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 300
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, higher-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, 2013). 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. Toghiani et al. 2013), to our knowledge there are no published, comparative accounts of goal-driven 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 (Attoff 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 (Gagliano et al. 2013). They use stochastic search methods, such as genetic algorithms, to automatically and iteratively compute large sets of design alternatives (Deb 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 feed 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 3. 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 3. Sample Design Optimization Results from Gerber et al. 2012.

2.2. Design Optimization as Creativity Support

Design optimization tools are an under-researched class of creativity support tools. Professional engineers and architects perform what is dub constrained creativity: their creative ideas must perform an intended function, and satisfy 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.

2.3. Design Optimization and Creative Cognition

Engineering and architectural design are inherently generative disciplines. The set of design algorithmically computed alternatives for any given real-world design task is vast. Flager et al. calculates that for the one storey steel-frame building used in their energy and structural optimization study there were 9^12^ 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 turn the cloud process all the permutations."

Figure 3. Use of Design Optimization.

The starting point to 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 in it, if you have that vision you can encode it and parameterize it and explore it further. Now we have a rich flow 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 (many walking distance) and building energy efficiency. *Liters per second* is an example of a design parameter from the engine manifold example. Engineering constraints are such things as *"no more than 5 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 optimum was often used as the starting point for the 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 computational fluid dynamics 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, to 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 spiritual 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 those 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 (variables space) is very challenging,"* and because statistical analysis gives them insights to add or remove variables where 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 solution set 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 phases of design than in any other phase. They considered the optimum 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 launch."*

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 [illegible] in the conceptual design phase to rapidly identify and select "interesting" solutions. Pareto plots are a type of scatter plot 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 (Fiolo 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. (Ibid.)

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 effects 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.

3.2.1. 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]"

Our participants defined between 12 and 40 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 in 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 cover 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.

3.2.2. 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 5000 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 abstractions 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 the level of abstraction it demands. His view is that design optimization transforms the practice of architecture away from the 'rational' or concrete practice of creating CAD drawings to a more abstract practice:

"Novel [in CAD world] is rational. What you're working in [Revit], your goal is set. I'm designing a building. I'm producing construction documents. In Grasshopper [in 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.

3.2.3. 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 steel-like 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 truss, 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 installations set to explore 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 8% natural light, or they may be abstract, such as a structural engineering strategy or a mathematical purity.

3.5.4. 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 participants, 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 have 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 solution 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 (Mark 2007). Our 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 bolster 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, fuse and/or minimize these multiple levels of abstraction when designing optimization software.

Acknowledgements

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

References

[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

Design problems are complex and are ill-defined in the early stages of projects. To gain an insight into these problems, designers envision a space of various alternative solutions and explore various performance trade-offs, often manually. To assist designers with rapidly generating and exploring a design space, researchers introduced the concept of design synthesis methods. These methods promote innovative thinking and provide solutions that can augment a designer's abilities to solve problems. Recent advances in technology push the boundaries of design synthesis methods in various ways: a vast number of novel solutions can be generated using high-performance computing, in a timely manner; complex geometries can be fabricated using additive manufacturing; and integrated sensors can provide feedback for the next design generation using the Internet of things (IoT). Therefore, next synthesis methods should be able to provide designs that improve over time based on the feedback they receive from the use of the products. To this end, the objective of this study is to demonstrate a design synthesis approach that iterates a flexible set of design requirements gathered from sensor data, generates innovative alternative solutions targeted for additive manufacturing. To demonstrate this method, we present a case study of design iteration on a car chassis. First, we installed various sensors on the chassis and measured forces applied during various maneuvers. Second, we used these data to derive a flexible set of engineering problems as a collection of design requirements and constraints. Third, using an ensemble of topology and beam-based optimization techniques, we created a number of novel solutions. Finally, we selected one of the design solutions and because of some manufacturability constraints we 3D-printed a prototype for the next generation of design on one third scale. The results show that designs generated from the proposed method were up to 18% lighter than the existing designs. This paper also presents various lessons learned to help engineers and designers with a better understanding of challenges applying new technologies in this research.

