Computational Science Research

Industry Research

Life Sciences

Design & Social Impact

Complex System Research

Robotics Lab

Machine Learning

The Living

Simulation & Graphics Research

EXISTING

Research Studies

Parameters Tell the Design Story: Ideation and Abstraction in Design Optimization

Erin Bradner[1], Francesco Iorio[2], and Mark Davis[1]

[1]Autodesk Research
One Market, Suite 500
San Francisco, CA, 94105 USA
{erin.bradner, mark.davis}@autodesk.com

[2]Autodesk Research
210 King E. Suite 500
Toronto, ON M5A 1J7, Canada
francesco.iorio@autodesk.com

Keywords: Design Optimization, Optioneering, Ideation, Architectural Design, Engineering Design, Qualitative Research

Abstract

We report qualitative findings from interviews and observations detailing how professionals generate and evaluate design ideas using design optimization tools. We interviewed 10 architects and manufacturing design professionals. We frame our findings using the Geneplore model of creative cognition and classify examples of ideation and abstract design thinking arising from optimization workflows. Contrary to our expectations, we found that the computed optimum was often used as the starting point for design exploration, not the end product. We also found that parametric models, plus their associated parameters and simulations, serve as an alternate, highly-valued form of design documentation distinct from engineering schematics.

1. INTRODUCTION

We focus our research on optimization in a professional design practice. We select this focus because computational power that was once restricted to large government entities has become increasingly accessible to private entities (Thibodeau, 2011). Meanwhile, the professional use of parametric modeling has provided adaptive structures against which iterative simulations can be run. We've observed that computationally intense, heuristic searches of design spaces are becoming more commonplace in professional practice than in the past.

The proprietary nature of professional design processes, and their resulting designs, led us to choose an ethnographic approach to our research. While the use of simulation tools has begun to be systematically documented in the literature (e.g. Flager et al. 2013), to our knowledge there are no published, comparative accounts of post-hoc design and optimization used professionally. The qualitative findings reported here begin to fill this gap.

As software designers and high-performance computing experts, the high-level objective of our research is to expose the opportunity for new or improved computing architectures and user interfaces for generating, exploring and describing design spaces via optimization. In this study we advance toward that objective by first detailing how professional designers use design optimization to ideate – i.e., to generate and then explore solutions, to discover new and unexpected ideas, and to focus or expand their understanding through data visualization. We then articulate the multiple levels of abstraction engendered by optimization workflows, including: problem definition, evaluation, coding and documentation.

2. DESIGN OPTIMIZATION

Design optimization tools are creation tools that use parametric modeling, performance simulation and mathematical optimization to systematically generate and evaluate design alternatives (Holzer et al. 2007). Design optimization, also known as design optioneering (Gerber et al. 2012) and computational design (Abdill 2011), is a departure from traditional architecture and engineering practice. Typically, architects generate a relatively small set of design alternatives that represent specific points in a multi-dimensional design space (Flager and Haymaker 2007). In architecture, this small set of design alternatives may be communicated in the form of two or three foam physical cut scale models or a few dozen digital photo-realistic visualizations. Even with the support of state-of-the-art computer-aided design tools (CAD), individual designs are iterated relatively slowly and with considerable design effort. (Ibid.)

Conversely, architects and engineers using design optimization practices generate orders of magnitude more design alternatives by specifying design objectives in the form of design parameters and parameter ranges (Turrin et al. 2011). They use stochastic search methods, such as genetic algorithms, to automatically and iteratively compute large sets of design alternatives (Holzer et al. 2007). The designs that best fit the architects or engineer's predefined acceptance criteria survive multiple generations to spawn successive generations of unique, new designs.

Contrasting with traditional design practices, optimized designs are computed parametrically and lead algorithmically. The numerous design alternatives that are produced are often represented by a multi-dimensional plot of solutions and might be coupled with a matrix of thumbnails of rendered designs, as in Figure 1. Researchers investigating the approach argue that design optimization enables designers to "more efficiently, and with more certainty, explore complex and tightly coupled design solution spaces" (Ibid.) than traditional design practices.

Figure 1. Sample Design Optimization Results from (Sedlacz et al 2012).

2.1. Design Optimization as Creativity Support

Design optimization tools are an under-researched class of creativity support tools. Professional engineers and architects perform what we dub constrained creativity: their creative ideas must perform an intended function, and (usually specific performance criteria (Shah et al. 2003). These professionals concern themselves with how to generate and explore ideas that are optimized against multiple, competing objectives, variables or constraints. In building design, for example, it is necessary to simultaneously consider multiple complex objectives including site utilization, structural design, building form, energy use, buildability, and operating costs.

