The Dissertation Committee for Xingang Li
certifies that this is the approved version of the following dissertation:

Towards Human-Centered Generative Design: Cross-Modal Synthesis for Three-Dimensional Design Concept Generation

Committee:

Zhenghui Sha, Supervisor

Carolyn Seepersad

Richard Crawford

Etienne Vouga

Towards Human-Centered Generative Design: Cross-Modal Synthesis for Three-Dimensional Design Concept Generation

by

Xingang Li

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2024

Dedication

This dissertation is lovingly dedicated to my wife, Yajie Li, my son, Tianxin Li, my father, Shudang Li, my mother, Qingrong Nie, my sister, Yun Li, my father-in-law, Peiyan Li, and my mother-in-law, Xiaomei Liu. Their support, encouragement, and love have sustained me throughout my life.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Zhenghui Sha, whose guidance, patience, and support have been the cornerstone of my journey toward this Ph.D. degree. Dr. Sha's relentless dedication to and enthusiasm for high-quality research, inspiring work ethic, hardworking attitude, and profound wisdom have not only molded me into a better researcher and scientific writer but also a person of strong character and resilience. Working with Dr. Sha has been profoundly influential, extending beyond what can be adequately expressed in this brief acknowledgment. His unwavering support has been pivotal not only to the completion of this dissertation but also to my academic achievements as a whole. Without his guidance, the trajectory of my academic journey would undoubtedly have been different.

My deepest affection and gratitude go to my wife, Yajie Li, and my son, Tianxin Li. Yajie, your support, love, and belief in me have been my strength in the most challenging times. Tianxin, your joy and innocence have been the light and happiness of my life. To my dear family, my father, mother, big sister, father-in-law, and mother-in-law, your endless encouragement and support have been the bedrock of my perseverance and success. Your sacrifices and belief in me have shaped who I am today, and I am forever grateful.

My committee members, Dr. Richard Crawford, Dr. Carolyn Seepersad, and Dr. Etienne Vouga, deserve special mention for their meticulous reviews and constructive feedback, which have significantly enhanced the quality of my dissertation research.

I am immensely thankful to my research collaborators, Dr. Charles Xie from the Institute for Future Intelligence, Dr. Onan Demirel from Oregon State University, Dr. Molly Goldstein from the University of Illinois Urbana-Champaign, and Ye Wang from Autodesk, whose invaluable insights have enriched my research experience and helped me grow in academia. Their collaboration has been instrumental in pushing the boundaries of the projects I have participated in during my Ph.D. career.

I greatly appreciate all members of the SiDi Lab (Yinshuang Xiao, John Clay, Laxmi Poudel, Molla Hafizur Rahman, Siyu Chen, Yuewan Sun, Elisa Koolman, Cole Mensch, Ronnie Frank Pires Stone, Pawornwan (Bam) Thongmak, Phillip Gavino, Anuj Swaminathan, Daniel Weber, Jared Poe, Sumaiya Sultana Tanu, Saivipulteja Elagandula) for their stimulating discussions, suggestions, help, and collaborative environment that has been pivotal in my research journey. They have been not just lab mates but also close friends. Furthermore, I wish to express my sincere appreciation to all my friends and peers across China, the U.S., and Japan, for their invaluable support and assistance over the past five years.

A heartfelt thank you to the University of Arkansas and the University of Texas at Austin. My initial years at Arkansas laid the foundation of my research, and the subsequent years at Texas provided me with unparalleled opportunities and resources to excel. I would like to express my appreciation for the financial, academic, and logistic support provided by both universities, which have been instrumental in my research, academic studies, and life in the U.S. I would also like to acknowledge the financial support from the National Science Foundation through Grant 2207408.

This journey would not have been possible without the collective support and faith of everyone mentioned, and to them, I offer my sincerest gratitude.

Abstract

Towards Human-Centered Generative Design: Cross-Modal Synthesis for Three-Dimensional Design Concept Generation

Xingang Li, Ph.D.
The University of Texas at Austin, 2024

SUPERVISOR: Zhenghui Sha

There has been a fast-growing interest in generative design (GD) – an artificial intelligence (AI)-based approach for early-stage design that automates the creation and optimization of various design concepts. GD can quickly generate numerous design variants, allowing for efficient exploration of design space, and thus can shorten the product design cycle and reduce development costs. Despite advances in GD technologies, current GD approaches place AI at the center of the design process and lack direct involvement of human preference and judgment in the concept generation process. This raises significant concerns, as current GD methods lack mechanisms that incorporate human factors, such as aesthetic preferences, and the safety and comfort of designs. This omission could lead to the creation of design concepts that fail to meet human needs effectively. In contrast, human designers bring their expertise and knowledge to generate human-centered designs, playing an irreplaceable role in the design process. This is especially true in the early stages, where human domain knowledge and preferences are crucial in determining the potential to produce user-centered and user-friendly products. These elements critically influence the success of product design and development.

To fill this research gap, the overarching goal is to realize human-centered

generational design. Towards that goal, the **research objective** of my dissertation is to develop a human-supervised data-driven generative design framework that can actively keep humans in the loop, as well as in charge of the generation and evaluation of AI-assisted design concepts in early-stage design. In particular, my research is motivated to answer the following **central research question**: In what ways and to what extent can human designers' intent and preferences be incorporated as input to actively interact and guide the GD process to improve the quality and relevance of the design outcomes? The **central research hypothesis** is that human designers' intent and preferences can be incorporated as input to guide the data-driven design generation using cross-modal synthesis. Cross-modal synthesis is a machine learning technique that can generate data in one modality (such as images) based on another input modality. Our rationale is that human designers can use different design modalities (such as sketches of car models) or a combination of them to represent their intent and preferences (e.g., certain curvatures in a car's exterior design), and cross-modal synthesis methods can automatically transform them to desired design modalities, such as 3D CAD models. However, the development of cross-modal synthesis methods for engineering design needs to address several fundamental challenges. These challenges include the scarcity of design data, complexities in 3D design representations, large semantic gaps between different modalities (challenging to maintain design integrity and intent embedded in the input design modality), and vectorized design representations for AI training.

To test the central hypothesis, we aim to answer three Research Questions (RQs). (1) How feasible and to what extent can cross-modal synthesis methods with unimodal input incorporate human designers' intent and preferences as input to guide the data-driven design generation? (2) How feasible and to what extent can cross-modal synthesis methods with multimodal inputs incorporate human designers' intent and preferences as input to guide the data-driven design generation? (3) What are the effects of different representations of the generated designs on the data-driven design evaluation?

To address these RQs, we explore various methodologies. For RQ 1, we conduct a systematic review of deep learning methods for cross-modal tasks, from which we identify technology on how to develop cross-modal synthesis methods for engineering design. We then develop a novel neural network architecture, target embedding variational autoencoders, based on which we create two cross-modal synthesis methods with unimodal input for the task of (1) silhouette contour sketch to 3D mesh and (2) image to CAD sequence. In response to RQ 2, we propose a multimodal CAD dataset to enable and evaluate large language models' ability to generate CAD models from multimodal inputs (i.e., textual descriptions, sketches, and images). Lastly, RQ 3 is answered by developing a data-driven structure-aware generative design and evaluation approach and examining vectorized design representations to improve the assessment of generated designs.

From the results, we conclude that: (1) Cross-modal synthesis methods, capable of processing either unimodal or multimodal inputs, exhibit significant potential in capturing and integrating human designers' intent and preferences to guide the generation of early-stage design concepts in 3D representations. Although textual descriptions, sketches, and images may not fully encapsulate the designers' envisioned ideas—a challenge also prevalent in traditional design practices—our cross-modal synthesis approaches can still discern and interpret the underlying design preferences from these varied input modalities. Consequently, these methods can generate 3D designs that are closely aligned with the specified design requirements embedded in the input design modalities, thus bridging the gap between conceptual intent and tangible design artifacts. (2) The choice of mathematical design representations (e.g., vectors of design features) significantly influences the evaluation of generated designs in design performance prediction. Such influences are particularly significant when product geometries become more intricate and when dealing with systems design generation where complex interdependent relations between components exist. Although latent spaces are commonly utilized for these vectorized representations to accelerate AI-assisted design evaluation and optimization, such latent vectors may prove unsuitable if any information irrelevant to the engineering performance of interest is encapsulated during the formulation of design representations, consciously or unconsciously. This observation underscores the imperative for designers to consider the suitability of vectorized design representations for evaluation purposes from the beginning of developing DGD methodologies.

In summary, this dissertation represents a critical step forward in human-centered generative design. It advances the design field by addressing a crucial gap in existing GD methodologies, facilitating a human-centered GD approach. Specifically, we introduce a novel *Human-Supervised Data-Driven Generative Design Framework* with cross-modal synthesis methods for design generation and AI-assisted design evaluation methods, which enhance human control and interaction within the GD process. Notably, this work develops an innovative neural network architecture tailored for cross-modal synthesis in engineering design. This architecture is particularly adept at incorporating human intent and preferences into the GD of 3D design concepts. The proposed methodologies can significantly accelerate design ideation, enhance the exploration of design spaces, and incorporate downstream considerations into early-stage decision-making. The methodologies are domain-independent and can be employed across different products in industries to expedite the product development cycle and decrease associated costs. Furthermore, the methods have the potential to be translated into pedagogical tools in design education, preparing next-generation engineers for future careers in an evolving landscape that increasingly values human-AI collaboration in engineering design.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

The **objective of my dissertation research** is to develop a human-supervised data-driven generative design framework that can actively keep humans in charge of artificial intelligence (AI)-assisted 3D design concept generation and evaluation in early design.

AI is a rapidly growing technology that has played an essential role in supporting and even replacing humans in many situations of problem-solving and decision-making. The same trend is observed in engineering design. As shown in Figure 1.1, a traditional design process consists of four major stages (Won, 2001): *Analysis*: identify the design opportunities from user feedback and needs; *Conceptual Design*: generate ideas and concepts; *Preliminary Design*: evaluate and refine the generated concepts; and *Detailed Design*: develop a final solution. Such a design process is often iterative within each stage, as well as between stages. So far, AI has been applied to different design stages to automate repetitive and time-consuming tasks, such as customer needs analysis, generating design concepts, engineering analysis, and optimization at various design stages (Han et al., 2023; Chen and Fuge, 2019; Li et al., 2021c; Regenwetter et al., 2022).

In particular, in the early stage of design, such as the *Conceptual Design* stage in Figure 1.1, generative design (GD) methodology is drawing more and more attention and has been widely adopted in many engineering applications, such as wheel rim design (Oh et al., 2019; Yoo et al., 2020) and integrated into many commercial computer-aided design (CAD) software, such as Autodesk Fusion 360 (Keane, 2018). GD is an AI-driven design method that utilizes computer algorithms to automatically generate design concepts by considering human-defined objectives, parameter ranges, and constraints (McKnight, 2017; Krish, 2011; Matejka et al., 2018; Singh and Gu,

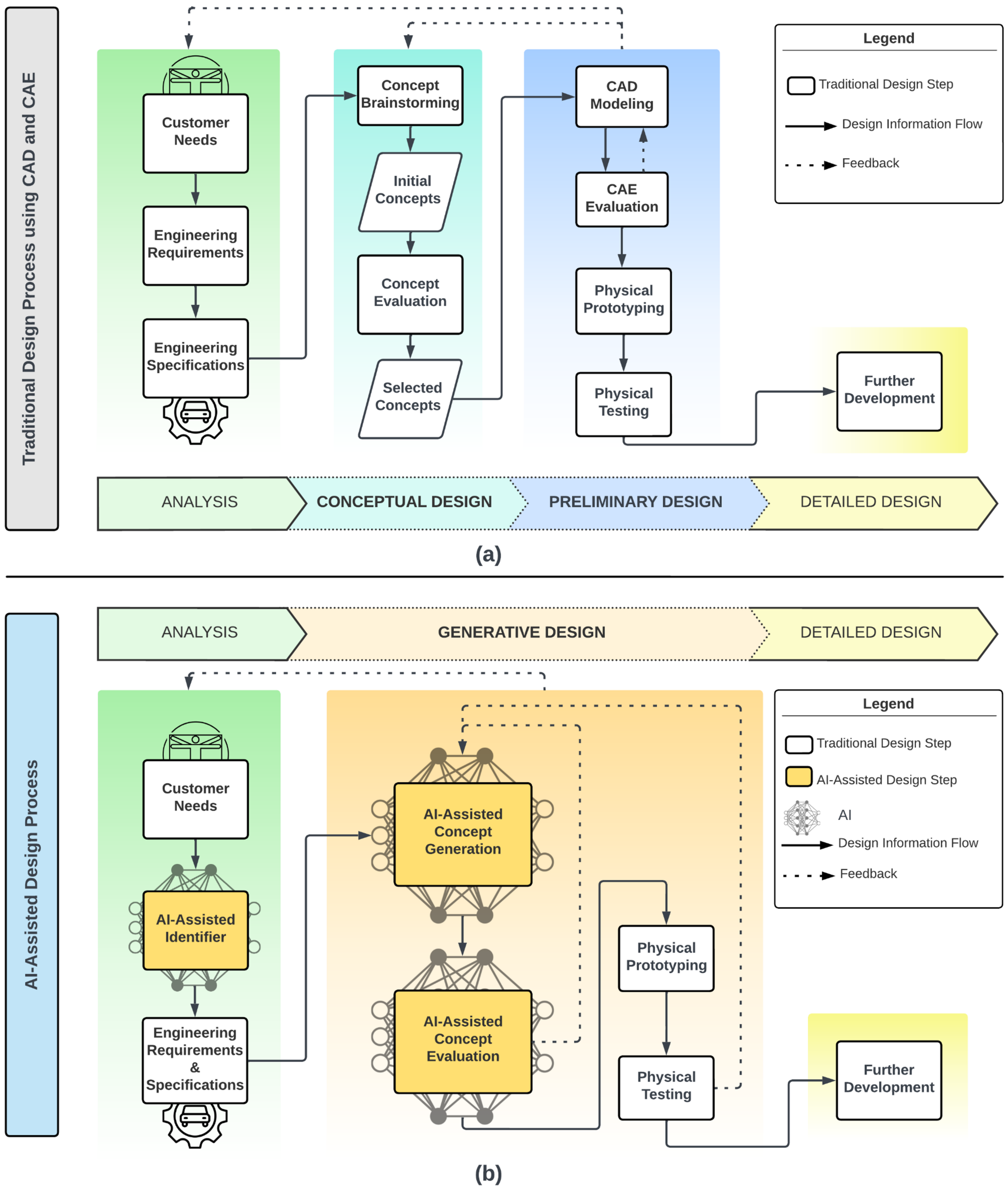


Figure 1.1: The comparison of (a) traditional engineering design process using CAD and CAE and (b) AI-assisted engineering design process

2012). GD automates the process of concept creation and design optimization, so it can quickly generate a large number of design variants, many of which could even be beyond human imagination. By repeating these computational routines many times, a variety of designs that meet the goal eventually emerge. Designers then review these outputs, often with the aid of interactive visual analytics for intuitive assessment and comparison across the board, and choose one or more designs for prototyping (van Kastel, 2018). As a result, it facilitates the exploration of the design space and the evaluation of various designs which can shorten the product design cycle and reduce the development cost (Mountstephens and Teo, 2020; Li and Lachmayer, 2019).

However, there is a critical gap in current GD technologies: they do not directly involve human preference and judgment in the concept generation process (Urquhart et al., 2022; Demirel et al., 2023a), yet the human role in early design is irreplaceable. For example, human designers' domain knowledge in judging functional performance and users' preferences for aesthetics are both important factors that can influence the final design embodiment. Rule-driven GD (RGD) methods, such as shape grammars, genetic algorithms, and topology optimization, prioritize engineering performance and often generate designs that are too complex to produce without additive manufacturing (McKnight, 2017); and, meanwhile, could ignore human preferences for aesthetics (Oh et al., 2019). Furthermore, discovering design rules for a target task could be time-consuming (Mountstephens and Teo, 2020), affecting RGD's development and thriving. On the contrary, data-driven GD (DGD) methods (see Figure 1.2 (a)), learn latent design representations from data without explicit design rules, using techniques such as variational autoencoders (VAEs) (Kingma and Welling, 2013) and generative adversarial networks (GANs) (Goodfellow et al., 2014). They learn a probability distribution of the training data and generate new designs from the learned distribution through random generation and interpolation (Li et al., 2023b). Although training data could implicitly capture human preferences, DGD methods still lack explicit human interaction and supervision over concept generation. As shown in Figure 1.2 (a), human preferences are aggregated and embedded in the training data of the final

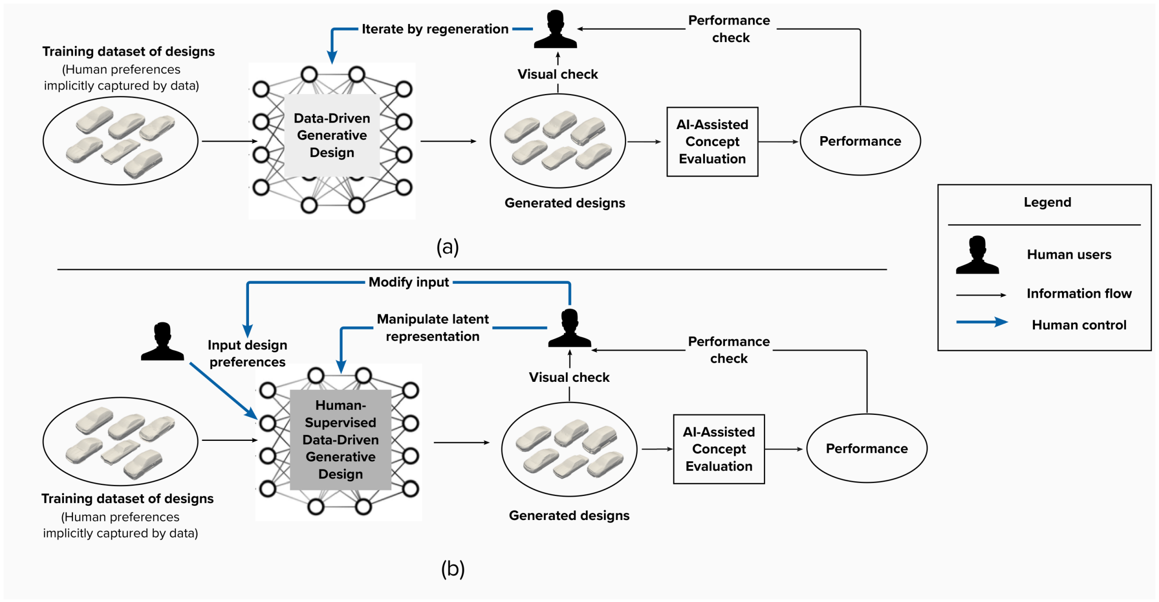

Training dataset of designs
(Human preferences implicitly captured by data)
Iterate by regeneration
Data-Driven Generative Design
Performance check
Visual check
Generated designs
AI-Assisted Concept Evaluation
Performance
(a)
Legend
Human users
Information flow
Human control
Modify input
Manipulate latent representation
Input design preferences
Human-Supervised Data-Driven Generative Design
Performance check
Visual check
AI-Assisted Concept Evaluation
Performance
Generated designs
Training dataset of designs
(Human preferences implicitly captured by data)
(b)

Figure 1.2: The comparison of (a) data-driven generative design process and (b) human-supervised data-driven generative design process

design artifacts in traditional DGD methods. Consequently, in a typical GD-based design process, designers specify objectives and constraints, delegate design generation to a GD engine, and select preferred designs from all generated designs. This process is open-loop and AI-in-the-charge, thus not placing humans at the center of the design generation. This raises significant concerns, as current generative design methods lack mechanisms that incorporate human factors, such as aesthetic preferences, and the safety and comfort of designs. This omission could lead to the creation of design concepts that fail to meet human needs effectively. In contrast, human designers bring their expertise and knowledge to generate human-centered designs, playing an irreplaceable role in the design process.

Due to the unique nature of early-stage design, humans shall remain at the center of the concept generation process, leveraging their knowledge, expertise, creativity, intuition, and judgment to support the creation of user-friendly and user-centered design solutions. Therefore, our ultimate goal is to address the research gap by developing a human-centered generative design and optimization framework. To achieve this goal, we have been working on investigating how humans can directly interact during the data-driven GD process and what the effects would be if designers or users were provided with a human-supervised data-driven GD tool during their idea generation process. We proposed a human-supervised data-driven GD framework as shown in Figure 1.2 (b). Compared to the traditional data-driven GD process shown in Figure 1.2 (a), the new framework provides interfaces to bring in human interactions in three different modes: 1) to allow human preferences to be added before the generative design process; 2) to enable humans to manipulate the latent design representation from the trained AI design agent; 3) to allow human feedback to modify the original human input to guide the re-generation of new designs or to modify the originally generated designs. Based on this new human-supervised DGD framework, the scope of my dissertation research is focused on the first mode and its associated research questions, as discussed in the section below.

1.2 Central Research Question, Research Hypothesis, Research Questions, and Research Tasks

The **Central Research Question** of my research is: *In what ways and to what extent can human designers' intent and preferences be incorporated as input to actively interact and guide the GD process to improve the quality and relevance of the design outcomes?* Our **Central Research Hypothesis** is that human designers' intent and preferences can be incorporated as input to guide the data-driven design generation using cross-modal synthesis.

Specifically, I am motivated to answer the following three **Research Questions (RQs)** to test this hypothesis.

- ***RQ 1:*** *How feasible and to what extent can cross-modal synthesis methods with unimodal input incorporate human designers' intent and preferences as input to guide the data-driven design generation?*

- ***RQ 2:*** *How feasible and to what extent can cross-modal synthesis methods with multimodal inputs incorporate human designers' intent and preferences as input to guide the data-driven design generation?*

- ***RQ 3:*** *What are the effects of different representations of the generated designs on the data-driven design evaluation?*

In Table 1.1, the research tasks associated with each RQ are presented. To answer RQ 1, I (1) conduct a literature review on deep learning methods for cross-modal tasks (or cross-modal synthesis methods) that have the potential to translate human preferences in one design modality to 3D designs, (2) build a novel neural network architecture for the cross-modal synthesis, and (3) apply the proposed neural network architecture to build a sketch-to-3D model and (4) an image-to-3D model.

For RQ 2, I (1) synthesize a multimodal CAD dataset that has dimensional information of the ground truth 3D designs, (2) use the images, sketches, and text with dimensional information from the dataset as input to evaluate multimodal large language models' (LLMs) inherent capability to generate 3D CAD concepts, and (3) propose ways to enhance the capability.

In DGD methods, such as many neural network-based methods, an important link is the creation of vectorized design representations of the target design artifact to be generated. To quickly evaluate the engineering performance of the generated designs to support human selection, **RQ 3** aims to investigate the effects of different representations of the generated designs on the prediction accuracy of the engineering performance. Therefore, the following four tasks must be accomplished: (1) obtaining different vectorized design representations of the generated designs; (2) collecting the engineering performance data of the generated designs using high-fidelity simulations; (3) developing data-driven design evaluation methods; and (4) comparing the prediction accuracy of the engineering performance using different combinations of the vectorized design representations and data-driven design evaluation methods.

Table 1.1: Research questions, tasks, and approaches

Research Questions	**Tasks**	**Approaches**
RQ 1 : How feasible and to what extent can cross-modal synthesis methods with unimodal input incorporate human designers' intent and preferences as input to guide the data-driven design generation?	• ***Task 1.1:*** Conducting a literature review on deep learning methods for cross-modal tasks (or cross-modal synthesis methods) that have the potential to translate human preferences in one design modality to 3D designs • ***Task 1.2:*** Building a novel neural network architecture for cross-modal synthesis of 3D designs • ***Task 1.3:*** Applying the proposed neural network architecture to build a sketch-to-3D model • ***Task 1.4:*** Applying the proposed neural network architecture to build an image-to-3D model	• A systematic literature review on deep learning methods for cross-modal tasks in engineering design by using design modalities (e.g., sketch, text) as input to guide 3D shape generation • Developing a novel Target-Embedding Variational Autoencoder (TEVAE) architecture for cross-modal synthesis by integrating target-embedding representation learning with variational autoencoders (VAEs) • Using the silhouette contour sketch as the input modality and mesh for the 3D output modality • Using images as the input modality and CAD sequences as the output modality; Using CAD program code to model the CAD sequence and proposing a vectorized design representation for the CAD program
RQ 2 : How feasible and to what extent can cross-modal synthesis methods with multimodal inputs incorporate human designers' intent and preferences as input to guide the data-driven design generation?	• ***Task 2.1:*** Synthesizing a multimodal CAD dataset that has dimensional information of the ground truth 3D designs • ***Task 2.2:*** Using the images, sketches, and text with dimensional information from the dataset as input to evaluate multimodal large language models' (LLMs) inherent capability to generate 3D CAD concepts • ***Task 2.3:*** Proposing ways to enhance the capability	• Automatic data synthesis pipeline to synthesize 3D designs and render them to images and sketches • Uisng Amazon Mechanical Turk to collect text data with dimensional information • Leveraging the CAD program code as a bridge to 3D CAD concepts. • Using GPT-4V(ision) as the multimodal LLM • Proposing a debugger using GPT-4V to iteratively correct the syntax errors of the CAD programming code to enhance the quality of the 3D CAD concepts
RQ 3: What are the effects of different representations of the generated designs on the data-driven design evaluation?	• ***Task 3.1:*** Obtaining different vectorized design representations of the generated designs • ***Task 3.2:*** Collecting the engineering performance data of the generated designs using high-fidelity simulations • ***Task 3.3:*** Developing data-driven design evaluation methods • ***Task 3.4:*** Comparing the prediction accuracy of the engineering performance using different combinations of the vectorized design representations and data-driven design evaluation methods	• Obtaining design representations of 3D car models using two methods: (1) latent vector from the latent space of trained VAEs; (2) vectorized design representation by embedding the generated 3D shapes using implicit field method and a 3D point matrix • Obtaining the drag coefficients of 3D car models in a computational manner using high-fidelity computational fluid dynamics software, Ansys Fluent • Building data-driven design evaluation methods using surrogate modeling techniques (e.g., gaussian process regression, locally linear embedding) • Using k-fold cross-validation to compare the prediction accuracy of different combinations

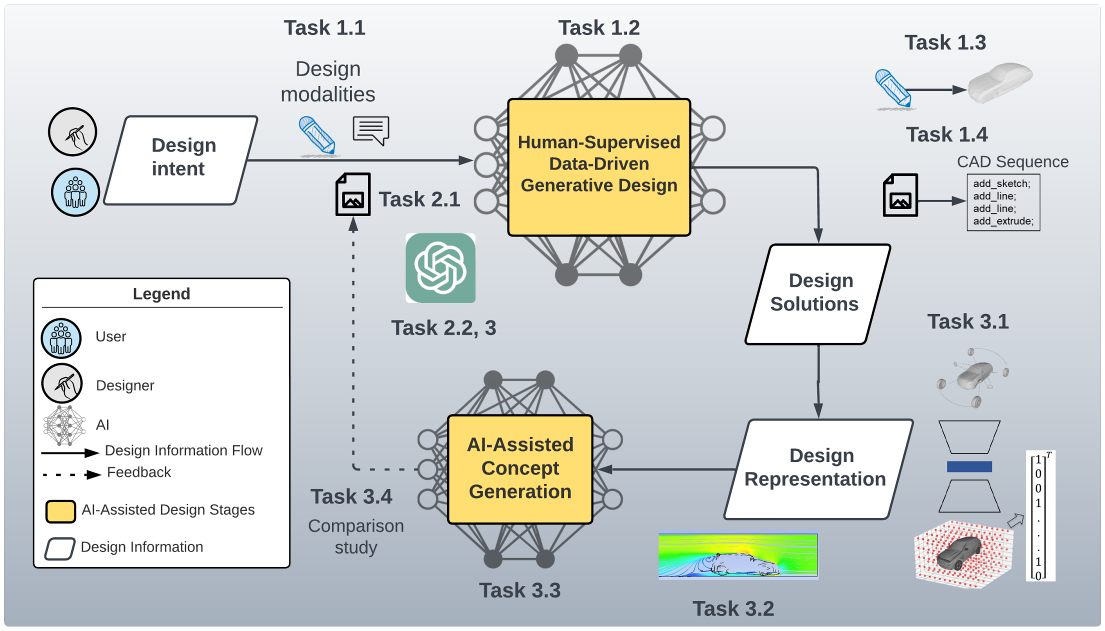


Figure 1.3: Overview of the tasks and corresponding approaches in the proposed human-supervised data-driven generative design framework

Exploring and resolving **RQ 1** and **RQ 2** offer valuable insights and methodologies for integrating human preferences and design intent into the data-driven design generation process via cross-modal synthesis methods with unimodal or multimodal input. This integration not only allows for an active interaction but also ensures that the outcomes of the generated designs align with the intentions and preferences of human designers. Furthermore, addressing **RQ 3** introduces effective and efficient methods for evaluating user-centered design concepts. Collectively, answering these three RQs establishes a framework for human-supervised DGD that facilitates the rapid generation and assessment of user-centered design concepts.

Figure 1.3 presents the research tasks, showing their connections in the proposed human-supervised DGD framework. The proposed research approaches to addressing the technical issues associated with each research task, as shown in Table 1.1, are explained. Regarding **Task 1.1**, the proposed approach involves a systematic literature review on deep learning of cross-modal tasks (DLCMT) in engineering design using design modalities (e.g., sketch, text) as input to guide design generation. In **Task 1.2**, I developed an approach that integrates target-embedding representation learning with variational autoencoders (VAEs) and built a novel target-embedding variational autoencoder (TEVAE) architecture for DGD methods to handle cross-modal design tasks. For **Task 1.3** and **4**, I am working to apply the proposed TEVAE to create different cross-modal GD methods to support human-supervised DGD, including a sketch-to-3D mesh method and an image-to-CAD sequence method, respectively.

In **Task 2.1**, I build an automatic data synthesis pipeline to synthesize 3D designs and render them into images and sketches and I use Amazon Mechanical Turk to collect text data with dimensional information. In **Task 2.2**, I propose to use the CAD programming code as an innovative bridge, facilitating the translation of human designers' multimodal inputs—including sketches, images, and textual descriptions—to the creation of 3D CAD objects. This approach involves exploring the effectiveness of multimodal LLMs, such as GPT-4Vision, in the generation of 3D CAD designs. For **Task 2.3**, I propose a debugger using GPT-4V to iteratively correct the syntax errors of the CAD programming code to enhance the quality of the 3D CAD concepts.

Task 3.1 involves obtaining vectorized design representations of 3D car models using two methods: 1) latent vector from the latent space of trained VAEs; and 2) vectorized design representation by embedding the generated 3D shapes using the implicit field method and a 3D point matrix. In **Task 3.2**, I obtained the drag coefficient data of the 3D car models from ShapeNet (Chang et al., 2015), a large-scale online dataset of 3D shapes, using high-fidelity computational fluid dynamics software, Ansys Fluent. Together with the vectorized design representations of car models, this dataset formed my training dataset for the construction of data-driven design evaluation methods. For **Task 3.3**, I have built data-driven design evaluation methods using surrogate modeling techniques and, particularly, two techniques:

the Gaussian process regression and locally linear embedding. I used k-fold cross-validation for **Task 3.4** to compare the prediction accuracy of different combinations of vectorized design representations and surrogate models and found the appropriate vectorized design representations in DGD for fast design evaluation.

1.3 Contributions

1.3.1 Overall Contributions

This dissertation advances human-centered generative design by addressing a crucial gap in existing methodologies, facilitating a human-centered approach. We introduce a novel *Human-Supervised Data-Driven Generative Design Framework* incorporating cross-modal synthesis methods and AI-assisted design evaluation techniques. Our approach enhances human control and interaction within the GD process. We have developed an innovative neural network architecture tailored for cross-modal synthesis in engineering design. This architecture effectively incorporates human intent and preferences into the generation of 3D design concepts. Our methodologies significantly accelerate design ideation, improve exploration of design spaces, and integrate downstream considerations into early-stage decision-making. They might have broad applicability across industries, speeding up product development cycles and reducing associated costs. Moreover, these methods can be adapted into educational tools for design students, preparing them for careers in an evolving landscape that increasingly values human-AI collaboration in engineering design.

1.3.2 Research Efforts and Technical Contributions

The development of cross-modal synthesis methods and AI-assisted evaluation methods for engineering design needs to tackle several fundamental challenges (Li et al., 2023b). These challenges include:

- Scarcity of design data: Unlike the abundant datasets of images or text, high-quality engineering data (such as 3D design data with corresponding engineering performance) are often confidential and challenging to obtain. However, acquiring such data is crucial for training reliable and effective AI models in engineering design.

- Complexities in 3D design representations: Engineering designs are often in 3D which can be represented in various forms, such as meshes, voxels, point clouds, implicit fields, and boundary representations. Each has its own merits and limitations in the context of AI-driven design.

- Bridging semantic gaps: There exists a notable semantic gap or information difference between different modalities (e.g., textual descriptions, sketches) and 3D CAD models, which complicates the synthesis process. Therefore, it is also challenging to maintain design integrity and intent embedded in the input design modality when transitioning between modalities.

- Vectorized representations for AI training: Identifying and utilizing appropriate vectorized design representations for neural network training is crucial for effective learning, generation, evaluation, and optimization processes. However, adapting these representations to various types of design data often requires innovative approaches and thus presents significant challenges.

To address the challenges, I have devised strategies to collect and synthesize datasets, build novel neural network architectures, and devise innovative vectorized design representations for 3D design data for both design generation and evaluation. The proposed methods can (a) enable active human interaction and guidance in the GD process through cross-modal synthesis, whether with unimodal or multimodal inputs, and (b) efficiently and effectively evaluate numerous generated design concepts by leveraging suitable vectorized design representations and surrogate models. These approaches lay the groundwork for the Human-Supervised Generative Design (HSGD) Framework as shown in Figure 1.3. This innovative framework integrates human preferences, expertise, and feedback into GD methodologies, marking a significant advancement in the field. It accelerates the design iteration process during the conceptual design phase, introduces downstream design considerations into early-stage decision-making, and addresses the fact that decisions made during the early stages can influence 70-80% of a product's lifecycle costs (Pahl et al., 1996).

The primary research efforts are outlined below.

- **Systematic review of methods for deep-learning of cross-modal tasks (DLCMT) for conceptual design of product shapes**

  We made the first effort to conduct a systematic literature review to identify DLCMT methods such as cross-modal synthesis methods, for the conceptual design to facilitate human-supervised generative design. DLCMT methods are likely to introduce new opportunities to support and enhance activities in the conceptual design stage for product shape design and beyond. We conduct a close examination of the existing literature aiming to identify the existing DLCMT methods and technologies that can be used for conceptual product shape design and the challenges associated with applying them. We also discuss potential solutions to these challenges and point out future research directions.

- **A novel target-embedding variational autoencoder (TEVAE) architecture by integrating target-embedding representation learning with variational autoencoders (VAEs) and cross-modal synthesis methods with unimodal input based on the TEVAE**

  Trying to ensure the continuity of the latent space and improve the cross-modal synthesis process, we upgraded the traditional target-embedding autoencoders (TEAs) to target-embedding variational autoencoders (TEVAEs) by replacing the autoencoders with variational autoencoders. The proposed novel TEVAE architecture has facilitated two innovative cross-modal synthesis methods: the silhouette contour sketch-to-3D mesh model and the image-to-CAD sequence model. The silhouette contour sketch-to-3D mesh model is the first sketch-to-3D model that can take a contour sketch as input and output authentic 3D shapes that reflect humans' design intent and preferences. The image-to-CAD sequence model is also the first proposed method for reverse engineering to reconstruct 3D CAD models given a single image. More importantly, the direct outcome of the model is CAD sequence, which can provide knowledge that facilitates a better understanding of the 3D CAD modeling process and can offer flexibility in modifying steps to change the 3D geometry. In addition, CAD sequences can be parsed to 3D models using native CAD environment but the reverse process is not possible. The TEVAE architecture is general enough to be applied to different problems for cross-modal synthesis.

- **Multi-modal large language models for 3D CAD generation**

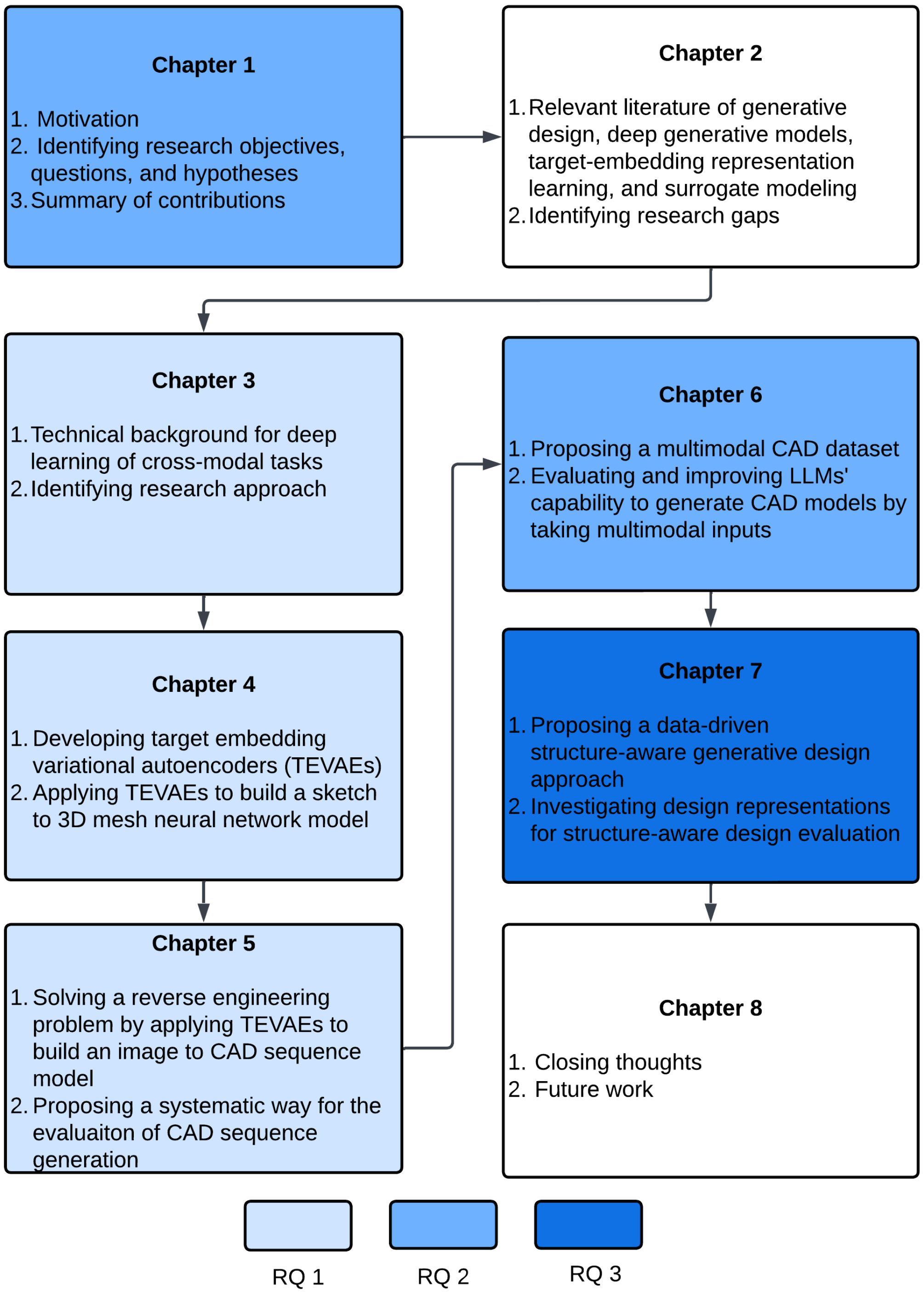
  We fill a research gap in the literature that no quantitative evaluation has been performed to assess the efficacy of multimodal LLMs in the CAD generation of 3D shapes for conceptual design. To that end, I first create a novel CAD dataset of five categories of mechanical components (i.e., shafts, nuts, flanges, springs, and gears with diverse geometry complexity) for testing multimodal LLMs, including textual descriptions, sketches, images, and 3D CAD models. In particular, textual descriptions of the target design objects are in natural languages with detailed dimensional information collected with Amazon Mechanical Turk, an online crowdsourcing platform. The effectiveness of the multimodal LLM GPT-4V model in 3D design generation is evaluated, and new knowledge of their strengths and limitations is obtained. In addition, a novel method is developed and implemented to enhance the GPT models' proficiency in generating 3D CAD models. Specifically, we develop a debugger to correct syntax errors in the synthesized CAD programs to improve their success rate of being translated to 3D CAD models.

- **Design representation for performance evaluation of 3D shapes in structure-aware generative design**

  Little is known about the efficacy of latent vectors acquired from the structure-aware data-driven generative design (DGD) training process, which encompasses both part-to-part structural information and geometric information. We fill this research gap by performing experiments to compare the performance of the latent vectors obtained from the training process of a structure-aware DGD model in predicting the engineering performance of the designs, with those obtained by embedding the generated 3D shapes (after training) using a 3D point grid (3DPG). The results indicate that while latent vectors are commonly used as vectorized design representations (VDRs) for AI-assisted design evaluation, they may not be suitable when the encoded information contains factors (e.g., structural information) that are of little relevance to the engineering performance of interest. The results have a broader impact on industry professionals because the use of appropriate VDR can lead to the improved predictive performance of design automation tools. A better prediction of engineering performance will also help designers make informed decisions in the early design stage when interacting with AI, facing a large number of design alternatives generated, thus potentially shortening the overall design cycle and reducing the development time.

1.4 Overview of the Dissertation

In Chapter 2, the relevant literature on generative design, deep generative models, target-embedding representation learning, and surrogate modeling is presented and the research gaps are identified. RQ 1 is answered in Chapters 3, 4, and 5. In Chapter 3, a systematic review of the technical background of deep learning of cross-modal tasks is presented. Chapter 4 presents the development of target embedding variational autoencoders (TEVAEs) and the application of TEVAEs to build a silhouette contour sketch-to-3D mesh neural network model. Chapter 5 introduces an image-to-CAD sequence model built on TEVAEs which can be employed for reverse engineering of 3D CAD objects. RQ 2 is answered in Chapter 6 which presents how we propose a multimodal CAD dataset to evaluate and improve LLMs' capability to generate CAD models by taking multimodal inputs. Chapter 7 answers RQ 3 and it presents a data-driven structure-aware generative design approach and our investigation of design representations for structure-aware design evaluation. Finally, Chapter 8 discusses the closing thoughts and future work. Figure 1.4 shows the dissertation overview and roadmap.


Chapter 1
1. Motivation
2. Identifying research objectives, questions, and hypotheses
3. Summary of contributions
Chapter 2
1. Relevant literature of generative design, deep generative models, target-embedding representation learning, and surrogate modeling
2. Identifying research gaps
Chapter 3
1. Technical background for deep learning of cross-modal tasks
2. Identifying research approach
Chapter 4
1. Developing target embedding variational autoencoders (TEVAEs)
2. Applying TEVAEs to build a sketch to 3D mesh neural network model
Chapter 5
1. Solving a reverse engineering problem by applying TEVAEs to build an image to CAD sequence model
2. Proposing a systematic way for the evaluaiton of CAD sequence generation
Chapter 6
1. Proposing a multimodal CAD dataset
2. Evaluating and improving LLMs' capability to generate CAD models by taking multimodal inputs
Chapter 7
1. Proposing a data-driven structure-aware generative design approach
2. Investigating design representations for structure-aware design evaluation
Chapter 8
1. Closing thoughts
2. Future work
RQ 1
RQ 2
RQ 3
Figure 1.4: Roadmap of the dissertation

Chapter 2: Literature Review

In this chapter, we review the existing literature in the fields of generative design, computer-aided design (CAD), target-embedding representation learning, and surrogate modeling. These areas represent the foundational technologies pivotal to achieving our research objective: the development of a human-supervised data-driven generative design framework. Following the review of the literature, we pinpoint a significant research gap that there is an absence of direct incorporation of human preferences and design intentions during the generative design process. In the concluding section of this chapter, we delve into strategies for addressing this identified gap, setting the stage for our contribution to the field.

2.1 Generative Design

Generative design (GD) methodology is drawing more and more attention and has been widely adopted in many engineering applications, such as wheel rim design (Oh et al., 2019; Yoo et al., 2020) and integrated into many commercial computer-aided design (CAD) software, such as Autodesk Fusion 360 (Keane, 2018). GD is an AI-driven design method that utilizes computer algorithms to automatically generate design concepts by considering human-defined objectives, parameter ranges, and constraints (McKnight, 2017; Krish, 2011; Matejka et al., 2018; Singh and Gu, 2012). GD automates the process of concept creation and design optimization, so it can quickly generate a large number of design variants, many of which could even be beyond human imagination. By repeating these computational routines many times, a variety of designs that meet the goal eventually emerge. Designers then review these outputs, often with the aid of interactive visual analytics for intuitive assessment and comparison across the board, and choose one or more designs for prototyping (van Kastel, 2018). As a result, it facilitates the exploration of the design space and the evaluation of various designs which can shorten the product design cycle and reduce the development cost (Mountstephens and Teo, 2020; Li and Lachmayer, 2019).

GD emerged as one of the most prominent computational design methodologies that use an iterative approach within a software program to generate outputs that meet certain constraints. A designer can fine-tune the feasible region with the help of the GD software by selecting or changing input values (or ranges) and their distribution. The GD software uses mechanical, materials, manufacturing, spatial, and cost-related constraints to generate more optimized designs.

However, there is a critical gap in current GD technologies: the absence of direct incorporation of human preferences and design intentions during the concept generation process (Urquhart et al., 2022; Demirel et al., 2023a). Despite the advancement of GD technologies, the indispensable role of human involvement in the initial stages of design remains evident. The domain expertise of human designers in evaluating functional performance, alongside the consideration of user aesthetic preferences, are crucial determinants that significantly impact the ultimate design outcome.

There are in general two types of GD methodologies: Rule-driven and data-driven (Regenwetter et al., 2022). Rule-driven GD (RGD) methodologies, such as shape grammars, genetic algorithms, and topology optimization, are primarily focused on optimizing engineering performance. These methods frequently yield designs of a complexity that requires additive manufacturing for realization (McKnight, 2017). Additionally, these approaches may inadvertently overlook the aesthetic preferences of users (Oh et al., 2019). The process of identifying relevant design rules for specific tasks is not only challenging but also time-intensive (Mountstephens and Teo, 2020). This presents a significant obstacle to the proliferation and advancement of RGD.

In contrast, data-driven GD (DGD) approaches are mainly developed based on deep generative models. Deep generative models are a class of machine learning models that aim to learn the underlying distribution of a dataset and generate new data points that are similar to the training examples (e.g., existing cars in the market), which have been applied in a wide range of fields, including image and speech recognition, natural language processing, and engineering design (Regenwetter et al., 2022). Two popular types of deep generative models are variational autoencoders (VAEs) (Kingma and Welling, 2013) and generative adversarial networks (GANs) (Goodfellow et al., 2014).

VAEs are a type of neural network that can learn a low-dimensional representation (usually called a latent vector) of high-dimensional data. VAEs consist of two neural networks: an encoder and a decoder. The encoder network takes in an input data point and maps it to a lower-dimensional representation, which is then fed into the decoder network. The decoder network then maps the lower-dimensional representation back to the original high-dimensional space, generating a new data point. VAEs are trained by minimizing a loss function that encourages the encoder to map similar data points to low-dimensional representations and the decoder to generate similar data points. The encoder network learns to generate a probability distribution over the low-dimensional representations of the input data, and the decoder network learns to generate a probability distribution over the high-dimensional data points given a low-dimensional representation.

GANs consist of two neural networks: a generator and a discriminator. The generator takes in a random noise vector and maps it to a data point, while the discriminator takes in a data point and predicts whether it is real or generated. GANs are trained by playing a minimax game between the generator and the discriminator. The generator tries to generate data points that are similar to the training examples and fool the discriminator into thinking they are real, while the discriminator tries to correctly classify whether a data point is real or generated. Through this adversarial training process, the generator learns to generate realistic data points that are similar to the training examples, while the discriminator learns to accurately distinguish between real and generated data points.

These methods derive latent design representations directly from data, without the need for explicit design rules. By learning the probability distribution of the training data, DGD methods are capable of generating novel designs by employing techniques of random generation and interpolation (Li et al., 2023b). Although DGD methods could implicitly capture human preferences, within the training dataset, they still fail to facilitate direct human engagement and oversight during the generative design process. DGD methods integrate and encode human preferences within the training data used to create final design artifacts.

Consequently, in a typical GD-centric design workflow, designers delineate objectives and constraints, delegate design generation to a GD engine, and subsequently select the most suitable designs from the generated pool. This approach, characterized by its open-loop nature and AI-in-the-charge, fails to position humans at the core of the design generation process. This oversight becomes particularly critical during the conceptual design phase, where human preferences (e.g., shapes, layouts, etc.) are paramount, contrasting with scenarios where engineering performance is the dominant criterion for design selection. To that end, for my dissertation research, I have primarily employed VAEs to construct human-supervised DGD methods to incorporate human designers' design intentions and preferences. More specifically, we devised a novel architectural framework called target-embedding variational autoencoder (TEVAE), which draws inspiration from target-embedding representation learning.

2.2 Computer-Aided Design

Modern computer-aided design (CAD) systems are instrumental in the creation, modification, analysis, and optimization of designs across various engineering fields. In this section, we review modeling techniques in CAD systems and data-driven CAD modeling methods.

2.2.1 Modeling Techniques in CAD Systems

Central to these systems are advanced modeling techniques such as constructive solid geometry (CSG), parametric modeling, boundary representations (B-rep), and feature-based design. Each of these methodologies contributes uniquely to the functionality and versatility of CAD systems.

CSG is a modeling technique that utilizes Boolean operations such as union, intersection, and difference to merge primitive shapes (e.g., blocks, cylinders, spheres, and cones) to create intricate solid models. These CSG models are structured as binary trees, with the primitive shapes as leaves and the Boolean operations as nodes. CSG offers several advantages, including the ease of modeling regular, symmetrical shapes and the ability to perform robust Boolean operations. However, it may be less efficient for representing highly detailed or irregular geometries. Despite this, the use of CSG in modern CAD systems is widespread, especially in applications that require precise control over the geometric and topological properties of the model (Requicha, 1980).

On the other hand, parametric modeling is a robust method used in CAD that employs parameters to define the geometry and constraints of a design. These parameters can include dimensions, angles, or other measurable properties. Making changes to these parameters will automatically update the entire model, maintaining the relationships and constraints between different elements. This approach improves design flexibility and efficiency, allowing designers to quickly iterate and refine models. Parametric modeling is essential in modern CAD systems to build complicated systems of designs for applications such as mechanical part design, architectural modeling, and industrial design, where iterative modifications are common (Shah and Mäntylä, 1995).

The B-rep format is the widely accepted standard for CAD models, offering precise analytical representations of surfaces and curves for advanced control of 3D shapes (Weiler, 1986). This method provides a detailed description of the shape and structure of a 3D object by representing its boundaries through faces, edges, and vertices. B-rep models are commonly used in CAD systems for their accuracy and flexibility, particularly in applications that demand intricate surface modeling, such as automotive and aerospace design. They also enable advanced operations such as surface smoothing, trimming, and joining (Mäntylä, 1987).

Feature-based design or modeling extends beyond traditional geometric modeling by including higher levels of information, making design environments more productive and allowing for greater automation in downstream applications. Features such as holes, slots, pockets, and bosses are directly tied to design intents and constraints, making the design process more intuitive and aligned with manufacturing capabilities. This approach enables designers to focus on higher-level functional aspects rather than low-level geometric details. Feature-based design is integral to modern CAD systems, supporting advanced applications such as automated manufacturing, assembly planning, and product lifecycle management (Shah and Mäntylä, 1995). In this method, users interactively define features by selecting topological entities (such as edges or faces) from a geometric model. Several schemes of representing features have been proposed, such as augmented graphs, syntactic strings in grammars, and object-oriented programming (Shah, 1991). Feature-based design offers significant advantages in creating rich, high-level databases that facilitate advanced design and manufacturing processes (Shah, 1991; Shah and Mäntylä, 1995).

2.2.2 Data-Driven CAD Modeling Methods

The advancements discussed above have significantly enhanced the capabilities of CAD tools, empowering designers to create complex models more efficiently. Modern CAD software such as SolidWorks, Fusion 360, CATIA, and Onshape are now more intuitive and user-friendly. However, there remains a steep learning curve to effectively use these programs, and proficiency in both design principles and software operations is crucial for designers. For example, an experienced designer accustomed to using pencils, rulers, and paper may struggle to adapt to a company's current requirements due to a lack of CAD skills.

The goal is to streamline the design process, allowing designers to focus on their creative vision rather than on time-consuming operations and adjustments using CAD systems. Ideally, systems would leverage input from designers—such as "A stylish table suited for dining"—to autonomously generate design concepts. Furthermore, it would be highly beneficial if computers could also suggest design ideas or subsequent CAD operations based on their understanding of the designer's historical CAD operations and general or specific design principles. This vision is becoming increasingly attainable with the surge in data-driven CAD modeling with breakthroughs in artificial intelligence (AI) technologies.

CAD models are structured designs, allowing for precise control and flexibility in the design process. They are particularly suitable for industrial and engineering designs, where precision and the ability to easily modify designs are crucial. As introduced previously, there are two main types of CAD models: CSG and parametric CAD models (including CAD sequence data and B-rep models) (Wu et al., 2021; Para et al., 2021). In contrast, discrete 3D representations, such as meshes and point clouds, are more static and less flexible in terms of parametric editing and design exploration (Wu et al., 2021; Para et al., 2021). In the following, we will review the status of data-driven methods for CAD models.

Numerous research efforts have focused on deep learning techniques for the creation of CAD models using CSG (Ren et al., 2021; Sharma et al., 2017, 2019; Kania et al., 2020). Despite its capabilities, CSG does not offer the adaptability found in modern CAD utilities that employ parametric modeling techniques. On the other hand, parametric CAD models are initiated from 2D outlines consisting of basic geometric shapes (e.g., squares and circles) connected with explicit constraints such as alignment and right angles, which serve as a basis for 3D modeling methods (e.g., extruding or sweeping). For parametric CAD models, associated deep learning approaches cover the creation and reconstruction of 2D engineering drawings and 3D objects. Contemporary studies (Willis et al., 2021a; Seff et al., 2021; Ganin et al., 2021; Para et al., 2021; Yang and Pan, 2022) have been concentrated on utilizing deep learning for generating CAD drawings, with a focus on generating 2D configurations that precede the creation of 3D shapes. The B-rep format is the widely used standard for 3D shape definition in CAD systems. Recent developments include learning-based methods for generating parametric curves (Wang et al., 2020) and surfaces (Sharma et al., 2020). Moreover, Smirnov et al. (Smirnov et al., 2020) have developed a generative approach to create a model topology that integrates these curves and faces using specific topological templates. Some techniques facilitate the creation of B-rep models with complex and varying topologies (Guo et al., 2022; Jayaraman et al., 2022; Wang et al., 2022b).

Advances have also been seen in automated CAD sequence creation to reconstruct 3D models, especially through the *Sketch-and-Extrude* method. Unconditional generation of these CAD sequences has been the focus of these methods (Wu et al., 2021; Xu et al., 2022), which are geared towards independently producing CAD sequences. Notably, Wu et al. (Wu et al., 2021) introduced DeepCAD, a pioneering generative model that learns from CAD modeling operations to generate modifiable designs. By comparing CAD operation sequences to natural language, they have utilized a transformer (Vaswani et al., 2017) framework, exploiting its proficiency in sequence comprehension and production, to suit the specific needs of CAD design tasks.

Generative models can indeed produce a wide array of designs, providing sources of inspiration and the exploration of varied design options. Nonetheless, these models do not inherently include the designer's intentions within the generative process. Therefore, the outcomes may not meet the designer's expected criteria or specific needs (Li et al., 2022c; Demirel et al., 2023a). This issue underscores the need for methods that enable designers to guide the generative process, ensuring the resultant designs are more in line with their objectives and preferences (Demirel et al., 2023a). To address this, diverse approaches have been proposed to guide the creation of CAD sequences for targets such as B-rep models (Willis et al., 2021b; Xu et al., 2021), voxels (Lambourne et al., 2022; Li et al., 2023a), point clouds (Uy et al., 2022; Ren et al., 2022), and sketches (Li et al., 2020b, 2022a). Specifically, Fusion 360 Gym (Willis et al., 2021b) was created to rebuild a CAD model from a B-Rep model by applying a technique based on face extrusion, which utilizes the planar faces present in the B-Rep model. Nevertheless, despite its capability for generating CAD sequences, the face-extrusion technique differs substantially from the more conventional sketch-extrusion approach typically employed by designers. This method also falls short when the input data, such as images, lacks sufficient planar or profile information.

These AI-driven CAD systems hold the promise of revolutionizing design by taking over the more tedious elements of CAD modeling, thereby boosting productivity and promoting creativity. Yet, there is a noticeable gap in the research, specifically on producing CAD sequences from images that could facilitate more intuitive and user-friendly interactions with intelligent CAD systems, which we refer to as **CAD Copilot**. We envision CAD Copilot to integrate capabilities like ChatGPT directly into CAD software, enabling users to interact with their design environment using multiple input modalities, such as natural language descriptions, images, and audio. CAD copilot could also be enabled to suggest design ideas and next-step CAD operations based on designers' historical CAD operations like Google Gmail's suggestions on writing. This could potentially streamline workflows, enhance design exploration, and democratize access to complex CAD tools.

2.3 Target-Embedding Representation Learning

Girdhar et al. propose a TL-embedding network (Girdhar et al., 2016) that is composed of a T-network for training and an L-network for testing. The T-network contains an autoencoder (encoder-decoder) network and a convolutional neural network (CNN). After training, the L-network can be used to predict 3D shapes in voxels from images. Similarly, Mostajabi et al. (Mostajabi et al., 2018) use an autoencoder and a CNN to perform the semantic segmentation task of images. Dalca et al. (Dalca et al., 2018) apply a similar network structure as (Girdhar et al., 2016; Mostajabi et al., 2018), consisting of a prior generative model, to generate paired data (biomedical images and anatomical regions) to solve the scarcity of labeled image data for anatomical segmentation tasks. Jarrett and Schaar (Jarrett and van der Schaar, 2020) categorize these studies as supervised representation learning methods. They observe that when the dimension of the target data space is higher or similar to the feature data space, a target-embedding autoencoder (TEA) can be more effective than a feature-embedding autoencoder (FEA). The authors verify that the TEA structure will guarantee learning stability by using a mathematical proof of a simple linear TEA and showing the empirical results from a complex non-linear TEA. Inspired by those existing works, we developed the target-embedding variational autoencoder (TEVAE) architecture, which has been applied in my research to build AI-assisted concept generation methods that are both predictive and generative by taking human factors as input.

2.4 Surrogate Modeling

Surrogate models are commonly used in engineering design, optimization, and simulation-based decision-making, where they can provide accurate predictions and enable rapid exploration of the design space. Surrogate models are simplified mathematical or statistical models that are developed to approximate the behavior of a complex, computationally expensive model (Wang and Shan, 2007; Sobester et al., 2008).

In many real-world applications, the underlying model that describes the relationship between input variables and output variables may be too computationally expensive to evaluate directly or may involve physical processes that are difficult to simulate. In such cases, surrogate models can be used to reduce the computational burden by creating a simpler, faster-to-evaluate model that captures the essential features of the original model. Surrogate models can be developed using various techniques such as regression analysis, machine learning algorithms, or optimization techniques. Surrogate models require that a design be represented as a fixed-length vector of design features (i.e., vectorized design representation) (Whalen and Mueller, 2022). We build the AI-assisted concept evaluation methods using surrogate model methods (e.g., Gaussian process, locally linear embedding) and compare the prediction accuracy of engineering performance using different design representations (e.g., latent vectors from the learned neural network, embedding feature vectors using the generated designs).

2.5 Bridging the Gap with Cross-Modal Synthesis

In identifying the significant gap within the current landscape of generative design—the absence of direct incorporation of human preferences and design intentions—it becomes imperative to explore innovative strategies that can effectively bridge this divide. Our research identifies cross-modal synthesis as a promising avenue towards achieving this goal. Cross-modal synthesis, by its nature, offers a unique framework for integrating diverse inputs, including human feedback, into the generative design process. This approach not only enables the direct translation of human insights into design outputs but also enhances the adaptability and relevance of the design outcomes to human needs and preferences.

The potential of cross-modal synthesis in our context lies in its ability to facilitate a more dynamic interaction between the human supervisor and the generative design framework. By leveraging cross-modal synthesis, we propose a novel method where human preferences and design intentions are encoded into a form that is seamlessly integrated into the generative process. This methodology ensures that the design output is not only data-driven but also closely aligned with human aesthetic and functional requirements. Furthermore, the application of cross-modal synthesis in this framework addresses the critical need for a more intuitive and interactive design process. It allows for the iterative refinement of designs based on real-time human feedback through intuitive design modalities, such as text, sketches, or images, thus significantly enhancing the efficiency and effectiveness of the generative design process.

In conclusion, our exploration into the utilization of cross-modal synthesis stands as a cornerstone of our approach to mitigating the identified gap in the field. This strategy not only aligns with our objective of developing a human-supervised data-driven generative design framework but also sets a new direction for future research in this area. By advancing this innovative approach, we aim to contribute significantly to the field, offering a model that better incorporates human insight into the generative design process, thereby enriching the quality and applicability of design outcomes.

To deepen our understanding of cross-modal synthesis methods in engineering design, Chapter 3 is dedicated to a systematic literature review on the methods for deep learning of cross-modal tasks (DLCMT). This review aims to uncover the technologies, potentials, and challenges inherent in cross-modal tasks, e.g., cross-modal synthesis to build our knowledge foundation for the dissertation research.

Chapter 3: Deep Learning Methods of Cross-Modal Tasks for Conceptual Design of Product Shapes: A Review

Abstract

In conceptual design, product shape design is one of the most paramount aspects. When applying deep learning-based methods to product shape design, two major challenges exist: 1) design data exhibit in multiple modalities, and 2) an increasing demand for creativity. With recent advances in deep learning of cross-modal tasks (DLCMT), which can transfer one design modality to another, we see opportunities to develop artificial intelligence (AI) to assist the design of product shapes in a new paradigm. In this paper, we conduct a systematic review of the retrieval, generation, and manipulation methods for DLCMT that involve three cross-modal types: text-to-3D shape, text-to-sketch, and sketch-to-3D shape. The review identifies 50 articles from a pool of 1341 papers in the fields of computer graphics, computer vision, and engineering design. We review 1) state-of-the-art DLCMT methods that can be applied to product shape design and 2) identify the key challenges, such as lack of consideration of engineering performance in the early design phase, that need to be addressed when applying DLCMT methods. In the end, we discuss the potential solutions to these challenges and propose a list of research questions that point to future directions of data-driven conceptual design.

3.1 Introduction

[1] The product shape is essential in the conceptual design of engineered products because it can affect both the aesthetics and the engineering performance of a product (Ulrich, 2003). Figure 3.1 shows the flow of information and the key steps for the design of product shapes at the conceptual design stage(Ulrich, 2003), where the information can be categorized into three modalities: natural language (e.g., text), sketches (e.g., 2D silhouette), and 3D shapes (e.g., meshes). We call them design modalities. Generally, customer needs and engineering requirement documents are in the form of natural languages. Design sketches and drawings are effective ways of brainstorming and expressing designers' preferences. Low-fidelity design concepts and prototypes from the conceptual design stage are often represented by 3D shapes in digital format. Design Search, Design Creation, and Design Integration are the core steps of conceptual design to gather information from existing design solutions for inspiration and to develop novel design concepts to better explore the design space (Ulrich, 2003).

Early design automation methods, such as grammar- and rule-based methods, rely primarily on human design experience and knowledge to generate design alternatives (Chakrabarti et al., 2011). In contrast, deep learning methods can learn latent design representations from data without explicit design rules or grammar, so they have been increasingly adopted in many engineering design applications. So far, however, deep learning methods have been applied mainly in the later stages of engineering design for design automation (Regenwetter et al., 2022). It is challenging to apply deep learning methods to the conceptual design stage (i.e., the early design stage) for several reasons. For example, data in the conceptual design stage exhibit

[1]This chapter has been published in the following paper in the Journal of Mechanical Design. Li, X., Wang, Y., and Sha, Z. (January 10, 2023). Deep Learning Methods of Cross-Modal Tasks for Conceptual Design of Product Shapes: A Review. *ASME. J. Mech.* Des. April 2023; 145(4): 041401. `https://doi.org/10.1115/1.4056436`. I am the leading author of this paper and was primarily responsible for the conceptual framework, literature review, and writing of the manuscript.

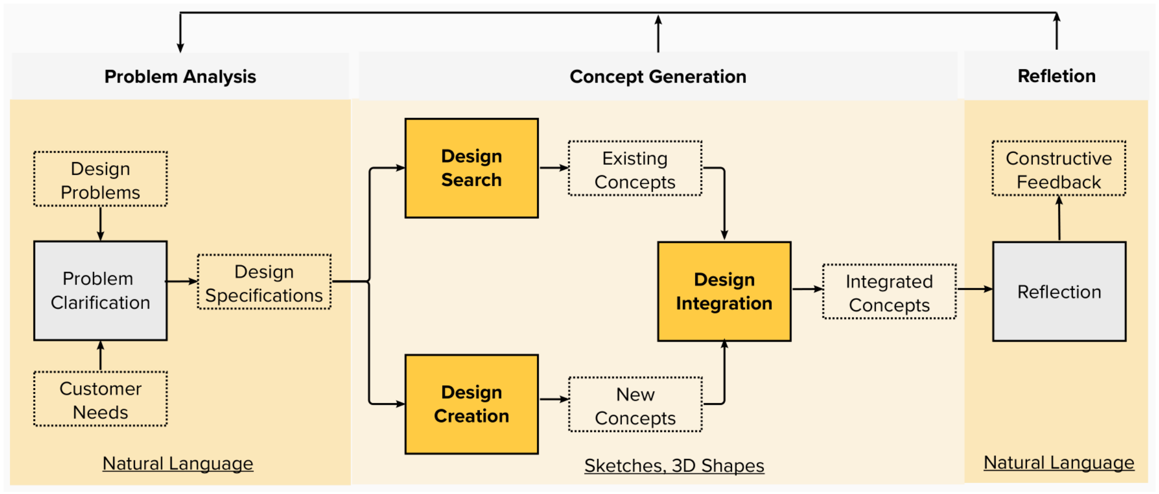


Figure 3.1: Iterative conceptual design stage in the development of engineered products

multiple modalities, but deep learning methods are usually applied to handle a single design modality. Moreover, in conceptual design, designers often gather a large set of information for design inspiration in different design steps, but deep learning methods tend to focus on one specific design task at a time. Finally, human (either user or designer) input and interactions are desired in conceptual design to improve design creativity and human-centered design, but most current design methods developed using deep learning do not interact directly with human data, but only implicitly capture human preferences from training datasets, as shown in Figure 3.2.

