Dream Lens: Exploration and Visualization of Large-Scale Generative Design Datasets

Justin Matejka, Michael Glueck, Erin Bradner, Ali Hashemi, Tovi Grossman, and George Fitzmaurice
Autodesk Research, Toronto Ontario Canada
{*first.last*}@autodesk.com

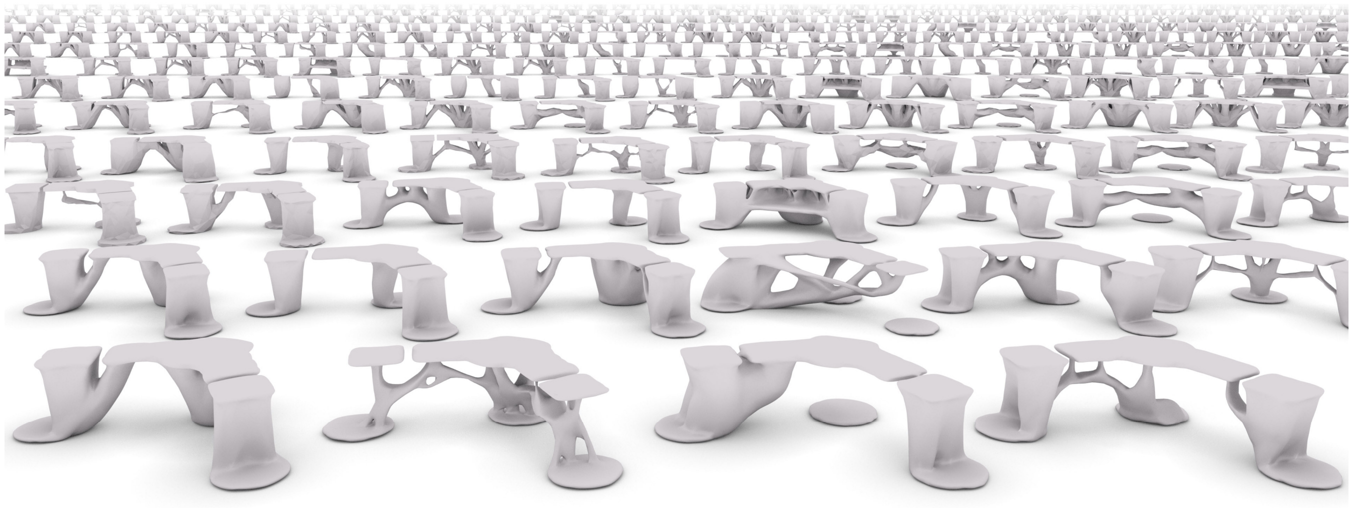

Figure 1. Conceptual illustration of a collection of design variations for a single task: lifting a computer monitor 80mm off a desk.

ABSTRACT
This paper presents *Dream Lens*, an interactive visual analysis tool for exploring and visualizing large-scale generative design datasets. Unlike traditional computer aided design, where users create a single model, with generative design, users specify high-level goals and constraints, and the system automatically generates hundreds or thousands of candidates all meeting the design criteria. Once a large collection of design variations is created, the designer is left with the task of finding the design, or set of designs, which best meets their requirements. This is a complicated task which could require analyzing the structural characteristics and visual aesthetics of the designs. Two studies are conducted which demonstrate the usability and usefulness of the *Dream Lens* system, and a generatively designed dataset of 16,800 designs for a sample design problem is described and publicly released to encourage advancement in this area.

INTRODUCTION
Over the past 20 years, 3D modelling and CAD tools have seen major advancements in the functionality which they offer and the complexity of designs which they can create. Despite these advancements, the overall approach and interactive workflow used to design geometry has undergone little change: users leverage a set of primary creation and editing tools to build towards a single candidate design – a somewhat bottom-up approach.

However, with ever increasing computing power and new simulation methods, a relatively new technique of "generative design" has been introduced [27]. With this top-down approach, the designer specifies high-level goals and constraints to the system, and allows the system to automatically generate geometry meeting those goals. Using this emerging computational workflow, a designer is no longer constrained to creating a single design solution. By varying the goals, constraints, or algorithm parameters, a *divergent* generative design system can create many solutions, and this technique has been applied to problems as diverse as creating lightweight airplane partitions [38], designing furniture [39], and optimizing office layouts [25] (Figure 2). With high-performance computing and cloud services, this process can be massively parallelized, allowing such systems to generate thousands of design alternatives.

With this additional power and capability, comes the daunting task for the user to navigate through the candidate designs and find a single or set of suitable designs for their needs. This is a complicated task as it could require analyzing structural characteristics of the designs, as well as examining their visual aesthetics. Few previous systems have been developed to help visualize and explore design alternatives

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5620-6/18/04…$15.00
https://doi.org/10.1145/3173574.3173943

for geometry [11, 12, 23], and most of those were designed to support only a small number of designs.

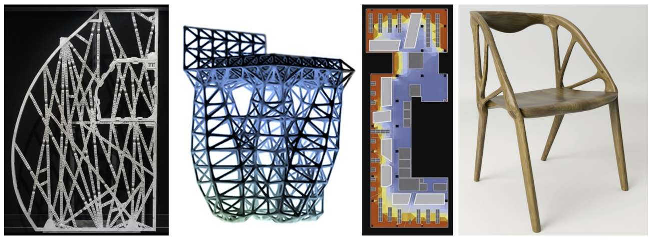
Figure 2. A selection of objects using a *divergent* generative design approach. From left to right: airplane partition, truss-based chair, office layout, and the Elbo chair.

In this paper, we present an interactive visual analysis tool that helps designers explore a large space of 3D design solutions. Building upon a prior framework for exploring large collections of video data [24], the system allows users to inspect both the visual appearance of the geometric models, as well as the associated metadata describing their physical properties. In addition to visually exploring the solution set, users can rank designs using an example based approach, to help narrow the results to designs with the most desirable features, and view attribute examples, to help understand the impact of each attribute within the metadata.

Our work contributes a system for exploring large collections of 3D models that have subtle variations in their aesthetics and physical properties, and we make our extensive dataset publicly available to encourage further work in this area. Initial feedback from two user studies indicates the potential benefits of the system and the overall excitement around the concept of visualizing generative design solutions.

RELATED WORK

Dataset Exploration

The Video Lens framework [24] and associated Baseball Video Lens system were developed for searching and filtering through large collections of video and associated metadata. The Dream Lens system builds upon the Video Lens framework, by extending the functionality of existing components, and introducing new components and interactions specifically for the domain of exploring a large collection of generatively designed 3D objects.

There are a large number of research projects dedicated to the general task of multi-dimensional dataset exploration [15, 19, 28] over a large set of domains including publications [21], online shopping [32], and image search [8, 33]. The VisOpt Slider [20] looks at a similar task of design exploration, and presents a unique approach to encoding the parameter space of an attribute directly on a slider widget.

Interaxis [18] looks at the difficult problem of mapping multi-dimensional data onto a two-dimensional scatter plot. They address this by letting users drag examples to the extreme ends of the scatter plot to define how each access should behave. Our work uses a similar "by example" approach to defining rankings based on user preferences.

Design Optimization and Generative Design

While generative design [4] is still a relatively new and emerging area, there have been a number of projects looking into the areas of generative design and design synthesis.

GEM-NI is a graph-based generative design tool allowing designers to explore multiple design alternatives in parallel [37]. While focused on direct-manipulation tools, the system also has a "Design Gallery" for parameter range exploration, although it is limited to viewing several dozen designs. Ulu and Kara [31] present an algorithm for generatively creating structures to support existing objects. The resulting designs are created by specifying points on the object and ground where the support structure should connect. This type of design problem is similar to the "monitor stand" example problem described in this work. However while their work focuses on the algorithmic details of the solver and provides limited support for creating design variations, our work uses an existing solver, and looks at the problem of filtering through a large set of variations.