To the extent that design optimization tools enable users to be "not only more productive but also more innovative" (Shneiderman et al. 2006), they fall into the application category of creativity support tools. Creative activities that lead to innovation include idea generation, easy exploration, rapid experimentation and fortuitous combinations. (Ibid.) Designers using design optimization tools combine ideas through the algorithmic generation of design alternatives, they experiment through adjusting design parameters, and they explore through examining data visualizations of solution sets. Optimization tools do not simply automate idea generation. They support the creative process of idea generation, exploration and refinement.

2.2. Design Optimization and Creative Cognition

Engineering and architectural design are inherently generative disciplines. The set of designs algorithmically computed alternatives for any given real-world design task is vast. Flager et al. calculate that for the one story steel-frame building used in their energy and structural optimization study there were 95^{20} possible solutions (Flager et al. 2009). Cognitive limitations prevent humans from imagining even a small fraction of the possible alternatives in a problem with high dimensionality and millions of solutions. Not even the most skilled designer can handle this level of mental complexity. Variable interactions are particularly difficult to imagine: How do window size, glazing type and building orientation interact to produce the most energy efficient building? Furthermore, time constraints prevent designers from exhaustively exploring the solution set for top performing solutions.

In our qualitative research we observed that both architects and engineers explore a wide range of design parameters and constraints by applying iterative design techniques to "solve" for their designs. Geneplore is a heuristic model of creative cognition (Finke et al. 1996) that we found particularly useful in framing the design

3.2. Ideation

Design ideas from architecture and engineering are the product of creative ideation whereby ideas are generated and evaluated for suitability using qualitative and quantitative evaluation criteria (Shah et al. 2003). As indicated by the Geneplore model (Ward et al. 1995), there are two distinct activities: generation and evaluation. The traditional ideation process is iterative (Flager and Haymaker 2007). When design optimization techniques are applied, the design process becomes less iterative yet more design candidates, by several orders of magnitude, are produced. One participant described the difference in the two processes in these words:

"The typical design workflow is to design then throw to the analyst. Redesign. And then keep playing catch. It's inefficient. [Design optimization] captures the criteria that are important to you then have the cloud process all the permutations."

Figure 3 Use of Design Optimization

The starting point in the design optimization process thus shifts from specifying geometry in a CAD system:

design → evaluate → select or redesign

... to defining the design problem:

define → generate + explore → select or redefine

Next we'll outline the activities of generating, exploring, describing, and selecting solutions, as summarized in Figure 3.

3.2.1. Generate Solutions

Architects and engineers often cannot compute nor comprehend the effects of combining multiple variables in a complex system. Design optimization tools procedurally generate design solutions from a machine-readable definition of the design constraints and parameters. Typically, the design optimization system returns a large number (100s to 1000s) of design solutions. One study participant explained this automated idea-generating process thusly:

"Good design has inspiration to it, if you have that vision you can encode it and parameterize it and explore it further. Now we have a rich library of options."

Examples of designs that our participants optimized include high-rise buildings, hospital patient care wards, an engine manifold, and installation art. The elements of the design that were optimized include, but are not limited to: aesthetic form, structural form, mechanical performance, human performance (views, walking distances) and building energy efficiency. *Glass per second* is an example of a design parameter from the engine manifold example. Engineering constraints are such things as *"no more than 3 bars of pressure drop."* An architectural constraint might be *"must maintain an average of 60% natural light."*

3.2.2. Explore Solutions

We appropriate the term *explore* from the Geneplore model of creative cognition to describe the way users purposefully adjust their parameter ranges and review the resulting solution sets. One user referred to this as an act of discovery: *"The key is the discovery phase. As you adjust ranges, you discover options you may not have expected."* Surprisingly to us, both architects and engineers in our study said that the computed optimums was often used as the starting point for this exploration. In one case, the structurally optimal alternative design for a high-rise building was used to seed the exploration process for its aesthetic design. The structural engineer from this group explained that, *"the optimum is a way to talk about form,"* indicating that the solution they chose from the set of high-performing solutions in their structural optimization served as the conceptual sketch on which the aesthetic designers could iterate.

3.2.3. Describe the Solution Space

The solution space is the set of all solutions computed by the design optimization algorithm. We observed two primary ways that design optimization describes the solution space for users. There are visualizations and data plots. Visualizations can be further classified into performance visualizations (e.g., simulations) and aesthetic visualizations. A heat map showing regions of stress on a 3D model of a metal bracket under a simulated load is an example of a performance visualization. Our study participants explained that they use visualizations to develop their understanding of the impact of their design choices on aspects of performance. In one case, a mechanical engineer optimizing the design of a manifold attributed a major design insight to a computation fluid dynamic visualization he reviewed:

"[the visualization] gave us the insight to open the diameter... the diameter of the opening could not be expanded but the diameter of the shaft could be flared to increase flow."