With recent development in deep learning of cross-modal tasks (DLCMT) [2], we see the opportunities of applying these methods to address the aforementioned challenges, particularly in product shape design, such as car body and plane fuselage (Smirnov et al., 2020; Guillard et al., 2021). DLCMT allows explicit human input in

[2]DLCMT is a class of problems, aiming to translate one modality of data to another, e.g., from text to 3D shapes. To solve this problem, there is a large body of literature on cross-modal representation learning (CMRL). CMRL aims to build embeddings using information from multiple modalities (e.g., texts, audio, and images) in a common semantic space, which allows the model to compute cross-modal similarity (Liu et al., 2020a). In this paper, our review is not limited to reviewing CMRL methods but also includes other deep learning methods that can solve cross-modal problems.

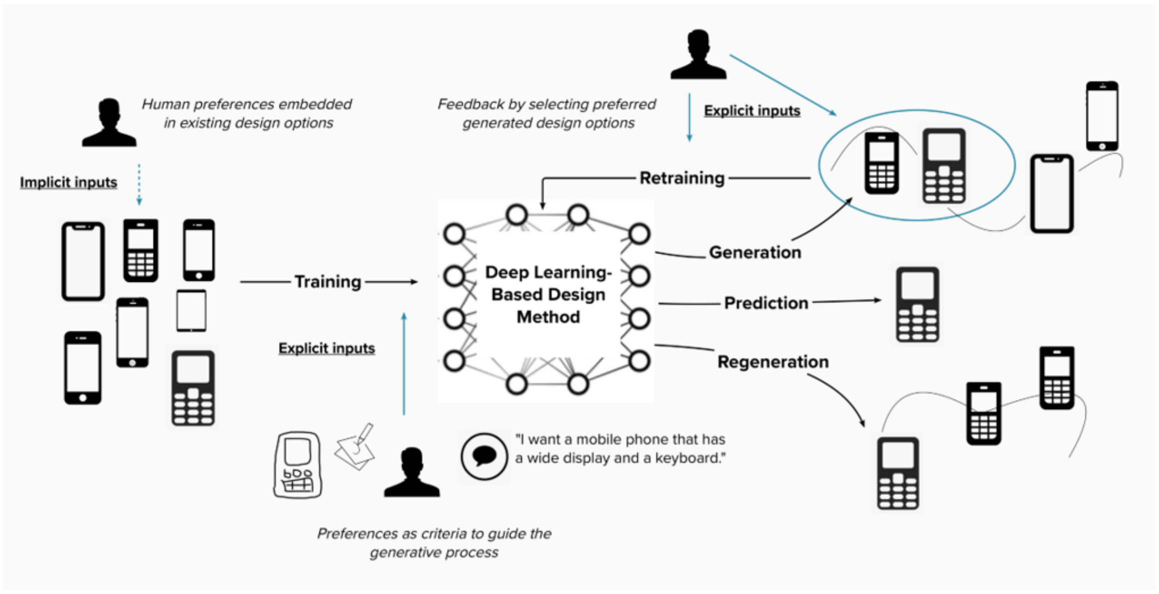

Human preferences embedded in existing design options
Implicit inputs
Feedback by selecting preferred generated design options
Explicit inputs
Retraining
Training
Deep Learning-Based Design Method
Generation
Prediction
Regeneration
Explicit inputs
"I want a mobile phone that has a wide display and a keyboard."
Preferences as criteria to guide the generative process

Figure 3.2: Deep learning-based design process with humans in the loop

one design modality and translates it to another modality, e.g., from natural language or sketches to 3D shapes, as shown in Figure 3.2. In DLCMT, there are cross-modal retrieval, generation, and manipulation methods. Cross-modal retrieval methods can be used to search an existing design repository for inspiring design ideas. Cross-modal generation methods can be used to explore a design space to generate new design concepts. Lastly, cross-modal manipulation methods can further edit and manipulate existing designs to refine designs. These three categories of methods can be used in the Design Search, Design Creation, and Design Integration steps (Figure 3.1), respectively. In this paper, we conducted a systematic review of the state-of-the-art methods for DLCMT. Through a close examination of the existing literature, our objective is to identify the DLCMT methods and technologies that can be used to facilitate the conceptual design and the challenges associated with applying them.

A total of 50 recently published journal articles and conference papers are identified and closely reviewed from the fields of computer graphics, computer vision, and engineering design. We focus on the text, sketches, and 3D shapes because they are the main design modalities in conceptual design. Specifically, we reviewed deep learning methods for three types of cross-modal tasks: text-to-sketch, text-to-3D, and sketch-to-3D. We found that most of the literature comes from computer graphics and computer vision, with few attempts at engineering design applications. This poses new challenges and opportunities for adapting the models and techniques developed to solve engineering design problems and, particularly, to bridge human input and interactions with deep learning methods in the conceptual design of engineered product shapes.

The remainder of this paper is organized as follows. Section 3.2 introduces background knowledge on conceptual design, design modalities, and our motivation for the review. Section 3.3 presents the methodology for our systematic review. We tabulate all the reviewed articles and present four statistics from the literature in Section 3.4. We then discuss the literature in detail and answer the research questions of the systematic review in Section 3.5. In the end, we propose a list of six research questions that will inform future research directions in Section 3.6 and conclude our work with closing remarks in Section 3.7.

3.2 Background

3.2.1 Conceptual Design

Conceptual design lies in the early phase of a design process in which the form and function of a product are explored (Otto and Wood, 2001). In conceptual design, it is crucial to explore the design space as much as possible, and designers are demanded to generate creative designs so that the products are likely to succeed in the market (Yang, 2009; Hyun and Lee, 2018). As shown in Figure 3.1, we adapt and reinterpret the five-step concept generation method in conceptual design (Ulrich, 2003). The five steps are Problem Clarification, Design Search, Design Creation, Design Integration, and Reflection. Through these five steps, the method transfers information, such as customer needs, engineering requirements, and design ideas, to design concepts in the form of sketches and 3D shapes. The corresponding input

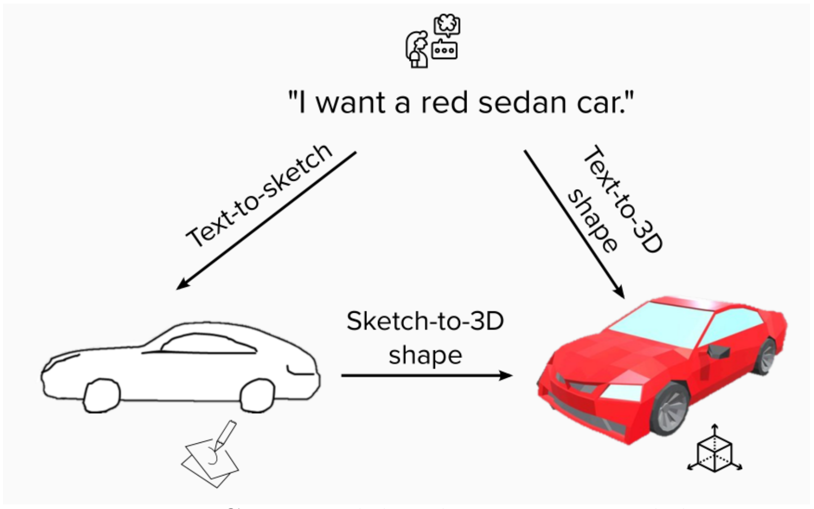


Figure 3.3: Cross-modal tasks in conceptual design

and output of each step are represented by dotted rectangles. The process is linear in sequence from left to right, but almost always iterative. For example, feedback from Reflection could influence Problem Clarification and its subsequent steps. Each design step can also be iterative so that the design problem can be better understood, and the design space can be better explored (Ulrich, 2003).

In the conceptual design phase, the shape of a product is one of the most important considerations that are influential on the aesthetics of a product and its engineering performance (Ulrich, 2003; Mountstephens and Teo, 2020). In this paper, we focus primarily on reviewing the DLCMT methods that can be applied for product shape design in the three concept generation steps, i.e., Design Search, Design Creation, and Design Integration, because they are the core steps for design concept exploration.

3.2.1.1 Design Search

Design Search is the step of collecting information on existing design solutions to a design problem. In practice, several ways, such as patents, literature, and benchmarking, can be used to gather useful information (Ulrich, 2003). By analyzing those existing products, designers can summarize their advantages and disadvantages, so that they can make necessary and customized changes to existing designs to create satisfying ones. However, the repository of existing design options could be huge, so the search process would be time-consuming and cumbersome, placing significant cognitive and physical burdens on designers. One possible solution to this problem is to use an AI-assisted search process, where designers can predefine search criteria and utilize computers to search for relevant design solutions.

3.2.1.2 Design Creation

Design Creation emphasizes exploring novel design concepts. Designers brainstorm ideas and explore the design space to create novel design concepts based on the knowledge of designers (Ulrich, 2003). Design ideas are often presented as sketches and text descriptions during conceptual design (Ahmed et al., 2018). Text descriptions are used to document and describe designers' ideas, while sketches can help visualize design concepts, further triggering creative design ideas (Krish, 2011; Pratt et al., 2005; Menezes and Lawson, 2006). Low-fidelity 3D models are then created for better visualization and further development. However, creating 3D models involves a lot of manual work and could be time-consuming. To facilitate the creation of novel 3D shapes, generative design methods can be used to automate the process.

3.2.1.3 Design Integration

The Design Integration is the step where designers aim to systematically integrate the information collected from previous steps to generate the integrated design concept(s) (Ulrich, 2003). For product shape design, designers usually need to edit and manipulate designs collected from the Design Search and Design Creation steps. But, it can be challenging to modify these designs computationally because their representations have certain formats (e.g., a 3D shape in voxels or point clouds or a sketch of a raster image). Some formats are not editable and must be translated into other formats, such as mesh or B-rep. Therefore, automating the modification with

Table 3.1: The text types of natural language data used in DLCMT and the examples

Text Type	Examples
Natural language descriptions (NLD)	"It's a round glass table with two sets of wooden legs that clasp over the round glass edge".
Object names	"chairs", "cars", "planes"
Semantic keywords	"circular short", "rectangular wooden"

human inputs can significantly simplify the process.

3.2.2 Modalities in Conceptual Design

As shown in Figure 3.1, there are three main design modalities: natural language (NL), sketches, and 3D shapes, in conceptual design. In an example of car body design, as shown in Figure 3.3, the three modalities could be "I want a red sedan car" (NL), hand-sketching a car with desired features (sketch), and then creating a computer-aided design (CAD) model of the car (3D shape). NL allows people to convey and communicate ideas and thoughts. It is also the primary means for documentation, such as documentation of customer needs and engineering requirements. Sketches are often used to brainstorm design concepts because sketching can stimulate designers' creative imagination (Krish, 2011; Pratt et al., 2005; Menezes and Lawson, 2006). Then, a 3D shape is often built to provide better visualization and a low-fidelity prototype model for further evaluation and development of a concept.

NL data are often in the format of the text, which is usually the keyword in DLCMT methods. As shown in Table 3.1, there are mainly three types of text used as input in DLCMT, which are natural language descriptions (NLD), object names, and semantic keywords. 2D sketches can be represented in multiple ways, such as a pixel image [3] in static pixel space and vector image in dynamic stroke coordinate space (Xu

[3]Images can include both sketches and natural photos. In the literature, we notice that DLCMT methods of natural photos usually use "image" while the methods of sketches use "sketch" as the keyword, respectively. Also, in engineering design, sketches are usually considered as lines and strokes. To identify DLCMT methods for engineering design, we exclude corresponding methods of

et al., 2020; Ha and Eck, 2018). There are also generally two types of 3D sketches in the literature, and we refer to them as Type I and Type II, respectively. Type I: This kind of 3D sketch is represented in a 2D space. But compared to regular 2D sketches, they look like 3D objects. Type II: The 3D sketches that can be represented in a 3D space (either real or computational). Such a type of 3D sketch data can be captured and generated using virtual reality (VR) tools or motion sensing devices. They can also be created using 3D sketching software (e.g., SolidWorks or Autodesk). 3D shapes are typically built as B-rep models using CAD software in engineering design. However, in computer graphics and the 3D deep learning fields, 3D shapes are usually represented as meshes, point clouds, and voxel grids. Compared to CAD models, these 3D representations typically have lower fidelity with fewer geometric details and structural information because (1) coarse resolution might be used to represent the shapes due to the limitations of computational resources (Chen et al., 2018; Fukamizu et al., 2019), (2) certain representations are not good at representing geometric details and topological structure by nature (e.g., point clouds; see Table 3.2 for more information), (3) the conversion of one representation to another might lose geometric or topological information (Nozawa et al., 2020, 2022).

3.2.3 Review Motivation

Our motivation for this literature review is driven by the following two major challenges posed in conceptual design. The recent advancement in DLCMT gives us opportunities to address these challenges and bring new design experiences in conceptual design.

Challenge 1: Multi-modalities. There are multiple design steps (e.g., Design Search, Design Creation, and Design Integration) in the conceptual design stage, which involve information and data with different design modalities. Designers conduct design activities with different modalities during the conceptual phase to

"image".

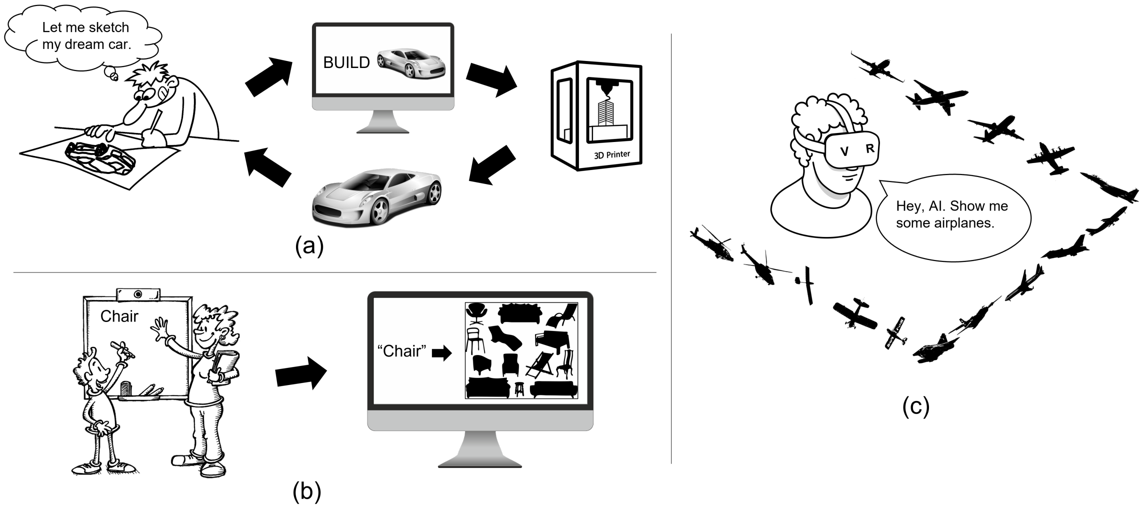


Figure 3.4: Potential design applications enabled by DLCMT: (a) Democratization of product design; (b) AI-based pedagogical tools for educating and training students or novice designers; (c) Immersive design environment

best explore the design space and generate novel ideas (Wendrich, 2018; Song et al., 2022).

Deep learning methods that can be used for Design Creation have been the focus (Regenwetter et al., 2022), but most of them are focused on handling a single design modality as pointed out by (Song et al., 2020; Li et al., 2022b). Typically, these methods use unimodal data of designs either in 2D (Chen et al., 2020; Oh et al., 2019; Dering et al., 2018) or 3D (Shu et al., 2020; Zhang et al., 2019; Li et al., 2021c; Brock et al., 2016). In addition, there is a lack of either unimodal or cross-modal methods that are useful for Design Search and Design Integration (Li et al., 2022b).

But not until recently, we see studies in the engineering community utilizing DLCMT to assist concept creation or design evaluation (Qin et al., 2022; Li et al., 2022c; Song et al., 2020). DLCMT methods take into account multiple design modalities, such as texts and sketches. There are retrieval, generation, and manipulation methods for DLCMT and they can be applied to different steps in conceptual design: (1) DLCMT retrieval methods can be used for Design Search since they can search existing data and return designs that best match the query of users (e.g., returning several chairs given a query by sketch) (Qi et al., 2021); (2) Generation methods (e.g., sketch-to-3D shape generation methods (Li et al., 2022c; Lun et al., 2017)) can be used to automate the Design Creation process; (3) Manipulation methods can allow designers to modify the designs from another design modality. For example, using a text-to-3D manipulation method (Michel et al., 2022), designers can modify a 3D design by providing a simple text description without direct manipulation of the design, and this can significantly reduce the time for the design modification.

Challenge 2: Creativity. Design creativity is critical in conceptual design which can largely affect the success of a product in the market. There are three main aspects (i.e., design novelty, contextual information, and human-computer interaction) that should be addressed for design creativity in the context of deep learning-based design processes.

- *Design novelty.* Deep learning methods (e.g., VAEs and GANs) can generate new data that are not seen in the training dataset but are still based on interpolation within the boundary of the training data. Therefore, the new designs generated from the deep learning-based design process share great similarities with the existing ones used as training data. To improve design creativity, there have been a few deep learning-based methods that focus on developing neural network architectures to generate creative designs by enabling deep-learning models' extrapolating capabilities (Elgammal et al., 2017; Chen and Ahmed, 2021). These methods pose new opportunities for design because they can generate truly novel designs.

- *Contextual information.* On the other hand, humans have played an essential role in design creativity. However, despite advances in the development of network architecture, one observation is that human input and interaction are not much emphasized in the deep learning-based design process (Regenwetter et al., 2022). Burnap et al. (Burnap et al., 2016) pointed out that a human's perception of the quality of the design concepts generated is often not in agreement with their numerical performance measures. The reason could be that in most deep learning-aided design processes, designers can only passively select the preferred design concepts from a set of computer-generated design options, but human designers may have contextual information (Judd and Steenkiste, 2003) on a design problem which is hard to be captured by the training data.

- *Human-computer interaction.* As a result, there is a need to actively involve designers in a deep learning-based design process (Mountstephens and Teo, 2020; Regenwetter et al., 2022). Some efforts in this regard have recently been made in engineered product design. For example, the method introduced by (Valdez et al., 2021) allows users to manipulate the latent space vectors learned by a GAN model to create preferred design options. Despite recent advances, we believe that design creativity can be further improved by involving humans in the design process to allow more intuitive and natural human input (e.g., text and sketch). Natural language and sketches are the most common human input in conceptual design, and DLCMT methods can intake these human inputs and transfer their modalities from one to another to promote creativity. That is manifested in the envisioned deep generative design process with humans in the loop, as shown in Figure 3.2. In such a process, designers can continuously supplement new design ideas during human-computer interaction to guide computers to generate creative and feasible design concepts.

In addition, there should be many design processes and applications that can be facilitated by DLCMT and we show three typical examples in Figure 3.4. *Design application 1:* DLCMT methods can be used to facilitate design democratization, allowing ordinary people to customize designs based on individual preferences (Starly et al., 2019). *Design application 2:* There are also opportunities to develop AI-based pedagogical tools to teach students or train novice designers, allowing them to explore design alternatives with naive input, for example, just a simple word (Sanghi et al., 2022). *Design application 3:* Immersive design uses virtual reality (VR), augmented reality (AR), and mixed reality (MR) to create a realistic digital environment in which a user is virtually immersed and can even physically interact with the digital environment (Giunchi et al., 2021). The DLCMT methods can be integrated into immersive design applications to enhance the design experience in human-computer interaction.

In summary, DLCMT methods are likely to introduce new opportunities to support and enhance activities in the conceptual design stage for product shape design and beyond. We conduct a close examination of the existing literature aiming to identify the existing DLCMT methods and technologies that can be used for conceptual product shape design and the challenges associated with applying them. We will also discuss potential solutions to these challenges and point out future research directions.

3.3 Methodology

This study adopts a systematic literature review approach (Khan et al., 2003) with the procedure of formulating research questions for a review, identifying relevant studies, evaluating the quality of the studies, summarizing the studies, and interpreting the findings.

3.3.1 Research Questions

We are motivated to ask two research questions (RQs).

RQ 1. What DLCMT methods can be used in the following three steps of conceptual design?

(1) Design Search

(2) Design Creation

(3) Design Integration

RQ 2. What are the challenges in applying DLCMT to conceptual design and how can they be addressed?

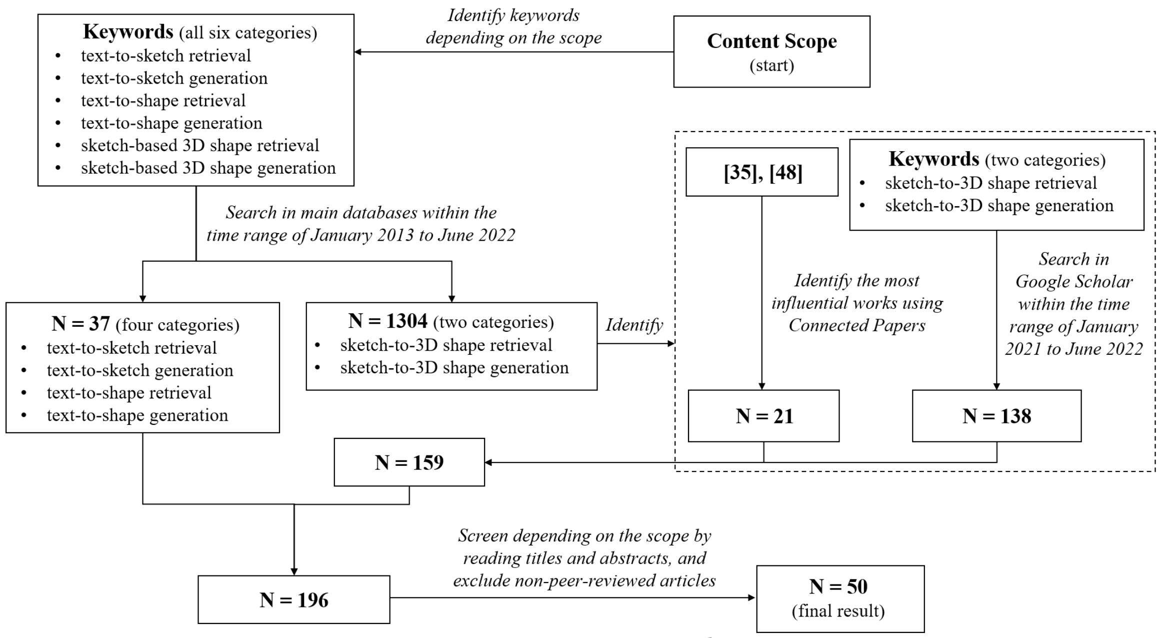
3.3.2 Literature Search

3.3.2.1 Content Scope and Keywords

We defined the *content scope* using the following three criteria to search the literature relevant to deep learning of cross-modal tasks (DLCMT): (1) Conceptual design: Design Search, Design Creation, and Design Integration steps (highlighted in Figure 3.1). (2) Shape design: discrete, physical, and engineered products. (3) Design modality: text, sketch, and 3D shape.

The keywords identified and used in the literature search process are *"text-to-sketch retrieval", "text-to-sketch generation", "text-to-shape retrieval", "text-to-shape generation", "sketch-based 3D shape retrieval",* and *"sketch-based 3D shape generation".* For "sketch-based 3D shape generation", we include the other three commonly used names: *"sketch-based 3D shape reconstruction", "sketch-based 3D shape synthesis",* and *"3D shape reconstruction from sketches".*

The reasons for choosing these keywords come from the following aspects. (1) DLCMT between two different modalities of text, sketch, and 3D shape, should have six permutations of cross-modal. In this paper, we focus on the following three cross-modal tasks: text-to-sketch, sketch-to-3D shape, and text-to-3D shape, which are then concatenated with retrieval or generation to form the initial keywords (e.g., text-to-sketch generation). We did not include sketch-to-text, 3D shape-to-sketch, and 3D shape-to-text because sketches or 3D shapes are often the most common artifacts, and the design information flows in an order of text, sketches, and 3D shapes during the conceptual design. (2) we focus on Design Search which corresponds to retrieval methods, Design Creation which corresponds to generation methods, and Design



Content Scope (start)
Identify keywords depending on the scope
Keywords (all six categories)
text-to-sketch retrieval
text-to-sketch generation
text-to-shape retrieval
text-to-shape generation
sketch-based 3D shape retrieval
sketch-based 3D shape generation
Search in main databases within the time range of January 2013 to June 2022
N = 37 (four categories)
text-to-sketch retrieval
text-to-sketch generation
text-to-shape retrieval
text-to-shape generation
N = 1304 (two categories)
sketch-to-3D shape retrieval
sketch-to-3D shape generation
Identify
[35], [48]
Keywords (two categories)
sketch-to-3D shape retrieval
sketch-to-3D shape generation
Identify the most influential works using Connected Papers
Search in Google Scholar within the time range of January 2021 to June 2022
N = 21
N = 138
N = 159
N = 196
Screen depending on the scope by reading titles and abstracts, and exclude non-peer-reviewed articles
N = 50 (final result)

Figure 3.5: Literature search process

Integration which corresponds to manipulation methods [4]. In addition, for the sketch-to-3D shape retrieval or generation methods, we made some modifications to the keywords according to the naming convention in the literature (see a comprehensive review on deep learning methods for free-hand sketch (Xu et al., 2020)). For example, we used "sketch-based 3D shape retrieval" instead of "sketch-to-3D shape retrieval" and the other three common terms introduced previously.

3.3.2.2 Literature Search Process

As shown in Figure 3.5, we finally selected 50 articles that meet our scope of review. Searches were conducted on the main databases of the literature (i.e.,

[4]We did not explicitly search for cross-modal manipulation methods because these methods cannot be found directly using specific keywords, but can be indirectly identified during the search for cross-modal retrieval and generation methods. For example, we found the work Text2Mesh (Michel et al., 2022), using the keyword "text-to-shape generation" because that keyword appears in the literature review section of the article, but the work should belong to manipulation methods after carefully reading its content. However, this might leave room for a more comprehensive review of the cross-modal manipulation methods by developing a different search strategy in the future.

the *source scope*): ScienceDirect, Web of Science, Scopus, IEEEXplore, ACM Digital Libraries, and Google Scholar within the time range of January 2013 to June 2022 (i.e., the *time scope*: the studies published in the past 10 years). The reason for choosing that time range is that many significant improvements in deep learning methods occurred after 2013, for example, variational autoencoders (VAEs, 2013) (Kingma and Welling, 2013) and generative adversarial networks (GANs, 2014) (Goodfellow et al., 2014). Since then, they have been widely applied in various applications, including the cross-modal tasks reviewed in this paper.

The initial search yielded 1,341 seed articles, including duplicates, of which the majority (i.e., 1,304 papers) is related to two categories: sketch-based 3D shape retrieval and generation, with only 37 articles for the other four categories (i.e., text-to-sketch retrieval: 0; text-to-sketch generation: 3; text-to-3D shape retrieval: 10; and text-to-3D shape generation: 24) (see details in Table A.1 in Appendix A). To make the review manageable, for the two categories of sketch-to-3D works, we decided to identify the most influential studies from those 1,304 papers using Connected Papers [5]. We found that (Wang et al., 2015) and (Lun et al., 2017) are pioneering work for deep learning-based sketch-to-3D shape retrieval and generation, respectively (Li et al., 2022b). Therefore, they were used as the origin papers to find their most relevant work via Connected Papers (see Figure A.1 in Appendix A for the two generated graphs). The search by Connected Papers identified 21 articles including (Wang et al., 2015) and (Lun et al., 2017) that meet our *content scope*.

Another finding was that the publication year of the articles in the two literature graphs turned out to be up to 2020, which could indicate that relevant articles published after 2020 have not gained enough attention to be considered influential by Connected Papers. The finding motivated us to further find the most recent studies for these two categories, so we decided to search relevant articles within the time

[5]Access link: `https://www.connectedpapers.com/`. Connected Papers allow readers to enter an origin paper and can generate a graph of papers with the strongest connections to the origin paper by analyzing about 50,000 research papers.

range from January 2021 to June 2022 in Google Scholar only, because we found that Google Scholar is more inclusive compared to other databases (i.e., the results from other databases turn out to be a subset of the results obtained from Google Scholar. See the comparison in Table A.1 in Appendix A). 138 articles were found in this search process. In total, 196 papers were found to merit close examination and review.

We then reviewed the titles and abstracts of all these articles to judge their relevance to our *content scope*. We excluded 12 preprints, one Master thesis, and one Ph.D. dissertation from those 196 papers because the preprints are not peer-reviewed or officially published. Finally, 50 articles were considered the most relevant and therefore closely reviewed.

3.4 Summary Statistics of The Literature

We summarized all 50 articles in terms of the following variables: method type, publication year, representation of design modalities, training dataset(s), object class of the training data, generalizability, user interface, user study, and publication source in Table A.2 of Appendix A which provides a complete list of these articles and the corresponding values for each of these variables. We report the statistics of four variables here, including the type of DLCMT, user interface, user study, and publication source, as an example, and introduce the others in detail in Section 3.5.

We did not find any work related to text-to-sketch retrieval, possibly due to the lack of interest in practical applications. We obtained 2 articles for text-to-3D shape retrieval, 6 articles for text-to-3D shape generation, 4 articles for text-to-sketch generation, 19 articles for sketch-to-3D shape retrieval, 18 articles for sketch-to-3D generation, and 5 articles for cross-modal design manipulation. Among these works, (Chen et al., 2018) can work for text-to-3D shape retrieval and generation; (Liu et al., 2022) can perform text-to-3D shape generation and manipulation; (Jin et al., 2020; Guillard et al., 2021) are shown to be capable of sketch-to-3D shape generation and manipulation.

Only 15 peer-reviewed publications are relevant to text-to-3D shape retrieval, text-to-3D shape generation, text-to-sketch generation, and cross-modal design manipulation, but we observe a recent surging interest in these topics especially text-related ones, possibly due to advances in natural language processing (e.g., Contrastive Language-Image Pre-Training (CLIP) (Radford et al., 2021)) since our preliminary literature review (Li et al., 2022b).

There are 13 studies (Huang and Canny, 2019; Huang et al., 2020a; Li et al., 2021a; Qin et al., 2022; Guillard et al., 2021; Li et al., 2018; Delanoy et al., 2018; Han et al., 2017; Du et al., 2021; Luo et al., 2021; Wang et al., 2022a; Giunchi et al., 2021; Stemasov et al., 2022) that provide user interfaces. The user interface application serves as a way to show the effectiveness of the proposed deep learning approach, which can also better facilitate human-AI interaction for creative designs. Especially, (Giunchi et al., 2021; Stemasov et al., 2022) provide user interfaces in virtual reality (VR) and augmented reality (AR) settings, respectively, which can further improve the user experience of human-computer interaction in immersive design. Additionally, 12 studies (Yuan et al., 2021; Huang et al., 2020a; Huang and Canny, 2019; Giunchi et al., 2021; Du et al., 2021; Luo et al., 2021; Li et al., 2018; Delanoy et al., 2018; Lun et al., 2017; Han et al., 2017; Wang et al., 2022a; Michel et al., 2022) conducted user studies to further validate their methods and user applications. User studies can serve as a way to hear from human users so that researchers can improve the proposed methods from users' feedback. It can also help study human-computer interaction in a real situation.

The articles reviewed are from conference proceedings (32) and journals (18). Most DLCMT methods come from the domains of computer science and computer engineering with only two papers (Li et al., 2022c; Qin et al., 2022) from the engineering design community.

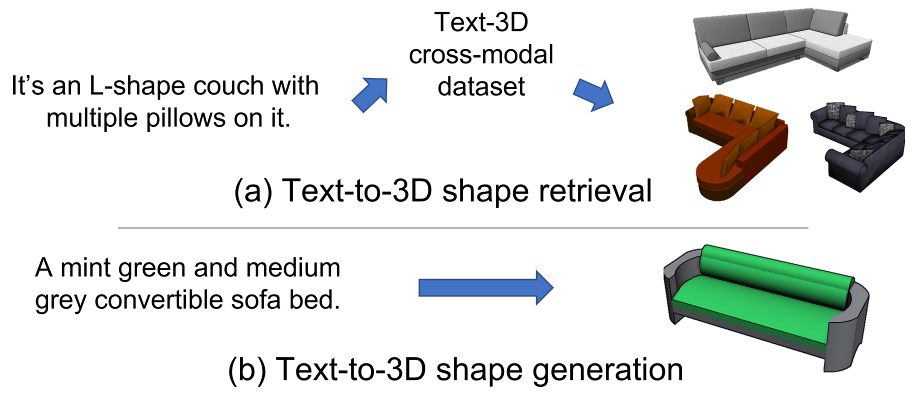

It's an L-shape couch with multiple pillows on it.
Text-3D cross-modal dataset
(a) Text-to-3D shape retrieval
A mint green and medium grey convertible sofa bed.
(b) Text-to-3D shape generation

Figure 3.6: Demonstration of *(a) Text-to-3D shape retrieval*: retrieving 3D shapes that best match the natural language descriptions (NLD) from a given dataset or repository; and (b) *text-to-3D shape generation*: automatically generating a 3D shape that matches the NLD. The examples of NLD and images are obtained from ShapeNet (Chang et al., 2015).

3.5 Review and Discussion

In this section, we summarize our review of the papers in each of the cross-modal task categories and discuss their technical details, from which we draw insights into the challenges and opportunities of applying such methods in the engineering design field and discuss potential solutions to the challenges.

3.5.1 RQ 1-(1): What DLCMT methods can be used in Design Search of conceptual design?

3.5.1.1 Text-to-3D Shape Retrieval

The history of text-to-3D shape retrieval methods can be traced back to Min et al. 2004 (Min et al., 2004), who used pure text information (query text and description associated with 3D shapes) for the 3D retrieval task, which is essentially a text-text matching.

For state-of-the-art deep learning methods as we introduce below, it is a common strategy to learn a cross-modal representation for text and 3D shapes using cross-modal representation learning techniques (see (Liu et al., 2020b)) for more information. Figure 3.6 (a) demonstrates the process of a text-to-3D retrieval task. As a pioneering and representative work for this task, Chen et al. (Chen et al., 2018) first constructed a joint embedding of text and 3D shapes using an encoder composed of a convolution neural network (CNN) and a recurrent neural network (RNN) on text data and a 3D-CNN encoder on 3D voxel shapes. A triplet loss was applied and learning-by-association (Haeusser et al., 2017) was used to align the embedded representations of text and 3D shapes. They also introduced a 3D-text cross-modal dataset including two sub-datasets: 1) ShapeNet (Chang et al., 2015) (chairs and tables only) with a natural language description and 2) geometric primitives with synthetic text descriptions. However, the computational cost caused by the cubic complexity of 3D voxels limits this method to the machine learning of low-resolution voxels. Consequently, the learned joint representations will have low discriminative ability. Han et al. (Han et al., 2019) built a $Y^2Seq2Seq$ network architecture using a Gated Recurrent Unit (GRU, one variation of RNN) to encode features of multiple-view images to represent the shape. To obtain the joint embedding of text and sketches, they trained the network using both intermodality and intramodality reconstruction losses, in addition to the triplet loss and classification loss. Therefore, the proposed network could learn more discriminative representations than (Chen et al., 2018).

3.5.1.2 Sketch-to-3D shape Retrieval

Sketch-to-3D shape retrieval has been extensively studied using non-deep learning methods (Li et al., 2014a). These methods usually consist of three steps: 1) automatically select multiple views from a given 3D shape in the hope that one of them is similar to the input sketch(es); 2) project the 3D shape into 2D space from the selected viewpoints; 3) match the sketch against the 2D projections based on predefined features. However, the selection of best viewpoints, as well as the design of predefined matching features, could be subjective and random, which motivates the development of deep learning-based methods that can avoid the subjective selection of views and learn features from the data of sketches and 3D shapes (Wang et al.,

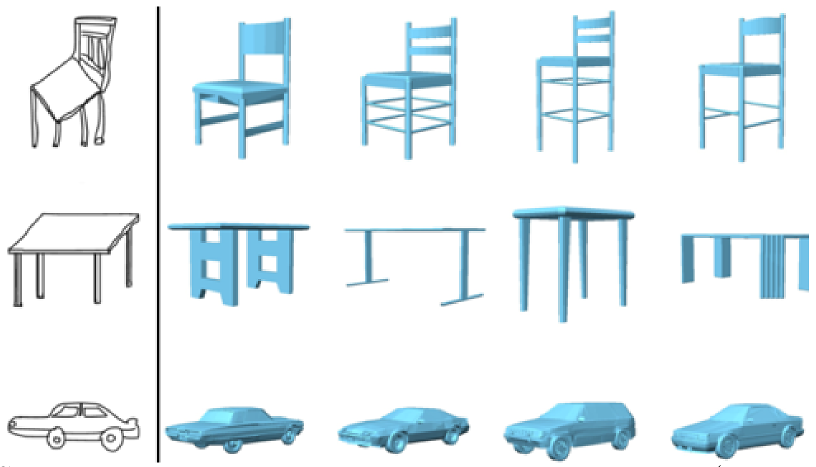


Figure 3.7: *Sketch-to-3D shape retrieval* method by Wang et al. (Wang et al., 2015). For each row, the 2D drawing is the query sketch and the 3D models are the retrieved 3D shapes from an existing dataset, Princeton Shape Benchmark (PSB) (Shilane et al., 2004). The figure is used with permission.

2015). In light of the scope of this review, we focus on deep learning methods for sketch-to-3D shape retrieval.

Wang et al. (Wang et al., 2015) initialized the effort and proposed to learn feature representations for sketch-to-3D shape retrieval as shown in Figure 3.7, which avoided computing multiple views of a 3D model. They applied two Siamese CNNs (Chopra et al., 2005) for views of 3D shapes and sketches, respectively, and a loss function defined on the within-domain and cross-domain similarities. To reduce the discrepancies between the sketch features and the 3D shape features, Zhu et al. (Zhu et al., 2016) built a pyramid cross-domain neural network of sketches and 3D shapes. They used the network to establish a many-to-one relationship between the sketch features and a 3D shape feature. Dai et al. (Dai et al., 2018, 2017) proposed a novel deep correlated holistic metric learning method with two distinct neural networks for sketch and 3D shape. Such a deep learning method mapped features from both domains into one feature space. In the construction of its loss function, both discriminative loss and correlation loss was used to increase the discrimination of features within each domain and the correlation between domains. Chen et al. (Chen and

Fang, 2018) developed a GAN-based deep adaptation model to transform sketch features into 3D shape features, of which correlations can be enhanced by minimizing the mean discrepancy between modes. Xia et al. (Xia et al., 2021) proposed a novel semantic similarity metric learning method based on a “teacher-student” strategy by using a teacher network to guide the training of the student network. The teacher network was trained to extract the semantic features of the 3D shapes. The student network was then trained by using the pre-learned 3D shape features to learn the sketch features. Similarly, Yang et al. (Yang et al., 2022) applied a sequential learning strategy to learn 3D shape features without 2D sketches first and then used the learned features of 3D shapes to guide the learning of sketch features. During the query process, they further integrated clustering algorithms to categorize subclasses in a shape class to improve retrieval accuracy. In the methods mentioned above, deep metric learning (Kaya and Bilge, 2019) was applied to mitigate the modality discrepancy between the sketch and the 3D shape.

There are also methods that study how to represent 3D shapes more comprehensively so that 3D shapes can better correspond to sketches. Xie et al. (Xie et al., 2017) proposed a method to learn a Wasserstein barycenter of CNN features extracted from 2D projections of a 3D shape. They constructed the metric network to map sketches and the Wasserstein barycenters of 3D shapes to a common deep feature space. Then a discriminative loss was formulated to learn the deep features. The deep features learned could then be used for the sketch-to-3D shape retrieval. Chen et al. (Chen et al., 2019) proposed a novel stochastic sampling method to randomly sample rendering views of the sphere around a 3D shape and incorporated an attention network (see (Niu et al., 2021) for a comprehensive review) to exploit the importance of different views. They also developed a novel binary coding strategy to address the time-efficiency issue of sketch-to-3D shape retrieval.

Another direction to reduce the large cross-modality difference between 2D sketches and 3D shapes is to deal with noise in the sketch data. Liang et al. (Liang et al., 2021) pioneered this direction by developing a method called noise-resistant sketch feature learning with uncertainty, which achieved the new state-of-the-art for sketch-based 3D shape retrieval. Liu et al. (Liu and Zhao, 2021) proposed a Guidance Cleaning Network to remove low-quality sketches that have much noise, which is like a data cleaning process. The authors showed superior results over state-of-the-art methods because the learning of noisy data was suppressed.

All the methods introduced above achieve state-of-the-art results on commonly used sketch-to-3D retrieval datasets, such as Princeton Shape Benchmark (PSB) (Shilane et al., 2004), SHREC13 (Li et al., 2013), and SHREC14 (Li et al., 2014b). The multiview CNN (MVCNN) (Su et al., 2015) has been widely used in all these methods to generate features from projection images of 3D shapes. Different from these methods aiming to retrieve objects by coarse category-level retrieval of 3D shapes given an input sketch, Qi et al. (Qi et al., 2021) introduced a novel task of fine-grained instance-level sketch-to-3D shape retrieval, with the aim of retrieving one specific 3D shape that best matches the input sketch. They created a set of paired sketch-to-3D shape data of chairs and lamps from ShapeNet (Chang et al., 2015). Then, they built a deep joint embedding learning-based model with a novel cross-modal view attention module to learn the features of sketches and 3D shapes. As the first effort to find local image correspondences between design sketches, Navarro et al. (Navarro et al., 2021) proposed a synthetic line drawing dataset rendered from 3D shapes from ShapeNet (Chang et al., 2015). The authors obtained a learned descriptor, namely Sketch-Zoom descriptor, for dense registration in line drawings and showed its promising application in sketch-3D shape retrieval by identifying local correspondences between sketches.

There is also interest in using CAD data in 3D shape retrieval. Qin et al. (Qin et al., 2022) developed a sketch-to-3D CAD shape retrieval approach using the variational autoencoder (VAE) and structural semantics. They created their training dataset by collecting 3D CAD models from local companies and obtained their six-view projections as sketch data. Manda et al. (Manda et al., 2021) developed a new sketch-3D CAD model dataset, CADSketchNet, from the Engineering Shape

Benchmark (ESB) (Jayanti et al., 2006) and Mechanical Components Benchmark (MCB) (Kim et al., 2020) datasets. The authors also analyzed various deep learning-based sketch-to-3D retrieval approaches using the proposed dataset and reported the comparison results.

Efforts have also been made to bridge the semantic gap between sketches and 3D shapes to improve sketch-based 3D shape retrieval. Ye et al. (Ye et al., 2016) presented a CNN-based 3D sketch-based shape retrieval (CNN-SBR) architecture based on 3D sketch (Type II) data obtained from SketchANet (Yang and Hospedales, 2015). Using data augmentation to prevent overfitting, they achieved a significant improvement compared to other learning-based methods. Building on previous work (Ye et al., 2016; Li et al., 2016), Li et al. (Li et al., 2021a) proposed a novel interactive application supported by CNN-SBR. The method used Microsoft Kinect, which can track the 3D locations of 20 joints of a human body, to track the 3D locations of a user's hand to create a 3D sketch. The proposed method was tested on a proposed dataset and achieved state-of-the-art performance in 3D sketch-based 3D shape retrieval.

The idea of utilizing a 3D sketch (Type II) as query input has been further applied to virtual reality (VR) and augmented reality (AR) settings to facilitate the immersive design. Building on the method proposed in (Giunchi et al., 2018), Giunchi et al. (Giunchi et al., 2021) designed a multimodal interface for 3D model retrieval in VR with both sketch and voice input. The authors implemented a consistent translation method between queries of 3D sketch and voice, allowing their integration during a single search session. Similarly, ShapeFindAR (Stemasov et al., 2022) combined both 3D sketch and textual input to enable in-situ spatial search of a 3D model repository in an AR setting. The server was built using a REST (representation state transfer) application programming interface provided by Flask, a web framework for the Python programming language.

3.5.2 RQ 1-(2): What DLCMT methods can be used in Design Creation of conceptual design?

3.5.2.1 Text-to-3D Shape Generation

The task of text-to-3D shape generation is illustrated by Figure 3.6 (b). To accomplish this task, Jahan et al. (Jahan et al., 2021) proposed a semantic label-guided shape generation approach, which can take one-hot semantic keywords as input and generate 3D voxel shapes without color and texture. The proposed method was trained using chairs, tables, and lamps obtained from the Co-Segmentation (COSEG) dataset (Wang et al., 2012) and ModelNet (Wu et al., 2015b). Based on their work on text-to-3D shape retrieval task using a joint embedding of text and 3D shape, Chen et al. (Chen et al., 2018) further combined the joint embedding model with a conditional Wasserstein GAN (WGAN) framework (Arjovsky et al., 2017), which enables the generation of colored voxel shapes in low resolution. To improve the surface quality of the generated 3D shapes, several studies have been conducted using the proposed 3D-text cross-modal dataset by Chen et al. (Chen et al., 2018). Li et al. (Li et al., 2020a) proposed to use class labels to guide the generation of 3D voxel shapes with the assumption that shapes with different labels (e.g., chairs and tables) have different characteristics. They added an independent classifier to the WGAN framework (Arjovsky et al., 2017) to guide the training process. The classifier could be trained together with the generator to enable more distinctive class features in the generated 3D shapes. To further improve the quality of 3D shapes generated with color and shape, Liu et al. (Liu et al., 2022) leveraged implicit occupancy (Mescheder et al., 2019) as the 3D representation and proposed a word-level spatial transformer (Vaswani et al., 2017) to correlate shape features with semantic features of text by decoupling shape and color predictions for learning features in both texts and shapes.

The methods introduced above only support the generation of 3D shapes in individual categories (e.g., the chair category or the table category). The generalizability (the ability to generalization) of these methods remains challenging due to the unavailability and limited size of the paired data of 3D shapes and text de-

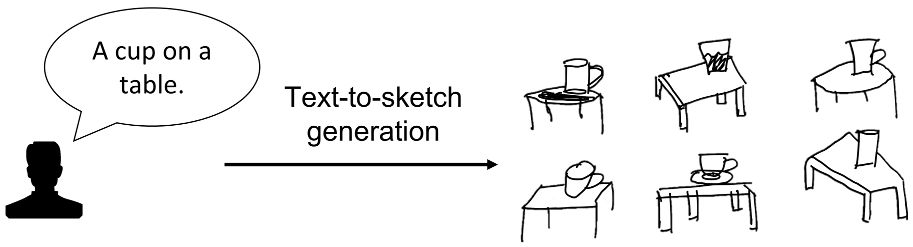


Figure 3.8: Demonstration of *text-to-sketch generation*, which can generate sketches that correspond to users' natural language descriptions (NLD).

scriptions. To improve generalizability, some researchers have tried to utilize some pre-trained models (e.g., Contrastive Language-Image Pre-Training (CLIP) (Radford et al., 2021)) and zero-shot learning techniques (Xian et al., 2018). Sanghi et al. (Sanghi et al., 2022) proposed a method called CLIP-forge, which could generate 3D voxel shapes from text descriptions for ShapeNet (Chang et al., 2015) objects. It required training data (i.e., rendered images, voxel shapes, query points, and occupancy) obtained from 3D shapes without text labels. They first learned an encoding vector of a 3D geometry and then a normalizing flow model (Dinh et al., 2016) of that encoding vector conditioned on a CLIP (Radford et al., 2021) feature embedding.

CLIP-Forge has good generalizability to ShapeNet (Chang et al., 2015) categories. To further improve the generalizability to classes outside common 3D shape datasets (e.g., ShapeNet (Chang et al., 2015) and ModelNet (Wu et al., 2015b)), Jain et al. (Jain et al., 2022) combined Neural Radiance Field (NeRF) (Mildenhall et al., 2020) with an image-text loss from CLIP (Radford et al., 2021) to form Dream Fields. A Dream Field is a neural 3D representation that can return a rendered 2D image given the desired viewpoint. After training, the method could generate colored 3D neural geometry from text prompts without using 3D shape data, resulting in better generalizability.

3.5.2.2 Text-to-Sketch Generation

Sketches can inspire design ideas (Krish, 2011; Pratt et al., 2005; Menezes and Lawson, 2006), and text-to-sketch tools could help designers efficiently capture fleeting design inspirations. The generation of images from text descriptions (i.e., text-to-image synthesis/generation) has seen great progress recently (Frolov et al., 2021). Unlike text-to-image generation, text-to-sketch synthesis is more challenging and can only rely on rigid edge/stroke information without color features (i.e., pixel values) in an image (Yuan et al., 2021).

Text2Sketch (Wang et al., 2018b) applied a Stagewise-GAN (i.e., generative adversarial network) to encode human face attributes identified from text descriptions and transforms those attributes into sketches, which were trained on a manually annotated dataset of text-face sketches. Although the method was applied in face recognition instead of product design, it is worth being introduced here because the method is inspiring and could be applied to the design domain if a different dataset is used. Yuan et al. (Yuan et al., 2021) constructed a bird sketch dataset by modifying the Caltech-UCSD Birds (CUB) dataset (Wah et al., 2011), based on which they trained a novel GAN-based model, called T2SGAN. The model featured a Conditional Layer-Instance Normalization module that could fuse the image features and sentence vectors, thus efficiently guiding the generation of sketches.

The methods mentioned above were developed for single-object sketch synthesis, and there are also methods for multi-object generation, which could be useful for generating designs part by part. An example of such methods is shown in Figure 3.8. Huang et al. (Huang and Canny, 2019) developed Sketchforme by adopting a two-step neural network: 1) a transformer-based mixture density network for the scene composer to generate high-level layouts of sketches, and 2) a sketch-RNN (Ha and Eck, 2018) based object sketcher to generate individual object sketches. The scene composer and the object sketcher were trained using the Visual Genome dataset (Krishna et al., 2017) and the “Quick, Draw!” dataset (Jongejan et al., 2016), respectively. Since different datasets of text and sketches can be used, this method helped avoid the requirement for paired data of text description and sketches of an object. Based on (Huang and Canny, 2019), Huang et al. (Huang et al., 2020a) took a further step and proposed an interactive sketch generation system called Scones. It used a

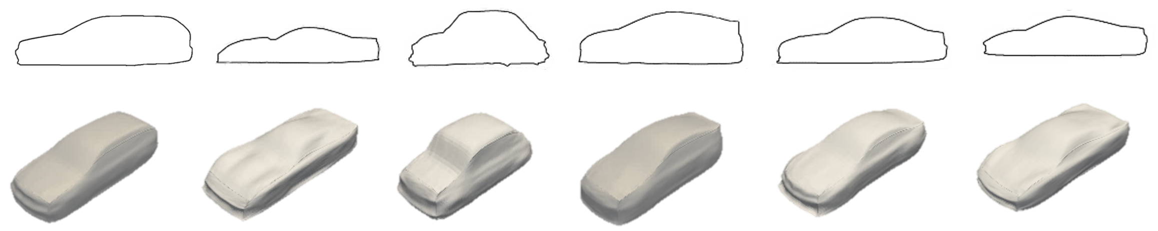


Figure 3.9: *Sketch-to-3D shape generation* method by Li et al. (Li et al., 2022c). The first row shows the input 2D silhouette sketches, and the corresponding predicted 3D mesh shapes are shown in the second row.

Composition Proposer to propose a scene-level composition layout of objects and an Object Generator to generate individual object sketches.

3.5.2.3 Sketch-to-3D Shape Generation

There are mainly two paradigms for 3D shape reconstruction from 2D sketches: the geometric-based method and the learning-based method. Sketch-based interfaces for modeling are a major branch of geometric-based methods (Olsen et al., 2009) and we do not review this line of work in light of the scope of review. We also excluded some methods that apply deep learning techniques, but require predefined geometric models to guide 3D reconstruction, such as the methods presented in (Nishida et al., 2016; Han et al., 2017). We focus on reviewing deep learning-based methods without using predefined geometric models that require the design of rules.

Deep learning-based sketch-to-3D shape generation without any predefined geometric models was initialized by Lun et al. (Lun et al., 2017). They proposed an encoder-multiview-decoder architecture that can extract multiview depth and normal maps from a single sketch or multiple sketches and output a 3D shape in point clouds. The resulting 3D point cloud shape can be converted to a 3D mesh shape for better visualization. 2.5D visual surface geometry (e.g., depth and normal maps) is a representation that can make a 2D image appear to have 3D qualities (He et al., 2021b; Su et al., 2018). Similarly to (Lun et al., 2017), many works use the strategy of predicting 2.5D information first to guide the generation of 3D shapes. Nozawa et al. (Nozawa et al., 2020) extracted depth and mask information from a single input sketch by an encoder-decoder network. Then, a lazy learning (Aha, 2013) method was performed to find similar samples in the dataset to synthesize a 3D shape represented by point clouds. Later, Nozawa et al. (Nozawa et al., 2022) extended (Nozawa et al., 2020) by changing the architecture with a combination of GAN and lazy learning.

To improve the surface quality of the shapes resulting from their previous work (Delanoy et al., 2018), Delanoy et al. (Delanoy et al., 2019) proposed to first predict one normal map per input 3D sketch (Type I). Then they fused all normal maps predicted from multiview sketches to the predicted 3D voxel shape to optimize the resulting surface mesh. Li et al. (Li et al., 2018) introduced an intermediate CNN layer to model the direction of dense curvature and used an additional output confidence map along with the depth and normal maps extracted using CNNs to generate high-quality 3D mesh shapes. They also provided a user-interaction system for 3D shape design. Similar to the idea of obtaining an intermediate 2.5D representation, Yang et al. (Yang et al., 2021) proposed a skeleton-aware modeling network to generate 3D human body models using skeletons as the intermediate representation. The network can first interpret sparse joints from input sketches and then predict the Skinned Multi-Person Linear model (Pavlakos et al., 2019) parameters based on joint-wise features. Although this work focuses on the generation of human bodies, the proposed network can inspire design researchers to consider predicting important feature points to guide the generation of 3D shapes. Li et al. (Li et al., 2022c) proposed a predictive and generative target-embedding variational autoencoder and demonstrated its effectiveness by solving a sketch-to-3D shape generation problem. The authors used a 3D extrusion shape obtained by extruding a 2D silhouette sketch as an intermediate representation, which transferred the problem to a 3D-3D prediction problem. The approach can predict a high-quality 3D mesh shape from a silhouette sketch without inner contour lines, as shown in Figure 3.9. In addition to the prediction function, the proposed approach can also generate numerous novel 3D mesh shapes using its generative function.

The efforts of providing an easy-to-use sketching system can be beneficial to novice users for customized design. Delanoy et al. (Delanoy et al., 2018) proposed an interactive sketch-to-3D generations system. They used a CNN to transform 3D sketches (Type I) to 3D voxel shapes, and another CNN as an updater to update the predicted 3D shape while users are providing more sketches. The voxel shapes can then be transferred to 3D mesh shapes. However, the output 3D shapes are low-quality due to the high memory consumption of the voxel representation. To improve the surface quality of the resulting 3D shapes, mesh and implicit field have been applied by some interaction systems. For example, Han et al. (Han et al., 2017) proposed a novel sketching system to generate 3D mesh human faces and caricatures using a CNN-based deep regression network. The method was trained on a newly proposed dataset extended from FaceWarehouse (Cao et al., 2013). Du et al. (Du et al., 2021) designed a novel sketching system composed of a part generator and an automatic assembler to generate part-aware man-made objects with complex structures. They used implicit occupancy (Mescheder et al., 2019) as the 3D representation which can be transferred to a 3D mesh shape with detailed geometry. Similarly, Wang et al. (Wang et al., 2021) introduced a novel sketch-to-3D shape method that can segment a given sketch and build a transformation template that is then used to generate multifarious sketches. These sketches are then taken as input to an encoder-multiview-decoder network similar to (Lun et al., 2017) to generate a 3D point cloud shape. Luo et al. (Luo et al., 2021) proposed a coarse-to-fine-grained 3D mesh modeling system using 3D sketches as input for animalmorphic head design. A coarse mesh can be first generated by the input 3D sketch. Then, a novel pixel-aligned implicit learning approach is used to guide the deformation of the coarse mesh to produce a more detailed mesh. Guillard et al. (Guillard et al., 2021) introduced an interactive system to reconstruct and edit 3D shapes using implicit field representation, DeepSDF (Park et al., 2019) format, from 2D sketches using an encoder-decoder architecture, which can output mesh shapes.

The aforementioned methods are usually trained using one individual category of objects and can only deal with 3D shape generation from sketches within that specific category. To improve the generalizability of the method, Jin et al. (Jin et al., 2020) proposed a novel network consisting of a VAE (i.e., variational autoencoder) and a volumetric autoencoder to learn the joint embedding of sketches and 3D shapes using various classes of objects. The trained network has good generalizability and can be used to predict 3D voxel shapes based on 2D occluding contours. Zhang et al. (Zhang et al., 2021) are the first to generate a 3D mesh shape from a single free-hand sketch. They proposed a view-aware network based on GAN to explicitly condition the process of generating 3D mesh shapes on viewpoints. The method can improve generation quality and bring controllability to output shapes by explicitly adjusting viewpoints, which can be well generalized to out-of-distribution data.

The methods introduced above have to be trained using supervised learning, which means that the training data must be pairs of sketches and 3D shapes (i.e., labeled data). Wang et al. (Wang et al., 2018a) proposed an unsupervised learning method for sketch-to-3D shape reconstruction. They embedded unpaired sketches and rendered images from 3D shapes to a common latent space by training an adaption network via autoencoder with adversarial loss. During the inference of 3D shapes from sketches, they retrieved several nearest-neighboring 3D shapes from the training dataset as prior knowledge for a 3D GAN to generate new 3D shapes that best match the input sketch. This method can only output very coarse 3D voxel shapes but provides an interesting idea based on unsupervised learning for sketch-to-3D shape generation.

In addition to the usage of popular 3D shape representations (e.g., point clouds, voxels, meshes, and implicit representation) in sketch-to-3D shape generation, new 3D representations are gaining more and more attention in this field. For example, Smirnov et al. (Smirnov et al., 2020, 2019) proposed a novel deformable parametric template composed of Coon patches that can naturally fit into a conventional CAD modeling pipeline. The resulting 3D shapes can be easily converted to NURBS representation, allowing edits in CAD software.

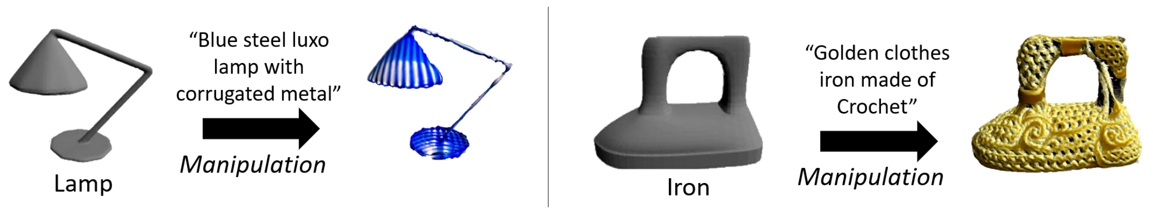

Lamp
"Blue steel luxo lamp with corrugated metal"
Manipulation
Iron
"Golden clothes iron made of Crochet"
Manipulation

Figure 3.10: *Text-to-3D shape manipulation* method, Text2Mesh by Michel et al. (Michel et al., 2022). The method can manipulate an existing mesh shape by adding color, texture, and geometric details driven by a target natural language description. The figure is used with permission.

3.5.3 RQ 1-(3): What DLCMT methods can be used in Design Integration of conceptual design?

In this section, we introduce some works relevant to text-to-3D shape and sketch-to-3D shape integration methods. These methods allow designers to further edit and manipulate 3D designs by changing text prompts or sketches.

The sketch-to-3D shape generation method introduced by Jin et al. (Jin et al., 2020) could be further used to manipulate a given 3D voxel shape to target input sketches with the learned joint embedding space. However, it focuses on manipulating the outline of a given 3D shape. To enable manipulation of color and shape, CLIP-NeRF (Wang et al., 2022a) was proposed based on CLIP (Radford et al., 2021), which has a disentangled conditional NeRF (Mildenhall et al., 2020) architecture by introducing a shape code to deform the 3D volumetric field and an appearance code to control the colors. The method can edit a given colored 3D voxel shape to meet the target semantic description of color and shape. The text-to-3D generation method (Liu et al., 2022) can also allow intuitive manipulation of the color and shape of a generated 3D mesh shape simply by changing the input semantic keywords of color or shape.

To enable detailed edits or manipulation of geometries, in some works a differentiable renderer has been applied. Sketch2Mesh (Guillard et al., 2021) introduced in Section 3.5.2.3 can also perform shape editing due to the integrated differentiable renderer. Using the representation power of CLIP (Radford et al., 2021), Michel et al. (Michel et al., 2022) proposed Text2Mesh (see Figure 3.10) to manipulate a given 3D mesh shape by predicting color and local geometric details that conform to the description of the target text.

There have been a series of DLCMT methods that can be applied to product shape design in different design steps of conceptual design. As a summary of the review, DLCMT methods indeed provide opportunities to address the two major challenges as discussed in Section 3.2 because they can (1) take various design modalities as input and provide methods catering to Design Search, Design Creation, and Design Integration, and (2) improve design creativity by actively involving human input (Huang and Canny, 2019; Huang et al., 2020a; Du et al., 2021; Luo et al., 2021). Taking advantage of these opportunities and implementing the appropriate DLCMT methods in conceptual design can therefore accelerate the search and iteration of design concepts (e.g. (Chen et al., 2018; Wang et al., 2015; Giunchi et al., 2021)) and the modification of designs (e.g., (Michel et al., 2022; Jin et al., 2020; Sanghi et al., 2022; Han et al., 2017)). We also observe that DLCMT methods could be particularly useful in design applications, such as design democratization, design education, and immersive design (e.g., (Giunchi et al., 2021; Stemasov et al., 2022; Ye et al., 2016; Wang et al., 2015; Chen et al., 2018)).

3.5.4 RQ 2: What are the Challenges in Applying DLCMT to Conceptual Design and How Can They be Addressed?

Examination of the literature has helped us identify several challenges in applying DLCMT methods to conceptual design. DLCMT has been focusing on shape synthesis, which can be applied in product shape design, as discussed above. However, Regenwetter et al. (Regenwetter et al., 2022) state that 3D synthesis work is only tangential to engineering design because they focus more on visual appearance, rather than functional performance or manufacturability. Although we partially agree with (Regenwetter et al., 2022) that the overlap between shape synthesis and engineering design is insignificant in light of the importance of shape design, we must admit that product shape is not the only focus in conceptual design. Other factors, such as engineering performance, system design features, and manufacturability, should also be considered and can be incorporated into the data-driven design cycle even in the early stages of the design.

In this section, we discuss in detail the challenges of applying DLCMT methods to engineering design from four aspects, including the lack of cross-modal datasets that incorporate engineering performance and manufacturability, complex systems design using DLCMT, 3D representations in DLCMT, and the generalizability of DLCMT methods.

3.5.4.1 The Lack of Cross-Modal Datasets that Incorporate Engineering Performance and Manufacturability

Data is the fuel for deep learning-based design methods. Data sparsity is a challenging issue for data-driven design methods, and there is generally a deficiency of big practical data (Regenwetter et al., 2022), regardless of the data modality, to train useful and meaningful models for engineered products. Unlike the computer science community, where numerous open source unimodal or cross-modal datasets, such as (Chang et al., 2015; Wu et al., 2015b; Li et al., 2013; Chen et al., 2018), are available to researchers to compare their methods with state-of-the-art methods. For example, 16 articles (e.g., (Chen et al., 2018; Qi et al., 2021; Navarro et al., 2021; Guillard et al., 2021; Zhang et al., 2021)) use ShapeNet (Chang et al., 2015) as the training data of their methods. There is a lack of similar benchmark datasets in the engineering design field. Even if those datasets from computer science can also be beneficial to the engineering design community, they mainly focus on the shape of objects and have little emphasis on downstream engineering-related information. Using text-to-3D shape methods as an example, a user could say "I want an SUV with low fuel consumption". An SUV car shape could be easily generated, but we would not know whether the drag coefficients of the generated designs meet the requirement or not. We might ask the following question: How could a computer understand that

NL description and translate it into a primitive SUV car shape taking into account the drag performance? Therefore, finding answers to this question could be an interesting research direction.

Similarly, it is also worth exploring how other downstream engineering requirements and constraints (e.g., manufacturability) can be counted when applying DLCMT to engineering design. We have not found any DLCMT methods that take into account engineering performance and manufacturability. One challenge here is the lack of such datasets. The difficulties primarily rest in the cost (either monetary or time) of running high-fidelity computational or physical experiments. Moreover, certain experimental data could be confidential for commercial or military purposes. The availability of large cross-modal datasets with engineering performance and manufacturability information could greatly ease the verification and validation of existing methods for DLCMT and promote the development of new DLCMT methods for the design of engineered products.

3.5.4.2 Complex Systems Design Using DLCMT

A few DLCMT studies ((Huang and Canny, 2019; Huang et al., 2020a; Du et al., 2021)) aim to generate designs part by part considering the structural relationship among components, which can be potentially applied to the design of systems. But this leaves a large space for engineering design researchers to investigate in the future. The challenges of addressing systems design using DLCMT mainly stem from the structural complexity of an engineered product, such as dependencies, constraints, and the relationship between components.

An engineered product is usually a system consisting of interconnected parts with complex dependencies. To take into account parts' dependency information, there are generally two ways to support the conceptual design of a product at the system level when applying DLCMT methods. In the first method, each component of the product is generated separately using DLCMT, and then the components are assembled either automatically using rules-based computer algorithms or manually (Huang and Canny, 2019; Huang et al., 2020a). The second method is often referred to as part-aware generative design (Li et al., 2021c; Gao et al., 2019c; Mo et al., 2019b). The objective of using DLCMT methods for part-aware design is to learn the structural relationships and dependencies between parts directly from the training data so that parts generation and assembly can be automatically completed.

Compared to the first method, the second method can save time and the cost of additional assembly steps. Those steps are often non-trivial, especially when one wants to computerize the assembly process in CAD software. In addition, part-aware generative design methods better capture the geometric details of 3D shapes (Gao et al., 2019c; Mo et al., 2019b). For example, in the transition regions between two components (e.g., the connection regions between the side rear mirrors and the car body). These geometric details may significantly influence the engineering performance (for example, aerodynamic drag) of a design.

