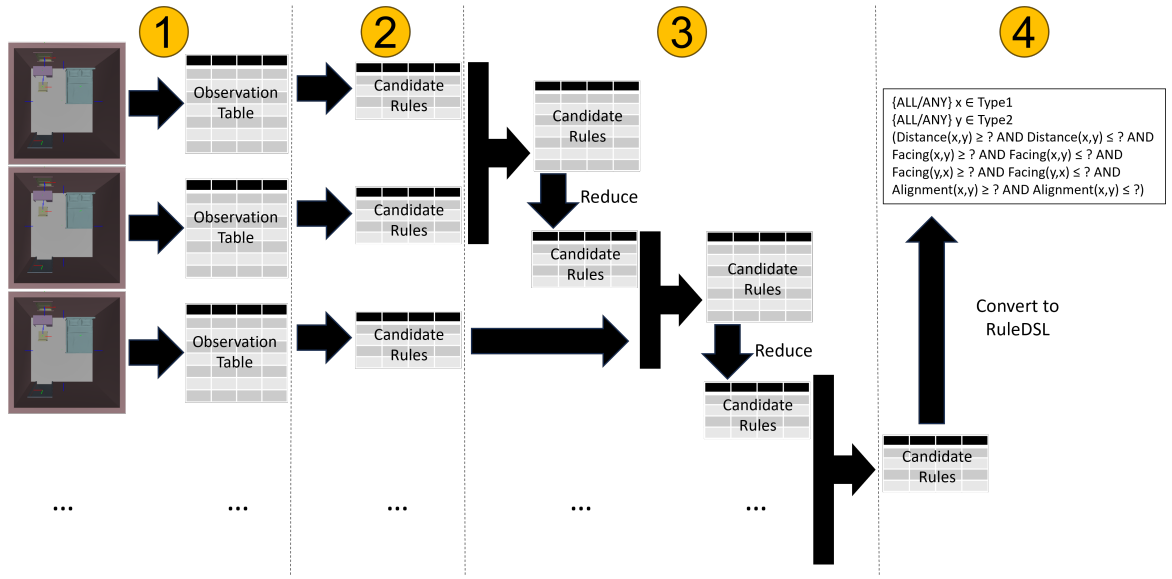

Figure 1: The rule learning method pipeline. (Step 1) Extract observations from the examples. (Step 2) Create candidate rules from the observations based on the types and clauses. (Step 3) Combine candidate rules over multiple examples. (Step 4) Convert candidate rules to RuleDSL.

Rule-Learning Method

The rule-learning method Sydora and Stroulia (2024) takes a rule template in a similar structure and adjusts the relation check values (the 0.5m and 0DEG in the example) of the rules such that they are the tightest bounds of check values that encapsulate all the observed geometric relation values in the input examples.

First, the rule template is determined. This involves selecting the types (Chair, Table, etc.), the clauses (ALL, ANY), and the relations of interest (Distance, Facing, etc.). The method iterates over all possible pairs of types and clause combinations observed in the examples, to create all possible candidate rules. The selected relations are based on geometric relationships, typically mentioned in the literature as influencing layout quality. The selection of these relations dictates the expressiveness of the output rules, as information that cannot be encompassed by these functions is unlikely to be covered by the examples. The goal is that the selected relations cover much of the users' design information and thus will increase the likelihood that the learnt rules capture user design information. In this study, the relations selected are *Distance*: using the length of the shortest line between the two meshes; *Facing*: the angle between the forward direction vector of the first object and the vector pointing to the center of the second object (We include both *Facing(A,B)* and *Facing(B,A)* as order matters here); *Alignment*: the angle between the forward direction vectors of two objects.

The following steps outline the rule-learning process, which is diagrammatically depicted in Figure 1:

(Step 1) First, examining one example at a time, **observations** are collected in an observation table. Observations are instances of type pairs (i.e. chair c1 with table t1) and the four relation values between them. For example, an specific observation may be *c1, t1, Distance(c1,t1)=0.5m, Facing(c1,t1)=0DEG, Facing(t1,c1)=90DEG, Alignment(c1,t1)=90DEG*.