Aesthetic visualizations are also 3D renderings of design alternatives. When used by architects, these visualizations allow systematic exploration of building form. These visualizations are important since, as one structural engineer put it, *"slight variations in form, in a designer's eye, are either elegant or [illegible]."*

Data plots, commonly in the form of scatter plots, are another type of data visualization we observed being used. Respondents expressed a range of opinions about the utility of different types of plots. One study participant sang the praises of parallel coordinate charts while another pledged his allegiance to Pareto plots. A manager said this about the computational designers he manages and their discussions of different plot types: *"they talk about tradeoffs in a spirited way."* We suspected that different plot types allowed certain designers to extract more information about design tradeoffs more quickly than others. We theorized that there was an interaction between the sophistication of users' statistical knowledge and the utility they assign to the different types of plots.

3.2.4. Describe the Design Space

The term design space describes the mathematical definition of the design. It includes the design variables, and constraints, the ranges or discrete values assigned to these variables and any other bounding criteria that can be expressed mathematically. Study participants used statistical analyses such as correlations to examine the interaction between variables. They actively sought these statistical descriptions of the design space because *"understanding dependencies between different systems [illegible] is very challenging"* and because statistical analyses gives them insights to add or remove variables when warranted. For example, they remove variables found to be highly correlated to reduce the "dimensionality" of their problem space, they split variables based on variable sensitivities, and they adjust parameter ranges up and/or down to account for design tolerances.

3.2.5. Select Solutions

Something we found counterintuitive was that many participants in our study selected the highest performing solution from a optimizer as to initiate their design ideation process, rather than to conclude it. This remained befuddling until we learned that our participants were using computational design more frequently in the earliest, most conceptual phase of design than in any other phase. They considered the optimizer to be the computational equivalent of a back-of-the-napkin sketch. Borrowing again from the Geneplore model of creative cognition, the computed optimum is the pre-inventive structure that seeds the ideation process: *"instead of starting with nothing, you start with something...your optimum gives you a starting [illegible]."*

That said, it was not only the single highest solution that was important and useful, but often the full set of high performing solutions. Our study participants reported consulting Pareto plots (tradeoffs) in the conceptual design phase to rapidly identify and select "interesting" solutions. Pareto plots are a type of scatter plots commonly used to distinguish high-performing solutions in the set of all feasible solutions. The Pareto set is composed exclusively of the high performers: for any given solution in the Pareto set, it is impossible to increase performance along one axis without decreasing performance along another axis. A mechanical designer qualified his interest in identifying the highest performing designs during the conceptual design phase by explaining that he is looking to determine *"what direction the performance is trending."* He iteratively generates high performing solutions using visualizations such as Pareto plots to explore how adjusting the parameters by hand affects the overall quality of the resulting solutions.

3.3. Abstraction

To define a design problem requires abstracting the problem into mathematical descriptions such that the design optimization tool can compare alternatives. An empirical study of engineers has shown that the level of abstraction and precision in a problem definition affects the quality and quantity of ideas generated ([illegible] 1999). Problems defined with low precision produce few and poor solutions because engineers fixate on concrete solutions too early. Conversely, too many solutions also produce poor solutions because managing too many solutions diminishes the engineers' ability to identify, evaluate and modify the best solutions. Successful designers balance their search for solutions. ([illegible])

We did not evaluate the quality of design solutions our study participants produced by design optimization tools; however, our interviews with architects and engineers suggest that design optimization tools maximize the positive effects of precision in problem definition while minimizing the negative effect of large solution sets. Furthermore, we suspect that the demands of abstracting the problem into a precise definition, which our participants stress is nontrivial, improves the designers' ability to evaluate solutions. We discuss multiple levels of abstraction in the design optimization workflow next.

3.3.1. Problem Definition as Abstraction
"Think about the underlying logic and how you can transform it."

Using design optimization tools necessitates abstraction. Users must precisely define the problem space across multiple dimensions. One participant succinctly summarized the design problem for an office tower this way:

"I want to balance aesthetics and sustainability. We wanted a cohesive skin. [goal] The client wanted 40% natural light. [constraint]"

One participant defined between 12 and 44 design parameters for the range of projects they worked on. They explained that it was not sufficient to simply describe the building or object geometry to the problem definition; rather, they needed to extract the underlying logic of their design problem. They routinely need to abstract variables so they can "optimize things that aren't the same" such as energy efficiency and structural efficiency. Designers also report that they often look at multiple competing criteria at the same time, such as daylight exposure, and view quality from windows in a building. Abstracting design goals and parameters into quantifiable representations is the first level of abstraction that design optimization demands from users.

3.3.2. Code as Abstraction
"You need to write the rules correctly."

The second level of abstraction required by design optimization is programming. Most, but not all, of the design optimization problems that our study participants work on require that they script algorithms to procedurally generate solutions. In the case of hospital design, the design goals and parameters interact according to a specific logic embedded in the optimization code, since hospital design is subject to rigorous building codes and regulations.