As mentioned above, there are a few studies, i.e., text-to-sketch generation (Huang and Canny, 2019; Huang et al., 2020a) and sketch-to-3D (Du et al., 2021) methods for DLCMT attempting to integrate the concept of part-aware design, but most methods treat the design object as a single monolithic part without a systems design perspective. Considering engineering applications, treating a design as a whole piece could limit the transition of the generated design shapes to later design stages, since components are usually manufactured separately. Attention has been paid to by the engineering design community (Chen and Fuge, 2019; Li et al., 2021c) for part-aware design. However, how to enable part-aware design in DLCMT remains underexplored and is an important research direction.

3.5.4.3 3D Representations in DLCMT

Designs can be factored using different representations for storage, computation, and presentation. For example, 3D representation matters both visual quality

Table 3.2: Comparison of pros and cons of the three representations to deep learning methods

3D Representation	Pros	Cons
Voxels	• The data structure in fourth-order tensor makes it easy to be adapted in 3D convolution operations in deep learning methods • Can deal with 3D shapes with arbitrary topology	• Low visual quality • High computational cost because the number of the 3D representation parameters scale with the increase of spatial resolution in cubes • Cannot be directly used in engineering analyses (e.g., finite element analysis (FEA)) for performance evaluation
Point clouds	• Compatible with the output data format of common scanning software • Compact for data storage and management • Can deal with 3D shapes with arbitrary topology	• Low visual quality • No detailed geometric information about relationships between points making it hard to convert to meshes • Cannot be directly used in engineering analyses (e.g., FEA) for performance evaluation
Meshes	• High visual quality • Compact for data storage and management • Widely-accepted 3D representation in computer graphics • Compatible with downstream engineering software, such as the FEA and computational fluid dynamics (CFD) tools	• Discrete and disordered elements make it challenging to be processed by deep learning methods • Hard to deal with 3D shapes with arbitrary topology
Implicit representation	• High visual quality • Easy adaption to deep learning methods • Compact for data storage and management • Can deal with 3D shapes with arbitrary topology	• Need to use rendering techniques to extract the isosurface of the 3D shapes for visualization • Cannot be directly used in engineering analyses (e.g., FEA) for performance evaluation

and computational cost when implementing DLCMT, and the choice between them is often a difficult decision. Furthermore, in engineering design applications, the choice of 3D representation also influences the compatibility with downstream engineering analysis in CAD and CAE software. In what follows, we share our insight into the challenges associated with 3D representation in both aspects.

3D shapes with high visual quality and rich geometric details can help designers better understand a design concept. Voxels, point clouds, and meshes are the most commonly used representations for 3D geometry. Similar to the pixels of images, voxel grids are naturally adapted to the convolutional neural network (CNN) model, which is the major reason for its prevalence in 3D geometry learning research. The majority of the DLCMT methods (e.g., (Chen et al., 2018; Han et al., 2019; Arjovsky et al., 2017; Li et al., 2020a; Sanghi et al., 2022; Delanoy et al., 2018; Wang et al., 2018a)) uses voxels for 3D shape representation. Voxel shapes are usually needed to be converted to mesh shapes for better visualization. However, the transformed mesh shapes will look coarse if the resolution of the voxel shapes is low. This could negatively influence the subjective evaluation of the shape of a design concept, and the design concept might be overlooked by designers. An intuitive way to improve the resolution of the resulting 3D voxel shapes is to use high-resolution training data, but this may not be feasible due to the limited computing resources for training the neural network. Fukamizu et al. (Fukamizu et al., 2019) provided a two-stage strategy to synthesize high-resolution 3D voxel shapes from natural language, which could be an inspiring method for dealing with low-resolution issues. Point clouds (Qi et al., 2017; Lun et al., 2017; Nozawa et al., 2020, 2022) are more efficient in representing 3D objects, but do not cover geometric details. For example, it does not encode the relationship between points and the resulting topology of an object, leading to a challenging conversion to meshes. Using meshes (Li et al., 2018; Han et al., 2017; Zhang et al., 2021; Yang, 2003) for 3D representation could generally alleviate the low visual quality and data storage problems, but, in the meantime, it is challenging to prepare meshes for deep learning methods due to their discrete face structures and unordered elements. Furthermore, the topology of 3D shapes cannot be easily handled using meshes. Implicit representation of 3D shapes (Guillard et al., 2021; Park et al., 2019; Du et al., 2021; Luo et al., 2021) represents the surface of a shape by a continuous volumetric field that encodes the boundary of the shape as the set at the zero level of the learned implicit 3D shape function. It can better address different topologies of 3D shapes and requires less data storage, which is a promising representation for high-resolution 3D shapes. See Table 3.2 for the pros and cons of applying those four representations to deep learning methods.

In addition to the above four representations, there are a few new 3D representations that are promising for handling the trade-off between the effectiveness of training neural networks and the quality of the resulting 3D shapes. Neural Radiance Field (NeRF) (Mildenhall et al., 2020; Jain et al., 2022; Wang et al., 2022a) is a method for generating novel views of scenes or objects. It can take a set of input images of an object and render the complete object by interpolating between the images. NeRF (Mildenhall et al., 2020) is also topology-free and can be sampled at high spatial resolutions. However, 3D shapes represented by NeRF are “hidden in the black box" and we can only observe them through images rendered from different viewpoints. All the 3D representations mentioned above (i.e., voxels, point clouds, meshes, NeRF, and implicit representation) are generally not adapted to CAD software. This often brings about compatibility issues that could impede downstream editing and engineering analyses of the generated 3D shapes. To solve these problems, there are typically two ways. One way is to convert them to CAD models (e.g., converting STL/OBJ meshes to B-Rep solids). Another way is to handle the CAD shape data directly in deep learning models. Deep learning of unimodal CAD data is still an underexplored field, although some methods (Wu et al., 2021; Para et al., 2021; Ganin et al., 2021; Willis et al., 2021a; Jayaraman et al., 2021) and CAD datasets (Koch et al., 2019; Seff et al., 2020; Gryaditskaya et al., 2019; Regenwetter et al., 2021) have recently been introduced. DLCMT directly using CAD data (Smirnov et al., 2020) can be even more challenging due to the domain gap between design modalities and turns out to be a promising research direction.

Choosing the most appropriate 3D representation compatible with the adopted deep learning technique remains a challenging task. It involves considerations of data availability, data preprocessing, computational cost, visual quality of the resulting 3D shapes, data postprocessing, and the ability to adapt to later design stages.

3.5.4.4 Generalizability of DLCMT Methods

Finally, we noticed that efforts have been made to make the DLCMT methods more generalizable, independent of the variation between design objects (e.g., (Jin et al., 2020; Michel et al., 2022)). There are advantages and disadvantages to generalizing the methods. On the one hand, the diversity in different methods helps address the unique nature of different design problems, so a generalized approach may not be optimal for solving a specific design problem. On the other hand, generalizability allows a method to apply to a wider range of design problems. We focus on discussing the advantages here since we observe trending efforts (e.g., (Sanghi et al., 2022; Jain et al., 2022)) aiming to improve the generalizability of DLCMT methods in the review. It is challenging for deep learning methods to be generalized across multiple design problems (Regenwetter et al., 2022). The generalizability of a deep learning method means its ability to generalize to classes of objects beyond those used for training data. For engineering design applications, due to the sparsity of training data and the special treatment designed in the neural network architecture for a specific problem, a deep learning-based design method is difficult to generalize even in the cases where one design modality (e.g., 2D sketches or 3D shapes) is involved, let alone the generalization issues of applying DLCMT methods that involve multiple different modalities.

Some methods (Sanghi et al., 2022; Jain et al., 2022) utilize transfer learning techniques (e.g., zero-shot learning) and pre-trained models (e.g., CLIP (Radford et al., 2021)) or specially designed neural network architectures (e.g., unsupervised learning methods (Wang et al., 2021)) to improve generalizability, which could be good starting points for the engineering design community to further explore other possibilities. The challenge of generalizing the methods for DLCMT couples with other challenges and requires a community-wide effort to share datasets, create data repositories, define benchmark problems, and develop testing standards.

In summary, we have discussed the opportunities and challenges associated with applying DLCMT methods to conceptual design and proposed potential solutions to overcome the challenges with the insight gained from this literature review effort. The insights generated can potentially point to promising research directions for future studies.

3.6 Research Questions for Future Design Research

We notice that the opportunities and challenges identified previously are highly related to several trending topics in the engineering design community. In this section, we propose six research questions (RQs) that relate DLCMT to these trending topics: RQ (1) $\rightarrow$ design representations (Fuge, 2022); RQ (2) $\rightarrow$ generalizability and transferability of deep learning-based design methods (Song et al., 2022); RQ (3) $\rightarrow$ decision-making in AI-enabled design process (Li et al., 2021b); RQ (4) and (5) $\rightarrow$ human-AI collaboration (Song et al., 2020); RQ (6) $\rightarrow$ design creativity in deep learning-based design process (Elgammal et al., 2017). These RQs also point to potential research directions (see Section 3.7 for detail) where DLCMT can lead to. We hope these RQs can arouse a wide range of discussion and call for more efforts within the engineering design community to develop and apply DLCMT methods to address the challenges associated with conceptual design and beyond.

(1) What are the guidelines for selecting the most appropriate design representations in DLCMT?

(2) How much can the generalizability and transferability of the latent representation of multimodal data learned from DLCMT be extended across different product shape categories?

(3) Since DLCMT methods can shorten the cycle of generating designs and even connect to the downstream engineering analyses and manufacturing requirements, how could the information coming from the later design stages influence the regeneration of design concepts, and thereby a designer's decisions?

(4) DLCMT methods have the potential to facilitate the data-driven design process with humans in the loop, but how can we balance the involvement of humans and computers, and facilitate effective bidirectional human-AI communications to better stimulate designers' creativity at the human-AI interface?

(5) With the establishment of the human-AI interaction in the conceptual design based on DLCMT, what could the co-evolution between humans and AI look like?

(6) Although design creativity can be augmented by bringing humans in the loop when using DLCMT methods for product shapes generation, these methods

could suffer from the limitation of data interpolation inherently rooted in data-driven design methods. Fundamental questions, such as what new mechanisms and neural network architectures can be built to enable the algorithm to extrapolate beyond the training data, thus more effectively augmenting designers' creativity, shall be further explored in the future.

3.7 Closing Remarks

In this paper, we conducted a systematic review of the methods for deep learning of cross-modal tasks (DLCMT), including text-to-sketch, text-to-3D shape, and sketch-to-3D shape retrieval and generation methods, for the conceptual design of product shapes. Those methods could be applied in the Design Search, Design Creation, and Design Integration steps of conceptual design. Unlike other deep learning methods applied in engineering design, DLCMT allows human input of texts and sketches, which can explicitly reflect designers' and/or users' preferences. As designers can be more actively involved in such a design process, human-computer interaction and collaboration are promoted, thereby it has a great potential to improve the conceptual design of products using a data-driven design process with humans in the loop compared to traditional design automation methods and computer-aided design methods. DLCMT could also facilitate the engineering design education and democratization of product development by allowing intuitive inputs (e.g., text descriptions and sketches), and an immersive design environment by integrating VR, AR, and MR techniques.

With the attempt to apply new 3D data representations in DLCMT and the availability of more public datasets, opportunities open up for the development of new methods for DLCMT. However, the deficiency of training datasets, trade-off in the choice of representations of 3D shapes, lack of consideration of engineering performance, manufacturability, and part-aware design, and the ability of generalization still challenge the engineering design community to apply DLCMT to engineered product design. We would like to encourage attention and efforts from the engineering design community.

There are a few limitations in the current literature review that the authors would like to acknowledge and share. First, the set of keywords used to search the literature has covered all topics in our scope of the review. However, other topics, such as shape-to-text generation (namely, shape captioning in the literature), could also be of interest to the engineering design community. Second, for the topics of sketch-to-3D shape retrieval and generation, we did not include all relevant articles, although we have covered the most influential and the most recent publications.

In the future, we will continue the review and conduct a more comprehensive analysis of the relevant works on DLCMT. Besides the review effort, we see the merit of conducting a comparative study to further understand the effects of DLCMT on the conceptual design by enabling and disabling the DLCMT-based assistance in the design process. We believe that the methods reviewed, the discussion of opportunities, challenges, potential solutions, and future research directions of applying DLCMT to conceptual product shape design can benefit the data-driven design research in the engineering design community. We hope this review effort can also facilitate the discussion and attract more attention from the engineering design community and industry stakeholders when applying DLCMT to improve the conceptual design of product shapes and beyond.

Chapter 4: A Predictive and Generative Design Approach for 3D Mesh Shapes Using Target-Embedding Variational Autoencoder

Abstract

We present a predictive and generative design approach to supporting the conceptual design of product shapes in 3D meshes. We develop a target-embedding variational autoencoder (TEVAE) neural network architecture, which consists of two modules: 1) a training module with two encoders and one decoder (E^2D network); and 2) an application module performing the generative design of new 3D shapes and the prediction of a 3D shape from its silhouette. We demonstrate the utility and effectiveness of the proposed approach in the design of 3D car bodies and mugs. The results show that our approach can generate a large number of novel 3D shapes and successfully predict a 3D shape based on a single silhouette sketch. The resulting 3D shapes are watertight polygon meshes with high-quality surface details, which have better visualization than voxels and point clouds and are ready for downstream engineering evaluation (e.g., drag coefficient) and prototyping (e.g., 3D printing).

4.1 Introduction

[1] [2] Sketching plays an essential role in sparking creative ideas to explore emerging design concepts (Pratt et al., 2005). For example, in car design, characteristic contour lines are often used to represent silhouettes in supporting the conceptual design of car body shapes (Reid et al., 2010; Gunpinar et al., 2019), which complements many other ideation approaches, such as freehand sketches, design analogies, and prototypes. Compared to freehand sketches, silhouettes regularize the sketching process and thereby make sketching easier and more manageable. This is particularly useful for designers who lack professional sketching skills. However, silhouettes, as a specific type of 2D sketch, are often ambiguous and lack geometric details. In later design stages, such as embodiment design, a 3D computer-aided design (CAD) model is often required to more accurately evaluate the engineering performance of a design concept. 3D shapes can also provide better visualization and thus help designers better understand the design, inspiring them to develop new shapes and refine geometric details. Therefore, the question is: can we build a system to predict and automatically generate 3D shapes just based on silhouettes?

Such a system will yield several benefits. First, it automates the 2D-to-3D reconstruction process, thereby saving labor and time. Designers can allocate more resources for better design iteration and ideation. Second, all silhouettes created during the conceptual design stage can be evaluated against the desired engineering

[1] This chapter except for Section 4.5 has been published in the following paper in the Journal of Mechanical Design. Li, X., Xie, C., and Sha, Z. (August 8, 2022). A Predictive and Generative Design Approach for Three-Dimensional Mesh Shapes Using Target-Embedding Variational Autoencoder. *ASME. J. Mech. Des.* November 2022; 144(11): 114501. `https://doi.org/10.1115/1.4054906`. I am the leading author of this paper and was primarily responsible for developing the methodologies, conducting experiments, analyzing the results, and writing the manuscript.

[2] Section 4.5 of this chapter has been published as a case study in the following paper in the International Journal of Human-Computer Interaction. Demirel, H. O., Goldstein, M. H., Li, X., & Sha, Z. (2023). Human-Centered Generative Design Framework: An Early Design Framework to Support Concept Creation and Evaluation. *International Journal of Human-Computer Interaction*, 40(4), 933–944. `https://doi.org/10.1080/10447318.2023.2171489`. I am the co-author of this paper and was primarily responsible for developing the methodologies, conducting experiments, analyzing the results, and writing the manuscript for Sections 2.1, 4.1, and 5.1.

performance in 3D form. So, designs that would have better performance will not be ruled out too early when performance-driven decisions (i.e., rational decisions) are not yet obtained. Third, ordinary people would not be discouraged from showing their design ideas merely due to their lack of CAD experience or sketching skills. This may have significant educational implications for training novice designers and facilitate the democratization of design innovation. Lastly, enterprises may use this system to enable user interface soliciting consumer preferences for design customization.

However, automatically reconstructing 3D shapes directly from 2D sketches is a challenge because it is an ill-defined problem due to insufficient and imperfect information from simple strokes (Schmidt et al., 2009). To tackle this challenge, inspired by the target-embedding autoencoder (TEA) network (Jarrett and van der Schaar, 2020; Girdhar et al., 2016), we propose a novel target-embedding variational autoencoder (TEVAE) (see Fig. 4.1(a)). The TEVAE architecture consists of two modules: 1) A training module with an E^2D network that has two encoders and one decoder. 2) An application module performing two functions: *generative design* function, such as shape interpolation and random generation of new 3D shapes; and *predictive design* function (i.e., 3D shape prediction from silhouette sketches). The integration of generative and predictive functions is beneficial in that it makes the structure of the neural network compact, thus saving training costs. To demonstrate the utility and generalizability of the proposed approach, we apply it to two case studies in the design of 3D car bodies and mugs.

The contributions of this paper are summarized below.

a). To the best of our knowledge, this is the first attempt to develop a system integrating both predictive and generative functions of 3D mesh shapes from silhouettes. The TEA has a classic autoencoder that can perform pseudo-generative tasks, but essentially, is not a generative model. Our TEVAE applies variational autoencoders (VAEs) and becomes a true generative model. It can also learn a continuous and smooth latent representation of the data.

b). Predicting 3D shapes from silhouettes is more challenging because a silhouette sketch provides less information (e.g., depth and normal maps) than traditional freehand sketches with inner contour lines. To that end, we introduce an intermediate step (e.g., extrusion or rotation) to first convert the silhouette to a 3D primitive shape. This transforms the original 2D-to-3D problem into a 3D-to-3D problem, which promotes a stable training process and the generation of reliable and viable 3D shapes.

c). Building upon a graph convolutional mesh VAE (Yuan et al., 2020), our approach can directly output high-quality 3D mesh shapes that are more storage-efficient for high-resolution 3D structure, compared to point clouds (Lun et al., 2017) and voxels (Jin et al., 2020). 3D meshes also facilitate engineering analyses because they are compatible with existing computer-aided engineering (CAE) software [3].

d). A data automation program is developed for training data pairs of 3D shapes that can be used in any TEA-like neural network for supervised learning problems.

4.2 Literature Review

In this section, we review the existing research that is most relevant to our work.

[3] "Meshes" are used for 3D representation here. In CAE software, there is a concept called "meshing". Meshing is a process that breaks down the continuous geometric space of an object into a discrete number of shape elements. All 3D representations including meshes and native CAD data format (e.g., IGES, DWG, and STL) that can be directly input to CAE software have to go through the meshing process for analysis.

4.2.1 Learning-Based Sketch-to-3D Generation Methods

Point clouds and voxels have been widely used as 3D representations for sketch-to-3D generation (Lun et al., 2017; Nozawa et al., 2021; Jin et al., 2020). These methods need to postprocess the resulting 3D shapes to meshes for better visualization, which still suffer from low surface quality. There are studies attempting to directly produce high-quality mesh shapes. For example, concurrent with the development of our approach, Guillard et al. (Guillard et al., 2021) propose a pipeline to reconstruct and edit 3D shapes from 2D sketches. They train an encoder/decoder architecture to regress surface meshes from freehand sketches. The method applies a differentiable rendering technique to iteratively refine the resulting 3D shapes. Similarly, Xiang et al. (Xiang et al., 2020) integrate a differentiable rendering approach to an end-to-end learning framework for predicting 3D mesh shapes from line drawings.

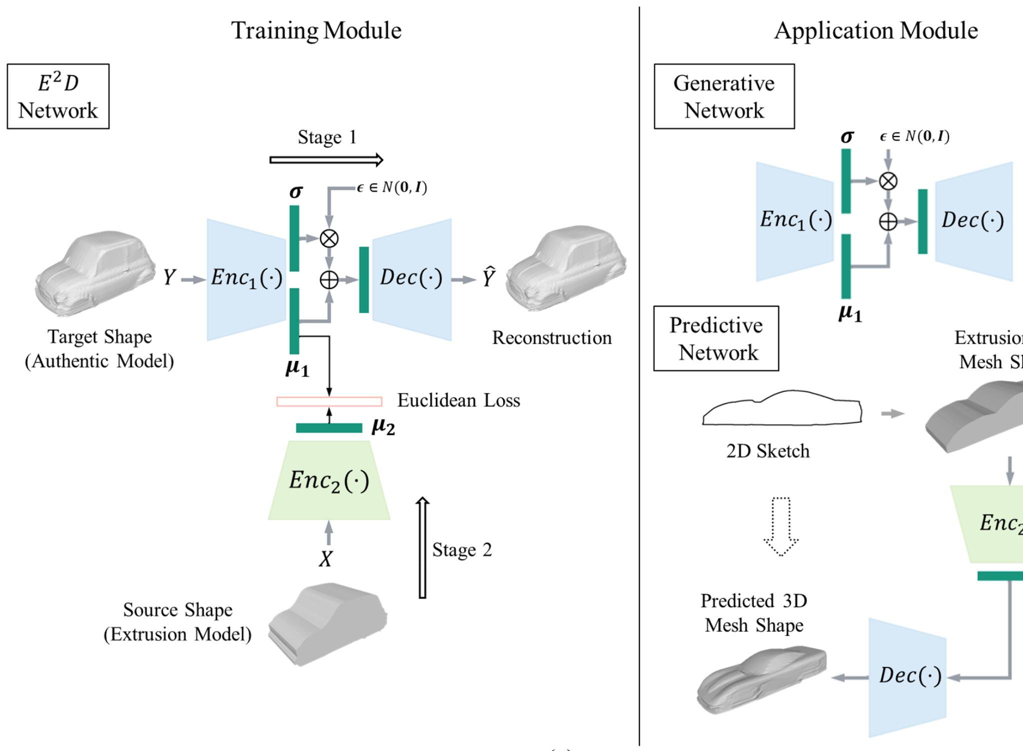
All methods above are promising and have inspired us to explore a more challenging task, i.e., to predict a 3D shape from a simple silhouette sketch. Our approach is similar to (Nozawa et al., 2020, 2021; Han et al., 2017; Guillard et al., 2021) in that we only need one single sketch as input, but we create a new neural network architecture that can predict a 3D shape from a single silhouette and simultaneously generate novel 3D shapes. The direct output shapes are 3D meshes with high-quality surface details, thus, requiring no postprocessing.

4.2.2 Learning-Based Generative Design Methods

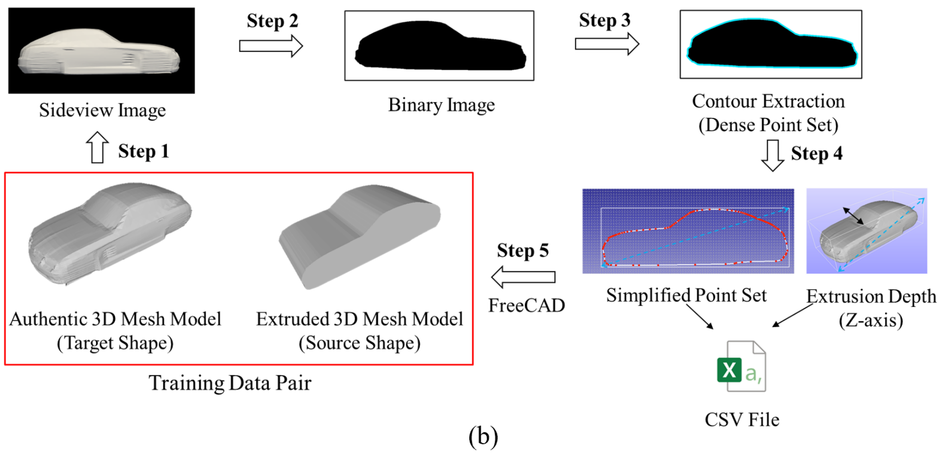
Learning-based generative design (GD) methods have been primarily developed based on two techniques, generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational autoencoders (VAEs) (Kingma and Welling, 2013). There are several approaches for 2D designs (Oh et al., 2019; Dering et al., 2018; Fujita et al., 2021), but they are not appropriate for design applications that require 3D models. In 3D applications, Shu et al. (Shu et al., 2020) present a method that combines GAN and the physics-based virtual environment introduced in (Dering et al., 2018) to generate high-performance 3D aircraft models. Zhang et al. (Zhang et al., 2019) propose a method using a VAE, a physics-based simulator, and a functional design optimizer to synthesize 3D aircraft with prescribed engineering performance. Building upon (Oh et al., 2019), Yoo et al. (Yoo et al., 2020) develop a deep learning-based CAD/CAE framework that can automatically generate 3D car wheels from 2D images. Gunpinar et al. (Gunpinar et al., 2019) apply a spatial simulated annealing algorithm to generate various silhouettes of cars, which are then extruded to 3D car models. Those models can be further refined by sweeping a predefined cross-section sketch. However, simply extrusion does not guarantee satisfactory outcomes, and the resulting 3D car models look unreal.

4.2.3 Target-Embedding Representation Learning

Girdhar et al. propose a TL-embedding network (Girdhar et al., 2016) that is composed of a T-network for training and an L-network for testing. The T-network contains an autoencoder (encoder-decoder) network and a CNN. After training, the L-network can be used to predict 3D shapes in voxels from images. Similarly, Mostajabi et al. (Mostajabi et al., 2018) uses an autoencoder and a CNN to perform the semantic segmentation task of images. Dalca et al. (Dalca et al., 2018) apply a similar network structure as (Girdhar et al., 2016; Mostajabi et al., 2018), consisting of a prior generative model to generate paired data (biomedical images and anatomical regions) to solve the scarcity of labeled image data for anatomical segmentation tasks. Jarrett and Schaar (Jarrett and van der Schaar, 2020) categorize these studies as supervised representation learning methods. They observe that when the dimension of the target data space is higher or similar to the feature data space, a target-embedding autoencoder (TEA) can be more effective than a feature-embedding autoencoder (FEA). The authors verify that the TEA structure will guarantee learning stability by using a mathematical proof of a simple linear TEA and showing the empirical results from a complex non-linear TEA. Inspired by those existing works, we construct the target-embedding variational autoencoder (TEVAE) architecture.



(a)



(b)

Figure 4.1: (a) The proposed approach using target-embedding variational autoencoder (TEVAE); (b) The preparation of data pairs

4.3 Approach

The proposed target-embedding variational autoencoder (TEVAE) architecture, shown in Fig. 4.1 (a), consists of two modules: a training module and an application module.

4.3.1 The E^2D Network and the Two-Stage Training

The key component of the training module is the E^2D network that consists of two encoders and one decoder, which is constructed by concatenating an encoder (labeled as $Enc_2(\cdot)$) to a mesh VAE (an encoder-decoder network, labeled as $Enc_1(\cdot)$ and $Dec(\cdot)$) (Yuan et al., 2020). The $Enc_1(\cdot)$ maps target shapes (S_t, i.e., the original authentic 3D mesh shapes) to a low-dimensional latent space, and the $Dec(\cdot)$ maps latent vectors from that latent space to 3D mesh shapes. With the same network structure as $Enc_1(\cdot)$, $Enc_2(\cdot)$ takes source shapes (S_s, the 3D mesh shapes extruded from silhouette sketches of the target shapes) as the input and maps them to the same dimensional latent space as the mesh VAE.

We adopt the same loss function developed in (Yuan et al., 2020) to train $Enc_1(\cdot)$ and $Dec(\cdot)$. For $Enc_2(\cdot)$, we create a new loss function L_2 as below.

$$L_2 = \alpha D_{Regress} + D_R, \tag{4.1}$$

where α is the weight for the regression loss and

$$D_{Regress} = \frac{1}{2M} \sum_{i=1}^{M} \left\| {\boldsymbol{\mu_2}}^i - {\boldsymbol{\mu_1}}^i \right\|_2^2 \tag{4.2}$$

denotes the Euclidean loss, where ${\boldsymbol{\mu_2}}^i$ is the output of $Enc_2(\cdot)$ using X^i as input and ${\boldsymbol{\mu_1}}^i$ is the mean vector obtained from the latent space of the mesh VAE using the input of Y^i. D_R is the regularization loss applied to improve the generalization ability of the $Enc_2(\cdot)$.

We apply a two-stage training strategy (Mostajabi et al., 2018) to jointly train the mesh VAE and $Enc_2(\cdot)$ from scratch. In Stage 1, the mesh VAE is trained independently. In Stage 2, we fix all the learning parameters of the mesh VAE and train $Enc_2(\cdot)$ by minimizing L_2.

4.3.2 The Predictive Network and Generative Network

After the E^2D network is trained, we connect $Enc_2(\cdot)$ to $Dec(\cdot)$ to form the predictive network. It can take a 3D extrusion mesh shape as input and output a 3D mesh shape that is similar to the input shape but has finer geometric details, making it authentic and aesthetic. We use the trained mesh VAE as the generative network, which can perform generative design tasks, including shape reconstruction, interpolation, and random generation.

4.3.3 Preparation of Data Pairs

Data pairs $\{S_s^i, S_t^i\}_{i=1}^N$ are needed to train the E^2D network. Fig. 4.1(b) shows the process of obtaining one training data pair using a car body as an example. From the sideview image of an authentic 3D car model, we extract its contour points, from which we can obtain an extrusion model using the FreeCAD Python API. We develop a set of Python scripts that fully automate the whole process, which is made open-source for the community [4]. We process $N = 1240$ car models obtained from (Umetani, 2017) and $N = 203$ mug models from (Gao et al., 2019c). For car models, we keep only car bodies by removing all the other parts, such as mirrors, wheels, and spoilers.

4.3.4 Shape Preprocessing and Feature Representation

The E^2D network requires input mesh shapes in both the source shape set ($\{S_s^i\}_{i=1}^N$) and the target shape set ($\{S_t^i\}_{i=1}^N$) to have the same topology (i.e., the same number of vertices and the same mesh connectivity). However, the mesh typologies between the two datasets can be different. For simplicity, we use a uniform topology

[4]https://github.com/Xingang1990/TEVAE

for both datasets and the non-rigid registration method (Zollhöfer et al., 2014) is applied to meet this requirement. Nonrigid registration is a widely used technique in the computer graphics field to map one point set (e.g., point cloud, mesh) to another. A uniform unit cube mesh with 19.2k triangles (9602 vertices) is used to register all mesh shapes. This makes all shapes have the same topology as the cube mesh, but remain the same as their original shapes and geometric details.

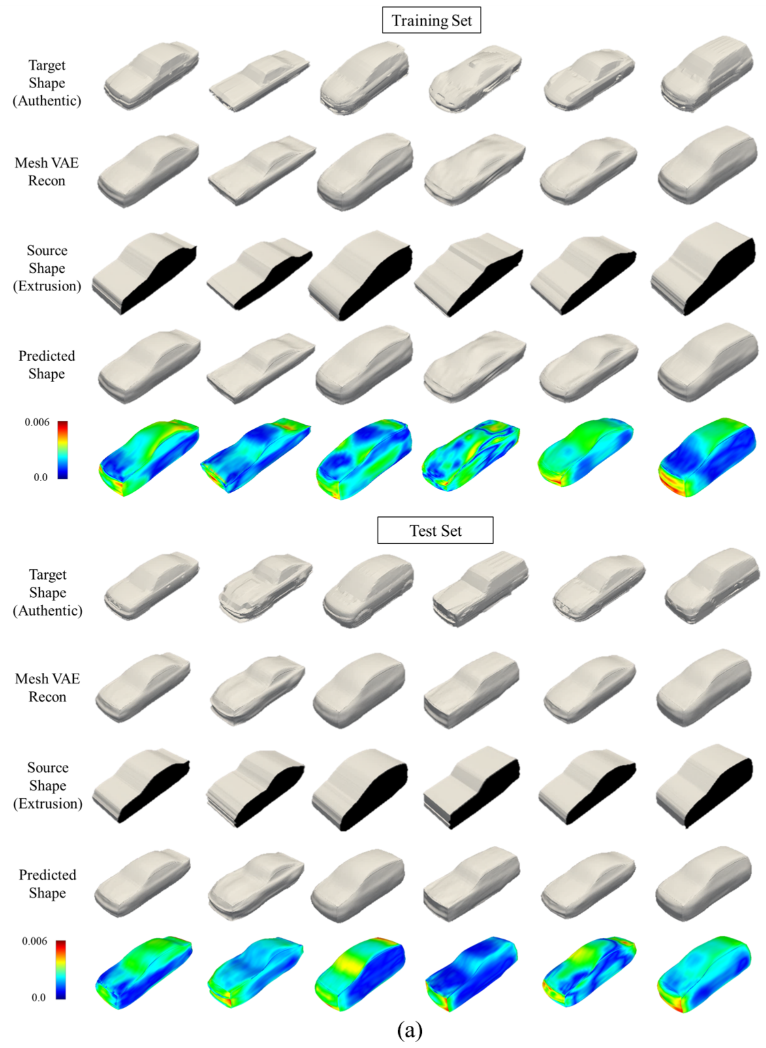
The As-Consistent-As-Possible (ACAP) method (Gao et al., 2019a) is applied to extract features of a 3D shape to input to the E^2D network. We deform the aforementioned uniform cube mesh to a target 3D mesh shape by multiplying deformation matrices, from which nine unique numbers can be extracted for each vertex of the mesh shape. Thus, a shape with v vertices can be represented by a feature matrix $M_f \in \mathbb{R}^{v\times 9}$, where $v = 9602$ in our implementation. We can get the feature representations of the source shape dataset $X = \{X^k\}_{k=1}^N$ and the target shape dataset $Y = \{Y^l\}_{l=1}^N$, where $N = 1240$ for the car models and $N = 203$ for the mug models, and $\{X^i, Y^i\}$ forms the input feature of one data pair.

More details of the approach and the training of the E^2D network are provided in the Supplementary Material in Appendix B.

4.4 Case Studies and Results

4.4.1 Implementation of the Two-Stage Training

For training the mesh VAE in Stage 1, the input target shape dataset $Y = \{Y^l\}_{l=1}^N$ is randomly divided into the training set (80%) and the test set (20%). For training the $Enc_2(\cdot)$ network in Stage 2, we also do an 80-20 split of the source shape dataset $X = \{X^k\}_{k=1}^N$, and meanwhile use the data pair to make sure the i^{th} target shape (S_t^i) corresponds to the i^{th} source shape (S_s^i) in both the training and test sets.



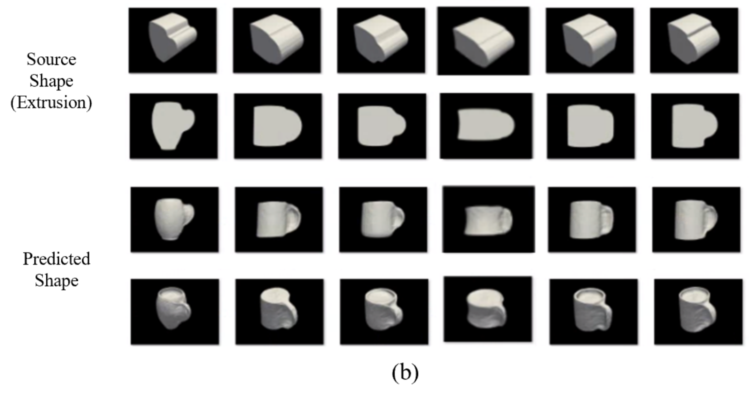
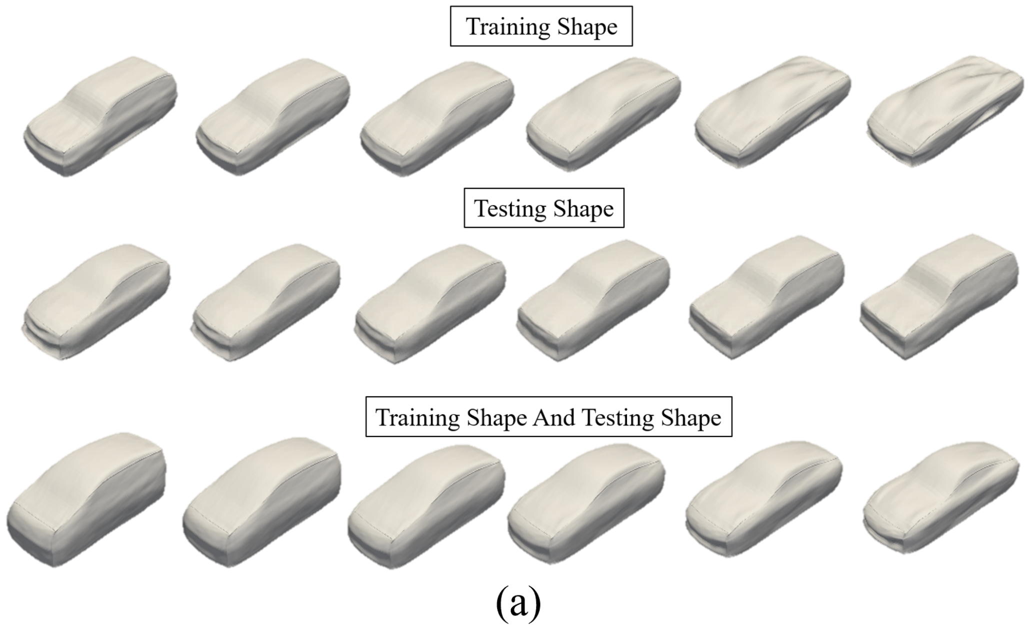


Figure 4.2: (a) Results of car bodies: predicted shapes (in the fourth row) and reconstructed shapes (in the second row); (b) The prediction results of mugs



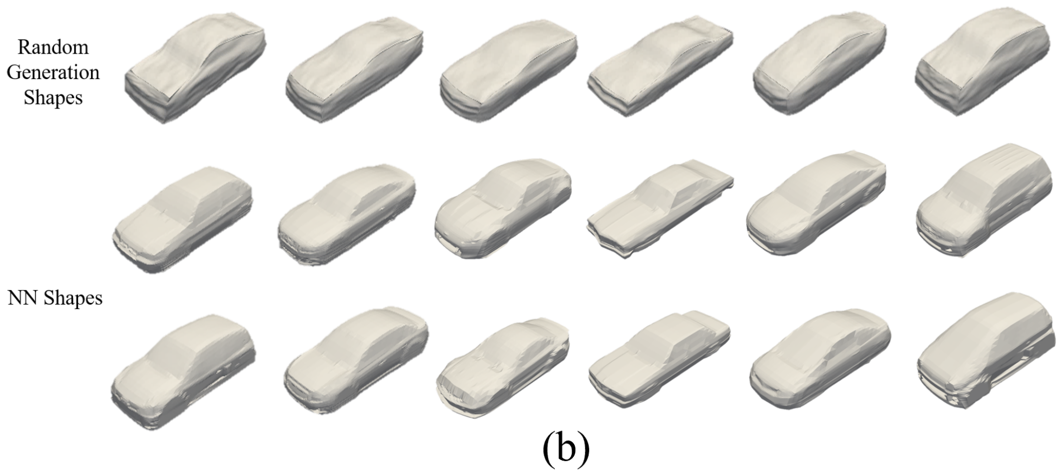


Figure 4.3: (a) The results of shape interpolation for three cases; (b) The results of random generation shapes along with two nearest neighbor (NN) shapes

4.4.2 The Predictive Network

The predictive network aims to predict a 3D shape from an input silhouette sketch. We conducted experiments on the prediction of the training set and the test set. The results of the car models are shown in Fig. 4.2(a). For the result of the training set, the first row shows the target shapes, and the following rows are their corresponding reconstruction shapes from the mesh VAE, extrusion shapes (with silhouettes marked in dark), and the predicted shapes, respectively. The results indicate that, given an input extrusion shape from the corresponding silhouette sketch, the predictive network is capable of predicting an authentic 3D shape, as illustrated in the fourth row. It should be noted that even though we are targeting shapes (ground truth) in the first row, the best results that can be achieved from the predictive network are the reconstruction shapes in the second row. The reconstruction shapes and the corresponding predicted shapes look identical in terms of visual appearance, but are different in geometric details. To show the difference, we compute the Hausdorff distance between those shapes and visualize the distance values in the fifth row. Similar results are also observed for the shapes in the test set, which indicates a good generalization of the network because the test set shapes are unseen data for the network. This is particularly important in real-world applications, where user input often does not resemble existing shapes in a training dataset.

The prediction results of the mug models are shown in Fig. 4.2(b). The first two rows are the source shapes that are obtained from extruding the silhouettes, while the last two rows are the corresponding predicted shapes. Mugs are generally non-extrudable from side-view silhouettes, so the extrusion shapes look more like toast instead of mugs. However, our approach can still predict authentic mug shapes. Please note that, besides extruding, other 3D modeling techniques, such as revolving and sweeping, can also be used to obtain 3D primitive shapes. Extruding is adopted in this study for ease of implementation. In addition, it provides us with basic geometric features for 3D shape prediction.

4.4.3 The Generative Network

For the generative network, different generative operations, such as shape reconstruction, interpolation, and random generation, can be performed. The reconstructed 3D shapes are already shown in the second row for both the training set and the testing set in Fig. 4.2(a). For shape interpolation, new 3D shapes are synthesized by linearly interpolating two target 3D shapes through their encoded latent vectors. We demonstrate the results of shape interpolation in three cases using the case study of car models (see Fig. 4.3(a)): 1) interpolation between two training shapes, 2) between two test shapes, and 3) between a training shape and a test shape. In each case, the first and the last columns are the shapes to be interpolated, and the in-between columns are linearly interpolated shapes. It can be observed that there is a gradual transition of the shape geometry between the two target shapes.

For random shape generation, latent vectors are randomly sampled from the latent space of the Mesh VAE, and decoded by the trained $Dec(\cdot)$ to 3D mesh shapes. Fig. 4.3(b) shows that the generative network can generate novel car models (in the first row) that are not seen in the original dataset. This is validated by finding their nearest neighbors (NNs) (the second and third rows) in the original dataset based on the Hausdorff distance. A quick visual comparison between the randomly generated car models and their NNs tells the differences, and they are indeed new shapes.

4.5 Graphic User Interface Development

Gunpinar et al. (2019) propose to use nine characteristic lines (e.g., front bumper, grille, rear windshield, and trunk) to represent the silhouette of a car. Bézier curves (Bézier, 1968) can be used to mathematically represent these lines with predefined control points (e.g., three control points for quadratic curves and four points for cubic curves). With such a system, users can create a diverse set of silhouettes by adjusting the control points of each characteristic line. In this work, we integrate the technique from (Gunpinar et al., 2019) into our GD framework. This approach

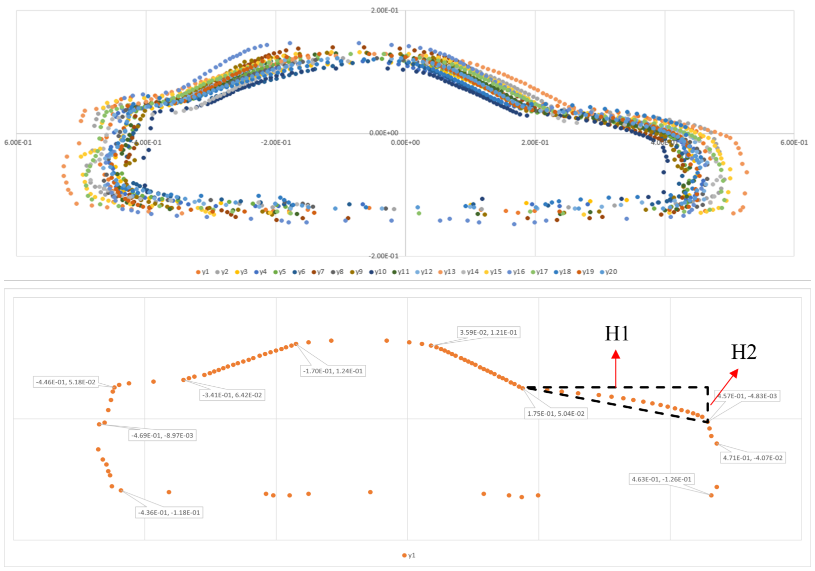


Figure 4.4: Contour points of the 20 sampled car models with an example to show how we obtain the boundary for control points of a characteristic curve.

enables designers to create 3D mesh shapes using the sketch-to-3D mesh model. Our framework can take human input as 2D silhouette sketches and output authentic 3D mesh shapes. The GD-based concept creation approach introduces how one can collect 2D automobile silhouette sketches from designers and auto-populate 3D vehicle designs that are authentic.

In developing the graphic user interface (GUI) that enables the above GD-based vehicle design, it was essential to identify the boundary within which users can adjust the control points of their desired curves and silhouette sketches. To obtain a reasonable boundary limit for each characteristic line, authentic automobile models were used as references. We randomly sampled 20 models from a set of automobile models (Umetani, 2017) and obtained the contour points using the method introduced in Figure 4.1(b). The top portion of Figure 4.4 shows the contour points of the 20 car models sampled. The starting and ending points of each line were manually identified. The smallest right triangle that can enclose one characteristic line was created by identifying its two legs (e.g., H_1 and H_2 in the lower image of Figure 4.4). After getting the data for all 20 car models, we averaged the length of each leg to get

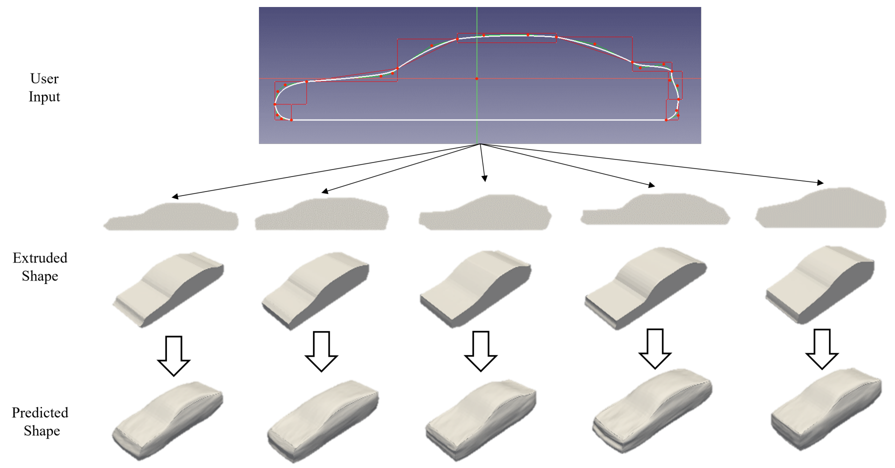


Figure 4.5: Prediction of 3D shapes from 2D sketches input from users.

the boundary limits. To promote more diverse sketches, we set a rectangle boundary for some lines based on the features of different lines (i.e., lines with more variations of shapes), as shown in the first row of Figure 4.5. Users can move around the control points within the corresponding boundary to input the silhouette sketch that best matches their preferences.

We created a human-computer interface using FreeCAD 0.18 [5], which was used to collect human input of silhouette sketches. These sketches served as 2D profiles, which were used to create extruded car models of meshes. Figure 4.5 shows some of the extruded car models in side view and isometric view based on 2D silhouette sketches. After the 3D extrusions were created, the predictive model stepped in and generated authentic 3D shapes. The final row in Figure 4.5 shows a few examples of the predicted car models. It is observed that the predicted 3D shapes resemble the 2D sketches in terms of the side view, but provide finer geometric details in the third dimension, thus making the car shapes authentic. Since those 2D sketches are entered by users based on their preferences, the predicted 3D shapes will facilitate their ideation process. We expect that these 3D shapes will provide an additional outlet for users to help better understand their designs.

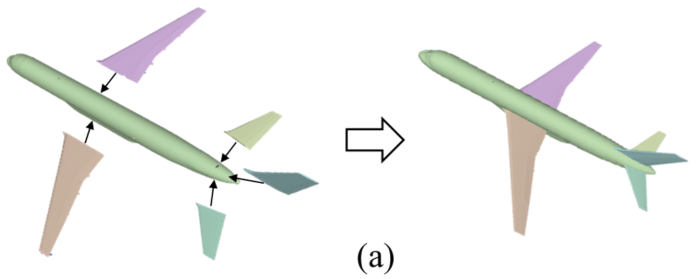
[5]Obtained from `https://wiki.freecadweb.org/Main_Page`

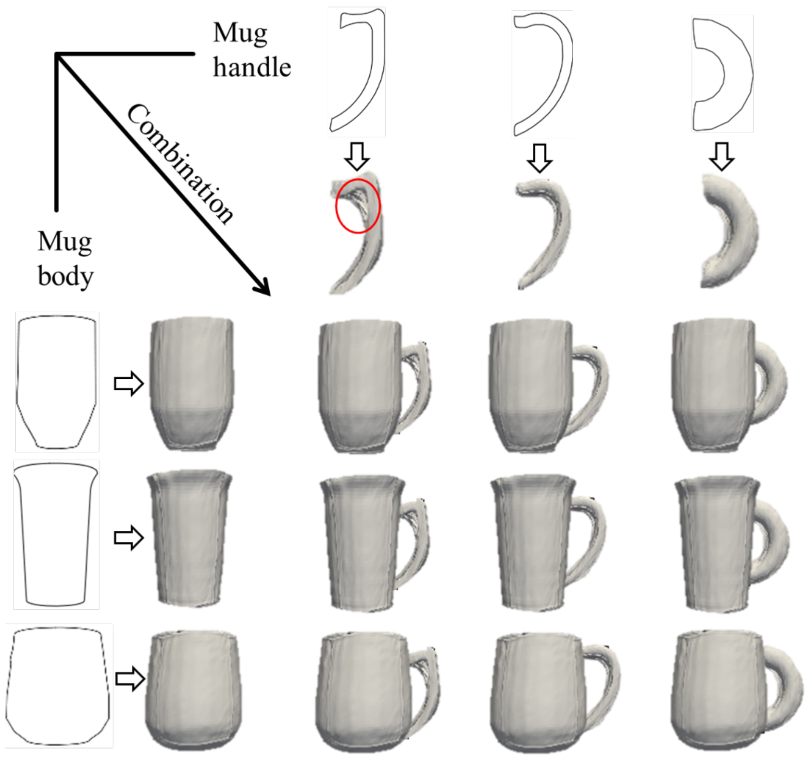
4.6 Conclusion

To tackle the challenge of predicting a 3D shape from a silhouette sketch, we present a novel target-embedding variational autoencoder (TEVAE) network that enables a 2D-to-3D design approach. Our approach can effectively predict a 3D shape from a silhouette sketch. The predicted 3D shape is consistent with the input sketch and is authentic with rich geometric details. Such a design transformation could greatly shorten the iteration between the design ideation to CAD modeling. The approach can also generate novel 3D shapes, and thus could better inspire designers for their creative work. The resulting 3D shapes are represented in meshes, which are ready for downstream engineering analyses, evaluation, and prototypes (e.g., 3D printing).

Quantity yields quality, and this can be achieved by broadening the initial pool of concept ideas (Yang, 2003). We believe that the presented approach can help designers explore the design space more efficiently and stimulate creative design ideas in the early design stages. From the methodology point of view, this new generative design approach is general enough to be applied in many applications where 3D shape modeling and rendering are necessary. As long as the sketch can provide a major perspective view of an object, like the frontview of a human body and the sideview of a bottle, the corresponding authentic 3D shape can be predicted and novel shape concepts can be generated. In addition, our approach is friendly to ordinary people who have few professional sketching skills, since it only requires a simple silhouette of an object as input.

There are a few limitations in the current study that the authors would like to share. First, the current model only handles genus-zero shapes and ignores any through holes (e.g., the hole between the body and the handle of a mug in Fig. 4.2(b)) in the original shape due to the non-rigid registration (Zollhöfer et al., 2014). However, many design artifacts are usually non-genus-zero (e.g., a mug with a through hole between the body and the handle) or have more complex geometry consisting of


(a)



(b)

Figure 4.6: Complex geometries (e.g., a toy plane) or non-genus-zero shapes (e.g., a mug) can be partitioned into several genus-zero shapes. Then, the proposed approach can be applied to each component for shape exploration and synthesis.

many components, e.g., a plane model can have a body, two wings, and three tails, etc., as shown in Fig. 4.6 (a).

To address this limitation, a part-aware method (Gao et al., 2019c; Li et al., 2021c) may be a potential solution. We perform a quick experiment using a part-aware mug design problem (see Fig. 4.6(b)). In this particular application, users can first draw an outline sketch for an individual component (e.g., a mug body or a handle). Then, the corresponding 3D mesh shape can be predicted and new shapes can be generated. Lastly, the resulting individual components can be combined into a holistic structure allowing non-genus-zero topology. However, the part-aware strategy could not work for parts that are non-genus-zero and unable to be further decomposed into genus-zero components, e.g., a chair back with hollow-out structures and holes. In this experiment, we applied a cube mesh template to register mug models using non-rigid registration (Zollhöfer et al., 2014) as introduced previously. However, we observed that artifacts with a large curvature could not be perfectly registered (e.g., the mug handle in Fig. 4.6(b) highlighted by a circle). This issue can be alleviated by using different templates of 3D primitives, e.g., a sphere or a cylinder.

In addition to the part-aware method, other methods based on new 3D representations could also be applied to address the first limitation. For example, primitive-based methods can use a set of primitive surfaces to represent a 3D shape (Paschalidou et al., 2021; Vasu et al., 2021; Jones et al., 2021; Groueix et al., 2018). Implicit 3D representation (e.g., signed distance fields (Chen and Zhang, 2019; Park et al., 2019)) can characterize 3D surfaces implicitly, and the resulting 3D geometries can be converted to mesh representation. These methods can capture the topology changes of 3D shapes without using a template mesh for data registration, thus deserving our future exploration.

Second, the constructed 3D shapes are consistent with the designer's sketch in terms of sideview, but they might not be the same as what the designer has in mind. To address this limitation, we plan to integrate interactive modeling techniques into the proposed GUI for users to further adjust the generated 3D shapes according to their preferences.

Third, the generative network performs well in shape reconstruction and shape interpolation, but the success rate of random shape generation is lower than one-third due to the sparsity of the training data. Therefore, the random shape generation function is not fully reliable in practice for now. This problem could be solved by obtaining more 3D shape data using data augmentation methods, such as the one presented in the study of (Nozawa et al., 2020), to improve the diversity and quality of the training dataset. These limitations motivate us to further improve the current model in the future.

Chapter 5: Image2CADSeq: Computer-Aided Design Sequence and Knowledge Inference from Product Images

Abstract

This research introduces a novel data-driven approach with an Image2CADSeq neural network model. This model aims to reverse engineer CAD models by processing images as input and generating CAD sequences. These sequences can then be translated into B-rep models using a solid modeling kernel. Unlike B-rep models, CAD sequences offer enhanced flexibility to modify individual steps of model creation, providing a deeper understanding of the construction process of CAD models. To quantitatively and rigorously evaluate the predictive performance of the Image2CADSeq model, we have developed a multi-level evaluation framework for model assessment. The model was trained on a specially synthesized dataset, and various network architectures were explored to optimize the performance. The experimental and validation results show great potential for the model in generating CAD sequences from 2D image data.

5.1 Introduction

Computer-aided design (CAD) systems can significantly reduce design time by avoiding the need for traditionally required labor-intensive manual drawings (Rosato and Rosato, 2003). Contemporary CAD systems such as Fusion 360, SOLIDWORKS, and OnShape enable designers to create and modify CAD models [1] through a sequence of CAD operations. However, in certain scenarios, the CAD model of a product may not be readily available due to various factors, including outdated documentation, lack of digital records, and commercial reasons. Reverse engineering (RE) is employed to overcome these obstacles, utilizing measurement and analysis tools to reconstruct CAD models (Varady et al., 1997; Buonamici et al., 2018).

Integrating RE with CAD systems can not only allow designers to leverage the advantages of existing products while incorporating their own innovative ideas and improvements but can also be used for design knowledge restoration and management. However, the traditional RE process faces two major limitations. First, it focuses on reconstructing 3D models rather than CAD sequences. Compared to 3D models, a CAD sequence provides access to the historical construction process and associated design knowledge and it facilitates geometry modification using parametric modeling. Second, the process has been performed primarily manually, making it labor-intensive and time-consuming. Recently, researchers have explored data-driven methods, such as converting 3D point clouds (Uy et al., 2022; Ren et al., 2022) or voxels (Lambourne et al., 2022; Li et al., 2023a) into CAD models. Nevertheless, these 3D input data are often challenging to acquire due to inaccessibility and unavailability. 3D scanning could be a solution, yet quality is often unsatisfactory and cost is an unavoidable factor to consider when acquiring specialized equipment and expertise.

Compared to point clouds or voxels, images are easier to acquire given the popularity of mobile devices. Thus, our question arises: How can we reverse engineer CAD sequences directly from 2D images in supporting designers to interpret and edit

[1]CAD models are structured, parametric, or operation-based 2D or 3D design, allowing for a high level of control and flexibility in the design process. They are particularly suitable for industrial and engineering designs, where precision and the ability to easily modify designs are crucial. There are two main types of CAD models: 1) constructed solid geometry (CSG) and 2) parametric CAD models including CAD sequence data and boundary representation (B-rep) models. In contrast, discrete 3D representations, such as meshes and point clouds, are more static and less flexible in terms of parametric editing and design exploration (Wu et al., 2021; Para et al., 2021).

CAD models during the design and modeling process? After a thorough literature review, we realize that there is a scarcity of research exploring answers to this question. Therefore, our objective is to develop a data-driven approach that can generate a sequence of CAD operations based on a single image (referred to as “Image2CADSeq” hereafter for brevity).

The contributions of the proposed approach are summarized as follows.

a). To the best of our knowledge, this study is the first attempt to predict a sequence of CAD operations given a single image input (i.e., single-view image to CAD sequence prediction). We developed a target-embedding variational autoencoder (TEVAE) architecture (Li et al., 2022c) to solve this problem. The proposed approach has the potential to streamline the CAD design process, reducing the time and effort required to create 3D models from 2D sketches or photographs.

b). We created a novel data synthesis pipeline based on the design grammars defined in the domain-specific language (DSL), Fusion 360 Gallery. The pipeline can generate synthetic data that resemble real-world images and CAD models. It can also be used as a data augmentation method to improve the quality and quantity of existing training data sets, making the data more diverse and robust for training image2CADSeq models.

c). We developed a multi-level evaluation framework to assess the Image2CADSeq performance, encompassing three components: CAD sequences, 3D models, and the corresponding images. Specifically, the evaluation of the CAD sequence is performed at multiple levels and hierarchies (see Section 5.4.5 for details) to quantitatively assess the predictive performance of the proposed model architectures.

We anticipate that the proposed approach has the potential to revolutionize existing CAD systems by making the CAD model reconstruction process more accessible. This would enable both experienced and novice designers to actively contribute to the design, promoting design collaboration and design education. Moreover, it has the potential to provide a unique pathway to involve end users in the design process, promoting design democratization.

The remainder of this paper is organized as follows. In Section 5.2 and 5.3, we provide an overview of the background related to data-driven 2D-to-3D generation and technical background about the applied techniques. Section 5.4 outlines the methodology in the development of our Image2CADSeq model. Subsequently, Sections 5.5 and 5.6 present and analyze the experimental results, summarizing the primary findings and acknowledging limitations. Conclusions and closing remarks are presented in Section 5.7, where we present key insights and suggest potential directions for future research.

5.2 Literature Review

In this section, we first provide a background of deep learning techniques applied to 2D-to-3D generation using discrete 3D representations, such as voxels and meshes, as well as the generation of CAD models using constructive solid geometry (CSG) and 2D sketches. Following this introduction, we then present a review of deep learning methods specifically tailored for 3D parametric CAD models, which are most relevant to our work.

5.2.1 Research on 2D-to-3D Generation

Our research is related to the domain of 2D-to-3D generation, led by advances in computer graphics and computer vision. For an in-depth understanding of the current developments and challenges in this area, we direct readers to a comprehensive review (Shi et al., 2022). The use of discrete 3D representations, such as voxels, point clouds, and meshes, has been prevalent. Despite their widespread use, these representations often focused on generating visually appealing objects without necessarily considering their engineering aspects, such as dimensions, engineering performance, and compatibility with engineering software (Li et al., 2023b). This mismatch hinders the seamless integration with downstream applications, such as editing and engineering analysis of synthesized 3D shapes, underlining the necessity for adopting CAD-specific data formats in our research.

Constructive solid geometry (CSG) is one of the fundamental methods for creating CAD models. It applies Boolean operations (e.g., union, intersection, and subtraction) to basic geometric shapes (primitives), such as cuboids, spheres, and cylinders. CSG is known for its lightweight structure, which allows for easy modifications by altering the parameters of these primitives and their spatial transformations. There have been various studies on the deep learning methods of CAD model generation using CSG (Ren et al., 2021; Sharma et al., 2017, 2019; Kania et al., 2020). Despite its merits, CSG lacks the versatility used in contemporary CAD tools that utilize parametric modeling.

Parametric CAD models start as 2D sketches comprising geometric primitives (e.g., line segments and arcs) with explicit constraints, such as coincidence and perpendicularity, establishing the foundation for 3D construction operations (e.g., extrusion and revolution). The relevant deep learning methods of parametric CAD models include 2D engineering sketch and 3D model generation and reconstruction. There has been a series of studies recently (Willis et al., 2021a; Seff et al., 2021; Ganin et al., 2021; Para et al., 2021; Yang and Pan, 2022) dedicated to the generation of CAD sketches through the application of deep learning approaches. The emphasis in these works is on generating 2D layouts rather than dealing with the generation of 3D components. We will be focused on introducing deep learning methods for 3D CAD model generation and reconstruction since they are more relevant to our work.

5.2.2 Deep Learning of Parametric 3D CAD Models

Boundary representation (B-rep) format is the standard format for representing 3D shapes in CAD, which defines objects based on their boundary surfaces, edges, and vertices connected through specific topology. Numerous learning-based approaches have emerged for the generation of parametric curves (Wang et al., 2020) and surfaces (Sharma et al., 2020). In addition to curve or surface generation, Smirnov et al. (Smirnov et al., 2020) introduced a generative model for creating topology that combines parametric curves and surfaces to create solid models, depending on predefined topological templates. In addition, various methods have been proposed to enable the direct generation of B-rep models with arbitrary topology (Guo et al., 2022; Jayaraman et al., 2022; Wang et al., 2022b). Different from these works, our focus lies in generating CAD sequences that can be translated into B-rep models using a solid modeling kernel, such as Fusion 360 CAD software.

Significant progress has been made in the generation of CAD sequences for the reconstruction of 3D models particularly through *Sketch-and-Extrude* modeling operations. Recently, there have been methods (Wu et al., 2021; Xu et al., 2022) for generative models specifically designed for the unconditional generation of CAD sequences. These models aim to autonomously create CAD sequences without relying on specific conditions or inputs. Specifically, Wu et al. (Wu et al., 2021) presented the first generative model, DeepCAD, that learns from sequences of CAD modeling operations to produce editable CAD designs. By drawing an analogy between CAD operations and natural language, the authors propose to utilize a transformer (Vaswani et al., 2017) architecture aiming to leverage the capabilities of transformer models in understanding and generating sequences, adapting them to the context of CAD design operations.

Generative models indeed serve as valuable tools for randomly generating a multitude of designs, offering inspiration and exploration of diverse possibilities. However, these models lack the capability to directly incorporate designers' intent into the generation process. Consequently, the designs generated can deviate from the expectations or specific requirements of the designers. This discrepancy highlights the need for mechanisms that allow designers to guide or influence the output, ensuring that the generated designs align more closely with their intent and preferences. To that end, several methods have been introduced to allow the CAD sequence generation given the target of B-rep models (Willis et al., 2021b; Xu et al., 2021), voxels (Lambourne et al., 2022; Li et al., 2023a), point clouds (Uy et al., 2022; Ren et al., 2022), and sketches (Li et al., 2020b, 2022a). Particularly, Fusion 360 Gym (Willis et al., 2021b) was developed to reconstruct a CAD model given a B-Rep model, utilizing a face-extrusion technique that relies on existing planar faces within the B-Rep model. However, despite the potential for CAD sequence generation, the face-extrusion method differs significantly from the more natural sketch-extrusion method commonly used by human designers. Moreover, this technique is ineffective when confronted with a lack of available planar or profile data in the input data, such as images.

Our work aims to fill a research gap in the existing literature by focusing on the task of generating CAD sequences from images. In particular, we expand upon the transformer-based autoencoder initially introduced in DeepCAD (Wu et al., 2021) and convert it into a TEVAE architecture developed in our previous work (Li et al., 2022c). In the case study, we apply the domain-specific language of Fusion 360 Gym (Willis et al., 2021b) that demonstrates our approach to predicting CAD sequences that involve *Sketch-and-Extrude* operations.

5.3 Technical Background

An autoencoder (AE) is a type of neural network that aims to learn a compressed representation of input data (Hinton and Salakhutdinov, 2006). It consists of two main parts: an encoder and a decoder. The encoder compresses the input into a lower-dimensional latent space, while the decoder reconstructs the input data from this compressed representation. The goal is to minimize the difference between the original input and its reconstruction, leading to efficient data encoding. Variational Autoencoders (VAEs) (Kingma and Welling, 2014) extend traditional AEs by introducing a probabilistic way to the encoding process. This probabilistic approach allows VAEs not only to reconstruct input data but also to generate new data that is similar to the input. Autoencoders have been applied to uncover the useful underlying structures of data which is typically known as representation learning (Bengio et al., 2013).

Representation learning is a set of techniques in machine learning that automatically discovers the representations with reduced dimensionality from raw data for downstream tasks, such as regression or classification (Khastavaneh and Ebrahimpour-Komleh, 2019). While VAEs are particularly known as generative models for their effectiveness in generating complex data, they offer a more robust, regularized, and probabilistic approach to representation learning compared to traditional AEs (Li et al., 2022c; Gomari et al., 2022).

Although most research has focused on employing AEs and VAEs in unsupervised or semi-supervised scenarios, it is worth noting that autoencoders also demonstrate utility in supervised contexts (Jarrett and van der Schaar, 2020). Specifically, incorporating an auxiliary feature-reconstruction task proves beneficial in enhancing supervised classification problems (Le et al., 2018), which are known as feature-embedding autoencoders (FEAs). Furthermore, Jarrett and Schaar (Jarrett and van der Schaar, 2020) propose target-embedding autoencoders (TEAs) and demonstrate their effectiveness theoretically and empirically. As implied by their names, TEAs focus on encoding the target's information into a latent space, while FEAs encode the feature's information.

TEAs have been applied to various problems. Girdhar et al. (Girdhar et al., 2016) introduce a TL-embedding network, comprising a T-network (an autoencoder network for the horizontal bar and an encoder for the vertical bar) during training.

Upon training completion of the T-network, it facilitates the derivation of the L-network, enabling the prediction of 3D voxel shapes from input images. A similar network architecture has also been applied for semantic image segmentation tasks (Mostajabi et al., 2018; Dalca et al., 2018). Drawing inspiration from these preceding studies, Li et al. (Li et al., 2022c) propose to use a VAE to replace the AE and form a target-embedding variational autoencoder (TEVAE) architecture, demonstrating its effectiveness in predictive and generative tasks for car and mug design examples.