INTRODUCTION

In the current design of engineering products, designers, instead of exploring a space of various alternative solutions, often rely on previous designs to create the next generation of products. This process hinders designs creativity and discourages designers to find innovative solutions [1]. This lack of out-of-the-box thinking is one of the reasons that engineering products have a similar pattern in their design. For instance, a configuration study on twin-engine, propeller-driven, aircraft reported that the baseline design accounted for 50% of all configurations [2]. This limitation in designers' creative thought by adhering to a pre-established set of ideas in the design process is known as design fixation [3]. Fixation leads to duplication of efforts and makes a design process difficult to adapt to new innovations in the field [4].

To reduce fixation and explore innovative design, designers create a space of various alternative solutions and explore various performance trade-offs. To assist designers with rapidly generating a design space, researchers introduced the concept of design synthesis methods [5]. These methods promote innovative thinking and provide solutions that can augment a designer's abilities to solve complex problems. These techniques have been tested by generating a number of novel yet feasible designs such as aircraft configurations [6], wheel rims and cooling fins [6], satellites [7], power trains [8], and gear boxes [9, 10]. One of the limitations of the current synthesis methods

Figure 1: Proposed synthesis method

IMPLEMENTATION OF THE PROPOSED METHOD

We extended the [illegible] generative design [illegible] Autodesk Research [illegible] case study to implement, test and validate this methodology. We chose to design a small car chassis [illegible] due to the structural nature of the design problem [illegible] design synthesis prototype system, the ease of fabrication [illegible] methods which allows for fast experimentation, and the potential to draw comparisons with well-established design practices present in the automotive industry. [illegible] design synthesis methods previously developed by the authors and the long-range impact that this study could have on [illegible] of automobile design and manufacture.

In this paper, the term [illegible] refers to the interface of [illegible] with external components. The term [illegible] is used to represent a [illegible] design space that a synthesis algorithm [illegible]. The term 'initial chassis' refers to the first generation of a chassis designed and engineered for the case study. The following sections describe how each step of the method has been implemented and tested.

Controlled design trial

The initial design was produced using [illegible] chassis [illegible] and tubes constructed by domain professionals in the field of automotive fabrication. [illegible] the model defining the design problem for synthesis [illegible] high-fidelity 3D model of the [illegible] and all connected parts for the chassis, a high-resolution polygonal mesh of the initial space frame chassis design and its connected components was obtained by employing a laser scanning method (Figure 2). [illegible] sensor data capture was coordinated with the high-resolution mesh to install strain gauges at appropriate locations on the initial design. Preparation for the shape synthesis algorithm required a [illegible] mesh representation in order to produce valid results and the mesh was decimated and separated from the overall chassis to include only the essential geometry that interfaces directly with external components.

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

Sensor Data Acquisition

The sensors were placed in locations where boundary conditions [illegible] mechanical contact with other parts (e.g. suspension components). [illegible] of the strain gauges required the values reported to be normalized with respect to the orientation of the sensors. Figure 3 shows the typical installation for the strain gauge sensors as installed on the chassis before the road test. A total of [illegible] sensors were installed at the interfacing [illegible] of the initial chassis before data acquisition.

After being outfitted with the full sensor package, the chassis was driven in the California desert in extreme conditions by professional race drivers (Figure 5). A total of [illegible] were performed where sensor data were collected, where the driver operated the vehicle in scenarios including maximum acceleration and deceleration, [illegible] turn, impact from [illegible] and [illegible]. The data from the [illegible] were inspected and converted into a format suitable for entry as load cases within [illegible], the generative design tool.

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

Figure 5. Initial design fully equipped for testing with drivetrain and sensors installed

Data Analysis and Translation

We used an Excel plug-in from National Instrument known as TDM Excel for data analysis and translation. We reconciled all of the sensor data and computed forces applied to the chassis in tension, compression, and shear. This set of load profiles is an essential component for the problem definition step in our design synthesis system.