One participant explained the challenge of converting building code to software code in these words: *"you need to write the rules correctly for what you are trying to check."* In one architectural case 3000 lines of code produced 3000 design alternatives, each satisfying rules for solar gain, cost, buildability and acoustics.

When architects abstract designs into rules and make observations such as *"you need to write the rules correctly,"* they begin to sound more like software programmers than architects. One of our interview participants rejected design optimization precisely because of this level of abstraction it demands. His view is that design optimization transforms the practice of architecture away from the "minimal" or concrete practice of creating CAD drawings to a more abstract practice:

"[illegible] [in CAD tool] is minimal. When you're working in [illegible], your goal is set. I'm designing a building. I'm producing construction documents. In Grasshopper [a design optimization tool] there is no prescribed approach nor outcome. I can design a tree or a car."

This study participant asserted that when architects use design optimization to produce *"1,000 variations, you're not being an architect any more. You are a computer programmer, a box driver."* This comment implies that the level of abstraction inherent in the coding aspect of design optimization fundamentally alters architecture from a practice of documenting design into CAD, to a practice of abstracting rules into software.

3.3.3. Evaluation as Abstraction
"Then they start to push each other around."

Sound decisions about tradeoffs in multi-objective design problems require sophisticated statistical thinking. As we noted earlier, even skilled designers often struggle to comprehend the effects of combining multiple variables in a complex system. A particularly germane example from our data came from an interview with a structural engineer. He told us of output from design optimization that combined structural strategies in an interesting and unexpected configuration. Design optimization had produced a design wherein the widest section of the building he was designing had a tied-line structure with cross-bracings tapering as they went up, while two narrower sections used two different structural strategies: compression and truss. When we asked for his reaction to the configuration he replied: "you have these strategies in your mind, but you may not know how they will interact." The abstractions in this example are the structural strategies. The design optimization system has no formal definition of a tied-line, compression or truss. It simply computes the thickness and position of the beams. The designer abstracts what he sees in the output into structural strategies.

One transcript from an interview with an architect/artist who designs and fabricates installation art is explicit with abstract criteria that he used to evaluate the results from the optimization. Examples of these criteria are: minimal surfaces, the absence of double curvature, and "mathematical purity". To visualize the design optimization process he output the calculations directly to a 3D model and watched the model transform as the problem was being solved. At the start, the design was very angular but once the design optimization ran, the geometric elements started to "push each other around to find equilibrium."

These two examples imply that evaluations may be concrete, such as the percent of solutions that meet or exceed 60% natural light, or they may be abstract, such as a structural engineering strategy or a mathematical purity.

3.3.4. Documentation as Abstraction – "Not a perfect translation of genotype to phenotype"

We were surprised to learn the extent to which architects abstracted their designs. One participant stressed that the "parameters need to tell the design story." An architectural firm we interviewed described the process by which they encoded the expression of architectural forms and performance parameters into an "architectural genome." Another architectural firm referred to the "master files" that they produced using their design optimization practice. These types of files, genomes and master files, are the symbolic descriptions of the design problem. These files are the design story as told by the parameters. This type of documentation is a fundamental departure from the set of elevations and floor plans in traditional design. The obvious analogy, used by one participant, is that the master file is a genotype and the set of CAD drawings is a phenotype. One study participant cautioned us that there is "not a perfect translation of genotype to phenotype" implying that design optimization does not replace the expertise of the architects.

Design optimization requires that professionals abstract the design problem far more systematically and comprehensively than in traditional, iterative design processes. As a result of this obligatory abstraction, design optimization has enabled design professionals to generate and explore more mathematically complex design alternatives than they would otherwise. This process of ideation through abstraction appears to be the creative engine of design optimization. That said, abstract thinking is difficult even for trained professionals. One participant reported that plots of solution sets that are output from design optimization are more difficult for him to evaluate than CAD models: "A [CAD model] is something you can respond to…I like that. I don't like that." This was echoed by another respondent:

"The readability of the output is not there… even the visual examples are hard to digest for us, much less for the clients who don't know the expertise."

We were surprised to find this pattern in our interviews: user interfaces for data exploration in design optimization produce a poor user experience. This finding led us to consider what attributes of these plots contributed to the poor user experience. At a cognitive level, we also wanted to learn how architects and engineers conceptualize the statistical analyses in design optimization, yet we have no data in our interviews on this topic. In a future study we plan to develop a framework describing how architects and engineers use data visualizations and apply that to prototype user interfaces for exploring solution spaces.