In this study, we constructed the Image2CADSeq neural network model by comparing both TEA and TEVAE architectures. Our objective is to assess their effectiveness in the prediction task of images to CAD sequences. The results revealed a significant superiority of the TEVAE architecture over the TEA architecture in terms of prediction performance as detailed in Section 5.5.3.

5.4 Methodology

The flowchart depicted in Figure 5.1 illustrates the proposed systematic approach to predicting CAD sequences given images, a process we refer to as Image2CADSeq. To effectively tackle this challenge, we initiate with a clear problem definition that divides the task into manageable components. Our objective is to harness the power of deep learning to predict a CAD sequence—a series of CAD operations characterized by specific operation types and their corresponding parameters—from an image. The image could be a rendering from a CAD model or a real-world photograph of a 3D object.

Due to the intricate nature of CAD sequences, we employ a CAD program as a representational tool for CAD sequences. A CAD program enables designers to script their designs programmatically in a specialized scripting environment, such as the Fusion 360 API, FreeCAD API, or CADQuery. CAD programs can convert CAD sequences described in text into script language that is interpretable and executable by computers. To overcome the inherent lack of structured format in the

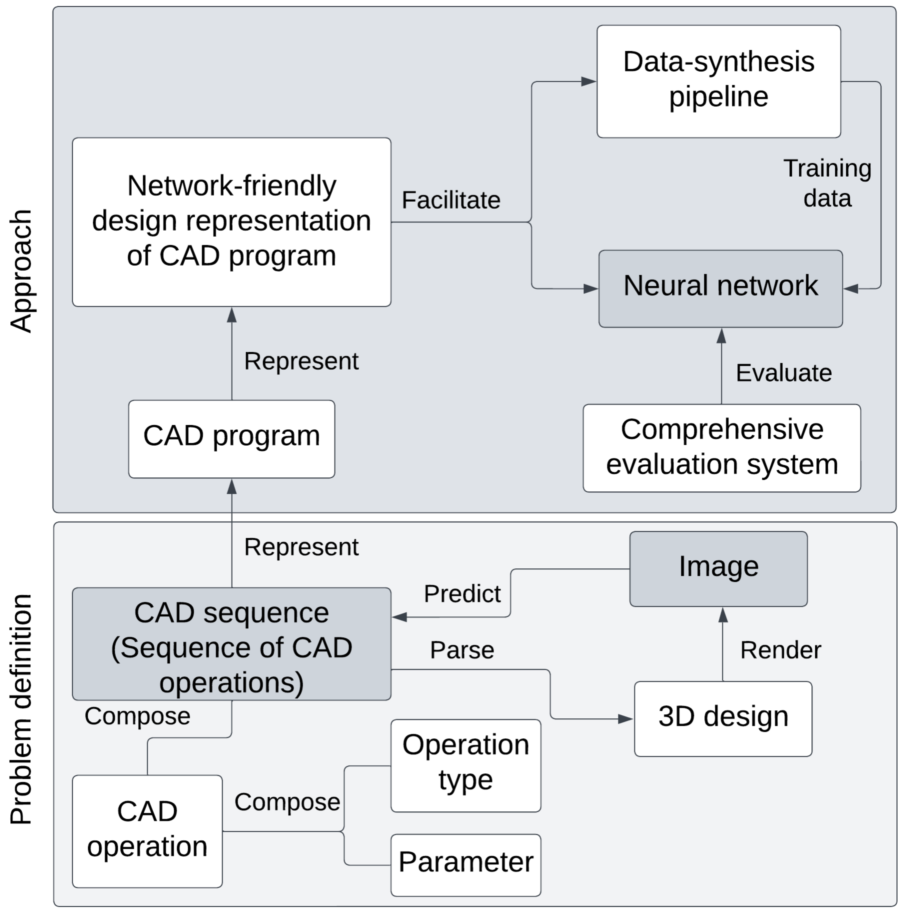


Figure 5.1: Approach overview.

Table 5.1: Fusion 360 Gallery Domain Specific Language

Operation type			**Operation parameters**	**Function with parameters**
Sketch	Sketch Plane		identifier (I): "XY", "XZ" or "YZ", etc.	add_sketch(I)
	Curves	Line	start point: (x_1, y_1); end point: (x_2, y_2)	add_line(N, N, N, N)
		Arc	start point: (x_1, y_1); center point: (x_c, y_c); sweep angle (α)	add_arc(N, N, N, N, N)
		Circle	center point: (x_c, y_c); radius: r	add_circle(N, N, N)
Extrude			profile id ([I]); distance: d; operation (O): NewBody, Cut, etc.	add_extrude([I] N, O)

CAD sequence data, the CAD program is then streamlined into a vectorized representation conducive to neural network processing. This representation can facilitate not only the development of our neural network's architecture but also the creation of a data-synthesis pipeline tasked with generating the training data for the neural network. In addition, given the complexity of the Image2CADSeq task, we develop a comprehensive evaluation system that rigorously assesses our neural network models' performance, thereby ensuring the reliability and accuracy of our approach. This holistic evaluation is crucial in refining the Image2CADSeq model and guiding its evolution to meet the demanding standards of CAD sequence prediction.

In this study, we employ a particular domain-specific language (DSL), namely Fusion 360 Gallery (abbreviated as Gallery for conciseness) (Willis et al., 2021b) for the CAD program, as a solid case to demonstrate our approach. Therefore, before the presentation of how we devise the vectorized design representation for the CAD sequence data, we provide an introduction to the Gallery DSL below.

Table 5.2: Comparison of the CAD programs for creating a cylinder with a base circle radius of 5 and a height of 10 using Fusion 360 Python API, Fusion 360 Gallery DSL, and the Simplified Gallery DSL. Note: Bullet numbers are used to indicate the effort that a designer would typically make to create a design.

	Fusion 360 Python API	Fusion 360 Gallery DSL	Simplified Gallery DSL
API setup	1. app = adsk.core.Application.get() 2. ui = app.userInterface 3. product = app.activeProduct 4. design = adsk.fusion.Design.cast(product) 5. rootComp = design.rootComponent	1. client = start_client()	N/A
Create a new sketch on the XY plane	1. sketches = rootComp.sketches 2. xyPlane = rootComp.xYConstructionPlane 3. sketch = sketches.add(xyPlane)	**1. r = client.add_sketch('XY')** 2. response_json = r.json() 3. response_data = response_json["data"] 4. sketch_name = response_data["sketch_name"]	1. add_sketch('XY')
Draw a circle in the sketch	1. circles = sketch.sketchCurves.sketchCircles 2. centerPoint = adsk.core.Point3D.create(0, 0, 0) 3. circles.addByCenterRadius(centerPoint, radius=5)	**1. r = client.add_circle(sketch_name=sketch_name, pt1={"x": 0, "y": 0}, radius=5)** 2. response_json = r.json() 3. response_data = response_json["data"]	1. add_circle(sketch_name='sketch1', pt1={"x": 0, "y": 0}, radius=5)
Get the profile	1. prof = sketch.profiles.item(0)	1. iterator = iter(response_data["profiles"]) 2. profile_id = next(iterator)	N/A
Create an extrusion	1. extrudes = rootComp.features.extrudeFeatures 2. extInput = extrudes.createInput(prof, adsk.fusion.FeatureOperations.NewBodyFeature Operation) 3. distance = adsk.core.ValueInput.createByReal(10) 4. extInput.setDistanceExtent(False, distance) 5. extrudes.add(extInput)	**1. r = client.add_extrude(sketch_name=sketch_name, profile_id=profile_id, distance=10, operation="NewBodyFeatureOperation")**	1. add_extrude(sketch_name='sketch1', profile_id=1, distance=10, operation="NewBodyFeatureOperation")

5.4.1 Fusion 360 Gallery Domain Specific Language

Table 5.1 presents a summary of the core elements (i.e., CAD-related elements) in Gallery DSL, which enables the representation of a 2D/3D design as a CAD program, and Python is used to implement the CAD operations (Willis et al., 2021b). Gallery DSL now supports two major types of CAD operations: *Sketch* and *Extrude*. Each CAD operation is decomposed into two fundamental components: the operation type and its corresponding parameters. These elements are analogously mirrored in the Gallery DSL as function names and their associated parameters.

A *Sketch* operation includes the definition of a *Sketch Plane* and *Curves* on it. A *Sketch Plane* can be created by the $add\_sketch(I)$ function, where I is a plane identifier that can be specified from the three canonical planes "XY", "XZ", or "YZ" or other planar faces (e.g., the side face of a cube) present in the current geometry. The *Sketch Plane* can then be used as the reference coordinate system in 2D for specifying the coordinates. A sequence of *Curves*, including *Line*, *Arc*, or *Circle*, can be drawn using $add\_line(N, N, N, N)$, $add\_arc(N, N, N, N, N)$, and $add\_circle(N, N, N)$, respectively, Where N is a real number representing the required parameters for a particular operation. As two numbers are needed to define one point, *Line* uses four numbers for start and endpoints; *Arc* needs five numbers: start point, center point, and sweep angle; and *Circle* is specified with three numbers: two for position and one for radius. Executing a *Sketch* can result in enclosed regions, termed profiles in CAD language. An *Extrude* operation can extrude a profile from 2D into 3D by using $add\_extrude(I, N, O)$, where I is an identifier for the profile, and N is a signed number defining the depth of extruding along the normal direction of the profile. The Boolean operation (O) specifies the behavior of the extruded 3D volume, e.g., add to or subtract from other 3D bodies.

While Gallery DSL currently does not support certain objects, such as spheres and springs, it still covers a vast range of them by using expressive *Sketch* and *Extrude* operations with Boolean capability (Willis et al., 2021b). Therefore, it is a good starting point for supporting learning-based methods (Willis et al., 2021b; Wu et al., 2021). In the future, the other CAD operations, such as *Revolve*, *Sweep*, and *Fillet*, can be added to the Gallery DSL to expand its design grammar for a full-fledged CAD tool. In this study, we leverage the current status of the Gallery DSL by only considering the *Sketch* and *Extrude* operations.

Gallery DSL acts as a simplified interface to the more complex Fusion 360 Python API. In essence, it democratizes access to sophisticated CAD design through a more intuitive Python-based interface, effectively bridging the gap between complex CAD operations and the user's ability to execute them efficiently. For instance, as demonstrated in Table 5.2, when creating a cylinder with a base circle radius of 5 and a height of 10, the CAD program using Gallery DSL requires only about two-thirds of the efforts needed with the Fusion 360 Python API and does not require sophisticated definitions of various variables. This makes the Gallery DSL code easier to operate, and more user-friendly and accessible. However, Gallery DSL still requires a substantial amount of coding efforts in non-CAD-related elements, beyond the core functions as listed in Table 5.1.

Table 5.3: Variables $t, I, x, y, \alpha, r, [I], d, O, s$ for the vectorized design representation of Gallery DSL

		Variable	Characteristic	Value range
Operation type		t	Discrete	{0, 1, 2, 3, 4, 5, 6}
Operation parameters	Identifier of sketch plane	I	Discrete	{0, 1, 2}
	End point x	x	Continuous	[-1, 1]
	End point y	y	Continuous	[-1, 1]
	Sweep angle	α	Continuous	[-1, 0) or (0, 1] (*180)
	Radius	r	Continuous	(0, 1]
	Identifier of profile	$[I]$	Discrete	{0, 1, 2, …}
	Extrusion distance	d	Continuous	[-1, 0) or (0, 1]
	Boolean operations	O	Discrete	{0, 1, 2, 3}
Auxiliary factor	Scale factor	s	Discrete	0~255 (e.g., 10)

To further improve its readability and accessibility, we simplify the Gallery DSL by isolating its key CAD-related functions referred to as Simplified-Gallery DSL or Sim-Gallery DSL for briefness, as shown in Table 2. We create the Sim-Gallery DSL following the concept of parametric modeling. In parametric modeling, a design is a sequence of operations that progressively modify the current geometry of an object. This process can be well represented in the Sim-Gallery DSL by a series of pure CAD operation functions. This simplification reduces the process to just three steps for creating a cylinder: $add\_sketch(\cdot)$, $add\_circle(\cdot)$, and $add\_extrude(\cdot)$. Additionally, we have developed a parsing method in Python to convert this simplified version back to the standard Gallery DSL. This ensures compatibility with Fusion 360 CAD software, facilitating seamless integration and execution of the CAD programs written in the Sim-Gallery DSL. It can also facilitate the design of the vectorized representation for the CAD sequence data as introduced in Section 5.4.2.

5.4.2 Design Representation of CAD Programs

A standardized design representation is essential for neural networks to effectively interpret CAD programs. Thus, it becomes crucial to devise an efficient method to represent each CAD operation and the entire CAD program. There are three major challenges:

1. Diversity in CAD operations: Different CAD programs comprise varying numbers of operations.
2. Variability in parameters: Different CAD operations involve different numbers of parameters.
3. Type of parameters: Parameters can be either continuous or discrete values.

To tackle these challenges, we propose to use a design representation with a unified data structure. We identified 10 variables $(t, I, x, y, \alpha, r, [I], d, O, s)$ from the Sim-Gallery DSL, detailed in Table 5.3. In what follows, we elaborate on our approach to handling these variables.

(1) $t \in \{0, 1, 2, 3, 4, 5, 6\}$ represents the operation types with $0 - 4$ representing *add_sketch*, *add_line*, *add_arc*, *add_circle*, and *add_extrude*. The values 5 and 6 are used to represent the start (SOP) and the end (EOP) of a CAD program, which are not typical CAD operations but are included for the learning process to indicate a complete CAD program as required by a transformer model (Vaswani et al., 2017; Carlier et al., 2020; Wu et al., 2021).

(2) $I \in \{0, 1, 2\}$ indicates the *Sketch Plane* using one of the canonical planes: "XY", "XZ", or "YZ".

(3,4) x and y are the coordinates of the endpoint for *Line* and *Arc*, while they represent the center point when the operation type is *Circle*. We excluded the start point required by *Line* and *Arc* from the design representation by obtaining it from the precedent curve to make sure all curves are connected one after another, making the vectorized representation more compact. There are two extra considerations for this setting: (i) If one curve has no precedent, we default its start point to the origin $(0, 0)$ when parsing the design representation. (ii) For *Arc* that requires a center point instead of an endpoint, we calculate the coordinates for the center point based on its start point, endpoint, and sweep angle.

(5) α represents the sweep angle of an *Arc*.

(6) r is the radius of a *Circle*.

(7) $[I]$ represents the profile index in the *Sketch*.

(8) d represents the signed distance of the depth for *Extrude*.

(9) $O \in \{0, 1, 2, 3\}$ is used to indicate the Boolean operations: join, cut, intersect, or add, respectively.

(10) s is an auxiliary factor that can be used to scale a CAD model.

In addition, to standardize the treatment of both continuous and discrete parameters, inspired by (Carlier et al., 2020; Wu et al., 2021), we discretize continuous parameters through quantization. This involves: (a) Confining continuous values to a subset of $[-1, 1]$ (e.g., $(0, 1]$ for radius and $[-1, 1]$ for endpoint x and y; (b) Dividing each range into 256 equal segments, enabling representation as 8-bit integers (i.e., $0 - 255$); (c) For the sweep angle (α), we multiply it by 180 during interpretation; (d) Handling scale factor (s): Although the scale factor can be a non-negative continuous value, we limit it to 256 levels for consistency with other continuous values' quantization. Consequently, the 10 variables can encode both the operation type and its associated parameters. From the 10 variables, a fixed-dimensional vector can be formalized as a unified design representation for each CAD operation, and the unused parameters will be filled with values of -1. See Figure 5.3 for an example.

The subsequent consideration involves standardizing CAD programs of varying sizes (the number of CAD operations involved). For example, besides the start and end marks, SOP and EOP, a cylinder can be created in three operations as shown in Table 5.2, while a triangular prism in five operations ($add\_sketch(\cdot)$, $add\_line(\cdot) \times 3$, and $add\_extrude(\cdot)$). To achieve a consistent data structure across all CAD programs, we introduce a treatment, called *maximum program length*. Then, CAD programs shorter than this maximum length are extended by appending end marks (EOP) until they reach the predetermined length.

In this study, we use 7 variables to construct a 7-dimensional vector $[t, I, x, y, \alpha, r, d]$ for each CAD operation (i.e., one step/line in a CAD sequence). Additionally, we assign default values to the other three variables $[I], O$, and s, setting them as 0, 3, and 10 correspondingly. Different variables can be selected which will influence the complexity of the data structure and thus the complexity of designs. In addition, the maximum length for CAD programs is set to 10. As a result, the design representation of a CAD program will be a matrix, namely, the feature matrix. Mathematically, the feature matrix denoted as P, is expressed as $P = \left[\mathbf{o^1}, \quad \mathbf{o^2}, \quad \ldots \quad, \mathbf{o^{N_c}}\right]^T \in \mathbb{R}^{10\times 7}$, where $\mathbf{o^i} \in \mathbb{R}^7$ is a CAD operation vector, and $N_c = 10$ is the sequence length of the CAD program. Refer to Figure 5.3 for an example of how a cylinder is converted to a feature matrix, including the quantization of its parameters as explained earlier.

5.4.3 Neural Network Model Architecture, Training, and Application

The application of target-embedding representation learning in deep learning, particularly for cross-modal tasks, has shown considerable efficacy (Jarrett and van der Schaar, 2020; Li et al., 2023b). In alignment with this, we have developed the Image2CADSeq model, utilizing a target-embedding representation learning method, as illustrated in Figure 5.2. It features an encoder-decoder network for Stage 1 (S1),

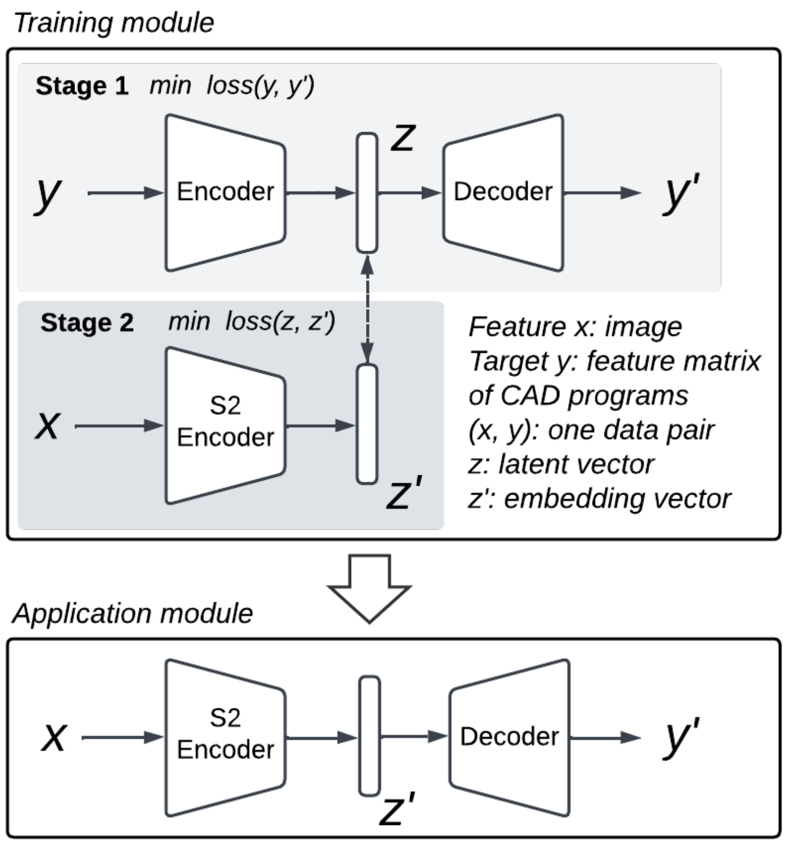


Figure 5.2: Image2CADSeq model using a target-embedding representation learning method

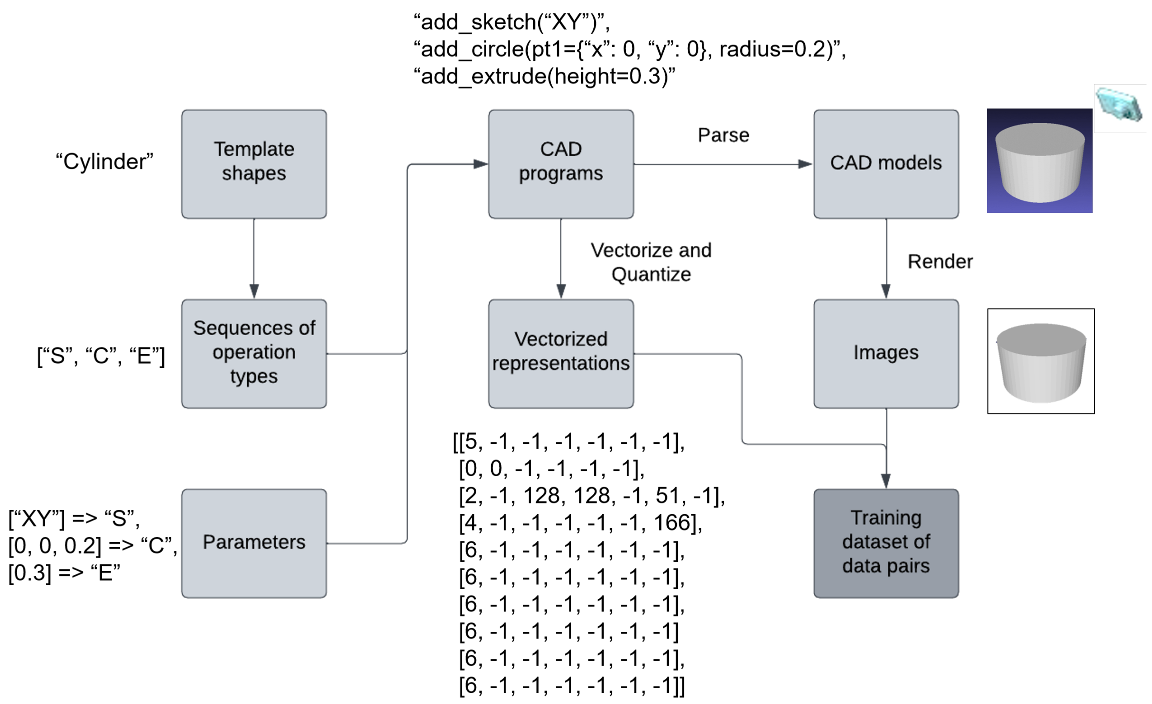


Figure 5.3: Synthesis pipeline for the training dataset of data pairs of image and vectorized CAD sequence, exemplified using a cylinder model

which is geared towards unsupervised learning and enables the efficient encoding of target objects (i.e., matrix feature of CAD programs) within a latent space. An additional encoder is integrated into Stage 2 (S2), focusing on supervised learning to regress the previously learned latent space using feature objects (i.e., images) as input.

The Image2CADSeq model employs a two-stage training strategy (Mostajabi et al., 2018; Li et al., 2022c). In Stage 1, the focus is on independent training of the encoder-decoder network. The objective is to minimize the reconstruction loss between the actual matrix feature of CAD programs (y) and its reconstructed equivalent (y'). Completing this stage involves fixing the learnable parameters of the neural network model and saving the learned model, thereby capturing a latent space of y. Stage 2 shifts the focus to independent training of the S2 encoder by minimizing the discrepancy between the latent vector, derived from the learned latent space, and the embedding vector produced by the S2 encoder using an image as input. Importantly, each image used in this stage is directly associated with its feature matrix from Stage 1. This image and its corresponding feature matrix are associated with the same 3D object, and they form one data pair. The alignment of the latent vector with the embedding vector is performed specifically for these data pairs, ensuring that the S2 encoder training is precisely tuned to the corresponding images. This approach ensures a cohesive and targeted learning process. We present a novel data synthesis pipeline to generate training data pairs in Section 5.4.4.

After training the Image2CADSeq model, the S2 encoder is integrated with the decoder from S1, creating the application module. This module is capable of predicting a feature matrix given an image input. Subsequently, this feature matrix can be translated into a CAD program using the Sim-Gallery DSL. Finally, the CAD program can be parsed into a 3D object by utilizing Fusion 360 software.

5.4.4 Data Synthesis Pipeline

With the proposed design representation of the CAD programs and Fusion 360 software, we introduce an automatic data synthesis pipeline, as illustrated in Figure 5.3. This method is tailored to generate training data pairs comprising feature matrices of CAD programs and the corresponding images, essential for training the Image2CADSeq model. The process begins with preparing a list of basic shape templates, such as cylinders, employing the *Sketch-and-Extrude* paradigm of the Gallery DSL. For these basic shapes, we establish a series of template operations (e.g., add_line, add_circle). The corresponding parameter values of these operations are then generated based on the range specified in Table 5.3. By integrating these template operations with their respective parameters, a complete CAD program is formulated, which is then translated into 3D CAD models through Fusion 360 software. These models are then rendered to obtain their images. Additionally, the CAD programs are vectorized and quantized to derive their feature matrices, as discussed in Section 5.4.2. An image paired with its feature matrix, both derived from the same CAD program, constitutes a data pair. The method is exemplified using a cylinder model in Figure 5.3.

5.4.5 Evaluation Metrics

In the Image2CADSeq task, there are three key elements: the image, the CAD program, and the 3D CAD model. The CAD program consists of a sequence of CAD operations, each involving an operation type and its associated parameters. To assess the effectiveness of our approach, we have developed a set of evaluation metrics, as illustrated in Table 5.4.

For the evaluation of 3D CAD models and images, we utilize established metrics such as the intersection over union (IoU) and mean squared error (MSE), respectively, as shown in Table 5.4(a). However, assessing the quality of CAD programs poses a challenge due to the scarcity of suitable metrics in the literature. To address

Table 5.4: Comprehensive evaluation metrics for the image, the CAD program, and the 3D CAD model

Table (a): Evaluation metrics for the 3D CAD models and images

Evaluation focus	Metrics	Interpretation
CAD programs to 3D models	Parsing rate	Successful rate of parsing the CAD programs to 3D models
Localization and reconstruction of 3D models	Intersection over union (IoU)	Given the successful parsing, the degree of similarity between the reconstructed 3D CAD models and the ground truth models
Image	Mean squared error (MSE)	The measurement of the average squared difference between the pixel values of the rendered image and ground truth image

Table (b): Evaluation metrics for the CAD programs

Evaluation focus		Metrics		Interpretation
		L1: Operation type	L2: Parameter	
H1: Sequence evaluation	The CAD program	Accuracy of the CAD programs (ACP)		The ratio of the predicted CAD programs that precisely align with the ground truth ones
	The sequence of CAD operation types	Accuracy of the sequence of operation types (ASOT)	-	The ratio of the predicted CAD operation type sequences that exactly match the ground truth ones
		Edit distance of the sequence of operation types (EDSOT)	-	The level of similarity between the predicted operation type sequence and the ground truth
H2: Sequence-based Operation type-wise evaluation	The CAD operation type	Accuracy of the operation types (AOT)	-	The proportion of CAD operation types in the predicted sequences that align with their corresponding operation types in the ground truth
		-	Accuracy of the parameter^1 (AP^1)	The agreement of associated parameters when the CAD operation type is correctly predicted considering the sequential order
H3: Set-based operation type evaluation	The multiset of CAD operation types	Multiset similarity of operation types (MSOT)	-	The similarity of predicted CAD operation types in a CAD program to those in the ground truth CAD program without considering the order
		-	Accuracy of the parameter^2 (AP^2)	The agreement of associated parameters when the CAD operation type is correctly predicted without considering the order

this gap, we introduce a novel evaluation system for CAD programs based on the proposed matrix representation, detailed in Table 5.4(b). To comprehensively evaluate the information loss between the predicted CAD programs with the ground truth, this system incorporates both hierarchical (H1-3) and double-layered (L1,2) aspects as shown below, facilitating a multi-dimensional assessment of CAD program prediction. We expect this evaluation system to become a standard for tasks involving the prediction of CAD programs, as further discussed in the following.

- H1: Sequence evaluation – Evaluates the accuracy of the entire CAD program and the specific order in which certain CAD operation types follow.
- H2: Sequence-based operation type evaluation – Examines the accuracy of each individual operation type within the sequence.
- H3: Set-based operation type evaluation – Assesses the operation types as a collective set, without considering the sequential order. Even if the operation type sequence varies, a prediction is considered superior if it accurately predicts a higher number of operation types due to the preservation of information.

- L1: Operation type layer – Evaluates the accuracy of the CAD operation types.
- L2: Parameter layer – Assesses the accuracy of the parameters associated with each CAD operation type.

Recall that a feature matrix P can be expressed as $P = \begin{bmatrix}\mathbf{o^1}, & \mathbf{o^2}, & \dots & ,\mathbf{o^{N_c}}\end{bmatrix}^T$. The vector $\mathbf{o^i} \in \mathbb{R}^7$ is a CAD operation vector which can be noted as $\mathbf{o^i} = \begin{bmatrix} t \\ \mathbf{p} \end{bmatrix}$, where t is an integer indicating the operation type, and $\mathbf{p}$ is a vector of integers representing the corresponding parameters (see Section 5.4 for more details). In what follows, we explain the evaluation metrics using the same notation.

ACP. Accuracy of CAD programs (ACP) is calculated by Equation (5.1), representing the ratio of the predicted CAD programs that are precisely aligned with the ground truth ones, where N is the total number of test data for evaluation, P^i denotes the ground truth CAD program, while $\hat{P}^i$ represents the corresponding predicted CAD program, $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the condition is true, and 0 otherwise.

$$\text{ACP} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(P^i = \widehat{P^i}), \tag{5.1}$$

ASOT & EDSOT. Two metrics are defined for evaluating the sequence of operation types: 1) accuracy of the sequence of operation types (ASOT) and 2) edit distance of the sequence of operation types (EDSOT). ASOT assesses the proportion of predicted CAD sequences with operation types (without considering the associated operation parameters) that match exactly the ground truth, as defined by Equation (5.2). The notation adheres to those introduced in Equation (5.1). In addition, $P^i[:,1]$ represents the first column of P^i which is the sequence of operation types in a CAD program, and similarly for $\hat{P}^i[:,1]$.

$$\text{ASOT} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(P^i[:,1] = \widehat{P^i}[:,1]), \tag{5.2}$$

$$M[i,j] = \begin{cases} \max(i,j) & \text{if} \min(i,j) = 0, \\ \min\begin{cases} M[i-1,j]+1 \\ M[i,j-1]+1 \\ M[i-1,j-1]+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \tag{5.3}$$

In the case of EDSOT, it measures the level of similarity between the predicted CAD operation type sequence and the ground truth. While there exist various metrics to calculate the edit distance, we utilize the Levenshtein distance as shown in Equation (5.3), which is commonly employed to compare sequential data in applications, such as computational biology (Navarro, 2001). Given two strings a and b of lengths m and n, respectively, the Levenshtein distance $L(a,b)$ can be calculated using dynamic programming. We define a matrix M of size $(m+1)\times(n+1)$, where $M[i,j]$ represents the minimum number of operations (i.e., insertions, deletions or substitutions) required to transform the substring $a[1:i]$ into the substring $b[1:j]$. After calculating the values for all entries of the matrix M, the Levenshtein distance is given by $L(a,b) = M[m,n]$.

AOT. As shown in Equation (5.4), the accuracy of the operation types, denoted as AOT, is computed as the proportion of CAD operation types in the predicted sequences that align with their corresponding operation types in the ground truth, taking into account the order. Common notations are used as in previous equations. In addition, the function $|\cdot|$ is employed to determine the length of a sequence, and l^i is defined as $\min(|P^i[:,1]|, |\widehat{P^i}[:,1]|)$, representing the number of operation types that need to be compared in a sequence for the ith data point of the test data for evaluation.

$$\text{AOT} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{l^i}\mathbb{I}(P^i[j,1] = \widehat{P^i}[j,1])}{\sum_{i=1}^{N}|(P^i[:,1])|} \tag{5.4}$$

AP^1. The accuracy of parameter^1 (AP^1) is determined by assessing the agreement of associated parameters when the CAD operation type is correctly predicted considering the sequential order, as defined in Equation (5.5). Conditions (c1-3) serve as the input criteria for the indicator functions. This metric function serves as the second layer beneath the first layer, AOT, indicating that parameter evaluation occurs exclusively when the operation type is accurately predicted (i.e., c1). For c2, recall our use of 8-bit integers (i.e., $0-255$) to represent the parameter values. Regarding c3, the parameter η denotes the permissible tolerance for differences between the predicted parameters and their ground truth values. For example, given a specific permissive tolerance $\eta \in [0,255]$, if a ground truth parameter value is $z \in [0,255]$, to be counted as a correct prediction, the predicted parameter value $\hat{z}$ must satisfy the following conditions: $|\hat{z}-z| \leq \eta$ and $\hat{z} \in [0,255]$. Furthermore, the summation over k is a consequence of each CAD operation vector $\mathbf{o} \in \mathbb{R}^7$ having its first dimension representing the operation type, while the subsequent dimensions (i.e., dimensions

2-7) pertain to the associated parameters.

$$
\begin{aligned}
&\mathrm{AP}^1 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{l^i}\sum_{k=2}^{7}(\mathbb{I}(\mathrm{c1})\cdot\mathbb{I}(\mathrm{c2})\cdot\mathbb{I}(\mathrm{c3}))}{\sum_{k=2}^{7}\sum_{i=1}^{N}|(P^i[:,1])|}\\
&\mathrm{c1}: P^i[j,1] = \widehat{P^i}[j,1]\\
&\mathrm{c2}: 0 \leq \widehat{P^i}[j,k] \leq 255\\
&\mathrm{c3}: |P^i[j,k] - \widehat{P^i}[j,k]| \leq \eta
\end{aligned}
\tag{5.5}
$$

MSOT. The multiset similarity of operation types (MSOT) is a metric that compares the similarity of predicted CAD operation types in a CAD program to those in the ground truth CAD program, without taking the sequential order into account. In mathematics, a set is defined as a collection of elements where the order of these elements is irrelevant and duplicate elements are not permitted. Conversely, a multiset follows a similar principle as a set, but it allows the inclusion of repeated elements. Thus, a set can be seen as a special case of multiset where each element occurs only once. To implement the MSOT, we adapted two commonly used metrics: Tanimoto coefficient (TC) and cosine similarity (CS) in the Cheminformatics field for carrying out molecular similarity calculations (Bajusz et al., 2015). As the order of the elements in a multiset is not concerned in this case, we can represent a multiset as a vector, where each element corresponds to the count of a particular element in the multiset. For instance, in this study, we have a universe of elements for all the operation types $\{0, 1, 2, 3, 4, 5, 6\}$, a multiset of a triangular prism $\{5, 0, 1, 1, 1, 4, 6\}$ can be represented by a vector $[1, 3, 0, 0, 1, 1, 1]$ with each number representing the count of a particular operation type. Denote $\mathbf{a}$ and $\mathbf{b}$ as the vectors of two multisets and the TC between $\mathbf{a}$ and $\mathbf{b}$ can then be calculated as Equation (5.6), where $\mathbf{a}\cdot\mathbf{b}$ denotes the dot product between the two vectors (sum of the element-wise multiplication), $||\cdot||$ denotes the Euclidean norm. CS measures the cosine of the angle between two vectors and CS between $\mathbf{a}$ and $\mathbf{b}$ is calculated as Equation (5.7).

$$\mathrm{TC}(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b})/(||\mathbf{a}||^2 + ||\mathbf{b}||^2 - \mathbf{a}\cdot\mathbf{b}) \tag{5.6}$$

$$\mathrm{CS}(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b})/(||\mathbf{a}|| \times ||\mathbf{b}||) \tag{5.7}$$

AP^2. Similar to AP^1, the accuracy of parameter^2 (AP^2) serves as the second layer in the evaluation of CAD operations which can be similarly calculated using Equation (5.5). Notably, in AP^2, the assessment does not take into account the order of operations. In this study, an operation type can occur multiple times in a CAD program, such as the *Line* operation in a triangular prism. This introduces a challenge regarding which instance of the *Line* operation in the predicted CAD program should be matched with the corresponding instance in the ground truth CAD program for parameter comparison. Despite this challenge, AP^2 retains practical significance, particularly in scenarios where there are no repeated elements in the CAD operations. In addition, it is essential to maintain AP^2 to preserve the integrity of the evaluation system.

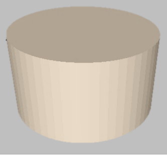
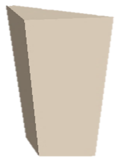
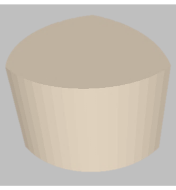
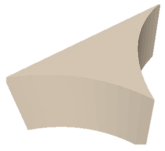
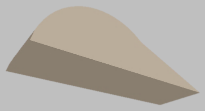
5.5 Experiments and Results

In this section, we introduce our experiments and results in detail. We begin with the introduction to training data preparation, followed by our experiments on different strategies of synthesizing datasets, different neural network architectures, and then, the results.

5.5.1 Training Data Preparation

Based on the data synthesis pipeline outlined in Section 5.4.4, we developed a collection of 5 template shapes (TS), as depicted in Table 5.5. These shapes are crafted using *Sketch-and-Extrude* operations, detailed in Table 5.1. To create the *Sketch* of each shape, we utilized *Line*, *Arc*, and *Circle* operations, and applied the *Extrude* operation to generate the 3D volume of these shapes. TS 1-3 each correspond to unique sequences of operation types, while TS 4 and 5 are associated with three varied sequences. An example for each template shape is also presented.

Table 5.5: Template shapes for the synthesis of training data

Template Shape (*Sketch*)+ *Extrue*	Template sequence of operation types	Example
TS 1 (*Circle*)	[“S”, “C”, “E”]	
TS 2 (Triple *Line*)	[“S”, “L”, “L”, “L”, “E”]	
TS 3 (Triple *Arc*)	[“S”, “A”, “A”, “A”, “E”]	
TS 4 (Double *Arc* + single *Line*)	[“S”, “L”, “A”, “A”, “E”] [“S”, “A”, “L”, “A”, “E”] [“S”, “A”, “A”, “L”, “E”]	
TS 5 (Single *Arc* + Double *Line*)	[“S”, “A”, “L”, “L”, “E”] [“S”, “L”, “A”, “L”, “E”] [“S”, “L”, “L”, “A”, “E”]	

We employ two distinct strategies for dataset preparation: (1) Random generation of CAD programs (i.e., dataset without rules): We assign random values to CAD operation parameters for the template sequences. This randomness is within predefined value ranges as outlined in Table 5.3. For example, for an $add\_line(\cdot)$ operation, we randomly select a number from the range $[-1, 1]$ to determine the coordinates of endpoints. This method ensures diversity in the dataset by incorporating a wide range of parameter values, reflecting various potential CAD designs. (2) Generation of CAD programs with embedded design rules (i.e., dataset with rules): Contrary to the random approach, this method involves parameter selection based on the design rules that we defined. For example, the extrusion depth of a circle is determined by the coordinates of its center point. This strategy mimics the purpose-driven process typical of real-world design scenarios. By incorporating design rules into data synthesis, we embed design knowledge in the data. This method is expected to test how the embedded design principles would affect the effectiveness of CAD reconstruction from images. The combination of random generation and rule-based design in dataset preparation allows for a comprehensive evaluation of our system's capabilities in varying scenarios.

Under each synthesis strategy, we synthesized 2,000 shapes corresponding to every sequence type outlined in Table 5.5, except for the sequence of TS 1, for which we synthesized 6,000 shapes. This was taken to ensure a balanced dataset in terms of both the length of sequences and the number of shapes for each template shape category. Consequently, this led to the creation of 22,000 CAD models for each strategy. These models were derived by processing the synthesized CAD programs through Fusion 360 software. For imaging purposes, all these 3D models were rendered using a uniform perspective camera positioned at $(20.0, 20.0, 20.0)$ looking towards the origin $(0.0, 0.0, 0.0)$ and all the images are in the resolution of 512×512 pixels. This process resulted in the image set $X = \{x_k\}_{k=1}^{22000}$. The corresponding feature matrices $Y = \{y_k\}_{k=1}^{22000}$ were also saved during synthesis. The final training dataset was $\{x_k, y_k\}_{k=1}^{22000}$.

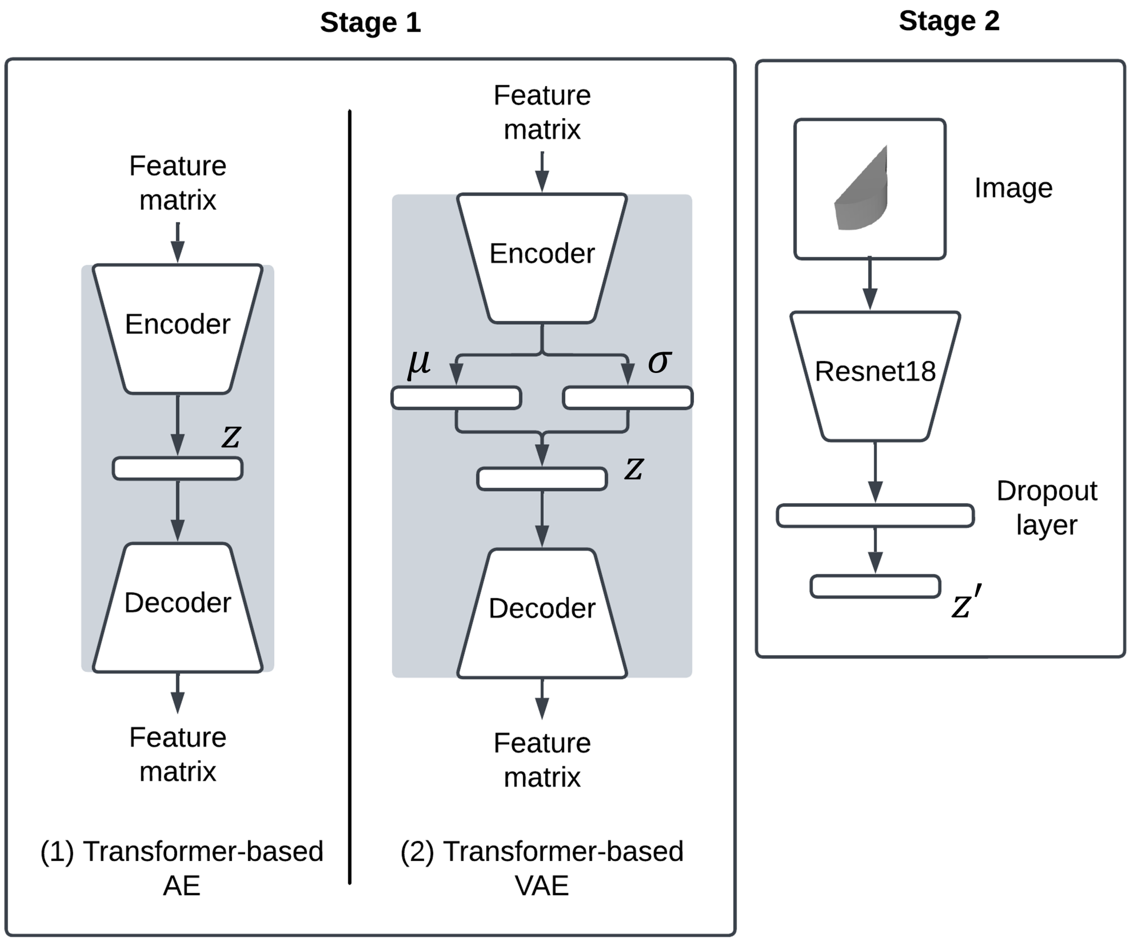


Figure 5.4: Implementation of the Image2CADSeq model. Two different encoder-decoder architectures were explored in Stage 1: (1) Baseline model: a transformer-based autoencoder (AE), adapted from DeepCADNet (Wu et al., 2021), and (2) Enhanced model: a transformer-based variational autoencoder (VAE), which extends the AE architecture. In Stage 2, the encoder is developed based on ResNet18 (He et al., 2016), employing a dropout layer before the final layer to mitigate overfitting and enhance generalization.

5.5.2 Image2CADSeq Model Architecture and Training

In Figure 5.4, we illustrate the development of the Image2CADSeq model. The construction of the model involved the application of target representation learning techniques, commonly employing target-embedding autoencoders (TEA) that utilize an autoencoder to obtain the latent representation of target objects (Jarrett and van der Schaar, 2020). In a recent development, Li et al. (Li et al., 2022c) introduced a target-embedding variational autoencoder (TEVAE) by extending the autoencoder to a variational autoencoder (VAE). This approach has demonstrated superior effectiveness in the cross-modal synthesis of 3D designs. However, their study did not include an empirical comparison between the two architectures in terms of the generated results. Therefore, as part of our study's contributions, we investigated two models: 1) a baseline transformer-based AE and 2) an improved version in the form of a transformer-based VAE for Stage 1. The aim was to compare their performance and explore a better architecture for the Image2CADSeq model.

We modified the first and last layer of the transformer-based AE, adapted from DeepCAD (Wu et al., 2021), to capture and reconstruct the feature matrices of the CAD programs in this study. Its primary objective is to learn and interpret the latent space of these matrices, providing a robust foundation for accurate feature representation. The AE model is used to establish a baseline for understanding and processing the complex structures inherent in CAD designs. It uses a typical reconstruction loss coupled with a regularization loss. Reconstruction loss ensures accurate reconstruction of input features in the output, while regularization loss prevents overfitting, promoting a more generalized model capable of handling various CAD designs.

Extending the AE architecture, the VAE introduces a probabilistic approach to encoding, which is tailored to construct a smoother latent space, surpassing the AE in terms of flexibility and adaptability. The VAE employs a more complex loss function with KL-divergence loss, reconstruction loss, and regularization loss. The

KL-divergence loss is pivotal in managing the probabilistic aspect of VAE, ensuring that the encoded distributions are effectively regularized. This, along with the reconstruction and regularization losses, forms a comprehensive approach to learning, capturing both the variance and the intricate details of CAD sequences. Stage 2 of the development incorporates an encoder based on ResNet18 (He et al., 2016). In addition, a dropout layer is positioned between the encoder and the embedding vector layer to prevent overfitting to the training data, thus maintaining its efficacy on unseen data. We utilized a regression loss between the embedding vector of the image and its corresponding latent vector obtained from the latent space in Stage 1 and a regularization loss to promote the generalizability of the Stage 2 encoder.

We divided each of the two training datasets, i.e., the dataset with or without rules, into three subsets: train, validation, and test set with a proportion of 8:1:1. The validation set was used to monitor the training process, preventing the model from overfitting to the train set data and ensuring that the model's generalization capabilities to the unseen data, e.g., test set data. We employed a grid search strategy in Stage 1 to find optimal hyperparameters of the neural network models and the training, aiming to minimize the training loss while maintaining good generalizability of the models. The AE and VAE models exhibited similar training trends, leading us to use the same hyperparameter set for both models. In our experiments, for Stage 1, a latent dimension size of 256 proved optimal for both models, resulting in the lowest reconstruction loss for the test set data among trials with dimensions of 64, 128, 256, and 512. Other hyperparameters include 500 epochs of training with a batch size of 512, the Adam optimizer, and a learning rate of 0.001. Moving to Stage 2, we initiated training with a pre-trained ResNet18 model (He et al., 2016) that possesses a broad comprehension of various images. The S2 encoder was trained for 50 epochs using the Adam optimizer with a learning rate of 0.0001 and a batch size of 128. A dropout ratio of 0.4 was applied in the dropout layer.

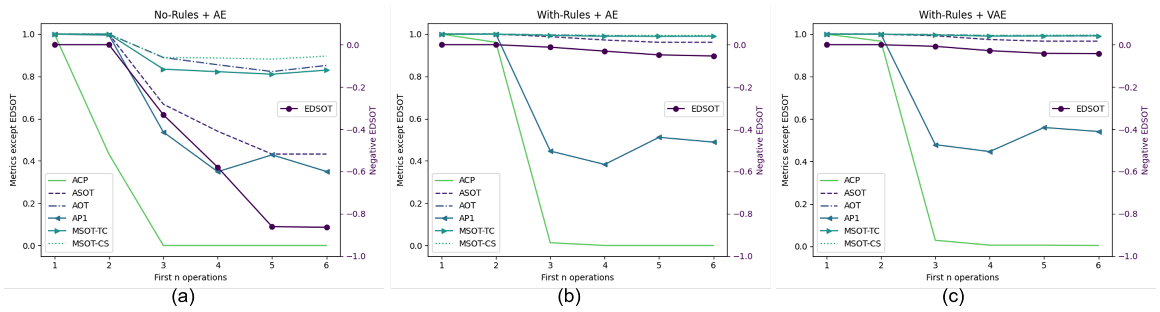


Figure 5.5: Evaluation of the Image2CADSeq model's performance using two distinct architectures with two datasets. (a) Case 1: Results from the network utilizing the TEA architecture trained on the dataset without rules. (b) Case 2: Results from the same TEA architecture but trained on the dataset with rules. (c) Case 3: Results using the TEVAE architecture trained with the dataset with rules. Each figure illustrates the variation of the network's performance metrics (shown in Table 5.4) versus the first n CAD operations in a CAD program. Specifically, a tolerance $\eta = 3$ is chosen for metrics that involve the calculation of the accuracy of parameters, such as ACP and AP^1.

5.5.3 Results

In this section, we present the results of our experiments on the performance of the Image2CADSeq model.

5.5.3.1 Overall Evaluation of the CAD Programs

Figure 5.5 provides a comparison of the Image2CADSeq model's performance, evaluated under two different architectures and two datasets. Specifically, there are three subfigures, each representing a unique combination of architecture and dataset. Figure 5.5 (a) illustrates the performance of the network when employing the TEA architecture in conjunction with the dataset without rules. In contrast, Figure 5.5 (b) presents results derived from the same TEA architecture, but the network is trained on the dataset with rules. We observed a significant improvement when employing the dataset with rules for model training. Consequently, we tested the TEVAE architecture using the dataset with rules only, and the results are presented in Figure 5.5 (c).

The figures illustrate the model's performance across various metrics (as defined in Table 5.4 (b)) when applied to the first n operations in a CAD program. We limit n to 6 to encompass the longest template sequences. According to Table 5.5, the maximum length of the template sequences is 5 for CAD operations in addition to a non-CAD operation SOP, marking the start of a program. Typically, higher metric values indicate superior performance, except for the EDSOT, where lower values are preferable. To maintain a uniform direction of performance across all metrics and enhance the readability of the plotting, we present the EDSOT in its negative form in the figures. Furthermore, when calculating metrics related to parameter accuracy, such as the accuracy of CAD programs (ACP) and the accuracy of parameter^1 (AP^1), we introduce a tolerance level ($\eta = 3$). This tolerance accounts for permissible deviations in the quantized continuous variables that have 256 levels but does not extend to discrete variables, such as the sketch plane identifier that has only 3 levels (refer to Table 5.3). The tolerance reflects the design problem's criteria, allowing certain margins of error in parameter predictions that can be customized in different scenarios.

The metrics, classified into three hierarchical categories in Section 5.4.5, include the accuracy of CAD programs (ACP), the accuracy of the sequence of operation types (ASOT), and the edit distance of the sequence of operation types (EDSOT) for H1 sequence evaluation; the accuracy of the operation types (AOT) and the accuracy of parameter^1 (AP^1) for the evaluation of the H2 sequence-based operation type; and the multiset similarity of operation types (MSOT) for the evaluation of the H3 set-based operation type. We analyze these results according to this hierarchical structure.

Upon analyzing the results of Figure 5.5 (a) for the TEA trained using the dataset without rules (referred to as Case 1), we observe a downward trend in all metrics as the sequence length increases. This is intuitive, and predicting longer sequences is inherently more difficult for the model. The ACP metric drops to zero at $n = 3$, indicating that the model struggles to accurately predict the entire CAD program including both the operation types and parameters even when the sequence is relatively short. Notably, the ACP's decrease to roughly 0.4 at $n = 2$ suggests the model's specific difficulty in predicting the sketch plane given the input images, because the second CAD operation—*Sketch*—defines the sketch plane's position. In addition, both ASOT and the negative EDSOT metrics exhibit declines beginning with $n = 3$, together with the results of ACP, showing the limited sequence prediction capability of the model when the sequences get longer.

Moreover, despite the model's low values in H1 metrics and the low values in AP^1 of H2, it scores highly on the AOT metric of H2 and maintains high values of MSOT-TC and MSOT-CS in H3. This discrepancy and inconsistency probably result from the characteristics of the dataset, in which different shape categories share similar CAD sequences (see Figure 5.5). For example, a ground truth (GT) sequence of TS 3, $[\text{“}S\text{”}, \text{“}A\text{”}, \text{“}A\text{”}, \text{“}A\text{”}, \text{“}E\text{”}]$, could be mistakenly predicted as a TS 4 sequence, e.g., $[\text{“}S\text{”}, \text{“}L\text{”}, \text{“}A\text{”}, \text{“}A\text{”}, \text{“}E\text{”}]$ by the model. Although such a prediction would be deemed as an incorrect sequence prediction when evaluated against ASOT, it would score well (i.e., 4 correct and 1 incorrect operation type) in terms of AOT due to the correctly predicted operation types. A dataset encompassing a broader array of design objects with diverse sequences of CAD operations might mitigate such discrepancies in the metric values, such as real-world designs collected from human designers. More significantly, it underscores the need for a comprehensive evaluation framework for image-to-CAD sequence prediction, ensuring that models are thoroughly assessed from multiple perspectives. Otherwise, the results of the performance of the models might be biased.

In Figure 5.5 (b), we show the performance of the TEA architecture trained on the dataset with rules (referred to as Case 2), in contrast to the earlier results in Figure 5.5 (a). In particular, this treatment improves significantly in most metrics, except for ACP and AP^1. These exceptions, however, do not overshadow the overall enhancement in the model's ability to predict sequences accurately. Specifically, while ACP does not show a significant improvement, its slower rate of decline

Table 5.6: Results when evaluated at the first 6 operations of the CAD programs

	AE no-rules	AE with-rules	VAE with-rules
ACP	0.000	0.000	**0.004**
ASOT	0.432	0.961	**0.967**
AOT	0.852	0.991	**0.993**
AP1	0.350	0.489	**0.541**
MSC-TC	0.830	0.990	**0.992**
MSC-CS	0.896	0.994	**0.996**
EDSOT (-)	-0.864	-0.053	**-0.042**

at $n = 2$ and 3 indicates an improvement in the model's performance of predicting sketch planes. Furthermore, although AP^1 does not show a significant improvement, it convergences to a higher value than the previous data treatment, suggesting an improved performance in parameter prediction.

The significant differences between Figures 5.5 (a) and (b) highlight the positive impact that design rules can have on the performance of the model in Image2CADSeq predictions. However, despite the improvement when including design rules, the model still faces challenges in accurately predicting parameters.

Building upon the insights from the first two cases, we evolved our model from TEA to TEVAE and trained it using the dataset with rules (referred to as Case 3). The results of the TEVAE model are detailed in Figure 5.5 (c). It displays high accuracy in most metrics similar to the baseline performance of the TEA model from Case 2 but largely surpasses its performance in ACP and AP^1. Particularly, the ACP metric shows a significant improvement in the TEVAE model and achieves a higher value at $n = 3$ (does not decrease to zero as in Case 2). The AP^1 metric also reveals an upward trend, settling at a higher value than previously seen with the TEA model. For a more complete comparison of the three cases, we summarize the results of all metrics at $n = 6$ in Table 5.6. This summary demonstrates that the TEVAE model, when trained using the dataset with design rules, not only surpasses the TEA counterpart using the same dataset but also gives the best results across all metrics evaluated in all three cases.

5.5.3.2 Analysis on the Overall Parameter Accuracy

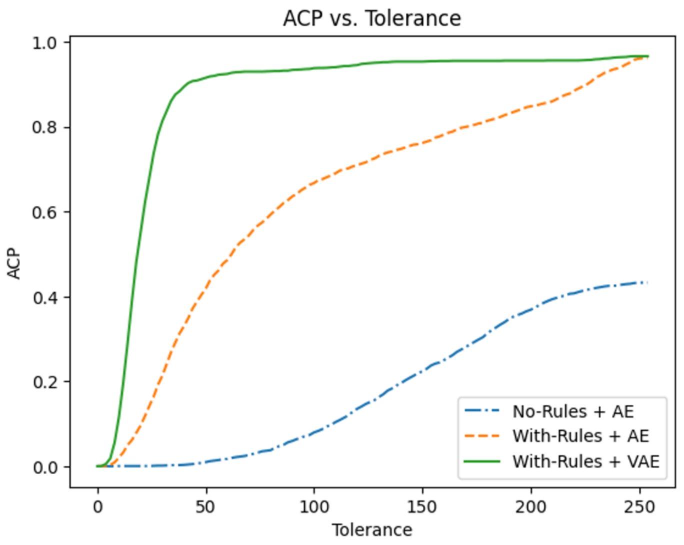
The models demonstrate high accuracy in predicting the sequence of CAD operations but are less precise in parameter prediction. To facilitate a clearer comparison between the three cases with respect to parameter prediction accuracy, we have included Figure 5.6 to illustrate the relationship between parameter accuracy and tolerance using metrics ACP and AP^1.

Especially, in Figure 5.6 (b), the blue dashed line with triangle markers represents the AP^1 value achieved by randomly guessing parameters given a specific tolerance (i.e., the random model), but without considering the *Sketch* parameter (i.e., the identifier of the sketch plane I) whose values are not allowed for tolerance. We derived the equation for the random model in Equation (5.8). The equation can be simplified to $\mathrm{AP}^1 = (-\eta^2 + 511\eta + 256)/65536$. This line acts as a baseline to evaluate the model's effectiveness in accurately predicting parameters if the sketch parameter is not considered.

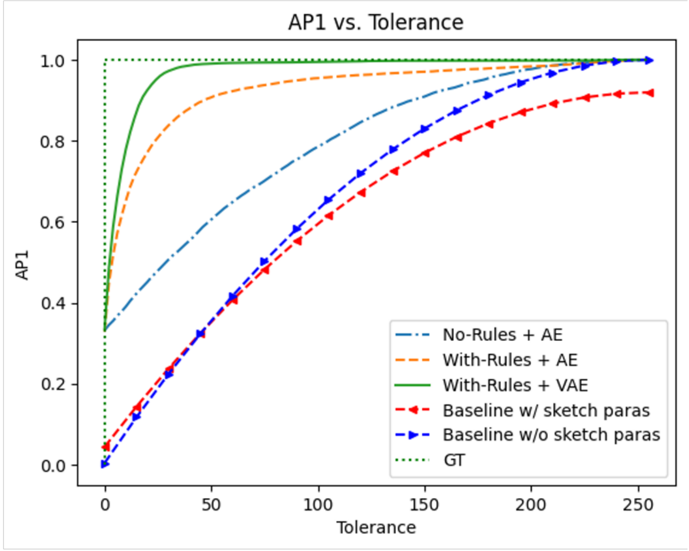
$$\mathrm{AP}^1 = \frac{1}{256}(\frac{(2\eta+1)(256-2\eta)+2(\frac{2\eta(1+2\eta)}{2}-\frac{\eta(1+\eta)}{2})}{256} \tag{5.8}$$

$$\mathrm{AP}^1 = \frac{1}{3}\cdot\frac{11}{91}+\frac{(-\eta^2+511\eta+256)}{65536}\cdot\frac{80}{91} \tag{5.9}$$

To consider the *Sketch* parameter I, we need to take into account the characteristics of the dataset (i.e., the ratio of each parameter taken among all possible parameters in the design representation of the CAD programs as outlined in Section 5.4.2). Accordingly, we obtain Equation (5.9) plotted as the red dashed line with triangle markers in Figure 5.6 (b). Note that the number of the sketch parameter



(a)



(b)

Figure 5.6: Overall parameter accuracy versus the tolerance levels evaluated using (a) ACP and (b) AP1 for the three cases. Especially, for (b), we included the baseline of the random prediction with or without considering the *Sketch* parameter I and the ground truth line.

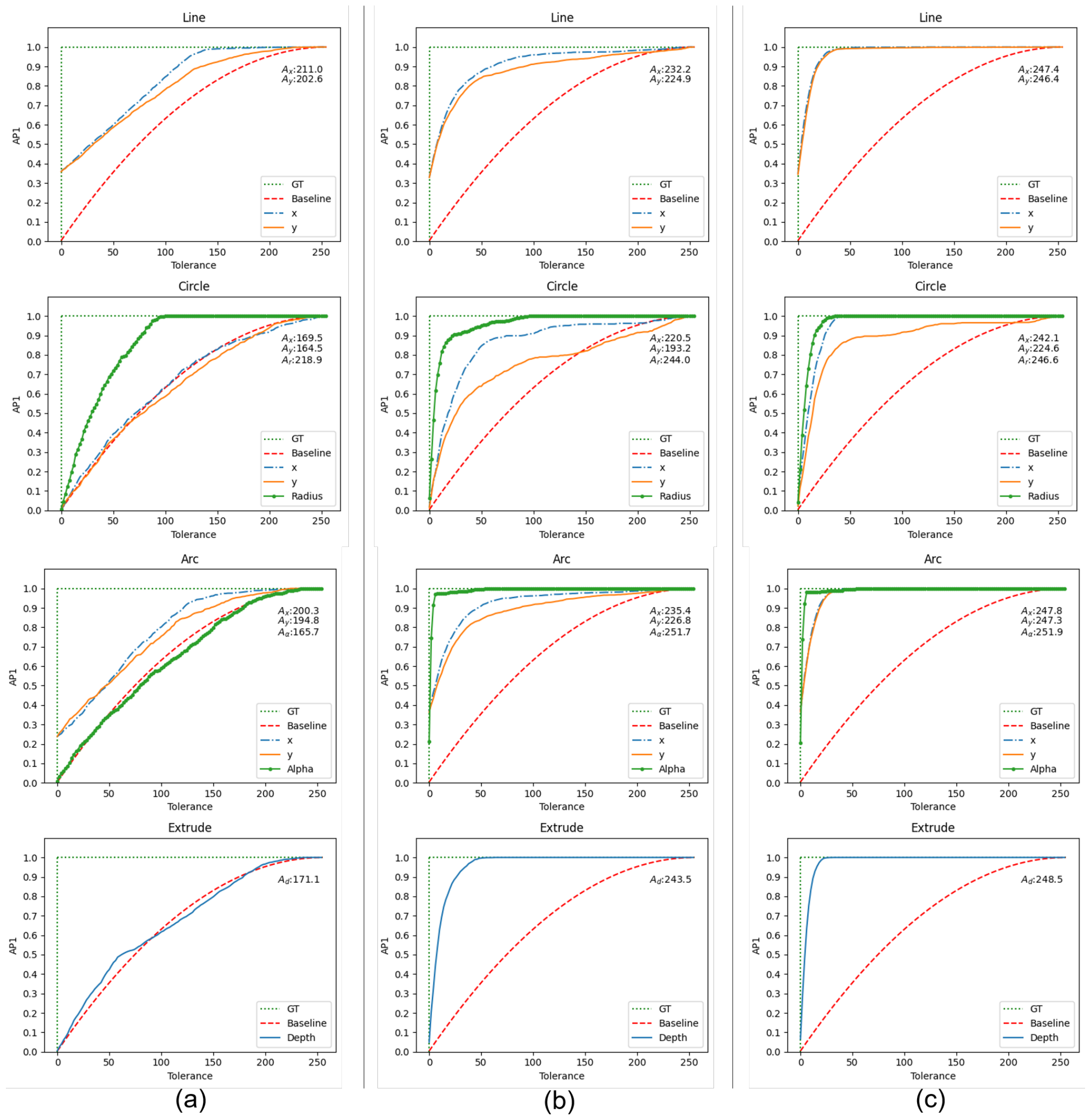


Figure 5.7: The variation of AP1 versus tolerance for the operation parameters for the *Line*, *Circle*, *Arc*, and *Extrude* operations, in that order. Column (a) shows the results of the network that utilizes the TEA architecture and is trained on the dataset without design rules. Column (b) shows the results of the same TEA architecture but trained on the dataset with rules. Column (c) shows the results using the TEVAE architecture and the dataset with rules.

takes $\frac{11}{91}$ of all parameters. Additionally, a green dotted line is used to indicate the ideal scenario where the parameters are perfectly predicted with zero tolerance (i.e., GT). Other lines in Figures 5.6 (a) and (b) depict the corresponding metric values for different cases, providing a comprehensive view of the model's performance in parameter prediction.

In both (a) and (b) of Figure 5.6, we consistently see that the metric values increase with rising tolerance levels. A notable point in Figure 5.6 (a) is that the ACP values for all three cases reach their highest at a tolerance of 255 and the corresponding values are 0.432, 0.961, and 0.967 for each case, in accordance with the ASOT values presented in Table 5.6. This can be interpreted as the result that when we evaluate the entire CAD program in terms of ACP given that all parameters are accurately predicted, we are essentially assessing ASOT. In Figure 5.6 (b), a crucial observation is that all three lines exceed the baseline of the random guess of parameters. This indicates that the models are effectively learning parameter prediction no matter if there are design rules embedded in the training data or not.

Additionally, a significant observation in both figures is how differently the models respond to changes in tolerance. Specifically, the TEVAE model, when trained using the dataset with rules, exhibits the highest sensitivity to changes in tolerance in contrast to Cases 1 and 2. This trend suggests that the TEVAE model excels in parameter prediction compared to the TEA model. The accuracy of these predictions depends on both the quality of the training data (for example, in this study, differentiated by the inclusion or exclusion of design rules) and the architecture of the model.

5.5.3.3 Analysis of Parameter Accuracy Based on Operation Types

To gain more insight into how the models perform in parameter prediction, we plotted the variation of AP^1 versus the tolerance for the operation parameters for each CAD operation type. Spanning columns (a) to (c), the rows in each column show variations in AP^1 against tolerance levels ($\eta = 0 - 255$) for specific parameters, corresponding to different CAD operations, *Line*, *Circle*, *Arc*, and *Extrude*. These results look into the model's adaptability and accuracy across various CAD operations, providing a comprehensive understanding of its capabilities in different model architectures and datasets. Each figure includes a red dashed line representing the baseline as defined in Equation (5.8). In addition, the green dotted line illustrates the perfect prediction of the parameters with zero tolerance. The other lines show the AP^1 for specific parameters related to the respective CAD operations. To facilitate a more quantitative comparison of how well the parameters are predicted, we also computed the area under the curve (AUC) for each parameter, as indicated in the upper right corner of each figure.

Column (a), the result of Case 1, shows that the x, y coordinates of the center of the *Circle*, the sweep angle α of the *Arc*, and the depth d of the *Extrude* align closely with the baseline. This suggests that the TEA model, when trained on a dataset without explicit design rules, performs similarly to random guessing for these specific parameters. However, the model still demonstrates the ability to learn certain patterns from the dataset, as evidenced by its recognition of the endpoint of the *Line*, the radius of the *Circle*, and the endpoint of the *Arc*.

