Navigating Design Spaces: Finding Designs, Design Collections, and Design Subspaces

Abstract

Generative design systems can generate a wide panoply of solutions, from which designers search for those that best suit their interests. However, without guidance, this search can be highly inefficient, and many interesting solutions may remain unexplored. This problem is mitigated with automated exploration methods. Still, the ones typically provided by generative design tools are mostly based on black-box methods that drastically reduce the role of the designer, while more straightforward white-box mechanisms are dispersedly found in specific applications. This paper proposes the Navigator tool, which gathers a set of white-box mechanisms that automate the generation of default, random, similar, and hybrid designs and design subspaces, while also supporting the generation of design collections. The proposed mechanisms were tested with two generative systems that create, respectively, tower and chair designs. We expect that, by providing understandable mechanisms for navigating design spaces, designers can become more engaged in the search process.

Keywords

Generative Design, Design Space Exploration, Randomness, Hybridity, Similarity

Introduction

The design process comprehends the exploration of design alternatives and the selection of a solution that better responds to a given design problem. The generation, evaluation, comparison, modification, and selection of alternatives is crucial at the design concept phase as it can lead to better solutions.[1] Traditionally, alternatives are manually generated – for instance, by producing a multitude of sketches or editing a digital model. However, in such processes, designers deal with a single state at a time and the modification of one design takes considerable amounts of time and effort.

Generative Design Systems (GDSs) enable the fast generation and exploration of design alternatives by the means of computational processes. They can potentiate the creation of unforeseen solutions and thus tickle designers' imagination and creativity.[2] The set of all possible design alternatives (complete or not, feasible or not) producible by a GDS is called the design space. Exploring the design space consists of navigating from one alternative to another until a satisfactory solution (or set of solutions) is achieved.[3] Three types of design spaces can be identified: *implicit space* (the set of all designs that a system can possibly generate), *explicit space* (the set of visited designs)[3], and *hysterical space* (the set of designs from an implicit space that is achievable by recombining data from an explicit space).[1] The latter give the designer novel and potentially meaningful directions that, although may be parametrically close to what they have previously explored, may look entirely different.

GDSs can easily generate an enormous number of designs, since the design space grows exponentially with the number of parameters and the size of their ranges. Even a system with a very small number of parameters can generate a complex design space – as John von Neumann once said, it only takes four parameters to describe an elephant.

Visiting all possible solutions is out of question, thus users are left with the task of navigating throughout the design space until they find one design that satisfies their needs. However, manual exploration processes can easily become time-consuming, tedious, and inefficient. One can typically visualize few designs at a time[2] and, from all the solutions, not all are acceptable or complete. In fact, using these processes designers will end up exploring very few design alternatives.[3] This asks for tools that support designers in exploring solution spaces, allowing them to visit designs that would not be considered otherwise.

There are several exploration strategies, such as: parametric exploration (manual manipulation of parameter values), similarity-based exploration (semi-automated exploration of parametrically close alternatives), and optimization-based exploration (automatic selection of alternatives according to certain goals).[4] These approaches possess different degrees of automation, ranging from more manual to more automatic, which affect users' intervention in the search process. They also have different degrees of computational cost, which influence the speed in the search for solutions: e.g., optimization-based methods are more expensive (as they involve cycles of generation-analysis-regeneration), while parametric methods are less expensive (as they only involve cycles of generation).

Generative design tools comprehend a series of exploration methods, which can be classified by using the white-box and black-box metaphors. The first address explainable methods that allow users to be more interventive (thus able to influence the search) and are less computationally expensive, when compared to the latter. GDSs can be developed and used in several tools, such as Algorithmic Design (AD) tools (e.g., Grasshopper and Dynamo), where such systems are algorithmically described. Unfortunately, AD tools present limitations with regards to solution exploration and visualization, as they mostly address black-box exploration methods and often present a single model at a time, instead of multiple models.[4] Other tools are more oriented towards navigating GDSs, such as design space explorers, although these are more focused on the visualization and interaction aspects of the exploration process.[2,5] White-box methods can be dispersedly found in specific GDSs.[1,6]

In this paper, we propose the *Navigator* tool, which gathers and clarifies some of the most well-known white-box exploration mechanisms, combining human judgment with fast automated generation. These mechanisms generate designs and design subspaces from default, random, similar, and hybrid processes, giving the user the potential to explore solutions in the *hysterical space*.[1]

In the following sections we summarize some of the existing exploration methods, present the methodology employed in the development of *Navigator* and illustrate its applicability with two case studies.

Related Work

Exploration methods allow users to navigate throughout the design space until finding one or more satisfactory solutions. In this section, we briefly describe some of these methods, with particular emphasis on the white-box ones, as these are the focus of this paper.

Black-box Exploration

Goal-oriented search is one of the most used black-box methods by AD tools nowadays. Automatic search techniques, combining generation, analysis, and optimization, select the best solutions in a design space considering one or more design goals (e.g., structural performance, cost, and tallness). This technique is used, for instance, in the Autodesk's Refinery tool.[7] Other methods are based on the premise that users' choices are often intuitive, rather than being based on exact criteria.[2] In those cases, machine learning can be employed to refine the design space according to users' subjective preferences. This technique is used, for instance, in Autodesk's Dream Lens – based on user ranks, the system automatically selects the designs that better match their choices.[2]

These black-box methods are helpful in finding interesting solutions but lack an explainable definition and limit the designer's role in the exploration process, unlike white-box methods. Unfortunately, the latter have received less attention than the former, although they can be found in more specific generative design applications, as detailed in the next subsection.

White-box Exploration

There are different white-box methods to generate designs and design spaces. Designs can be found through several strategies, obtaining random, similar, and hybrid designs, design matrices and design collections.

Random designs, resulting from an arbitrary selection of parameter values, are probably one of the easiest and popular methods to select designs.[1,8] However, without any kind of guidance, the probability of randomly selecting a meaningful design from a large design space is very small.[3] In fact, random designs can look absurd, incomplete, or rather abstract. Nevertheless, these are the kind of solutions that are expectable at the design concept phase.[9] Remarkably, these can trigger unexplored ideas and lead to potentially interesting designs.[2,3] Beyond that, random choices are often used as staring points (e.g., in optimization processes) and are also valuable for debugging and refining a generative system.

Similar designs take inspiration from an existing design to create a new design. These designs can be obtained by different processes, namely by randomly generating values within a narrow range close to a design's parameter values[7] or by estimating the similarity degree between a design and other designs of the space – given by a distance (e.g., Euclidean similarity or Bray–Curtis dissimilarity) between pairs of design alternatives – and selecting the designs that have a lower degree.[4] While this is a valuable process for generating alternatives, consideration should be given to not overwhelming users with very similar variations.[1]

Hybrid designs share features of two or more designs. Hybrid designs can be generated from different strategies, namely, from a weighted interpolation between two or more designs[6,10], which gives a smooth transition from one design to another – a process also known as *morphing*. Another technique is to recombine parameter values, or voxels from several voxel-based models (i.e., models composed of small cubes in 3D space).[11,12]

Design matrices are sets of designs arranged in rows and columns. These can be generated by the Cartesian product of two sets of parametric values, which can be obtained from the interpolation of parametric ranges.[1,2,5,7] This strategy allows the user to perceive the meaning of a parameter and how it affects the design.