4. CONCLUSION

This research describes the use of design optimization tools across multiple professional disciplines including architecture and engineering. It leverages a creative cognition model of ideation (Ward et al. 1995) to frame the activities we observed, including: generating design solutions, evaluating design solutions and describing both the problem and solution space. It examines an understudied creativity support tool, design optimization, and it articulates the role ideation and abstraction play in design optimization. One key finding is that that professionals use design optimization to gain understanding about the design space, not simply to generate the highest performing solution. Professionals reported that the computed optimum was often used as the starting point for design exploration, not the end product. A second key finding is that in some design organizations parametric models plus their associated parameters and simulations are serving as an alternate, highly valued form of design documentation distinct from engineering schematics. Much like a genome encodes a genotype, a parametric model and associated simulations encodes the expression of form and performance across any given set of parameters. The parameters tell the design story.

A next step for our research is to develop user interface design principles and prototypes that remove abstractions between solution sets and individual design solutions. User interfaces should enable users to easily pivot between exploring a variation set and examining a specific solution. They should also map decision-making criteria, such as parameter values and variable interactions, directly onto 3D models of individual designs. For any given design there are numerous nearby variations that are generated by incrementing up or down a variable range. We are exploring various user interfaces to present the detail of individual solutions while preserving the context of nearby solutions.

There are two distinct but related reasons why design optimization is important to the research community. First, this class of tool suffers from poor user experience (Flager 2007). One professional users reported this in interviews, and expert academic users report "the lack of user-friendly, mature and comprehensive user interfaces limits the usage in practice." (ibid.) Poor user experience is due in part from a dearth of user experience research in the field of design optimization (Flager and Haymaker 2007). With this study, we hope to ameliorate the lack of research in this area.

Secondly, this research is important because users of design optimization are making buildings more structurally sound with less building material; they are making automobile engines more efficient, and they are improving the quality of care in hospitals by making them more comfortable places to work. Design optimization tools are used for sustainable designs, yet are built on stratified layers of abstraction that in turn place considerable cognitive demands on the user. Abstractions exist when the design problems are defined, coded, and interpreted through data visualizations. Our findings suggest that the abstractions inherent in the workflows may conceptually distance designers from their designs, or lead them to make important design decisions based on incomplete information. This research may give tool developers, ourselves included, insight into how to simplify, focus and/or minimize these multiple levels of abstraction when designing optimization software.

Acknowledgements

We thank Michael Haupt for his coaching on design optimization techniques.

References

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Proceedings of the ASME 2016 International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE 2016
August 21-24, 2016, Charlotte, North Carolina

DETC2016-59627

EMBEDDED SENSORS AND FEEDBACK LOOPS FOR ITERATIVE IMPROVEMENT IN DESIGN SYNTHESIS FOR ADDITIVE MANUFACTURING

Mehdi Nourbakhsh
Autodesk Inc.
San Francisco, CA, USA

Nigel Morris
Autodesk Inc.
Toronto, Ontario, Canada

Michael Bergin
Autodesk Inc.
San Francisco, CA, USA

Francesco Iorio
Autodesk Inc.
Toronto, Ontario, Canada

Daniele Grandi
Autodesk Inc.
San Francisco, CA, USA

ABSTRACT

INTRODUCTION

is that they do not generate solutions based on usage feedback, which is crucial for iterative improvement of products.

With widespread availability of IoT technologies, product designs can now be cheaply and effectively instrumented to gather data on usage patterns over long time spans and large amount of users. Therefore, new synthesis methods should support leveraging sensor data and generating novel solutions not only for one generation of product design, but also for extensions of a product lifecycle, improving the correlation between analysis models and real performance metrics. To this end, the objective of this study is to demonstrate a data-augmented design synthesis approach that, based on usage pattern gathered from sensor data, generates numerous alternative solutions targeted at additive manufacturing.

RELATED WORK

Developing means of generating a large number of alternatives that satisfy design requirements is known as design synthesis [1]. To synthesize design problems, researchers mainly focus on generating solutions based on functions, grammars, or analogical knowledge. In function-based synthesis, designers decompose the intended functionality of a design problem into sub-functions and generate conceptual solutions that satisfy product functionality [8] [13]. In grammar-based design synthesis, designers define a language of design, including vocabulary, and rules to transform their initial design into various novel solutions [9] [12]. The analogy-based design methods (e.g., biologically inspired design) develop solutions by drawing inspiration from previous design knowledge [5] [11].

Product design using design synthesis methods requires three main steps (Figure 1). First, designers need to appropriately define their problem. Then, they should select and apply appropriate synthesis techniques to their problem and generate a space of solutions that satisfies all of the design requirements. Finally, designers need to select and fabricate their product at the end of the design process.