(Step 2) To create **candidate rules** from one example, the observations are combined based on the clauses (ALL and ANY) of the rule. The general rule of thumb is that *ALL* case combines the relation values of all the objects of the same type, while the *ANY* case produces a new candidate rule. "Combining" in this case refers to creating a value range where the upper bound is the maximum of the two values and the lower bound is the minimum of the values. For example, given the observations *Distance(c1,t1)=0.5m* and *Distance(c2,t1)=0.2m*, the candidate rule range for *ANY t ∈ Tables; All c ∈ Chairs* would contain *Distance(c,t) ∈ [0.2m,0.5m]*. Each example can produce many candidate rules, even for the same type and clause.

(Step 3) The next step involves combining the candidate rules from multiple examples. Because there can be multiple candidate rules, all candidate rules must be combined with each other to create a comprehensive set of candidate rules over all examples. This is done iteratively, starting with an empty set of combined rules and combining the candidate rules of each new example to the current set of combined rules. Two candidate rules are combined by going over each property range, taking the minimum and the maximum values of the ranges, and creating a new rule with a broader range that covers both constituent ranges. There are two distinct ways to significantly reduce the number of candidate rules. First, many candidate rules can be either duplicates or supersets of other rules, i.e., the ranges of one rule fully cover all the ranges of another. In this case, the smaller ranged candidate rule should be kept, as passing this rule will always pass the broader rule. In

addition, once all candidate rules are constructed, keeping only the *N* rules with the most tightly bound ranges (sum of all range values) can reduce the number of rules and was shown to also improve rule quality (Sydora and Stroulia (2024)).

(Step 4) The final step of the process is to convert each of the candidate rules to a RuleDSL-formatted rule. This takes the candidate rule ranges and formats them into the code as seen on the right side of Figure 1. This allows utilizing the RuleDSL editors, model checker, and generative design tools.

User Created Layout Collection

To test and evaluate the rule learner's ability to learn and recreate perceived quality, a dataset of user-created layouts was collected.[1] Using the university-wide weekly email postings, we recruited participants, 18 years old and above, to design layouts and evaluate their quality. Participants were invited for one one hour in-person session and received a $15 gift card for participating.

In the layout editor, participants were shown an initial empty room and various furnishing objects on the right side of the room in random order as seen in Figure 2. Their task was to place all furnishing objects in the room to complete the layout; the layout could not be submitted until all objects were placed. Each move action was recorded, including the action timestamp, the Unique Identifier (ID) of the object moved, its type, its new location, and its new orientation.

Participants were given the nine room scenarios in Table 1 to complete in random order. They did one test scenario to understand the user interface, i.e. open one scenario, and test out the controls without submitting this test scenario. During the session, they were asked to create as many layouts as they could in the session time.

Each layout in the dataset contained two files: *(1) The 3D layout file* containing the geometry of each object as well as the object specifications such as unique ID and type; and *(2) A metadata file* with a designer score which was a self-evaluation of the expertise of the participant who created the layout, an action sequence list, and brief design notes transcribed from the verbal layout descriptions.

20 participants were recruited for the layout collection. Data from two participants who were unable to complete all nine layouts in the allotted session time were excluded. The remaining 18 participants created one layout for each of the nine room scenarios for a total of 162 layouts.

Our study system records the complete sequence of participant actions: each time an object is moved or rotated, its new location and orientation are added to the participant's session log. This provided insights into the degree of difficulty each room scenario presented for each individual participant. By examining the session logs, we observed that some participants did many moves in short periods, i.e. many minor edits, while others did fewer but longer, more well-thought-out moves. Therefore, action times were considered more meaningful than the number of moves.

To develop a metric corresponding to the scenario difficulty, we normalized the total time each scenario took for each participant: A time of 0 was given to the layout that took that participant the fastest to create, and a time of 1 was given to the layout that took them the longest, with all other layout times being normalized based on these two.