Design collections comprise designs typically from the same style but from different types (e.g., tableware collections composed of plates, bowls, mugs, and cups). These can be obtained by a system where a change in one design is propagated to the remaining designs of the collection.[6]

Besides designs, exploration methods can also generate design spaces. Design spaces characterize a coherent language whose designs share some recognizable features and principles concerning appearance, function, and meaning.[13] Languages can feature, for instance, a type (e.g., tableware[6]), a style (e.g., Mughul gardens[13]), or a brand (e.g., Harley-Davidson motorcycles[14]).

Design spaces can be extended or restricted. More generic design spaces (i.e., design superspaces) or more specific design spaces (i.e., design subspaces) can be obtained by adding or deleting parameters or relaxing or narrowing parametric ranges, respectively.[4] This was successfully demonstrated with shape grammars, where specific grammars are achieved by restricting rules, parameters, and/or labels of a more generic grammar. For example, a motorcycle grammar can be constrained to become a Harley-Davison grammar[14].

Hybrid design spaces result from the combination of two or more design spaces. For instance, hybrid (or composite) grammars can be obtained by merging two or more grammars according to different methods, such as: (1) merging shapes, (2) mixing shapes and rules, or (3) merging rules.[14,15] Note that, unless the grammars to be merged are subsets of a more generic grammar (thus sharing the same structure and vocabulary), the blending process will often require their modification. Hybrid grammars can be used to analyze the influences and analogies of different design languages and create new languages. For example, a crossover vehicle grammar can be restricted to existing vehicle subclasses (coupes, pickups, and SUVs), a hybrid class (combining features of several classes), and an original vehicle subclass.[16]

Similar design spaces are obtained from two or more overlapping design spaces, characterizing the similar features between them. This was tested with overlapping grammars, which have rules that are shared among them and rules specific to each one.[17] The similar (or common) grammar is given by the intersection of the grammars, i.e., using the rules and parametric ranges that are common to both.

Although the white-box mechanisms detailed in this section are available in several applications, these are specific to a given generative system. Generative design tools, on the contrary, provide several features for designers to create GDSs, but lack white-box mechanisms to navigate the design space generated by such systems. In fact, in comparison with the *development* of GDSs, the *exploration* of the design space produced by such systems has been less researched.[3] In the next section, we propose an implementation of such mechanisms to be used by any GDS. We assume that the GDS is a white box itself, i.e., that the correlation between the parameters and the generated design is perceivable.

Methodology

This section describes the methodology employed in the development of a design space exploration tool, called *Navigator*. The tool includes mechanisms to produce designs and design spaces, which are synthetically described in **Table 1** and illustrated in **Figure 1**. The subsequent paragraphs detail the calculation procedures.

Table 1. Navigation mechanisms (name, symbol, and short definition).

Designs	
Default Design, D_D	The default design of a design space
Random Designs, D_R	A design or set of designs randomly generated from a design space
Similar Designs, D_S	A design or set of designs similar to another design, in relation to a design space
Hybrid Designs, D_H	A hybrid design or set of designs generated by merging a set of designs
Design Collection, D_C	A set of designs generated from one design and a set of transformations
Design Matrix, D_M	A matrix of designs generated from one design and variations of two of its parameter values
Design Spaces	
Default Design Space, DS_D	A design space generated from a set of designs
Random Design Space, DS_R	A design space randomly generated
Similar Design Space, DS_S	A design space generated from the intersection of design spaces
Hybrid Design Space, DS_H	A design space generated from the union of design spaces
Design Space Size, $\|DS\|$	The size (or cardinality) of a design space

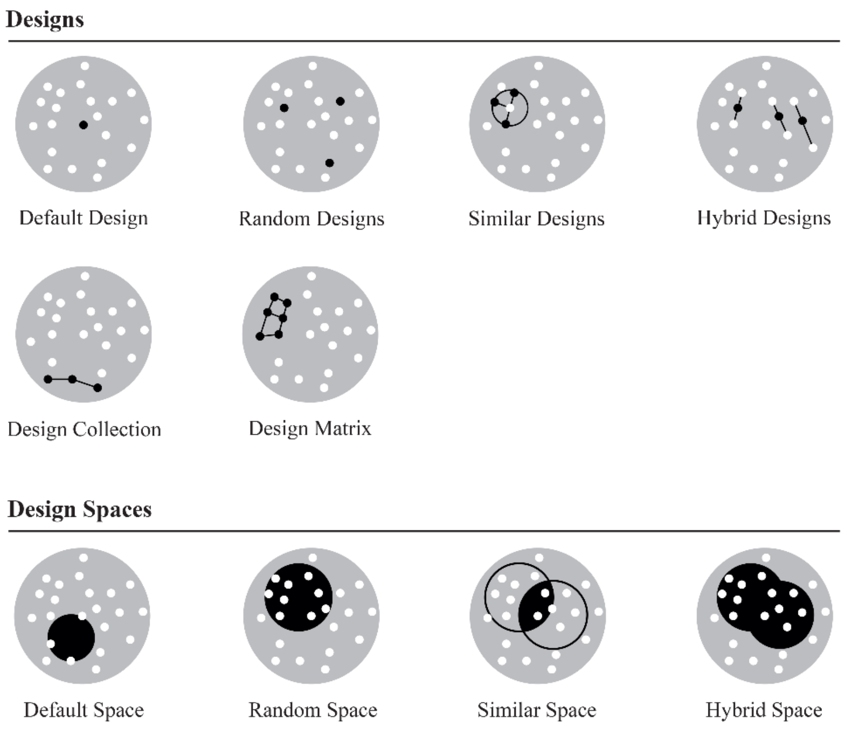

Figure 1. Navigation mechanisms (in black) given from a design space (in grey).

A design D is defined by a list of n parameters $p_1, \ldots, p_n$ and respective values $v_1, \ldots, v_n$. A design space DS is defined by a list of n parameters $p_1, \ldots, p_n$ and respective range of variations. When a parameter p_i is categorical (ordinal or unordered), its range is represented by an ordered set of m elements $\{e_1, \ldots, e_m\}$ (although in the case of unordered parameters this order is irrelevant); when a parameter p_i is numeric, its range is represented by the interval $[m_i, M_i]$, where m_i and M_i are the minimum and maximum values for p_i, respectively. From these definitions, the methods for calculating the navigation mechanisms are presented below for each dimension, given a set of user-specified inputs. Such methods are then replicated for all dimensions.