Figure 1: Current synthesis methods

A number of studies focus on improving the synthesis methods for designing products. For instance, developing various grammar rules for creation of function structures in product design [14], creating rules for the design of sheet metals [15], defining vocabulary and rules for automating satellite design [7], and improving the quality of design rules by easily step focus in the development phase rather than during their application [6]. These studies exhibit three fundamental limitations. First, they have been demonstrated to be effective on a single generation of the product design cycle. In practice, most successful products have multiple generations and evolve from one generation to the next. Design synthesis based on rules and grammars may produce a completely different design in each generation, which may lead to increased uncertainty in the design selection process. Second, applying grammar rules or creation of function structures requires tools and expertise that is not generally available in the current product design practice [5]. Finally, current synthesis methods cannot directly incorporate any usage feedback of the designs they generate: in each product generation, designers manually analyze usage feedback data and based on their intuition attempt to incorporate improvements to their products. At time of writing, we could not find in the literature any synthesis method capable of generating improved design solutions based on the usage feedback of products.

PROPOSED METHOD

The method proposed in this study extends the current design synthesis techniques to be applied beyond the conceptual design stage for improving multiple generations of products based on usage feedback (Figure 2). Given an initial product design, our method requires putting it through sets of controlled design trials to capture its performance and behavior characteristics. Data are collected using both sensors that are already embedded in the design and additional sensors applied to the design ad hoc for the controlled experiments. In addition, a 3D model of the design is captured and used to identify locations of boundary conditions and elements for the design space. Next, the usage data are analyzed for the problem definition stage in which users define their high-level engineering problem as a collection of design requirements and constraints. In the next step, an ensemble shape synthesis algorithm, composed of topology, and beam-based optimization algorithms, synthesizes various geometries based on the combinations of volume constraint and number of iterations in the topology optimization with density and degree of connectivity in the beam-based optimization. Designers then select and fabricate their design, which forms part of the input for the next design generation.

Figure 1: Proposed synthesis method

IMPLEMENTATION OF THE PROPOSED METHOD

[illegible]

[illegible]

Controlled design trial

[illegible]

[illegible]

Figure 2: High resolution mesh created from laser scan of initial chassis design

Sensor Data Acquisition

[illegible]

[illegible]

Figure 3: Strain gauge sensors as installed in suspension area of initial chassis design

Data Analysis and Translation

Problem definition

Design Synthesis

Synthesis Algorithm

Figure 7: Design Synthesis Flow

[illegible]

[illegible]

[illegible]

Manufacture of Improved Design

[illegible]

RESULTS

[illegible]

Data Acquisition

[illegible]

Table 1. Chassis loads in one of the load cases

	Front/ Back of Car	Upper/ Lower A-Frame	Front/Bac k Set of A-Frame	Max Load (kN)	Min Load (kN)
1	Front	Upper	Front	181.2	-11.6
2	Front	Upper	Back	122.8	-8.8
3	Front	Lower	Front	-74.8	-67.2
4	Back	Lower	Back	-182.5	-2247
5	Back	Tie Rod	–	-15.4	5.6
6	Back	Shock Strut	–	67.2	-5.8
7	Back	Upper	Front	-27.3	10.7
8	Back	Upper	Back	-90.6	-27.9
9	Back	Lower	Front	40.1	6.9
10	Front	Lower	Back	24.3	48.0
11	Front	Steering Arm	–	-10.3	-67.3
12	Shock	Shock Strut	–	-21.8	-23.6

These loads and the geometry of parts were used to define the design problem (Figure 8). Some loads and geometries were combined in order to simplify the problem definition.

Figure 8. Problem definition

Generative Synthesis Methods

Here we demonstrate the results of our synthesis method. The intermediate stages of synthesis and final selected design are shown in Figure 9 and 10. The final design reduced the weight of the initial design from 739 lbs down to 702 lbs.

Figure 9. Results of the topology optimization for n=52 and n=99

Figure 10. Beam optimization results for selected design. Top: Initial beam network (1398s). Bottom: Final optimized beam network (2838s).

Lessons Learned from Design Synthesis

[illegible]

[illegible]

[illegible]

Lessons learned from the Manufacture of Improved Design

[illegible]

CONCLUSION

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

REFERENCES

[1] [illegible], R. L. and Arciszewski, T. 2014, "Design Evolution: a [illegible] of many colors," Design Computing and Cognition'12, Springer, pp. 103-120.

[2] [illegible], J. 2004, "[illegible] propeller-driven aircraft configurations," American Institute of Aeronautics and Astronautics, St. Louis Section.

[3] [illegible], D. G. and Smith, E. M. 1991, "Design Evolution," Design Studies, 12(2), pp. 1-13.

[4] [illegible], J. 2011, "Overcoming Organizational Fixation: Creating and Sustaining an Innovation Culture," The Journal of Creative Behavior, 45(2), pp. 116-133.

[5] Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N. V., and Wood, K. L. 2011, "Computer-Based Design Synthesis Research: An Overview," Journal of Computing and Information Science in Engineering, 11(2), pp. 021003-021003.

[6] Hood, T. and Shea, K. 2016, "An interactive visual approach to developing and applying parametric three-dimensional spatial grammars," AI EDAM, 27(Special Issue 04), pp. 333-356.