Problem definition

The known mounted geometry of the chassis linkage interfaces and the acquired sensor data were then used to build the problem definition, which involved establishing boundary conditions and obstacle regions. Boundary conditions were described by applying load conditions to all mounting points which the chassis featured, including mounts for seats, shocks, A-arms, engine, transmission, differentials, radiator, tank, instrument panels and steering wheel. Autodesk Memento was used to subdivide the 3D scan of the chassis to extract the mesh geometry of these mounting points, which can be seen in Figure 6 along with a scan of the engine in its intended position. The locations of the mounting points was then adjusted in order to account for an improved suspension geometry, steering and longer body, and a different engine. In order to accelerate the design synthesis, the geometry of parts was simplified and an enveloping geometry was modeled to encompass several interfaces simultaneously, as can be seen in the results. Load cases for each part were defined by indicating the direction and magnitude of the maximum force experienced by the chassis, calculated from the strain gauge data. Missing loading information was inferred from the weight of the part being mounted plus a safety factor. Lastly, obstacle regions were defined using the meshes of scanned components, such as the engine and the transmission, and by identifying keep-out zones necessary for the use, assembly, and maintenance of the car.

Figure 6. Top view of the location of the interface geometry with the scanned engine block in the rear of the chassis. (1) A-arms, (2) shock mounts, (3) steering rack mounts, (4) seat mounts, (5) steering column mount, (6) fan mount.

Design Synthesis

Given the above problem definition, our goal was to synthesize a solution space from which the user could select their preferred final design. We employed a novel synthesis method that combines a topology optimization initialization with a beam network optimization. The topology optimization initialization allows us to procedurally carve away volume that does not support the loads specified in the problem definition giving us a reduced initial space to fill with the beam network. We ultimately used the beam network since this best modeled the chromoly tube network that was the preferred manufacturing method for the car chassis based on our construction capabilities. The parameters of this method that defined our space were (g, n, r, k) where g is the topology optimization volume constant that determines the 'thickness' of the solution, n is the number of topology optimization iterations, r is the density of the beam nodes and k is the degree of connectivity of the beam nodes. The overall algorithm is outlined below and in Figure 7.

Synthesis Algorithm:

1. Initialize design space volume Ω with convex hull
2. Generate solution space
   a. Iteratively topology optimize
      i. Compute strain energy density using finite element analysis (FEA) solver
      ii. Advect volume according to shape derivative
   b. Initialize beam network
      i. Sample beam nodes within volume
      ii. Connect nodes to form beams according to distribution
   c. Optimize beam network thickness
3. User selects optimal design from solution space

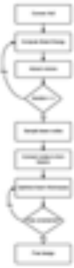

Table 1. Chassis loads in one of the load cases

	Front/Back of Car	Upper/Lower A-Frame	Front/Back of A-Frame	Max Load (kN)	Min Load (kN)
1	Front	Upper	Front	-95.2	-11.6
2	Front	Upper	Back	-22.9	-8.8
3	Front	Lower	Front	-76.8	-47.2
4	Back	Lower	Back	-102.5	-224.7
5	Back	Tie Rod	–	-15.4	5.6
6	Back	Shock Strut	–	47.2	-5.8
7	Back	Upper	Front	-27.3	10.7
8	Back	Upper	Back	-59.6	-27.9
9	Back	Lower	Front	40.1	6.9
10	Front	Lower	Back	24.3	48.0
11	Front	Steering Arm	–	-55.3	-47.3
12	Shock	Shock Strut	–	-21.4	-22.6

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.

Ensemble Synthesis Methods

Here we demonstrate the results of our synthesis method. The intermediate stages of synthesis and final selected design are shown in Figures 9 and 10. The final design reduced the weight of the initial design from 735 lbs down to 262 lbs.

Figure 9. Results of the topology optimization for n=92 and n=99

Figure 10. Beam optimization results for selected design. Top: Initial beam network (1998n). Bottom: Final optimized beam network (2828n).

Lessons Learned from Design Synthesis

Lessons learned from the Manufacture of Improved Design

CONCLUSION

REFERENCES

[1] [illegible], 2014, "Design [illegible]," Design Computing and Cognition'12, Springer, pp. [illegible].