Default Design: given a design space DS, it returns, for each parameter, a *default* value. In the case of numerical parameters, it returns the mid-range value between the minimum and maximum values (Equation 1). This approach is adequate for the frequent case of parameters that have linear effects on the design (e.g., the effect of a circle's radius on its perimeter); in the case of nonlinear effects (e.g., the effect of a circle's radius on its area), a different approach that takes the nonlinearity into account may be more appropriate. In the case of categorical parameters, it returns the first element of the range. This mechanism is useful to quickly create default designs for given design spaces.

$$D_D = \frac{m + M}{2} \tag{1}$$

Random Designs: given a design space DS, it returns, for each parameter, a random value between the minimum and maximum values (Equation 2), or a random element from the set of variations. Given a number n, it returns a set of n random values. Random designs are useful for both developers (for debugging and refinement of the design space) and users (by suggesting new directions or ideas).[2] This mechanism can also be restricted to *valid designs*, where random designs are generated only if their parameter values meet a certain condition (e.g., at least one parameter value is greater than a certain measure).

$$D_R = \text{random}(m, M) \tag{2}$$

Similar Designs: given a design D characterized by the set of values v, a design space DS, and a weight $w \in [0,1] \subset \mathbb{R}$, it returns, for each parameter, a design similar to the design D (Equation 3). The weight indicates the degree of similarity with the original: the higher the weight, the less similar the design, and the closer it will be to the limits of the design space. If the weight is 0, the resulting design corresponds to the original design; if the weight is 1, the resulting design is the most possible dissimilar design (located on the border of the design space). Values of categorical parameters remain unchanged. If instead a weight w we give a range of weights $[w_1, w_2]$ and a number n, it returns a set of n similar designs for n weights in the range $[w_1, w_2]$. Similar designs allow the user to 'ask' the system to 'make more designs like this one' or 'show me different designs', by manipulating the weight.

$$D_s = v \pm w \cdot \max(v - m, M - v) \tag{3}$$

Hybrid Designs: given a set of designs $D_1, \dots, D_n$, a hybrid design can be calculated from one of two methods: given a set of weights $w_1, \dots, w_n$, the *mean method* returns, for each parameter, the weighted mean value between the design parameter's values (Equation 4a), while the *random method* returns a random combination of the said values (Equation 4b). In the first method, we can control the degree of similarity that the hybrid design has with existing designs, which we cannot control in the second method. As in the *Default Design* mechanism, the first approach better suits parameters with linear effects. In categorical parameters, the random method is applied. A set of hybrid designs can be obtained by giving a number n and, in the case of the mean method, giving a range of weights $[w_{i_1}, w_{i_2}]$ instead of a weight w_i.

$$D_H = \frac{\sum_{i=1}^{n} v_i w_i}{\sum_{i=1}^{n} w_i} \tag{4a}$$

$$D_H = \text{random}(v_1, \dots, v_n) \tag{4b}$$

Design Collection: given a design D and a set of transformations $t_1 \dots t_m$, such that t_i replaces a set of values of parameters $p_1 \dots p_n$ in design D with new values $v_1 \dots v_n$, a design collection is the set containing the original design D and the m resulting designs from applying the transformations to D (Equation 5). New values may or may not depend on other parameters' values. Transformations may be motivated by a given design space DS (if a value v does not fit a range in the DS, v will be replaced by the default value of the range).

$$D_C = \{D, t_1(D), \dots, t_m(D)\} \tag{5}$$

Design Matrix: given a design D and incremental variations m and n in two of its parameter values, it returns an $m \times n$ matrix of designs D_M (Equation 6). Each design D_{ij} is the entry at the i^{th} row and the j^{th}

column of D_M. This kind of incremental variation allows someone who is not familiar with the GDS to relate the parameters with their visual output.[4] Note that, although this approach can be easily generalized to more than two parameter values, it becomes difficult to visualize.

$$D_M = \begin{bmatrix} D_{11} & \cdots & D_{1n} \\ \vdots & \ddots & \vdots \\ D_{m1} & \cdots & D_{mn} \end{bmatrix} \tag{6}$$

Default Design Space: given a set of designs $D_1, \ldots, D_n$, it returns, for each parameter, the minimum and maximum values among them (Equation 7). In the case of a categorical parameter, it returns all the distinct elements of their ranges of variation (sorted in ascending order). The resulting design space will reproduce the input designs and new designs in between. This mechanism can be used, for example, to quickly define a language from a set of designs sharing that language.

$$DS_D = [\min(v_1, \ldots, v_n), \max(v_1, \ldots, v_n)] \tag{7}$$

Random Design Space: given a design space DS, it returns, for each parameter, two random values between the minimum and maximum values, sorted in ascending order (Equation 8), or a set of distinct random values among the elements of the range (for categorical parameters). Note that this mechanism will not bring much benefit if one wants to generate a random design within a random design space, but it can be used to create new sub-languages from an existing language.

$$DS_R = [D_{R_1}, D_{R_2}], D_{R_1} \leq D_{R_2} \tag{8}$$

Similar Design Space: given a set of design spaces $DS_1, \ldots, DS_n$, it returns the intersection of the parametric ranges[17]. This is given by the second smallest $\min_2$ and the second largest $\max_2$ values (Equation 9), or by the set's intersection (in the case of categorical parameters). If the intervals do not overlap (i.e., with disjoint sets), it will return the interval between them (for numerical parameters) or the elements of the first set (for categorical parameters). The reason why we considered non-overlapping intervals was that, for large sets of parametric ranges, it is probable that at least one of them does not overlap; thus, we prefer to provide an approximate solution than to not provide one at all. The similar design space gathers the common features of a set of design spaces, featuring an ambiguous language that can fit any of the original ones.

$$DS_S = \left[\min_2\left(m_{1,} \ldots, m_n, M_1, \ldots, M_n\right), \max_2\left(m_{1,} \ldots, m_n, M_1, \ldots, M_n\right)\right] \tag{9}$$

Hybrid Design Space: given a set of design spaces $DS_1, \ldots, DS_n$, it returns the union of the parametric ranges[17]. This is given by the smallest and largest values (Equation 10) or by the set's union (for categorical parameters). If the intervals do not overlap, it will return an interval containing the values in the original intervals and the values between said intervals (in the case of numerical parameters). The hybrid design space gathers all the features of a set of design spaces.

$$DS_H = \left[\min\left(m_{1,} \ldots, m_n, M_1, \ldots, M_n\right), \max\left(m_{1,} \ldots, m_n, M_1, \ldots, M_n\right)\right] \tag{10}$$

Design Space Size: given a design space DS, it returns the product of the parametric ranges' sizes (Equation 11a). The size is defined by the minimum m_i and maximum M_i values and the step s_i (by default, the step is 1) of an interval (Equation 11b), or by the number of elements in the range (for categorical parameters). Unlike the aforementioned mechanisms, this one is not meant for navigation, but for the analysis of a given design space.

$$|DS| = size_1 \cdot \ldots \cdot size_n \tag{11a}$$

$$size_i = 1 + \frac{(M_i - m_i)}{s_i} \tag{11b}$$

The mechanisms described in this section were implemented in a tool called *Navigator*. The parameters and respective values and ranges are stored in spreadsheets, a common practice in parametric design.[3] Because *Navigator* can work separately from generative design tools, it can be applied to any set of data in the format of parameters, values, and ranges. One advantage of this approach is that we can generate spreadsheets without generating the design. These files can then be used to create designs in any generative design tool that can process spreadsheet-like files.