[7] Schaefer, J. and Rudolph, S. 2005, "Satellite design by design grammars," Aerospace Science and Technology, 9(1), pp. 81-91.

[8] Helms, B. and Shea, K. 2012, "Computational Synthesis of Product Architectures Based on Object-Oriented Graph Grammars," Journal of Mechanical Design, 134(2), pp. 021008-021008.

[9] Königseder, C. and Shea, K. 2016, "Systematic rule analysis of generative design grammars," AI EDAM, 28(Special Issue 01), pp. 227-238.

[10] Lin, Y.-s., Shea, K., Johnson, A., Coultate, J., and Pears, J. "A method and software tool for automated gearbox synthesis," Proc. ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 111-121.

[11] Pahl, G. and Beitz, W. 2013, Engineering design: a systematic approach, Springer Science & Business Media.

[12] Shea, K. and Cagan, J. 1999, "Languages and semantics of geometrical discrete structures," AI EDAM, 13(04), pp. 241-251.

[13] [illegible], M., [illegible], K. D., and [illegible], D. 1989, "The structure-mapping engine: Algorithm and examples," Artificial Intelligence, 41(1), pp. 1-63.

[14] [illegible], P. and Campbell, M. I. 2005, "A study on the grammatical construction of function structures," AI EDAM, 19(03), pp. 175-210.

[15] Patel, J. and Campbell, M. I. 2010, "An Approach to Automate and Optimize Concept Generation of Sheet Metal Parts by Topological and Parametric Decoupling," Journal of Mechanical Design, 132(5), pp. 051001-051001.

[16] Autodesk, 2016, "Project Dreamcatcher," https://www.autodeskresearch.com/projects/dreamcatcher

[17] 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(1-2), pp. 227-246.

[18] [illegible], G., Jouve, F., and Toader, A.-M. 2004, "Structural Optimization by the Level-Set Method," Free Boundary Problems: Theory and Applications, P. Colli, C. Verdi, and A. Visintin, eds., Birkhäuser Basel, Basel, pp. 1-15.

[19] [illegible], F., [illegible], G., Auley, A., Shea, K., [illegible], J., and [illegible], M. 2014, "Fully-Constrained Design: A general and scalable method for discrete member sizing optimization of steel truss structures," Computers & Structures, 140(0), pp. 55-65.

[20] National Instruments, 2016, "NI DIAdem: Locate, Analyze and Report on Measurement Data," National Instruments.

INVESTIGATING THE USE OF CONTROLLED NATURAL LANGUAGE AS PROBLEM DEFINITION INPUT FOR COMPUTER-AIDED DESIGN

ABSTRACT

KEYWORDS

Computer-Aided Design, Natural Language Input, Controlled Natural Language, Problem Definition

1. INTRODUCTION

[illegible]

[illegible]

[illegible]

2. [illegible]

[illegible]

[illegible]

[illegible]

3.3.1. Semantic Categories

By semantic categories, we mean the range of problem definition statements that describe different aspects of the design problem. The categories include functions, objectives, and constraints of the design as well as properties of environment objects (Table 5). [illegible]

[illegible]

Functions	**Definition:** [illegible] **Example:** [illegible]
Objectives	**Definition:** [illegible] **Example:** [illegible]
Constraints	**Definition:** [illegible] **Example:** [illegible]
Properties of environ. objects	**Definition:** [illegible] **Example:** [illegible]

3.3.2. System

[illegible]

[illegible]

[illegible]

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

[illegible]

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

3.4. EXPERIMENTAL CONDITIONS

Each participant was randomly assigned to one of the two conditions (described below). A between-subjects design was used [illegible].

[illegible]

[illegible]	[illegible]	
[illegible]	[illegible]	[illegible]

[illegible]

[illegible]

[illegible] CONCLUSIONS AND FUTURE WORK

[illegible]

[illegible]

REFERENCES

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible] "Engineering Design: A Systematic Approach, 2nd Edition", Springer-Verlag, London, UK.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible] (1954). In addition, [illegible] and [illegible] [illegible] [illegible]

[illegible] of premium [illegible] information.

[illegible]

[illegible]
[illegible]
[illegible]
[illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]. The [illegible] [illegible] [illegible] [illegible] [illegible] the [illegible] [illegible] to [illegible] [illegible]. The [illegible] [illegible] [illegible] [illegible] the assumption that [illegible]

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] without any [illegible] or [illegible] provided. Twenty minutes were given for the first task. In the second task, participants were asked to [illegible]. If they

iii. On a scale of 1 to 5 [illegible] would you rate the usefulness of using the CM, [illegible]?

Percentage of [illegible] statements translated to [illegible]. For Task 2 of the NCM condition, participants [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible] natural language could not be translated into the CSL. Most of them described the non-functional design features as mentioned earlier. Subsequently, participants of the NL-CSL condition [illegible]

7. ACKNOWLEDGMENTS

We thank the Natural Sciences and Engineering Research Council of Canada for financial support.

