DreamSketch: Early Stage 3D Design Explorations with Sketching and Generative Design

Rubaiat Habib Kazi, Tovi Grossman, Hyunmin Cheong, Ali Hashemi, George Fitzmaurice
Autodesk Research, Toronto, ON, Canada

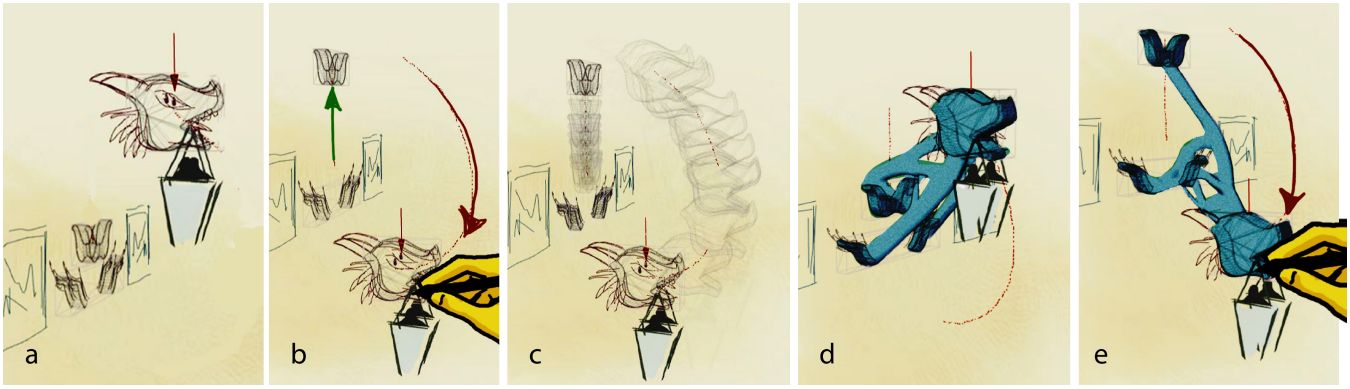

Figure 1: We bring the power of generative design in the early stages of design exploration with freeform 3D sketches. (a) The user defines a problem definition by roughly sketching a dragon-shaped lamp holder, the anchor to the wall, and a load value (red arrow). (b) The user then defines a *design variable* (red) and a *constraint* (green) to specify a range of possible spatial configurations. (c) The design space of the resulting problem definition is specified, with a range of possibilities. (d-e) Our system then generates optimal solutions for the entire design space using topology optimization, along with their performance data. The user navigates through the results by directly manipulating the objects.

ABSTRACT
We present DreamSketch, a novel 3D design interface that combines the free-form and expressive qualities of sketching with the computational power of generative design algorithms. In DreamSketch, a user coarsely defines the problem by sketching the design context. Then, a generative design algorithm produces multiple solutions that are augmented as 3D objects in the sketched context. The user can interact with the scene to navigate through the generated solutions. The combination of sketching and generative algorithms enables designers to explore multiple ideas and make better informed design decisions during the early stages of design. Design study sessions with designers and mechanical engineers demonstrate the expressive nature and creative possibilities of DreamSketch.

Author Keywords
Sketching, 3D design, generative design, ambiguity, CAD.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI).

UIST 2017, October 22–25, 2017, Quebec City, QC, Canada

ACM ISBN 978-1-4503-4981-9/17/10…$15.00
https://doi.org/10.1145/3126594.3126662

"If you want to get the most out of a sketch, you need to leave big enough holes. Ambiguity creates the holes. The fact that sketches leaves a lot out, or leaves a lot to the imagination, is fundamental to the process" – Buxton [10]

INTRODUCTION
Product design processes, from the inception of an idea to its realization, typically start with free-hand, rough 2D sketches, and gradually move towards precise, detailed 3D CAD modeling [10]. Within this continuum, the design process stretches from implicit to explicit representations. In the early stages, design tends to be ambiguous, incomplete, and expressive with high levels of uncertainties and a range of possibilities, with sketches used as the main representation. On the other end of the continuum, CAD modeling is used for detailed design, where representations are explicit, with precise geometric models, dimensions, and specifications. Such representations are necessary for analyzing, validating, and finally fabricating the design. Traditionally, these two processes occur independently one after another, hence the representations used are disconnected.

Lately, the role of computers during the design process has been changing. Recent advancements in parametric design, artificial intelligence techniques, design simulation and optimization have enabled computational tools to play an active, participatory role in the design process. In general, these tools allow designers to iterate through a greater number of design possibilities compared to the traditional 3D modeling process [4, 14, 20, 29, 36]. In particular, generative design software uses the above mentioned technologies to

generate design solutions for a given problem specified by the user [1, 8, 9, 35, 37]. However, the typical input mode for problem definition is much alike traditional CAD tools, mainly based on precisely defined 3D models and form-based inputs. It does not support any implicit and free-form input such as sketching.

Sketching enables more ambiguous, incomplete, and expressive representation of potential designs. These traits of sketching are considered central to the process of creative and design thinking [10, 16]. Prior works in sketch-based interfaces for modeling have mostly focused on translating 2D strokes into a 3D model [2, 5, 32, 38, 39], which addresses the gap between the two representations. However, not much work has looked at how sketching can be used as an input paradigm for computational design, and how early-stage sketched-based designs can act as a seed for generative design algorithms.

How can we leverage the advantages of both sketching and generative design to help designers explore solutions during the early stage design? The primary challenge is to accommodate the vague, ambiguous, and unfinished quality of sketches within a CAD environment.

In this paper, we propose a hybrid 3D design paradigm that combines the expressive and free-form nature of sketching with the computational power of generative design algorithms. This interaction paradigm still allows designers to focus on creative exploration in their early stages, while also getting a glimpse at a larger space of design variations that will satisfy their problem.

We designed and developed DreamSketch, a sketch-based interface to couple design sketches with generative design technologies. In DreamSketch, the designer mainly uses sketching to *loosely* define the design problem using sketching, direct manipulation, and contextual tools (i.e., the manikin). Our system then synthesizes multiple design solutions for the problem using topology optimization [1, 9, 37]. The designer can navigate through these results by directly manipulating the sketches in the canvas. In addition, designers can examine the quantitative aspects (i.e., weight, volume, stress) of the generated solutions, which are usually not available in the early stages of the design process. Therefore, DreamSketch fills a significant gap in the early ideation stage of conceptual design by allowing designers to explore multiple solutions, yet make better informed choices before proceeding to more expensive 3D modeling and prototyping.

One challenge of this approach is that generative design methods are computationally expensive. As such, DreamSketch is not yet capable of providing design solutions in real-time. However, our work can be thought of exploring the possibilities of such tools when they do become real-time and interactive.

We conducted design sessions with designers and engineers using DreamSketch to explore creative artifacts. These sessions demonstrated the unique affordances, capabilities, expressive nature of this tool. We discuss the potential, implications, and limitations of such design method, and discuss future directions.

RELATED WORK

Our work is inspired by prior research in sketch-based user interfaces for modeling and conceptual design, generative design methods, and design sketching. We review related work in these areas.

Sketching for Design

Sketch-Based Interfaces for Modeling (SBIM)

The overarching goal of SBIM research is to allow sketching as a way of 3D design, from rough conception to fine detail construction [27]. Recovering a 3D model from 2D sketches is a fundamentally ill-defined problem, due to the lack of depth information of the strokes [31]. To this end, researchers simplified the 3D reconstruction problem by assuming geometric constraints in design drawings and 3D geometry, such as planarity, parallelism, orthogonality, cross-section lines [2, 32, 38, 39], polyhedral scaffolds [31], and axis-aligned planes [5]. However, due to the geometric assumptions, these techniques require relatively accurate drawings as input, which implies that the users need to have an accurate, precise model in their mind. In contrast, early design sketches are often ambiguous and incomplete. Such geometric assumption would burden the designer with precise inputs. Recently, sketch-based input has also been explored for procedural modeling for urban 3D buildings [26].

While prior works have explored different aspects of sketching for 3D design, in this paper, we aim to explore a new dimension of sketching. We developed an end-to-end system that accommodates the early stage, freehand qualities of drawing with generative design for shape exploration and synthesis.