Navigator allows designs to be exported and imported. Thus, designers can save a design at any point in the process, use existing designs as templates for further exploration, and easily share their designs. With these mechanisms, users can design from scratch or from existing designs or design spaces. Moreover, although the implemented mechanisms are based on simple straightforward methods, the user can complement them with more complex ones. In the following section, we will see how *Navigator* can be applied to generate designs of towers and chairs.

Evaluation

This section addresses the evaluation of the *Navigator* tool with two case studies. The first is a low-dimensional parametric-based system that generates designs of towers, while the second is a high-dimensional grammar-based system that generates designs of chairs. These cases illustrate the *Navigator*'s ability to suit different generative systems, with different design domains and complexity levels.

Towers

The first case study is a parametric-based system that generates designs representing a simplified facade of superellipse-sectioned towers. The GDS is defined by seven numerical parameters: height, floor height, twist angle, taper ratio, base width, base length, and squareness. The superellipse section is defined by the width, length, and squareness, and can assume diverse shapes, such as: a star (when squareness < 1), a rhombus (when squareness $= 1$), an ellipse (when squareness $= 2$), a rounded rectangle (when squareness $>$ 2), and a rectangle (as squareness approaches infinity). The taper ratio (varying between 0 and 1) and the twist angle (given in radians) define how much the tower narrows and rotates towards the top, respectively.

This GDS is implemented in Khepri, a portable text-based AD tool.[18] Khepri scales better with design complexity (when compared to visual-based AD tools) and can generate, evaluate, and optimize designs in several modeling, analysis, and optimization tools from a single algorithmic description. In this case, *Navigator* mechanisms were used in Khepri's environment to read and write CSV spreadsheets and generate designs in a modeling tool. **Figure 2** shows four towers generated by the manual manipulation of the squareness parameter (from left to right: 0.5, 1, 2, and 6) and twist angle (from left to right: 0, $\pi/4$, $\pi/2$, and π).

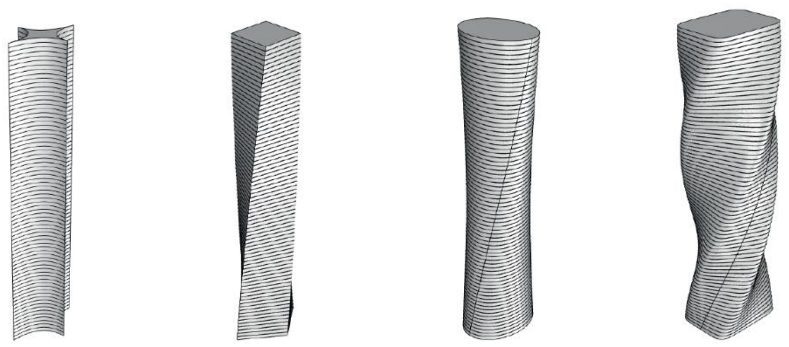

Figure 2. User-defined towers (left to right: squareness = 0.5, 1, 2, and 6, and twist angle = 0, π/4, π/2, and π).

The *Navigator*'s *Random Design* mechanism was used to generate the five random towers presented in **Figure 3**. The image shows, from left to right, towers with shapes resembling a twisted star prism, a conical spiral, a twisted cone, a double helix, and a pyramid. These designs reveal the variety of shapes we can obtain even with a relatively small design space.

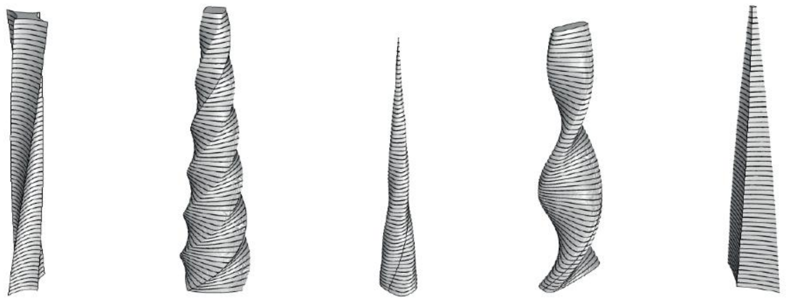

Figure 3. Randomly generated towers.

Figure 4 shows one user-defined design (on the left) and three similar designs generated with *Navigator*'s *Similar Design*. The similarity of the designs in relation to the source design on the left is decreasing from left to right, while the weights are increasing (respectively, 0.1, 0.5, and 1). One can see that the design whose weight is 0.1 is very similar to the original, while the design whose weight is 1 looks very dissimilar.

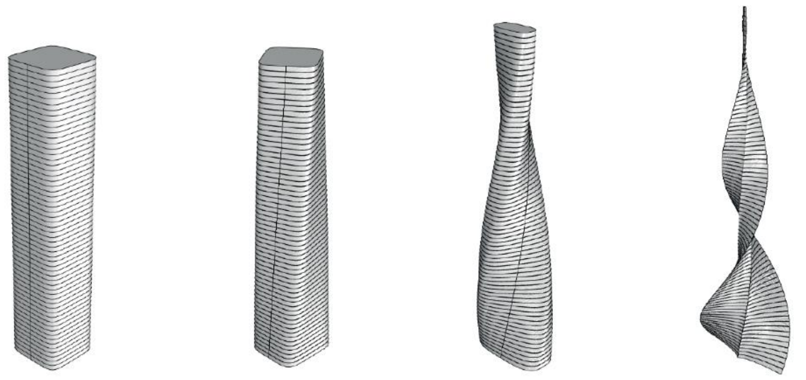

Figure 4. User-defined tower (on the left) and three similar towers (weights, from left to right: 0.1, 0.5, and 1).

Navigator's *Hybrid Design* allows us to make interpolations between two designs. This is illustrated in **Figure 5**, which shows us two user-defined designs (the leftmost and the rightmost), and two hybrid designs in between (from left to right, with a weight of 0.6 and 0.3 in relation to the leftmost design).