Table 1: Room Scenarios

Scenario	Initial Layout	Object List
Bathroom 1 (Small)	Small rectangular windowed room	Sink Shower Toilette Bathtub
Bedroom 1 (Large)	Large rectangular windowless room	Large bed Cabinet Armchair Side table (x2)
Bedroom 2 (Small)	Small rectangular windowed room	Desk Chair Small bed Side table Shelf
Kitchen 1 (Simple)	Large rectangular windowless room	Base corner cabinet Base cabinets (x3) Fridge Oven/Range Sink
Kitchen 2 (Dense)	Large rectangular windowless room	Base corner cabinet Base cabinets (x3) Fridge Oven/Range Sink Dining table Chair (x2)
Living room 1 (L-Shape, Simple)	L-shape windowed room	Couch TVstand Coffee table Armchair Side table Shelf (x2) Plant
Living room 2 (L-Shape, Dense)	L-shape windowed room	Couch TVstand Coffee table Armchair (x3) Side table (x2) Shelf (x2) Plant
Living room 3 (Rectangular, Simple)	Large rectangular windowless room	Couch TVstand Coffee table Armchair Side table Shelf (x2) Plant
Living room 4 (Rectangular, Dense)	Large rectangular windowless room	Couch TVstand Coffee table Armchair (x3) Side table (x2) Shelf (x2) Plant

[1]This research received research ethics approval from the University of Alberta Research Ethics Board, Project Name "Interior Design Layout Data Collection and Evaluation", No. Pro00124859, February 28, 2024.

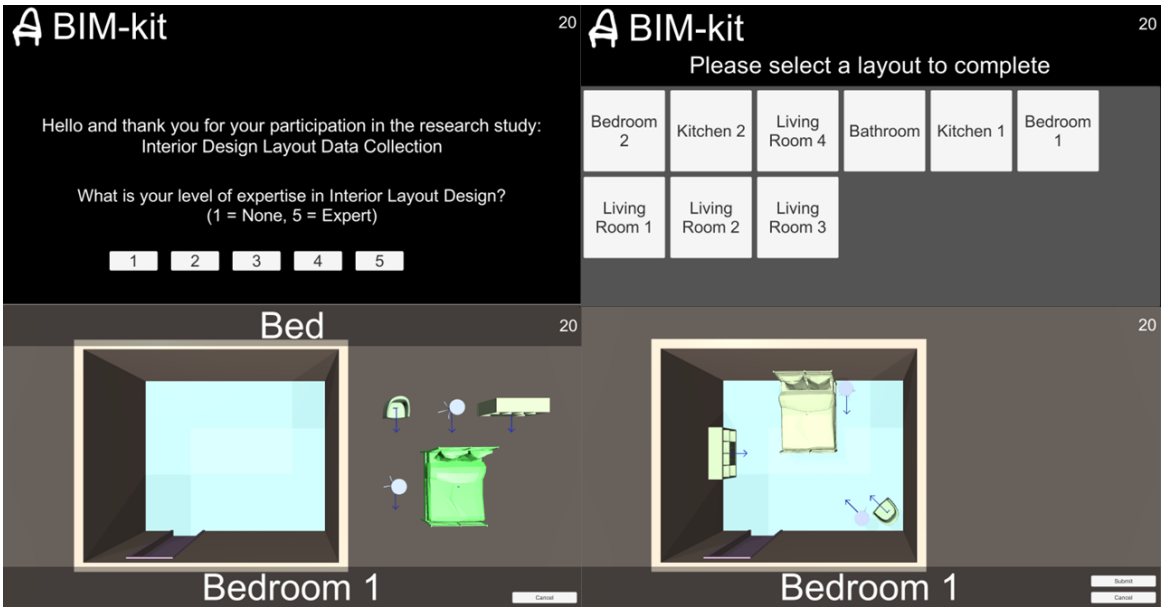

Figure 2: Layout collection User Interfaces (UIs). Top left: Initial Skill Level Question. Top Right: Room Scenario selection in random order; their ranges of difficulty can be seen in Figure 3. Bottom Left: Starting furnishing locations. Bottom Right: Example of layout placement.

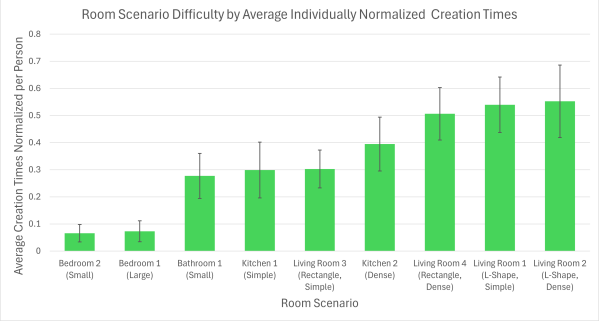

Figure 3: Room design difficulty approximation based on average comparison of design durations normalized by each participant.