Sketching for ideation and 3D conceptual designs

Tools like Mental Canvas [13], Insitu [28], 3D6B [18], and more recently Tiltbrush[1] facilitates the creation of concept drawings by organizing their strokes in 3D, without trying to infer precise 3D models and structures. While effective for design conception and visual thinking, these tools are not coupled with simulation & CAD tools like ours, and they do not explicitly address the *ambiguous* aspects of drawing to define a range of possibilities.

Concept variations & ambiguous intentions

In the HCI literature, GEM-NI [40] explored parallel creation, visualization, and exploration of alternative 2D

[1] Google Tiltbrush: https://www.tiltbrush.com/

generative graphic designs. Other works looked at aiding the user in drawing tasks by suggesting a set of alternatives [17] or shadow images [23]. In the context of design sketching, Sketchsoup [3] allows designers to explore a range of design concepts by embedding a set of rough, unstructured 2D sketches into a 2D interpolation space to generate novel designs. Perhaps, the closest work to ours is Gross and Do's seminal Electronic Cocktail Napkin [16], that supports ambiguity by carrying alternative interpretations, and imprecision with constraints for 2D design diagrams. In similar spirit, Murugappan et al. developed FEAsy for structural analysis in early design [25]. Our work seeks inspiration from these works, but in the context of 3D design, and coupled with generative design algorithms. As such, our formulation and approach to ambiguity and abstraction is different from theirs.

Generative Design

Generative design tools take problem definition as input and produce feasible and/or optimal solutions for the given problem. In engineering, several generative design tools are commercially available, including Altair's OptiStruct and solidThinking, Autodesk's Nastran Shape Generator, and Siemen's Frustum. The core to all these tools is topology optimization [1, 8, 9, 37], which is a mathematical method that optimizes a layout of material distribution within a given design domain. Typically, the objective of the problem is to minimize compliance, which can be described as the flexibility of an object under a load, as well as the total volume. In addition to rigidity, recent works also explored using desired appearance, defined by a user-specified exemplar pattern, as an objective in a joint optimization problem [24]. Our work aims to bring the capabilities of generative design in the early stages of design, while accommodating implicit problem definitions through sketching. This would make generative design more accessible to novice designers and artists, who may not be familiar with 3D CAD tools.

SKETCH + CAD = A HYBRID DESIGN MEDIUM

What are the unique attributes of early stage sketches? How do they contrast and complement the qualities of generative design? We now review some of the cardinal aspects of concept sketches based on existing literature, and motivate the necessity of a hybrid design medium to couple these qualities with post-ideation CAD tools.

Properties of Sketching

Sketches are ambiguous

Design sketches are inherently ambiguous and explorative. In the early stages, designers typically draw a number of quick-and-dirty sketches capturing design inspirations, variations, and alternatives of a visual concept. This allow designers to their mental design space to others; often leaving regions of the design ambiguous and subject to interpretation. As Goel [15] suggests - "*Ambiguity is important because one does not want to crystallize ideas too early and freeze design development*". Goel argues that sketching can provide a better match with such ambiguity than structured CAD systems [15].

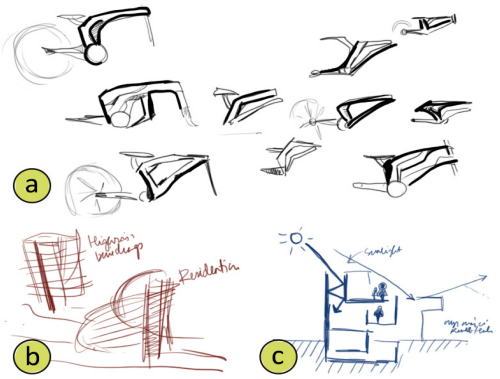

Figure 2: Unique aspects of conceptual design sketches. (a) *Ambiguous*: concept sketches represent a range of possibilities in a design space. (b) *Incomplete*: Not enough information to construct precise 3D geometric representations. (c) *Expressive*: designers use variety of visual and conceptual notations to communicate design ideas & intentions.

Sketches are incomplete

The incomplete nature of sketches permits designers postponing details for the later stage [7, 10]. This also allows designers not to commit to solutions too early. Due to the vague nature of sketches, often there is not enough information to construct precise geometric representations or structures of the intended design. However, CAD tools require precise representations. As such, we need to coarsely approximate the geometry of the sketches, without burdening the designers with precise, detailed inputs.

Sketches express ideas

Perhaps, the most important aspect of sketching is to be able to rapidly express and communicate ideas with peers, clients, and self [10]. Prior works in cognition [11, 34] indicate that designers deal with both *visual* (e.g., 'what', 'where') and *non-visual* (e.g., 'functional thoughts', 'knowledge') information. According to Baskinger & Bardel [7], one such method of information gathering in design sketching is "5W's and H" -

- *Who* is the design for? (Characters, entities)
- *Where* will it be used? (Contextual information)
- *When* will it be used? (Sequence & time-frame)
- *What* is the activity, quantity, or object?
- *How* will it be used?

Collectively, these entities equip designers with powerful visual thinking tools to schematize and express their ideas, goals, and intentions. Therefore, our medium should accommodate the expressive nature of design sketches to capture context, activities, goals, entities, and other forms of information.

Gross and Do [16] observed that designers prefer to use paper and pencil as a natural input mode because of these

freehand drawing qualities, and underscored the necessity of capturing these traits in computational design tools.

Generative Design Workflows

The following workflow is used in typical generative design tools such as Altair's OptiStruct, Autodesk's Shape Generator, and Siemen's Frustum. To set up a problem, a user must import or create a reference 3D model that represents an existing part to be redesigned. Then, various constraints and properties are specified with respect to the part using traditional UI interactions such as forms, context menus, etc. Once a problem is properly defined, the user can execute generation and a single optimal solution is returned (Figure 3).

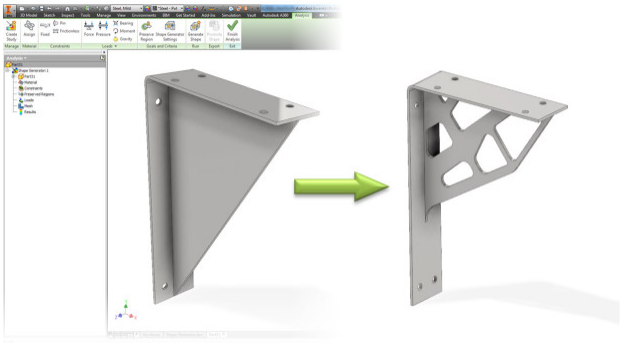

Figure 3: Shape Generator in Autodesk Inventor.

One could recognize that the above workflow is extension of traditional CAD workflows, requiring precise 3D models, calculations, and inputs, and hence used in the detailed design stages (i.e., post-ideation). Also, only a single optimal solution is returned at the end, resembling a convergent process. These traits do not support the fluid, ambiguous, and imprecise nature of the early stage design process and could hinder the creativity of the designer. In addition, this typical workflow does not support exploration of multiple solutions.

A Hybrid Approach

In this paper, we intend to bring the benefits of generative design in the early stages of design conception so that designers can start exploring the design space and make better informed design decisions earlier. By doing so, we intend to explore how generative design would play a more active, participatory role in early stages of design, while accommodating the unique attributes of freeform sketches.

DREAMSKETCH: USER INTERFACE

We designed and implemented DreamSketch, a 3D sketching tool coupled with generative design algorithms for conceptual 3D design and exploration. DreamSketch capitalizes the ambiguous, incomplete, and expressive nature of sketching for defining the problem. The UI supports juxtaposition of planar strokes in 3D that are used for problem definition, with a range of 3D meshes representing alternative solutions generated. The interface contains a main authoring canvas, a toolkit menu, and a GUI menu for parameter controls (Figure 4).