In Column (b) for Case 2, there is an evident improvement in all metric values compared to Case 1. This improvement highlights the enhanced ability of the TEA model to predict parameters. The significant distance of these values from the baseline indicates that the model has effectively learned the design rules embedded in the training data, enabling it to predict the corresponding parameters more effectively.

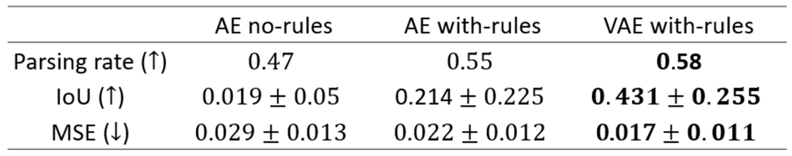
In Column (c), the results demonstrate an even better performance. All metric values not only surpass those in Column (b), but they also show a further deviation from the baseline, indicating a significant enhancement of the model's predictive performance. These values are closely approaching the ground truth (GT) line, underscoring the refined ability of the TEVAE model to learn and apply the embedded design rules from the training data.

5.5.3.4 Overall Evaluation of the 3D CAD Models and Images

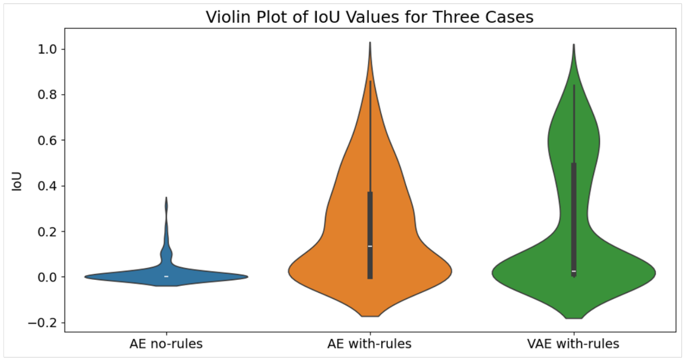
Figure 5.8 shows the summary of the parsing rate, intersection over union (IoU), and mean squared error (MSE) outlined in Figure 5.4 of the three cases. We perform an analysis of IoU and MSE by calculating the mean and standard deviation for each metric in the table. Following this computation, we depict the distribution of these values for each metric using a violin plot in Figure 5.8. These plots show that the data distributions of the IoU and MSE values move toward improved performance regions (i.e., higher values for IoU and lower values for MSE). Aligning with our observations in the evaluation of the CAD programs, as introduced in previous sections, the TEVAE trained using the dataset with design rules achieves the best performance among all three cases.

We show some qualitative results from the Image2CADSeq model with the TEVAE architecture in Figure 5.9. Subfigure (a) demonstrates near-perfect predictions, characterized by a high degree of accuracy in both the shape category determined by the sequence of CAD operation types and the parameters that determine the size and position of the shape. Subfigure (b) shows satisfactory predictions, where the model correctly identifies shape categories, yet exhibits discrepancies in size and position estimation. Subfigure (c) represents inadequate predictions, marked by the model's failure to accurately predict the categories of shapes. Each subfigure is arranged in a three-row format. The first row presents the initial input image. This is followed by the second row, which showcases two elements: the predicted CAD sequence and the rendered image of the resulting CAD model. The third row offers a comparative visual analysis, where the ground truth 3D shape is illustrated in a wireframe format against the predicted CAD model, rendered in solid.

The results indicate that the model is capable of generating a CAD sequence to reconstruct a 3D CAD model by accurately capturing and integrating both the



	AE no-rules	AE with-rules	VAE with-rules
Parsing rate (↑)	0.47	0.55	**0.58**
IoU (↑)	0.019 ± 0.05	0.214 ± 0.225	$\mathbf{0.431 \pm 0.255}$
MSE (↓)	0.029 ± 0.013	0.022 ± 0.012	$\mathbf{0.017 \pm 0.011}$



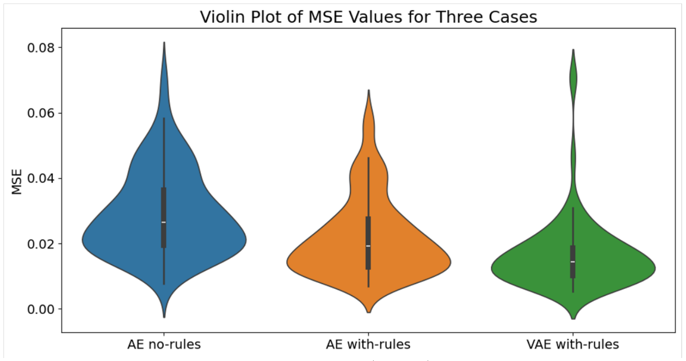
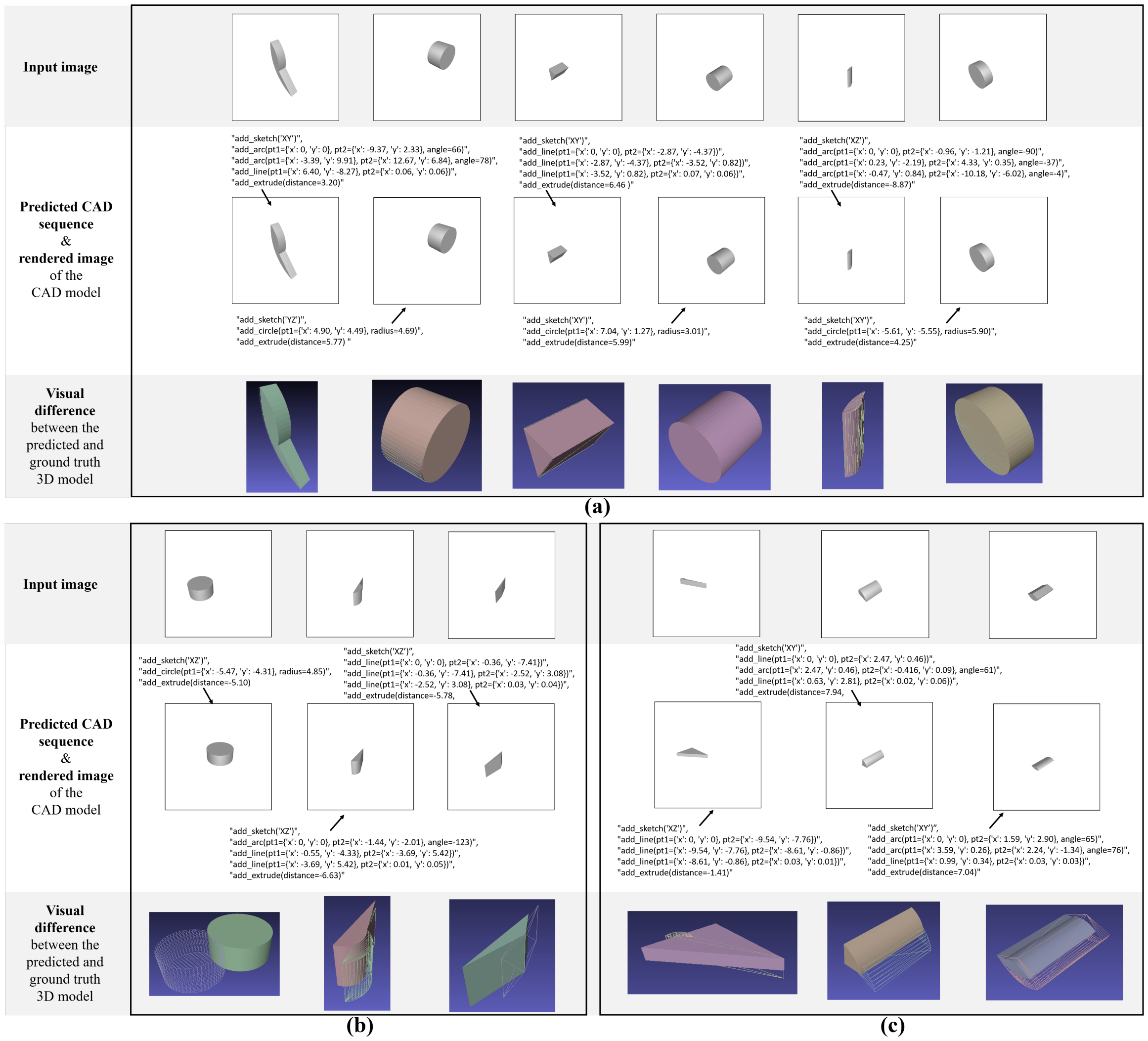


Figure 5.8: Parsing rate, intersection over union (IoU), and mean squared error (MSE) of the three cases. To analyze IoU and MSE, the mean and standard deviation are computed for each metric within the table. Subsequently, a violin plot is presented illustrating the distribution of these computed values for each metric.

spatial positioning and the geometric details reflected in the input image. However, there remains room for improvement, particularly in predicting the correct CAD sequence and especially the associated parameters, as also shown by the aforementioned quantitative results.

5.6 Discussion

In this section, we discuss the insights obtained from the experiments as well as the limitations of the current study.


Input image
Predicted CAD sequence & rendered image of the CAD model
Visual difference between the predicted and ground truth 3D model
(a)
(b)
(c)

```
"add_sketch('XY')",
"add_arc(pt1={'x': 0, 'y': 0}, pt2={'x': -9.37, 'y': 2.33}, angle=66)",
"add_arc(pt1={'x': -3.39, 'y': 9.91}, pt2={'x': 12.67, 'y': 6.84}, angle=78)",
"add_line(pt1={'x': 6.40, 'y': -8.27}, pt2={'x': 0.06, 'y': 0.06})",
"add_extrude(distance=3.20)"
```

```
"add_sketch('XY')",
"add_line(pt1={'x': 0, 'y': 0}, pt2={'x': -2.87, 'y': -4.37})",
"add_line(pt1={'x': -2.87, 'y': -4.37}, pt2={'x': -3.52, 'y': 0.82})",
"add_line(pt1={'x': -3.52, 'y': 0.82}, pt2={'x': 0.07, 'y': 0.06})",
"add_extrude(distance=6.46 )"
```

```
"add_sketch('XZ')",
"add_arc(pt1={'x': 0, 'y': 0}, pt2={'x': -0.96, 'y': -1.21}, angle=-90)",
"add_arc(pt1={'x': 0.23, 'y': -2.19}, pt2={'x': 4.33, 'y': 0.35}, angle=-37)",
"add_arc(pt1={'x': -0.47, 'y': 0.84}, pt2={'x': -10.18, 'y': -6.02}, angle=-4)",
"add_extrude(distance=-8.87)"
```

```
"add_sketch('YZ')",
"add_circle(pt1={'x': 4.90, 'y': 4.49}, radius=4.69)",
"add_extrude(distance=5.77) "
```

```
"add_sketch('XY')",
"add_circle(pt1={'x': 7.04, 'y': 1.27}, radius=3.01)",
"add_extrude(distance=5.99)"
```

```
"add_sketch('XY')",
"add_circle(pt1={'x': -5.61, 'y': -5.55}, radius=5.90)",
"add_extrude(distance=4.25)"
```

```
"add_sketch('XZ')",
"add_circle(pt1={'x': -5.47, 'y': -4.31}, radius=4.85)",
"add_extrude(distance=-5.10)
```

```
"add_sketch('XZ')",
"add_line(pt1={'x': 0, 'y': 0}, pt2={'x': -0.36, 'y': -7.41})",
"add_line(pt1={'x': -0.36, 'y': -7.41}, pt2={'x': -2.52, 'y': 3.08})",
"add_line(pt1={'x': -2.52, 'y': 3.08}, pt2={'x': 0.03, 'y': 0.04})",
"add_extrude(distance=-5.78,
```

```
"add_sketch('XZ')",
"add_arc(pt1={'x': 0, 'y': 0}, pt2={'x': -1.44, 'y': -2.01}, angle=-123)",
"add_line(pt1={'x': -0.55, 'y': -4.33}, pt2={'x': -3.69, 'y': 5.42})",
"add_line(pt1={'x': -3.69, 'y': 5.42}, pt2={'x': 0.01, 'y': 0.05})",
"add_extrude(distance=-6.63)"
```

```
"add_sketch('XY')",
"add_line(pt1={'x': 0, 'y': 0}, pt2={'x': 2.47, 'y': 0.46})",
"add_arc(pt1={'x': 2.47, 'y': 0.46}, pt2={'x': -0.416, 'y': 0.09}, angle=61)",
"add_line(pt1={'x': 0.63, 'y': 2.81}, pt2={'x': 0.02, 'y': 0.06})",
"add_extrude(distance=7.94,
```

```
"add_sketch('XZ')",
"add_line(pt1={'x': 0, 'y': 0}, pt2={'x': -9.54, 'y': -7.76})",
"add_line(pt1={'x': -9.54, 'y': -7.76}, pt2={'x': -8.61, 'y': -0.86})",
"add_line(pt1={'x': -8.61, 'y': -0.86}, pt2={'x': 0.03, 'y': 0.01})",
"add_extrude(distance=-1.41)"
```

```
"add_sketch('XY')",
"add_arc(pt1={'x': 0, 'y': 0}, pt2={'x': 1.59, 'y': 2.90}, angle=65)",
"add_arc(pt1={'x': 3.59, 'y': 0.26}, pt2={'x': 2.24, 'y': -1.34}, angle=76)",
"add_line(pt1={'x': 0.99, 'y': 0.34}, pt2={'x': 0.03, 'y': 0.03})",
"add_extrude(distance=7.04)"
```

Figure 5.9: Qualitative analysis of Image2CADSeq model using TEVAE. (a) Near-perfect predictions: High accuracy in shape, size, and position. (b) Satisfactory predictions: Correct shape categories, and inaccurate sizes or positions. (c) Inadequate predictions: Wrong prediction of shape categories. Each subfigure includes the first row - Input image; the second row - Predicted CAD sequence and rendered image of the resultant CAD model; the third Row - Visual comparison between the ground truth (wireframe) and predicted 3D model (solid).

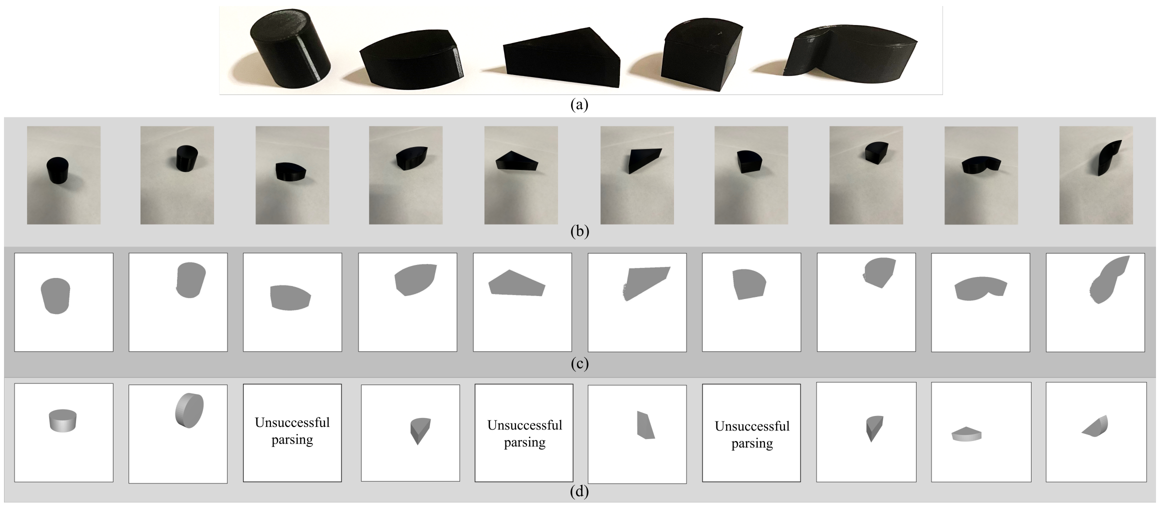


Figure 5.10: Validation experiments using real-world images. (a) 3D-printed design objects; (b) Photographs (3024×4032 pixels) of the objects taken using a smartphone; (c) Processed images (512×512 pixels) for the input to the Image2CADSeq model; (d) Rendered images of the resultant 3D model from the predicted CAD sequence. Unsuccessful cases are also shown.

5.6.1 The Impact of Design Rules

The results in Section 5.5.3 consistently show that the TEA model trained on the dataset with rules outperforms the one without rules. This result underscores the beneficial impact of design rules on the predictive performance of the model in Image2CADSeq tasks.

This has three implications: (1) Compared to the dataset with randomly generated dimension information, the dataset that incorporates design rules introduces latent patterns and relations for the model to learn. This suggests that our model is effective in capturing those embedded rules and thus generating more accurate and more realistic designs since real-world CAD models are often created with domain-specific knowledge and design rules. (2) The inclusion of design rules can enhance the model's generalizability. It also plays a critical role in minimizing the probability of creating unfeasible CAD designs. (3) Since practical CAD designs often follow domain-specific standards and knowledge, data are therefore inevitably associated with rules. Thus, the proposed method has strong practical implications.

5.6.2 TEA VS. TEVAE

The improved predictive performance in TEVAE is due to the use of a variational autoencoder (VAE) that can capture the latent design representations of CAD programs. The effectiveness of the TEVAE architecture is demonstrated through improved performance across various metrics, significantly exceeding the results achieved in the TEA model. The superior performance of the TEVAE model can be largely attributed to the three advantages offered by VAEs. (1) Unlike traditional AEs, VAEs create a latent space that follows a well-defined and continuous distribution, such as the Gaussian distribution typically used. This design facilitates smoother interpolation between data points, enhancing the capture of meaningful variations in CAD designs. (2) The encoder in a VAE is more efficient in extracting relevant and prominent features from CAD programs than a standard AE. This efficiency stems from the VAE's focus on capturing the underlying data distribution, rather than merely replicating input data. (3) The inclusion of the KL-divergence term in the VAE's loss function helps reduce overfitting. It promotes the model to capture a broader data distribution rather than memorizing specific instances. This enhances TEVAE's generalizability on new, unseen data.

5.6.3 Limitations and Future Work

It is indeed a challenge in our research to further improve the prediction accuracy of operation parameters. An important observation from our experiments is the near-perfect reconstruction capability in Stage 1 training with an accuracy of the CAD program (ACP), up to 99.9%. However, the problem arises during Stage 2 training, which involves regressing the latent space learned in Stage 1 using images as input. The difficulty lies in aligning the latent representation of the image with that of the CAD programs. To address this issue, we plan to explore modal alignment techniques as introduced in the recent literature (Li et al., 2023b; Song et al., 2020). They could offer a promising solution to unify different modalities in a single latent space to promote cross-modal synthesis. In addition, building upon the initial two-stage training framework, incorporating an optimization pipeline stands as a promising strategy to enhance the alignment between the CAD program outputs and the input images. Such an optimization process would iteratively refine the generated CAD sequences, minimizing the discrepancy between the predicted CAD programs and the intended design expressed in the input images. By leveraging techniques, such as gradient descent, within this pipeline, the system could better adapt the latent representations, leading to improved fidelity in the cross-modal synthesis. This optimization could be guided by a loss function specifically designed to capture the semantic differences between the generated CAD model and the image, possibly utilizing differentiable rendering for direct feedback on the quality of the 3D model as perceived from the image perspective.

We have been focused on synthesizing simple geometrical shapes, such as cylinders and tri-prisms. While these basic geometries are fundamental to more complex designs, our focus on them has limited the network's capability to handle intricate, real-world design tasks. Recognizing this, we acknowledge the need to train the Image2CADSeq model with more diverse and complex datasets to tackle advanced design challenges. We plan to collect more sophisticated geometries that mirror the complexities encountered in actual design environments, often embodied as assemblies comprising multiple interconnected components. To achieve this, we plan to explore two primary strategies: (1) Enhancing our data synthesis pipeline: We intend to integrate a wider range of complex geometries into our current data synthesis pipeline. This expansion will allow the network to learn from a broader spectrum of shapes and structures, better preparing it for real-world applications. (2) Using real-world design datasets: Another avenue involves harnessing datasets that include historical CAD modeling process data. An example is the Autodesk Fusion 360 Gallery dataset (Willis et al., 2021b), which offers a rich source of real-world design examples. Our objective here is to extract CAD sequences that correspond to more intricate designs, similar to the simplified CAD programs outlined in Table 5.2. This approach will enable the network to learn from actual design processes, further enhancing its applicability to practical scenarios.

At the end of this study, we are curious about the model performance in real-world applications where users can take photos of physical artifacts and instantly transform them into CAD sequences. Therefore, we conducted a validation experiment in which five types of template shapes, as listed in Table 5.5, were first 3D printed as shown in Figure 5.10 (a). Then, high-resolution photographs (3024×4032 pixels) of these printed objects were captured using a smartphone, as shown in the second row of Figure 5.10. Subsequently, these photographs were resized to 512×512 pixels to facilitate input into the Image2CADSeq model for the generation of CAD sequences. The third row illustrates this preprocessing stage, while the fourth row presents the rendered images of the resultant 3D models, including instances of unsuccessful parsing. The parsing rate achieved in this experiment was 70%, mirroring the proficiency level indicated in Figure 5.8. For the 3D models successfully parsed, four were predicted as correct categories but with inaccurate parameters. The remaining three were incorrectly predicted. This indicates that the model performance, when using real-world 3D objects and their image data, is inferior, compared to using the synthesized dataset (i.e., the 3D data generated in CAD) as reported in Section 5.

The result underscores the need for further enhancements in the Image2CADSeq model to improve its accuracy in inferring the CAD representations of images of real-world objects. In particular, enhancing the model's ability to accurately predict parameters is essential to improve the parsing rate. Moreover, to address the model's current limitation in handling various images with different colors, textures, perspectives, or specific artistic styles, data augmentation methods, such as the incorporation of objects in different colors and in various lighting conditions or backgrounds, could be beneficial. Such treatments are expected to enrich the model's training data and, thereby, improve its ability to process a wider array of real-world image data.

5.7 Conclusion

In this study, we developed a novel Image2CADSeq model to predict CAD sequences from images. This network, exemplified by the performance of the TEVAE model, aims to revolutionize design methodologies by enabling the conversion of images into CAD sequences. A CAD sequence offers more benefits than pure 3D CAD models, such as greater flexibility in modifying CAD operations and managing the historical process/knowledge of CAD model construction.

For training purposes, our focus is on synthesized data representing simple shape primitives. In addition, we propose an evaluation framework that can comprehensively assess model performance. The results obtained are very promising, yet improvement can still be made. Therefore, our future efforts will be directed towards (1) Enhancing geometric complexity. We will expand the model's capabilities to encompass a broader spectrum of geometries. This expansion aims to align the model more closely with those in real-world design applications; (2) Incorporating diverse design data. A key area of development involves the integration of more varied and realistic design datasets. This can greatly facilitate the machine learning process; (3) Advancing training methodologies. We plan to explore innovative network architectures and training methodologies to improve the efficiency and adaptability of the model; (4) Incorporating industry standards. Engaging with industry experts will be crucial to guide the development of the model. Their insights will ensure that the model meets practical needs and adheres to industry standards.

In summary, the proposed approach has potential to lead to transformative changes in existing CAD systems, revolutionizing the product development cycle. Additionally, it has the potential to promote the democratization of design, allowing people with limited experience or expertise to actively participate in CAD. For example, this approach can help regular customers engage in product design and concept generation, promoting personalized design and creation and human-centered generative design (Demirel et al., 2023b; Li et al., 2023b).

Chapter 6: LLM4CAD: Multi-Modal Large Language Models for Three-Dimensional Computer-Aided Design Generation

Abstract

Little is known about the ability of multimodal LLMs to generate 3D design objects, and there is a lack of quantitative assessment. In this study, we develop an approach to enable two LLMs, GPT-4 and GPT-4V, to generate 3D CAD models (i.e., LLM4CAD) and perform experiments to evaluate their efficacy. To address the challenge of data scarcity for multimodal LLM studies, we created a data synthesis pipeline to generate CAD models, sketches, and image data of typical mechanical components (e.g., gears and springs) and collect their natural-language descriptions with dimensional information using Amazon Mechanical Turk. We positioned the CAD program (programming script for CAD design) as a bridge, facilitating the conversion of LLMs' textual output into tangible CAD design objects. We focus on two critical capabilities: the generation of syntactically correct CAD programs (Cap1) and the accuracy of the parsed 3D shapes (Cap2) quantified by intersection over union. The results show that both GPT-4 and GPT-4V demonstrate great potential in 3D CAD generation. The potential of multimodal LLMs in enhancing 3D CAD generation is clear, but their application must be carefully calibrated to the complexity of the target CAD models to be generated.

6.1 Introduction

The emergence of large language models (LLMs), including the generative pre-trained transformer (GPT) series (Brown et al., 2020), represents a significant advancement in the capabilities of artificial intelligence (AI) to interact with the world. These models, trained on vast datasets, exhibit remarkable proficiency in "understanding" the nuances of human language and generating text that mirrors human-like communication (Kasneci et al., 2023). However, the inherent vagueness of natural language continues to pose a significant challenge, especially when it comes to conveying complex instructions to LLMs. To this end, cutting-edge multimodal LLMs, such as OpenAI's GPT-4 Vision (GPT-4V) (OpenAI, 2023a) and Google's PaLM-E (Driess et al., 2023), have been proposed. These models are designed to process more input modalities besides text, such as images, thereby broadening how users can interact with LLMs for more sophisticated tasks.

The utility of LLMs in processing natural language data has extended their application in design research for conceptual design (Kocaballi, 2023; Filippi, 2023; Ma et al., 2023). One particular limitation of these studies is that they use textual information only as the input. However, it might be difficult to effectively describe intended design artifacts and associated parameters through text only, which often encompass the structural and layout specifications of a component and the desired shapes of the component. Furthermore, conceptual design is inherently multimodal, frequently incorporating visual elements ranging from sketches for design ideation to engineering drawings for fabrication and assembly (Li et al., 2023b; Song et al., 2023c).

Therefore, recent design research emphasizes the significance of multimodal machine learning (MMML) in improving the conceptual design by integrating diverse modalities (Li et al., 2023b; Song et al., 2023c; Li et al., 2022c). Multimodal input, such as images and sketches beyond texts, could potentially improve LLMs' performance in understanding designers' intent, thus generating more precise and quality design output. Therefore, multimodal LLMs that can take multiple input modalities have great potential for their application in AI-assisted conceptual design, promising to revolutionize design tools and human-AI collaborative design. However, to our knowledge, no quantitative evaluation has been performed to assess the efficacy of multimodal LLMs in the CAD generation of 3D shapes for conceptual design.

To fill this research gap, we develop an approach to enable multimodal LLMs for ***3D CAD Generation*** (hereafter referred to as LLM4CAD) and conduct a quantitative analysis to evaluate LLM4CAD's effectiveness in conceptual design. Specifically, we seek to understand the capabilities of multimodal LLMs to generate high-quality 3D design concepts with precise dimensions and to identify strategies to improve their capabilities. This study is driven by two research questions (RQs): *1) To what extent can multimodal LLMs generate 3D design objects when employing different design modalities or a combination of various modalities? 2) What strategies can be developed to enhance the ability of multimodal LLMs to create 3D design objects?*

To enable LLM4CAD, one technical challenge is that LLMs cannot directly create 3D shapes, such as meshes, voxels, and boundary representations. However, LLMs can generate programming codes. Therefore, we developed an approach to enable an indirect synthesis of 3D design objects by generating CAD programs (Nelson et al., 2023). To quantitatively evaluate the performance, we propose a data synthesis pipeline along with an evaluation framework. This evaluation specifically focuses on two capabilities. **Cap1**: the success rate of the generated programming codes in program-to-CAD translation, and **Cap2**: the extent to which these resultant 3D design objects align with the ground-truth shapes. We summarize our contributions as follows.

1. This study created a new CAD dataset of five categories of mechanical components (i.e., shafts, nuts, flanges, springs, and gears with diverse geometry complexity) for multimodal LLMs, including textual descriptions, sketches, images, and 3D CAD models. In particular, textual descriptions of the target design

objects are in natural languages with detailed dimensional information collected with Amazon Mechanical Turk (AMT) [1], an online crowdsourcing platform.

2. The effectiveness of the GPT-4 and GPT-4V models in 3D design generation was evaluated, and new knowledge of their strengths and limitations was obtained.

3. A new method was developed and implemented to enhance the GPT models' proficiency in generating 3D CAD models. Specifically, we develop a debugger to correct syntax errors in the synthesized CAD programs to improve their success rate of being translated to 3D CAD models.

We found that GPT-4 and GPT-4V models have demonstrated the significant potential of LLM4CAD especially when enhanced by the proposed debugger. However, they still struggle with handling complex geometries. Additionally, GPT-4V's performance was examined with four input modes including text-only, text with sketch, text with image, and a combination of text, sketch, and image. The results show that on average GPT-4V particularly excels when processing purely text-only input, outperforming multimodal inputs in both Cap 1 and Cap 2. This observation is counterintuitive because a prevailing belief in the field of MMML is that incorporating varied input modalities should improve a machine learning (ML) model's predictive accuracy due to an increased amount of information for learning and inference. However, when examining category-specific results of mechanical components, multimodal inputs start to gain prominence with more complex geometries (e.g., springs and gears) in both Cap 1 and Cap 2.

Based on these observations, it is clear that the current multimodal LLMs (e.g., GPT-4V) still face limitations in handling multimodal inputs for generating 3D CAD objects. However, the detailed insights from the category-specific results show that multimodal inputs become more effective as the complexity of design objects

[1]`https://www.mturk.com/`

increases. Therefore, these limitations do not diminish their potential benefits in real-world design scenarios characterized by complex objects. The ability of multimodal LLMs to process diverse input modalities remains a promising avenue for enhancing 3D CAD generation technologies.

The remainder of this paper is organized as follows. In Section 6.2, we provide an overview of the background related to multimodal machine learning and LLMs for engineering design. Section 6.3 outlines the methodology for data collection and generation, as well as the evaluation of multimodal LLMs. Subsequently, Sections 6.4 and 6.5 present, analyze, and discuss the experimental results, from which we summarize the primary findings and acknowledge limitations. Conclusions and closing remarks are made in Section 6.6, where we present key insights and suggest potential directions for future research.

6.2 Literature Review

In this section, we review the most relevant literature to our work including multimodal machine learning and large language models for engineering design.

6.2.1 Multimodal Machine Learning in Engineering Design

Multimodal machine learning (MMML) approaches exhibit significant promise in enhancing the field of engineering design, as evidenced by recent review studies (Li et al., 2023b; Song et al., 2023c). Specifically, when confronted with inputs comprising multiple modalities, such as a combination of text and sketches, MMML techniques can integrate this information through a process known as multi-modal fusion. This fusion enables integrating data from diverse modalities to facilitate prediction tasks such as regression or classification. The application of multimodal fusion in different areas (e.g., audio-visual speech recognition, image captioning) is becoming popular(Baltrušaitis et al., 2018). The data from different modalities can supplement each other, aiding in increasing the accuracy of predictions. Even if one modality is missing, predictions can still be viable. While there might be overlap in information from multiple modalities, this redundancy can strengthen the reliability of the predictions (Baltrušaitis et al., 2018).

Despite these advantages, the extent to which multimodal fusion can enhance engineering design remains largely unexplored, with only a few pioneering works looking into this area (Song et al., 2023a; Su et al., 2023). For example, Song, Miller, and Ahmed (Song et al., 2023a) pioneered a multimodal learning model that integrates sketch and textual description modalities using a cross-attention mechanism. This approach facilitated a comprehensive assessment of design concepts, revealing that MMML significantly enhances the model's predictive and explanatory capabilities. The findings underscore the advantages of employing multimodal representations in conceptual design evaluation.

MMML for engineering design is still in its initial stage, presenting ample opportunities for extensive research exploration into the theory and methodology for enhancing design evaluation and generation. Our study investigates multimodal large language models (LLMs)' capability to generate 3D design concepts when taking multiple design modalities (e.g., a combination of text, image, and sketch) as input compared to unimodal input (e.g., text), contributing to the field of MMML for engineering design.

6.2.2 Large Language for Engineering Design

Natural language processing (NLP) is a foundational technology in AI advancements, primarily focusing on enabling computers to understand and interact with humans using natural language (Chowdhary and Chowdhary, 2020). Building upon the foundation of various NLP technologies, the emergence of LLMs such as the generative pre-trained transformer (GPT) series (Brown et al., 2020) marks a significant leap in AI proficiency.

A remarkable example of LLMs is ChatGPT (Cha, 2022), launched in 2022 by

OpenAI. ChatGPT, an advanced Chatbot built upon the GPT-3.5 model, provides detailed and structured responses based on specific user prompts. Its capabilities span a broad spectrum of language understanding and generation tasks, including multilingual translation, creative writing, and programming code creation and debugging. A distinctive feature of ChatGPT is its ability to recall previous conversation segments, enabling more coherent and sustained interactions (Wu et al., 2023b; Ray, 2023). ChatGPT is the state-of-the-art LLM and stands apart from earlier NLP and LLM tools due to its exceptional conversational skills and reasoning abilities across various domains (Wu et al., 2023a; Haleem et al., 2022; Abdullah et al., 2022).

Researchers have been examining how ChatGPT can be applied to enhance the engineering design process, from conceptual design to manufacturing (Filippi, 2023; Kocaballi, 2023; Wang et al., 2023; Makatura et al., 2023; Nelson et al., 2023; Wu et al.). For instance, Kocaballi (Kocaballi, 2023) undertook a hypothetical design project that leveraged ChatGPT to create personas in the roles of designers or users. The approach facilitated various design-related activities, including conducting user interviews, generating design concepts, and evaluating user experiences. However, these studies primarily focus on the text generation capabilities of LLMs by taking textual input. Taking the generation of design concepts as an example, it can be advantageous to employ the generated text for brainstorming design ideas (Ma et al., 2023). However, translating these conceptual ideas into concrete 3D designs still presents a significant challenge.

While LLMs' ability to directly generate 3D objects (e.g., meshes, voxels, and boundary representations) seems limited, an alternative approach involves the generation of 3D designs using CAD programming languages such as CADQuery and OpenSCAD. This can be achieved by exploiting LLMs' capacity for program synthesis (Gulwani et al., 2017) and some research has been investigating the potential of LLMs in producing 3D designs through CAD programs, which involves interpreting human language instructions and converting them into CAD designs (Makatura et al., 2023; Nelson et al., 2023). Nevertheless, these studies are still limited to textual descriptions for the design intent and it is often challenging to convey complex tasks solely through text.

The evolution of OpenAI's GPT architecture, transitioning from the text-only GPT-3.5 and GPT-4 to its multimodal successors, GPT-4 Vision (GPT-4V) (OpenAI, 2023b,a; Ray, 2023) marks significant advancements. It offers opportunities to incorporate multiple modalities besides text. However, little is known about its practicality in and for engineering design, such as 3D CAD generation. That motivates our study to conduct a quantitative analysis on to what extent multimodal LLMs can generate 3D design objects when employing different design modalities or a combination of various modalities. We employed GPT-4 and GPT-4V as examples of unimodal and multimodal LLMs for our experiments due to their acknowledged outstanding performance.

6.3 Methodology

We develop an approach to enabling LLM4CAD by taking various design modalities and assessing the extent of their capabilities, as shown in Figure 6.1. This approach consists of three major steps: 1) Data Synthesis: multimodal design data collection and generation, 2) Code Generation: CAD program code generation, and 3) Evaluation: the evaluation of 3D CAD model generation in terms of success rate and precision.

For clarity of illustration, we consider a gear as a representative 3D design object. The process begins with the generation of ground-truth (GT) 3D CAD models, dimensional data is recorded alongside the generation process. Direct rendering techniques can be employed to obtain images from the 3D shapes. Meanwhile, textual descriptions incorporating dimensional information, and sketches can be acquired through automated algorithms or human involvement. With the input data in three design modalities (i.e., text, image, and sketch), we evaluate the capability of the GPT-4 and GPT-4V models to generate CAD programs which are then parsed into

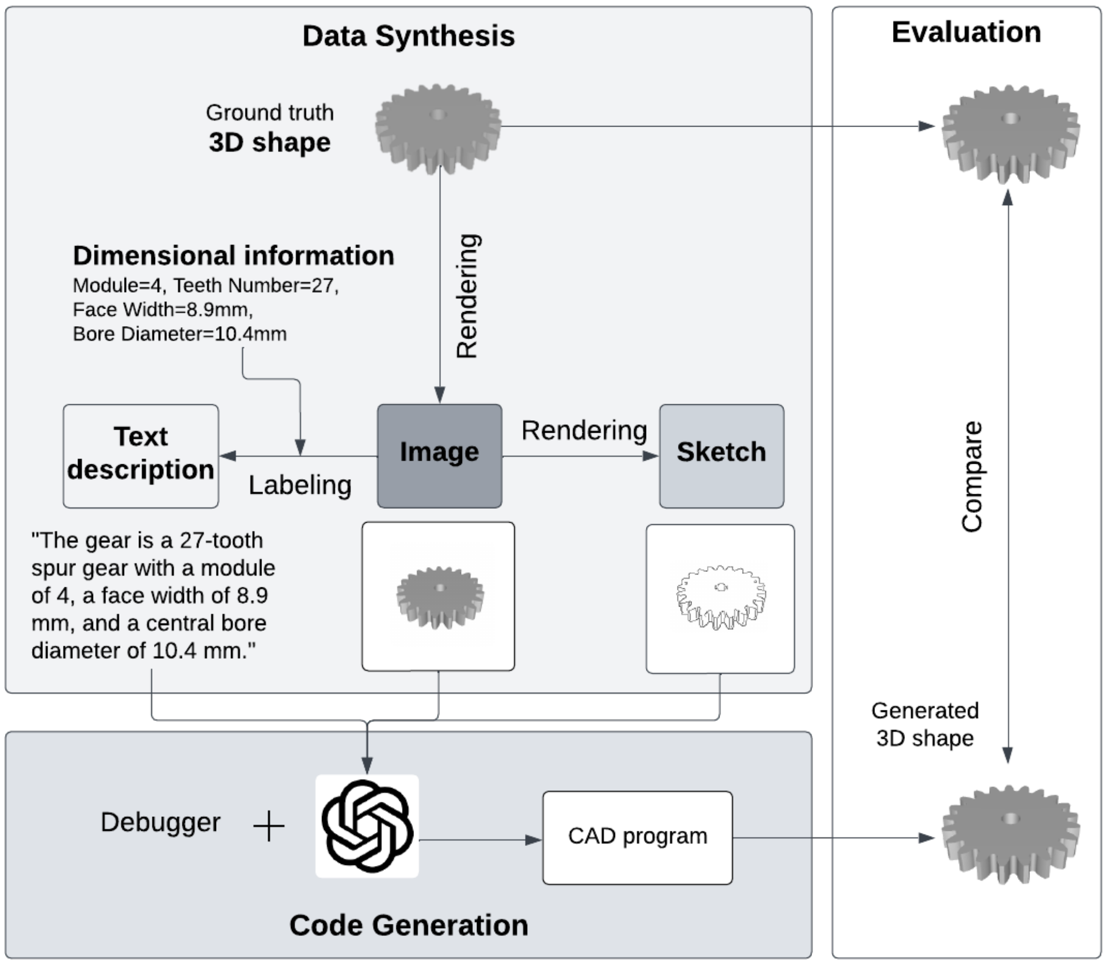


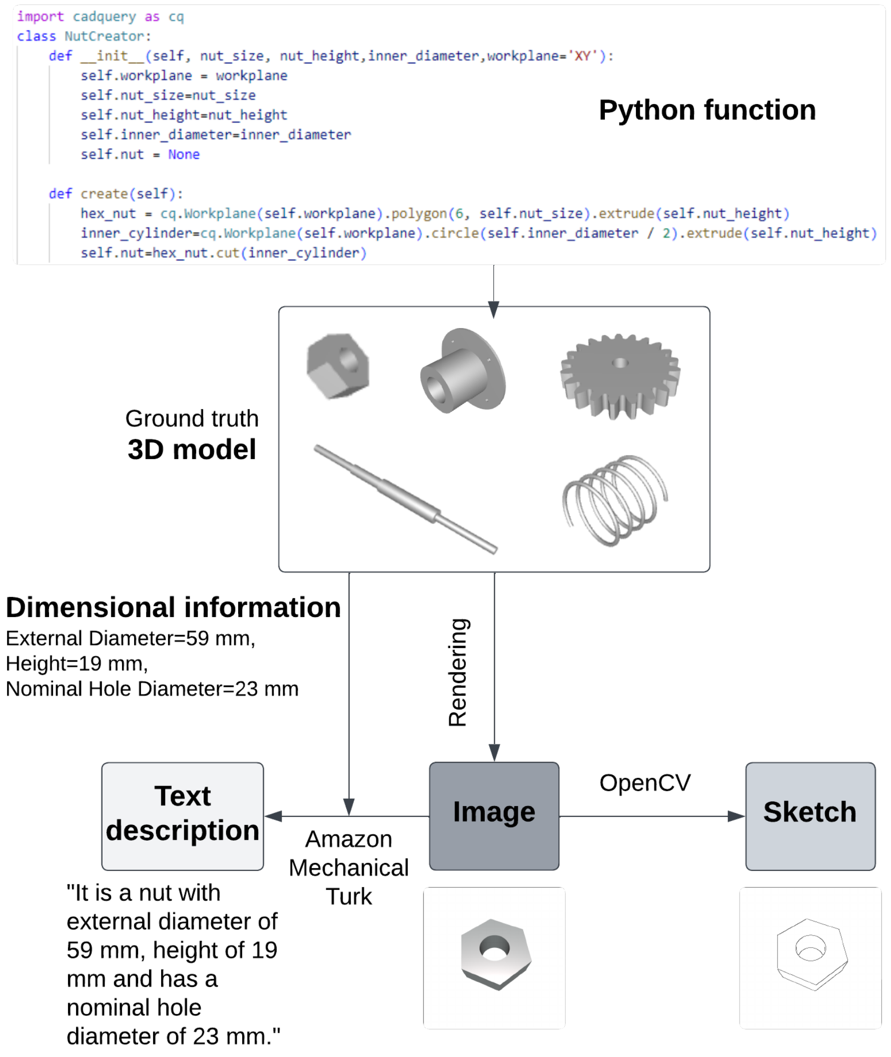
Figure 6.1: The overview of our approach

3D shapes. The quality of the resulting 3D shapes is then benchmarked against the GT shapes to gauge the efficacy of generation. In addition, we proposed a debugger to enhance the models' capabilities in CAD program generation.

6.3.1 Data Synthesis

We choose mechanical components as target 3D objects given their pivotal role in engineering design. Upon examining the literature and online resources, we could not find any CAD model dataset of mechanical components that incorporates multiple design modalities and detailed dimensional information. The existing datasets of mechanical components (Kim et al., 2020; Lee et al., 2022; Manda et al., 2021) do not provide essential dimensional data. Therefore, they are not appropriate for this study because a quantitative evaluation of the generated CAD models is impossible since no dimensional information can be provided to LLMs as input. This motivates us to develop a new synthesis pipeline for CAD objects with detailed dimensional information. Such a dataset would benefit various machine learning tasks in engineering design, where the details of the design specifications are critical.

As shown in Figure 6.2, a semi-automated pipeline for Data Synthesis is developed to generate the textual descriptions, images, sketches, and GT 3D shapes of five common types of mechanical components: shafts, nuts, flanges, springs, and gears. They were chosen for their popularity in engineering design and their varying levels of complexity, allowing us to test the robustness of our approach and investigate how geometric complexity would influence the results. The complexity of mechanical components in our research is determined by the solvability of the design problem (Summers and Shah, 2010) based on the results of our trials and errors. The solvability refers to the capability of LLMs to formulate a CAD program to meet the input design requirements. Generating CAD programs for components such as shafts, nuts, and flanges, which primarily utilize *Sketch* and *Extrude* operations in CAD, is relatively simple. However, we find that the creation of CAD programs for springs and gears presents significant challenges. This complexity arises from the need to employ the *Evolve* CAD operation for springs, and integrate auxiliary Python libraries or conduct complex calculations for determining the profiles of gear teeth. Thus, in this paper, we refer to shafts, nuts, and flanges as simple geometries while springs and gears are complex geometries.



```
import cadquery as cq
class NutCreator:
    def __init__(self, nut_size, nut_height,inner_diameter,workplane='XY'):
        self.workplane = workplane
        self.nut_size=nut_size
        self.nut_height=nut_height
        self.inner_diameter=inner_diameter
        self.nut = None

    def create(self):
        hex_nut = cq.Workplane(self.workplane).polygon(6, self.nut_size).extrude(self.nut_height)
        inner_cylinder=cq.Workplane(self.workplane).circle(self.inner_diameter / 2).extrude(self.nut_height)
        self.nut=hex_nut.cut(inner_cylinder)
```

"It is a nut with external diameter of 59 mm, height of 19 mm and has a nominal hole diameter of 23 mm."

Figure 6.2: The pipeline for Data Synthesis

6.3.1.1 3D Shapes, Images, and Sketches

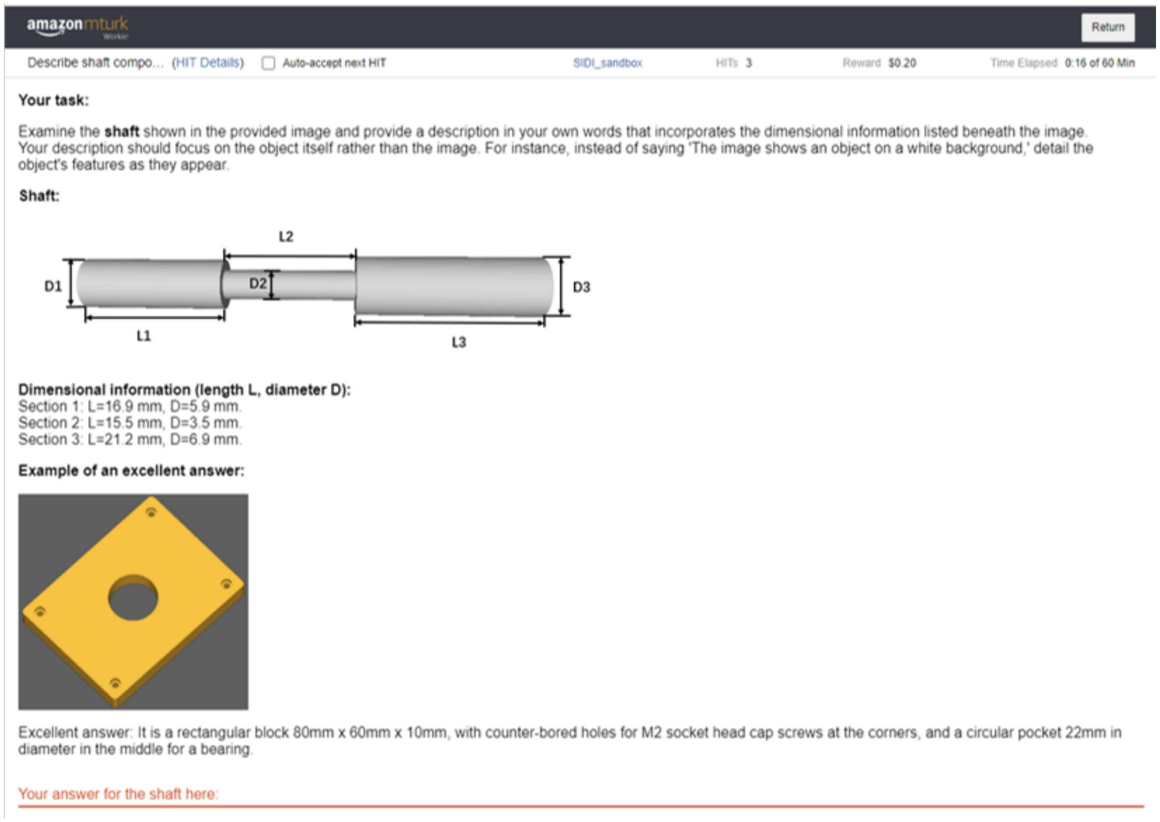
Using CADQuery (Version 2.3.1) [2], a Python-based CAD programming language, we created five distinct Python classes, each corresponding to one of the mechanical component categories. In each class, the design is parameterized, so a variety of designs can be generated from a defined design space. An example of the classes is given in Figure 6.2. To achieve uniform sampling of the design space, we employed Latin Hypercube Sampling (McKay et al., 2000) of design parameters (such as the external diameter and height of a nut). For shafts, we synthesized 250 shapes for each of the four types of shafts (with each type having 2, 3, 4, and 5 sections, respectively), totaling 1,000. For the other four components, 1,000 shapes for each are created. The dimensional information of these shapes was recorded alongside the GT 3D models. A piece of the dimensional information of a nut is given in the figure.

The 2D image representation of these 3D shapes is obtained from computer rendering. Subsequently, sketches of these images were produced by sketch-style rendering using OpenCV. Although hand sketches of mechanical components from human participants would be a better data source for research validity, the efficiency and effectiveness of rendered sketches from computer algorithms have been demonstrated (Manda et al., 2021). With the consideration of such trade-offs, we decided to use computer renderings for the sketch data.

6.3.1.2 Textual Descriptions With Dimensional Information

It is feasible to synthesize textual data integrated with dimensional information from images via GPT models (Luo et al., 2024). However, as we need to input this textual data into GPT models for analysis, it might introduce a risk of biasing the results. To that end, we tested the other popular automatic captioning methods, such as the Contrastive Language-Image Pretraining (CLIP) model (Radford et al., 2021), but found the results unsatisfactory for mechanical components.

[2] `https://cadquery.readthedocs.io/en/latest/installation.html`



Your task:

Examine the **shaft** shown in the provided image and provide a description in your own words that incorporates the dimensional information listed beneath the image. Your description should focus on the object itself rather than the image. For instance, instead of saying 'The image shows an object on a white background,' detail the object's features as they appear.

Shaft:

Dimensional information (length L, diameter D):
Section 1: L=16.9 mm, D=5.9 mm.
Section 2: L=15.5 mm, D=3.5 mm.
Section 3: L=21.2 mm, D=6.9 mm.

Example of an excellent answer:

Excellent answer: It is a rectangular block 80mm x 60mm x 10mm, with counter-bored holes for M2 socket head cap screws at the corners, and a circular pocket 22mm in diameter in the middle for a bearing.

Your answer for the shaft here:

Figure 6.3: An example of the HITs on Amazon Mechanical Turk

To mitigate this and ensure the quality of the textual data, we chose to crowdsource textual descriptions through Amazon Mechanical Turk (AMT), a platform renowned for its efficacy in gathering data across a broad demographic spectrum. This diversity, spanning geographical, cultural, and age-related differences, is crucial for the richness of our dataset and aligns with established precedents in engineering design research for collecting data on human subjects (Mason and Suri, 2012; Lopez et al., 2018). We designed human intelligence tasks (HITs) on AMT to recruit participants for our study. These individuals were instructed to describe mechanical components in natural language based on provided images. A critical requirement of these descriptions was the incorporation of specific dimensional information, which was presented alongside the images. This approach ensures that our data collection method not only captures the varied interpretations of mechanical components but also includes precise dimensional information, enhancing the utility and accuracy of the dataset.

For the five distinct mechanical component categories — shafts, nuts, flanges, springs, and gears — each category is represented by a unique standardized image for visual depiction within a HIT. Specifically, the category of shafts is further distinguished by incorporating four separate HITs and each HIT with a standardized image. These images correspond to the four distinct types of shafts, which are categorized based on the number of sections they contain. Thus, eight HITs were created and published, corresponding to the four types of shafts and the other four mechanical components as aforementioned. We published 1000 assignments for each category of the mechanical components (250 for each of the 4 HITs of shafts (250×4) and 1000 for each of the other 4 HITs). Within a single HIT, every assignment featured the same image with its dimensional information. According to the rules of AMT, once a participant completes an assignment within a HIT, they cannot work on the other assignments within the same HIT, thereby avoiding repetitive responses. An example of an assignment under the HIT for triple-section shafts is shown in Figure 6.3. Accompanying each image, a piece of dimensional information describing the component is provided. Annotations are included on each image to highlight key features of the mechanical components to clarify the relationship between the dimensional information and the component's features. Furthermore, an example of a bearing pillow block with a human's description incorporating dimension information is provided as a reference to aid participants in understanding the task's requirements.

After completing the data collection via AMT, we conducted a cleaning process for the textual data to ensure the accuracy, consistency, and relevance of the information provided by the participants. Approximately 67% of the responses were deemed to be of high quality. The final dataset included a collection of textual descriptions: 628 for shafts, 671 entries for nuts, 692 for flanges, 679 for springs, and 661 for gears. After cleaning, we replaced the generic dimension information within the textual descriptions with specific, accurate specifications paired with the corresponding mechanical components. The integration of dimension information is expected to significantly enhance the richness and applicability of our dataset. The statistics of textual data and representative samples are presented in Table 6.1.

6.3.2 Code Generation and Debugger

We show the pipeline of the Code Generation, and Evaluation processes in Figure 6.4. The experiment was conducted by utilizing the models' application programming interface (API) and instructing them to generate CAD program code via CADQuery. Similar to the generation of GT 3D shapes, we employed Version 2.3.1 for CADQuery here as well. To interact with the OpenAI API model, we assign a persona to it, defining it as an AI assistant specialized in designing 3D objects with CadQuery. We initiate the request with a combination of a description and a specific prompt. The given prompt instructs: "Generate CadQuery code to construct the specified mechanical component. The code must exclusively utilize CadQuery and can not incorporate any other CAD design packages or software, ultimately exporting the component as an STL file." The resulting CAD program was subsequently converted into 3D shapes for analysis.

Table 6.1: Statistics of the textual data collection and cleaning process accompanied with representative examples

	Published tasks	**Filtered responses**	**Examples**
Shafts	1000	628	It is a shaft which have four section. In first it have 14.9mm length and 15.5mm diameter. In second, it have12.8mm length and 21.3mm diameter. In third, it have 20.3mm length and21.1mm diameter. In fourth it have 25.6 mm length and 2.6 mm diameter.
Nuts	1000	671	It is a hexagon nut with an external diameter of 47mm, a nominal hole diameter of 7mm and a height of 14mm.
Flanges	1000	692	It is a Flange which has 124mm diameter and 14mm thickness with raised face which has 86mm diameter and 16mm bore diameter with 144mm face height.
Springs	1000	679	The spring is a coil with a diameter of 8mm and a pitch of 14mm. It's 46mm long when uncompressed, made of wire with a 1.5mm radius. It seems strong and flexible, suitable for many uses.
Gears	1000	661	It is a gear which has 6 module and 44 teeth number with face width is 8.7mm and a bore diameter which is equal to 19.3mm.

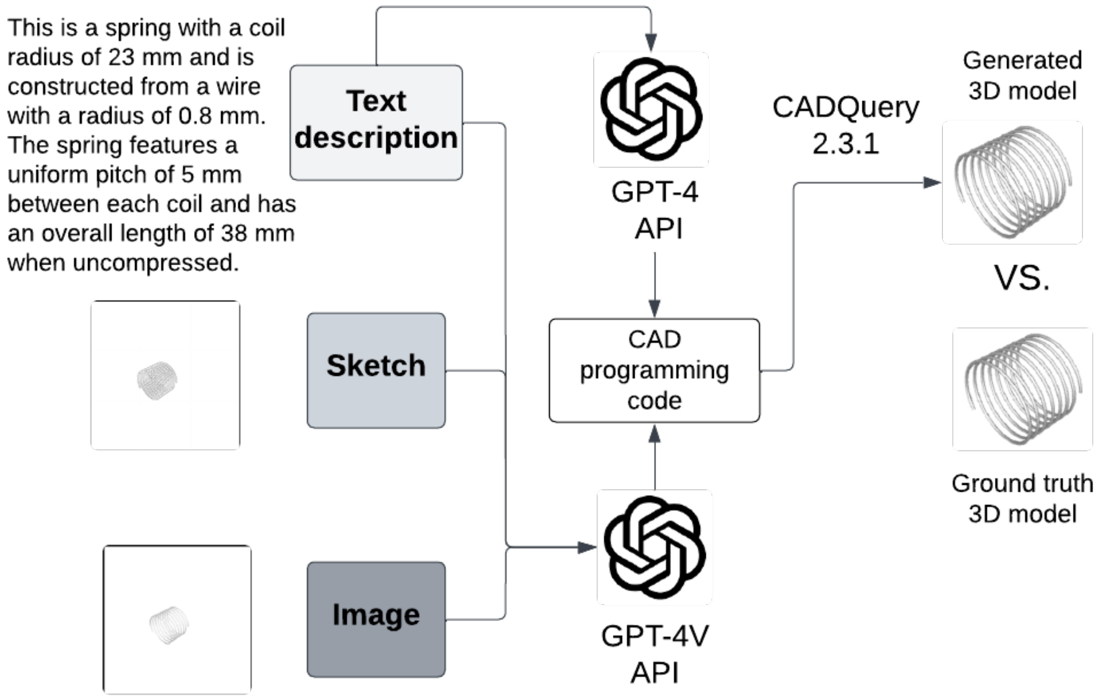

This is a spring with a coil radius of 23 mm and is constructed from a wire with a radius of 0.8 mm. The spring features a uniform pitch of 5 mm between each coil and has an overall length of 38 mm when uncompressed.
Text description
Sketch
Image
GPT-4 API
GPT-4V API
CAD programming code
CADQuery 2.3.1
Generated 3D model
VS.
Ground truth 3D model

Figure 6.4: The pipeline for Code Generation, and Evaluation

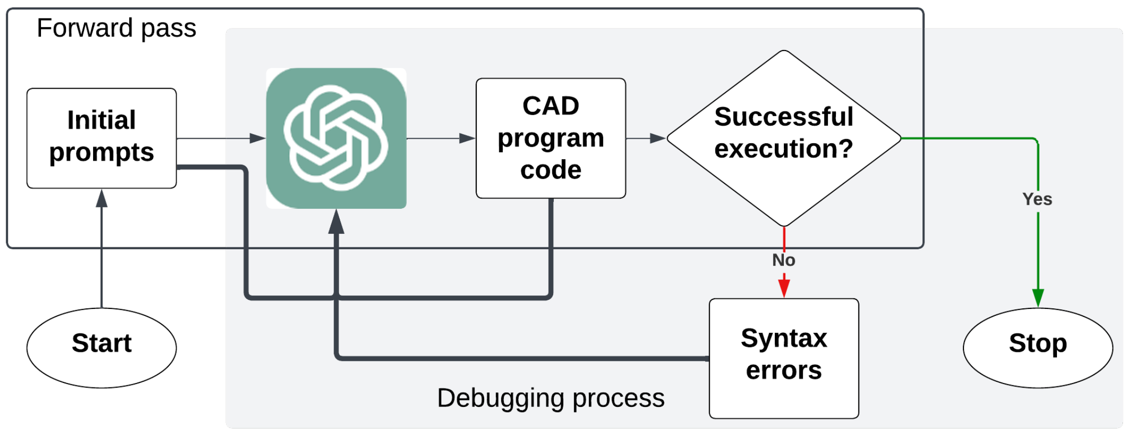


Figure 6.5: The process of generating CAD program code using GPT models with a debugger that can iteratively correct the syntax errors (if any) of the CAD program code. The bolded lines indicate the information flow of the debugging process.

To enhance the quality of the GPT models' output, we proposed a debugger as shown in Figure 6.5 integrated with the "forward pass" as described previously. The initial prompts (e.g., textual, image, and sketch data) conceivably represent user inputs, commands, or parameters that directly influence the code synthesis mechanism. The "forward pass" ends after executing the generated CAD program code no matter if the execution is successful or not, which is used to test the inherent capability of GPT models. For the "debugging process," the code is subjected to an execution trial to ascertain its functional integrity. In the event of a successful execution, the process will be terminated. Conversely, an unsuccessful execution indicates the presence of syntax errors within the code, requiring the activation of the debugger. Syntax errors encompass a spectrum of programming language misuse, such as typographical errors to the misapplication of language constructs. The "debugging process" is an iterative procedure dedicated to the identification and correction of errors in the code. Both the previous conversation content (including the user requirements and GPT's responses) and the associated error messages are fed to the same API for the "debugging process". This recursive process is imperative to refine the CAD program code, ensuring its accuracy and reliability before finalization.

6.3.3 Evaluation

We employed two key metrics to quantify the capabilities of LLM4CAD: the parsing rate for Cap1 and the intersection over union (IoU) for Cap2. The parsing rate metric evaluates the extent to which the generated CAD program code could be parsed successfully without errors, acknowledging that generating error-free code by GPT models is not guaranteed. Upon successful parsing, the quality of the resulting 3D shapes is measured against the GT shapes by calculating the IoU, thus providing a quantifiable measure of the generation accuracy relative to the input modality. The IoU metric, a critical measure of accuracy, quantifies the overlap between the generated shape and the GT shape as a ratio of their intersection to their union. This metric is widely used and particularly insightful for evaluating the geometric fidelity of the generated designs relative to the GT. Given our focus on the geometry of the generated shapes rather than their positions within a given space, we implemented a pre-step to align the principal axes of the generated shapes with those of the GT shapes by rotation and translated the generated shapes to align their centroids with those of the GT shapes. This transformation process ensures that the calculation of IoU is based only on the geometric accuracy of the shapes, excluding any discrepancies that might arise from their positioning or orientation.

6.4 Experiments and Results

In this section, we introduce the experiment details and the results.

6.4.1 Experiment Details

The details of our experiment settings are outlined in Table 6.2. We conducted a comparative analysis between GPT-4 and GPT-4V. The API models "gpt-4-1106-preview" and "gpt-4-1106-vision-preview" were employed for GPT-4 and GPT-4V,

Table 6.2: Details of the experiment settings

Mechanical components	**API**	**Input modality**	**Model**	**Metrics**
3D shapes • Shaft (4 types) • Nut • Flange • Spring • Gear	gpt-4-1106-preview	Text	GPT-4 model	Parsing rate and IoU
			GPT-4 model + Debugger*	
	gpt-4-1106-vision-preview	Text	GPT-4V model	
		Text + sketch		
		Text + image	GPT-4V model + Debugger*	
		Text + sketch + image		

*: Debugging times = 3

respectively [3]. These represented the most up-to-date versions of the API available at the time of our study. While GPT-4 accepts only textual input, GPT-4V can process both textual and rasterized data inputs. We explored various modalities and combinations thereof as inputs for the GPT-4V model. In theory, there are other possible input modes (such as sketch only and sketch + image). However, they do not provide dimensional information from the textual descriptions for the GPT models and cannot fulfill our objective of conducting a quantitative comparison between the generated 3D design objects and their GT counterparts. As a result, our selection was strategically narrowed down to input modes that include textual descriptions, ensuring the necessary dimensional data is available for accurate analysis and comparison. In both scenarios, we first assessed GPT models' inherent capabilities, followed by the implementation of the debugger to evaluate its effectiveness in improving model performance. Specifically, we limited the debugging process to three times in the current study.

[3] `https://platform.openai.com/docs/models/overview`

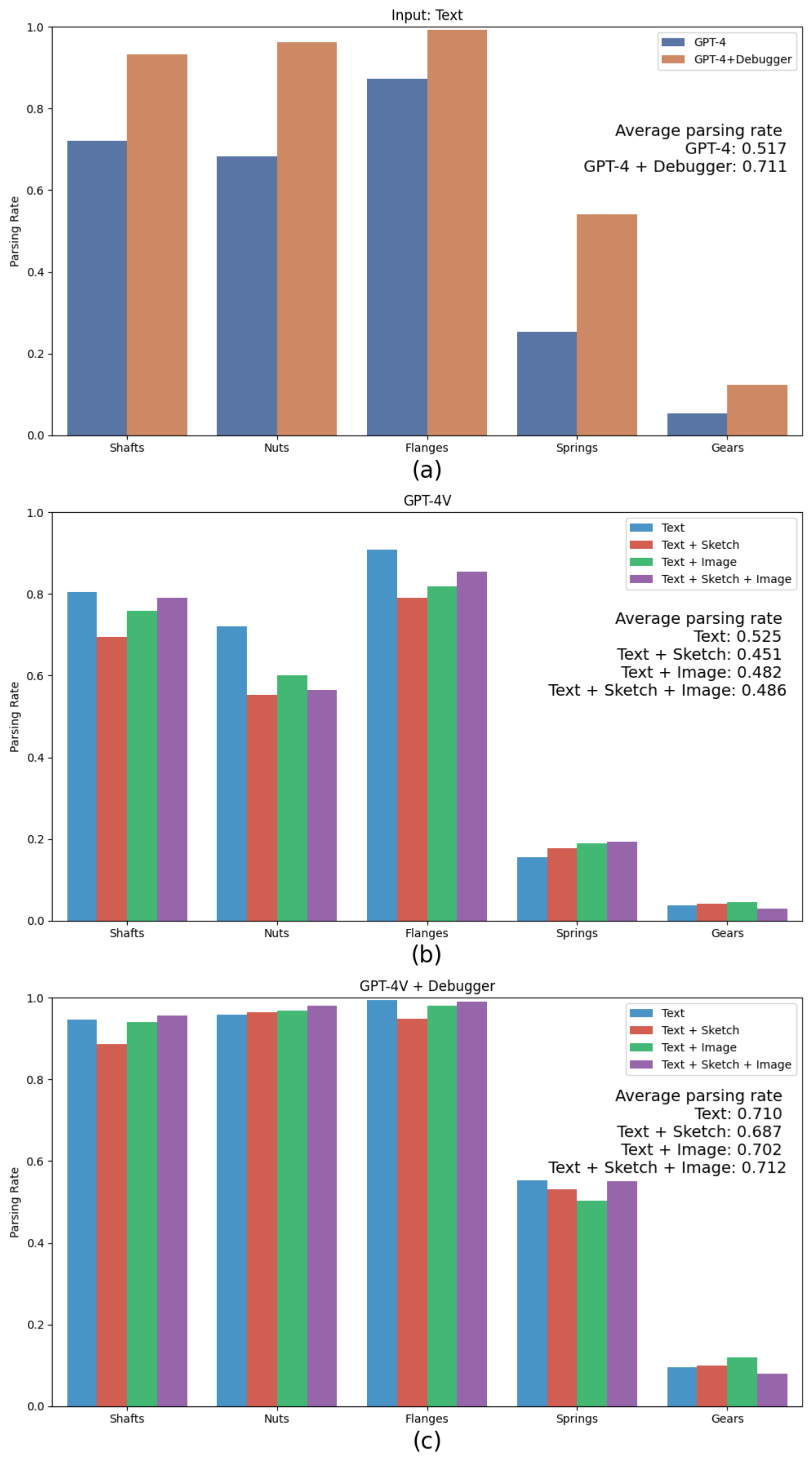


Figure 6.6: Results of the parsing rate for the five categories of shafts, nuts, flanges, springs, or gears. (a) GPT-4 VS. GPT-4 + Debugger with text-only input; (b) GPT-4V model; (c) GPT-4V + Debugger.