Then all normalized creation times were averaged over all participants, the results of which can be seen in Figure 3. Evidently, the bedroom scenarios were the fastest to create, followed by the bathroom, kitchens and the "easy" living room, followed by the "challenging" living rooms. The challenging living rooms were living rooms that had either more objects (higher density), were L-shaped, or both (Living Room 2). As anticipated, difficulty was very closely correlated to the number of objects. We should note here that not all objects were equally easy to identify. This was particularly evident with the shower object in the bathroom scenario, as some participants were unclear about where the access side of the shower was. This had the side effect of making some room scenarios more difficult to understand and thus creating slight hesitations.

Rule Learning Perceived Quality Evaluation

To evaluate the rule learning method's ability to learn and express implicit design patterns in RuleDSL, the quality of our user-created layouts is compared against the quality of computer-generated layouts. The computer-generated layouts are guided by the rules output from the rule learning method. The intuition is that if the perceived qualities of the user-created layouts are the same as those of the computer-generated layouts, then the method was able to capture design knowledge relevant to high perceived quality. Furthermore, as the knowledge is represented in rules, this would also suggest that perceived quality can be evaluated through a rule-compliance evaluation.

The rule learning method evaluation process involves two stages: (i) creating the learnt layouts based on the learnt rules produced by the rule-learning method described above; and (ii) conducting a user survey to evaluate the original layouts and the automatically generated new ones.

Learnt Layout Generation

The process of creating the learnt layouts consists of two steps. First, rules are learnt by passing the user-created layouts into the rule learning method. In our experiments, learnt rules could have either 2, 5, or all of the 18 user-created layouts used as input. This enabled investigation of the effect of the number of input layouts on the ability of rule learning to capture perceived quality, particularly with fewer input layout counts, where, based on our initial tests, the changes in quality appeared more noticeable. For the case of the 2 and 5 learnt layouts, the input layouts were randomly selected from the 18 available. In our experiments, we performed this random selection five times to produce five learn layout inputs for each input layout count and each room scenario.

Next, layouts were generated using the Jump Search (JS) layout generation methodSydora and Stroulia (2023). using as input (i) the learnt rules, (ii) no rules (random placements), or (iii) expert rules. The authors created the ex-

pert rules by translating rules from various sources (websites, design books, related publications) into RuleDSL code. These expert-rule-generated layout evaluations were included to benchmark the capability of RuleDSL and JS to generate layouts, although they are limited by the quality of the rules they attempt to replicate. Five layouts were generated for expert and random layouts in the experiments as well. All generated layouts were given a high search iteration budget such that layouts in terms of their input rule quality were maximal.

This process was repeated for each of the nine room scenarios, however, to manage the number of designs shown to the participants, the survey contained only five room scenarios of varying design difficulty: Bedroom 1, Bathroom, Kitchen 2, Living Room 2 (Hard), and Living Room 3 (Easy). In total, 215 layouts were selected for review; 5 room scenarios, each with 18 user-created, 5 random, 5 expert-rule, and 5 learnt of each 2-, 5- and 18-learnt-rule layouts.

Layout Evaluation Survey

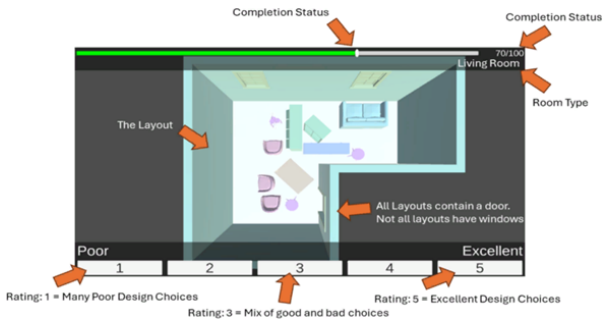

Figure 4: Survey website screenshot.

Different people perceive the quality of a layout differently. Someone with extensive knowledge of interior design might be able to identify a layout that functions better than another, while others may not be able to explain exactly why a particular layout is good or bad. It is also possible that a design error can be deemed more significant by one person than another, who may consider it only a minor flaw. To get an approximate numerical representation of how good a layout is, i.e. to evaluate its perceived quality, a survey was created where participants evaluate a layout's quality on a scale from 1 (a poor design) to 5 (an excellent design).

