CADQL

A Transformer-Based Architecture for CAD Model Generation and Validation

UPSLN

(501) 743-9680 | crw@upsln.com

Creating AI-driven CAD systems is a difficult challenge, given how complex geometric modeling and manufacturing rules can be. In this paper, we introduce CADQL (CAD Query Language), a cutting-edge transformer system that brings together geometric understanding and natural language processing to create and validate CAD models. Our approach uses special attention mechanisms, multi-modal encoders, and thorough validation checks to address the unique problems in CAD generation.

Background and Relevant work

Technical Foundations

CAD model generation requires understanding of various geometric primitives (points, curves, surfaces) and operations (extrusion, boolean operations, fillets). Traditional CAD systems use boundary representation (B-rep) and constructive solid geometry (CSG) to maintain geometric validity. Recent approaches have attempted to apply machine learning to this domain, but faced challenges in maintaining geometric consistency and manufacturing constraints.

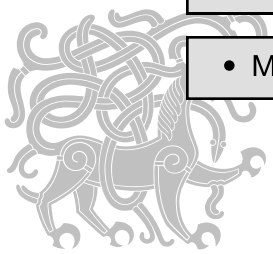
Current Approaches and Limitations

Current approaches to AI-driven CAD generation typically fall into three categories: rule-based systems that encode expert knowledge, graph-based approaches that represent CAD history as operation trees, and deep learning approaches that attempt to learn geometric relationships. CADQL builds upon these foundations while introducing novel attention mechanisms specifically designed for geometric understanding.

Novel Contributions

Our work advances the field through:

- Unified geometric attention mechanisms
- Multi-modal encoding strategy
- Robust validation pipeline
- Manufacturing-aware constraints



Introduction

1.1 Research Objectives

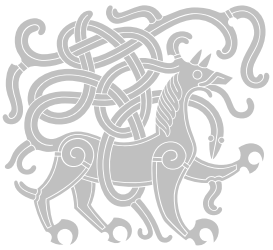
- Develop a transformer-based architecture for CAD model generation
- Create robust validation mechanisms for geometric and manufacturing constraints
- Bridge the gap between natural language and CAD operations
- Establish a framework for future AI-driven CAD development

1.2 Data representation

The creation of CAD models traditionally requires significant expertise in both design principles and specific CAD software platforms. While recent advances in artificial intelligence have shown promise in various domains, the application of deep learning to CAD model generation presents unique challenges due to the precise nature of geometric constraints and manufacturing requirements. CADQL aims to bridge natural language descriptions and precise CAD operations through a specialized transformer architecture. The system focuses on converting textual commands into workable CAD operations, promoting geometric and topological consistency, addressing manufacturing feasibility, and offering a robust validation pipeline.

1.3 Paper Structure

This paper is organized as follows: Section 2 provides technical background and related work, Section 3 details the CADQL architecture, Section 4 describes the implementation, Section 5 presents validation and testing approaches, and Section 6 discusses results and future directions.



CADQL Architecture

2.1 Core Components

CADQL's architecture consists of several specialized components:

1. **Unified Geometric Attention**

- Multi-scale geometric feature processing
- Flash attention optimization for efficient computation
- Adaptive sparsity patterns based on geometric complexity
- Built-in geometric bias support
- Block-based processing for large assemblies

2. **Multi-Modal Encoders**

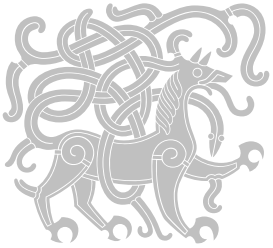
- CAD Encoder for processing geometric primitives
- Text Encoder for natural language commands
- Function Call Encoder for operation sequences
- Hybrid Positional Encoder combining multiple encoding strategies