6.4.2 Results

In this section, we present the results of the parsing rate and IoU. Additionally, we show examples to qualitatively compare the generated 3D design objects with their corresponding ground-truth (GT) shapes.

6.4.2.1 Results of the Parsing Rate

Figure 6.6 shows the results comparing the parsing rates of the GPT-4 and GPT-4V models in various categories of mechanical components with or without the debugger. The average parsing rate values of both models are also annotated in the figure. Overall, there is a variance in model performance relative to the complexity of the mechanical components being parsed. Both models demonstrate higher efficacy in generating code for simple geometries, such as shafts, nuts, and flanges, than complex geometries (e.g., springs and gears).

Figure 6.6 (a) details the performance of the GPT-4 model when processing text inputs. It is observed that the inclusion of a debugger significantly enhances the model's parsing rate. In Figure 6.6 (b), the analysis extends to the GPT-4V model dealing with multiple input modalities, including text-only, text with sketch, text with image, and a combination of text, image, and sketch. In terms of the average parsing rate, the GPT-4V model achieves its highest performance with the text-only input mode, while the results are relatively consistent across the other three input types. For each category of the mechanical components, the text-only input achieves the best in shafts, nuts, and flanges. However, when dealing with more complex geometries (e.g., springs and gears), multimodal input modes are better than or as good as the text-only input. For example, the input of text with image is the best in gears, and the input using the combination of text, sketch, and image achieves the best in springs.

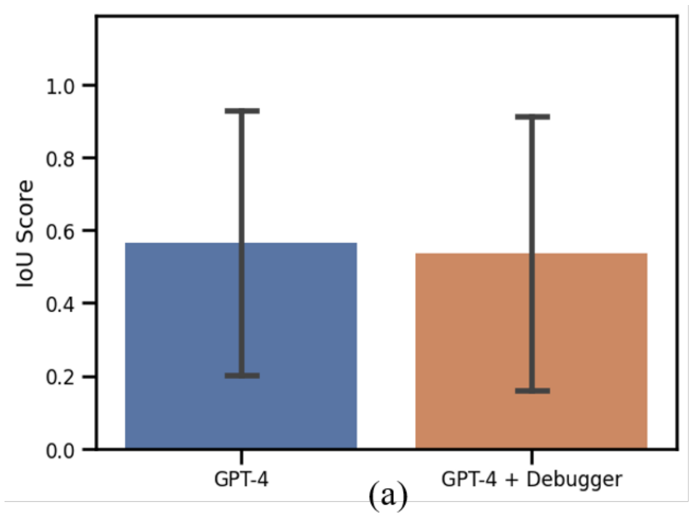
The result of the comparison between Figures 6.6 (b) and (c) mirrors the trend observed in the GPT-4 model, demonstrating an improved parsing rate with the introduction of a debugger. In addition, Figure 6.6 (c) shows the difference in the average parsing rate across the four input modes is reduced and these values are approaching a similar level. In addition, for the category-specific results, the four input modes achieve similar levels of parsing rate values for each category. The pattern that the text-only input is dominant over the other three input modes for simple geometries as observed in Figure 6.6 (b) no longer holds. Multimodal input modes become more effective for both simple and complex geometries by introducing the debugger.

A comparative analysis focusing on text-only inputs between the GPT-4 and GPT-4V models indicates a significant difference in performance. Specifically, the GPT-4V model exhibits a higher average parsing rate (0.525) compared to its GPT-4 counterpart (0.517). However, this advantage diminishes upon the integration of a debugging process (0.711 for GPT-4 VS. 0.710 for GPT-4V).

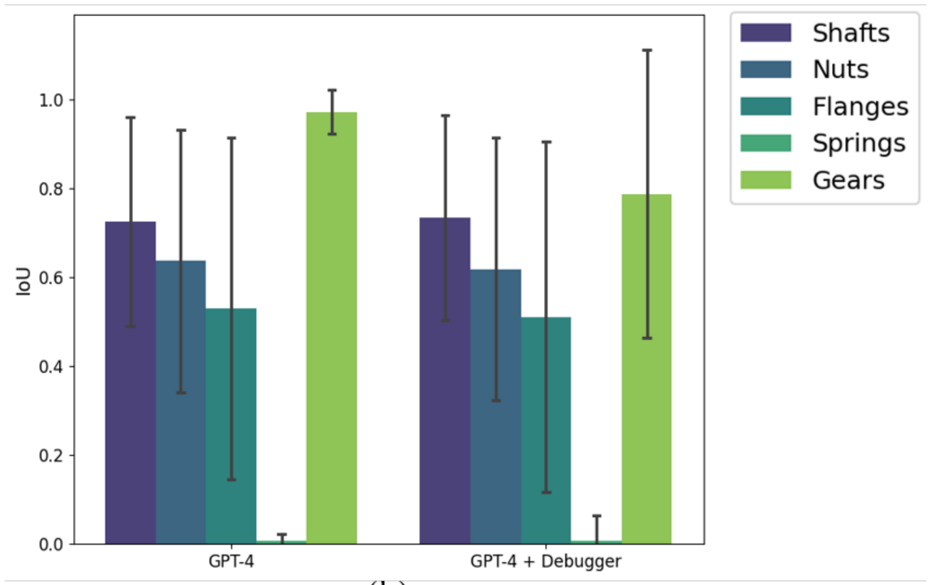
6.4.2.2 Results of the Intersection Over Union

Figure 6.7 shows the performance evaluation of the GPT-4 model, presenting (a) an overview of the average performance and (b) a detailed breakdown of the performance by component category. In terms of overall performance shown in Figure 6.7 (a), there is a significant decrease in the average IoU upon inclusion of the debugger with statistical analysis (P-value=0.013 obtained from an independent T-test). On the other hand, when examining the performance across specific component categories in Figure 6.7 (b), the P-values are 0.511, 0.321, 0.378, 0.951, and <0.01 for shafts, nuts, flanges, springs, and gears, respectively as summarized in Table 6.3. Significant difference is only detected for gears but there is no significant difference for the other four categories.

Figure 6.8 (a) provides a comprehensive summary of the GPT-4V model's overall performance across distinct input modalities. As summarized in Table 6.3 (a), when evaluating the impact of the debugger for each input mode, significant



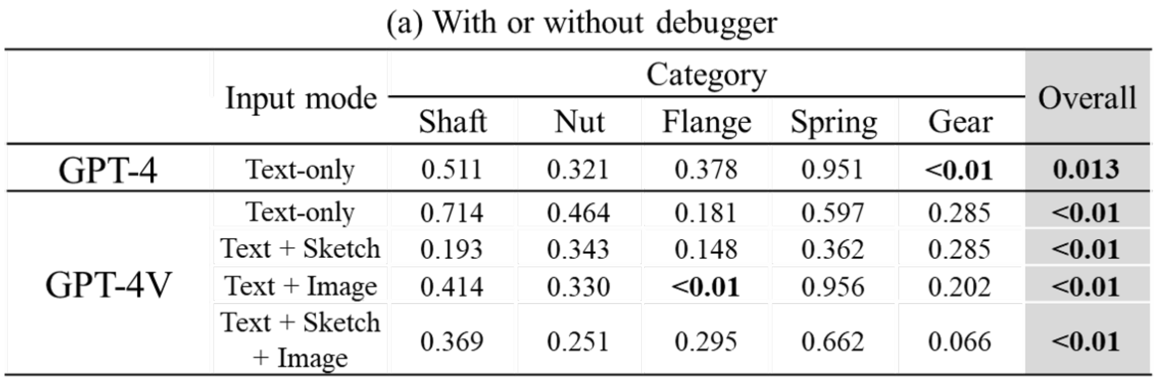
(a)



(b)

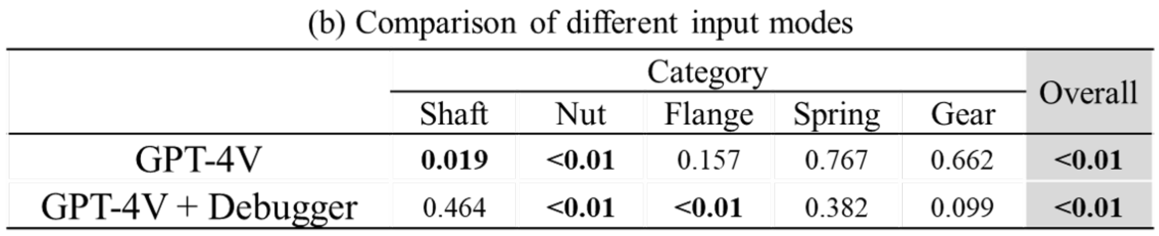
Figure 6.7: Results of IoU of (a) the overall performance and (b) the category-specific performance of the GPT-4 model

Table 6.3: Summary of the P-values of statistical analysis for the IoU values: (a) T-test for the effects of including the debugger for either GPT-4 or GPT-4V, (b) One-way ANOVA for the effects of different input modes when using GPT-4V with or without the debugger



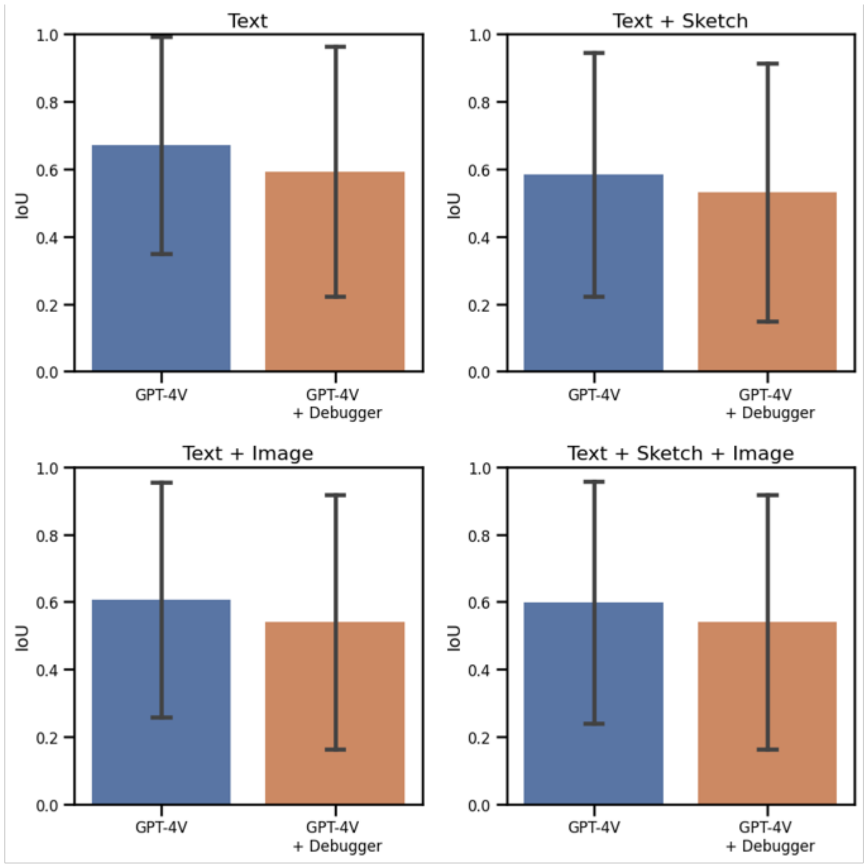
(a) With or without debugger

	Input mode	Category					Overall
		Shaft	Nut	Flange	Spring	Gear	
GPT-4	Text-only	0.511	0.321	0.378	0.951	**<0.01**	**0.013**
GPT-4V	Text-only	0.714	0.464	0.181	0.597	0.285	**<0.01**
	Text + Sketch	0.193	0.343	0.148	0.362	0.285	**<0.01**
	Text + Image	0.414	0.330	**<0.01**	0.956	0.202	**<0.01**
	Text + Sketch + Image	0.369	0.251	0.295	0.662	0.066	**<0.01**

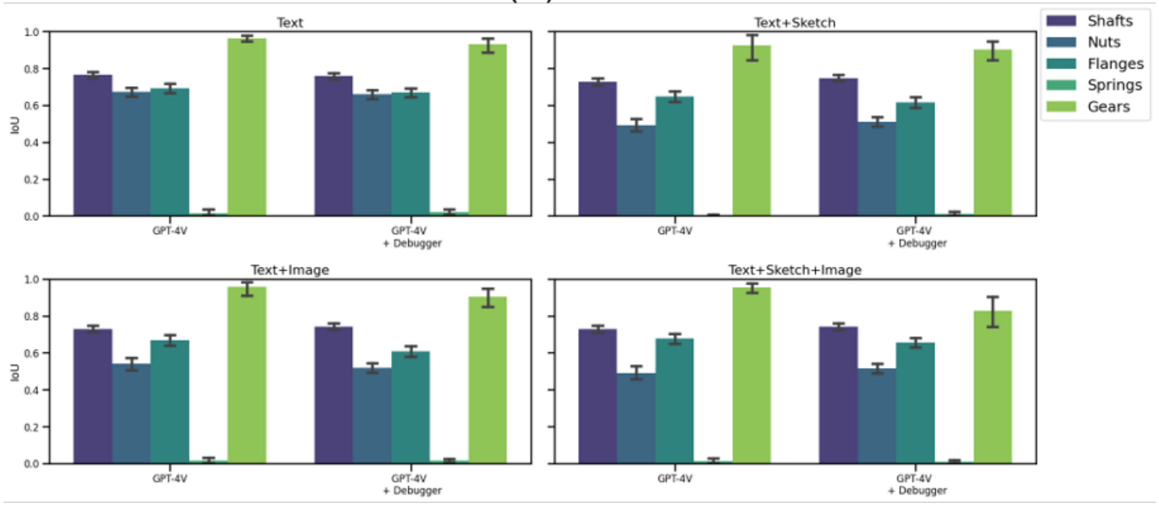


(b) Comparison of different input modes

	Category					Overall
	Shaft	Nut	Flange	Spring	Gear	
GPT-4V	**0.019**	**<0.01**	0.157	0.767	0.662	**<0.01**
GPT-4V + Debugger	0.464	**<0.01**	**<0.01**	0.382	0.099	**<0.01**



(a)



(b)

Figure 6.8: Results of IoU of (a) the overall performance and (b) the category-specific performance of the GPT-4V for four input modes

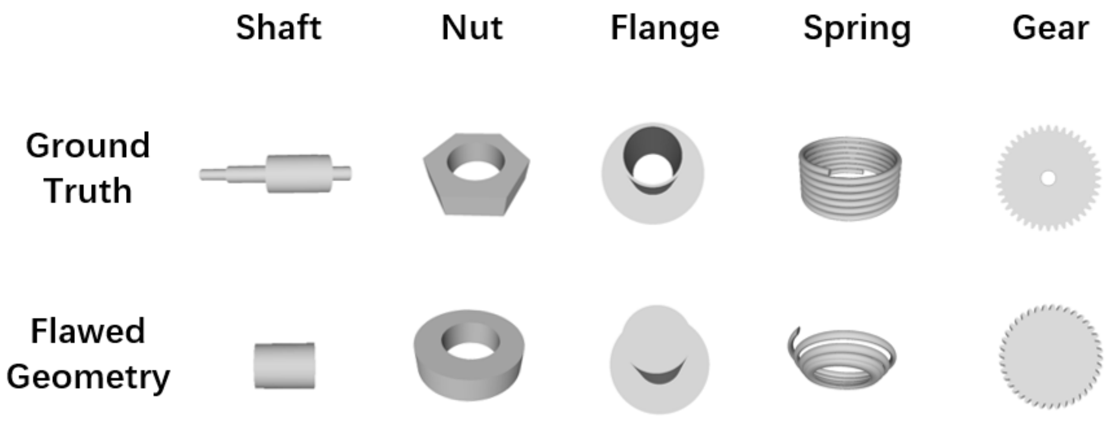


Figure 6.9: Examples of flawed geometry generated by the GPT-4V model

differences are observed between GPT-4V and GPT-4V + Debugger (all P-values < 0.05 using an independent T-test). This result suggests that the introduction of the debugger may inadvertently affect the precision of generated 3D design objects. However, in general, there is no significant difference when examining the category-specific results, except for the following two cases: (1) gears for GPT-4 and (2) flanges for GPT-4V when using the text+image input mode.

Additionally, as shown in Table 6.3 (b), a one-way ANOVA is applied to the GPT-4V results, revealing a statistically significant difference (P-value < 0.01) between the different input modes. Subsequent pairwise comparisons were conducted using Tukey's Honestly Significant Difference (HSD) test to pinpoint the specific modalities that exhibit significant discrepancies. The analyses indicate that the text-only input mode achieved higher IoU values compared to the other three input modes. This trend persists after integrating a debugger into the GPT-4V model, further solidifying the text-only mode's superior performance. Nonetheless, when examining the category-specific results, except for certain simple geometries (e.g., nuts for both GPT-4V and GPT-4V + Debugger, shafts for GPT-4V, and flanges for GPT-4V + Debugger) where the text-only input achieves significantly higher IoU values indicated by the Tukey's HSD tests. Especially, for more complex geometries (i.e., springs and gears), text-only input is not superior to the multimodal input modes.

Furthermore, a comparative analysis between the GPT-4 and GPT-4V models focusing on the text-only input mode indicates the GPT-4V model exhibits a significantly higher IoU score compared to GPT-4 (P-value$<$0.01). This trend persists even when a debugger is incorporated (i.e., GPT-4 + Debugger VS. GPT-4V + Debugger).

6.4.2.3 Qualitative Results

Figure 6.9 presents the flawed geometries generated by the GPT-4V model within five distinct component categories compared to the GT shapes. For shaft components, the issue is the exclusion of multiple shaft sections. In the context of nuts, the prevalent error consists of producing a circular nut instead of the specified hexagonal configuration. This issue could stem from the GPT models' limitations in generating CAD programs that require a sequence of precise operations. For instance, forming a hexagon needs six distinct steps involving the *Line* operation, with a specific angle between each segment. This process demands a high degree of accuracy and an understanding of geometric principles that may be difficult for GPT models to replicate. In contrast, GPT models may find it much easier to utilize a *Circle* operation to create the base shape so they "slept on the job". For flanges, the geometric fault is the omission of the flange's borehole. In the case of springs, the error commonly observed is the improper formation of the helix. Similarly, gears exhibit an issue similar to that of the flange components, characterized by the loss of the borehole.

The results highlight the models' current limitations in handling tasks that require detailed procedural generation and a deep understanding of spatial relationships. They might intentionally return incorrect CAD programs due to the difficulty in returning the correct ones. Improving their capability to accurately execute complex sequences of operations such as those needed for detailed CAD modeling remains an area for further development.

6.5 Discussion

Based on the observations of the results, we extend our discussion to three aspects: 1) the effects of multimodal input for GPT-4V; 2) GPT-4 VS. GPT-4V; and 3) the effects of the debugger in 3D CAD generation. Furthermore, we acknowledge the limitations of our study and propose potential avenues for future research.

6.5.1 Effects of Multimodal Input for GPT-4V

6.5.1.1 Effects on IoU

For the IoU outcomes of GPT-4V shown in Figure 6.8, the statistical analyses on the overall performance in Table 6.3 reveal that the text-only input mode outperforms the other three modes with input modalities of text+sketch, text+image, and text+image+sketch. This trend was also observed for the integration of the debugging process, further underscoring the superior efficacy of the text-only input mode. This observation challenges our assumption on multimodal machine learning (MMML) that integrating various input modalities enhances the predictive capabilities of the ML models. The possible explanations for this result may be based on the following three aspects.

First, the simplicity of text-only data might help reduce computational burden and noise, leading to more efficient processing and accurate results. On the one hand, this implies that, under certain conditions, the advantage of multimodal might be negated by the associated data complexity. On the other hand, it implies that textual descriptions, especially those that include dimensional information, can provide substantial and adequate information for the GPT-4V model to "comprehend" the design requirements of mechanical components. Second, the hypothesis that integrating various input modalities could improve the predictive performance of machine learning (ML) models may be contingent upon how relevant the information from these modalities is to the design target, how precise it is, and how well the model can interpret the data. In our study, it is possible that the GPT-4V model

Table 6.4: Summary of the parsing rate and IoU for GPT-4 and GPT-4V models using the text-only input

	GPT-4 VS. GPT-4V (Average across five categories)	
	Parsing rate	IoU
Inherent capability	0.52 VS. **0.53**	(0.57±0.36) VS. (**0.67±0.32**)
Enhanced capability with the debugger	0.71 VS. 0.71	(0.54±0.38) VS. (**0.59 ±0.37**)

mistakenly processed information from images or sketches. In fact, we undertook qualitative analyses, examining 10 images of each mechanical component through the GPT-4V API. These experiments revealed occasional misinterpretations, such as recognizing a three-section shaft as a two-section shaft. Third, it appears that the GPT-4V model is naturally more proficient at processing textual data than image data. This is particularly true in tasks that require precise spatial localization and perspective relationships. For example, GPT-4V often generates a tapered spring due to the effect of perspective in the rendered image. This suggests a possible limitation in the model's ability to accurately interpret renderings of mechanical components compared to its success with more commonly represented objects such as humans or vehicles (Luo et al., 2024).

Nonetheless, the category-specific results indicate that as the geometries of the CAD models become more complex, multimodal input modes become more effective. Considering the real-world design scenarios where the products are more complex in terms of both geometries and structures, multimodal input modes are expected to behave better than text-only input modes. Exploring this further would present an intriguing avenue for future research.

6.5.1.2 Effects on Parsing Rate

Similar to the IoU outcomes, the text-only input mode surpasses the three multimodal input modes in terms of the overall average parsing rate. However, this trend diverges when examining results specific to different categories of mechanical components. For more complex mechanical geometries (e.g., springs and gears), multimodal inputs demonstrate an advantage, either matching or exceeding the performance of text-only input. For instance, as shown in Figure 6.6 (b), the text+sketch+image input is the best for springs, but text+image turns out to be the most effective input for gears. This emphasizes the value of incorporating visual information alongside textual data in improving the model's efficiency in parsing complex geometries.

With the addition of the debugger to the GPT-4V model, the efficacy of multimodal inputs aligns more closely with that of text-only input for each category and in terms of the overall result compared to the result obtained without using the debugger. This observation indicates the debugger's potential to amplify the model's proficiency in utilizing visual data, particularly in handling complex geometries.

6.5.2 GPT-4 VS. GPT-4V

The results of our experiments revealed the advantage of the GPT-4V model over the GPT-4 model in processing text input, both for the inherent and enhanced (i.e., with the debugger) versions, in the generation of 3D CAD models. This superiority is evidenced in terms of a comparable parsing rate but much higher IoU scores, as summarized in Table 6.4. According to the GPT-4V system card, OpenAI's official evaluation report of GPT-4V (OpenAI, 2023a), GPT-4V is built upon the GPT-4 architecture as quoted here: "As GPT-4 is the technology behind the visual capabilities of GPT-4V, its training process was the same." Furthermore, the API models for both GPT-4 and GPT-4V utilized in our study share an identical knowledge cutoff date of April 2023. Given that GPT-4V is designed to accommodate visual inputs alongside textual data, its performance in processing text is anticipated to be comparable to that of GPT-4. Nonetheless, the reason for the observed differences in their performance is not clear at this stage. In addition, the absence of comparative studies in the literature specifically addressing the text-processing capabilities of GPT-4 and GPT-4V underscores further research's need to clarify these differences.

6.5.3 Effects of the Debugger

Figure 6.6 demonstrates that the integration of the debugger enhances the parsing rates for each category of the mechanical components and in terms of the overall average performance for both GPT-4 and GPT-4V models, underscoring the debugger's efficacy in iteratively correcting syntax errors within the generated CAD program codes.

However, this enhancement in parsing rate comes at a cost of the reduction in IoU values for both models as detailed in Figures 6.7 and 6.8 in terms of the overall average result although this reduction is not generally observed for the category-specific results.

This decrease suggests that the GPT models used with a debugger may prioritize the correction of syntax errors and compromise on accurately fulfilling the design requirements (i.e., text, images, or sketches). This hypothesis is supported by our qualitative experiments with the ChatGPT Pro version which is built on the GPT-4 model. We observed instances where, in the process of debugging syntax errors, ChatGPT prioritized correcting the CAD program code over sticking to the design requirements, despite having access to the entire conversation history. This resulted in an oversimplification of the code, which may ultimately lead to incorrect geometry (e.g., generating a round nut for the required hexagonal nut). Addressing this limitation requires future research to improve the debugger's functionality to balance between syntax correction and simplification. However, this does not mean that the GPT models would totally ignore the design requirements. We conducted experiments using a Version 2 Debugger where we only fed the synthesized CAD programs back to the GPT models during the debugging process without giving the input modalities. We got much lower parsing rate values and IoU values compared to the Debugger we proposed to use. This also provides a practical insight that the input modalities should be fed to the GPT models when designing the Debugger. Furthermore, the debugger currently relies on the zero-shot learning capability of LLMs, which means that it operates without specific instructions other than being asked to fix syntax errors based on its own understanding. Future work in this area aims to identify the most common errors by analyzing the responses from experiments and directing the debugger to focus specifically on resolving these issues in order to improve the debugging success rate. Additionally, there is potential to fine-tune LLMs to excel in correcting errors in CAD programs.

While the debugger presents a viable strategy for GPT model enhancement, alternative approaches, including model fine-tuning and the incorporation of function calls, could be potential ways to advance the application of GPT models in 3D CAD generation. Model fine-tuning is to adjust a GPT model by further training it on a specialized dataset, such as the multimodal CAD dataset proposed in this study, to enhance its ability to perform 3D CAD generation. In addition, function calls involve generating output by calling existing functions (e.g., the Python class for creating a shaft) to create 3D shapes.

6.5.4 Limitations and Future Work

In this study, we evaluated five representative categories of mechanical components with different geometric complexities. Although the insights gained from the current synthesized dataset are valuable, we acknowledge that the sample size is relatively small compared to the wide array of mechanical components. To obtain an in-depth understanding of the role that multimodal LLMs play in the generation of 3D CAD models, an expansion of the CAD dataset is essential. It is also critical to note that designs often consist of interconnected components in the form of assemblies rather than individual components (Li et al., 2023c, 2021c). This requires improvements in the current data synthesis pipeline, specifically the inclusion of additional categories of CAD models and the capability to synthesize system design objects. In addition, the examination of additional CAD programming languages, such as OpenSCAD and Fusion 360 Python API, as well as other LLMs, such as Google's Gemini, will help gain more comprehensive views on the capability of multimodal LLMs in CAD generation of 3D shapes.

In addition to design concept generation, LLMs can also be utilized for various manufacturing purposes (Picard et al., 2023). One such application is in material selection, where LLMs can assist in identifying and evaluating suitable materials based on specific requirements such as strength, durability, and cost-effectiveness. Furthermore, LLMs can also be employed to explore different ways to manufacture products, including advanced techniques such as additive manufacturing or traditional methods such as casting and machining. By leveraging LLMs for these purposes, manufacturers can streamline the decision-making process and optimize their production processes, ultimately leading to improved efficiency and quality in manufacturing operations. However, as demonstrated in the work (Picard et al., 2023), while LLMs have potential, their inherent capabilities (e.g., zero-shot learning capability) are still limited. Future research should focus on developing methods such as fine-tuning LLMs using multimodal datasets that incorporate features (e.g., material properties and manufacturing methods). This exploration is worthwhile for building valid and effective manufacturing methods using LLMs.

6.6 Conclusion

This study is motivated by answering two research questions: *1) To what extent can multimodal LLMs generate 3D design objects when employing different design modalities or a combination of various modalities? 2) What strategies can be developed to enhance the ability of multimodal LLMs to create 3D design objects?* Therefore, we first developed an approach to enable multimodal LLMs in 3D CAD generation. Then, we studied the performance of the GPT-4 and GPT-4V models with different input modalities, including the text-only, text+sketch, text+image, and text+sketch+image data.

Both GPT-4 and GPT-4V showed significant potential in the generation of

3D CAD models, especially with the enhancements enabled by a debugging process. Additionally, in our experiment of GPT-4V, we tested four input modes: text-only, text with sketch, text with image, and a combination of text, sketch, and image. Surprisingly, GPT-4V's performance with text-only input surpassed that of the other three multimodal inputs on average. This observation challenges the common belief in MMML that incorporating varied input modalities always improves a machine learning model's predictive accuracy due to increased information for learning and inference. However, when examining category-specific results of mechanical components, multimodal inputs start to gain prominence with more complex geometries (e.g., spring and gears) in terms of the successful parsing rate of the generated CAD programs and the geometric accuracy.

From these observations, we see that the current multimodal LLMs are still limited in handling multimodal inputs when applied to LLM4CAD. However, the insights from the category-specific results indicate that multimodal LLMs have great potential benefits in real-world design scenarios characterized by complex objects although it remains challenging for them to generate complex design objects. Improving the capability of these models to process diverse input modalities and proposing strategies to improve their capability to handle complex design objects are promising research avenues.

To further address the two RQs posed and achieve a comprehensive understanding, future studies should broaden the research scope to include a more diverse dataset featuring more complex 3D design objects. Strategies, including model fine-tuning and the integration of function calls, to enhance the utility of multimodal LLMs for CAD are worthy to explore. Moreover, while this study focused on the CAD generation of 3D shapes during the conceptual design phase, future research can explore other stages of the engineering design process, such as customer needs analysis, design evaluation, and manufacturing. This will contribute to a deeper understanding of how LLMs, particularly multimodal LLMs, can be employed to facilitate the overall engineering design process, thus making contributions to advanced design methodologies.

Chapter 7: Design Representation for Performance Evaluation of 3D Shapes in Structure-Aware Generative Design

Abstract

We conducted a comparative analysis to study surrogate models' performance in predicting the engineering performance (e.g., drag and lift) of 3D shapes using vectorized design representations (VDRs) from two sources: the trained latent space of structure-aware data-driven generative design (DGD) models (encoding both structural and geometric information) and an embedding method (encoding only geometric information), in order to assess the effectiveness of these VDRs. We performed the analysis in two case studies, 3D car models (with an interest in drag coefficients) and 3D aircraft models (with an interest in both drag and lift coefficients). Our results demonstrate that the use of latent vectors as VDRs can significantly deteriorate the predictive performance of surrogate models. Moreover, we observe that increasing the dimensionality when encoding 3D shapes' geometric information in the embedding method may not necessarily improve surrogate models' predictive performance, and this phenomenon is more obvious in the situation when the VDRs contain more noise.

7.1 Introduction

[1] Design researchers have applied artificial intelligence (AI) techniques to support various design activities, including design exploration and optimization, design synthesis, and the extraction of human preferences for designs to help human designers make decisions during the design process (Rahman et al., 2019; Panchal et al., 2019; Rahman et al., 2020; McComb et al., 2017; Rahman et al., 2021). Among various AI techniques, generative design (GD) techniques are receiving more attention in both industry and academic fields (McKnight 2017; Krish 2011; Matejka et al. 2018; McComb et al. 2020; Chen et al. 2020). GD is a term for a class of tools that can generate novel yet realistic designs leveraging computational and manufacturing capabilities (Shea et al., 2005). There are some widely used GD techniques, such as genetic algorithms and shape grammars (Singh and Gu, 2012). GD has been applied in several commercial CAD software, such as Autodesk Fusion 360, PTC Creo, and Siemens NX. However, current GD methods are driven primarily and solely by engineering performance, so the generated designs often do not agree with conventional aesthetics (Oh et al., 2019). Additionally, generated designs may be too complex to be created without using additive manufacturing (McKnight, 2017). These problems can be alleviated by deep generative models (Oh et al., 2019), capable of learning to produce new data given a set of training examples. State-of-the-art deep generative models, such as the variational autoencoder (VAE) (Kingma and Welling, 2013) and the generative adversarial network (GAN) (Goodfellow et al., 2014), have been applied in various fields, including computer vision, computational creativity, architecture, and engineering design (Regenwetter et al. 2022; Li et al. 2023b; de Miguel Rodríguez et al. 2020; Yi et al. 2019).

[1]This chapter has been published in the following paper in the Design Science Journal. Li X, Xie C, Sha Z. Design representation for performance evaluation of 3D shapes in structure-aware generative design. *Design Science*. 2023;9:e27. doi:10.1017/dsj.2023.25. I am the leading author of this paper and was primarily responsible for developing the methodologies, conducting experiments, analyzing the results, and writing the manuscript.

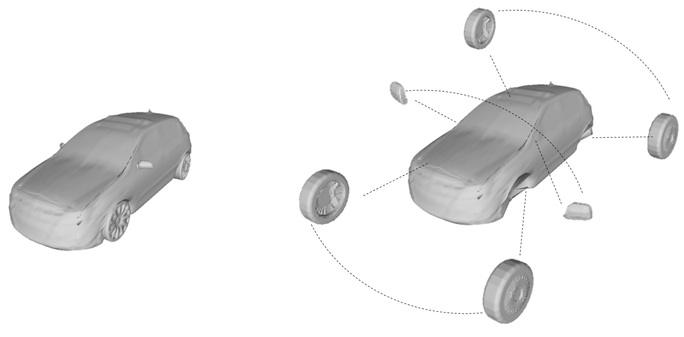


Figure 7.1: Comparison between a monolithic shape (a) and a structure-aware shape (b), where dash lines indicate structural interdependencies (i.e., support and symmetry) between components.

In the design literature, these deep generative models are often referred to as data-driven generative design (DGD) methods. DGD methods have been increasingly used to improve design creativity and facilitate conceptual design, such as airfoil design (Chen et al. 2020; Chen and Ahmed 2021), car wheel design (Oh et al. 2019; Yoo et al. 2020), and car shape design (Li et al. 2021c, 2022c). DGD methods can learn to synthesize designs from data without explicit human configuration by training a deep neural network model and learning a latent vector space with a predefined (often reduced) dimensionality. Such a latent vector space is a low-dimensional representation of the design space from which the data were observed. Since the training process combines features from all existing designs, new designs that are not seen from existing data can be sampled from the latent design space (Krish, 2011; Cunningham et al., 2020). Therefore, DGD methods have become an important tool for the generation of conceptual design ideas due to their ability to quickly generate a large number of novel designs.

Traditionally, most DGD methods treat each design data as a monolithic whole (i.e., one single object without considering the interconnections of components as shown in Figure 7.1(a)) for model training (Shu et al. 2020). In this paper, we refer to them as traditional DGD methods. Recently, there have been emerging interests in developing structure-aware DGD methods (Gao et al. 2019b; Mo et al. 2019a; Chen and Fuge 2019; Li et al. 2021c). Compared to traditional DGD methods, structure-aware DGD methods can handle complex geometries consisting of interconnected components and learn interdependencies between components (i.e., structure-aware shapes as shown in Figure 7.1(b)) to enable automatic assembly of deep-generated components in a system (Gao et al., 2019b). It can also provide designers with increased flexibility to make local design modifications by altering or substituting individual parts.

To evaluate the engineering performance of designs from DGD methods, there are two different ways. One is to conduct high-fidelity simulations, for example, based on computational fluid dynamics (CFD) or finite element analysis (FEA). However, the downside of these simulations is the high computational cost. For example, assessing the aerodynamic performance of a 3D car model using CFD software could take hours to complete. Therefore, it is impractical to evaluate the vast number of design alternatives obtained from DGD methods in support of fast design decision-making. The other way is to use surrogate models that have a relatively lower fidelity but can significantly accelerate the evaluation process. Surrogate modeling is a supervised machine learning technique to approximate the output based on the labeled training dataset (i.e., pairs of inputs and their corresponding outputs) (Sun et al. 2020; Whalen and Mueller 2022). Generally, in these surrogate modeling methods, each design is represented as a fixed-length vector of design parameters, referred to as vectorized design representation (VDR). VDR enables compact encoding of complex design configurations, making it easy to process and analyze design data mathematically and computationally. As one type of VDRs, latent vectors have recently been widely adopted in design generation, evaluation, and optimization (Umetani and Bickel, 2018; Burnap et al., 2016; Li et al., 2022c; Chen et al., 2020). Latent vectors are obtained from a latent space during the training process in neural network models, such as VAEs and GANs. A latent space is often continuous and low-dimensional (compared to the dimensionality of the training data) and packs complex data distributions. Vectors in such a latent space can capture the underlying structure and important features of the training data.

Previous studies have primarily concentrated on utilizing latent vectors from traditional DGD methods as the VDR for design evaluation. Little is known about the efficacy of latent vectors acquired from the structure-aware DGD training process, which encompasses both part-to-part structural information and geometric information. The research question, therefore, arises: What would be the appropriate VDR for a computational pipeline in support of the evaluation of structure-aware deep-generated shapes? In particular, is it reliable to directly use the latent vectors readily available from the training process of a structure-aware DGD model?

To answer this question, we performed experiments to compare the performance of the latent vectors obtained from the training process of a structure-aware DGD model in predicting the engineering performance of the designs, with those obtained by embedding the generated 3D shapes (after training) using a 3D point grid (3DPG), as shown in Figure 7.2. We conducted the comparative study in two case studies: 1) predicting the drag coefficients of car designs and 2) predicting both the drag and lift of aircraft designs. Our results indicate that while latent vectors are frequently used in surrogate models, they may not be suitable when the encoded information includes factors that have minimal relevance to the engineering performance under investigation (e.g., the SPVAE vectors containing structural information in our study). Employing such VDRs can actually hinder the prediction accuracy of a surrogate model. This new knowledge is significant because a proper VDR is crucial to the accuracy of the engineering analysis and, therefore, the validity of a 3D shape and its associated economic impact. For example, the drag evaluation of car body shapes significantly influences their fuel economy estimate. With a 10% reduction in aerodynamic drag, the highway fuel economy will improve by approximately 5% and the city fuel economy by approximately 2% (ARC). Therefore, it is crucial to consider the physics underlying engineering performance metrics and select VDRs that integrate relevant information, such as geometric information, to enhance the prediction accuracy of surrogate models.

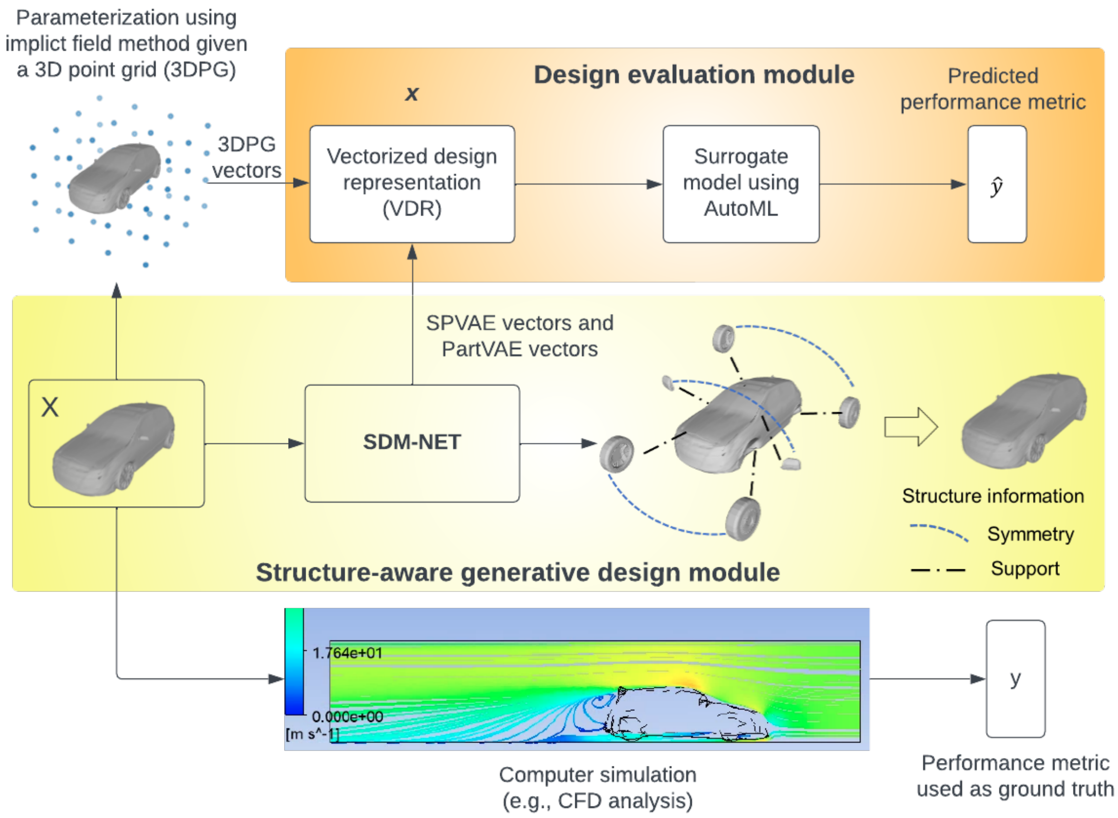

Parameterization using implict field method given a 3D point grid (3DPG)
3DPG vectors
Design evaluation module
x
Predicted performance metric
Vectorized design representation (VDR)
Surrogate model using AutoML
ŷ
SPVAE vectors and PartVAE vectors
X
SDM-NET
Structure information
Symmetry
Support
Structure-aware generative design module
1.764e+01
0.000e+00
[m s^-1]
y
Computer simulation (e.g., CFD analysis)
Performance metric used as ground truth

Figure 7.2: Overview of the research approach consisting of two key modules: the structure-aware generative design module and the design evaluation module

The remainder of this paper is organized as follows. Section 7.2 provides a review of relevant research on both traditional and structure-aware DGD methods and surrogate models. The DGD methods and surrogate models adopted, as well as the proposed research approach, are presented in Section 7.3. We then present and discuss the experimental results and summarize the main findings in Sections 7.4 and 7.5. The paper is concluded in Section 7.6, in which we summarize the closing insights and potential future research directions.

7.2 Literature Review

In this section, we present a review of the existing literature that is most relevant to this study, including data-driven generative design methods, structure-aware generative design methods, and surrogate models for design evaluation.

7.2.1 Data-Driven Generative Design Methods in Engineering design

Data-driven generative design (DGD) methods can conduct an efficient design space exploration by generating a large number of various new design concepts from existing design data (e.g., images or 3D shapes) without an explicit set of design variables (Achour et al., 2020). In engineering design, DGD methods are developed mainly based on two techniques, generative adversarial networks (GANs) and variational autoencoders (VAEs), in addition to a few others, such as recurrent neural networks (RNNs) and reinforcement learning (RL) (Regenwetter et al., 2022).

For example, focusing on 2D design applications, Oh et al. (2019) integrate a topology optimization (TO) technique with GANs to generate numerous aesthetic design options taking into account engineering performance. Their method was applied to the design of 2D car wheel rims. Chen et al. (2020) develop a Bezier-GAN model that can learn from shape variations in an existing 2D airfoil database to parameterize aerodynamic designs so that the resulting parameterization can accelerate design optimization. Dering et al. (2018) set up a physics-based virtual environment that combines an RNN model to enhance the quality of deep-generated designs of 2D cruise ships. Fujita et al. (2021) propose a framework for the generation of design concepts by applying TO and a variational deep embedding method in a 2D bridge design problem.

In 3D design applications, Shu et al. (2020) present a method that combines GANs and a physics-based virtual environment introduced by Dering et al. (2018) to generate high-performance 3D aircraft models. Zhang et al. (2019) propose a method using VAEs, a physics-based simulator, and a functional design optimizer to synthesize 3D aircraft with prescribed engineering performance. Building on the 2D wheel generative design work (Oh et al., 2019), Yoo et al. (2020) develop a deep learning-based CAD/CAE framework that can automatically generate 3D car wheels from 2D images by point extraction (i.e., to extract the points from contour lines of the wheels) and sketch extrusion.

All of the methods mentioned above address the form and functionality of designs in either 2D or 3D forms, which can help designers explore the design space by automatically generating a large number of design concepts with informed engineering performance. However, they all consider designs as one single monolithic piece and ignore the interrelations between components in a product or an assembly.

7.2.2 Structure-Aware Design Generation Methods

Acknowledging that real-world designs usually consist of multiple parts, Chen and Fuge (2019) develop hierarchical GANs to synthesize designs with inter-part dependencies. The method is demonstrated using a design case of 2D airfoils. However, 3D shapes are often the final form of most products, and structure-aware 3D design studies mostly come from the computer science community. Li et al. (2017) introduce a generative recursive autoencoder for shape structures (GRASS) based on recursive neural networks. GRASS trains independent networks for the geometry and structure of parts and generates 3D voxel shapes by producing a hierarchical series of bounding boxes filled with voxels. Nash and Williams (2017) propose a generative model of part-segmented 3D objects, namely, the shape variational autoencoder (ShapeVAE). Given a collection of dense surface points with surface normals of part-segmented objects, ShapeVAE can learn a low-dimensional shape embedding to synthesize new and realistic 3D shapes represented by point clouds, which can then be converted to 3D meshes using the surface normals. Mo et al. (2019a) introduce StructureNet, a generative autoencoder that learns shape structure using graph neural networks. StructureNet uses graphs to encode hierarchical representations of shapes. After training, it can generate 3D shapes formed by box structures or 3D point cloud shapes. Gao et al. (2019b) propose Structured Deformable Meshes Net (SDM-NET) which consists of PartVAEs and a Structured-Part VAE (SPVAE) for the generation of 3D mesh shapes. PartVAE is used to learn individual part geometry and SPVAE is used to learn part geometries and the structure of the 3D models. SDM-NET can directly output 3D mesh shapes with high surface quality. Compared to point clouds and voxels, meshes can better capture the geometric details (e.g., smoothness, curvature) of 3D objects without consuming large storage space. Therefore, mesh representation is more suitable for engineering design that requires fine-grained details of geometry so that they can be accurately measured, prototyped, and tested for engineering performance. In this study, we adopt SDM-NET as our structure-aware generative design module to generate 3D mesh shapes with high surface quality.

7.2.3 Surrogate Models and AutoML in Engineering Design

Engineering performance evaluation is a critical link in engineering design, optimization, and computational manufacturing. But it is usually computationally expensive. For example, CFD evaluation of the aerodynamic performance of 3D automobile models requires solving the Navier-Stokes equation, which could take hours and days depending on the level of fidelity and computer configurations (Umetani and Bickel, 2018). Therefore, the development of a cost-effective surrogate model holds practical significance. The primary purpose of a surrogate model is to act as an approximation model, replacing intricate and time-consuming computations, to facilitate a fast evaluation of designs without compromising on accuracy (see Queipo et al. (2005) and Wang and Shan (2006) for a review). Typically, surrogate models require that the design be represented as a fixed-length vector (i.e., vectorized design representation (VDR)) (Umetani and Bickel, 2018; Chen et al., 2020).

The process of training surrogate models typically involves utilizing labeled design data, where the labels represent performance metrics of interest. This training can be approached as a supervised learning problem by employing machine learning (ML) techniques. Creating an ML model that achieves excellent performance often requires much investment in terms of computational time and resources in tasks such as feature engineering, model selection, and hyperparameter optimization. Recently, there has been a growing interest in applying Automated Machine Learning (AutoML) to accelerate the process of training optimal surrogate models. AutoML leverages sophisticated algorithms to explore a wide range of models and hyperparameters. This automated search process often leads to better-performing models compared to manual experimentation due to its ability to navigate effectively through extensive search spaces and discover the most favorable configurations (He et al., 2021a).

Although design researchers have been employing common surrogate models outlined in the literature and following industry practices (Cunningham et al., 2019; Whalen and Mueller, 2022), there is limited awareness of AutoML within the engineering design community (Regenwetter et al., 2023). Regenwetter et al. (2023) took a lead in this regard by comparing the performance of surrogate models constructed using traditional methods (e.g., decision trees, k-nearest neighbors, XGBoost (Chen and Guestrin, 2016), and neural networks with Bayesian optimization) against those built using AutoML frameworks. They demonstrated that AutoML outperforms other surrogate models in a bicycle design application and called for the attention of the design community to explore the use of AutoML frameworks. Based on their findings, we compare the performance of different VDRs by adopting two AutoML frameworks, i.e., Auto-sklearn (Feurer et al., 2015) and AutoGluon (Erickson et al., 2020), due to their superior performance compared to other alternatives.

7.3 Research Approach

As shown in Figure 7.2, the proposed approach consists of two key modules: the structure-aware generative design module, which employs the SDM-NET (Gao et al., 2019b); and the design evaluation module, which utilizes surrogate models implemented with AutoML frameworks. The structure-aware generative design module (Section 7.3.1) harnesses the capabilities of SDM-NET to enable efficient exploration of design spaces by incorporating structural information and can generate designs that not only exhibit aesthetic appeal but also possess fine geometric details. On the other hand, the design evaluation module (Section 7.3.2) uses AutoML techniques to construct surrogate models that approximate the engineering performance of interest. While focusing on a car body design as the primary case study to showcase the proposed approach, we ensure that the methodology maintains its generalizability. Therefore, we present a second case study on the aircraft design. See Section 7.4 for details.

7.3.1 Structure-Aware Generative Design Module

We implement SDM-NET (Gao et al., 2019b) for the structure-aware generative design module to generate 3D mesh shapes. This module consists of two types of VAEs: PartVAE and SPVAE. Given a 3D shape consisting of several parts, a PartVAE can learn the geometry of an individual part, and SPVAE can learn the geometries and the structure of parts jointly. We make no novel modifications to the network architecture of SDM-NET. Therefore, we only explain the key steps (see Figure 7.3) to facilitate the understanding of the latent spaces of the structure-aware generative design module. The generative design module is trained using a two-stage training strategy by training the PartVAEs first and then the SPVAE.

7.3.1.1 Two-Stage Training of the PartVAEs and SPVAE

A 3D car model is first segmented into seven parts (i.e., one car body, two mirrors, and four wheels). 3D models from open source databases, such as ShapeNet (Chang et al., 2015), are often unstructured and unoriented triangle meshes. Thus, such 3D mesh shapes cannot be directly used in DGD methods without proper preprocessing (e.g., voxelized or re-meshed). These shapes may also contain interior parts (e.g., seats and steering wheels) that are not desired, since we focus on the external geometry only. As a widely used technology in computer graphics that maps

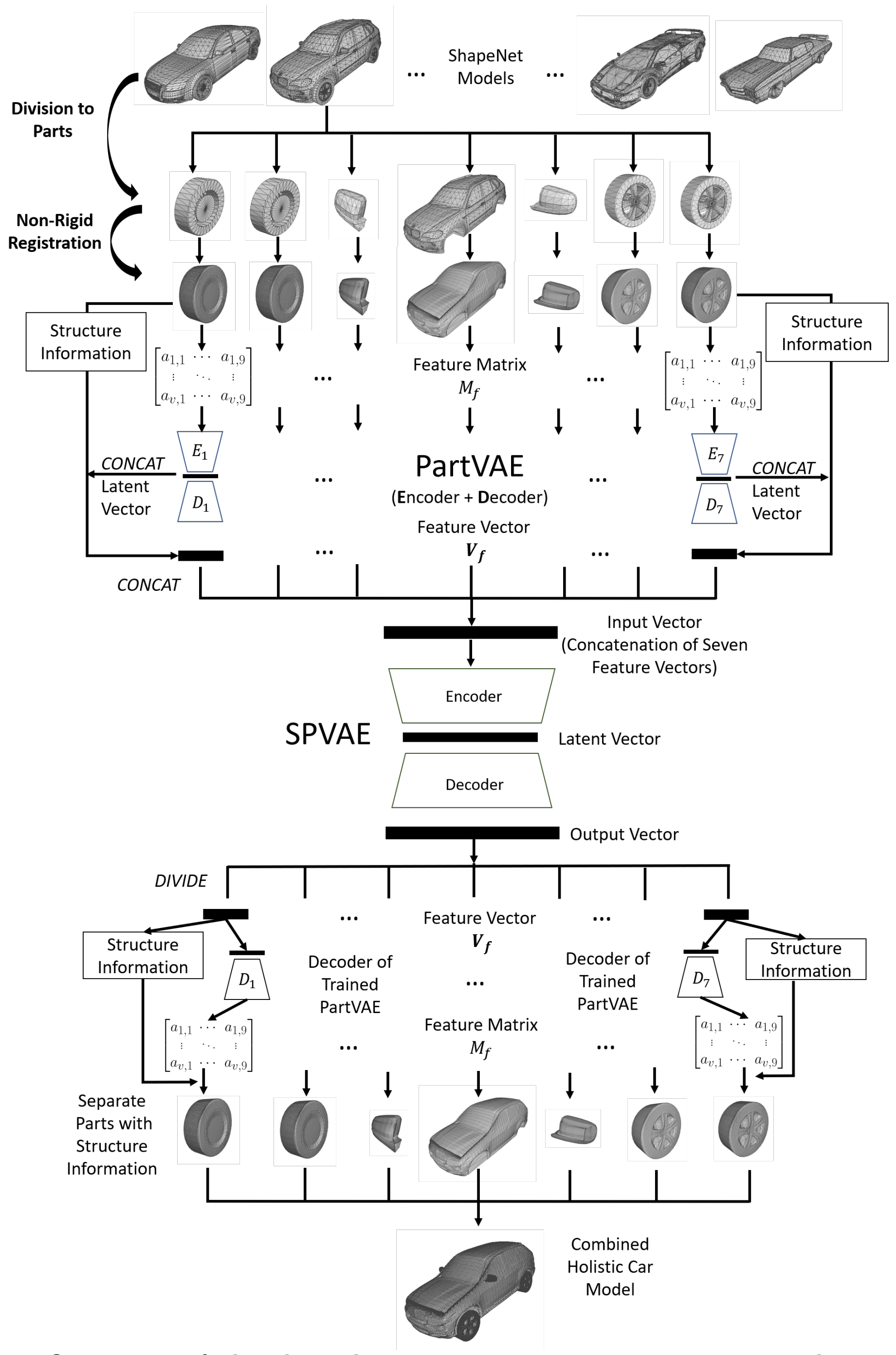


Figure 7.3: Overview of the data-driven structure-aware generative design module demonstrated using a car design case. The design module is implemented using SDM-NET (Gao et al., 2019b) that consists of PartVAEs and SPVAE.

one point set to another, non-rigid registration (Zollhöfer et al., 2014) is applied to re-mesh each part (e.g., car body) using a watertight template mesh shape. In our study, we use a cube as the template mesh that contains 19.2k triangles (9602 vertices). All re-meshed parts are watertight with the same mesh connectivity of the template mesh from which design features will be extracted.

The As Consistent As Possible (ACAP) method (Gao et al., 2019a) is applied to extract the design features of a part for the input of its corresponding PartVAE. We deform the same cube mesh as the one used in non-rigid registration to a target part by multiplying transformation matrices, from which nine unique numbers can be extracted for each vertex of the mesh shape. Thus, a part with v vertices can be represented by a feature matrix $M_f \in \mathbb{R}^{v \times 9}$, where $v = 9602$ in our implementation. One feature matrix can be obtained from each part of a car model, which will be the input to one PartVAE. Thus, seven PartVAEs are trained for car models. After training, the latent space of each PartVAE can be obtained and the latent vector corresponding to a part will be concatenated with the structural information (i.e., support and symmetry information) of the part to form a feature vector $\mathbf{v_f}$. All feature vectors from all parts of a car model will then be concatenated to form the input vector of SPVAE. The SPVAE can then be trained using the concatenated input vectors.

7.3.1.2 Structure-Aware Generative Design of 3D Shapes in Meshes

The trained generative design module can enable structure-aware generative design tasks, such as shape interpolation and random shape generation. As introduced, both the geometric information of parts and inter-part structural information are encoded into the SPVAE latent space.

As shown in Figure 7.3, when provided with an SPVAE vector obtained from the latent space learned by SPVAE, the SPAVE decoder can transform it into an output vector. This resulting vector can be subdivided into seven distinct vectors, each representing a specific car part. These individual vectors contain both the encoded structural information and a separate vector that encodes the geometry of the corresponding part. Afterward, the vector can be decoded using the decoder of the corresponding PartVAE model, resulting in a feature matrix. This feature matrix can then be further processed with the template cube mesh to create a car part using the reverse process of extracting the feature matrix as introduced in Section 7.3.1.1. Separate parts can be combined into one holistic car model with their structural information. It should be noted that independent of the SPVAE, the seven PartVAEs can also be used to generate individual car parts. However, it is not guaranteed to obtain a reasonable car model when combining those parts, as the structural information is not included.

7.3.2 Design Evaluation Module

As shown in Figure 7.2, we construct surrogate models to enable a rapid and reliable evaluation of the engineering performance of interest for the design evaluation module. To achieve that, we need to determine the appropriate vectorized design representation (VDR) and the surrogate modeling frameworks. To train a surrogate model, the label data (the engineering performance of interest, such as drag and lift coefficients) can be obtained from computer simulations, e.g., computational fluid dynamics (CFD) analysis.

7.3.2.1 Vectorized Design Representation

Research has demonstrated the effectiveness of latent vectors derived from the latent space of a trained data-driven generative design (DGD) model in design evaluation (Umetani and Bickel, 2018; Chen et al., 2020). As shown in Figure 7.3, there are two types of latent vectors that can be obtained from the trained structure-aware generative design module, namely SPVAE vectors and PartVAE vectors. They are readily available once the training process is completed.

In addition to the commonly used latent vectors for VDR, we propose a new method that combines a signed distance field (SDF) technique with a 3D point grid (3DPG) inspired by the work (Badías et al., 2019) to generate an alternative form of VDR, namely 3DPG vectors. As illustrated in Figure 7.2, we first construct a 3DPG filled with evenly distributed points. The dimensions of the 3DPG (**L**ength × **W**idth × **H**eight) are chosen according to the largest bounding box of the 3D models in the dataset so that the 3DPG can include all the 3D models. Once a 3D model is put into the 3DPG, each point will be assigned a value of 1 if it falls inside the 3D model or a value of 0, otherwise. The SDF method is used to determine the status of each point. Signed-distance is the distance of a given point $\mathbf{p}$ from the boundary of a set, with its sign determined by whether the point is in the set or not. The signed distance of each point can be calculated and transferred to 0 or 1 using Equation (7.1). Then, all binary values for all points will be concatenated into a 3DPG vector.

$$\phi(\mathbf{p}) = \begin{cases} 1, & \text{if } \mathbf{p} \in \Omega, \text{i.e.}, \mathrm{SDF}(\mathbf{p}) < 0, \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

The status of points in the 3DPG varies for different 3D models, so each 3D model can be uniquely parameterized into a 3DPG vector with a dimension equal to the number of points in the 3DPG. For example, a car model will be parameterized into a 20,000-dimensional vector if there are 20,000 points in the 3DPG. The SDF method (Badías et al., 2019) is effective in handling meshes that are not watertight, but requires the mesh to represent the outer surface (shell) of a 3D object in order to accurately determine the status of individual points in the 3DPG. However, in our case, we cannot directly apply this method to the final holistic 3D shapes created by combining parts from the structure-aware generative design module. This is because these shapes are essentially a combination of multiple shells of the parts. To ensure that the 3DPG vectors can better capture the geometric information of these 3D shapes, we first convert each combined shape into a single shell shape using ManifoldPlus (Huang et al., 2020b) before sending it to the 3DPG.

7.3.2.2 Surrogate Models Using AutoML Frameworks

There are three main reasons for us to utilize AutoML frameworks in constructing the surrogate models: 1) AutoML routinely outperforms experienced data scientists in identifying optimal supervised learning models (Hutter et al., 2019); 2) All of the best-performing AutoML frameworks today rely on some forms of model ensembling techniques that combine predictions from multiple basic models and have long been known to outperform individual models (Dietterich, 2000); and 3) It has been demonstrated that AutoML outperforms the strongest gradient-boosting and neural network surrogate models identified through Bayesian optimization in a bicycle design application (Regenwetter et al., 2023). In summary, AutoML provides a streamlined workflow for training and deploying models, making it suitable for various machine learning applications. Therefore, we use AutoML to build optimal surrogate models to fully understand the potential of different VDRs in performance evaluation and prediction.

Specifically, we apply two popular AutoML frameworks, Auto-sklearn (Feurer et al., 2015) and AutoGluon (Erickson et al., 2020). Auto-sklearn has been the winner of numerous AutoML competitions (Guyon et al., 2019). It employs efficient multi-fidelity hyperparameter optimization strategies and the combination of numerous models through an ensemble selection strategy. AutoGluon was introduced recently and has been reported to outperform many other alternatives in various applications (Erickson et al., 2020). It applies an innovative layer-stack ensembling method. Additionally, AutoGluon incorporates k-fold bagging to minimize the risk of overfitting.

7.3.2.3 The Objectives of the Design Evaluation Module

The design evaluation module is to approximate the input-output relationship defined by the computer simulation $f(\cdot)$ as shown by Equation (7.2), where $\boldsymbol{x}$ represents a VDR of a 3D shape X, and y denotes the corresponding performance metric of interest which is used as the ground truth value. Similarly, the surrogate model can be defined by Equation (7.3), where $\hat{y}$ is the predicted performance of interest and $g(\cdot)$ is the approximation of $f(\cdot)$ implied by the surrogate model. The objective of the surrogate model can be seen as an optimization problem defined by Equation (7.4).

$$y = f(\mathbf{x}) \tag{7.2}$$

$$\hat{y} = g(\mathbf{x}) \tag{7.3}$$

$$\hat{y}^* = \text{argmin}_{\hat{y}} \left| y - \hat{y} \right|, \forall \mathbf{x} \tag{7.4}$$

The pair data, i.e., the VDRs and their corresponding engineering performance values, form the training dataset for the surrogate models. We split the training dataset into a train set and a test set. The surrogate models will be trained using the train set only, and the test set serves as unseen data to test the generalizability of the trained surrogate models. In order to conduct a fair comparison of the performance of various types of VDRs in predicting engineering performance, we train an optimal surrogate model for each combination (i.e., one type of VDR and one particular AutoML framework) under identical conditions. For example, the training data and the configurations of the AutoML framework are kept the same. In addition, we use Auto-sklearn (Feurer et al., 2015) and AutoGluon (Erickson et al., 2020) to examine whether the results would be independent of a particular AutoML framework used.

For a comprehensive comparison, we adopt three evaluation metrics: the mean absolute error (MAE), the root mean squared error (RMSE), and the coefficient of determination (R^2) as calculated by the following equations, where n is the total number of observations, y_i is the actual value of the observation i, $\hat{y}_i$ is the predicted value of the observation i and $\bar{y}$ is the mean of the actual values. We also perform the paired t-test on the absolute errors (AE) values ($|y_i - \hat{y}_i|$) of two groups to test if there is a statistically significant difference between the prediction accuracy when using different VDRs. The paired t-test is not performed on squared errors (SE) since they are essentially squared AE, and the test is not conducted on the R^2 as it represents a statistics of a group of data.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{7.5}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{7.6}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{7.7}$$

7.4 Implementation Details and Results

In this section, we present the implementation details and results of the structure-aware generative design module and the design evaluation module. All experiments were run on a Linux workstation with a TITAN RTX GPU and a 20-core Intel Xeon Silver 4114 CPU.

7.4.1 Design Cases and Datasets

We demonstrated the proposed approach and conducted a comparative study in two design cases: the car and aircraft designs. The datasets used for the structure-aware generative design module and the design evaluation module are summarized in Table 7.1.

7.4.1.1 Training Data for the Structure-Aware Generative Design Module

For the training of the structure-aware generative design module, we collected 1824 car and 2690 aircraft mesh models from Gao et al. (2019b), which have been

Table 7.1: Summary of the datasets for the two design cases: the car and aircraft designs

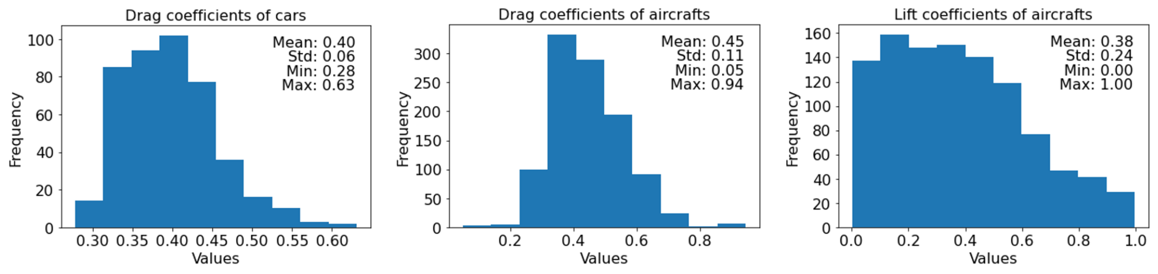
			Car	Aircraft
Structure-aware generative design module	3D model data	Source	Gao et al. (2019) derived from ShapeNet (Chang et al. 2015) and ModelNet (Wu et al. 2015)	Gao et al. (2019) derived from ShapeNet (Chang et al. 2015) and ModelNet (Wu et al. 2015)
		Number	1824	2690
	Filter		• Automatic filter: seven parts (i.e., one body, two mirrors, and four wheels) • Manual filter: regular passenger car models	• Automatic filter: eight parts (i.e., one fuselage, two wings, three tails, and two engines)
	Number of training data		1161	1597
Design evaluation module	Engineering performance data	Source	Song et al. (2023) developed from ShapeNet (Chang et al. 2015) using data augmentation and OpenFoam (Jasak et al. 2007)	Edwards et al. (2021) developed from ShapeNet (Chang et al. 2015) using OpenFoam (Jasak et al. 2007)
		Number	9070	4045
	Filter		• Overlapping with the training data of the design module • Value range (0, 1)	• Overlapping with the training data of the design module • Value range (0, 1)
	Number of training data		439	1047

divided into parts using a semantic segmentation approach (Yi et al., 2016). These data are derived from ShapeNet (Chang et al., 2015) and ModelNet (Wu et al., 2015a). We developed an algorithm to automatically select models that have all seven parts (i.e., one body, two mirrors, and four wheels) for the car models and all eight parts (i.e., one fuselage, two wings, three tails, and two engines) for the aircraft models. Also, for car models, since we focused on regular passenger car models (e.g., sedans, SUVs), we manually excluded the other car types, including buses, Formula One, and trucks. This gave us a total of 1161 car models and 1597 aircraft models, which were used as training data for the structure-aware generative design module.

7.4.1.2 Training Data for the Design Evaluation Module

We collected the engineering performance data of car models and aircraft models by leveraging two open-sourced datasets: 9070 car models labeled by drag coefficients (Song et al., 2023b) and 4045 aircraft models with drag and lift coefficients (Edwards et al., 2021). The 3D model data of the two datasets are both derived from ShapeNet (Chang et al., 2015). The corresponding performance values are obtained from the computational fluid dynamics (CFD) simulation tool OpenFOAM (Jasak et al., 2007).