Recent work by Doraiswamy et al. [13], and Ashour & Kolarevic [3] have looked at generatively producing building designs considering properties such as "the view" and daylight. These papers note that aesthetic considerations are often ignored in optimization models [7, 14, 30], yet are influential in the design process. To overcome this, they build aesthetic characteristics into the simulation model. Our approach for aesthetic considerations is to dedicate a large portion of the interface to viewing the design geometry.

Design Variations and Comparisons

A number of systems have looked at how to compare multiple 3D models, including 3D Diff [12] and Mesh Git [11], or design results when varying an input parameter space [23, 29]. However those systems are designed for comparing two or three 3D models, or several dozen graphical variations, and do not look into the problem of comparing larger collections. Lienhard et al. [22] create "thumbnail galleries" for a set of procedurally generated models. These galleries are very effective for showing a range of possible designs, but do not support visualizing the parameters.

Systems such as ShapeSynth [5], Fit and Diverse [35], work by Alhashim et al. [1], Jain et al. [16], and Kalogerakis et al. [17] have the ability to produce a large collection of design variations given a relatively small set of examples as input. Using a different approach, Yumer et al. [36] developed a system to procedurally generate thousands of high-quality models using autoencoder networks. However, these systems have relatively limited support for interactively exploring the collection of generated models.

THE DATASET

This section describes the process used to create a large collection of generatively designed solutions to a single design problem. Data sets of this type are non-trivial to set up, and time consuming to generate. We believe the dissemination of high-quality generatively designed datasets is important for advancing research in this emerging area, and are unaware of any such publicly available data. To this end, the full dataset is available for download (~58GB) [1].

To create a collection of design variations, we used Project Dreamcatcher [40], an internally developed experimental platform for producing generative design solutions to engineering design problems. The system allows a user to input the requirements and constraints in the form and structural loads of their design, rather than designing the solution manually. Multiple shape and topology optimization algorithms are employed to synthesize model geometries that optimally satisfy these criteria [2, 6, 34].

Problem Definition

We wanted to test the *Dream Lens* system with a broad range of designers, and given the time requirements for creating large sets of design solutions, we wanted to do this with a single dataset. To accomplish this, we needed a design problem and related dataset that was easy to understand without any specialized domain knowledge, could be used by a broad range of people, and was a problem that could be satisfied by a wide variety of designs options.

The design task we came up with which satisfies the above criteria is to create a 3D printed "monitor stand" to raise a monitor off the surface of a desk (Figure 2). Specifically, the stand is customized for the base of a Dell UltraSharp 2407 monitor and needs to raise the base 80mm above the tabletop.

Using a traditional CAD approach, we would design the stand itself, and then perhaps run a simulation to ensure that our design could support the required weight. However, to approach this problem from a generative design/shape synthesis perspective, we instead provide a set of input geometries, constraints, and forces – and allow the simulation engine to generate a design.

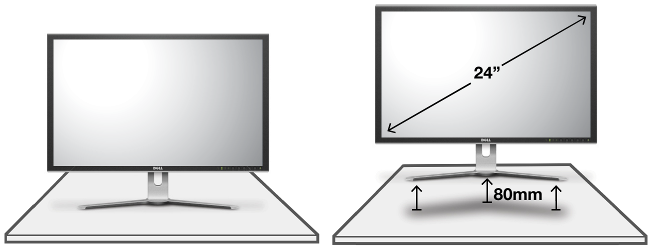

Figure 3. Sample design task, raising a computer monitor off the surface of a desk.

In this case, the designer supplied geometry consists of three disks at the bottom which represent the "feet" where the monitor stand will contact the desk, and three flat sections placed 80mm above the desk which trace the shape of the monitor's base. Additionally, the tabletop is modelled as an "obstacle" to prevent the generated geometry from extending below the desk's surface (Figure 4).

[1] *www.autodeskresearch.com/publications/dreamlens*

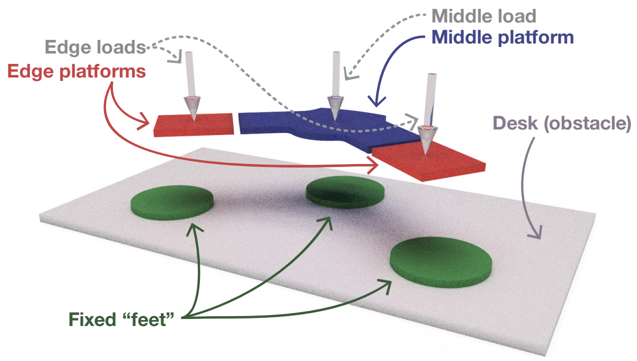

Figure 4. Problem definition describing the locations of the feet, platforms, and desk surface geometry, and the position and direction of the static forces.

The static weight of the monitor is 8.3kg, so the platforms need to support at least 8.3kg × 9.8N/kg = 81.3N of force. To account for additional loads (when placing the monitor on the stand, for example), the stand should be able to support a load of at least 200N, and the ability to support heavier loads is desirable as well. The weight of the monitor is modeled as two independent forces: a *middle load* on the middle platform, and an *outer load* distributed evenly to the two outside platforms (Figure 4).

An example design solution to this problem with *middle* and *outer loads* of 500N each, is shown in Figure 5. In addition to the geometry of the design, a number of properties are calculated including the static properties of: *Center of Mass (x, y, z), Weight, Overhang Percentage, Surface Area, Area/Volume Ratio*, as well as properties related to the performance under the load conditions: *Maximum Displacement, Max. Strain, Total Strain, Max. Vonmises,* and *Objective Value (for the simulation)*.

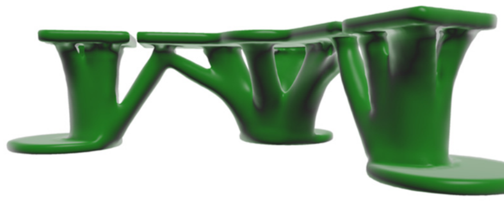

Figure 5. A single design, produced with middle and outer loads of 500N each.

Variation Creation

The previous section describes the general definition used to model the problem, and the process used to generate a single design. This is an example of *convergent* generative design, where the algorithm converges to a single solution. But rather than looking at a single option, we are interested in design difficulties when the computer generates thousands of variations to choose from. We refer to this as *divergent* generative design. In many parametric design systems [3, 13, 36, 37], variations are created by varying some combination of the parameters used for describing the geometry. In our case, rather than varying parameters of the geometry directly, we vary the parameters of the problem *definition* which is fed to the simulation engine, which in turn results in a rich variety of design outputs.

There are four parameters in the problem definition and solver configuration which we choose to vary (with the parameter values listed in brackets):

Middle Load: The load directed downward on the middle section of the platform. [100N, 250N, 500N, 1000N, 2000N].

Outer Load: The load directed downward on the outer sections of the platform, set independently of the *middle load.* [100N, 250N, 500N, 1000N, 2000N].

Voxel Size: A parameter of the solver which sets how big the voxels (the 3D version of a 2D pixel) will be when performing calculations. [1.5mm, 3.0mm, 4.5mm].

Mu: Another parameter of the solver, which indicates a constant value for volume minimization within the solver. [-0.000008, -.0.000004, -0.000002].

Finally, since topology optimization is an *iterative* process, a new design is created at each iteration, and the result at any intermediate iteration represents a "valid" design. Set-ups ran for 74 iterations, producing 74 designs each.