[2] [illegible], 2003, "[illegible] configurations," [illegible] American Institute of Aeronautics and Astronautics, St. Louis Section.

[3] [illegible], 1991, "Design [illegible]," Design Studies, [illegible], pp. [illegible].

[4] [illegible], 2013, "Overcoming Organizational Fixation: Creating and Sustaining an Innovation Culture," The Journal of Creative Behavior, [illegible], pp. [illegible].

[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. [illegible].

[6] [illegible] and Shea, K., 2013, "An interactive visual approach to developing and applying parametric three-dimensional spatial grammars," AI EDAM, [illegible] (Special Issue 04), pp. [illegible].

[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 03), 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. [illegible].

[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 grammatical discrete structures," AI EDAM, 13(04), pp. [illegible].

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

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

[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, 2015, "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] Allaire, 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], Shea, K., [illegible], and [illegible], 2014, "Fully Constrained Design: A general and scalable method for discrete member sizing optimization of steel truss structures," Computers & Structures, [illegible], pp. [illegible].

[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

[illegible]

ABSTRACT

[illegible]

KEYWORDS

[illegible]

1. INTRODUCTION

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

3.3.1. Semantic Categories

For semantic categories, we assume the kinds 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 3). [illegible]

Table 3 – [illegible]

Functions	**Definition:** [illegible] **Example:** [illegible]
Objectives	**Definition:** [illegible] **Example:** "The weight of the design must be minimized."
Constraints	**Definition:** [illegible] **Example:** "The width of the design cannot be greater than 5cm."
Properties of environ. objects	**Definition:** [illegible] **Example:** "The weight of the shelf is ... kg."

3.3.2. Syntax

[illegible]

[illegible]

Table [illegible]

Functions	[illegible]
Objectives	[illegible]
Constraints	[illegible]
Environ. objects	[illegible]

Table 4 – [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-subject design was used (Table 5).

Table 5 – [illegible]

[illegible] natural language [illegible] this approach can provide [illegible] [illegible] the different [illegible] and [illegible] a significant challenge.

We emphasize that the [illegible] CNL is not to capture the entire knowledge of designers during the design process. [illegible] formal data that can leverage the [illegible] potential benefits of computational design. [illegible] explore multiple design spaces [illegible] natural language [illegible] design [illegible] functions [illegible] information [illegible] The [illegible] study demonstrated the feasibility of a CNL in exploring [illegible] function definitions.

6. CONCLUSIONS AND FUTURE WORK

We demonstrated that using a CNL, compared to using natural language [illegible] problem definition [illegible] useful information for formulating [illegible] optimization problems. [illegible] using a CNL [illegible] problem definitions. [illegible] a CNL should be [illegible] the [illegible] of the designers' [illegible] and solutions.

Our future work will focus on understanding the natural language input and translating the input into formal data. We will create a [illegible] knowledge base of [illegible] such as [illegible] shapes, materials, etc. We will also explore application of natural language understanding techniques, such as syntactic analysis [illegible] and manual expression of problem [illegible]

REFERENCES

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

[illegible] "Towards a Model of Exploration in Computer Aided Design", In: [illegible]

[illegible] "A Survey of [illegible]" [illegible]

[illegible]

[illegible] "[illegible] Computer-Aided Design", [illegible]

[illegible]

[illegible] "[illegible] Computer Aided [illegible]", [illegible]

Pahl G., Beitz W. [illegible] "Engineering Design: A Systematic Approach, 3rd Edition", Springer-Verlag, London, UK.

[illegible] "Towards Natural Language Understanding in Organization-Based Design", Human-Computer Studies, [illegible]

[illegible]

[illegible]

[illegible] "Controlled Natural Language for Knowledge Representation", [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible] and [illegible] 1994. In addition, [illegible] and [illegible] [illegible] [illegible]

[illegible] of premium [illegible] information.

[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. [illegible] would you rate the usefulness of using the CM, [illegible]?