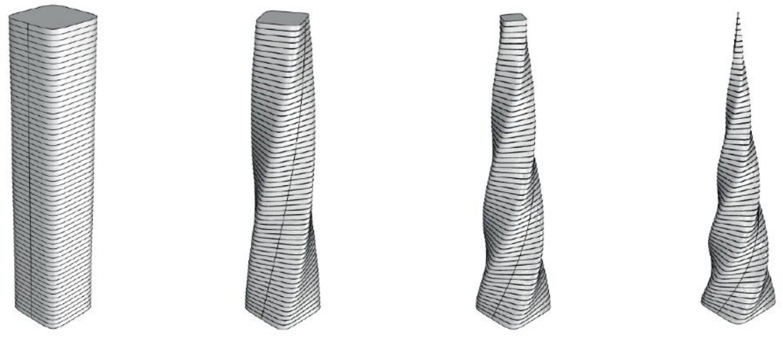

Figure 5. User-defined towers (the left and the rightmost) and hybrid towers in between (with weights of 0.6 and 0.3, from left to right).

Navigator can also generate collections of designs. For instance, **Figure 2** displays a collection of towers that can be generated with *Navigator*'s *Design Collection* mechanism. **Figure 6** shows a matrix of towers generated with *Navigator*'s *Design Matrix*. This set of designs presents two parameter values varying incrementally: the squareness varies from 1 to 4 from left to right and the twist angle varies from π/4 to 3π/4 from front to back.

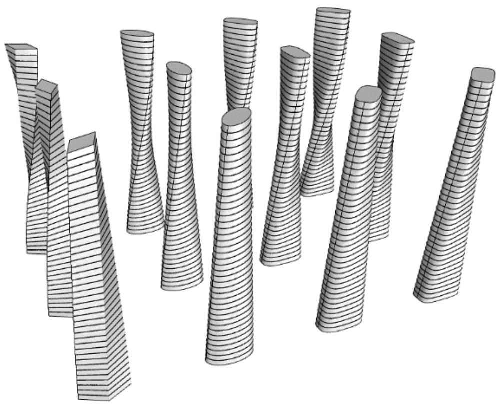

Figure 6. Matrix of towers (left to right: squareness = 1, 2, 3, and 4, and front to back: twist angle = π/4, π/2, and 3π/4).

We are now going to look at the generation of subspaces. Design spaces can be restricted to subspaces by limiting ranges, deleting parameters, or make an independent parameter a dependent one (e.g., to constrain the GDS to squared towers, the width must be equal to the length). In this case, we have defined two subspaces with the *Navigator*'s *Default Design Space*, that characterize two types of towers: twisted towers (with taper ratio = 1) and tapered towers (with twist angle = 0). **Figure 7** and **Figure 8** show four towers from each type, being the first three on the left user defined and the one on the right randomly generated.

In **Figure 7**, the tower on the left is inspired by the Zeidler's Al Majdoul Tower, the second is a replica of the first but with a rectangular section, and the third has four times more twist than the first one. The fourth is a randomly generated twisted tower.

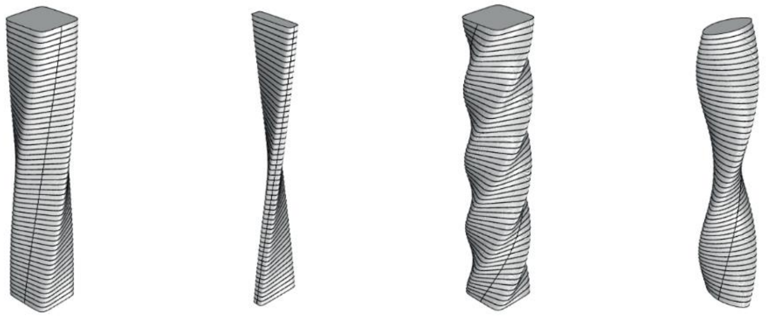

Figure 7. Twisted towers: three user defined and one randomly generated (from left to right).

From left to right in **Figure 8**, the first tower is inspired by the Skidmore, Owings & Merrill's John Hancock Center, the second is a variation of the first (changing the taper ratio from 0.6 to 0), the third was inspired by the Foster + Partners' Millennium Tower, and the fourth is a random tapered tower.

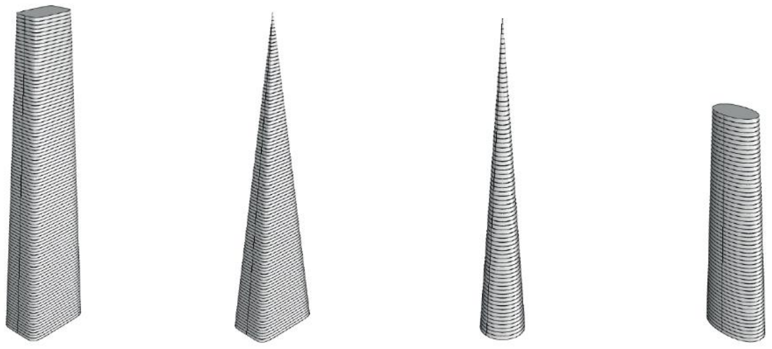

Figure 8. Tapered towers: three user defined and one randomly generated (from left to right).

From the two subspaces characterizing twisted and tapered towers, similar and hybrid spaces were created with *Navigator*'s *Similar* and *Hybrid Design Space*, respectively. **Figure 9** illustrates four towers, where the first two belong to the similar space and the last two belong to the hybrid space. For each pair of towers of one space, there is one that was manually defined and another that was randomly generated. We can observe that, as expected, the similar space is far more restricted than the hybrid space, as it can only generate straight towers (with twist angle = 0 and taper ratio = 1). Contrariwise, the hybrid space can generate a variety of tapered twisted towers.

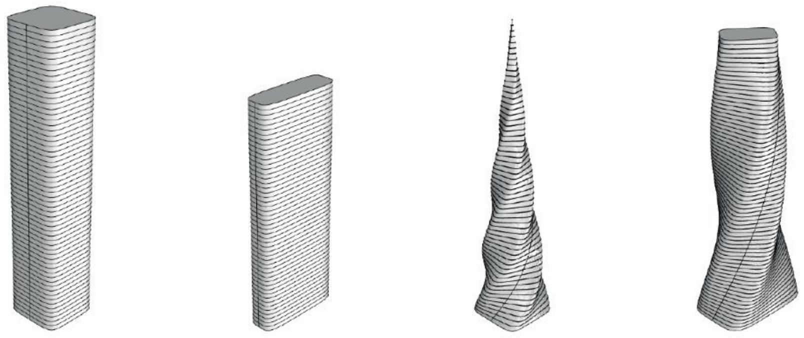

Figure 9. From left to right: similar user-defined tower, similar randomly generated tower, hybrid user-defined tower, and hybrid randomly generated tower.