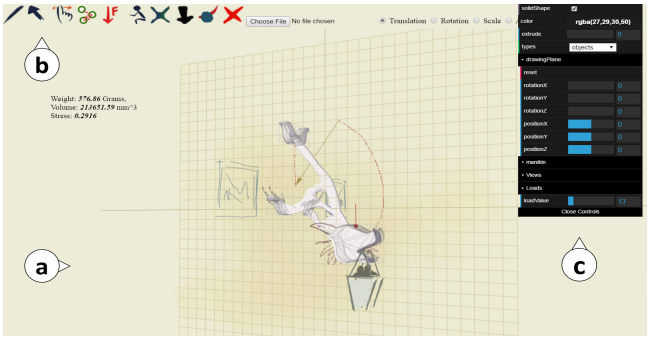

Figure 4: The UI of DreamSketch consists of the (a) main canvas, (b) toolkit menu (*draw*, *select & record design variable*, *drag variable*, *delete*, *generate*, *manikin*, *geometric attributes*), and (c) GUI menu for drawing canvas navigation in 3D, drawing tools (i.e., *color*, *extrusion*, *solid fill*), visualization (*hide/show drawing plane*, *obstacles*), and parameter controls (e.g., *manikin height*, *load values*).

Workflow: Define problem, Navigate results, & Iterate

In DreamSketch, the user starts by defining a design problem by specifying the design context, variables, constraints, and load cases, all using a sketch-based interface. The design context is specified by creating *objects* that the design must interact with. The *design variables* and *constraints* are created by specifying the possible positions and configurations of the interacting objects. *Load cases* are then defined on the *objects* to express the intended functionality of the design. *Proxy Objects* such as a human manikin, can also be used to guide and constrain the design process.

Given the problem definition, our system then procedurally generates multiple solutions using topology optimization that minimizes compliance and volume. This process is currently executed offline, and the execution time can be from several minutes to hours, depending on the problem definition and algorithm parameters.

The resulting design alternatives are then presented back to the user within the context of problem definition sketches, along with the quantitative attributes for each solution. The user can evaluate and navigate through the generated results by directly manipulating the spatial arrangements of the *objects*. The user can iteratively adjust the problem definition and generate new results. Once the design space has been explored to satisfaction, the designer can output the selected design(s) to fabrication tools or export the resulting geometry for further processing in other software tools. Figure 1 summarizes the overall workflow of the system.

Problem Definition in DreamSketch

We now discuss the UI tools used for problem definition in DreamSketch.

3D Sketching and Navigation

The user can draw with a tablet pen or a mouse, and sets down strokes directly on a planar 2D *canvas*. The *canvas* can be positioned and rotated in 3D space, using the position and rotate sliders on three coordinate axes. We overlay a transparent grid on the canvas so that the user can easily see

their orientation onscreen, and use it for guidance. The user can rotate, scale, and pan the camera view. By default, we use perspective projection. But, the user can also switch to orthographic projection for quick sketching in 2D, without worrying about depth. By default, the strokes are automatically mirrored with respect to the *xy*-plane to create symmetric shapes.

Design Context via Objects

Objects are the primary 3D visual entities in DreamSketch to express the design context. The intended design solution must interact with these objects to fulfil its functionality. An *object* consists of a set of user drawn strokes, and a 3D mesh that coarsely define the geometric volume of the intended shape. Constructing precise 3D geometry from rough drawings is a challenging problem, requiring precise, unambiguous user input. However, in the early stages, designers do not have a precise, unambiguous geometry in mind. Therefore, we allow users to coarsely approximate the intended shape with proxy geometry [33], without requiring detailed input.

To define a 3D shape, the user sketches a set of strokes on the canvas, and then extrudes them to create a 3D mesh. We used a simple heuristic for extrusion. Our system selects the 2D planar stroke with maximum arc length, and extrudes the stroke in the normal direction of the *canvas* to create a mesh. While not precise, this is a simple, yet effective approach to approximate the 3D volume quickly and easily for typical use cases. (Figure 5). The resulting mesh is then displayed into the canvas in conjunction to the hand-drawn strokes. To keep a user's focus on ideation rather than surface quality and continuity [6, 18, 30], the mesh is rendered with wireframes in a sketchy style (Figure 5c). Overall, our goal was to keep the workflow simple, yet expressive to accommodate crucial characteristics of freehand design drawings, including over-tracing, hatching, and shading

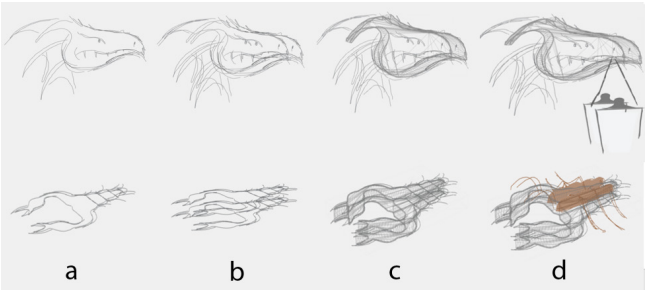

Figure 5: An *object* consists of a *proxy geometry* and freeform strokes. (a) Strokes drawn in the drawing canvas. (b) The extrude operation mirrors the strokes with respect to the canvas. (c) The 3D geometry created from the extrusion visualized in wireframe rendering. (d) The user continues adding strokes to the *object* for more details and context.

In addition to manually designing objects, the user can import existing 3D models as an *object* in the scene, and select, edit, delete, or transform (translate or rotate) any *object* using a free transform tool.

Object Types

Once created, the user specifies the type of an *object* as either an *interface object* or an *obstacle*. *Interface objects* are used to define load cases and also integrated as part of the solution geometries. An *obstacle* defines an empty volume, which forces the algorithm not to include that volume as part of the solution geometries.

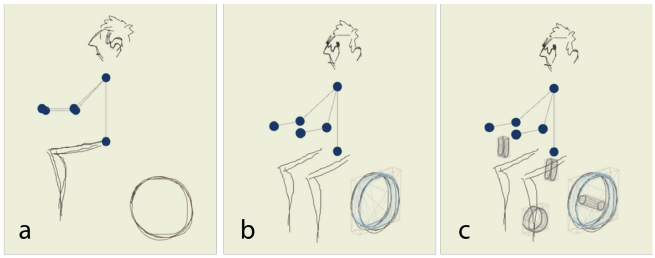

Figure 6: *Object* types. (a) The user loads the manikin, and sketches the back wheel. (b) An *object* is created by extrusion, and set as an *obstacle* to define an empty volume for the final design. (c) The user then sketches the *interface objects* for the bike handle, paddle, rear bracket, and seat post.

Design Variables

Design variables provide a means to define a design space, accommodating and exploring a wide range of possibilities for a design. To define a *design variable*, the user selects an *object*, and moves it along the *canvas* to record a *design variable* (Figure 7). The *design variable* is visually represented with a polygon path in the 3D scene, and it represents all possible positions of the associated *object*. The user can remove or override an existing *design variable* by simply tapping the *object*, or recording a new one respectively. In our current implementation, a *design variable* consists of a polygon path only.

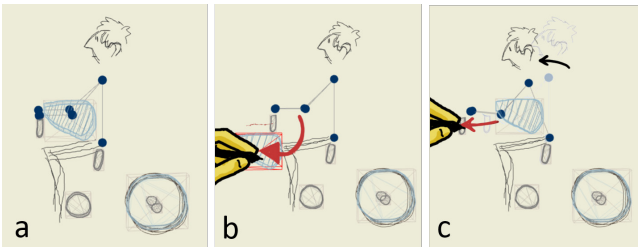

Figure 7: Defining a *design variable*. (a) An *obstacle object* sketch (blue). To generate variations, the user then specifies all possible positions of the *obstacle* (b) and the bike handle (c) by recording *design variables* with direct manipulation. Note that the manikin hand is attached to its nearest *object* (handle), and updates its pose accordingly (c).

Once an *object* is defined, it can be dragged along the polygon path by enabling the *drag* tool. In our current implementation, *design variable* movements are restricted to *xy*-plane only. In theory, the user can have many *design variables* in problem definition to express many possible spatial configurations between the *objects* in a scene. Each *design variable* represents a dimension in the design space; *n* independent *design variables* creates an *n*-dimensional design space. In practice, this exponentially increases the

possible solutions, making the generation process computationally expensive and time consuming. As such, we restrict each design solution to at most three variables.