Survey participants were recruited using the University of Alberta university-wide weekly email postings and the Department of Computing Science Graduate Student Slack channel. A complete survey was expected to take no longer than 30 minutes, assuming they spend 10 to 15 seconds per layout. Potential participants were asked to recuse themselves if they had participated in the earlier layout collection phase of the study. As an incentive to improve participant recruitment, we offered the option to enter a draw for one of three $50 gift cards.

The survey required participants to review and score 100 layouts of the possible 215 layouts to fully participate. Survey participants were each asked to score layouts one at a time based on perceived quality on a scale from 1 (a poor design) to 5 (an excellent design); the survey UI can be seen in Figure 4.

Participants would not see the same layout twice. To ensure that all layouts received a roughly equal number of evaluations, the number of times a layout was evaluated was recorded. Then, of the layouts not yet evaluated by a user, the ones with fewer total evaluations were given a very high probability of being shown next. The probabilities were given a soft minimum or:

$$p(Layout) = \frac{e^{-l}}{\sum_{i=1}^{k} e^{-l_i}}$$

where l is the number of times the layout was reviewed and evaluated, and k is the number of layouts not evaluated by the participant.

Survey Results and Discussion

A total of 153 participants registered for the survey, and 130 of them completed the survey. Of the minimum range of scores assigned by any user (their maximum entered score minus minimum entered score), only one participant had a range of 2, eight had a range of 3, with the remaining 121 all used the full range of 4. The minimum variance of any user was 0.29; only four participants had a variance under 0.8, with the maximum variance being 1.7. These numbers demonstrate that the majority of participants generally gave varying scores to the layouts, i.e., not one single evaluation score to all layouts.

The minimum total number of participant scores received by a layout was 56. Figure 5 shows the average scores for each room scenario's layout. User layouts scored the highest (> 3.27) on average, while unsurprisingly, random layouts scored low (< 1.37). Expert-rule-created layouts scored moderately high (> 2.62) for bathrooms, bedrooms, and easy living rooms, while hard living rooms and kitchens scored low (< 2.12). Layouts learnt from two input layouts performed moderately high (> 2.74) for all scenarios except the kitchen scenario (1.88). Learning from 5 layouts had moderate performance (> 2.48) for bathrooms, bedrooms, and easy living rooms with a slight dip for 18 learning from layouts (> 2.08). For learning from 5 and 18 for hard living rooms and kitchens again scored low (< 1.62), close to random.

Our hypothesis is that if perceived layout quality evaluations are not significantly different from user-created layout evaluations, then the rule learner must have been able to capture quality design knowledge about the input layouts. Therefore, we observe the 95% confidence intervals of our layout evaluations to determine for which room scenario and number of input rules our rule learner can capture perceived quality design knowledge.

The confidence intervals for the evaluation in Figure 5 can be seen in Table 2. Based on the confidence intervals, there is no significant difference between the user-created layout