Chairs

The second case study considers the [*removed for blind review*] design tool.[8] It provides a good case scenario for *Navigator* as it has a high-dimensional design space, with 162 parameters (both numerical and categorical). The generation is made by adding or deleting chair components (legs, seat, back, stretchers, base, and arms) and editing their dimensions and positions (e.g., seat width, depth, height, tilt angle, front and rear radii, and taper width). The design space is characterized by a wide diversity of bilaterally symmetric chairs, restricted to standard anthropometric dimensions. The designs comprise a simplified representation – they only address lines and arcs, planar surfaces, and constant thickness (except the chair's legs that can be tapered).

The chair design tool is based on a parametric set grammar. It presents an interface mostly composed of checkboxes and sliders whose manipulation edits a design represented in a modeling tool. Moreover, the tool can export and import designs from Excel spreadsheets. Users can create a design from scratch (adding elements step by step), from editing a design (of a library of predefined chairs or their own chairs), or from random generation. In this case, *Navigator* was used to read and write CSV spreadsheets that were converted to Excel spreadsheets to be read by [*removed for blind review*].

This GDS was devised from a corpus of 26 iconic modern chairs.[8] **Figure 10** shows four of such designs, from left to right: Eero Saarinen's Tulip, Mart Stam's S33, Arne Jacobsen's Ant, and Ernest Race's Antelope chair. The figure illustrates the ability of the system to generate a wide variety of chair types, namely: one- to four-legged chairs, arm and armless chairs, rounded and squared seats, and solid and splat back, among others.

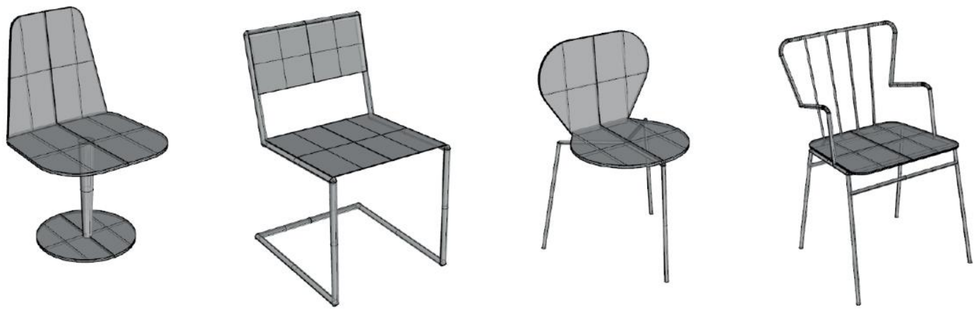

Figure 10. User-defined chairs inspired by existing designs (from left to right: Tulip, S33, Ant, and Antelope chairs).

Users can also produce random designs. **Figure 11** shows four designs generated with *Navigator*'s *Random Design*. In this case, the random mode was restricted to *valid* chairs, i.e., to designs with at least one leg, one seat, and one backrest element. This prevents the system to generate stools or incomplete designs but does not avoid random designs to look extravagant and unrealistic. Nevertheless, such unconventional designs turned out to be interesting for designers, who regarded them as a starting point for new ideas.[8] Also note that, although there is no ambiguity in the system, results have proven to be vehicles of unpredictable emergent meanings, such as the 'wheel' in the leftmost design of **Figure 11**. Yet, more feasible randomly generated designs can be obtained by restricting the design space, as we will see later.

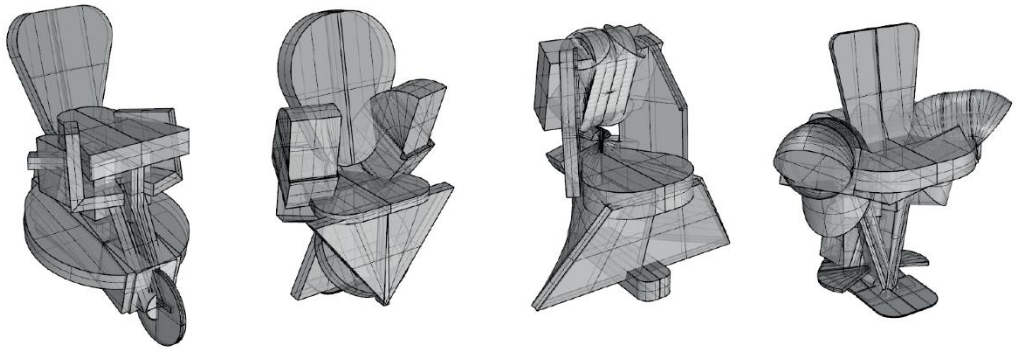

Figure 11. Randomly generated chairs.

Designs generated with the Navigator's *Similar Design* mechanism can be either very similar or very dissimilar to a given design. **Figure 12** shows the Ant chair (on the left) and three similar designs (the three rightmost). From left to right, the designs gradually become less similar to the Ant chair, as increasing weights are applied (0.1, 0.4, and 0.8, respectively). The rightmost design is close to the border of the design space, i.e., their parametric values are close to the limits of the parametric ranges. This design illustrates how broad the design space is.

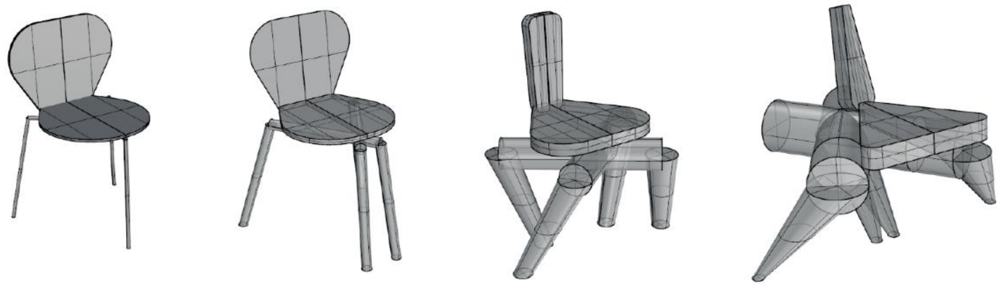
Figure 12. Ant chair (on the left) and three similar chairs (weights from left to right: 0.1, 0.4, 0.8).

Hybrid designs capture characteristics of two or more designs. **Figure 13** shows two chairs from the corpus: Thonet 214 chair and Ernest Race's Antelope chair (the leftmost and the rightmost, respectively). In the middle, we have two hybrid designs generated with the two different methods available in *Navigator*'s *Hybrid Design*: the first uses the mean method (with a weight of 0.5), and the second uses the random method.

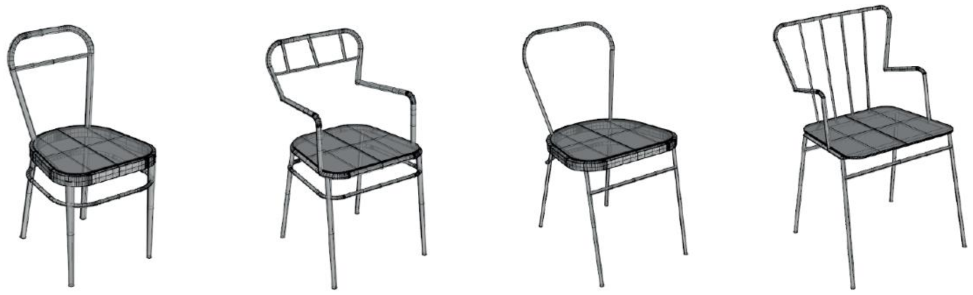
Figure 13. Hybrid chairs. From left to right: Thonet chair, hybrid using mean method, hybrid using random method, and Antelope chair.