Constraints

In DreamSketch, a design *constraint* refers to conditions under which the geometric parameters of an *object*, such as position, orientation, or scale is dependent upon the position of another *object*. For instance, in Figure 8c, the wine bottle orientation is constrained to the position of the bottle holder bases. When the bases move to a new position, the wine bottle geometry needs to rotate/re-position appropriately.

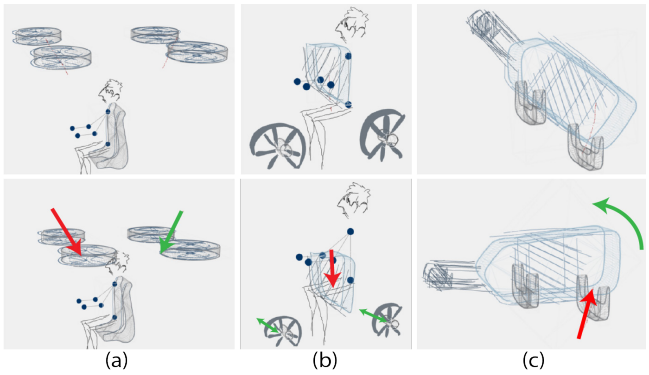

Figure 8: Examples of *constraints*. (a) A translation *constraint* (green) ensures the rotors are symmetric with respect to the rider, as the other rotors moves through *design variable* (red). (b) By design, the scale of the bike wheels is dependent to the position of the *obstacle* (blue) (c) As the position of the base changes (red), the rotational *constraint* (green) re-orients the bottle.

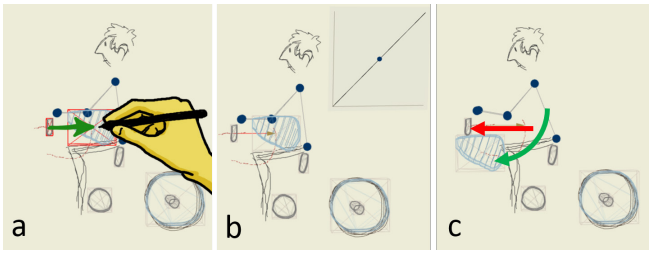

Figure 9: Defining a *constraint*. (a) The user sketches a *constraint* from the source (handle) to the destination (*obstacle*), and sets it as a translation *constraint*. (b) The functional relationship curve defining the geometric *constraint* (inset). (c) As the position of the handle (red) changes, the position of the *obstacle* changes accordingly (green).

We used a fast and flexible method - functional relationship [19] - to define *constraints* between the geometric parameters of two *objects*. To define a *constraint*, the user selects the *constraint* tool, and sketches an edge from the source *object* to the constrained *object* (Figure 9a). This creates a directed edge between them. The user can specify the type of geometric parameter (i.e., *scale*, *rotation*, *position*, *assembly*) for the functional mapping by selecting the corresponding parameter in the top menu. When an edge/constraint is created or selected, a function mapping widget visualizes the functional correspondence ($t_p = F(s_p)$) between the geometric attributes of the parent s_p (x-axis) and the child t_p (y-axis) (Figure 9b). By default, this function is set to identity ($s_p = t_p$), which the user can overwrite by directly sketching a new curve within the widget. The widget is equipped with an exploration cursor [19] that the user can drag within the mapping space to help identify relevant mapping points.

Manikin

Kim and Bae explored the use of hand-tracking sensors to aid designers with the sense of scale, context, and hand posture information [22]. In similar spirit, DreamSketch supports the use of a manikin into the scene to think and express the design in a realistic environment. Additionally, it enables the designer to define and validate *design variables* and solutions. For instance, in Figure 7c, once the designer specifies a *design variable* for the bike handle, our system registers the hand (end-effector) of the manikin with the nearest *object* (bike handle). As the position of the handle changes, the manikin pose changes accordingly to reflect the pose of the rider. This enables the designer to qualitatively evaluate the validity of the *design variables* by looking at the corresponding poses for different possible positions of the handle. The user can also vary the height of the manikin using a slider to evaluate the feasibility of a design configuration for a range of heights (Figure 10). Such functionalities enable the user to consider design ergonomics in early stages of design to eliminate invalid explorations and solutions, which is typically considered in the later stages of design pipeline [41].

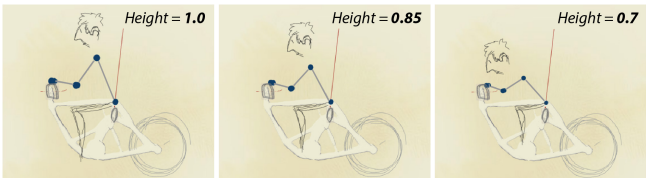

Figure 10: The user exploring the validity (comfort and reach) of a solution for different manikin heights. The position of the hand and waist are fixed for different height configurations.

Load Cases

Problem definition in DreamSketch requires a *load case* defined on *interface objects*. A load case consists of 1) a set of *forces*, applied on *interface objects*, that the resulting design should be able to withstand and 2) an associated set of fixed *anchors*, defined also on *interface objects*. Essentially, one could think that design solutions are created to propagate the forces applied on the receiving *objects* to the *anchor objects* without failing. The user can specify a force vector by sketching using the force tool (Figure 11a). The length of the stroke is proportional to the force magnitude. Once the user sketches a force vector, by default our system marks all other *interface objects* (hence, not including *obstacles*) as *anchors* for that force (Figure 11b). The default heuristics work well for typical cases. However, for more advanced use cases (such as the bike frame design), users can toggle the *fixed objects* for that force (Figure 11c). Users can also edit the force values using a slider.

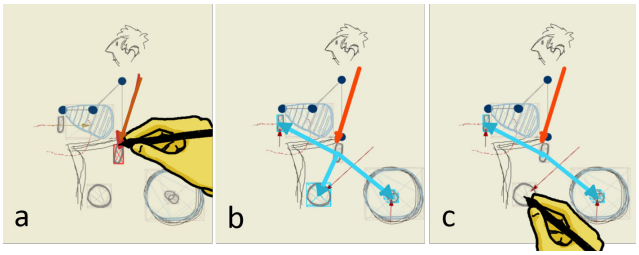

Figure 11: Defining *loads* and *anchors*. (a) A user sketching a force using the *load* tool. (b) The default *anchor objects* (blue) for the selected *load* (orange). (c) The user can toggle an *anchor object* by tapping the corresponding *object*.

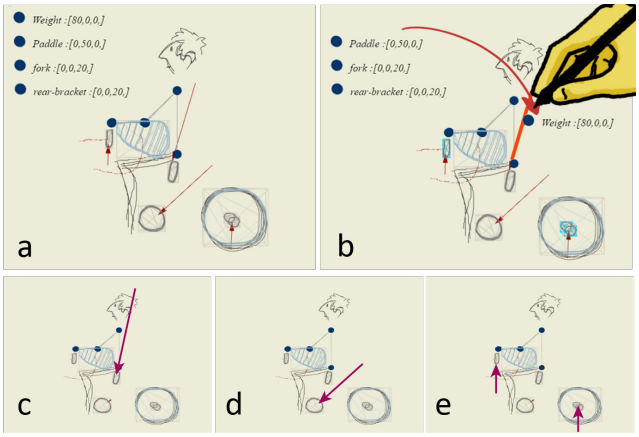

Figure 12: Data-driven *load conditions*. (a) The user exports a CSV file consisting of the values of each load for different *load conditions*. (b) The user then drags each data column to bind it to a corresponding *load* (orange). Our system iteratively visualizes the forces for each *load condition*. (c) A *load condition* for a user just sitting. (d) A force of a user pedaling. (e) Forces due to bumps and friction.

In mechanical design, design objects often consist of multiple *load cases* for advanced use cases. For instance, a bicycle has multiple use cases (Figure 12). For each use case, we have different set of forces applied, on a combination of handles, seat, or pedals. A user can define multiple *load cases* as part of problem definition.

Our system also enables users to export a data file (.CSV format), and drag the force magnitudes to the corresponding load vectors. This data-driven approach enable designers to design complicated *load cases* easily. For simplicity, we normalized the force value ranges to match with the maximum dimension (600*mm* X 600*mm* X 600*mm*) of our canvas. For instance, it is impractical to have a 1000kg load for a 1000 mm^3 *object*.