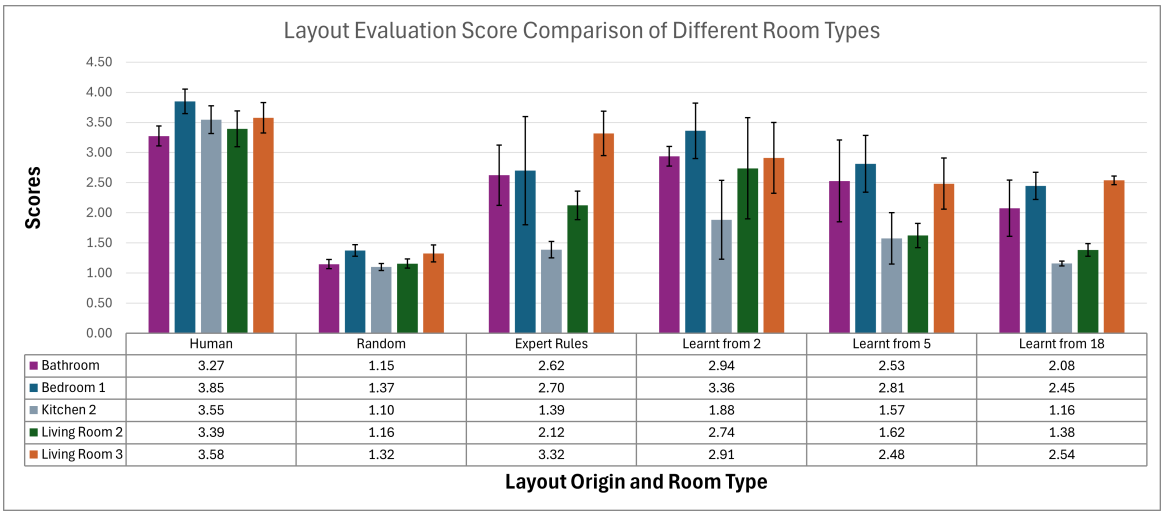

Figure 5: Results of the layout evaluation survey.

Table 2: Survey Evaluation Score 95% Confidence Intervals for each Room Scenario.

Scenario	Human	Random	Expert Rule	Learn 2	Learn 5	Learn 18
Bathroom	[3.11, 3.44]	[1.07, 1.22]	**[2.12, 3.12]**	[2.77, 3.10]	**[1.85, 3.21]**	[1.61, 2.54]
Bedroom 1	[3.65, 4.05]	[1.28, 1.47]	[1.80, 3.60]	**[2.90, 3.82]**	[2.34, 3.29]	[2.22, 2.67]
Kitchen 2	[3.32, 3.78]	[1.04, 1.16]	[1.25, 1.52]	[1.23, 2.54]	[1.15, 2.00]	[1.12, 1.20]
Living Room 2	[3.10, 3.69]	[1.08, 1.23]	[1.89, 2.36]	**[1.90, 3.58]**	[1.42, 1.82]	[1.28, 1.49]
Living Room 3	[3.32, 3.83]	[1.18, 1.46]	**[2.95, 3.69]**	**[2.32, 3.50]**	[2.06, 2.91]	[2.47, 2.61]

* Bold cells indicate confidence interval overlap with user-created layout evaluations (Human).

evaluations (Human) and the evaluations for Bathroom and Living Room 3 expert rules; Bedroom 1, Living Room 2, and Living Room 3 learnt from 2 input layouts; and Bathroom learnt from 5 layouts.

The findings suggest that it is indeed possible to capture the design knowledge using the intermediate DSL, however, this is only evident when learning from 2 in some room scenarios and learning from 5 in the bathroom scenario. Furthermore, as more input layouts are introduced, the rule learning suffers and produces decreasing quality learnt layouts.

Increasing Design Variability An interpretation of the results, which suggests the method works best with fewer input rooms, is that as more layouts are introduced, more variability is introduced, resulting in a higher likelihood of outlier designs or designs that contain some decision that does not align with the majority of participant designs. As the proposed method expands the ranges of relation values when combining candidate rules, all outlier design relation values are included, resulting in broader relation value ranges. From a rule learning approach, this is not a negative effect; by being included in the inputs, the outliers are assumed valid rule-compliant examples. However, it becomes clear that they become problematic for layout generation when the rules are not restrictive enough to create cohesive, plausible-looking designs.

Input Layout Quality The participants who created the layouts did not self-identify as experts; they all rated their expertise in interior design at a score of 3 or lower out of 5. The quality of the user-generated layouts may have been higher had we recruited interior designers, as it is possible that experts would have followed more similar guidelines, and thus we could have potentially gotten higher quality learnt layouts as well. The quality of the layouts was also not known when they were input into the rule learner, as the quality survey was only conducted once all layouts, both user- and computer-generated, were collected. A better approach may have been to first collect the layouts, score the layouts through the survey, then use the top-scored layouts only and eliminate layouts with lower perceived quality. Having only the higher perceived quality layouts as input may have also resulted in higher perceived quality layouts being learnt.

Rule Template A natural follow-up question is how good the template selection was. Evidently, the template did not capture the design knowledge very well in all scenarios. Therefore, analyzing templates with different relations and more complexity (not simply AND-ing the relation ranges) will be required. RuleDSL does support more complex relationships, however, the approach will require a significantly more elaborate rule construction. In general, this experiment emphasizes that the perception of quality is complex and that capturing and recreating perceived quality is challenging, as it is quite subjective, as evident by the large ranges in many of the layout evaluation scores.