Combining these factors leads to 16,800 total designs:

	5	*Middle Loads*
	5	*Edge Loads*
	3	*Voxel Sizes*
	3	*Mu Values*
×	74	*Iterations*
	16,800	Total Designs

Each design took an average of 3.1 minutes to generate, requiring over 36 days of GPU time in total. The system used 10 GPUs in parallel, completing the process in four days.

While all designs in this "full" collection are at least marginally unique, successive iterations from the same run vary only slightly. To minimize the number of very similar designs, a collection with results from six iterations (20, 30, 40, 50, 60, 74) was created. Additionally, "extreme outlier" designs with computed properties more than 4 standard deviations from the mean were removed, resulting in a "filtered" set of 1,242 designs (a number of which are shown in Figure 1). All of the designs meet the criteria of raising the monitor 80mm, yet exhibit a high degree of variation in both visual appearance and physical characteristics.

In the end, our dataset contains the 3D geometry of each design, a static rendering of each design from a consistent viewpoint, and a metadata file describing all the input parameters used, and properties calculated for each design.

THE DREAM LENS SYSTEM

To address the task of searching and filtering through a large collection of design variations, we created the *Dream Lens* system. While the system was developed for exploring sets of 3D model designs created with a generative design process, we believe it would prove useful for any large set of design variations, independent of how they were created.

Design Goals

The development of the *Dream Lens* system was directed by two main design goals:

D1: Highlight both Appearance, and Properties

We have found that users of generative design solutions tend to fall into two broad categories: The "engineer" types, who heavily focus on the simulated properties (such as weight, strain, etc.) of the produced designs, and pay little or no attention to the appearance of the artifact; and the "artist" types, who are less concerned with the model properties, but care greatly about the aesthetics of the generated design. We believe both usage types are valid and important, and want the Dream Lens system to support both types of users (and those in between). Additionally, by emphasizing both the properties and appearance of the designs, perhaps the "engineers" may be encouraged to appreciate the aesthetics a bit more, and the "artists" might better appreciate the physical properties, creating a more well-rounded design.

D2: Encourage Exploration

When introduced to the idea of algorithms producing thousands of candidate designs, some people question the need for variation and would rather the computer "just pick the best one for me". We believe that the real power of a *divergent* generative design system is that it can produce a collection of designs where picking the "best" is a non-trivial task due to opposing factors which need to be reconciled (weight vs. strength, for example), or the aesthetics of the design are important and need to be considered. We feel encouraging and simplifying exploration of the design space is important to make design-choice tasks more successful.

Implementation

Dream Lens is a C# Microsoft WPF application built as an extension to the Video Lens Framework [24]. In order to take the initial framework (which was designed for exploring video collections) and make it suitable for exploring design variations, *Dream Lens* both extends the functionality of components developed in Video Lens, and introduces new components and interactions specifically suited for this new domain. Specifically, *Dream Lens* makes use of Single Attribute Controllers (SAC) and Multi-Attribute Grid (MAG) components from Video Lens (with modest improvements); while the Attribute Example View, Design Viewer (for 2D images and 3D models), Stacked Model View, Interactive Design Tooltip, and Ranking by Example controls were newly developed expressly for this work.

The Interface

The *Dream Lens* interface (Figure 6) is split into two main sections: the *design viewer* on the left, and a collection of tools for visualizing, selecting, and filtering the parameters on the right – giving relatively equal screen space to the appearance and the properties of the designs (D1).

Throughout the interface a constant mapping of "1 dot = 1 design" is followed. That is, each dot in any of the controls in the *property explorer* represents a single design. As the cursor hovers over designs in one control, those designs are highlighted in all the other controls, and selections made in one control are immediately reflected in the others. On startup, the interface is loaded with the 1,242 designs from the "filtered" dataset.

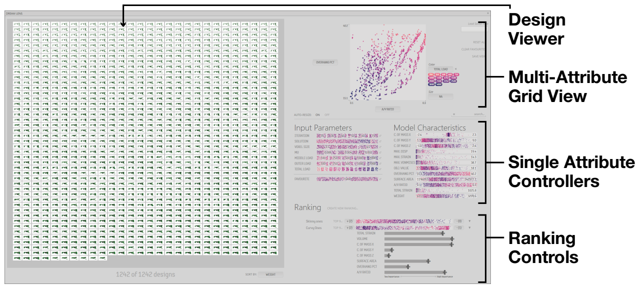

Figure 6. Dream Lens interface.

Single-Attribute Controllers (SAC)

Each attribute from the metadata is represented by a single attribute controller (SAC) [24]. The *SAC* is labeled with the attribute name on the left, and has a strip showing the values of that attribute on the right (Figure 7). The *value strip* contains one dot for each design, showing the distribution of values with the attribute.

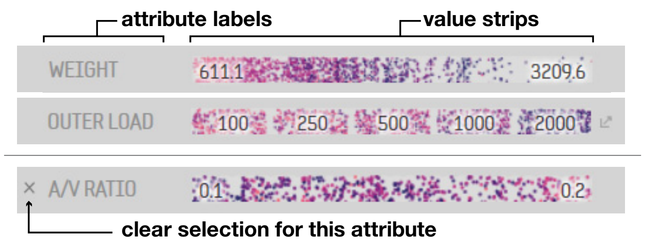

Figure 7. Three example Single-Attribute Controllers. In the bottom one, a selection has been made, as indicated by the 'x' button, which will clear the selection.

Filtering events based on a single attribute is accomplished through clicking or dragging within the value strip. Since the *SACs* are linked, selections made in one *SAC* are immediately reflected in all others.

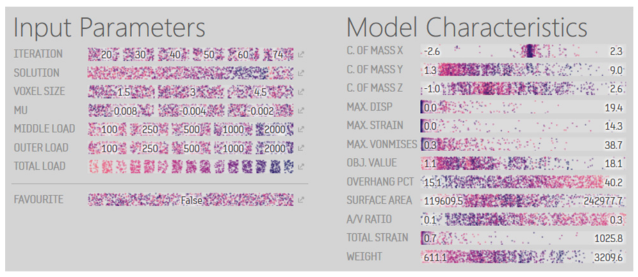

Figure 8. The full collection of single-attribute controls, separated in two columns.

An enhancement to the *SACs* over the implementation described in Video Lens is the colouring of the individual data points. Previously the individual points were rendered in greyscale, here, the points respect the color mapping set in the multi-attribute grid (described later in this section).

The single-attribute controllers are arranged in two columns, with the "Input Parameters" on the left, and the "Model Characteristics" on the right (Figure 8). The *input parameters* column contains data about the setup and parameters used in the creation of each design, and the *model characteristics* columns contains the computed properties of the designs.

Attribute Examples

When exploring a collection of designs, it is useful to see what effect individual parameters have on the design. For example, questions such as, "What does changing the *middle load* do to the design?" and "What does a model with a high *overhang percentage* look like compared to one with a low overhang?" are common during the exploration process. To help answer these questions, right clicking on the attribute launches a large spring-loaded panel showing a set of four examples from the value range for that attribute (Figure 9).

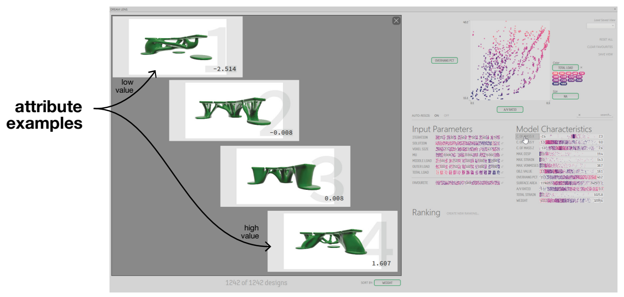

Figure 9. Attribute Example view for *Center of Mass X.*

The representative models are chosen by first calculating four key values from across the range of the variable: namely the minimum value, the maximum value, and the values one and two-thirds between them. Our initial thought was to then select the model closest to each of the four key values, however that model may have unique characteristics not related to the variable in question, thus not serving as a very good "representative" example. Instead, for each of the key values, the 10 closest models are found. Then, the centroid of these models is computed by averaging all of the attributes for these designs, and the most "representative" example of this group is calculated as the design with the lowest root mean square error (RMSE) from the centroid.