In DreamSketch, the minimum required input for a problem definition is a single *force* and multiple *interface objects* (to be synthesized by generative design algorithms). *Obstacles*, *constraints*, *design variables*, and *contextual sketches* are optional input requirements.

Geometry Generation and Navigation

Once the problem definition is specified, the user presses the *generate* button, and our system generates the optimal solution for different combinations of the design variables. We discretize the domain of each design variable into a fixed number of segments (four), and generate results for all possible combinations of those variable segments.

Navigating Results

Users can navigate through the generated results by directly dragging the *objects* through their *design variables* (Figure 13). Our system finds and displays the closest corresponding solution for each configuration. The user can also set a solution as 'marked', to be reviewed at the end of the design session.

Quantitative Attributes

For each solution, we display a selection of its quantitative attributes, including volume, weight, and stress. The quantitative aspects of design allow designers to compare different solutions and make informed decisions in the early stages of design.

Iterative Refinements

Users can continue refining the design iteratively by modifying the problem definition, either by modifying *variables*, *constraints*, and *load conditions*, or by adding or removing *interface objects* and *obstacles*. The key difference of DreamSketch (or any other generative design tool) compared to a traditional design process is that it allows iterations by modifying problem definitions for a design space and exploring the solutions generated by the computer. We hypothesize that this approach enables designers to explore a greater number of possibilities than iterating through by only modifying the solutions.

Rendering

We used non-photorealistic rendering techniques to the resulting solutions. The goal of DreamSketch is to inspire the designers by showing the possibilities, inviting further exploration, and changes. The visual rendering is coupled with its intended purpose to keep the results suggestive.

IMPLEMENTATION

Our system runs in a web browser, and is implemented using Javascript and the Three.JS library that uses WebGL for 3D graphics rendering and animation. Once the user defines a design problem, our system passes the problem definition input files to the topology optimization module via a Python API. The system then generates optimized geometries for all design configurations specified in the problem definition, along with quantitative attributes (i.e., weight, volume, stress), and returns those results back to the DreamSketch interface in the browser.

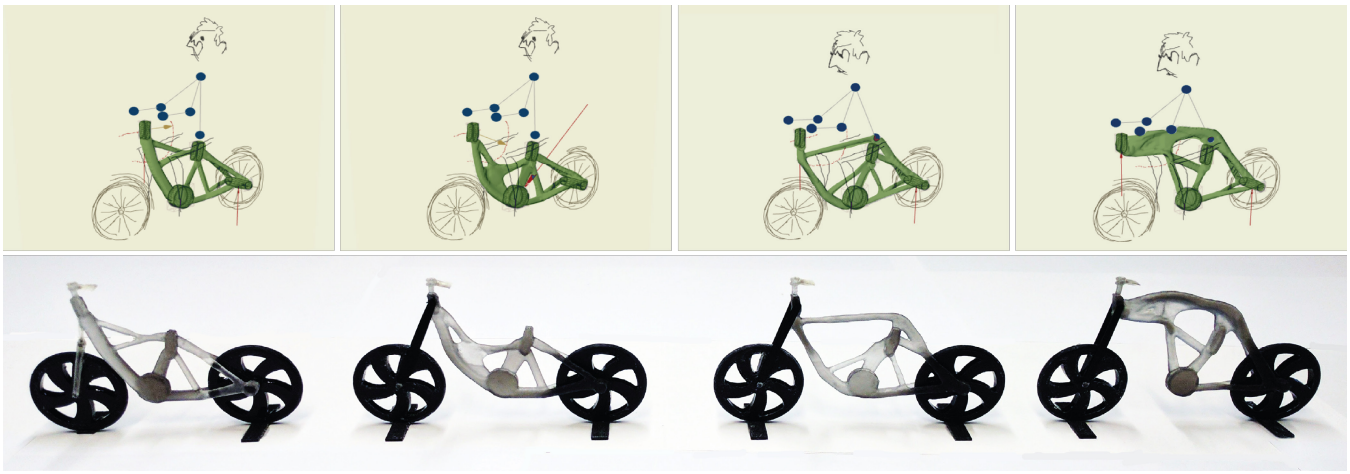

Figure 13: The user navigating through the resulting solutions by directly manipulating the bike handle (top). The execution time took approximately 4 minutes for each solution (16 minutes in total). The corresponding 3D printed artifacts (bottom).

Problem Definition Input

We discretize each continuous design variable into a fixed number of segments and create distinct *problem definition* for each possible combination of the design variable segments. If the designer specifies a design space consisting of n independent *design variables*, and we discretize each *design variable* into x segments, then our system generates x^n distinct *problem definitions*. In our current implementation, $x = 4$. So, if we have a design space consisting of three *design variables*, our system generates 4^3 (= *64*) distinct *problem definitions*. The *problem definitions* are saved as JSON files following the input data schema for the topology optimization module, and the associated problem definition *objects* are saved as mesh (.obj) in the local hard drive, to be accessed by the module.

Topology Optimization

The full details of the specific topology optimization module which we use is beyond the scope of this paper and its contribution. The topology optimization module is based on level-set shape optimization [1, 37]. For a given problem, the objective function is set to minimize the compliance (the inverse of stiffness, hence measures a degree of flexibility) and the volume. The weights of these two objectives can be controlled by a parameter, which is set dynamically based on a proprietary heuristic. Alternative optimization methods [8, 9] could be substituted, and still preserve the overall user experience which our work contributes. In general, topology optimization methods also take material properties into account. By default, we have set the material as *aluminum* for all the examples in this paper.

Execution Time & Performance

In practice, topology optimization methods are computationally expensive, and do not run in real-time. The execution time is highly dependent on the problem definition, and a number of other factors, including resolution, *load conditions*, number of iterations, and PC configuration. Higher resolution and iteration number significantly increases the execution time, but the resulting solution exhibits finer details and performance. As illustrated in Figure 14, the generation of a single bike frame can range from 30 seconds to 40 minutes depending on the desired resolution and number of iterations which the algorithm runs. We hypothesize that designers will gradually use finer results, as they progress towards the final design.

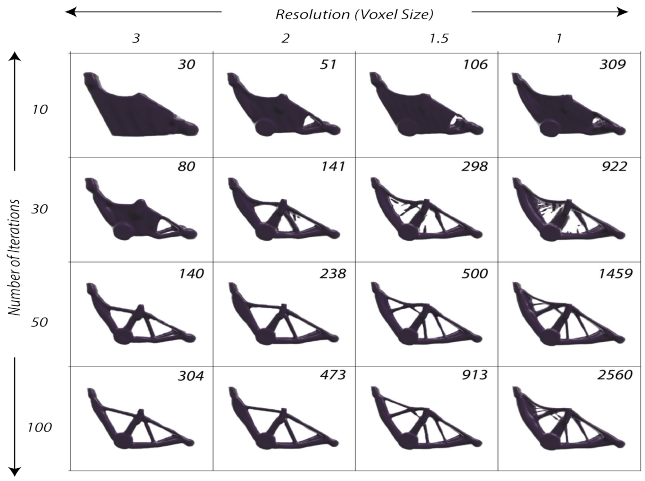

Figure 14: Execution time (in seconds) for a single problem definition of a bike frame. The dimension of the geometry is *226x172x57* for *voxel size*=1. The program was executed in an Intel Xeon E5 2650 v3 @ 2.30GHz CPU, with 64.0GB memory and NVIDIA Tesla K40c graphics card.

Inverse Kinematics

For the manikin movement, we implemented 2D inverse kinematics using the cyclic coordinate descent [21] method. In our current implementation, we fix the position of the manikin waist. As the user interactively manipulate the *design variable* attached to the hand (i.e., end effector), our system interactively computes the position of the elbow and neck, and update the manikin in the scene. Currently, the hand movements are constrained to *xy*-plane.

DESIGN SESSIONS

We conducted informal, qualitative design sessions with engineers and designers to gain insights about the overall design workflow, capabilities, and potential applications, and identify any limitations or opportunities for future advancements. These sessions were also used to gather feedback on how our system compares to existing approaches, although we do not perform any sort of formal comparison to existing commercial tools.