To match the labeled 3D models with the training data used in the structure-aware generative design module, we selected the overlapping models between the two datasets for each case study. Furthermore, we only selected models with drag or lift coefficients within the range of 0 to 1 to ensure data quality and reliability. This gave us a total of 439 car models with corresponding drag coefficients and 1047 aircraft models with drag and lift coefficients. These data were used as training data for the design evaluation module. Figure 7.4 shows the histograms of the drag coefficients for the car models, and the drag coefficients and lift coefficients for the aircraft models, along with their descriptive statistics in the legend of Figure 7.4.



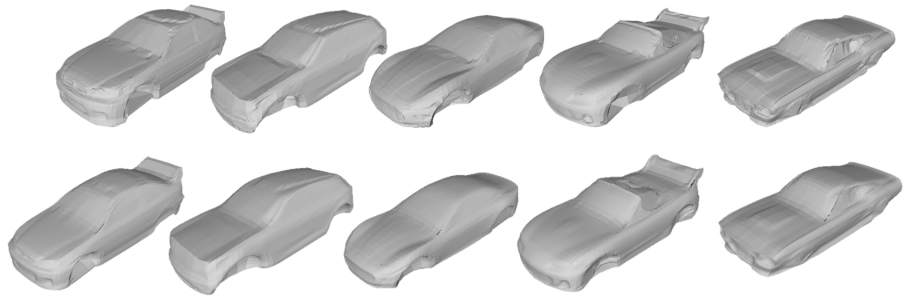
(a) (b) (c)

Figure 7.4: Histograms of (a) drag coefficients of car models, (b) drag coefficients of aircraft models, and (c) lift coefficients of aircraft models with the mean, std, min, and max values.

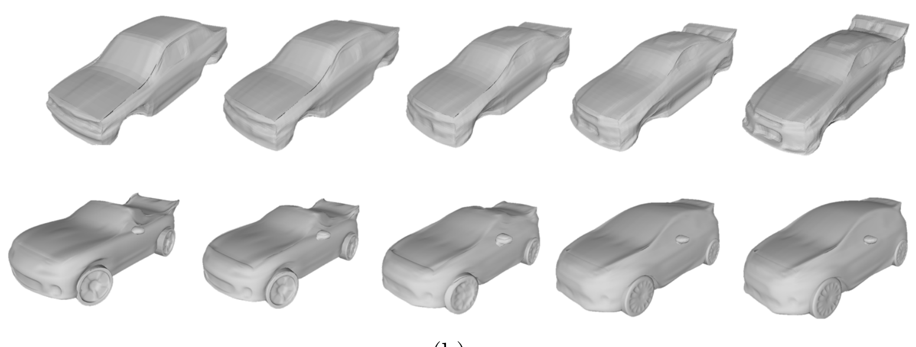
7.4.2 Structure-Aware Generative Design Module

We used the same strategies for both car models (1161) and aircraft models (1597) to train the structure-aware generative design module as detailed below. We randomly split the training dataset into train data (75%) and test data (25%). 64 and 128 were chosen for the dimensionality of the latent spaces of PartVAEs and the SPVAE, respectively, because they yield the lowest reconstruction errors, as shown by Gao et al. (2019b). To ensure effective training, we implemented the two-stage training strategy discussed in Section 7.3.1.1. This involved initially training PartVAEs, followed by training the SPVAE until the networks converged. Throughout the training process, we monitored and evaluated all associated training loss terms (i.e., reconstruction loss and Kullback–Leibler (KL) divergence loss for both train and test data). The convergence of these loss terms indicated that the networks were successfully trained. In the Appendix C, we document the training loss values throughout the training process in Figure C.1. The training for the car models took approximately 72 hours (i.e., PartVAEs: 10000 epochs; SPVAE: 20000 epochs), while it took approximately 120 hours (i.e., PartVAEs: 5000 epochs; SPVAE: 10000 epochs) to complete the training for the aircraft models.

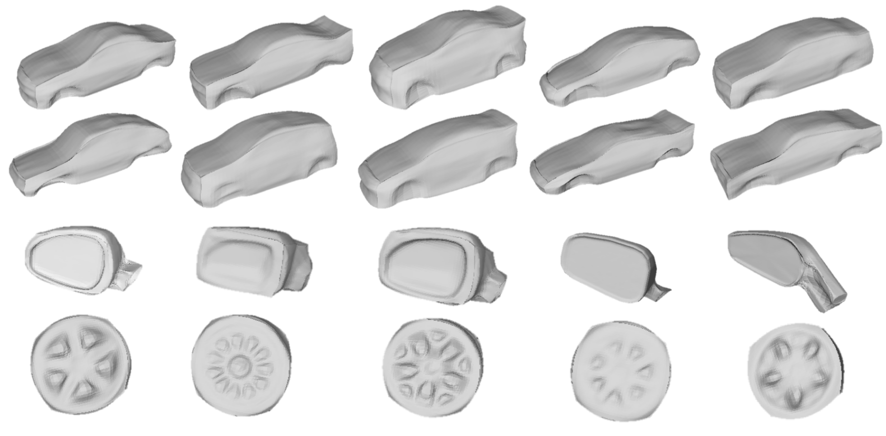
In Figure 7.5(a), several reconstructed car body models are displayed. The first row presents the original models, while the second row displays the corresponding



(a)

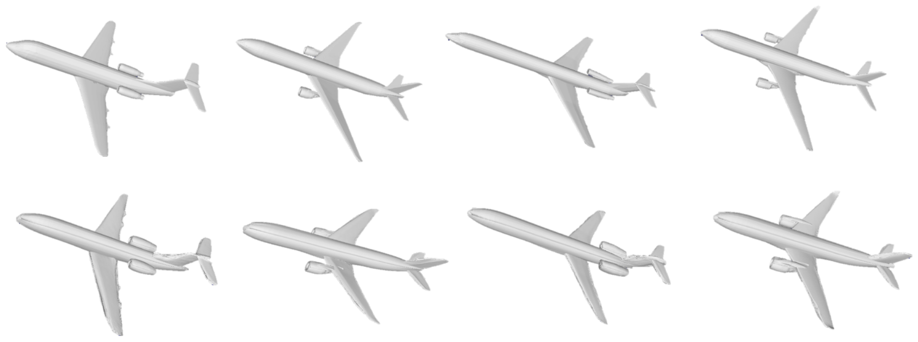


(b)

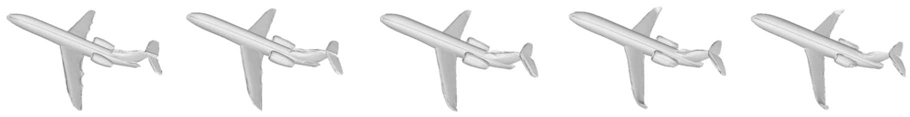


(c)

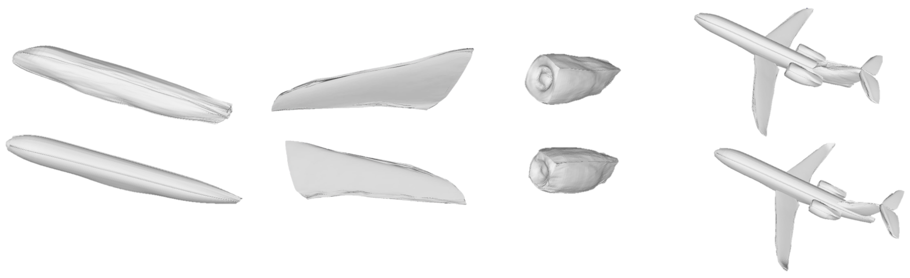
Figure 7.5: Examples of the generated shapes for car models. (a) Reconstruction of car bodies. (b) Shape interpolation of car bodies and merged car models. (c) Random generation of separate parts.



(a)



(b)



(c)

Figure 7.6: Examples of the generated shapes for aircraft models. (a) Reconstruction of aircraft models. (b) Shape interpolation of merged aircraft models. (c) Random generation of separate parts (fuselage, wings, and engines) and merged aircraft models.

reconstructed models. Figure 7.5(b) shows a few car bodies and combined car models by linearly interpolating the shapes of the first and last columns. The in-between columns from the second to the fourth column are linearly interpolated shapes. We can observe a gradual transition of the geometry from the first column to the last column. In addition, SPVAE vectors can be randomly sampled from the learned latent space to generate random shapes (car bodies, mirrors, and wheels) as shown in Figure 7.5(c). Similarly, the results for aircraft models are presented in Figure 7.6. Figure 7.6(a) exhibits several instances using shape reconstruction, where the top row shows the original models, and the bottom row shows their corresponding reconstructed versions. Figure 7.6(b) illustrates the combined aircraft models created by linearly interpolating the shapes of the first and last columns. Notably, the interpolation effectively captures the transformation of the wings, progressing from a completely flat configuration to a slightly curved shape towards the wingtips. Figure 7.6(c) exhibits examples of randomly generated aircraft parts: fuselage, wings, and engines, from left to right, as well as combined aircraft models.

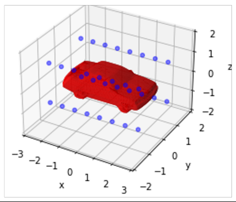
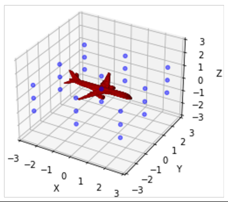
Theoretically, we can sample as many latent vectors as possible from the latent space for random shape generation. Shape interpolation can be performed between every pair of car models with any number of in-between interpolation shapes. Thus, we can generate thousands of unseen designs, and the generated designs look reasonable in terms of visual appearance and have great geometry details. The results also indicate that the latent spaces of PartVAEs and the SPVAE are trained well which can serve as VDRs for the design evaluation module.

7.4.3 Design Evaluation Module

7.4.3.1 Latent Vectors and 3DPG Vectors for the VDR

To evaluate and determine the most effective VDR in predicting the engineering performance investigated, we prepared two representative VDRs: latent vectors (the commonly-used VDR; and are obtained from the training process of the DGD

Table 7.2: Summary of the vectorized design representations (VDRs) for the car and aircraft models

		Car	Aircraft
Latent vectors (dimension)	PartVAE	• All_parts vectors (448) • Body vectors (64)	• Wings vectors (128) • All_parts vectors (512)
	SPVAE	SPVAE vectors (128)	SPVAE vectors (128)
	Time to obtain a latent vector	• $\approx 0s$ (inference time) • $\approx 223s$ (training time per car model)	• $\approx 0s$ (inference time) • $\approx 271s$ (training time per aircraft model)
3DPG vectors	Largest bounding box in $L \times W \times H$	$0.86 \times 0.37 \times 0.29$	$0.91 \times 0.88 \times 0.30$
	Scale factor of 3D models	$\times 5$	$\times 5$
	Dimension of the 3DPG in $L \times W \times H$	$5 \times 2 \times 2$	$5 \times 5 \times 2$
	Number of points in the 3DPG in $L \times W \times H$	• $35 \times 12 \times 12 = 5040$ • $40 \times 16 \times 16 = 10240$ • $50 \times 20 \times 20 = 20000$	• $35 \times 12 \times 12 = 5040$ • $40 \times 16 \times 16 = 10240$ • $50 \times 20 \times 20 = 20000$
	Illustration of the SDF method		
	Time to obtain a 3DPG vector	$\approx 35s$	$\approx 35s$

models) and 3DPG vectors (the proposed VDR; and require additional steps to vectorize the generated designs) for the training of the design evaluation module as introduced in Section 7.3.2.1. The configurations of these VDRs are summarized in Table 7.2.

Regarding the latent vectors (as depicted in Figure 7.3), we utilized two types of VAE vectors. First, we employed the 128-dimensional SPVAE vectors for both car and aircraft models. Second, we concatenated the PartVAE vectors from all parts, resulting in the creation of all_parts vectors. For car models, the all_parts vectors have a dimension of $64 \times 7 = 448$, while for aircraft models, the dimension is $64 \times 8 = 512$. The major difference between all_parts vectors and SPVAE vectors is that all_parts vectors encode geometric information only, while the SPVAE vectors encode both geometric and structural information. Additionally, we took into account the significance of the car body in calculating the drag coefficient, as well as the significance of the wings (airfoils) in determining the lift coefficient (Fairman, 1996). Thus, we specifically chose the body vectors (64-dimensional) for the car models and the wing vectors (two wings, 64×2 =128-dimensional) for the lift prediction of the aircraft models.

For 3DPG vectors, we found that the largest bounding box dimensions ($L \times W \times H$) for car models to be $0.86 \times 0.37 \times 0.29$, while for aircraft models, it was found to be $0.91 \times 0.88 \times 0.30$. The 3D models are all normalized (Chang et al., 2015) and the values of the bounding boxes in the mesh files do not have a unit because they are dimensionless, but they are proportional to the size of actual 3D models. To ensure the inclusion of all models in the training data for the design evaluation module, we set the 3DPG dimensions to $5 \times 2 \times 2$ for car models and $5 \times 5 \times 2$ for aircraft models for convenience. They can be set to different values as long as the ratio $L/W/H$ is maintained. Each car or aircraft model can then be scaled up by a factor of 5 to better fit into the corresponding 3DPG.

After setting up the 3DPG, we can obtain a 3DPG vector as outlined in Section

7.3.2.1 by utilizing the SDF method. Although only the 20,000-dimensional vector was shown to be effective in Badías et al. (2019), we tested three different configurations for 3DPG vectors: 1) $35 \times 12 \times 12 = 5040$, 2) $40 \times 16 \times 16 = 10240$, and 3) $50 \times 20 \times 20 = 20000$. The time cost for parameterization remains constant at approximately 35 seconds, regardless of any changes in the dimension of the VDRs or the specific car or aircraft models being used. The reason for this is that the SDF method involves performing a virtual laser scan of the input 3D model, and the computational time is dominated by the resolution of the scan. Although it appears that it took no time to obtain the latent vectors from a trained structure-aware generative design model, the training itself can be time-consuming. It took approximately 223 seconds per car model and 271 seconds per aircraft model according to the training time introduced in Section 7.4.2 while obtaining 3DPG vectors does not involve any training process.

7.4.3.2 Prediction Results Using Different VDRs And Surrogate models

We ended up with a total of 439 car models, each having an associated drag coefficient. These car models were represented by six types of VDRs, including 128-dimensional SPVAE vectors, 448-dimensional all_parts vectors, 64-dimensional body vectors, as well as three types of 3DPG vectors with dimensions of 5040, 10240, and 20000, respectively. Similarly, for the 1047 aircraft models with associated drag and lift coefficients, we had 128-dimensional SPVAE vectors, 512-dimensional all_parts vectors, 128-dimensional wing vectors, and the same types of 3DPG vectors as car models. By utilizing each type of VDR along with the corresponding engineering performance label data, we randomly divided the training dataset into two parts: a train set (80%) and a test set (20%). Importantly, this division remained the same for all VDRs within a particular design case to make a fair and consistent comparison. We then trained an optimal surrogate model using Auto-sklearn (Feurer et al., 2015) and AutoGluon (Erickson et al., 2020).

These AutoML models have the capability to automatically reserve a portion of the train set data as validation data. We trained all AutoML models by minimizing

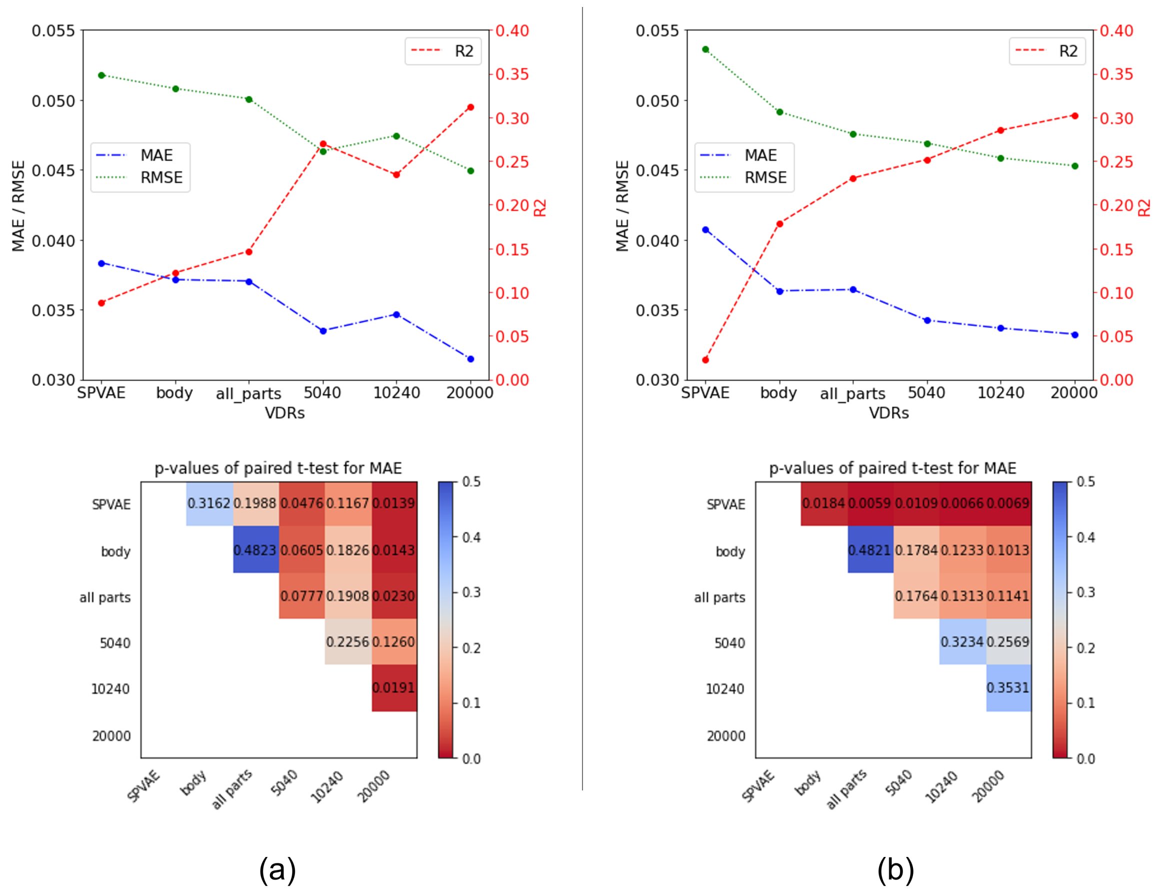
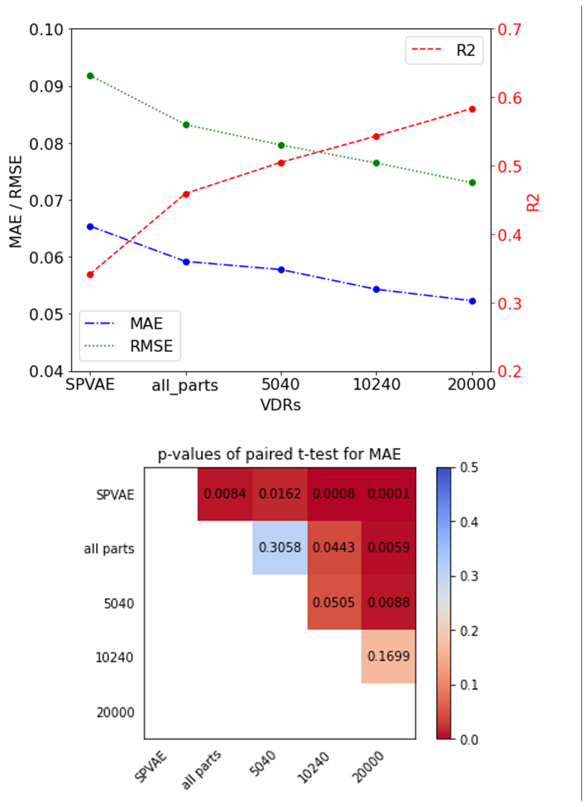


Figure 7.7: The comparison of prediction accuracy for the test set data of car models using different VDRs with drag coefficients with three evaluation metrics: MAE (↓), RMSE (↓), and the R^2 (↑) using (a) Auto-sklearn or (b) AutoGluon. The p-values resulting from the paired t-test for AE values are used to show the statistical difference.

RMSE on the validation data for optimal surrogate models. Given our focus on understanding the generalizability of the trained surrogate models to unseen data, we primarily present the prediction results specifically for the test data. We have made the dataset, code and all results for the design evaluation module open-source for the purpose of reproducibility and for further research interests [2].

Results for predicting drag coefficients of the car models. Several insights

[2]`https://github.com/Xingang-Li/structure_aware_design_evaluation`



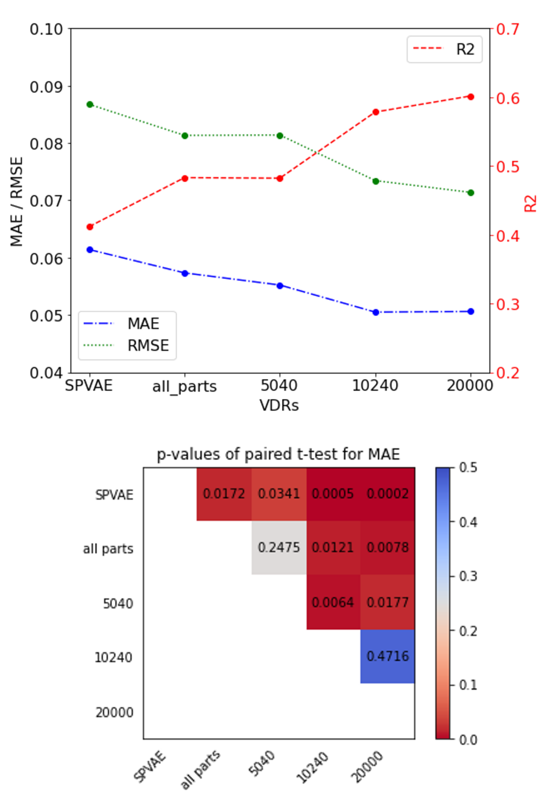


Figure 7.8: The comparison of prediction accuracy for the test set data of aircraft models using different VDRs with drag coefficients with three evaluation metrics: MAE (↓), RMSE (↓), and the R^2 (↑) using (a) Auto-sklearn or (b) AutoGluon. The p-values resulting from the paired t-test for AE values are used to show the statistical difference.

can be drawn based on the results of MAE, RMSE, and R^2 presented in Figure 7.7[3]. Regardless of the AutoML frameworks used, the 3DPG vectors consistently exhibit higher accuracy than the latent vectors. SPVAE vectors achieve the lowest accuracy, while the 20000-dimensional 3DPG vectors achieve the highest accuracy among all alternative VDRs. For 3DPG vectors, the mean accuracy increases in general with higher dimensions. The best combination of the VDR and AutoML framework is observed to be the 20000-dimensional 3DPG vectors and Auto-sklearn, resulting in an R^2 value of 0.312. On the other hand, the worst combination is observed to be the SPVAE vectors and AutoGluon, resulting in an R^2 value of 0.021.

The heatmaps in Figure 7.7 show the p-values associated with the paired t-test performed on the AE values. The results indicate that regardless of the specific AutoML framework utilized, the performance of the SPVAE vectors is consistently inferior to that of the 20000-dimensional 3DPG vectors, and this difference is statistically significant, i.e., $p = .0139$ when using Auto-sklearn and $p = .0069$ when using AutoGluon. Regarding the latent vectors, no significant differences are observed between the SPVAE vectors, the body vectors, and the all_parts vectors. The only exception is the difference between the SPVAE vectors and the all_parts vectors ($p = .0059$) in AutoGluon. Similarly, for 3DPG vectors, while the mean values of all three metrics show an increasing trend, no significant differences are observed between different VDRs.

Results for predicting drag coefficients of the aircraft models. Based on the results of MAE, RMSE, and R^2 presented in Figure 7.8, we can get several insights as follows. Similar to the findings in car models, the 3DPG vectors consistently demonstrate superior accuracy compared to the latent vectors across both AutoML

[3]The root mean square error (RMSE) and R^2 (coefficient of determination) are both statistics for group data. Consequently, the error bars are not applicable to them. Additionally, the error bars are not included for the mean absolute error (MAE) because the variance of absolute error (AE) across the entire dataset is not relevant in this analysis due to the use of the paired t-test rather than the independent two-sample t-test. This is because the different VDRs were obtained from the same data group (i.e., the test data set), so the datasets for the t-test are not independent.

frameworks employed. Among all the alternative VDRs, it is also observed that SP-VAE vectors exhibit the lowest accuracy, while the 20000-dimensional 3DPG vectors achieve the highest accuracy. There is a notable trend of increasing accuracy as the dimension of the 3DPG vectors increases. The most favorable combination of the VDR and AutoML framework is observed with the 20000-dimensional 3DPG vectors with AutoGluon, yielding an R^2 value of 0.602. Conversely, the least favorable combination is observed with the SPVAE vectors and Auto-sklearn, resulting in an R^2 value of 0.341.

We also conducted a paired t-test on the AE values and the resulting p-values are visualized in the heatmaps in Figure 7.8. The heatmaps indicate significant differences ($p < 0.05$) between most pairs of VDRs, except for three cases: 1) all_parts vectors and 5040-dimensional 3DPG vectors with both Auto-sklearn and AutoGluon, 2) 5040 and 10240-dimensional 3DPG vectors (where the p-value slightly exceeds 0.05 with Auto-sklearn but was less than 0.05 with AutoGluon), and 3) 10240 and 20000-dimensional 3DPG vectors with both Auto-sklearn and AutoGluon.

Results for predicting lift coefficients of the aircraft models. As the results of MAE, RMSE, and R^2 shown in Figure 7.9, regardless of the AutoML frameworks used, the SPVAE vectors achieve the lowest accuracy while all_parts and wing vectors achieve the top two highest accuracies among all the alternative VDRs. However, there is a discrepancy between the performance of the all_parts and wing vectors using the two AutoML frameworks. 3DPG vectors perform poorly, and there is no noticeable trend of increasing accuracy with higher-dimensional 3DPG vectors, as previously observed in the prediction of drag coefficients. The best combination of the VDR and AutoML framework is observed with the wing vectors and AutoGluon, resulting in an R^2 value of 0.389. The worst combination is observed with the SPVAE vectors and Auto-sklearn, resulting in an R^2 value of 0.244.

For the t-test conducted on the AE values, the results of the p-values are shown in Figure 7.9. The heatmaps show that apart from four specific pairs in

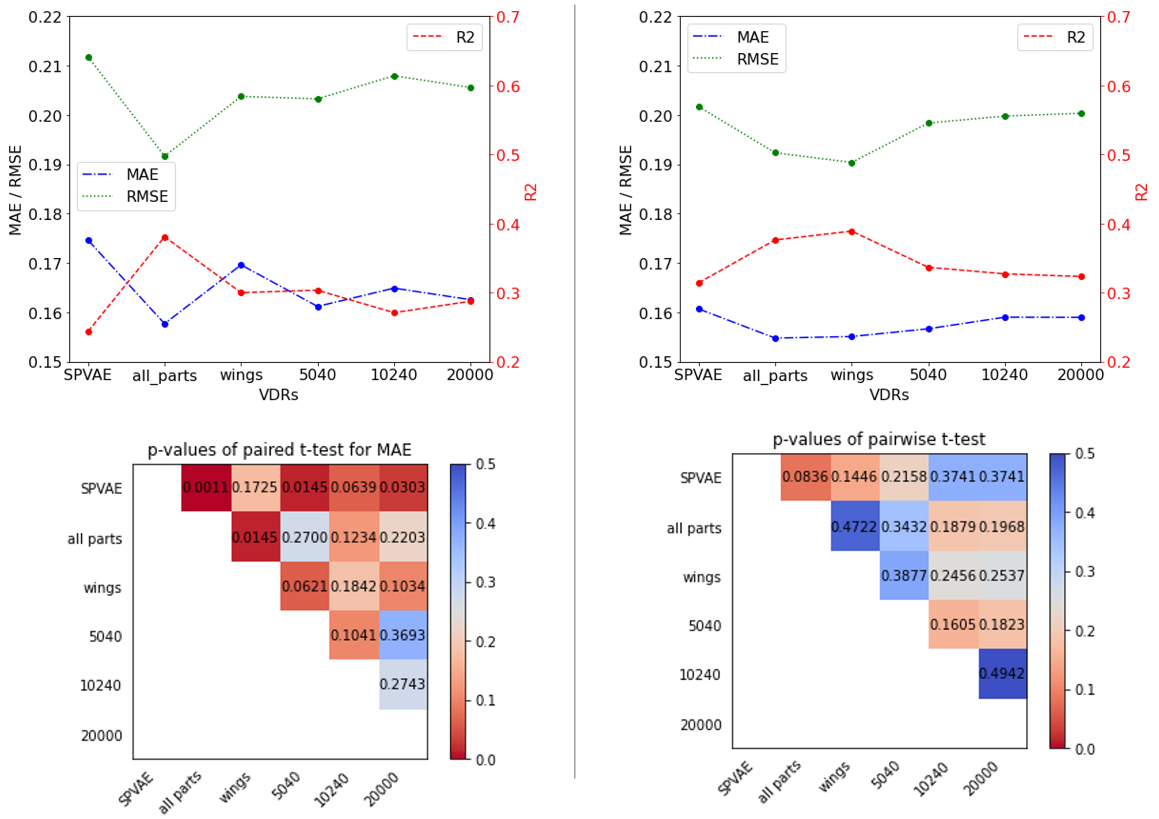


Figure 7.9: The comparison of prediction accuracy for the test set data of aircraft models using different VDRs with lift coefficients with three evaluation metrics: MAE (↓), RMSE (↓), and the R^2 (↑) using (a) Auto-sklearn or (b) AutoGluon. The p-values resulting from the paired t-test for AE values are used to show the statistical difference.

Table 7.3: The best combination of the VDR and AutoML framework for the surrogate models of car and aircraft models

		Car	Aircraft	
		Drag	Drag	Lift
VDR		20000-dim 3DPG vectors	20000-dim 3DPG vectors	Latent vectors of wings
AutoML Framework		Auto-sklearn	AutoGluon	AutoGluon
Performance Metric	R^2	0.312	0.602	0.389
	MAE	0.032	0.051	0.155
	RMSE	0.045	0.071	0.190

Auto-sklearn: 1) SPVAE vectors and all_parts vectors, 2) SPVAE vectors and 5040-dimensional 3DPG vectors, 3) SPVAE vectors and 20000-dimensional 3DPG vectors, and all_parts vectors and wing vectors, there is no significant difference ($p < 0.05$) observed between all combinations of VDRs when using the two AutoML frameworks.

We summarize the results for the best combination of the VDR and AutoML framework for the surrogate models and the corresponding prediction performance metrics of car and aircraft models in Table 7.3.

7.5 Disucssion

In this section, we provide a comprehensive analysis of the results obtained from the structure-aware generative design module (Section 7.5.1) and the design evaluation module (Section 7.5.2) and discuss the limitations and potential future research directions.

7.5.1 Structure-Aware Generative Design Module

Structure-aware generative design is an emerging and relatively unexplored field that holds promise in addressing the challenges of systems design problems using data-driven generative design (DGD) methods. Unlike the generative design of monolithic shapes that prioritize optimizing overall system performance, the structure-aware generative design focuses on capturing and integrating the details of parts' geometry and structure. By considering the structural characteristics of individual parts, the structure-aware generative design enables a more comprehensive understanding of the system. It can also facilitate the exploration of various design alternatives and iterations, empowering engineers to make well-informed decisions regarding part geometries and the structural relations between parts. This opens opportunities for the discovery of novel and optimized designs that may have been overlooked using traditional generative models, especially in the early stages of design.

In our study, we implemented SDM-NET (Gao et al., 2019b), a data-driven structure-aware generative model, as the structure-aware generative design module. While it holds significant potential to help designers effectively explore the design space, there is a major limitation in the current methodology. The validity of the generated designs in terms of their structural integrity heavily relies on visual inspection, and there is no quantitative method available to assess. While most of the generated designs are deemed acceptable, some may have unattached parts despite the structural information learned. One possible approach to ensure structural validity is to employ optimization techniques (Gao et al., 2019b), but it is crucial to develop a quantitative and automatic method to assess the structural validity of the generated designs, e.g., rating the designs in terms of their structural integrity and surface quality.

7.5.2 Design Evaluation Module

7.5.2.1 Drag prediction for car and aircraft models

The SPVAE vectors demonstrate the least accuracy, whereas the all_parts vectors consistently enhance the predictive accuracy in both case studies. In the context of car models, the body vectors also exhibit superior performance compared to the SPVAE vectors, as indicated by lower MAE and RMSE values and higher R^2. This superiority is further supported by a paired t-test using AutoGluon ($p =$

0.0184), as depicted in Figure 7.7. Both all_parts vectors and SPVAE vectors encode geometric information for all components of the 3D shapes. However, the SPVAE vectors also include structural information such as support and symmetry. This structural information is crucial to generate designs that account for the underlying structures of 3D shapes, as illustrated in Figure 7.3. Nevertheless, when the structural information is irrelevant to the engineering performance, such as the drag here, it can have a detrimental impact on the suitability of the SPVAE vectors as VDRs for surrogate models. In such cases, using latent vectors that only capture geometric information most relevant to the engineering performance of interest, such as the all_parts vectors or car body vectors, can be more advantageous for surrogate models. On the other hand, in situations where structural information plays a significant role in engineering performance, such as in determining the maximum allowable load for a bike frame design problem, using VDRs that incorporate structural information may have advantages over VDRs that only capture geometric information.

Likewise, by capturing the geometric information of all components of 3D shapes, 3DPG vectors possess significant potential to serve as more suitable VDRs for predicting drag coefficients. There is a general trend of improved accuracy, reflected in lower MAE and RMSE values, and higher R^2 values, as the dimensionality of 3DPG vectors increases. Specifically, the 20000-dimensional 3DPG vectors exhibit the highest level of accuracy. Increasing the dimensionality of the 3DPG vectors implies using more points to parameterize a design utilizing the 3D point grid, as described in Table 7.2. This augmentation in the number of points enables a more comprehensive capture of the geometric details of 3D shapes, thereby enhancing the prediction of the drag coefficient, which is closely influenced by the overall geometry of 3D shapes. But, it should be noted that 3DPG vectors also encode the positional information of various components due to the signed distance field, whereas all_parts vectors (concatenation of part vectors) do not contain such information. This distinction in encoding positional information could be one of the reasons why 3DPG vectors generally exhibit better performance compared to all_parts vectors in predicting drags in both design cases.

If considering statistical significance, however, augmenting the dimensionality of 3DPG vectors does not necessarily lead to a significant improvement in prediction accuracy as evident: 1) In the case of car models, there are generally no significant differences in the prediction accuracy between the 3DPG vectors; 2) Similarly, for the aircraft models, there are no significant differences between 10240- and 20000-dimensional 3DPG vectors. The lack of significant differences could be attributed to the relatively small size of the datasets. Specifically, the dataset for car models (439) is approximately 60% smaller than the dataset for aircraft models (1047). This discrepancy in dataset size, as depicted in Figure 7.1, causes a less number of pairs of VDRs with significant differences in the analysis of the car models compared to the aircraft models. In addition, the curse of dimensionality could be another reason affecting the performance of 3DPG vectors with higher dimensionality. Moreover, it is important to acknowledge that in order to achieve a similar or higher level of prediction accuracy compared to latent vectors (such as SPVAE vectors or all_parts vectors), 3DPG vectors should have a minimum of 5040 dimensions, as demonstrated in our experiments. We conducted tests using an extreme case of 128 dimensions (same as the SPVAE vectors), which resulted in a significant decrease in prediction accuracy and even yielded negative R^2 values in the case study of car design.

7.5.2.2 Lift Prediction for Aircraft Models

SPVAE vectors consistently exhibit the lowest prediction accuracy in lift prediction, similar to their performance in drag prediction regardless of the AutoML frameworks employed. Although there are variations in the outcomes produced by Auto-sklearn and AutoGluon, the use of the latent vectors of wings (i.e., wing vectors) in conjunction with AutoGluon demonstrates the highest accuracy in terms of a higher R^2 value of 0.389.

In a similar manner to the drag prediction, it is evident that there are no significant differences among 3DPG vectors with varying dimensions. Additionally, there are no significant differences between all_parts vectors and 3DPG vectors, as supported by the p-values (all are greater than 0.1) shown in Figure 7.9. However, when considering the MAE, RMSE, and R^2 metrics, the mean values indicate a slightly decreasing trend in prediction accuracy for 3DPG vectors as the dimensionality increases, and the performance of all_parts vectors surpasses that of 3DPG vectors, which deviates from the drag prediction scenario. This discrepancy can be attributed to two major factors: the encoded geometric information and the curse of dimensionality. The drag coefficient is affected by all components of the 3D shapes, while the lift coefficient is mainly determined by the wings (NASA; Fairman, 1996). Although the 3DPG vectors, which encode geometric information of all components, can be advantageous for drag prediction, they pose challenges when predicting lift because they include a considerable amount of irrelevant geometric information from non-wing parts. Consequently, the advantage of increasing the dimensionality of 3DPG vectors to capture more geometric details is diminished by the curse of dimensionality. As a result of this phenomenon, the performance of the 3DPG vectors in lift prediction even decreases to a level comparable to that of the SPVAE vectors, and even worse than that of the all_parts vectors (512-dimensional), as shown in Figure 7.9.

7.5.2.3 Summary of the Two Design Cases

Important insights can be derived from the two design cases involving drag and lift prediction. While latent vectors have frequently been employed as VDRs in surrogate models, they may not be the most appropriate option when encoded information includes a mixture of relevant and irrelevant information for the engineering performance of interest. Specifically, when utilizing structure-aware generated design, caution should be exercised when employing latent vectors that encode both geometry and structural information (such as SPVAE vectors in our case) that are often readily obtainable from the training. Instead, the underlying physics shall be examined to determine what geometric information would contribute most and whether the structural information is relevant to the engineering performance to be predicted.

For 3DPG vectors, increasing the dimensionality does not necessarily improve the predictive performance of surrogate models. This is especially true when the vectors contain more noise, i.e., the information irrelevant to engineering performance, such as in the lift prediction.

Limitations: 1) Despite trying different combinations of AutoML frameworks and VDRs, the resulting surrogate models achieved modest R^2 values of 0.312, 0.602, and 0.389 for drag prediction in cars and aircraft, and lift prediction in aircraft, respectively. These values fall short of high predictive accuracy if referring to the criterion of $R^2 = 0.67$ (Henseler et al., 2009). The primary reason is the limited availability of data. This has been evident by the difference between the R^2 value of drag prediction in aircraft (0.602) and that in cars (0.312) because there are more data points for the aircraft models compared to the car models (1047 vs. 439). 2) The SDF method, although effective in capturing the geometric information of 3D shapes, suffers from considerable computational cost, taking approximately 35 seconds to generate each high-quality 3DPG vector. We conducted experiments to explore the impact of reduced resolution in laser scans as outlined in Section 7.4.3.1. In particular, utilizing a lower resolution could reduce the processing time to 3 seconds. However, this reduction in resolution also led to a significant drop in prediction accuracy, with the R^2 value declining by up to 41%. To ensure data quality, computational cost poses a practical limitation for its application in interactive generative design. Therefore, it is valuable to investigate different implementations that can offer faster solutions to encode 3D shapes.

7.6 Conclusion and Future Work

Data-driven generative design (DGD) methods can effectively support design ideation and 3D shape synthesis. With the recent advances in structure-aware DGD, this study is motivated to answer the following question: what are the appropriate vectorized design representations (VDRs) for fast performance evaluation of the 3D shapes generated by the structure-aware DGD method? To answer this question, we first developed a structure-aware generative design module based on SDM-NET (Gao et al., 2019b) that can generate various new 3D shapes taking into account the interconnections between parts. Then, we realized the fast design evaluation module by constructing surrogate models using AutoML frameworks. Based on the integrated framework combining structure-aware DGD for design generation and surrogate modeling for design evaluation, we tested different types of VDR, including latent vectors (i.e., PartVAE vectors and SPVAE vectors) obtained from the generative design module and the 3D point grid (3DPG) vectors.

We observed that SPVAE vectors directly from the structure-aware generative design module achieved the worst prediction accuracy regardless of the design cases and AutoML frameworks used. The results indicate that while latent vectors are commonly used as VDRs for surrogate models, they may not be suitable when the encoded information contains factors (e.g., structural information) that are of little relevance to the engineering performance of interest. Therefore, it is crucial to consider the physics underlying the engineering performance investigated and select VDRs that incorporate the most relevant information to improve the prediction accuracy of surrogate models. The results could have a broader impact on industry professionals because the use of appropriate VDR can lead to the improved predictive performance of design automation tools. A better prediction of engineering performance will also help designers make informed decisions in the early design stage when interacting with AI, facing a large number of design alternatives generated, thus potentially shortening the overall design cycle and reducing the development time.

The limitations presented in Section 7.5 help us identify some future research directions in the development of more practical design applications for structure-aware generative design. First, the structural integrity of the generated designs is assessed through visual inspection, without a quantitative method. Developing an automatic and quantitative evaluation method for structural validity will greatly benefit future applications of structure-aware generative design. Second, we demonstrate our method in scenarios where the engineering performance of a product is closely related to its shape geometry. Interestingly, we observed that the inclusion of structural information could have a detrimental effect on the suitability of SPVAE vectors as VDRs for surrogate modeling. More research and investigation are necessary to explore design cases in which structural information can significantly impact engineering performance, so we can test whether VDRs that incorporate both structural and geometric information may offer advantages over those solely capturing geometric information. Furthermore, to generalize the findings of this study, it is important to test more design cases or collect additional data for the two design cases. This would allow a deeper understanding and a wider application of the conclusions drawn from the study.

Chapter 8: Closing Thoughts and Future Work

This chapter summarizes the accomplished research work, contributions, and broader impacts of this dissertation. The challenges of the study of human-supervised data-driven generative design are identified which also lead to thoughts for future research.

8.1 Conclusions, Contributions, and Broader Impacts

8.1.1 Conclusions

Towards Human-Centered Generative Design, my dissertation research is motivated to answer the **Central Research Question** *"In what ways and to what extent can human designers' intent and preferences be incorporated as input to actively interact and guide the GD process to improve the quality and relevance of the design outcomes?"*. More specifically, we try to answer three Research Questions (RQs) as shown below.

- ***RQ 1:*** *How feasible and to what extent can cross-modal synthesis methods with unimodal input incorporate human designers' intent and preferences as input to guide the data-driven design generation?*
- ***RQ 2:*** *How feasible and to what extent can cross-modal synthesis methods with multimodal inputs incorporate human designers' intent and preferences as input to guide the data-driven design generation?*
- ***RQ 3:*** *What are the effects of different representations of the generated designs on the data-driven design evaluation?*

My **Central Research Hypothesis** posits that human designers' intent and preferences can be incorporated as input to guide the data-driven design generation using cross-modal synthesis. However, the development of such methods for engineering design needs to tackle several fundamental challenges (Li et al., 2023b). These challenges include the scarcity of design data, complexities in 3D design representations, large semantic gaps between different modalities and thus challenging to maintain design integrity and intent, and vectorized representations for AI training.

To tackle the identified challenges, I have developed a set of strategies that include synthesizing datasets, collecting data from human participants using crowdsourcing platforms, designing innovative neural network architectures, and formulating novel vectorized design representations for 3D design data applicable to both the generation and evaluation of designs. The strategies proposed can (a) facilitate active human engagement and direction in the generative design (GD) process through cross-modal synthesis, accommodating both unimodal and multimodal inputs, and (b) assess a wide array of generated design concepts efficiently and effectively by employing appropriate vectorized design representations and surrogate models. These techniques establish the foundation for the Human-Supervised Generative Design (HSGD) Framework as depicted in Figure 1.3. This groundbreaking framework seamlessly incorporates human insights, skills, and feedback into GD practices, marking a significant advancement in the field. It enhances the speed of the design iteration cycle during the conceptual design stage and incorporates considerations of downstream design earlier into the initial stages of decision-making. From the results, we conclude that:

- Cross-modal synthesis methods, capable of processing either unimodal or multimodal inputs, exhibit significant potential in capturing and integrating human designers' intent and preferences to guide the generation of data-driven generative design. Although textual descriptions, sketches, and images may not fully encapsulate the designers' envisioned ideas—a challenge also prevalent in traditional design practices—our cross-modal synthesis approaches can still discern and interpret the underlying design preferences and intentions from these varied input modalities. Consequently, these methods are designed to generate 3D designs that are in closer alignment with the specified design requirements embedded in the input design modality, thereby bridging the gap between conceptual intent and tangible design outcomes.

- The choice of vectorized design representations significantly influences the evaluation of generated designs within a data-driven framework, particularly as product geometries become more intricate and as structure-aware generative design methodologies are employed. While it is common practice to utilize the learned latent spaces for these vectorized representations to expedite AI-assisted design evaluation and optimization, such latent vectors may prove unsuitable if they encapsulate information irrelevant to the engineering performance of interest. This observation underscores the imperative for designers to consider the suitability of vectorized design representations for evaluation purposes from the beginning of developing data-driven design methodologies. This foresight is crucial to ensure that the representations employed are conducive to an accurate and meaningful assessment of the performance of designs, aligning with engineering objectives and requirements.

8.1.2 Contributions to Generative Design, Data-Driven Engineering Design, Human-AI Design Collaboration

Product shape design is one of the most paramount aspects of product development. Methods for deep learning of cross-modal tasks (DLCMTs), such as the cross-modal synthesis method, can transfer one design modality (e.g., text, sketch, image, and 3D designs) to another, and human preferences can be reflected by design modalities. Thus, we conducted the first systematic literature review of DLCMT methods and identified the opportunities and challenges in applying them to human-supervised generative design (HSGD) of product shapes. Building upon this knowledge foundation, we have developed a novel target-embedding variational autoencoder (TEVAE) architecture tailored for HSGD. The approach has been applied to the generative design of 3D shapes from sketches. Sketches are used to represent human preferences to guide GD of 3D designs. The required input takes the simplest format as silhouette contour sketches that designers can easily draw and modify. We have also developed a preliminary graphical user interface to enable interactive HSGD. In addition to sketches, we have been exploring the feasibility of utilizing various design modalities, such as images and text, to represent human preferences and guidance. We have also developed a method for generating CAD sequences from images. Moreover, we have also quantitatively evaluated the potential for utilizing multimodal large language models for HSGD and proposed ways, such as a debugger, to improve their capability to generate 3D CAD concepts. These innovative methods have established the groundwork for the HSGD Framework, enabling a more seamless integration of human preferences, expertise, and feedback with GD and creating innovative ways to promote data-driven design innovation.

AI-assisted fast concept evaluation methods are crucial for evaluating numerous design concepts created from GD. Moreover, engineering products are often systems that include several interconnected subsystems or components with a specific structure. Consequently, we have conducted research into fast evaluation methods in the context of structure-aware generative design (SAGD). It is vital to identify the suitable vectorized design representation (VDR) for evaluating designs in GD, which remains largely unexplored in SAGD. To that end, we built surrogate models for the fast evaluation method and conducted a comparative analysis of surrogate models' performance in predicting the engineering performance of 3D shapes using VDRs from two sources: the trained latent space of SAGD models encoding structural and geometric information and an embedding method encoding only geometric information. This study provides empirical evidence for the effectiveness of different types of VDRs in SAGD for surrogate modeling, thus facilitating the construction of better AI-assisted evaluation methods.

8.1.3 Summary of Contributions

In summary, this dissertation advances human-centered generative design by addressing a crucial gap in existing methodologies, facilitating a human-centered approach. We introduce a novel *Human-Supervised Data-Driven Generative Design Framework* incorporating cross-modal synthesis methods and AI-assisted design evaluation techniques. Our approach enhances human control and interaction within the GD process. We have developed an innovative neural network architecture tailored for cross-modal synthesis in engineering design. This architecture effectively incorporates human intent and preferences into the generation of 3D design concepts. Our methodologies significantly accelerate design ideation, improve exploration of design spaces, and integrate downstream considerations into early-stage decision-making. They might have broad applicability across industries, speeding up product development cycles and reducing associated costs. Moreover, these methods can be adapted into educational tools for design students, preparing them for careers in an evolving landscape that increasingly values human-AI collaboration in engineering design.

8.1.4 Broader Impacts on Engineering Design Education

My dissertation research has also generated broader impacts, particularly in shaping the concept of Generative Design Thinking (GDT), the cognitive process involved in GD (Li et al., 2021b; Goldstein et al., 2021; Brown et al., 2024; Clay et al., 2023, 2024; Koolman et al., 2024). To define GDT, we have introduced the Evolving Design Thinking (EDT) model, depicting the evolution of design thinking across three levels: Design Technology, Design Thinking, and Design Cognition. Our research seeks to conduct a comprehensive examination of the cognitive processes associated with GD to improve GD education and equip future generative designers with the necessary technological skills and design thinking. In addition, we have also developed a curriculum that traces the evolution of design paradigms: traditional design, parametric design, and GD. We have been gathering data through thinkaloud sessions using the design curriculum from students. The collected data will inform curriculum improvements and deepen our understanding of student cognition across traditional, parametric, and GD paradigms, ultimately leading to an enhanced education of GD.

8.2 Future Work

Future work in this domain will focus on overcoming the challenges of merging AI's computational power with the unique creative, intuitive, and insightful qualities of human designers to elevate human-AI collaboration in engineering design. Additionally, there is a pressing need to address the complexities involved in training AI for creativity in engineering design, confronting obstacles such as the scarcity of design data, the selection of suitable design representation and vectorization, and the semantic gaps between different modalities and 3D designs. These challenges complicate the preservation of design integrity and intent across modal transitions.

In this section, we introduce three potential avenues for future research, expanding upon the foundational work established in my dissertation. Firstly, a natural extension of my dissertation research is to explore deeper human-centered generative design (Demirel et al., 2023a). Little is known about how generative design processes can be reoriented to prioritize human factors (such as safety, and comfortability). This direction not only promises to extend the theoretical frameworks established in my dissertation but also to offer practical insights into designing more responsive and user-friendly systems. In addition, a pressing concern within the field of generative AI in product design is the technology's current inclination toward prioritizing visual aesthetics over functional practicality. This bias raises important questions about the balance between form and function in the design process, and how AI technologies might be developed or adapted to better address functional requirements without sacrificing aesthetic appeal. Investigating this issue could lead to more balanced approaches in the use of generative AI, ensuring that future product designs are not only visually compelling but also practically useful and user-centered. Lastly, manufacturing's demand for innovation, efficiency, and precision is growing, requiring the seamless integration of design and manufacturing phases. The advent of multimodal large language models, such as GPT-4 Vision (GPT-4V), promises a revolutionary shift by proficiently handling a mix of text and image data. This capability is crucial for closing the gap between design and manufacturing stages, potentially transforming the entire product lifecycle by promoting a more synchronized and intelligent workflow. It could be interesting and meaningful to explore how to redefine manufacturing by incorporating cutting-edge AI technologies.

Appendix A: Appendix for Chapter 3

A.1 Details of Literature Search

As introduced in Section 3.3.2, Table A.1 shows the number of articles found in major literature databases using keywords of “text-to-sketch retrieval” (TSkRet), “text-to-sketch generation” (TSkG), “text-to-shape retrieval” (TShRet), “text-to-shape generation” (TShG), “sketch-based 3D shape retrieval” (SkShRet), “sketch-based 3D shape generation” (SkShG), “sketch-based 3D shape reconstruction” (SkShRec), “sketch-based 3D shape synthesis” (SkShSyn), and “3D shape reconstruction from sketches” (ShRecSk). In addition, we used the time range of January 2021 to June 2022 to search for the most recent studies for sketch-to-3D shape retrieval and generation, the number of which is indicated in parentheses (e.g., (35) for ShRecSk).

Figure A.1 shows the articles that are most relevant to the two key articles ((Wang et al., 2015; Lun et al., 2017)) using Connected Papers (accessed in June 2022). Studies that meet the scope of our review are indicated using a quadrilateral in each sub-figure.

Table A.1: Studies found in major databases

	Keywords (Double quotation marks included)								
	TSkRet	TSkG	TShRet	TShG	SkShRet	SkShG	SkShRec	SkShSyn	ShRecSk
ScienceDirect	0	0	0	0	2	0	0	0	0
Web of Science	0	0	1	0	20	1	0	0	1
Scopus	0	0	1	1	454	5	1	0	95
IEEExplore	0	0	0	1	13	1	1	0	1
ACM Digital Libraries	0	0	1	0	14	0	0	0	3
Google Scholar	0	3	7	22	559 (96)	7 (5)	5 (2)	1 (0)	120 (35)
Total	0	3	10	24	1062	14	7	1	220

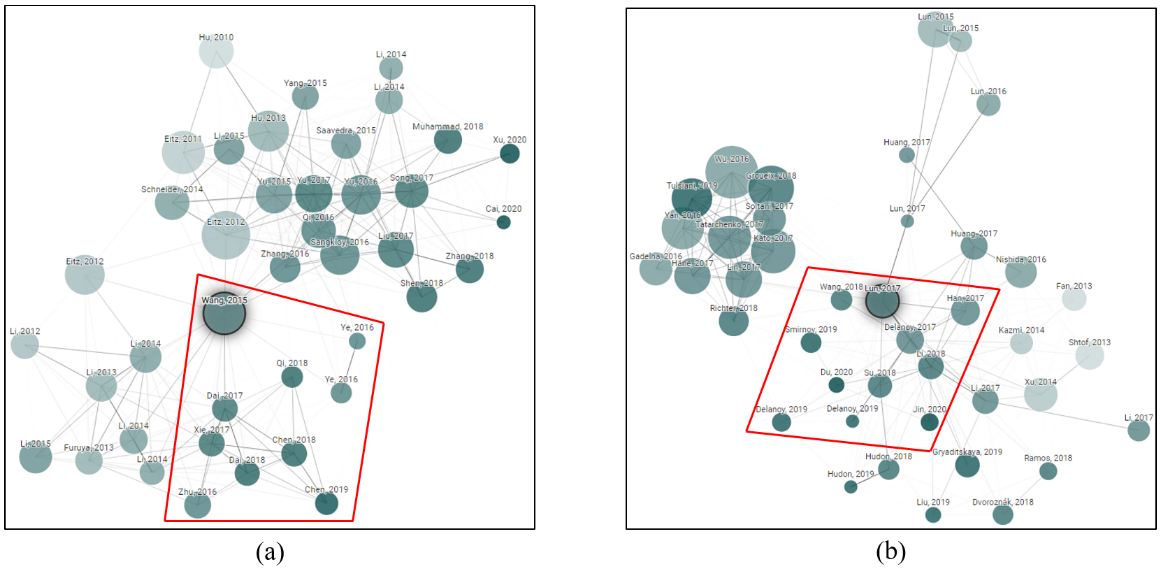


Figure A.1: (a) Studies for sketch-to-3D retrieval that are similar to (Wang et al., 2015); (b) Studies for sketch-to-3D generation that are similar to (Lun et al., 2017)

A.2 Paper Summary

We summarize and tabulate all 50 articles reviewed in Table A.2. There are 11 source journals and 20 conference proceedings, and their acronyms are shown below.

CG	Computers & Graphics
MS	Multimedia Systems
VC	The Visual Computer
CGF	Computer Graphics Forum
AEI	Advanced Engineering Informatics
TIP	IEEE Transactions on Image Processing
MTA	Multimedia Tools and Applications
TOG	ACM Transactions on Graphics
JMD	Journal of Mechanical Design
WCMC	Wireless Communications and Mobile Computing
PACMCGIT	The Proceedings of the ACM in Computer Graphics and Interactive Techniques
MM	International Conference on Multimedia
IUI	International Conference on Intelligent User Interfaces
CHI	Conference on Human Factors in Computing Systems
I3D	Symposium on Interactive 3D Graphics and Games
MMM	International Conference on Multimedia Modeling
IMX	ACM International Conference on Interactive Media Experiences
CVPR	Computer Vision and Pattern Recognition Conference
ICCV	International Conference on Computer Vision
ECCV	European Conference on Computer Vision
ACCV	Asian Conference on Computer Vision
AAAI	Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence
ICIP	IEEE International Conference on Image Processing
UIST	Annual ACM Symposium on User Interface Software and Technology
ICCS	International Conference on Computational Science
ICPR	International Conference on Pattern Recognition
ICLR	International Conference on Learning Representations
ICVRV	International Conference on Virtual Reality and Visualization
3DIMPVT	International Conference on 3D Imaging Modeling, Processing, Visualization and Transmission
VISIGRAPP	International Joint Conference on Computer Vision Imaging and Computer Graphics Theory and Applications
ICCEIA-VR	International Conference on Computer Engineering and Innovative Application of VR

Table A.2: Summary of Literature

Type of DLCMT	Reference	Method	Text Type	Sketch Type	3D Representation	Dataset	Object Class	Generalizability beyond trained classes	User Interface	User Study	Publication Source
Text to 3D shape retrieval	Han et al. (2019)	CNN, GRU	NLD	N/A	Voxel	3D-text dataset Chen et al. (2018)	Chairs, tables	No	No	No	Conference: AAAI
	(Chen et al., 2018)	Text encoder (CNN, GRU), shape encoder (3DCNN)	NLD	N/A	Voxel	Propose 3D-text dataset from ShapeNet (Chang et al., 2015)	Chairs, tables, synthetic objects	No	No	No	Conference: ACCV
Text to 3D shape generation	Jain et al. (2022)	Network based on CLIP (Radford et al., 2021)	NLD	N/A	NeRF	COCO (Lin et al., 2014)	Diverse classes	Yes	No	No	Conference: CVPR
	Sanghi et al. (2022)	Network based on PointNet (Qi et al., 2017), CLIP (Radford et al., 2021), OccNet (Mescheder et al., 2019)	Object names	N/A	Voxel	ShapeNet (Chang et al., 2015)	Diverse classes	No	No	No	Conference: CVPR
	Liu et al. (2022)	Shape autoencoder, word-level spatial transformer, shape generator (IMLE (Chen and Zhang, 2019))	NLD	N/A	Implicit representation, mesh	3D-text dataset (Chen et al., 2018)	Chairs, tables	No	No	No	Conference: CVPR
	Jahan et al. (2021)	Shape encoder, decoder label regression network,	Semantic keywords	N/A	Implicit representation, mesh	COSEG (Wang et al., 2012), ModelNet (Wu et al., 2015b)	Chairs, tables, lamps	No	No	No	Journal: CGF
	Li et al. (2020a)	GAN based network	NLD	N/A	Voxel	3D-text dataset (Chen et al., 2018)	Chairs, tables	No	No	No	Conference: ICVRV
	Chen et al. (2018)	Text encoder (CNN, GRU), shape encoder (3DCNN), GAN	NLD	N/A	Voxel	Propose 3D-text dataset from ShapeNet (Chang et al., 2015)	Chairs, tables, synthetic objects	No	No	No	Conference: ACCV
Text to sketch generation	Yuan et al. (2021)	GAN, Bi-LSTM	NLD	Static pixel space	N/A	Propose SketchCUB from CUB (Wah et al., 2011)	Birds	No	No	Yes	Conference: CVPR
	Huang et al. (2020a)	Composition proposer (transformer), object generator (Sketch-RNN (Ha and Eck, 2018))	NLD	Dynamic stroke coordinate space	N/A	CoDraw (Kim et al., 2017)	Diverse classes	Yes	Yes	Yes	Conference: IUI
	Huang and Canny (2019)	Scene composer (transformer), object sketcher (Sketch-RNN (Ha and Eck, 2018))	NLD	Dynamic stroke coordinate space	N/A	Visual Genome (Krishna et al., 2017), Quick, Draw! (Jongejan et al., 2016)	Diverse classes	Yes	Yes	Yes	Conference: UIST
	Wang et al. (2018b)	GAN based network	NLD	Static pixel space	N/A	Propose Text2Sketch from CUFSF (Zhang et al., 2011)	Human faces	No	No	No	Conference: ICIP

Table A.2: Continued

Type of DLCMT	Reference	Method	Text Type	Sketch Type	3D Representation	Dataset	Object Class	Generalizability beyond trained classes	User Interface	User Study	Publication Source
Sketch to 3D shape retrieval	Qin et al. (2022)	Autoencoder (GRASS (Li et al., 2017)), k-nearest neighbors	N/A	Static pixel space	B-Rep	Propose CAD model-sketches dataset	Diverse classes	Yes	Yes	No	Journal: AEI
	Yang et al. (2022)	3D model network, 2D sketch network (MVCNN (Su et al., 2015))	N/A	Static pixel space	Mesh	SHREC13 (Li et al., 2013), SHREC14 (Li et al., 2014b), SHREC16 (Li et al., 2016)	Diverse classes	Yes	No	No	Journal: MS
	Qi et al. (2021)	Sketch encoder, shape encoder (MVCNN (Su et al., 2015))	N/A	Static pixel space	Mesh	Propose fine-grained dataset from ShapeNet (Chang et al., 2015)	Chairs, lamps	No	No	No	Journal: TIP
	Manda et al. (2021)	MVCNN (Su et al., 2015), GVCNN (Feng et al., 2018), RotationNet (Kanezaki et al., 2019), MVCNN-SA (Shajahan et al., 2019)	N/A	Static pixel space	B-Rep	Propose CADSketchNet from ESB (Jayanti et al., 2006), MCB (Kim et al., 2020)	Diverse classes	Yes	No	No	Journal: CG
	Liang et al. (2021)	Sketch network, view network	N/A	Static pixel space	Mesh	SHREC13, SHREC14	Diverse classes	Yes	No	No	Journal: TIP
	Liu and Zhao (2021)	MVCNN (Su et al., 2015), Guidance Cleaning Network	N/A	Static pixel space	Mesh	SHREC13, SHREC14	Diverse classes	Yes	No	No	Conference: ICCEIA-VR
	Xia et al. (2021)	Student network, teacher network (MVCNN (Su et al., 2015))	N/A	Static pixel space	Mesh	SHREC13	Diverse classes	Yes	No	No	Conference: ICCS
	Li et al. (2021a)	CNN based network	N/A	Type II 3D sketch	Mesh	SHREC16STB (Ye et al., 2016)	Diverse classes	Yes	Yes	No	Journal: MTA
	Navarro et al. (2021)	CNN based network	N/A	Static pixel space	Mesh	Propose a line drawing dataset from ShapeNet (Chang et al., 2015)	Diverse classes	Yes	No	No	Journal: CGF
	Chen et al. (2019)	Sketch network, segmented stochastic-viewing shape network, view attention network	N/A	Static pixel space	Mesh	SHREC13, SHREC14, PART-SHREC14 (Qi et al., 2018)	Diverse classes	Yes	No	No	Conference: CVPR
	Dai et al. (2018)	Source domain network, target domain network (3D-SIFT (Darom and Keller, 2012)	N/A	Static pixel space	Mesh	SHREC13, SHREC14, SHREC16	Diverse classes	Yes	No	No	Journal: TIP
	Chen and Fang (2018)	MVCNN (Su et al., 2015), GAN, metric network, cross-modality transformation network	N/A	Static pixel space	Mesh	SHREC13, SHREC14	Diverse classes	Yes	No	No	Conference: ECCV

Table A.2: Continued

Type of DLCMT	Reference	Method	Text Type	Sketch Type	3D Representation	Dataset	Object Class	Generalizability beyond trained classes	User Interface	User Study	Publication Source
Sketch to 3D shape retrieval	Dai et al. (2017)	Source domain network, target domain network (3D-SIFT (Darom and Keller, 2012))	N/A	Static pixel space	Mesh	SHREC13 (Li et al., 2013), SHREC14 (Li et al., 2014b)	Diverse classes	Yes	No	No	Conference: AAAI
	Xie et al. (2017)	CNN, metric network	N/A	Static pixel space	Mesh	SHREC13, SHREC14	Diverse classes	Yes	No	No	Conference: CVPR
	Zhu et al. (2016)	Cross-domain neural network, pyramid cross-domain network	N/A	Static pixel space	Mesh	SHREC14	Diverse classes	Yes	No	No	Conference: AAAI
	Ye et al. (2016)	CNN based network	N/A	Type II 3D sketch	Mesh	Propose SHREC16STB	Diverse classes	Yes	No	No	Conference: ICPR
	Wang et al. (2015)	CNN, Siamese network	N/A	Static pixel space	Mesh	PSB (Shilane et al., 2004), SHREC13, SHREC14	Diverse classes	Yes	No	No	Conference: CVPR
Sketch and text to 3D shape retrieval	Stemasov et al. (2022)	Flask representation state transfer, HoloLens	NLD	Type II 3D sketch	Mesh, voxel	Thingiverse, MyMiniFactory	Diverse classes	Yes	Yes	No	Conference: CHI
	Giunchi et al. (2021)	CNN based network	NLD	Type II 3D sketch	Mesh	Propose Variational Chairs dataset from ShapeNet (Chang et al., 2015)	Chairs	No	Yes	Yes	Conference: IMX
Sketch to 3D shape generation	Li et al. (2022c)	Target-embedding variational autoencoder	N/A	Static pixel space	Mesh	Dataset (Umetani, 2017)	Cars, cups	No	No	No	Journal: JMD
	Nozawa et al. (2022)	GAN, lazy learning	N/A	Static pixel space	Point cloud, mesh	ShapeNet (Chang et al., 2015)	Cars	No	No	No	Journal: VC
	Du et al. (2021)	CNN, OccNet (Mescheder et al., 2019), 3DCNN	N/A	Static pixel space	Implicit representation, mesh	PartNet (Mo et al., 2019c)	Chairs, tables, lamps	No	Yes	Yes	Journal: CGF
	Wang et al. (2021)	Sketch component segmentation network, transformation network, VAE	N/A	Static pixel space	Point cloud, mesh	Dataset (Lun et al., 2017)	Characters, airplanes, chairs	No	No	No	Journal: WCMC
	Guillard et al. (2021)	Encoder (MeshSDF (Remelli et al., 2020)), decoder, differential renderer	N/A	Static pixel space	Implicit representation, mesh	ShapeNet (Chang et al., 2015)	Cars, chairs	No	Yes	No	Conference: ICCV

Table A.2: Continued

Type of DLCMT	Reference	Method	Text Type	Sketch Type	3D Representation	Dataset	Object Class	Generalizability beyond trained classes	User Interface	User Study	Publication Source
Sketch to 3D shape generation	Zhang et al. (2021)	View-aware generation network (encoder, decoder), discriminator	N/A	Static pixel space	Mesh	ShapeNet-Sketch (Kar et al., 2017), Sketchy (Sangkloy et al., 2016), TuBerlin (Eitz et al., 2012)	Diverse classes	Yes	No	No	Conference: CVPR
	Yang et al. (2021)	CNN based network	N/A	Static pixel space	Mesh	AMASS (Mahmood et al., 2019)	Human bodies	No	No	No	Conference: MMM
	Luo et al. (2021)	Voxel-aligned implicit network, pixel-aligned implicit network	N/A	Static pixel space	Implicit representation, mesh	Propose 3DAnimalHead	Animal heads	No	Yes	Yes	Conference: UIST
	Jin et al. (2020)	VAE, volumetric autoencoder	N/A	Static pixel space	Voxel, mesh	PSB (Shilane et al., 2004), benchmark (Chen et al., 2009)	Diverse classes	Yes	No	No	Conference: I3D
	Smirnov et al. (2020)	CNN based network	N/A	Static pixel space	B-Rep, mesh	ShapeNet (Chang et al., 2015)	Diverse classes	No	No	No	Conference: ICLR
	Nozawa et al. (2020)	Encoder-decoder, lazy learning	N/A	Static pixel space	Point cloud, mesh	ShapeNet	Cars	No	No	No	Conference: VISIGRAPP
	Smirnov et al. (2019)	CNN based network	N/A	Static pixel space	B-Rep, mesh	ShapeNet	Diverse classes	No	No	No	Conference: ICLR
	Delanoy et al. (2019)	CNN based network	N/A	Type I 3D Sketch	Voxel	COSEG (Wang et al., 2012)	Chairs, vases, synthetic shapes	No	No	No	Journal: CG
	Wang et al. (2018a)	Autoencoder, GAN	N/A	Static pixel space	Voxel	SHREC13 (Li et al., 2013), ShapeNet	Chairs	No	No	No	Conference: MM
	Li et al. (2018)	DFNet (encoder-decoder), GeomNet (encoder-decoder)	N/A	Static pixel space	Mesh	Dataset (Lun et al., 2017)	Characters	No	Yes	Yes	Journal: TOG
	Delanoy et al. (2018)	Singleview CNN, updater CNN	N/A	Type I 3D Sketch	Voxel	COSEG (Wang et al., 2012)	Chairs, vases, synthetic shapes	No	Yes	Yes	Journal: PACMCGIT
	Lun et al. (2017)	Encoder, multiview decoder	N/A	Static pixel space	Point cloud, mesh	The Models Resource, ShapeNet	Characters, airplanes, chairs	No	No	Yes	Conference: 3DIMPVT
	Han et al. (2017)	Deep regression network	N/A	Static pixel space	Mesh	Faceware-house (Cao et al., 2013)	Face caricatures	No	Yes	Yes	Journal: TOG

Table A.2: Continued

Type of DLCMT	Reference	Method	Text Type	Sketch Type	3D Representation	Dataset	Object Class	Generalizability beyond trained classes	User Interface	User Study	Publication Source
Text to 3D shape manipulation	Liu et al. (2022)	Shape autoencoder, word-level spatial transformer, shape generator (IMLE (Chen and Zhang, 2019))	NLD	N/A	Implicit representation, mesh	3D-text dataset (Chen et al., 2018)	Chairs, tables	No	No	No	Conference: CVPR
	Wang et al. (2022a)	Disentangled conditional NeRF, GAN CLIP (Radford et al., 2021)	Semantic keywords, object names	N/A	NeRF (Mildenhall et al., 2020)	Photoshapes (Park et al., 2018), Carla (Dosovitskiy et al., 2017)	Chairs, cars	No	Yes	Yes	Conference: CVPR
	Michel et al. (2022)	Neural style filed network, differentiable renderer, CLIP (Radford et al., 2021)	Semantic keywords, object names	N/A	Mesh	COSEG (Wang et al., 2012), Thingi10K (Zhou and Jacobson, 2016), ShapeNet (Chang et al., 2015), Turbo Squid, ModelNet (Wu et al., 2015b)	Diverse classes	Yes	No	Yes	Conference: CVPR
Sketch to 3D shape manipulation	Guillard et al. (2021)	Encoder (MeshSDF (Remelli et al., 2020)), decoder, differential renderer	N/A	Static pixel space	Implicit representation, mesh	ShapeNet	Cars, chairs	No	Yes	No	Conference: ICCV
	Jin et al. (2020)	VAE, volumetric autoencoder	N/A	Static pixel space	Voxel, mesh	PSB (Shilane et al., 2004), Benchmark (Chen et al., 2009)	Diverse classes	Yes	No	No	Conference: I3D

Appendix B: Appendix for Chapter 4

This is the Supplementary Material for Chapter 4. We show more details of the approach and the training of the E^2D network.