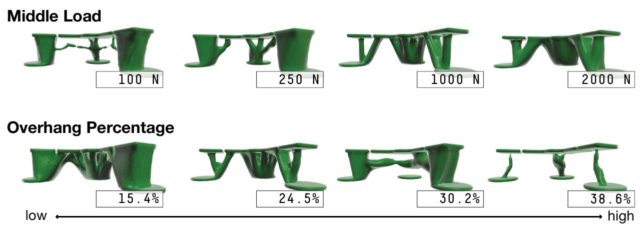

Figure 10. Representative designs for, *Middle Load* and *Overhang Percentage.*

Figure 10 shows the representative models selected and displayed for the *middle load* and *overhang percentage* attributes. We can see that an increased middle load results in a more substantial "center leg", and models with a high overhang leave much of the platform unsupported.

Multi-Attribute Grid (MAG)

The top-right corner of the interface contains the Multi-Attribute Grid (*MAG*) component [24]. This component displays a two-dimensional grid (Figure 11A) with attributes mapped to each of the horizontal and vertical axes (Figure 11B,C). Attributes can also be mapped to the colour and size of the data points (Figure 11D,E). These mappings can be changed by dragging-and-dropping attribute labels from the *SACs* to the respective positions around the *MAG*. By dragging attributes to the different axes/dot controls, we can see the relationship between multiple variables.

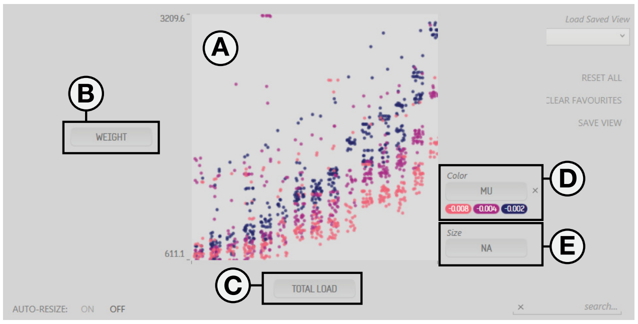

Figure 11. Multi-Attribute Grid (MAG) showing the relationship between Weight and Total Load.

Design Viewer

On the left side of the interface is the *Design Viewer*, for viewing the geometry of the design solutions. On startup, it displays small thumbnail images for all 1,242 designs in the filtered dataset (Figure 12). By default, the designs are sorted by *weight*, with the lightest designs at the top, and the heaviest designs at the bottom. However, the sort order can be changed by dragging-and-dropping a label from a single-attribute controller to the "sort-by" control at the bottom right corner of the design viewer (Figure 13).

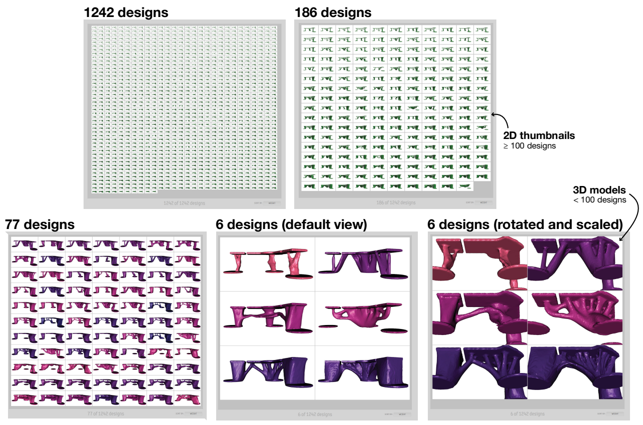

Figure 12. The design viewer with the initial (top left), and filtered views with 1242, 186, 77, and 6 design options.

As the collection of designs is reduced, less thumbnails are displayed, and the images are scaled to fill the space (Figure 12). When displaying less than 100 designs, the thumbnail images are replaced with interactive 3D models. This affords two large advantages over the static 2D images. First, the models can be coloured to respect the colour setting from the multi-attribute grid. Second, the camera position for these models can be manipulated to look at different parts of the model. The camera manipulations performed on one thumbnail are propagated to all models so one can orbit and zoom around all the models in concert, and view all designs from the same viewpoint.

When there are many designs, the images are quite small. To address this, a large tooltip is displayed to show an enlarged image of the design (Figure 13). Pressing the 'f' key when the tooltip is enabled saves the design to the list of "favourites" (as indicated by the star icon).

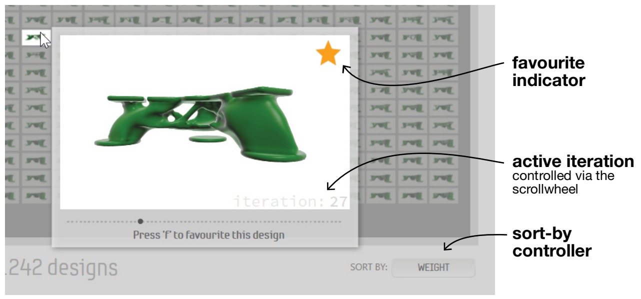

Figure 13. A design tooltip displayed when the cursor is over a model thumbnail.

We found it useful to see what a design looked like a few iterations earlier, or a few iterations later. Using the scroll wheel while the tooltip is active moves through successive iterations of the model in the "full" dataset (Figure 13).

Model Stacking

Besides looking at the models individually in a grid view, we wanted to explore the idea of viewing the models in a shared 3D space. To enable a visualization of this type, we created a "stacked" mode where rather than displaying each 3D model in its own grid cell, their viewports are each enlarged, positioned on top of each other, and composited (Figure 14).

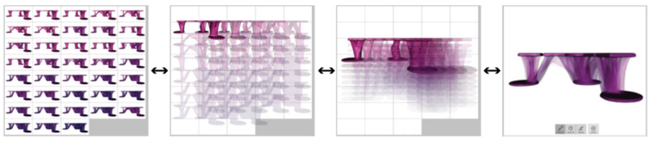

Figure 14. The standard 3D grid view (left), animating to the *stacked* view (right).

With this view, users can directly see the differences in topology or geometry among a small set of models and when stacking a larger number of models it is possible to see regions of high and overlap, in addition to common features among the individual designs (Figure 15).

The desired effect of the stacked view is to have each design contribute equally to each pixel of the display. Conceptually, the colour for each pixel in the display is equal to the sum of all values at that pixel for each individual design, divided by the number of models. Rather than directly manipulating pixel values, we used an alternative (and more performant) method to achieve the same result through standard alpha blending and a particular series of individual layer opacities. Namely, each layer has an opacity of $1/n$, where n is the height of the current layer (i.e., the bottom, or first, layer has an opacity of 1, the second $^1/_2$, the third $^1/_3$, etc.). When composited with default alpha blending [26], the desired effect of each model contributing equally to the composite.

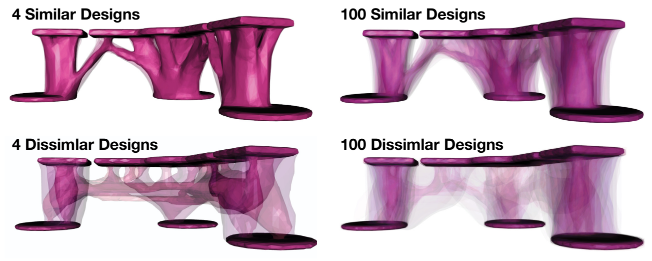

Figure 15. Sample *stacked* views for 4 (left) and 100 (right), similar (top) and dissimilar (bottom) designs.