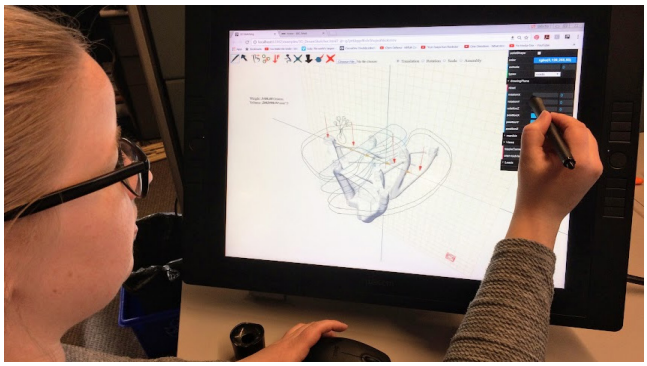

Figure 15: An user using Dreamsketch in a design session

Participants

We recruited seven participants (4 female, aged 24-40) for our design sessions. Participants completed a questionnaire about their background before the session began. Among them, one participant is an architect (*P2*), one industrial designer (*P5*) with 10+ years of experience, three mechanical engineers (*P1*, *P3*, *P6*), and two graphic designers (*P4*, *P7*) with no prior 3D design experience.

Procedure

Our design sessions for each participant lasted between 1.5-2 hours. All the sessions took place in our research lab, consisting of two steps. In the first step, each participant was given a brief overview of the system and was shown some resulting 3D designs made with our system. The instructor walked the participant through a step-by-step tutorial to familiarize the participant with the system, defining problem definitions, navigating solutions, and iterating through the design. In the second step, the participants designed a 3D artifact using our system. Participants filled out a questionnaire afterwards.

Discussion

In general, participants responded positively to the overall workflow, capabilities, and approach to 3D design using DreamSketch. Participants commented that the ability generate 3D shapes with freeform sketches, without having precise geometries, input, and calculations lowers the barrier for 3D design, ideation, and exploration (*P1*, *P3*, *P4*, *P7*).

P1: "*DreamSketch is quite different in the approach to generative design/topology optimization tools. I would classify it as a tool useful in the initial stage of the product development process. The very quick creation (no need to open a CAD tool, calculate loads) of meaningful geometry can help to kick-start the design process and helps with the communication. To use an optimization software, a set of ports have to be designed and imported and I have to think about all the boundary conditions and loads. DreamSketch lifts that burden, and I get to preliminary results much faster. The lack of accuracy is, of course, a feature. But at a later stage, the accuracy of position/ loads is of great importance*."

Our participants were intrigued by ability to define a design space with design variables, and explore different alternatives within that space with sketching and direct manipulation (*P1*, *P6*, *P7*). Participants also felt that the modeling capabilities of DreamSketch were sufficient to rapidly express their ideas (*P1*, *P3*, *P4*, *P6*, *P7*).

Often, the resulting geometries were very different than what the designer had anticipated, which inspired the designers to think into new design dimensions and space.

P3: "*The result was nothing I would have imagined. It ended up looking "industrial", and I never would have thought to go this route. To me, it looks a bridge. I never thought to take architectural influences into this design. However, now I am inspired to explore this directly*".

P6: "*It gives me revolutionary and radical designs with such simple inputs that are almost impossible to come to a human mind. I believe in this competitive market, one thing that could make a difference between two functional designs is aesthetic aspects of a design, and the shapes produced by DreamSketch due to their resemblance to the shapes we see in nature somehow resonate with our soul.*"

This indicates that generative design algorithms played an active and participatory role in the creative exploration and conceptual stages of design.

However, it is worth noting that, this (i.e., problem definition) is a relatively new form of thinking and approach for our designers (*P4*, *P7*). It took some time to comprehend the elements of problem definition – *interface objects, load cases*, *anchors* etc., and understand the scope of design. The instructor assisted them to specify their problem definition. The nature of generative design requires comprehension and understanding of the problem definition, constraints, and load conditions. We believe that our tool reduces the barrier of 3D designs, but some level of learning and familiarization with the tool will still be required.

Our participants – designers and engineers – represent two opposite sets of users with diverse backgrounds, goals, and approach towards design. We observed that our participants with engineering background (*P1*, *P6*, *P7*) were more meticulous about the *load values* to match with the scale of the design. These participants mentioned that DreamSketch gives them the creative freedom for design and ideation, which is missing in the existing tools for topology optimization. While, our participants with design background (*P2*, *P4*, *P5*, *P7*) acknowledged that having the quantitative information and optimized solutions makes them more aware and mindful about the functional aspects of design in conceptual stages.

Resulting Artifacts

Our participants crafted a wide range of artifacts during the design sessions (see Appendix). Due to longer execution time, we restricted the participants to define only one *design variable*. *P1* designed a quad-copter to carry a human (Figure 16). He sketched several symmetric *objects* to specify the rotors, an *object* (seat), and an *obstacle* to leave room for the rider. A *design variable* was defined to experiment with different positions of the rotors, and a *constraint* kept rotors spatially symmetric. A load case specified the weight of the rider. He then added more details to communicate his overall design intent and usage. *P3* and *P5* designed tables, and explored different variations for different positions of the table lid(s) (Figure 17) and table legs (Figure 18). As for Figure 17, the resulting artifacts vary quite significantly due to the nature of the problem definition.

P4, with no 3D design experience, sketched a reading lamp (Figure 20). She defined an *obstacle* to get arcade shaped solution. *P7*, also with no prior 3D design experience, designed a glider (Figure 19). She used a *constraint*, an *obstacle*, and the manikin to define the problem definition.

P6 designed a bike rack to be attached at the rear end of a car (Figure 21). He defined a *design variable* and two *constraints* to experiment with different possible positions of the bike rack handlers.

As evident in the resulting artifacts, all the participants could effectively communicate their design ideas, goals, and intentions. None of the participants used pen and paper for planning the design, and started directly sketching in DreamSketch. All the participants used *constraints*, *design variables*, and *obstacles*, which indicates that they found those features useful. Overall, the range of artifacts designed by the first-time users of DreamSketch demonstrate the ease of usage, expressiveness, and effectiveness of our tool.

LIMITATIONS & FUTURE WORK

Our participants also pointed out to some limitations that warrants discussion and future explorations.

Simulation & Execution time

The execution time for our participants' artifacts ranged from 30~60 minutes. As such, they were not able to iterate through the designs during the sessions. Making generative design tools faster and efficient is an important direction of future work. For the current work, we did not take advantage of parallel or cloud computing to simultaneously perform the topology optimization jobs for a given problem. However, such strategy can be used to cut down the generation time. We also intend to integrate other forms of simulation tools (i.e., fluid dynamics) to couple concept sketches with a wider range of computational capabilities. Additionally, one can potentially visualize the progression of a low-resolution version of GD algorithms in the canvas, so that the designer can iterate upon the problem definition during generation. The visualization and exploration of numerous other sub-optimal solutions is also worth exploring.

Input & sketching

The two participants (*P4*, *P7*) with no prior 3D design experience commented that initially they were not too comfortable with multi-view sketching in 3D, both conceptually, as well as with 3D navigation tools. Single-view [39] sketching techniques, or AR and VR tools might alleviate some of these navigational and conceptual barriers. Beyond polygon paths, we would like to accommodate more parameters as design variables to fully realize the creative opportunities of generative design.

Currently, designers can control the shape of the solution geometries by defining *obstacles*, or negative spaces. One of our future goals is to enable the GD algorithms to consider aesthetics preferences as a joint optimization problem [24], where the designer can specify preferred shape and style via sketching. However, we do not always see the surprising outcomes as a negative, as even designers with pre-existing aesthetic goals may appreciate seeing alternative solutions. Additionally, designers have some level of control over the design freedom that the algorithm has. Designers can specify more details to generate more predictable outputs.

There is a long-standing body of work in Sketch Based Interfaces for Modeling (SBIM). Overall, our goal is to explore and demonstrate how the qualities of freeform sketching can be integrated in SBIM, and couple them with the computational aspects of 3D design. Currently, our design and implementation choices try to balance between fluidity and fidelity, but fully preserving the merits of sketching in such a system is a challenge.