[illegible] translated to ESL: See Table 2 of the SC-CM, [illegible], participants

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

natural language could not be translated into the CSL. Nine of these described the non-functional design features in [illegible]. 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]

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

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Figure 3: A Dynamo node that creates a SpaceLattice object from an input bounding box, a list of line segments representing barriers, and the spacing between grid points.

3.2 Discretization

The line segments that represent barriers are primitive Dynamo objects and are not constrained to points on the space lattice grid. When these line segments are used to block connectivity between grid points, they are effectively being converted from a continuous vector-based representation to a discrete grid-based representation. There are multiple ways to perform this discretization, and the method chosen may affect both the results of the various analyses and the appropriate choice of grid resolution. We implemented two discretization methods: a basic method and an enhanced method.

The **basic discretization method** in Figure 4 is the most obvious approach. The initial layout shows a 4x5 grid of points that may or may not be accessible depending on which connections get severed by six barriers (represented by line segments) (Figure 4a). The first step is to establish a coarse grid for processing the barriers (Figure 4b). In this case it is essentially the same as the original grid. The next step is to use Bresenham's standard line drawing algorithm to process the barriers by filling in grid cells as if they were pixels in an image (Figure 4c). The final step is to ensure the original points are connected to their neighbors wherever the path is not cut off by blocked cells (Figure 4d). Observe that the path from A3 to B2 is cut off by a pair of blocked cells that meet at a corner. The path from C3 to D2 touches the corner of a blocked cell, but is not cut off since one could walk around.

The **enhanced discretization method** shown in Figure 5 differs from the basic method in two ways. First, the grid used to process the barriers is twice the resolution of the initial grid of points (Figure 5b). Second, after the barriers are processed using Bresenham's algorithm, blocked cells are expanded to fill their 4 nearest neighbors (Figure 5d), and then contracted by expanding the unblocked cells in the same manner (Figure 5e). These expansion and contraction steps fill in small gaps. The original points are then connected where possible. Notice that the path from B1 to C2 is cut off by a pair of blocked cells, but the path from C1 to D2 is accessible.

(a) Initial layout (b) Step 1: Add coarse grid

(c) Step 2: Draw barriers (d) Step 3: Connect paths

Figure 4: Basic discretization method with one resolution.

(a) Initial layout (b) Step 1: Add fine grid

(c) Step 2: Draw barriers (d) Step 3: Expand barriers

(e) Step 4: Contract barriers (f) Step 5: Connect paths

Figure 5: Enhanced discretization method where barriers are drawn at a finer resolution, then expanded, then contracted.

If we compare the outcome of the basic method in Figure 6b with that of the enhanced method in Figure 7b, we find that the same initial set of barriers can generate very different space lattices. For example, point (b4) is accessible only when using the basic method, and point (r) is only accessible using the enhanced method. The enhanced method produces a route that encircles the barriers on the right-hand side of the environment, whereas the basic method eliminates this route.

Our SpaceSeeker tool uses only 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 6 and 7 are somewhat contrived so that the gaps between the barriers are unrealistically 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 8. 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.

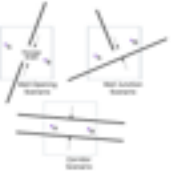

Figure 8: Scenarios used to compare discretization methods.

For each scenario, we varied the passage width from 0 to 4 in increments of 0.1, where the unit of measure is the resolution of the space lattice grid (i.e. the grids shown in Figures 6a and 7a). To clarify, suppose the grid resolution is 25 cm. In that case, a passage width of 1 means the passage is 25 cm wide, and a passage width of 2 means the passage is 50 cm wide.

For each scenario and passage width, we ran 10000 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 10000 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 (3 scenarios × 41 passage widths × 10000 tests) was repeated for both the basic and enhanced methods. The results are plotted in Figure 9. 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 9 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 close 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 presumably 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 25 cm essentially guarantees that travel paths will connect through passages of at least 60 cm. The basic method would require a finer resolution of 15 cm to ensure all 60 cm gaps are traversable.

Computational Science Research

Industry Research

Life Sciences

Design & Social Impact

Complex System Research

Robotics Lab

Machine Learning

The Living

Simulation & Graphics Research

EXISTING