Using 3D models with 5000 faces each, the standard WPF 3DModelViewer control, and an ATI 1080Ti graphics card, the a model stack of 100 designs can be navigated with an update rate of 15 frames per second.

Interacting with the Stack

Besides being able to navigate the stack of models with pan, zoom, and orbit controls, we implemented three techniques for interacting with the stack to filter the set of selected designs by interacting directly with the model geometry: *chisel*, *select*, and *edge* (Figure 16).

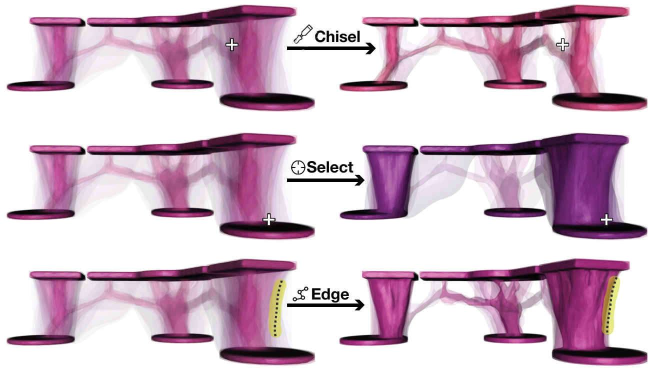

Figure 16. The effects of the chisel, select, and edge tools in the stacked model mode.

The *chisel* tool uses the interaction metaphor of a sculptor "chiseling" away at a block of marble to arrive at the desired design. When clicking on the stack with the chisel tool, all of the designs which have material under the cursor are filtered out. This is accomplished by performing a hit-test on each of the 3D viewports, and removing the models where there is an intersection with the cursor.

The inverse of the chisel tool is the *select* tool, where clicking on the stack of models causes all models which *do not* have material under the cursor to be removed, leaving only those models which *do* have material under the cursor.

The final tool for filtering in the stacked view is the *edge* tool, where the user draws a curve along the stack indicating a desired outline of the design, and only designs matching the requested edge profile are kept. If there is a particular "principle curve" the user would like, they can orient the model stack to where that edge is visible, and directly draw the desired curve to find all matching models.

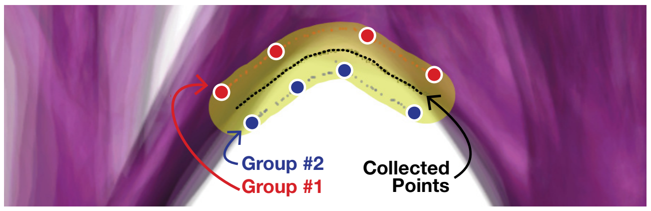

Figure 17. Graphical representation of the edge tool.

The edge tool works by having the user to draw a stroke over the model stack, which is shown as a yellow highlight (Figure 17). The points from this stroke are smoothed, and two strokes offset from the initial stroke at a distance of 30 pixels are calculated. Along these offset strokes, four points are selected, resulting in two "groups" of four points each. The algorithm then performs a hit test at these eight points, and finds models which are a "hit" in all points of one group, but a "miss" in all points of the other group. In this way, the algorithm finds the models which have an edge within the thickness of the highlighted stroke since the models "exist" on one side of the center line, but do not on the other.

Ranking by Example

The final component of the interface is the *ranking panel*. When faced with a large collection of design options, a common desire is to "rank" all of the designs. We have found it is often the case that users do not know exactly what criteria to use in constructing their ranking, but they do know which designs they like, and which ones they do not.

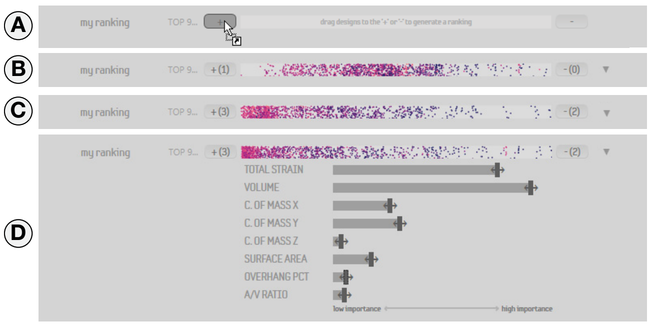

Figure 18. Steps involved in creating a ranking (A-C), and optionally, adjusting the importance weightings of the various attributes (D).

Taking advantage of this fact, we implemented a "ranking by example" interface were users choose some designs they like and (optionally) some designs they dislike, and the system automatically ranks the designs such that those most similar to the “liked” designs and dissimilar from the “disliked” designs ranked the highest (Figure 18).

To select “liked” or “disliked” designs, the user drags the thumbnail for the design onto the ‘+’ or ‘-’ labels respectively (Figure 18A). Once one example has been selected, a strip plot similar to those found in the *single-attribute controllers* appears, ranking all designs from highest-ranked on the left, to lowest-ranked on the right (Figure 18B). Additional examples can be specified to refine the ranking (Figure 18C).

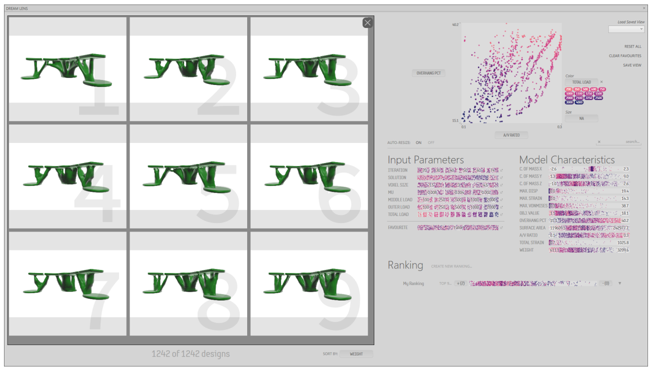

Figure 19. Top Nine view for the specified ranking.

The ranking is determined by analyzing the attributes of the liked and disliked designs (as detailed below). The arrow to the right of the strip plot expands the attribute weighting controls, allowing the weights to be manually adjusted.

Once a ranking has been created, the ranking behaves like the other “standard” *SACs* in terms of hovering and filtering behavior. It can also be dragged to the “sort-by” widget on the design viewer to sort the designs by this ranking, or to any of the axes/dot property controls of the *MAG*. Clicking on the “Top Nine” button allows the user to quickly see the highest ranked designs (Figure 19).

Ranking Algorithm

At a high level, the ranking algorithm works by calculating a score for each candidate design based on its similarity to each of the “liked” designs and dissimilarity from each of the “disliked” designs. By ordering the candidate designs by this score, we can see which designs are ranked the highest.

Specifically, the ranking algorithm takes three sets of designs, the *liked* and *disliked* designs, as well as the full set of all available designs:

$$\texttt{liked} = l_1, l_2, \ldots, l_n$$

$$\texttt{disliked} = d_1, d_2, \ldots, d_n$$

$$\texttt{candidates} = c_1, c_2, \ldots, c_n$$

Besides the lists of designs, we also need a list of attributes to be used for calculating the ranking:

$$\texttt{atts} = a_1, a_2, \ldots, a_k$$

We found using *model* characteristics rather than *input* parameters resulted in rankings which more closely matched user expectations, and used the following eight model characteristics: *center of mass (x,y,z), weight, total strain, surface area, overhang percentage,* and *area to volume ratio*.