Representation

An important aspect of our work is the juxtaposition of sketches and 3D meshes, and we are motivated to blur the boundaries between these two representations even more. In the future, we want analyze the 3D strokes to better infer contextual information and user intents. This would enable us to provide knowledge-based tools to the designers [12]. It would also be interesting to explore unified editing tools for both sketches and 3D meshes.

CONCLUSION

DreamSketch offers a new design workflow that enables the designer to design and explore a range of functional 3D designs by sketching the design intentions. In this paradigm, the designer expresses a design problem via sketching and the computer helps the designer explore solution ideas that may have not been considered by the designer. Our work aims to accommodate the freeform qualities of traditional media (pencil and paper) with generative design technologies – by supporting *ambiguities* with *design variables*, *incompleteness* with *proxy geometries*, and the *expressive* aspects of drawings with juxtaposing strokes and meshes in a hybrid medium. Our design study session demonstrates that DreamSketch encourages *engineers* to think more like *designers*, and *designers* to think more like *engineers*. We hope such design tools would blur the boundary between form and functional aspects of design.

REFERENCES

1. Allaire, G., Jouve, F., & Toader, A. M. (2002). A level-set method for shape optimization. Comptes Rendus Mathematique, 334(12), 1125-1130.
2. Andre, A., & Saito, S. (2011, August). Single-view sketch based modeling. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling* (pp. 133-140). ACM.
3. Arora, R., Darolia, I., Namboodiri, V. P., Singh, K., & Bousseau, A. (2016). SketchSoup: Exploratory Ideation using Design Sketches.
4. Attar, R., Aish, R., Stam, J., Brinsmead, D., Tessier, A., Glueck, M., & Khan, A. (2009). Physics-based generative design.
5. Bae, S. H., Balakrishnan, R., & Singh, K. (2008, October). ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology* (pp. 151-160). ACM.
6. Bae, S. H., Balakrishnan, R., & Singh, K. (2009, October). EverybodyLovesSketch: 3D sketching for a broader audience. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (pp. 59-68). ACM.
7. Baskinger, M., & Bardel, W. (2013). *Drawing Ideas: A Hand-drawn Approach for Better Design*. Watson-Guptill.
8. Bendsøe, M. P. (1989). Optimal shape design as a material distribution problem. Structural optimization, 1(4), 193-202.
9. Bendsoe, M. P., & Sigmund, O. (2013). Topology optimization: theory, methods, and applications. Springer Science & Business Media.
10. Buxton, B. (2010). Sketching user experiences: getting the design right and the right design. Morgan Kaufmann.
11. Dillon, M. R. (2010). Dynamic design: Cognitive processes in design sketching. *Indiana Undergraduate Journal of Cognitive Science*, *5*, 28-43.
12. Do, E. Y. L. (2005). Design sketches and sketch design tools. *Knowledge-Based Systems*, *18*(8), 383-405.
13. Dorsey, J., Xu, S., Smedresman, G., Rushmeier, H., & McMillan, L. (2007, October). The mental canvas: A tool for conceptual architectural design and analysis. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on* (pp. 201-210). IEEE.
14. Du, T., Schulz, A., Zhu, B., Bickel, B., & Matusik, W. (2016). Computational multicopter design. *ACM Transactions on Graphics (TOG)*, *35*(6), 227.
15. Goel, V. (1995). *Sketches of thought*. MIt Press.
16. Gross, M. D., & Do, E. Y. L. (1996, November). Ambiguous intentions: a paper-like interface for creative design. In *Proceedings of the 9th annual ACM symposium on User interface software and technology* (pp. 183-192). ACM.
17. Igarashi, T., & Hughes, J. F. (2001, November). A suggestive interface for 3D drawing. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (pp. 173-181). ACM.
18. Kallio, K. (2005). 3D6B editor: projective 3D sketching with line-based rendering.
19. Kazi, R. H., Chevalier, F., Grossman, T., & Fitzmaurice, G. (2014, October). Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (pp. 395-405). ACM.
20. Kazi, R. H., Grossman, T., Mogk, C., Schmidt, R., & Fitzmaurice, G. (2016, May). ChronoFab: Fabricating Motion. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 908-918). ACM.
21. Kenwright, B. (2012). Inverse kinematics–cyclic coordinate descent (CCD). *Journal of Graphics Tools*, *16*(4), 177-217.
22. Kim, Y., & Bae, S. H. (2016, October). SketchingWithHands: 3D Sketching Handheld Products with First-Person Hand Posture. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (pp. 797-808). ACM.
23. Lee, Y. J., Zitnick, C. L., & Cohen, M. F. (2011, August). Shadowdraw: real-time user guidance for freehand drawing. In *ACM Transactions on Graphics (TOG)* (Vol. 30, No. 4, p. 27). ACM.
24. Martínez, J., Dumas, J., Lefebvre, S., & Wei, L. Y. (2015). Structure and appearance optimization for controllable shape design. *ACM Transactions on Graphics (TOG)*, *34*(6), 229.
25. Murugappan, S., & Ramani, K. (2009, January). Feasy: a sketch-based interface integrating structural analysis in early design. In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (pp. 743-752). American Society of Mechanical Engineers.
26. Nishida, G., Garcia-Dorado, I., Aliaga, D. G., Benes, B., & Bousseau, A. Interactive Sketching of Urban Procedural Models. *ACM Transactions on Graphics (TOG)*, *35*(4), 130.
27. Olsen, L., Samavati, F. F., Sousa, M. C., & Jorge, J. A. (2009). Sketch-based modeling: A survey. *Computers & Graphics*, *33*(1), 85-103.
28. Paczkowski, P., Kim, M. H., Morvan, Y., Dorsey, J., Rushmeier, H. E., & O'Sullivan, C. (2011). Insitu:

sketching architectural designs in context. *ACM Trans. Graph.*, *30*(6), 182.

29. Parish, Y. I., & Müller, P. (2001, August). Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (pp. 301-308). ACM.
30. Piccolotto, M. A. (1998). *Sketchpad+ architectural modeling through perspective sketching on a pen-based display* (Doctoral dissertation, Cornell University).
31. Schmidt, R., Khan, A., Kurtenbach, G., & Singh, K. (2009, August). On expert performance in 3D curve-drawing tasks. In *Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling* (pp. 133-140). ACM.
32. Shao, C., Bousseau, A., Sheffer, A., & Singh, K. (2012). CrossShade: shading concept sketches using cross-section curves. *ACM Transactions on Graphics*, *31*(4).
33. Shao, T., Li, W., Zhou, K., Xu, W., Guo, B., & Mitra, N. J. (2013). Interpreting concept sketches. *ACM Transactions on Graphics (TOG)*, *32*(4), 56.
34. Suwa, M., & Tversky, B. (1996, April). What architects see in their sketches: Implications for design tools. In *Conference Companion on Human Factors in Computing Systems* (pp. 191-192). ACM.
35. Ulu, N.G, Kara, B.L., Generative interface structure design for supporting existing objects. *Journal of Visual Languages and Computing* (pp. 171-183).
36. Umetani, N., Igarashi, T., & Mitra, N. J. (2012). Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, *31*(4), 86-1.
37. Wang, M. Y., Wang, X., & Guo, D. (2003). A level set method for structural topology optimization. Computer methods in applied mechanics and engineering, 192(1), 227-246.
38. Wang, Y., Chen, Y., Liu, J., & Tang, X. (2009, June). 3D reconstruction of curved objects from single 2D line drawings. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (pp. 1834-1841). IEEE.
39. Xu, B., Chang, W., Sheffer, A., Bousseau, A., McCrae, J., & Singh, K. (2014). True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Transactions on Graphics*, *33*(4).
40. Zaman, L., Stuerzlinger, W., Neugebauer, C., Woodbury, R., Elkhaldi, M., Shireen, N., & Terry, M. (2015, April). Gem-ni: A system for creating and managing alternatives in generative design. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 1201-1210). ACM.
41. Zheng, Y., Liu, H., Dorsey, J., & Mitra, N. J. (2016). Ergonomics-inspired reshaping and exploration of collections of models. *IEEE Transactions on Visualization and Computer Graphics*, *22*(6), 1732-1744.