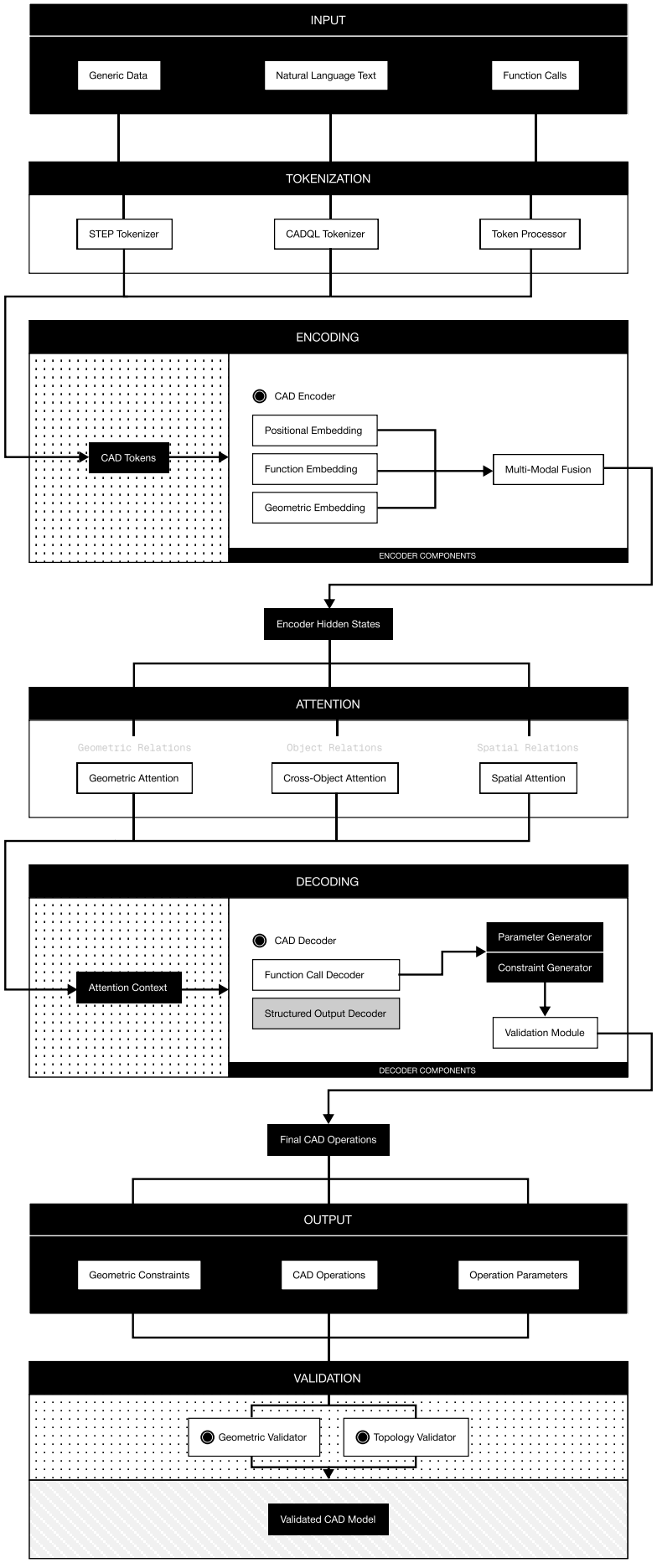
3. **Specialized Decoders**

- CAD Operation Decoder for generating geometric operations
- Function Call Decoder for parameter prediction
- Structured Output Decoder for hierarchical generation

4. **Validation Pipeline**

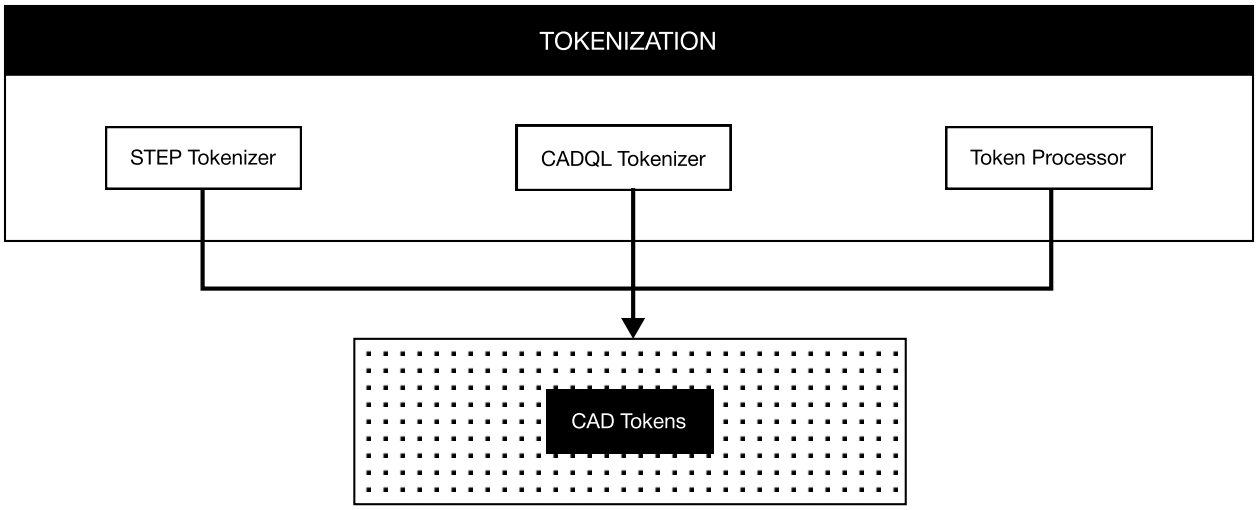
- Geometric validation for physical feasibility
- Topological validation for model consistency
- Manufacturing constraint validation