Given the above lists, the first step is to calculate the relative “weights” of the individual attributes, that is, compute a metric to determine how important each attribute is to the user’s ranking. We do this by seeing how closely together each of the attributes are within the `liked` and `disliked` sets. For example, if all of the *liked* designs were very similar in *weight*, but not similar in *total strain*, we infer that *weight* is an important factor in the user’s preferences, while *total strain* is not. This is calculated by first normalizing the values for all attributes to the range of [0..1] and by computing the standard error for each attribute (err_a) within each of the *liked* and *disliked* sets. Then, we normalize the standard errors between 0 and 1, and use that as the weighting factor:

$$\texttt{weight}_a = \frac{\max(err) - err_a}{\max(err) - \min(err)}$$

Next, a `likeScore` is calculated for each candidate design by computing its distance from each of the *liked* designs using a RMSE calculation with the difference between the individual attributes scaled by the weighting factor:

$$\texttt{likeScore}_c = \sum_{a \in attr} \left(\sqrt{\frac{\sum_{ld \in \texttt{likes}}^{n_l} \texttt{weight}_a \times \left| \frac{ld(a) - c(a)}{\max(a) - \min(a)} \right|^2}{n_a}} \right) \times \frac{1}{n_l}$$

A similar calculation is used to compute the `dislikeScore`, and then the `totalScore` is calculated as:

$$\texttt{totalScore}_c = \texttt{likeScore}_c + (1 - \texttt{dislikeScore}_c)$$

Since the `likeScore` and `dislikeScore` are normalized between 0 and 1, the total score will be a value between 0 and 2, with lower `totalScores` representing higher rankings.

We tested other algorithms for computing the rankings (such as linear regression), but found our approach gave more “predictable” results and has the desirable properties of producing results even if only one example is provided, and ensuring that if only one design is chosen for each of the *liked* and *disliked* categories, those designs are guaranteed to be at the top and bottom of the ranking, respectively.

Example Tasks

To demonstrate how the Dream Lens system could be used, we present walkthroughs for three sample tasks.

Task 1: Find models most suitable for 3D printing

For many 3D printing technologies, a combination of low weight and low overhang is desirable. To find designs best satisfying these criteria, first drag the *weight* and *overhang pct* properties to the axes of the multi-attribute grid (Figure 20, Step 1). Since there are no designs which combine the lowest of each property, the user performs a lasso selection around the designs near the bottom of the curve. This “Pareto frontier” [10], represents those designs which most optimally combine low-weight and low-overhang (Figure 15, Step 2).

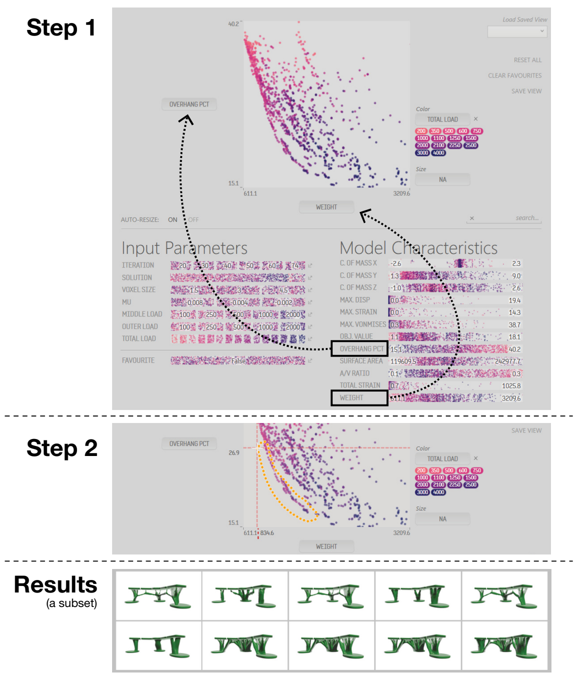

Figure 20. Steps to find those designs which best combine a low weight with a low overhang percentage.

Task 2: Find designs which satisfy multiple parties

When working in a collaborative environment, it is common to have multiple stakeholders with differing opinions on how the design should look or function. One approach to finding designs which will best satisfy multiple people is to have them each create a ranking with examples of designs they like (Figure 21, Step 1). By dragging the rankings for each person (Ann and Bob) to the MAG, we can find designs in the lower-left quadrant which score the highest on each of their personal rankings (Figure 21, Step 2). Another approach is to look at the "Top 9" results for each of the rankings, and see if Bob can find any designs he likes at the top of Ann's rankings, or vice versa (Figure 21, Step 2 alternate).

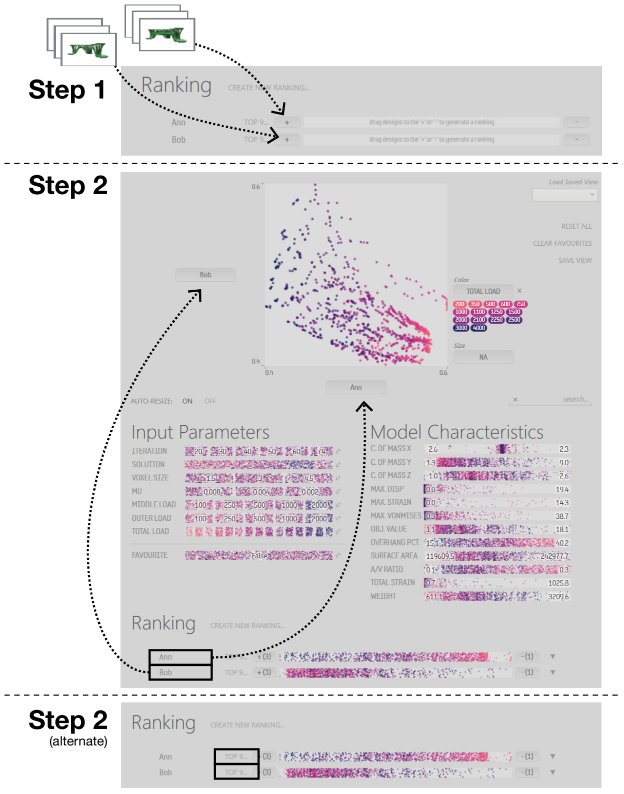

Figure 21. Possible workflow for reconciling preferences of multiple stakeholders.

Task 3: Find inspiration from "non-standard" designs

Interviews with users of generative design software [9] have suggested that in many cases the computed geometry is used as a starting point for design exploration, rather than the end product. In this use case, the user might want to see some examples of "non-standard" designs produced by the system, as a way to spark their own creativity.

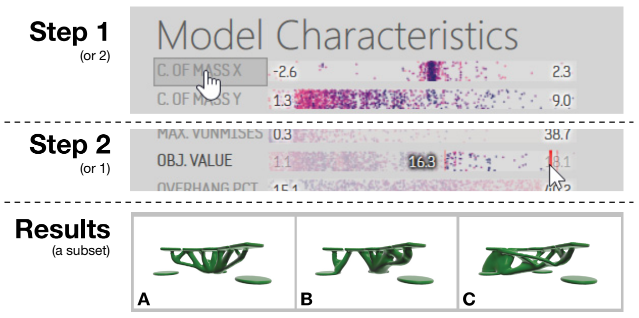

Figure 22. Possible steps for finding "non-standard" results within the dataset.

To do this, a user might have noticed that some attributes have most designs highly clustered in one area, with a smaller number of designs distributed in the other regions. They could right-click on the attribute label to see what the designs look like at the extreme ends of the variable (Figure 22, Step 1). Alternately, they could select the sparsely populated region of the slider to see only those designs which are on the "fringes" (Figure 22, Step 2). These designs could then be stacked in the *model viewer* and the *chisel* tool used to remove the "standard" designs.