APPENDIX: PARTICIPANT GENERATED EXAMPLES

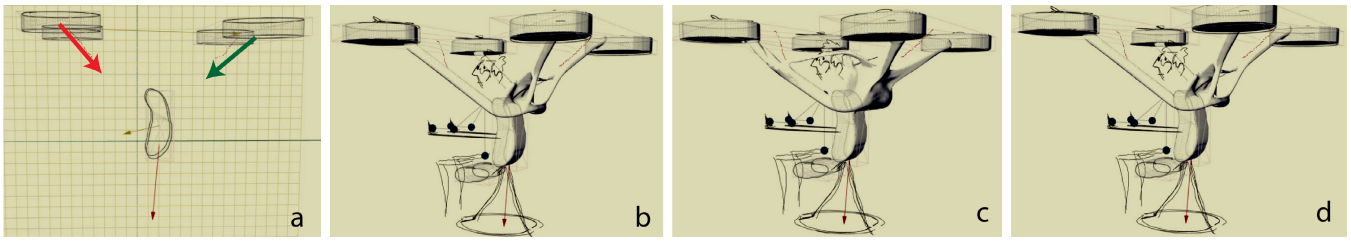

Figure 16: A drone designed by *P1*. (a) The problem definition consists of a single load, a design variable (red), and a constraint (green). The resulting solutions (b-d).

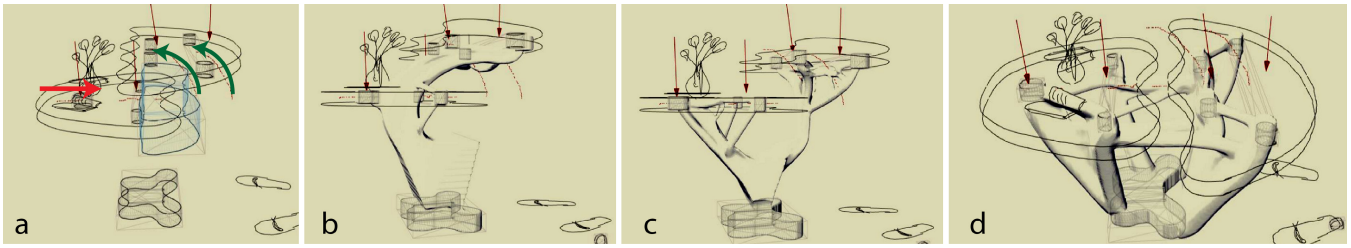

Figure 17: A table design by *P2*. (a) The problem definition. (b-d) The resulting solutions.

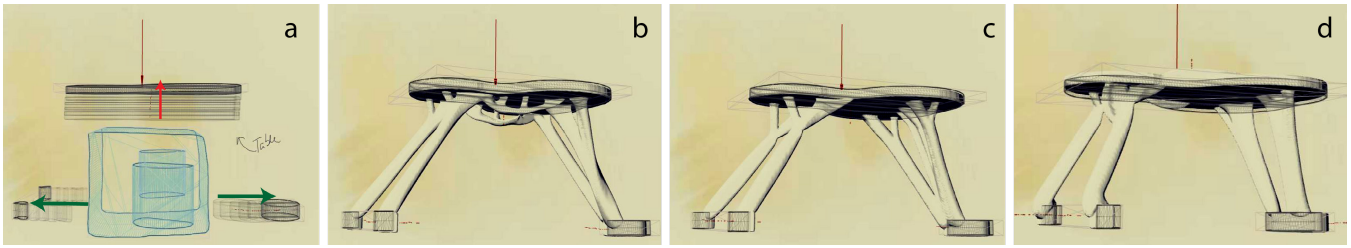

Figure 18: A table design by *P5*. The *design variable* and *constraints* defines the height and width variations (a). The resulting solutions (b-d).

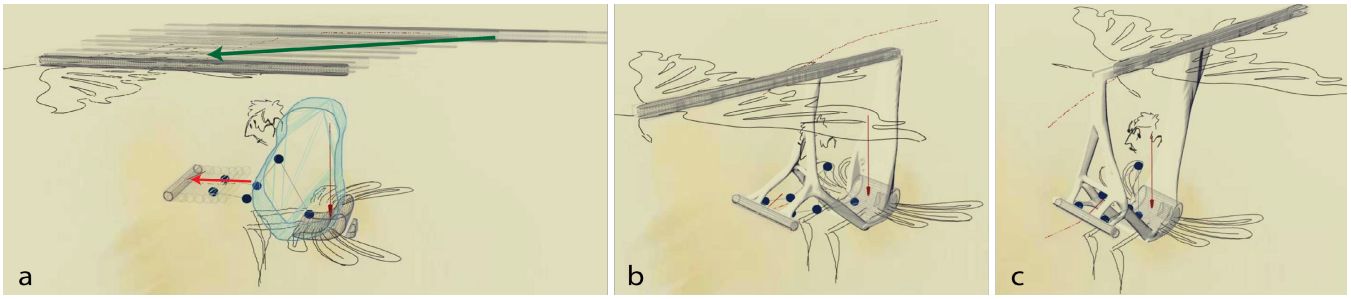

Figure 19: Glider design by *P7*. The manikin is used to define a *design variable* in the problem definition (a), and resulting solution variations (b-c).

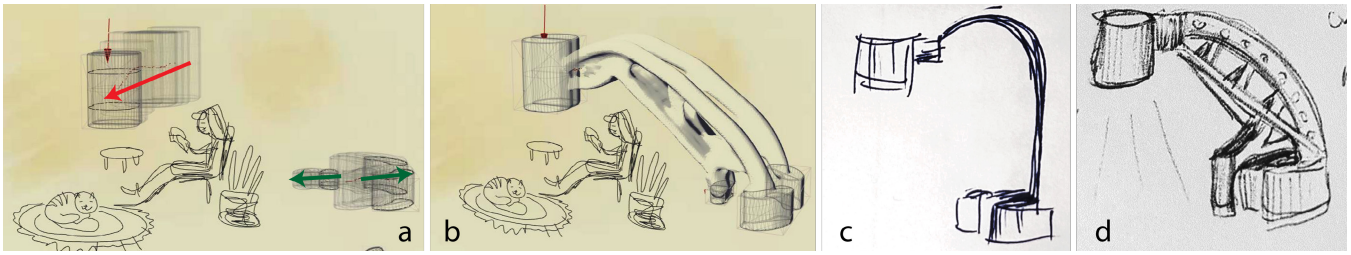

Figure 20: Reading lamp design by *P4*. The problem definition (a), and one of the resulting solutions (b). The designer initially expecting a shape like (c). However, the generative design solution inspired her to think in new dimension, and modify the design to bring more "industrial" look into the design.

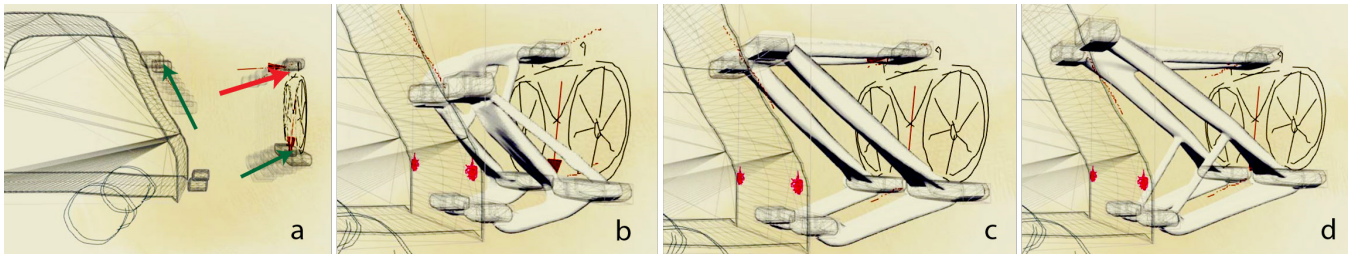

Figure 21: Bike-rack design by *P7*. The problem definition consists of one *design variable* and two *constraints* (a). The resulting solutions (b-d).

Figure 22: 3D printed artifacts designed by *P1* and *P2* using DreamSketch.