INPUT
Generic Data
Natural Language Text
Function Calls
TOKENIZATION
STEP Tokenizer
CADQL Tokenizer
Token Processor
ENCODING
CAD Tokens
CAD Encoder
Positional Embedding
Function Embedding
Geometric Embedding
Multi-Modal Fusion
ENCODER COMPONENTS
Encoder Hidden States
ATTENTION
Geometric Relations
Object Relations
Spatial Relations
Geometric Attention
Cross-Object Attention
Spatial Attention
DECODING
Attention Context
CAD Decoder
Function Call Decoder
Structured Output Decoder
Parameter Generator
Constraint Generator
Validation Module
DECODER COMPONENTS
Final CAD Operations
OUTPUT
Geometric Constraints
CAD Operations
Operation Parameters
VALIDATION
Geometric Validator
Topology Validator
Validated CAD Model

2.2 Tokenization System

An essential part of CADQL's design is the tokenization system, which serves as the bridge between humanreadable commands and the internal representations that our model manipulates. During tokenization, each instruction is decomposed into discrete, semantically rich tokens that align with the model's conceptual understanding of both linguistic and geometric constructs.



When a user supplies a description—such as requesting the creation of a cylindrical surface or the application of a fillet operation—the system first identifies relevant keywords, parameters, and contextual clues. These elements are then encapsulated within token objects, each annotated with the pertinent type, such as PRIMITIVE, OPERATION, or CURVE, while storing references to dimensions, constraints, or geometry data. This structured approach ensures that every meaningful piece of data advances through the model in a consistent, trackable form.

For instance, a PRIMITIVE token might capture the essential attributes of a box—its width, height, and depth —while an OPERATION token could govern transformations or boolean steps. Tokens categorized under CONSTRAINT or DIMENSION represent critical numerical and relational data, guaranteeing that subsequent geometry handling can integrate these parameters seamlessly. By nesting tokens within a hierarchical structure, CADQL maintains a clear lineage of dependencies, enabling the validation pipeline to reference both immediate and ancestral relationships across merged modeling contexts.

Internally, these tokens undergo various transformation passes, guided by the architecture's specialized attention layers. Each pass refines the raw token data into intermediate forms that increasingly approximate valid, manufacturable shapes. Consequently, the tokenization process functions as more than mere text parsing: it is the active interpretation of domain-specific knowledge, folding constraints, geometry references, and parametric data into tangible units of modeling logic.

2.3 Architecture Details

In this paper, we provide an examination of CADQL's architectural components, illustrating how geometric and textual information converge to form a coherent modeling workflow. By intertwining multimodal inputs, specialized attention layers, and validation mechanisms, CADQL demonstrates how natural language instructions can be systematically interpreted to produce valid CAD artifacts.

This section probes the foundational elements of CADQL, detailing how geometry-aware layers interact with embedded linguistic constructs to facilitate precise operation sequencing. Rather than structuring these processes as isolated lists, we focus on the underlying theories that inform each modeling stage and how these theories manifest through iterative data transformations.

2.4 Attention Mechanisms

CADQL implements several specialized attention mechanisms:

1. **Geometric Attention**

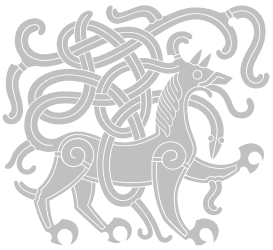
- Processes spatial relationships between entities
- Maintains geometric constraints
- Incorporates manufacturing knowledge

2. **Cross-Object Attention**

- Analyzes relationships between components
- Maintains assembly constraints
- Ensures proper feature ordering

3. **Function Use Attention**

- Learns patterns in CAD operation sequences
- Validates operation parameters
- Predicts operation success probability



Implementation

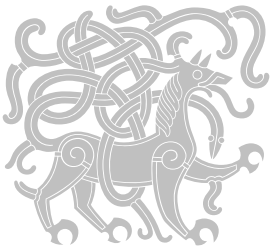
3.1 Core Components Implementation

In implementing CADQL, we embed geometry-aware features at each critical stage of the data pipeline. The core model employs advanced attention algorithms capable of identifying and preserving spatial relationships while concurrently resolving linguistic ambiguities. Through these attention modules, the tokenized commands and geometry references are aligned, ensuring that the processed output reflects both domain-specific constraints and the original design intent. Collectively, this synergy of textual and geometric data guides the system toward a well-defined series of commands, each validated against manufacturing considerations.