At first glance, these results (Figure 22, A-C) appear like they might not work, that they could tip over. However, this problem definition assumes the base will be "bolted" or solidly affixed to the table surface. So, if affixed to the table surface, designs A-C would actually work, and could serve as inspiration for other designs incorporating a more "lopsided" or "suspended" look.

EVALUATION

To evaluate Dream Lens, we ran two studies: one lab study with expert computer users, to validate that users with minimal training could complete a variety of tasks, and a larger scale qualitative study with professional designers to gather subjective feedback and suggestions.

Lab Study

For the qualitative study, six paid volunteers (3 male, 3 female) were recruited between the ages of 22-48 (mean = 32). None of the participants had used generative design software. Four of the participants reported performing design tasks (graphic design, UI design) as part of their job.

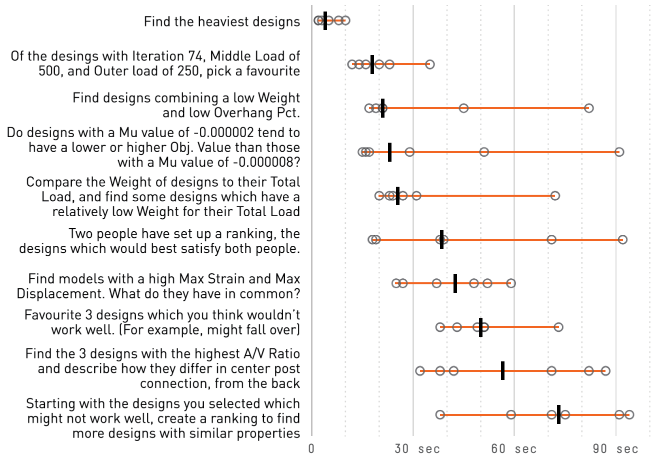

Figure 23. The task descriptions and completion times from the lab study. Black bar indicates median completion times.

The study took approximately 40 minutes: 5 minutes where the proctor described the task of designing a monitor stand, 5 minutes demonstrating the functionality of the system, then the participant took 5 minutes to become comfortable using the system. In the final 25 minutes the participant completed a series of 10 tasks (including three tasks similar to those described in the previous section) (Figure 23). The set of tasks were designed to replicate common scenarios which arise when exploring a generative design dataset.

All participants completed all tasks in less than 95 seconds, and four of the participants had average completion times of less than 40 seconds. In the post-study questionnaire, all participants rated the system either "easy" or "very easy" to use, and thought it would be "useful" or "very useful" to a designer in the generative design process.

Qualitative Feedback Sessions

To gather feedback from design professionals, 17 designers (2 female, 15 male) participated in one hour sessions where they were led through a series of tasks.

Many participants commented on the amount of data being displayed. Some users liked being exposed to so much:

I like programs that have as many variables as possible. I like to see all of my options. –P3

While others felt the interface was overwhelming:

The sliders are visually intensive. Do I really need a mini graph? A lot of things are fighting for my attention. – P12

In terms of the individual features, users were particularly drawn to the ranking panel…

I like that the algorithm does the work, and you can adjust it afterwards. – P10

It's somehow classifying what I call 'branchy'. I can't believe what it's doing. Oh wow! – P9

…as well as the stacked-model view:

With the models stacked, it feels like I'm working with a much smaller dataset, and the chisel tool feels more "natural" than working with the data filters. – P1

Overall, the participants enjoyed using the Dream Lens system, and were excited about the possibilities a Dream Lens-like system could mean for their design workflow.

DISCUSSION AND FUTURE WORK

We used the "filtered" dataset of 1,242 designs rather than the full set of 16,800 primarily because the designs from successive iterations were very similar. Better handling of very similar designs, such as clustering or grouping visually similar designs in the *design view*, would be a useful step for dealing with even larger, or less-diverse datasets. Although we removed "extreme outlier" designs, during the studies we found that the outlier designs which remained were some of the most interesting for users. It would be interesting to look at how to include extreme designs, without having those designs negatively affect the data visualizations. In addition to more diverse and larger datasets, it would be useful to explore how to handle datasets with a larger number of parameters. In *Dream Lens*, 20 parameters per-model are exposed, while in *Video Lens* [24], each event has 43 parameters – a different organization might be required if the dataset has a larger number of parameters per object.

In the stacked model view, we prototyped a flat (non-shaded) rendering, and volumetric rendering, but preferred the appearance and of the composited shaded rendering. Moving to a volumetric renderer might allow for a larger number of models to be "stacked", but would lose the crisp rendering of the edges of the individual models. Exploring alternative rendering techniques for this type of visualization would be an interesting future direction.

Due to the amount of time required to generate designs, *Dream Lens* works by loading in a large set of pre-generated designs to explore. It would be interesting to use *Dream Lens* in the early stages of generation to examine a sample of designs to "guide" the generation engine towards types of designs and parameters that the user likes. Additionally, after choosing a design, it would be interesting to be able to send that back to the simulation engine and ask it to "make more like this one" to enable a more iterative design process.

Overall, we believe *Dream Lens* represents a novel contribution to the area of interactively exploring large collections of 3D models, particularly those with subtle variations in their aesthetic and physical properties. Combined with contributing a dataset of 16,800 generatively designed solutions to the same design problem, we hope our work can serve to inspire future research in this area.