Navigator's mechanisms can also generate collections. This is a particularly useful feature in furniture design as it often comprises designs of different types that share the same style. *Navigator*'s *Design Collection* was used to generate the Antelope collection, composed of an armchair, a chair, a stool, and a table (**Figure 14**, from left to right). The leftmost design is inspired by the Antelope chair, and the others were obtained from slight parameter changes in relation to their predecessor: remove the arms (to obtain the chair), remove the back and eliminate the seat tilt and taper (to obtain the stool); and increase the seat width and decrease the seat height (to obtain the table).

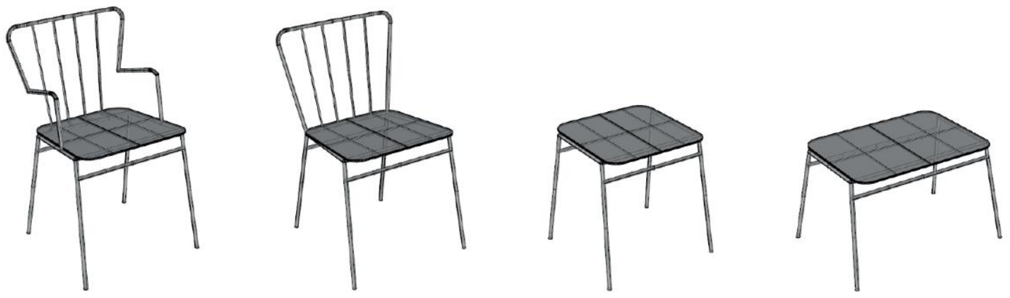

Figure 14. Antelope collection (from left to right: armchair, chair, stool, and table).

The design space can be restricted according to different criteria, such as: usability (e.g., omit parameters that are less frequently used), function (e.g., discard unstable chairs), type (e.g., consider armchairs only), manufacturing (e.g., constrain thickness to that of standard wood panels), aesthetics (e.g., address golden proportions), style (e.g., chairs from one designer), and dimensions (e.g., restrict to given anthropometric data). Conversely, the design space can be expanded (e.g., to include lounge chairs).

In this case, we restricted the design space to the styles of two renowned designers: the Portuguese designer Daciano da Costa (1930-2005) and the English designer Jasper Morrison (1959-). The styles were automatically created from a corpus of five wooden frame chairs from the named designers using *Navigator*'s *Default Design Space*. Beyond the corpus of designs, the design space also includes other existing designs from the same style and original designs within the style.[17]

Figure 15 shows four designs of Daciano's style: the three on the left were inspired by Daciano's Alvor (coffee shop), Palace, and Penta (restaurant) chairs, while the rightmost is a randomly generated design, arguably of the same style. **Figure 16** shows four designs of Jasper's style: the first three, from left to right, were inspired by Jasper's Basel, Bac, and Lightwood chairs, while the fourth was randomly generated. As one can see, the random designs in **Figure 15** and **Figure 16** look much more realistic than the ones in **Figure 11**, since the design space is much more restricted. Nevertheless, we can further fine-tune the spaces (e.g., the rightmost chair in **Figure 15** could benefit from a restriction in the orientation of the base rails to match that of the arms).

The distinguishing features of each style can be externalized by analyzing their parametric ranges (**Table 2**). On the one hand, Daciano's style is characterized by rigid straight lines. Chairs have a square sectioned frame, three or four vertical legs, a rectangular or semi-circular seat, no stretchers connecting the legs, and optional base and arms. On the other hand, Jasper's style is defined by softer and rounder shapes (which is reflected on the size of the design space – $1 \cdot 10^{12}$ times bigger than the Daciano space). Chairs have a square or round frame, four slightly inclined legs, a rectangular or rounded seat, and can optionally have stretchers and arms, but no base rails. Note that, although this analysis may provide a good starting point for style characterization, we must consider that the low level of shape detail (which disregards, e.g., the legs' curvatures), the small number of exemplar designs, and the lack of an expert's evaluation prevents this characterization from being more robust.

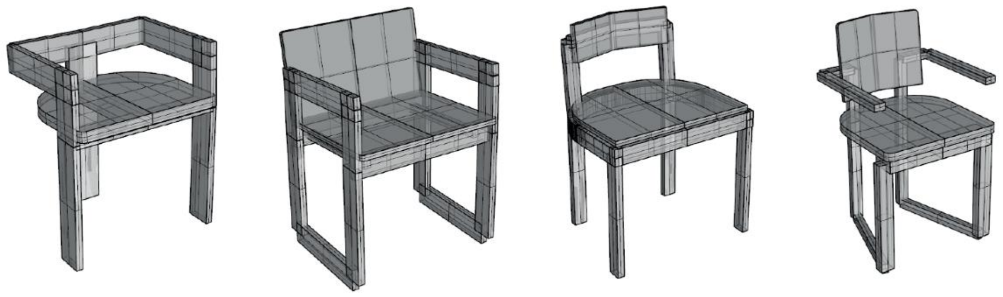

Figure 15. Daciano chairs: Alvor, Palace, Penta, and an in-style randomly generated chair.

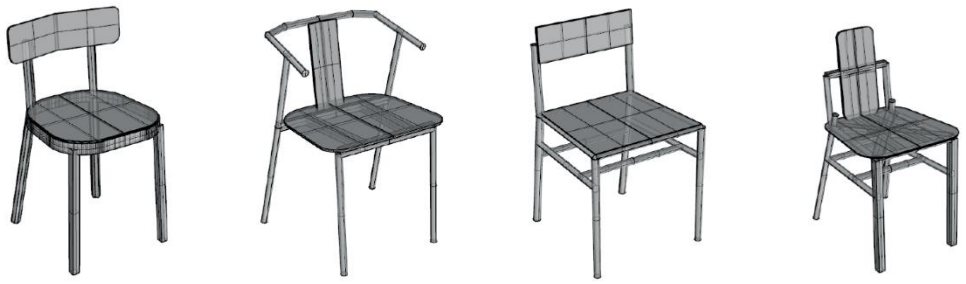

Figure 16. Jasper chairs: Basel, Bac, Lightwood, and an in-style randomly generated chair.