[illegible]

[illegible] "[illegible]", [illegible]

SpaceAnalysis: A Tool for Pathfinding, Visibility, and Acoustics Analyses in Generative Design Workflows

ABSTRACT

Author Keywords

ACM Classification Keywords

1 INTRODUCTION

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

3.2 Discretization

If we compare the outcome of the basic method in Figure [illegible] with that of the enhanced method in Figure [illegible], we find that the same initial set of barriers can generate very different space lattices. For example, point [illegible] is accessible only when using the basic method, and point [illegible] is only accessible using the enhanced method. The enhanced method produces a room that encircles the barriers on the right-hand side of the environment, whereas the basic method eliminates this room.

The [illegible] tends to [illegible] with the enhanced method. This decision was based on the intuition that the enhanced method would more faithfully capture users' intentions when applied to real-world buildings and urban environments. It is worth noting that as long as any intended gaps between barriers are large (compared with the resolution of the space lattice grid) then the choice of discretization method may not matter. The initial layouts in Figures [illegible] and [illegible] are somewhat contrived so that the gaps between the barriers are deliberately small relative to the grid spacing. Nevertheless, it is helpful to demonstrate how the two methods handle small openings and narrow corridors where the grid resolution becomes important.

5.3 Experiment

Here we systematically compare the basic and enhanced discretization methods using the three scenarios shown in Figure [illegible]. In each scenario, we are testing whether one can travel from point A to point B, which involves traversing a certain type of narrow passage. In the Wall Opening Scenario, the passage is a gap or doorway in the middle of a straight wall. In the Wall Junction Scenario there is also a gap or doorway, but the passage is situated at the corner of a T-junction formed by two walls. In the Corridor Scenario, the passage is the entire region between two parallel walls, a region that contains the two points.

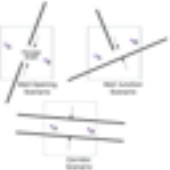

Figure 6: Scenarios used to compare discretization methods.

For each scenario, we varied the passage width from [illegible] to [illegible] in increments of [illegible], where the unit of measure is the resolution of the space lattice grid (i.e. the grids shown in Figures [illegible] and [illegible]). To clarify, suppose the grid resolution is [illegible] cm. In that case, a passage width of [illegible] means the passage is [illegible] cm wide, and a passage width of [illegible] means the passage is [illegible] cm wide.

For each scenario and passage width, we ran [illegible] tests to see whether one could travel from point A to point B. For each test, we rotated all of the geometry by a random angle, and translated it by a random displacement. Once the [illegible] tests were run, we recorded the traversal rate: the fraction of tests where travel was successful. For extremely small passage widths, all routes from A to B were certain to be cut off regardless of how the scenario was rotated or translated. In these cases, all tests failed and the traversal rate was 0. For large passage widths, it was certain there would be a path from A to B through the passage. In these cases, all tests succeeded and the traversal rate was 1. But for certain passage widths in between, the orientation and placement of the geometry relative to the grid determined whether a route could be established or whether it was cut off. In these cases, the traversal rate was between 0 and 1.

The entire set of trials ([illegible] scenarios × [illegible] passage widths × [illegible] tests) was repeated for both the basic and enhanced methods. The results are plotted in Figure [illegible]. For each scenario and discretization method, the curve has two notable passage widths: the minimum width where at least one test was successful, and the maximum width where all tests were successful. These widths are listed in Table 1.

Scenario	Minimum Width		Maximum Width	
	Basic	Enhanced	Basic	Enhanced
Wall Opening	[illegible]	[illegible]	[illegible]	[illegible]
Wall Junction	[illegible]	[illegible]	[illegible]	[illegible]
Corridor	[illegible]	[illegible]	[illegible]	[illegible]

Table 1: Passage widths at which traversal is found to be possible (minimum width) and guaranteed (maximum width).

There are two criteria we should use to evaluate the methods. First, the incline of the curves in Figure [illegible] should be as steep as possible. Ideally the curves would be step functions, meaning that travel from A to B would depend only on the passage width and not on the rotation or translation of the geometry relative to the grid. The second criterion is that the incline should be as far to the left as possible. The further the incline is to the left, the coarser the resolution one can choose while still ensuring passage through small doorways and narrow corridors. Coarser resolutions mean fewer grid points, faster analyses, and potentially better experiences for users interactively exploring numerous design options.

Based on these criteria, the enhanced method clearly outperforms the basic method in the Wall Junction and Corridor scenarios. The inclines are narrower and further to the left. With the enhanced method, a resolution of [illegible] cm essentially guarantees that travel paths will connect through passages of at least [illegible] cm. The basic method would require a finer resolution of [illegible] cm to ensure all [illegible] cm gaps are traversable.