REFERENCES

1. Alhashim, I., Li, H., Xu, K., Cao, J., Ma, R. and Zhang, H. 2014. Topology-varying 3D Shape Creation via Structural Blending. *ACM Trans. Graph.* 33, 4: 158:1–158:10. http://doi.org/10.1145/2601097.2601102
2. Allaire, G., Jouve, F. and Toader, A.-M. 2004. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics* 194, 1: 363–393.
3. Ashour, Y. and Kolarevic, B. 2015. Optimizing Creatively in Multi-objective Optimization. *Proceedings of the Symposium on Simulation for Architecture & Urban Design* (SimAUD '15), Society for Computer Simulation International, 128–135. Retrieved March 24, 2017 from http://dl.acm.org/citation.cfm?id=2873021.2873039
4. Attar, R., Aish, R., Stam, J., Brinsmead, D., Tessier, A., Glueck, M. and Khan, A. 2010. Embedded Rationality: A Unified Simulation Framework for Interactive Form Finding. *International Journal of Architectural Computing* 8, 4: 399–418. http://doi.org/10.1260/1478-0771.8.4.399
5. Averkiou, M., Kim, V.G., Zheng, Y. and Mitra, N.J. 2014. ShapeSynth: Parameterizing Model Collections for Coupled Shape Exploration and Synthesis. *Comput. Graph. Forum* 33, 2: 125–134. http://doi.org/10.1111/cgf.12310
6. Bendsoe, M.P. and Sigmund, O. 2013. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media.
7. Benjamin, D. 2012. Beyond Efficiency. *S. Marble, Design workflow*: 14–27.
8. Ben-Yitzhak, O., Yogev, S., Golbandi, N., Har'El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E. and Sznajder, B. 2008. Beyond basic faceted search. *Proceedings of the international conference on Web search and web data mining - WSDM '08*, ACM Press, 33. Retrieved from http://portal.acm.org/citation.cfm?doid=1341531.1341539
9. Bradner, E., Iorio, F. and Davis, M. 2014. Parameters Tell the Design Story: Ideation and Abstraction in Design Optimization. *Proceedings of the Symposium on Simulation for Architecture & Urban Design* (SimAUD '14), Society for Computer Simulation International, 26:1–26:8. Retrieved March 24, 2017 from http://dl.acm.org/citation.cfm?id=2664323.2664349
10. Corne, D., Knowles, J.D. and Oates, M.J. 2000. The Pareto Envelope-Based Selection Algorithm for Multi-objective Optimisation. *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature* (PPSN VI), Springer-Verlag, 839–848. Retrieved April 4, 2017 from http://dl.acm.org/citation.cfm?id=645825.669102
11. Denning, J.D. and Pellacini, F. 2013. MeshGit: Diffing and Merging Meshes for Polygonal Modeling. *ACM Trans. Graph.* 32, 4: 35:1–35:10. http://doi.org/10.1145/2461912.2461942
12. Doboš, J. and Steed, A. 2012. 3D Diff: An Interactive Approach to Mesh Differencing and Conflict Resolution. *SIGGRAPH Asia 2012 Technical Briefs* (SA '12), ACM, 20:1–20:4. http://doi.org/10.1145/2407746.2407766
13. Doraiswamy, H., Ferreira, N., Lage, M., Vo, H., Wilson, L., Werner, H., Park, M. and Silva, C. 2015. Topology-based Catalogue Exploration Framework for Identifying View-enhanced Tower Designs. *ACM Trans. Graph.* 34, 6: 230:1–230:13. http://doi.org/10.1145/2816795.2818134
14. Gaspar-Cunha, A., Loyens, D. and Hattum, F. van. 2011. Aesthetic Design Using Multi-Objective Evolutionary Algorithms. *Evolutionary Multi-Criterion Optimization*, Springer, Berlin, Heidelberg, 374–388. http://doi.org/10.1007/978-3-642-19893-9_26
15. Hearst, M. 2008. UIs for Faceted Navigation: Recent Advances and Remaining Open Problems. *International Journal of Machine Learning and Computing*, 337–343. Retrieved from http://research.microsoft.com/en-us/um/people/ryenw/hcir2008/doc/HCIR08-Proceedings.pdf
16. Jain, A., Thormählen, T., Ritschel, T. and Seidel, H.-P. 2012. Exploring Shape Variations by 3D-Model Decomposition and Part-based Recombination. *Comput. Graph. Forum* 31, 2pt3: 631–640. http://doi.org/10.1111/j.1467-8659.2012.03042.x
17. Kalogerakis, E., Chaudhuri, S., Koller, D. and Koltun, V. 2012. A Probabilistic Model for Component-based Shape Synthesis. *ACM Trans. Graph.* 31, 4: 55:1–55:11. http://doi.org/10.1145/2185520.2185551
18. Kim, H., Choo, J., Park, H. and Endert, A. 2016. InterAxis: Steering Scatterplot Axes via Observation-Level Interaction. *IEEE Transactions on Visualization and Computer Graphics* 22, 1: 131–140. http://doi.org/10.1109/TVCG.2015.2467615
19. Koren, J., Zhang, Y. and Liu, X. 2008. Personalized Interactive Faceted Search. *Proceedings of the 17th International Conference on World Wide Web* (WWW '08), ACM, 477–486. http://doi.org/10.1145/1367497.1367562
20. Koyama, Y., Sakamoto, D. and Igarashi, T. 2014. Crowd-powered Parameter Analysis for Visual Design Exploration. *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (UIST '14), ACM, 65–74. http://doi.org/10.1145/2642918.2647386
21. Lee, B., Smith, G., Robertson, G.G., Czerwinski, M. and Tan, D.S. 2009. FacetLens: Exposing Trends and

Relationships to Support Sensemaking Within Faceted Datasets. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09), ACM, 1293–1302. http://doi.org/10.1145/1518701.1518896

22. Lienhard, S., Specht, M., Neubert, B., Pauly, M. and Müller, P. 2014. Thumbnail Galleries for Procedural Models. *Comput. Graph. Forum* 33, 2: 361–370. http://doi.org/10.1111/cgf.12317
23. Marks, J., Andalman, B., Beardsley, P.A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J. and Shieber, S. 1997. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH '97), ACM Press/Addison-Wesley Publishing Co., 389–400. http://doi.org/10.1145/258734.258887
24. Matejka, J., Grossman, T. and Fitzmaurice, G. 2014. Video lens: rapid playback and exploration of large video collections and associated metadata. *Proceedings of the 27th annual ACM symposium on User interface software and technology*, ACM, 541–550.
25. Nagy, D., Lau, D., Locke, J., Stoddart, J., Villaggi, L., Wang, R., Zhao, D. and Benjamin, D. Project Discover: An Application of Generative Design for Architectural Space Planning.
26. Porter, T. and Duff, T. 1984. Compositing Digital Images. *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH '84), ACM, 253–259. http://doi.org/10.1145/800031.808606
27. Shea, K., Aish, R. and Gourtovaia, M. 2005. Towards integrated performance-driven generative design tools. *Automation in Construction* 14, 2: 253–264. http://doi.org/10.1016/j.autcon.2004.07.002
28. Stuart-Moore, J., Evans, M. and Jacobs, P. 2006. Interface Design for Browsing Faceted Metadata. *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries* (JCDL '06), ACM, 349–349. http://doi.org/10.1145/1141753.1141844
29. Terry, M. and Mynatt, E.D. 2002. Side Views: Persistent, On-demand Previews for Open-ended Tasks. *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology* (UIST '02), ACM, 71–80. http://doi.org/10.1145/571985.571996
30. Turrin, M., von Buelow, P. and Stouffs, R. 2011. Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. *Advanced Engineering Informatics* 25, 4: 656–675. http://doi.org/10.1016/j.aei.2011.07.009
31. Ulu, N.G. and Kara, L.B. 2015. DMS2015-33: Generative interface structure design for supporting existing objects. *Journal of Visual Languages & Computing* 31, Part B: 171–183. http://doi.org/10.1016/j.jvlc.2015.10.016
32. Vandic, D., Frasincar, F. and Kaymak, U. 2013. Facet Selection Algorithms for Web Product Search. *Proceedings of the 22Nd ACM International Conference on Conference on Information &#38; Knowledge Management* (CIKM '13), ACM, 2327–2332. http://doi.org/10.1145/2505515.2505664
33. Voigt, M., Werstler, A., Polowinski, J. and Meißner, K. 2012. Weighted Faceted Browsing for Characteristics-based Visualization Selection Through End Users. *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (EICS '12), ACM, 151–156. http://doi.org/10.1145/2305484.2305509
34. Wang, M.Y., Wang, X. and Guo, D. 2003. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering* 192, 1: 227–246.
35. Xu, K., Zhang, H., Cohen-Or, D. and Chen, B. 2012. Fit and Diverse: Set Evolution for Inspiring 3D Shape Galleries. *ACM Trans. Graph.* 31, 4: 57:1–57:10. http://doi.org/10.1145/2185520.2185553
36. Yumer, M.E., Asente, P., Mech, R. and Kara, L.B. 2015. Procedural Modeling Using Autoencoder Networks. *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST '15), ACM, 109–118. http://doi.org/10.1145/2807442.2807448
37. Zaman, L., Stuerzlinger, W., Neugebauer, C., Woodbury, R., Elkhaldi, M., Shireen, N. and Terry, M. 2015. GEM-NI: A System for Creating and Managing Alternatives In Generative Design. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15), ACM, 1201–1210. http://doi.org/10.1145/2702123.2702398
38. Generative Design at Airbus | Customer Stories | Autodesk. Retrieved January 8, 2018 from https://www.autodesk.com/customer-stories/airbus
39. Here's What You Get When You Design a Chair With Algorithms. *WIRED*. Retrieved January 8, 2018 from https://www.wired.com/2016/10/elbo-chair-autodesk-algorithm/
40. Project Dreamcatcher | Autodesk Research. Retrieved April 5, 2017 from https://autodeskresearch.com/projects/dreamcatcher