Similar and hybrid styles were generated from the styles of Daciano and Jasper, using *Navigator*'s *Similar* and *Hybrid Design Space* (**Table 2**). **Figure 17** shows two chairs from the similar style (left) and two chairs from the hybrid style (right). The similar design space is much more restricted than the hybrid design space ($3 \cdot 10^{53}$ times smaller), as it only allows the generation of square-sectioned chairs with four vertical legs, with no stretchers nor base. Contrariwise, although the hybrid style is restricted to wooden frame chairs, it can generate designs from more simple and rectilinear lines to more complex and curvilinear ones.

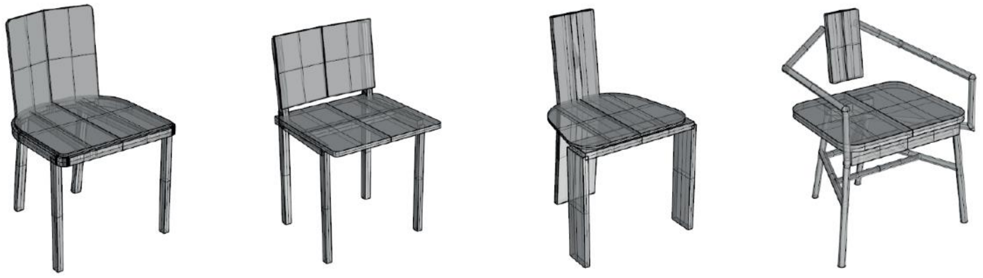

Figure 17. From left to right: similar user-defined chair, similar randomly generated chair, hybrid user-defined chair, and hybrid randomly generated chair.

Table 2. Some parameters and ranges of Daciano, Jasper, Similar, and Hybrid Design Spaces

	Daciano Space	**Jasper Space**	**Similar Space**	**Hybrid Space**
Frame section	square	round	square	{round, square}
Legs number	[3, 4]	4	4	[3, 4]
Leg splat angle	0	[0, 15]	0	[0, 15]
Leg rake angle	0	[15, 33]	[0, 15]	[0, 33]
Seat front radius	[0, 12]	[0, 60]	[0, 12]	[0, 60]
Sear rear radius	[0, 100]	[0, 80]	[0, 80]	[0, 100]
Stretchers	false	{false, true}	false	{false, true}
Base	{false, true}	false	false	{false, true}
Arms	{false, true}	{false, true}	{false, true}	{false, true}

Conclusion

Generative design allows us to quickly generate design alternatives. This benefits designers' creativity and can lead to better design solutions. However, design alternatives are usually overwhelming, and designers end up exploring a very small area of the design space. This is mitigated with exploration mechanisms, which guide the designer throughout the design space to unexplored and potentially interesting areas.

Generative design tools provide a good amount of black-box search mechanisms but provide less support for the more fast, interactive, and comprehensible white-box ones. Given this drawback, we implemented a compendium of well-known white-box mechanisms in the *Navigator* tool to find designs and design subspaces. Those mechanisms can generate default, random, similar, and hybrid designs and design spaces, design collections, and design matrices. The tool also provides a mechanism to compute the size of a design space. *Navigator* can be applied to any generative system; in this paper, it was illustrated with the generation of designs of towers and chairs.

Navigator promotes an engaging and playful experience in exploring design spaces. Designers can test different ways of navigation using randomness or existing designs while maintaining control of the process. Such exploration can aid users in finding areas of the design space that they probably would not find otherwise, which can lead to new, unforeseen, and exciting ideas. We have seen that meaningful alternatives do not need to be necessarily reliable or coherent, as they can lead at some point to new directions for further explorations. *Navigator*'s mechanisms also proved to be valuable for fine-tuning generative systems (e.g., increasing the parameter's range step to obtain higher geometric distinctions).

For future work, we intend to develop a user-friendly graphical interface for the manipulation of these mechanisms and for the simultaneous visualization of multiple design alternatives. We will then perform user tests with designers, to assess whether these mechanisms help them in their design process. Moreover, we plan to complement *Navigator* with some black-box mechanisms, such as machine learning guided exploration, to filter meaningful solutions for the designer.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

This work was supported by [removed for blind review], under projects [removed for blind review].

References

1. Sheikholeslami M. Design Space Exploration. In: *Elements of Parametric Design*. Abingdon, UK: Routledge, 2010, pp. 275–287.
2. Matejka J, Glueck M, Bradner E, et al. Dream Lens: Exploration and Visualization of Large-Scale Generative Design Dataset. In: *Proceedings of the 2018 CHI Conference*. 2018, pp. 1–12.
3. Woodbury R, Burrow A. Whither design space? *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 2006; 20: 63–82.
4. Erhan H, Wang IY, Shireen N. Harnessing Design Space: A Similarity-Based Exploration Method for Generative Design. *International Journal of Architectural Computing* 2015; 12: 217–236.
5. Mohiuddin A, Woodbury R. Interactive Visualization for Design Dialog. In: *Design Computing and Cognition '20*. 2021, pp. 507–526.
6. Castro e Costa E, Jorge J, Knochel AD, et al. Enabling parametric design space exploration by non-designers. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 2020; 34: 160–175.
7. Autodesk. Autodesk Project Refinery, http://www.autodesk.com/campaigns/refinery-beta (2016, accessed 10 February 2021).
8. Removed for blind review.
9. Lawson B. *How Designers Think: The Design Process Demystified*. 4th ed. Oxford, Burlington, MA: Elsevier/Architectural Press, 2005.
10. Bidgoli A, Veloso P. DeepCloud: The Application of a Data-driven, Generative Model in Design. In: *ACADIA 2018: Recalibration. On imprecision and infidelity*. 2018, pp. 176–185.
11. Roman M. Four Chairs and All the Others - Eigenchair: Data driven design. In: *Computation and Performance – Proceedings of the 31st eCAADe Conference*. 2013, pp. 405–414.
12. Sanchez M, Fryazinov O, Vilbrandt T, et al. Morphological Shape Generation through User-Controlled Group Metamorphosis. *Computers & Graphics* 2013; 37: 620–627.
13. Stiny G, Mitchell WJ. The grammar of paradise: on the generation of Mughul gardens. *Environment and Planning B* 1980; 7: 209–226.
14. Pugliese M, Cagan J. Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar. *Research in Engineering Design* 2002; 13: 139–156.
15. Chase S, Ahmad S. Grammar Transformations: Using Composite Grammars to Understand Hybridity in Design, With an Example from Medieval Islamic Courtyard Buildings. In: *Learning from the Past a Foundation for the Future (Special publication of papers presented at the CAAD futures 2005 conference)*. 2005, pp. 89–98.
16. Orsborn S, Cagan J, Pawlicki R, et al. Creating cross-over vehicles: Defining and combining vehicle classes using shape grammars. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 2006; 20: 217–246.
17. Removed for blind review.
18. Leitão A, Castelo-Branco R, Santos G. Game of Renders: The Use of Game Engines for Architectural Visualization. In: *Proceedings of the 24th CAADRIA Conference*. 2019, pp. 655–664.