B.1 Part 1: The E^2D Network

We construct the E^2D network by concatenating an encoder ($Enc_2(\cdot)$) to a mesh VAE Yuan et al. (2020). The $Enc_1(\cdot)$ has the following network structure: two graph convolutional (Conv) layers with a batch normalization (BN) layer and a *tanh* activation function following each graph Conv layer, one pooling layer, and a third graph Conv layer. The output of the last Conv layer is mapped to a 128-dimensional space that contains a mean vector ($\mu_{\mathbf{1}} \in \mathbb{R}^{128}$) and a deviation vector ($\sigma \in \mathbb{R}^{128}$) by two different fully-connected (FC) layers. The mean vector does not have an activation function and the deviation vector uses *sigmoid* as the activation function. The $Enc_2(\cdot)$ shares the same network structure as $Enc_1(\cdot)$ except that only one FC layer is used to form the latent vector ($\mu_{\mathbf{2}} \in \mathbb{R}^{128}$) because the standard deviation vector is not needed here. $Enc_1(\cdot)$ takes Y and $Enc_2(\cdot)$ takes X as input, where $X \in \mathbb{R}^{V\times 9}$ and $Y \in \mathbb{R}^{V\times 9}$ are feature representations of source shapes (S_s) and target shapes (S_t), respectively. V represents the number of vertices of a 3D mesh shape. S_s are 3D extrusion mesh models extruded from sideview sketches of the original authentic 3D mesh models (S_t).

The decoder ($Dec(\cdot)$) mirrors the encoders and it consists of an FC layer, a graph Conv layer and a de-pooling layer, followed by two graph Conv layers. Each Conv layer is connected to a BN layer and a *tanh* activation function except the third Conv layer. The $Dec(\cdot)$ takes the latent vector $\mathbf{z_1} = \mu_{\mathbf{1}} + \sigma\epsilon$ as input, where $\epsilon \in N(\mathbf{0}, \mathbf{I})$ which is a standard multivariate Gaussian distribution. The output of $Dec(\cdot)$ is $\hat{Y} \in \mathbb{R}^{V\times 9}$ which has the same dimension as the input Y and can be used to reconstruct 3D mesh shapes.

The entire E^2D network is trained by minimizing the following loss function:

$$L_{total} = \lambda L_1 + L_2, \tag{B.1}$$

where L_2 is the loss funciton for $Enc_2(\cdot)$ which has been shown in the paper, λ is a weight parameter, and L_1 is the loss function for the mesh VAE. L_1 can be written as:

$$L_1 = \alpha_1 L_{Recon} + \alpha_2 L_{KL} + L_{Reg}, \tag{B.2}$$

where α_1 and α_2 are the weights of different loss terms, and

$$L_{Recon} = \frac{1}{2M} \sum_{i=1}^{M} \left\| Y^i - \hat{Y}^i \right\|_F^2 \tag{B.3}$$

denotes the mean squared error (MSE) reconstruction loss, where Y^i and $\hat{Y}^i$ represent the input feature matrix of the i^{th} model and the corresponding output of the mesh VAE, respectively. $\|\cdot\|_F$ is the Frobenius norm of the matrix and M is the number of mesh shapes in the training dataset.

$$L_{KL} = D_{KL}(q(\mathbf{z_1} \| Y) \| p(\mathbf{z_1})) \tag{B.4}$$

represents the Kullback–Leibler (KL) divergence loss to promote the Gaussian distribution in the latent space, where $\mathbf{z_1}$ is the latent vector, $p(\mathbf{z_1})$ is the prior probability, $q(\mathbf{z_1} \| Y)$ is the posterior distribution given the feature matrix Y, and D_{KL} is the KL-divergence. L_{Reg} is the squared l_2 norm regularization loss of the parameters which is used to avoid overfitting to the training data to improve the generalization ability of the mesh VAE.

For the two-stage training in our application, in Stage 1, the mesh VAE is trained independently. The network is initialized at random and trained end-to-end by minimizing L_1 as shown in Equation (B.2). The 128-dimensional mean vectors $\mu_\mathbf{1}$ from the latent space of the trained mesh VAE, will be used in Stage 2 training. In Stage 2, we fix all the learning parameters of the trained mesh VAE and train $Enc_2(\cdot)$. $Enc_2(\cdot)$ is also initialized randomly. It is trained by minimizing L_2 as shown in Equations 4.1 and 4.2.

B.2 Part 2: Data Pair Preparation

We present more details of the process of how we prepare training data pairs (see Fig 4.1(b)) using a car body as an example. Data can be processed in the following steps.

1) Obtain the sideview image from an authentic 3D car mesh shape.

2) Get the binary image from the sideview image.

3) Extract the contour of the binary image, which results in a dense point set.

4) Get a simplified point set using a simple contour approximation method provided in the OpenCV-python package.

5) Get an extrusion shape using the FreeCAD Python API. Then, the original authentic 3D mesh shape (target shape) and the extrusion shape (source shape) form one data pair.

An approximation method is used in Step 4 because the extrusion model directly from the original contour points (a dense point set) has zigzags on its surface, which affects the quality of the extruded shapes. The contour approximation method can keep more points where the geometry is more complex (e.g., bumper lines, windshield lines) and fewer points where the geometry is simpler (e.g., the bottom line). In our implementation, as shown in Fig. B.1, this simplification process results in a point set that has around 400 points (varies a little for different car models), and we reduce the number of points by a factor of 1/4 to around 100 points for a further avoidance of zigzags on the models' surface, which can still preserve the contour shape well thanks to the simplification method. It can be observed that the resulting extrusion mesh model using the finally simplified point set has a smoother surface than that using the point set directly after the simplification. In addition, to make sure the extruded model has an equivalent scale as the original authentic model, for each point

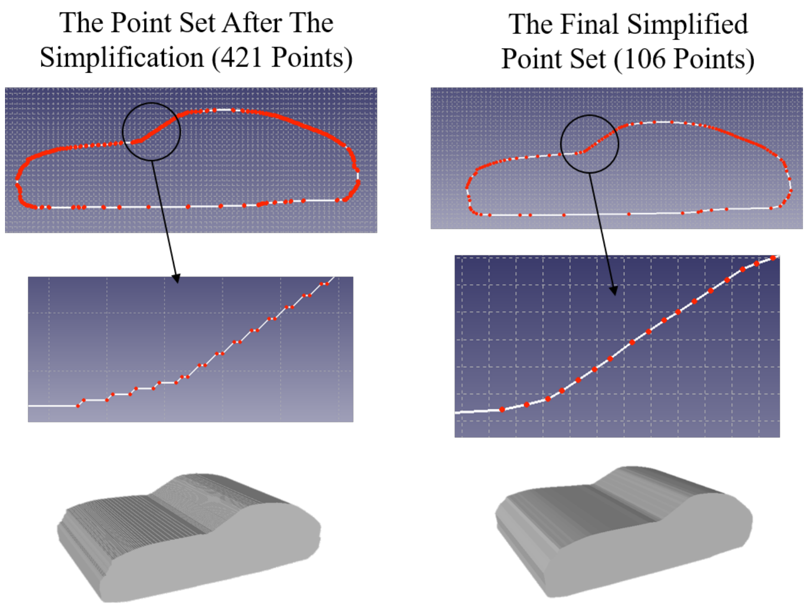

The Point Set After The Simplification (421 Points)
The Final Simplified Point Set (106 Points)

Figure B.1: An example showing the comparison between resulting extrusion mesh models using a point set after the approximation method and the final simplified point set

set, we calculate the diagonal length of its bounding box $Diag^{Pts}$ and the diagonal length of the bounding box of sideview of its corresponding original authentic mesh model $Diag^{S}$ as illustrated by blue dash lines in Fig. 4.1(b). We then scale down the coordinates of the finally simplified point set by a factor of $Diag^{S}/Diag^{Pts}$. We also move the center of the bounding box of the point set to the origin $(0, 0)$. The top right image in Fig. B.1 shows an example of the final simplified point set displayed in FreeCAD.

In Step 5, to obtain the extrusion model, we need to specify the extrusion depth in the orthogonal direction (z-axis) on the sideview. We use the z dimension size of the bounding box of the original authentic 3D mesh model as the extrusion depth. We store the x and y coordinates of the simplified point set and the extrusion depth in a CSV file in Step 4, which will be used as input for a Python script using the FreeCAD python API to generate a 2D sideview sketch first and then extrude the sketch to an extrusion mesh model.

We develop a set of Python scripts that fully automate the whole process, and the scripts are made open-source for the community [1].

B.3 Part 3: Feature Representation

We present more details of how we apply the as-consistent-as-possible (ACAP) algorithm Gao et al. (2019a) to obtain feature representations of 3D mesh shapes. Given a set of 3D mesh shapes with the same topology, each shape is represented by S_m, where $m \in [1, ..., n]$. $p_{m,i} \in \mathbb{R}^3$ is denoted as the i^{th} vertex of the m^{th} shape S_m. The first shape S_1 is the reference shape. Let N_i represent the index set of 1-ring neighbors of the i^{th} vertex on a 3D shape. 1-ring neighbors are all adjacent vertices that are connected to a vertex with one edge. We can then get the deformation matrix $T_{m,i} \in \mathbb{R}^{3\times 3}$ that represents the local shape deformation by Equation (B.5).

$$\arg\min_{T_{m,i}} \sum_{j \in N_i} c_{i,j} \left\| (p_{m,i} - p_{m,j}) - T_{m,i}(p_{1,i} - p_{1,j}) \right\|_2^2 , \tag{B.5}$$

where $c_{i,j}$ is the cotangent weight. Using polar decomposition, we can then get $T_{m,i} = R_{m,i}S_{m,i}$, where $R_{m,i} \in \mathbb{R}^{3\times 3}$ is an orthogonal matrix representing rotation deformation and $S_{m,i} \in \mathbb{R}^{3\times 3}$ is a real symmetry matrix describing the scale and shear deformation. $\log R_{m,i}$ is a skew-symmetry matrix, from which we can extract 3 entries (i.e., the upper triangular matrix excluding the diagonal elements that are always zeros). Additionally, we can get 6 entries (i.e., the upper triangular matrix that includes diagonal elements) from $S_{m,i}$ since it is a symmetry matrix. Thus, for each vertex, 9 features can be obtained and concatenated to a 9-dimensional vector $\mathbf{q_{m,i}}$. In a result, a mesh shape with V vertices can be represented by a feature matrix $M \in \mathbb{R}^{V\times 9}$. The first shape S_1 is a uniform cube with 9602 vertices, which is the same as the one used in the registration process, so $V = 9602$ in our applications.

[1]`https://github.com/Xingang1990/TEVAE`

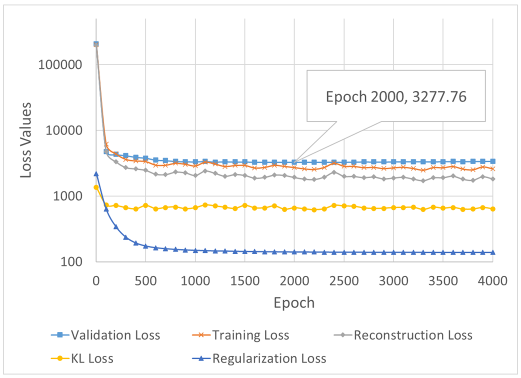
B.4 Part 4: Training Details of the Approach

We present the details of the training process using the case study of car design as an example, and the training of the neural network for mug shape generation follows the same procedure. We apply a two-stage training strategy to train the E^2D network. We set $\lambda = 1$ in the total loss function as shown in Equation (B.1), the value of which does not affect the training in our case *per se*, because we actually minimize L_1 and L_2 independently.

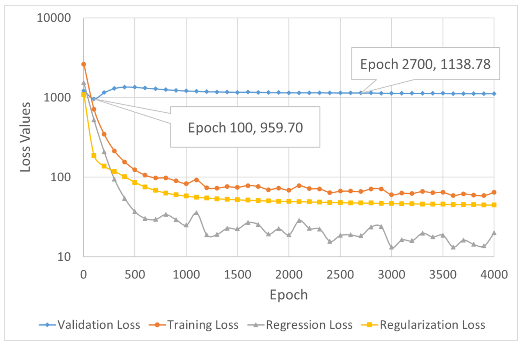
For training the mesh VAE in Stage 1 (i.e., $Enc_1(\cdot)$ and $Dec(\cdot)$), the input target shape dataset is randomly split into the training set (80%) and testing set (20%). The generalization ability of the mesh VAE (i.e., whether the trained model is overfitting to the training data or not) is evaluated by the validation loss on the testing data (unseen data). It is also worth investigating the impact of the L_{KL} term on the performance of the generative network. By trial-and-error, we find that $\alpha_1 = 40$ and $\alpha_2 = 10$ for Equation (B.2) produces the best results in terms of the reconstruction loss, validation loss, KL-divergence loss, and the regularization loss. In minimizing the losses, the Adam optimizer is applied with a learning rate of $lr_1 = 0.0001$. The batch size is set as 32, and the training batches are randomly sampled from the training dataset. We train the mesh VAE 4000 epochs and save the best model that has the least validation loss.

For training the $Enc_2(\cdot)$ network in Stage 2, we also do an 80-20 split, and meanwhile, use the data pair to make sure the i^{th} target shape (S_t^i) corresponds to the i^{th} source shape (S_s^i) in both the training and testing sets. This ensures that the 20% testing shapes are always unseen data. We set $\alpha = 1$ in Equation 4.2. The optimizer, the learning rate lr_2, and the number of epochs follow the same setting of Stage 1.

For stage 1, the loss values during the training process of car models are reported in Fig. B.2(a). It can be observed that the network is learning, and all loss terms start converging at around epoch 1000. The regularization loss is also gradually



(a)



(b)

Figure B.2: (a) The loss values from Stage 1 training of the car models; (b) The loss values from Stage 2 training of the car models

reduced, which can prevent the network from overfitting to the training dataset, and thus improve the generalization ability of the network. For stage 2, the loss values in every 100 epochs are shown in Fig. B.2(b). Note that the validation loss achieves the least value at epoch 100 and then converges to a higher loss value. However, in light of the total loss and the overall performance of the $Enc_2(\cdot)$ network, we select epoch 2700 as the best model, which has the second least validation loss, but with much lower regression loss and regularization loss compared to those at epoch 100.

Appendix C: Appendix for Chapter 7

The training loss values for both the car and aircraft models were recorded and documented in Figure C.1. It was observed that all loss terms reached convergence, indicating successful training. Specifically, for car models, the PartVAEs were trained for 10000 epochs, followed by training the SPVAE for 20000 epochs. On the other hand, for the aircraft models, the PartVAEs were trained for 5000 epochs, followed by training the SPVAE for 10000 epochs.

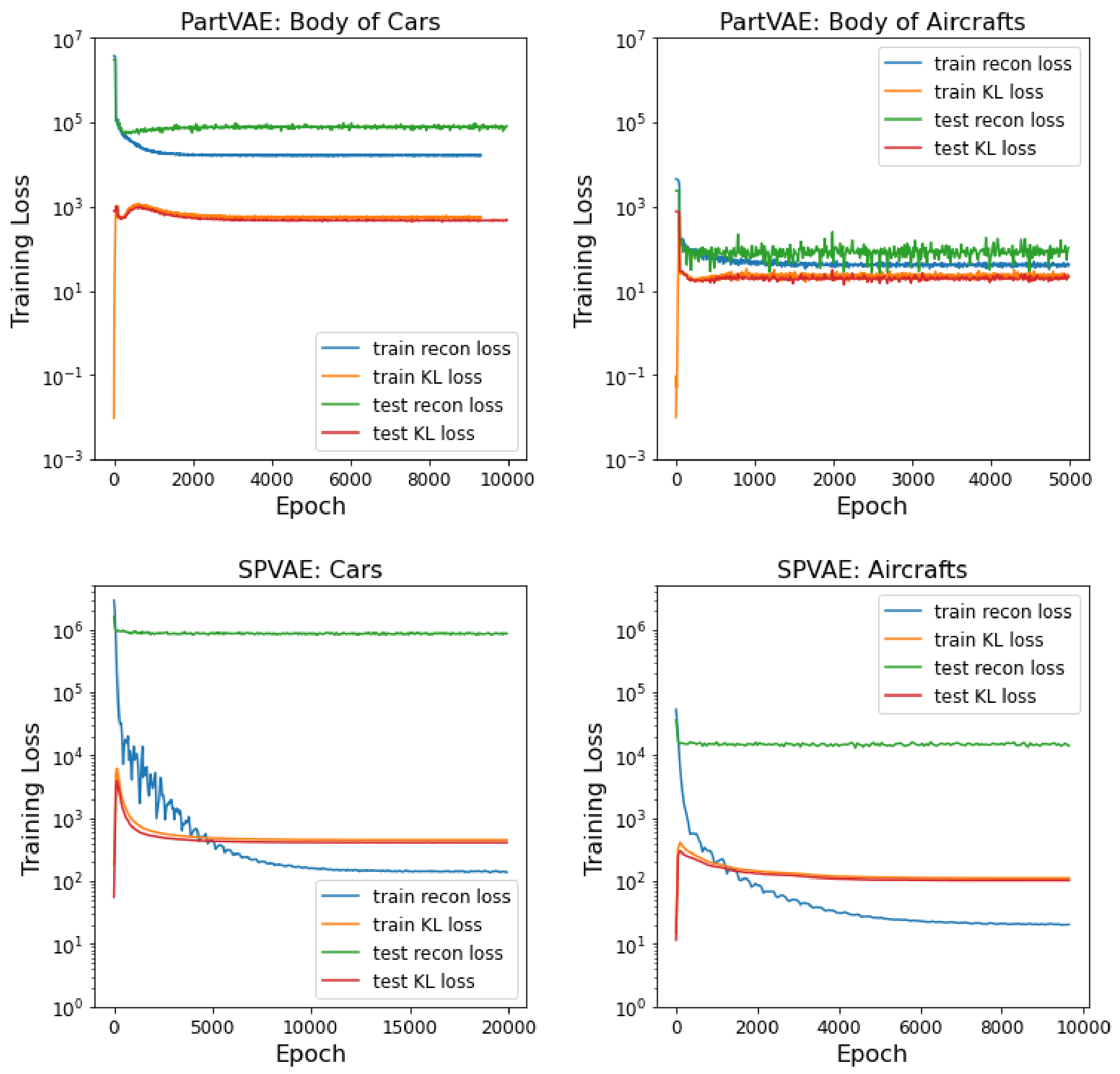


Figure C.1: The training loss values, including reconstruction loss and KL divergence loss, were recorded for both the car and aircraft models during the training process. However, for the purpose of showcasing the training of PartVAEs, we have chosen to highlight the loss values specifically for the training of the body components of the car and aircraft because similar trends were observed in the loss values for training other parts as well.

Works Cited

The effect of aerodynamic drag on fuel economy — Auto Research Center. http://www.arcindy.com/effect-of-aerodynamic-drag-on-fuel-economy.html. Accessed: 2022-11-28.

OpenAI Team. Introducing ChatGPT: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/, November 2022. Accessed: 2024-01-16.

Malak Abdullah, Alia Madain, and Yaser Jararweh. Chatgpt: Fundamentals, applications and social impacts. pages 1–8, 2022. doi: 10.1109/SNAMS58071.2022.10062688.

Gabriel Achour, Woong Je Sung, Olivia J Pinon-Fischer, and Dimitri N Mavris. Development of a conditional generative adversarial network for airfoil shape optimization. In *AIAA Scitech 2020 Forum*, page 2261, 2020.

David W Aha. *Lazy learning.* Springer Science & Business Media, 2013.

Faez Ahmed, Sharath Kumar Ramachandran, Mark D. Fuge, Samuel Todd Hunter, and Scarlett R. Miller. Interpreting idea maps: Pairwise comparisons reveal what makes ideas novel. *Journal of Mechanical Design*, 2018.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

Alberto Badías, Sarah Curtit, David González, Icíar Alfaro, Francisco Chinesta, and Elías Cueto. An Augmented Reality platform for interactive aerodynamic design and analysis. *International Journal for Numerical Methods in Engineering*, 2019. ISSN 0029-5981.

Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):1–13, 2015.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Pierre E Bézier. How renault uses numerical control for car body design and tooling. *Society of Automotive Engineer*, page 680010, 1968.

Andrew Brock, Theodore Lim, James Millar Ritchie, and Nick Weston. Context-aware content generation for virtual environments. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 50084, page V01BT02A045. American Society of Mechanical Engineers, 2016.

Alex Brown, Molly H Goldstein, John Clay, H Onan Demirel, Xingang Li, and Zhenghui Sha. A study on generative design reasoning and students' divergent and convergent thinking. *Journal of Mechanical Design*, 146(3), 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Francesco Buonamici, Monica Carfagni, Rocco Furferi, Lapo Governi, Alessandro Lapini, and Yary Volpe. Reverse engineering modeling methods and tools: a survey. *Computer-Aided Design and Applications*, 15(3):443–464, 2018.

Alexander Burnap, Ye Liu, Yanxin Pan, Honglak Lee, Richard Gonzalez, and Panos Y Papalambros. Estimating and exploring the product form design space using deep generative models. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 50107, page V02AT03A013. American Society of Mechanical Engineers, 2016.

Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2013.

Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361, 2020.

Amaresh Chakrabarti, Kristina Shea, Robert Stone, Jonathan Cagan, Matthew Campbell, Noe Vargas Hernandez, and Kristin L Wood. Computer-based design synthesis research: an overview. *Journal of Computing and Information Science in Engineering*, 11(2), 2011.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, and Hao Su. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Jiaxin Chen and Yi Fang. Deep cross-modality adaptation via semantics preserving adversarial learning for sketch-based 3d shape retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 605–620, 2018.

Jiaxin Chen, Jie Qin, Li Liu, Fan Zhu, Fumin Shen, Jin Xie, and Ling Shao. Deep sketch-shape hashing with segmented 3d stochastic viewing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 791–800, 2019.

Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Asian conference on computer vision*, pages 100–116. Springer, 2018.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Wei Chen and Faez Ahmed. Padgan: Learning to generate high-quality novel designs. *Journal of Mechanical Design*, 143(3), 2021.

Wei Chen and Mark Fuge. Synthesizing designs with interpart dependencies using hierarchical generative adversarial networks. *Journal of Mechanical Design*, 141(11):111403, 2019.

Wei Chen, Kevin Chiu, and Mark D Fuge. Airfoil design parameterization and optimization using bézier generative adversarial networks. *AIAA Journal*, 58 (11):4723–4735, 2020.

Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3d mesh segmentation. *Acm transactions on graphics (tog)*, 28(3):1–12, 2009.

Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

KR1442 Chowdhary and KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

John Zachary Clay, Xingang Li, Onan Demirel, Molly H Goldstein, Rundong Jiang, Charles Xie, Darya Zabelina, and Zhenghui Sha. Board 411: Thinking inversely in engineering design: Towards an operational definition of generative design thinking. In *2023 ASEE Annual Conference & Exposition*, 2023.

John Zachary Clay, Xingang Li, Rundong Jiang, Onan Demirel, Molly H Goldstein, Darya Zabelina, Charles Xie, and Zhenghui Sha. Engineering design thinking in the age of generative artificial intelligence. In *2024 ASEE Annual Conference & Exposition*, 2024.

James D Cunningham, Timothy W Simpson, and Conrad S Tucker. An investigation of surrogate models for efficient performance-based decoding of 3d point clouds. *Journal of Mechanical Design*, 141(12), 2019.

James D Cunningham, Dule Shu, Timothy W Simpson, and Conrad S Tucker. A sparsity preserving genetic algorithm for extracting diverse functional 3d designs from deep generative neural networks. *Design Science*, 6, 2020.

Guoxian Dai, Jin Xie, Fan Zhu, and Yi Fang. Deep correlated metric learning for sketch-based 3d shape retrieval. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Guoxian Dai, Jin Xie, and Yi Fang. Deep correlated holistic metric learning for sketch-based 3d shape retrieval. *IEEE Transactions on Image Processing*, 27(7):3374–3386, 2018.

Adrian V Dalca, John Guttag, and Mert R Sabuncu. Anatomical priors in convolutional networks for unsupervised biomedical segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9290–9299, 2018.

Tal Darom and Yosi Keller. Scale-invariant features for 3-d mesh models. *IEEE Transactions on Image Processing*, 21(5):2758–2769, 2012.

Jaime de Miguel Rodríguez, Maria Eugenia Villafañe, Luka Piškorec, and Fernando Sancho Caparrini. Generation of geometric interpolations of building types with deep variational autoencoders. *Design Science*, 6, 2020.

Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei A Efros, and Adrien Bousseau. 3d sketching using multi-view deep volumetric prediction. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–22, 2018.

Johanna Delanoy, David Coeurjolly, Jacques-Olivier Lachaud, and Adrien Bousseau. Combining voxel and normal predictions for multi-view 3d sketching. *Computers & Graphics*, 82:65–72, 2019.

H Onan Demirel, Molly H Goldstein, Xingang Li, and Zhenghui Sha. Human-centered generative design framework: An early design framework to support concept creation and evaluation. *International Journal of Human–Computer Interaction*, pages 1–12, 2023a.

H.O. Demirel, M.H. Goldstein, X. Li, and Z. Sha. Human-centered generative design framework: An early design framework to support concept creation and evaluation. *International Journal of Human-Computer Interaction*, 2023b. ISSN 15327590. doi: 10.1080/10447318.2023.2171489.

Matthew Dering, James Cunningham, Raj Desai, Michael A Yukish, Timothy W Simpson, and Conrad S Tucker. A physics-based virtual environment for enhancing the quality of deep generative designs. In *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2018.

Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, 2000 Proceedings 1*, pages 1–15. Springer, 2000.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Dong Du, Heming Zhu, Yinyu Nie, Xiaoguang Han, Shuguang Cui, Yizhou Yu, and Ligang Liu. Learning part generation and assembly for sketching man-made objects. In *Computer Graphics Forum*, volume 40, pages 222–233. Wiley Online Library, 2021.

Kristen M Edwards, Vaishnavi L Addala, and Faez Ahmed. Design form and function prediction from a single image. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85376, page V002T02A032. American Society of Mechanical Engineers, 2021.

Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on graphics (TOG)*, 31(4):1–10, 2012.

Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks generating "art" by learning about styles and deviating from style norms. In *8th International Conference on Computational Creativity, ICCC 2017*. Georgia Institute of Technology, 2017.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

Jonathan G. Fairman. Aerodynamics lift formula, 1996. URL `https://www.grc.nasa.gov/WWW/K-12/WindTunnel/Activities/lift_formula.html`.

Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 264–272, 2018.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28, 2015.

Stefano Filippi. Measuring the impact of chatgpt on fostering concept generation in innovative product design. *Electronics*, 12(16):3535, 2023.

Stanislav Frolov, Tobias Hinz, Federico Raue, Jörn Hees, and Andreas Dengel. Adversarial text-to-image synthesis: A review. *Neural Networks*, 144:187–209, 2021.

Mark Fuge. The frontiers in design representation (finder) summer school. `https://ideal.umd.edu/FinDeR/`, 2022. Accessed: 2022-10-01.

Kikuo Fujita, Kazuki Minowa, Yutaka Nomaguchi, Shintaro Yamasaki, and Kentaro Yaji. Design concept generation with variational deep embedding over comprehensive optimization. In *International Design Engineering Technical*

Conferences and Computers and Information in Engineering Conference, volume 85390, page V03BT03A038. American Society of Mechanical Engineers, 2021.

Kentaro Fukamizu, Masaaki Kondo, and Ryuichi Sakamoto. Generation high resolution 3d model from natural language by generative adversarial network. *arXiv preprint arXiv:1901.07165*, 2019.

Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34, 2021.

Lin Gao, Yu-Kun Lai, Jie Yang, Zhang Ling-Xiao, Shihong Xia, and Leif Kobbelt. Sparse data driven mesh deformation. *IEEE transactions on visualization and computer graphics*, 2019a.

Lin Gao, Jie Yang, Tong Wu, Yu Jie Yuan, Hongbo Fu, Yu Kun Lai, and Hao Zhang. SDM-NET: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics*, 38(6), nov 2019b. ISSN 15577368. doi: 10.1145/3355089.3356488.

Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019c.

Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016.

Daniele Giunchi, Stuart James, and Anthony Steed. 3d sketching for interactive model retrieval in virtual reality. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pages 1–12, 2018.

Daniele Giunchi, Alejandro Sztrajman, Stuart James, and Anthony Steed. Mixing modalities of 3d sketching and speech for interactive model retrieval in virtual reality. In *ACM International Conference on Interactive Media Experiences*, pages 144–155, 2021.

Molly H Goldstein, James Sommer, Natascha Trellinger Buswell, Xingang Li, Zhenghui Sha, and H Onan Demirel. Uncovering generative design rationale in the undergraduate classroom. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–6. IEEE, 2021.

Daniel P Gomari, Annalise Schweickart, Leandro Cerchietti, Elisabeth Paietta, Hugo Fernandez, Hassen Al-Amin, Karsten Suhre, and Jan Krumsiek. Variational autoencoders learn transferrable representations of metabolomics data. *Communications Biology*, 5(1):645, 2022.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.

Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia C Pont, Frédo Durand, and Adrien Bousseau. Opensketch: a richly-annotated dataset of product design sketches. *ACM Trans. Graph.*, 38(6):232–1, 2019.

Benoit Guillard, Edoardo Remelli, Pierre Yvernay, and Pascal Fua. Sketch2mesh: Reconstructing and editing 3d shapes from sketches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13023–13032, 2021.

Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.

Erkan Gunpinar, Salih Ertug Ovur, and Serkan Gunpinar. A user-centered side silhouette generation system for sedan cars based on shape templates. *Optimization and Engineering*, 20(3):683–723, 2019.

Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022.

Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michèle Sebag, et al. Analysis of the automl challenge series. *Automated Machine Learning*, 177, 2019.

David Ha and Douglas Eck. A neural representation of sketch drawings. In *International Conference on Learning Representations*, 2018.

Philip Haeusser, Alexander Mordvintsev, and Daniel Cremers. Learning by association–a versatile semi-supervised training method for neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 89–98, 2017.

Abid Haleem, Mohd Javaid, and Ravi Pratap Singh. An era of chatgpt as a significant futuristic support tool: A study on features, abilities, and challenges. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 2:100089, 10 2022. ISSN 27724859. doi: 10.1016/j.tbench.2023.100089.

Xiaoguang Han, Chang Gao, and Yizhou Yu. Deepsketch2face: A deep learning based sketching system for 3d face and caricature modeling. *ACM Transactions on graphics (TOG)*, 36(4):1–12, 2017.

Yi Han, Gaurav Nanda, and Mohsen Moghaddam. Attribute-sentiment-guided summarization of user opinions from online reviews. *Journal of Mechanical Design*, 145(4):041401, 2023.

Zhizhong Han, Mingyang Shang, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. Y2seq2seq: Cross-modal representation learning for 3d shape and text by joint reconstruction and prediction of view and word sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 126–133, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021a.

Yi He, Haoran Xie, Chao Zhang, Xi Yang, and Kazunori Miyata. Sketch-based normal map generation with geometric sampling. In *International Workshop on Advanced Imaging Technology (IWAIT) 2021*, volume 11766, pages 261–266. SPIE, 2021b.

Jörg Henseler, Christian M Ringle, and Rudolf R Sinkovics. The use of partial least squares path modeling in international marketing. In *New challenges to international marketing*. Emerald Group Publishing Limited, 2009.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Forrest Huang and John F Canny. Sketchforme: Composing sketched scenes from text descriptions for interactive applications. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, pages 209–220, 2019.

Forrest Huang, Eldon Schoop, David Ha, and John Canny. Scones: towards conversational authoring of sketches. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 313–323, 2020a.

Jingwei Huang, Yichao Zhou, and Leonidas Guibas. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*, 2020b.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges.* Springer Nature, 2019.

Kyung Hoon Hyun and Ji-Hyun Lee. Balancing homogeneity and heterogeneity in design exploration by synthesizing novel design alternatives based on genetic algorithm and strategic styling decision. *Advanced Engineering Informatics*, 38:113–128, 2018.

Tansin Jahan, Yanran Guan, and Oliver van Kaick. Semantics-guided latent space exploration for shape generation. In *Computer Graphics Forum*, volume 40, pages 115–126. Wiley Online Library, 2021.

Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022.

Daniel Jarrett and Mihaela van der Schaar. Target-embedding autoencoders for supervised representation learning. *arXiv preprint arXiv:2001.08345*, 2020.

Hrvoje Jasak, Aleksandar Jemcov, Zeljko Tukovic, et al. Openfoam: A c++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pages 1–20, 2007.

Subramaniam Jayanti, Yagnanarayanan Kalyanaraman, Natraj Iyer, and Karthik Ramani. Developing an engineering shape benchmark for cad models. *Computer-Aided Design*, 38(9):939–953, 2006.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11703–11712, 2021.

Pradeep Kumar Jayaraman, Joseph George Lambourne, Nishkrit Desai, Karl Willis, Aditya Sanghi, and Nigel JW Morris. Solidgen: An autoregressive model for direct b-rep synthesis. *Transactions on Machine Learning Research*, 2022.

Aobo Jin, Qiang Fu, and Zhigang Deng. Contour-based 3d modeling through joint embedding of shapes and contours. In *Symposium on Interactive 3D Graphics and Games*, pages 1–10, 2020.

R Kenny Jones, Rana Hanocka, and Daniel Ritchie. The neurally-guided shape parser: A monte carlo method for hierarchical labeling of over-segmented 3d shapes. *arXiv preprint arXiv:2106.12026*, 2021.

Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. The quick, draw!-ai experiment. *Mount View, CA, accessed Feb*, 17 (2018):4, 2016.

Glenn Judd and Peter Steenkiste. Providing contextual information to pervasive computing applications. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003).*, pages 133–142. IEEE, 2003.

Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet for joint object categorization and unsupervised pose estimation from multi-view images. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):269–283, 2019.

Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. Ucsg-net-unsupervised discovering of constructive solid geometry tree. *Advances in Neural Information Processing Systems*, 33:8776–8786, 2020.

Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *Advances in neural information processing systems*, 30, 2017.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103: 102274, 2023.

Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

Phillip Keane. Generative design comes to fusion 360. `https://www.engineering.com/story/generative-design-comes-to-fusion-360`, 2018. Accessed on 03/19/2023.

Khalid S Khan, Regina Kunz, Jos Kleijnen, and Gerd Antes. Five steps to conducting a systematic review. *Journal of the royal society of medicine*, 96 (3):118–121, 2003.

Hassan Khastavaneh and Hossein Ebrahimpour-Komleh. Representation learning techniques: An overview. In *The 7th International Conference on Contemporary Issues in Data Science*, pages 89–104. Springer, 2019.

Jin-Hwa Kim, Nikita Kitaev, Xinlei Chen, Marcus Rohrbach, Byoung-Tak Zhang, Yuandong Tian, Dhruv Batra, and Devi Parikh. Codraw: Collaborative drawing as a testbed for grounded goal-driven communication. *arXiv preprint arXiv:1712.05558*, 2017.

Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *European Conference on Computer Vision*, pages 175–191. Springer, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations*, 2014.

A Baki Kocaballi. Conversational ai-powered design: Chatgpt as designer, user, and product. *arXiv preprint arXiv:2302.07406*, 2023.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019.

Elisa Koolman, John Zachary Clay, Xingang Li, Rundong Jiang, Molly H Goldstein, Charles Xie, Onan Demirel, and Zhenghui Sha. A multi-case study of traditional, parametric, and generative design thinking of engineering students. In *The Eleventh International Conference On Design Computing and Cognition (DCC)*, 2024.

Sivam Krish. A practical generative design method. *Computer-Aided Design*, 43(1):88–100, 2011.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

Joseph George Lambourne, Karl Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. Reconstructing editable prismatic cad from rounded voxel models. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

Lei Le, Andrew Patterson, and Martha White. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. *Advances in neural information processing systems*, 31, 2018.

Hyunoh Lee, Jinwon Lee, Hyungki Kim, and Duhwan Mun. Dataset and method for deep learning-based reconstruction of 3d cad models containing machining features for mechanical parts. *Journal of Computational Design and Engineering*, 9(1):114–127, 2022.

Bo Li, Yijuan Lu, Afzal Godil, Tobias Schreck, Masaki Aono, Henry Johan, Jose M Saavedra, and Shoki Tashiro. *SHREC'13 track: large scale sketch-based 3D shape retrieval.* 2013.

Bo Li, Yijuan Lu, Afzal Godil, Tobias Schreck, Benjamin Bustos, Alfredo Ferreira, Takahiko Furuya, Manuel J Fonseca, Henry Johan, Takahiro Matsuda, et al. A comparison of methods for sketch-based 3d shape retrieval. *Computer Vision and Image Understanding*, 119:57–80, 2014a.

Bo Li, Yijuan Lu, Chunyuan Li, Afzal Godil, Tobias Schreck, Masaki Aono, Martin Burtscher, Hongbo Fu, Takahiko Furuya, Henry Johan, et al. Shrec'14 track: Extended large scale sketch-based 3d shape retrieval. In *Eurographics workshop on 3D object retrieval*, volume 2014, pages 121–130, 2014b.

Bo Li, Yijuan Lu, Fuqing Duan, Shuilong Dong, Yachun Fan, Lu Qian, Hamid Laga, Haisheng Li, Yuxiang Li, P Lui, et al. Shrec'16 track: 3d sketch-based 3d shape retrieval. 2016.

Bo Li, Juefei Yuan, Yuxiang Ye, Yijuan Lu, Chaoyang Zhang, and Qi Tian. 3d sketching for 3d object retrieval. *Multimedia Tools and Applications*, 80(6): 9569–9595, 2021a.

Bowen Li, Yue Yu, and Ying Li. Lbwgan: Label based shape synthesis from text with wgans. In *2020 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 47–52. IEEE, 2020a.

Changjian Li, Hao Pan, Yang Liu, Xin Tong, Alla Sheffer, and Wenping Wang. Robust flow-guided neural prediction for sketch-based freeform surface modeling. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018.

Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. Sketch2cad: Sequential cad modeling by sketching in context. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020b.

Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. Free2cad: Parsing freehand drawings into cad commands. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022a.

Haibing Li and Roland Lachmayer. Automated exploration of design solution space applying the generative design approach. In *Proceedings of the Design Society: International Conference on Engineering Design*, volume 1, pages 1085–1094. Cambridge University Press, 2019.

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. Secad-net: Self-supervised cad reconstruction by learning sketch-extrude operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16826, 2023a.

Xingang Li, H Onan Demirel, Molly H Goldstein, and Zhenghui Sha. Exploring generative design thinking for engineering design and design education. In *2021 ASEE Midwest Section Conference*, 2021b.

Xingang Li, Charles Xie, and Zhenghui Sha. Part-aware product design agent using deep generative network and local linear embedding. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, page 5250, 2021c.

Xingang Li, Ye Wang, and Zhenghui Sha. Deep learning of cross-modal tasks for conceptual design of engineered products: A review. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 86267, page V006T06A016. American Society of Mechanical Engineers, 2022b.

Xingang Li, Charles Xie, and Zhenghui Sha. A predictive and generative design approach for three-dimensional mesh shapes using target-embedding variational autoencoder. *Journal of Mechanical Design*, 144(11):114501, 2022c.

Xingang Li, Ye Wang, and Zhenghui Sha. Deep learning methods of cross-modal tasks for conceptual design of product shapes: A review. *Journal of Mechanical Design*, 145(4):041401, 2023b.

Xingang Li, Charles Xie, and Zhenghui Sha. Design representation for performance evaluation of 3d shapes in structure-aware generative design. *Design Science*, 9:e27, 2023c.

Shuang Liang, Weidong Dai, and Yichen Wei. Uncertainty learning for noise resistant sketch-based 3d shape retrieval. *IEEE Transactions on Image Processing*, 30:8632–8643, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Qi Liu and Shengjie Zhao. Guidance cleaning network for sketch-based 3d shape retrieval. In *Journal of Physics: Conference Series*, volume 1961, page 012072. IOP Publishing, 2021.

Zhengzhe Liu, Yi Wang, Xiaojuan Qi, and Chi-Wing Fu. Towards implicit text-guided 3d shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17896–17906, 2022.

Zhiyuan Liu, Yankai Lin, and Maosong Sun. *Cross-Modal Representation*, pages 285–317. Springer Singapore, Singapore, 2020a. ISBN 978-981-15-5573-2. doi: 10.1007/978-981-15-5573-2_9. URL `https://doi.org/10.1007/978-981-15-5573-2_9`.

Zhiyuan Liu, Yankai Lin, and Maosong Sun. *Cross-Modal Representation*, pages 285–317. Springer Singapore, Singapore, 2020b. ISBN 978-981-15-5573-2. doi: 10.1007/978-981-15-5573-2_9. URL `https://doi.org/10.1007/978-981-15-5573-2_9`.

Christian E Lopez, Scarlett R Miller, and Conrad S Tucker. Exploring biases between human and machine generated designs. *Journal of Mechanical Design*, 141, 2018. ISSN 1050-0472. doi: 10.1115/1.4041857.

Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. 3d shape reconstruction from sketches via multi-view convolutional networks. In *2017 International Conference on 3D Vision (3DV)*, pages 67–77. IEEE, 2017.

Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3d captioning with pretrained models. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhongjin Luo, Jie Zhou, Heming Zhu, Dong Du, Xiaoguang Han, and Hongbo Fu. Simpmodeling: Sketching implicit field to guide mesh modeling for 3d animalmorphic head design. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 854–863, 2021.

Kevin Ma, Daniele Grandi, Christopher McComb, and Kosa Goucher-Lambert. Conceptual design generation using large language models. *arXiv preprint arXiv:2306.01779*, 2023.

Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

Liane Makatura, Michael Foshey, Bohan Wang, Felix HähnLein, Pingchuan Ma, Bolei Deng, Megan Tjandrasuwita, Andrew Spielberg, Crystal Elaine Owens, Peter Yichen Chen, et al. How can large language models help humans in design and manufacturing? *arXiv preprint arXiv:2307.14377*, 2023.

Bharadwaj Manda, Shubham Dhayarkar, Sai Mitheran, VK Viekash, and Ramanathan Muthuganapathy. ‘cadsketchnet’-an annotated sketch dataset for 3d cad model retrieval with deep neural networks. *Computers & Graphics*, 99: 100–113, 2021.

Martti Mäntylä. *An introduction to solid modeling.* Computer Science Press, Inc., 1987.

Winter Mason and Siddharth Suri. Conducting behavioral research on amazon’s mechanical turk. *Behavior research methods*, 44(1):1–23, 2012.

Justin Matejka, Michael Glueck, Erin Bradner, Ali Hashemi, Tovi Grossman, and George Fitzmaurice. Dream lens: Exploration and visualization of large-scale generative design datasets. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

Christopher McComb, Jonathan Cagan, and Kenneth Kotovsky. Mining process heuristics from designer action data via hidden markov models. *Journal of Mechanical Design*, 139(11), 2017.

Christopher McComb, Binyang Song, Nicolas F Soria Zurita, Guanglu Zhang, Gary Stump, Corey Balon, Simon Miller, Michael Yukish, and Jonathan Cagan. Toward hybrid teams: A platform to understand human-computer collaboration during the design of complex engineered systems. 2020.

Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

Matthew McKnight. Generative design: What it is? how is it being used? why it’sa game changer. *KnE Engineering*, pages 176–181, 2017.

Alexandre Menezes and Bryan Lawson. How designers perceive sketches. *Design studies*, 27(5):571–585, 2006.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

Patrick Min, Michael Kazhdan, and Thomas Funkhouser. A comparison of text and shape matching for retrieval of online 3d models. In *International Conference on Theory and Practice of Digital Libraries*, pages 209–220. Springer, 2004.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019a.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *arXiv preprint arXiv:1908.00575*, 2019b.

Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019c.

Mohammadreza Mostajabi, Michael Maire, and Gregory Shakhnarovich. Regularizing deep networks by modeling and predicting label structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5629–5638, 2018.

James Mountstephens and Jason Teo. Progress and challenges in generative product design: a review of systems. *Computers*, 9(4):80, 2020.

NASA. Drag coefficient. URL `https://www.grc.nasa.gov`.

Charlie Nash and Christopher KI Williams. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. In *Computer Graphics Forum*, volume 36, pages 1–12. Wiley Online Library, 2017.

Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

Pablo Navarro, J Ignacio Orlando, Claudio Delrieux, and Emmanuel Iarussi. Sketchzooms: Deep multi-view descriptors for matching line drawings. In *Computer Graphics Forum*, volume 40, pages 410–423. Wiley Online Library, 2021.

Matt D Nelson, Brady L Goenner, and Bruce K Gale. Utilizing chatgpt to assist cad design for microfluidic devices. *Lab on a Chip*, 23(17):3778–3784, 2023.

Gen Nishida, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes, and Adrien Bousseau. Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016.

Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

Naoki Nozawa, Hubert PH Shum, Edmond SL Ho, and Shigeo Morishima. Single sketch image based 3d car shape reconstruction with deep learning and lazy learning. In *VISIGRAPP (1: GRAPP)*, pages 179–190, 2020.

Naoki Nozawa, Hubert PH Shum, Qi Feng, Edmond SL Ho, and Shigeo Morishima. 3d car shape reconstruction from a contour sketch using gan and lazy learning. *The Visual Computer*, pages 1–14, 2021.

Naoki Nozawa, Hubert PH Shum, Qi Feng, Edmond SL Ho, and Shigeo Morishima. 3d car shape reconstruction from a contour sketch using gan and lazy learning. *The Visual Computer*, 38(4):1317–1330, 2022.

Sangeun Oh, Yongsu Jung, Seongsin Kim, Ikjin Lee, and Namwoo Kang. Deep generative design: Integration of topology optimization and generative models. *Journal of Mechanical Design*, 141(11), 2019.

Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

OpenAI. Gpt-4v(ision) system card. 2023a. URL `https://api.semanticscholar.org/CorpusID:263218031`.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023b.

Kevin N Otto and K. Wood. *Product design: techniques in reverse engineering and new product development*. Prentice Hall, Upper Saddle River., 2001.

Gerhard Pahl, Wolfgang Beitz, Jörg Feldhusen, Karl-Heinrich Grote, et al. *Engineering design: a systematic approach*, volume 3. Springer, 1996.

Jitesh H Panchal, Mark Fuge, Ying Liu, Samy Missoum, and Conrad Tucker. Machine learning for engineering design. *Journal of Mechanical Design*, 141 (11), 2019.

Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34, 2021.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *arXiv preprint arXiv:1809.09761*, 2018.

Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3204–3215, 2021.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985, 2019.

Cyril Picard, Kristen M Edwards, Anna C Doris, Brandon Man, Giorgio Giannone, Md Ferdous Alam, and Faez Ahmed. From concept to manufacturing: Evaluating vision-language models for engineering design. *arXiv preprint arXiv:2311.12668*, 2023.

Michael J Pratt, Bill D Anderson, and Tony Ranger. Towards the standardized exchange of parameterized feature-based cad models. *Computer-Aided Design*, 37(12):1251–1265, 2005.

Anran Qi, Yi-Zhe Song, and Tao Xiang. Semantic embedding for sketch-based 3d shape retrieval. In *BMVC*, volume 3, pages 11–12, 2018.

Anran Qi, Yulia Gryaditskaya, Jifei Song, Yongxin Yang, Yonggang Qi, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Toward fine-grained sketch-based 3d shape retrieval. *IEEE Transactions on Image Processing*, 30:8595–8606, 2021.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

Feiwei Qin, Shi Qiu, Shuming Gao, and Jing Bai. 3d cad model retrieval based on sketch and unsupervised variational autoencoder. *Advanced Engineering Informatics*, 51:101427, 2022.

Nestor V Queipo, Raphael T Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P Kevin Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1):1–28, 2005.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

Molla Hafizur Rahman, Charles Xie, and Zhenghui Sha. A deep learning based approach to predict sequential design decisions. In *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2019.

Molla Hafizur Rahman, Shuhan Yuan, Charles Xie, and Zhenghui Sha. Predicting human design decisions with deep recurrent neural network combining static and dynamic data. *Design Science*, 6, 2020.

Molla Hafizur Rahman, Charles Xie, and Zhenghui Sha. Predicting sequential design decisions using the function-behavior-structure design process model and recurrent neural networks. *Journal of Mechanical Design*, 143(8):081706, 2021.

Partha Pratim Ray. Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, 2023.

Lyle Regenwetter, Brent Curry, and Faez Ahmed. Biked: A dataset and machine learning benchmarks for data-driven bicycle design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85383, page V03AT03A019. American Society of Mechanical Engineers, 2021.

Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. Deep generative models in engineering design: A review. *Journal of Mechanical Design*, 144(7): 071704, 2022.

Lyle Regenwetter, Colin Weaver, and Faez Ahmed. Framed: An automl approach for structural performance prediction of bicycle frames. *Computer-Aided Design*, 156:103446, 2023.

Tahira N Reid, Richard D Gonzalez, and Panos Y Papalambros. Quantification of perceived environmental friendliness for vehicle silhouette design. 2010.

Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoît Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable isosurface extraction. *Advances in Neural Information Processing Systems*, 33: 22468–22478, 2020.

Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, and Shuai Yi. Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, null:12458–12467, 2021. doi: 10.1109/ICCV48922.2021.01225.

Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing. In *European Conference on Computer Vision*, pages 482–498. Springer, 2022.

Aristides G Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys (CSUR)*, 12(4):437–464, 1980.

Dominick Rosato and Donald Rosato. 5 - computer-aided design. In Dominick Rosato and Donald Rosato, editors, *Plastics Engineered Product Design*, pages 344–380. Elsevier Science, Amsterdam, 2003. ISBN 978-1-85617-416-9. doi: https://doi.org/10.1016/B978-185617416-9/50006-5. URL `https://www.sciencedirect.com/science/article/pii/B9781856174169500065`.

Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malekshan. Clip-forge: Towards zero-shot text-to-shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18603–18613, 2022.

Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

Ryan Schmidt, Azam Khan, Gord Kurtenbach, and Karan Singh. On expert performance in 3d curve-drawing tasks. In *Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling*, pages 133–140, 2009.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *arXiv preprint arXiv:2007.08506*, 2020.

Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. In *International Conference on Learning Representations*, 2021.

Jami J Shah. Assessment of features technology. *Computer-aided design*, 23 (5):331–343, 1991.

Jami J Shah and Martti Mäntylä. *Parametric and feature-based CAD/CAM: concepts, techniques, and applications*. John Wiley & Sons, 1995.

Dimple A Shajahan, Vaibhav Nayel, and Ramanathan Muthuganapathy. Roof classification from 3-d lidar point clouds using multiview cnn with self-attention. *IEEE Geoscience and Remote Sensing Letters*, 17(8):1465–1469, 2019.

Gopal Sharma, Rishabh Goyal, Difan Liu, E. Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, null:5515–5523, 2017. doi: 10.1109/CVPR.2018.00578. URL `https://www.semanticscholar.org/paper/7d64392e4bcc4beed2e046676e2fabc253818335`.

Gopal Sharma, Rishabh Goyal, Difan Liu, E. Kalogerakis, and Subhransu Maji. Neural shape parsers for constructive solid geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:2628–2640, 2019. doi: 10.1109/TPAMI.2020.3044749. URL `https://www.semanticscholar.org/paper/01bb7873ed0d6571ac900879993e16e8cc400e85`.

Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3d point clouds. In *Computer Vision–ECCV 2020: 16th European*

Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16, pages 261–276. Springer, 2020.

Kristina Shea, Robert Aish, and Marina Gourtovaia. Towards integrated performance-driven generative design tools. *Automation in Construction*, 14 (2):253–264, 2005. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2004.07.002. URL `http://www.sciencedirect.com/science/article/pii/S0926580504000809`.

Zifan Shi, Sida Peng, Yinghao Xu, Andreas Geiger, Yiyi Liao, and Yujun Shen. Deep generative models on 3d representations: A survey. *arXiv preprint arXiv:2210.15663*, 2022.

Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Proceedings Shape Modeling Applications, 2004.*, pages 167–178. IEEE, 2004.

Dule Shu, James Cunningham, Gary Stump, Simon W Miller, Michael A Yukish, Timothy W Simpson, and Conrad S Tucker. 3d design using generative adversarial networks and physics-based validation. *Journal of Mechanical Design*, 142(7), 2020.

Vishal Singh and Ning Gu. Towards an integrated generative design framework. *Design studies*, 33(2):185–207, 2012.

Dmitriy Smirnov, Mikhail Bessmeltsev, and Justin Solomon. Deep sketch-based modeling of man-made shapes. 2019.

Dmitriy Smirnov, Mikhail Bessmeltsev, and Justin Solomon. Learning manifold patch-based representations of man-made shapes. In *International Conference on Learning Representations*, 2020.

András Sobester, Alexander Forrester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

B Song, NF Soria Zurita, G Zhang, G Stump, C Balon, SW Miller, M Yukish, J Cagan, and C McComb. Toward hybrid teams: A platform to understand human-computer collaboration during the design of complex engineered systems. In *Proceedings of the Design Society: DESIGN Conference*, volume 1, pages 1551–1560. Cambridge University Press, 2020.

Binyang Song, Scarlett Miller, and Faez Ahmed. Hey, ai! can you see what i see? multimodal transfer learning-based design metrics prediction for sketches with text descriptionss. In *ASME 2022 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2022.

Binyang Song, Scarlett Miller, and Faez Ahmed. Attention-enhanced multimodal learning for conceptual design evaluations. *Journal of Mechanical Design*, 145(4):041410, 2023a.

Binyang Song, Chenyang Yuan, Frank Permenter, Nikos Arechiga, and Faez Ahmed. Surrogate modeling of car drag coefficient with depth and normal renderings. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2023b.

Binyang Song, Rui Zhou, and Faez Ahmed. Multi-modal machine learning in engineering design: A review and future directions. *arXiv preprint arXiv:2302.10909*, 2023c.

Binil Starly, Atin Angrish, and Paul Cohen. Research directions in democratizing innovation through design automation, one-click manufacturing services and intelligent machines. *arXiv preprint arXiv:1909.10476*, 2019.

Evgeny Stemasov, Tobias Wagner, Jan Gugenheimer, and Enrico Rukzio. Shapefindar: Exploring in-situ spatial search for physical artifact retrieval using mixed reality. In *CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2022.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

Hanqi Su, Binyang Song, and Faez Ahmed. Multi-modal machine learning for vehicle rating predictions using image, text, and parametric data. *arXiv preprint arXiv:2305.15218*, 2023.

Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and Hongbo Fu. Interactive sketch-based normal map generation with deep neural networks. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–17, 2018.

Joshua D. Summers and Jami J. Shah. Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability. *Journal of Mechanical Design*, 132(2):021004, 01 2010. ISSN 1050-0472. doi: 10.1115/1.4000759. URL `https://doi.org/10.1115/1.4000759`.

Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

Karl T Ulrich. *Product design and development.* Tata McGraw-Hill Education, 2003.

Nobuyuki Umetani. Exploring generative 3d shapes using autoencoder networks. In *SIGGRAPH Asia 2017 technical briefs*, pages 1–4. 2017.

Nobuyuki Umetani and Bernd Bickel. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (TOG)*, 37 (4):89, 2018. ISSN 0730-0301.

Lewis Urquhart, Andrew Wodehouse, Brian Loudon, and Craig Fingland. The application of generative algorithms in human-centered product development. *Applied Sciences*, 12(7):3682, 2022.

Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11850–11860, 2022.

Sofia Valdez, Carolyn Seepersad, and Sandilya Kambampati. A framework for interactive structural design exploration. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 85390, page V03BT03A006. American Society of Mechanical Engineers, 2021.

Jamal van Kastel. *Visual Analytics for Generative Design Exploration: An interactive 3D data environment for a computational design system facilitating the performance-driven design process of a nearly Zero-Energy sports hall.* PhD thesis, Delft University of Technology, 2018.

Tamas Varady, Ralph R Martin, and Jordan Cox. Reverse engineering of geometric models—an introduction. *Computer-aided design*, 29(4):255–268, 1997.

Subeesh Vasu, Nicolas Talabot, Artem Lukoianov, Pierre Baque, Jonathan Donier, and Pascal Fua. Hybridsdf: Combining free form shapes and geometric primitives for effective shape manipulation. *arXiv preprint arXiv:2109.10767*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022a.

Fang Wang, Le Kang, and Yi Li. Sketch-based 3d shape retrieval using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1883, 2015.

Fei Wang, Yu Yang, Baoquan Zhao, Dazhi Jiang, Siwei Chen, and Jianqiang Sheng. Reconstructing 3d model from single-view sketch with deep neural network. *Wireless Communications and Mobile Computing*, 2021.

G. Gary Wang and S. Shan. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, 129(4): 370–380, 05 2006. ISSN 1050-0472. doi: 10.1115/1.2429697. URL `https://doi.org/10.1115/1.2429697`.

G Gary Wang and Songqing Shan. Review of metamodeling techniques in support of engineering design optimization. 2007.

Kehan Wang, Jia Zheng, and Zihan Zhou. Neural face identification in a 2d wireframe projection of a manifold object. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1622–1631, 2022b.

Lingjing Wang, Cheng Qian, Jifei Wang, and Yi Fang. Unsupervised learning of 3d model reconstruction from hand-drawn sketches. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1820–1828, 2018a.

Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. *Advances in neural information processing systems*, 33:20167–20178, 2020.

Xingzhi Wang, Nabil Anwer, Yun Dai, and Ang Liu. Chatgpt for design, manufacturing, and education. *Procedia CIRP*, 119:7–14, 2023.

Yikun Wang, Liang Chang, Yuhua Cheng, Lihua Jin, Zhengxin Cheng, Xiaoming Deng, and Fuqing Duan. Text2sketch: Learning face sketch from facial attribute text. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 669–673. IEEE, 2018b.

Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012.

Kevin J Weiler. *Topological structures for geometric modeling (Boundary representation, manifold, radial edge structure)*. Rensselaer Polytechnic Institute, 1986.

Robert E Wendrich. Multiple modalities, sensoriums, experiences in blended spaces with toolness and tools for conceptual design engineering. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 51739, page V01BT02A046. American Society of Mechanical Engineers, 2018.

Eamon Whalen and Caitlin Mueller. Toward reusable surrogate models: Graph-based transfer learning on trusses. *Journal of Mechanical Design*, 144(2), 2022.

Karl DD Willis, Pradeep Kumar Jayaraman, Joseph G Lambourne, Hang Chu, and Yewen Pu. Engineering sketch generation for computer-aided design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2105–2114, 2021a.

Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4):1–24, 2021b.

P-H Won. The comparison between visual thinking using computer and conventional media in the concept generation stages of design. *Automation in construction*, 10(3):319–325, 2001.

Fan Wu, Shih-Wen Hsiao, and Peng Lu. *Displays*. ISSN 0141-9382. doi: 10.1016/j.displa.2023.102623.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782, 2021.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10:1122–1136, 2023a. ISSN 2329-9274. doi: 10.1109/JAS.2023.123618. AIGC.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10 (5):1122–1136, 2023b.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015a.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015b.

Yu Xia, Shuangbu Wang, Lihua You, and Jianjun Zhang. Semantic similarity metric learning for sketch-based 3d shape retrieval. In *International Conference on Computational Science*, pages 59–69. Springer, 2021.

Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.

Nan Xiang, Ruibin Wang, Tao Jiang, Li Wang, Yanran Li, Xiaosong Yang, and Jianjun Zhang. Sketch-based modeling with a differentiable renderer. *Computer Animation and Virtual Worlds*, 31(4-5):e1939, 2020.

Jin Xie, Guoxian Dai, Fan Zhu, and Yi Fang. Learning barycentric representations of 3d shapes for sketch-based 3d shape retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5068–5076, 2017.

Peng Xu, Timothy M Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. Deep learning for free-hand sketch: A survey and a toolbox. *arXiv preprint arXiv:2001.02600*, 2020.

Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. In *International Conference on Machine Learning*, pages 24698–24724. PMLR, 2022.

Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl DD Willis, and Daniel Ritchie. Inferring cad modeling sequences using zone graphs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6062–6070, 2021.

Hairui Yang, Yu Tian, Caifei Yang, Zhihui Wang, Lei Wang, and Haojie Li. Sequential learning for sketch-based 3d model retrieval. *Multimedia Systems*, 28(3):761–778, 2022.

Kaizhi Yang, Jintao Lu, Siyu Hu, and Xuejin Chen. Deep 3d modeling of human bodies from freehand sketching. In *International Conference on Multimedia Modeling*, pages 36–48. Springer, 2021.

Maria C Yang. Concept generation and sketching: Correlations with design outcome. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 37017, pages 829–834, 2003.

Maria C Yang. Observations on concept generation and sketching in engineering design. *Research in Engineering Design*, 20(1):1–11, 2009.

Yongxin Yang and Timothy M Hospedales. Deep neural networks for sketch recognition. *arXiv preprint arXiv:1501.07873*, 1(2):3, 2015.

Yuezhi Yang and Hao Pan. Discovering design concepts for cad sketches. *ArXiv*, abs/2210.14451:null, 2022. doi: 10.48550/arXiv.2210.14451.

Yuxiang Ye, Bo Li, and Yijuan Lu. 3d sketch-based 3d model retrieval with convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2936–2941. IEEE, 2016.

Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. ISSN 0730-0301.

Xin Yi, Ekta Walia, and Paul Babyn. Generative adversarial network in medical imaging: A review. *Medical image analysis*, 58:101552, 2019.

Soyoung Yoo, Sunghee Lee, Seongsin Kim, Kwang Hyeon Hwang, Jong Ho Park, and Namwoo Kang. Integrating deep learning into cad/cae system: Case study on road wheel design automation. *arXiv preprint arXiv:2006.02138*, 2020.

Shaozu Yuan, Aijun Dai, Zhiling Yan, Zehua Guo, Ruixue Liu, and Meng Chen. Sketchbird: Learning to generate bird sketches from text. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2443–2452, 2021.

Yu-Jie Yuan, Yu-Kun Lai, Jie Yang, Qi Duan, Hongbo Fu, and Lin Gao. Mesh variational autoencoders with edge contraction pooling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 274–275, 2020.

Song-Hai Zhang, Yuan-Chen Guo, and Qing-Wen Gu. Sketch2model: Viewaware 3d modeling from single free-hand sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6012–6021, 2021.

Wei Zhang, Xiaogang Wang, and Xiaoou Tang. Coupled information-theoretic encoding for face photo-sketch recognition. In *CVPR 2011*, pages 513–520. IEEE, 2011.

Wentai Zhang, Zhangsihao Yang, Haoliang Jiang, Suyash Nigam, Soji Yamakawa, Tomotake Furuhata, Kenji Shimada, and Levent Burak Kara. 3d shape synthesis for conceptual design and optimization using variational autoencoders. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 59186, page V02AT03A017. American Society of Mechanical Engineers, 2019.

Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.

Fan Zhu, Jin Xie, and Yi Fang. Learning cross-domain neural networks for sketch-based 3d shape retrieval. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, and Christian Theobalt. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics (ToG)*, 33(4):1–12, 2014. ISSN 0730-0301.