Conclusion

In this paper, we perform an evaluation of a rule programmable-code learning method that utilizes an interior rule Domain Specific Language, RuleDSL. The evaluation is based on a new user-created dataset that contains nine room scenarios, each with eighteen layouts per scenario. The research question being addressed is the ability of the rule learning method to capture interior design knowledge relevant to computationally evaluating perceived quality. We evaluate the method's ability through a user survey, comparing the perceived quality scores of the input user layouts and layouts generated automatically following the rule learning methods' output RuleDSL rules. Our findings suggest that the method can capture perceived quality for some room scenarios when fewer user layouts are input.

With the increase in digitized design and interior datasets, the automatic generation of layouts that match the quality of input layouts has gained much attention. To date, no existing methods are concerned with learning strictly enforced rules, but rather focus on the visual quality produced. Our previous work (Sydora and Stroulia (2024)) was aimed at learning strict geometric rules, however, the ability of our method to create layouts of high perceived quality was not yet investigated. The evaluation and results of this paper address this question.

Future research will be dedicated to improving the method such that learnt rules will afford the same complexity and expressiveness as RuleDSL, with the goal being to produce rules that enforce strict geometric relationships while also capturing perceived visual quality.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Mitacs Canada. Ethics was approved from the University of Alberta Research Ethics Board (REB), Project Name "Interior Design Layout Data Collection and Evaluation", Ethics ID: Pro00124859, February 28, 2024.

References

Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., et al. (2023). Improving image generation with better captions. Technical report, OpenAI.

Fisher, M., Ritchie, D., Savva, M., Funkhouser, T., and Hanrahan, P. (2012). Example-based synthesis of 3d object arrangements. ACM Transactions on Graphics (TOG), 31(6):1–11.

Fu, H., Cai, B., Gao, L., Zhang, L.-X., Wang, J., Li, C., Zeng, Q., Sun, C., Jia, R., Zhao, B., et al. (2021). 3d-front: 3d furnished rooms with layouts and semantics. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 10933–10942.

Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. (2011). Interactive furniture layout using interior design guidelines. ACM Transactions on Graphics (TOG), 30(4):87:1–10.

Ritchie, D., Wang, K., and Lin, Y.-a. (2019). Fast and flexible indoor scene synthesis via deep convolutional generative models. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6182–6190.

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. (2017). Semantic scene completion from a single depth image. Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition.

Sydora, C. and Stroulia, E. (2020). Rule-based compliance checking and generative design for building interiors using bim. Automation in Construction, 120:103368.

Sydora, C. and Stroulia, E. (2023). Comparative analysis of room generation methods using rule language-based evaluation in bim. In European Conference on Computing in Construction (EC3), volume 4. European Council on Computing in Construction.

Sydora, C. and Stroulia, E. (2024). Learning interior design rules from synthetic rule-compliant example bim layouts. SSRN 5080076 Preprint.

Wang, K., Lin, Y.-A., Weissmann, B., Savva, M., Chang, A. X., and Ritchie, D. (2019). Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. ACM Transactions on Graphics (TOG), 38(4):1–15.

Wang, K., Savva, M., Chang, A. X., and Ritchie, D. (2018). Deep convolutional priors for indoor scene synthesis. ACM Transactions on Graphics (TOG), 37(4):1–14.

Wei, Q. A., Ding, S., Park, J. J., Sajnani, R., Poulenard, A., Sridhar, S., and Guibas, L. (2023). Lego-net: Learning regular rearrangements of objects in rooms. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19037–19047.

Yu, L.-F., Yeung, S. K., Tang, C.-K., Terzopoulos, D., Chan, T. F., and Osher, S. (2011). Make it home: automatic optimization of furniture arrangement. ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH, 30(4):86:1–12.

Zhang, L., Rao, A., and Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 3836–3847.

Zhang, Z., Yang, Z., Ma, C., Luo, L., Huth, A., Vouga, E., and Huang, Q. (2020). Deep generative modeling for scene synthesis via hybrid representations. ACM Transactions on Graphics (TOG), 39(2):1–21.