In code, the transition from token creation to final command generation involves iterative validations within each sub-module. The geometry layers check for topological soundness, the linguistic layers assess semantic compatibility, and specialized handlers verify compliance with manufacturing rules. By merging these validation steps in sequence, the CADQL pipeline forms a resilient transformation approach, capable of gracefully handling incomplete instructions or subtle conflicts among tokens.

3.2 Performance Enhancements

Flash attention implementation
Batch processing strategies
Memory management techniques
Parallel computation approaches



Geometric Processing & Validation

4.1 Geometric Processing Pipeline

The system includes robust geometric processing capabilities:

1. **Surface Processing**

- B-spline surface generation and modification
- Curvature analysis and optimization
- Continuity validation

2. **Curve Processing**

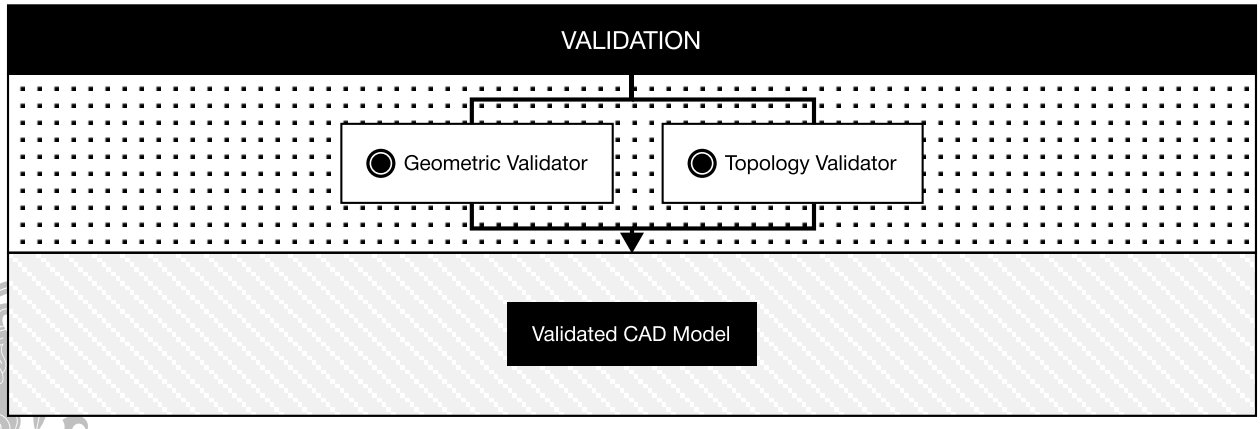
- B-spline curve generation
- Curve-surface intersection analysis
- Tangency and continuity management

3. **Topology Handling**

- Face-edge-vertex relationship management
- Boundary representation consistency
- Non-manifold detection and handling

4.2 Validation System

CADQL's validation suite uses various tests to evaluate model correctness and responsiveness to user instructions. The system also identifies geometry issues to ensure reliability for industrial applications.



Discussion

5.1 Capabilities and Limitations

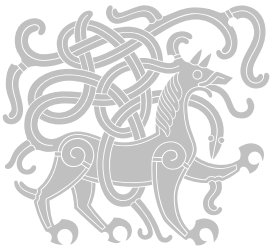
Despite these robust capabilities, certain theoretical factors continue to constrain the system. First, reliance on curated training data restricts adaptation to niche or unconventional geometry, as well as domains with limited examples. Second, real-time performance on very large assemblies is not yet assured and likely requires more advanced optimization layers. Finally, highly intricate freeform surfaces require specialized modules or hybrid approaches that combine conventional geometry engines with data-driven solutions.

5.2 Future Directions

We propose several hypothesis-driven directions for expanding CADQL:

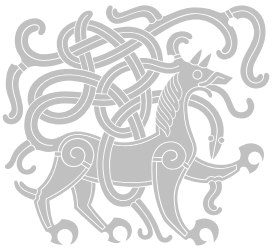
1. **Physics Integration:** Embedding real-time physics simulations will likely refine the system's ability to validate mechanical feasibility, improving design feedback.
2. **Manufacturing Process Planning:** By deepening integrations with rule-based knowledge sources, we anticipate a more comprehensive verification of manufacturability, including emerging production techniques.
3. **Collaborative Design Environments:** Multi-user interactions, possibly combined with version control or distributed systems, may yield flexible design workflows that evolve collaboratively.
4. **Material Property Handling:** Encoding materials and their constraints directly within geometry tokens could strengthen the synergy between design choices and final product performance.

Overall, these ongoing and future developments are expected to further validate CADQL's theoretical underpinnings, paving the way for a more robust, adaptive architecture that addresses the evolving demands of modern computer-aided design.



Conclusion

CADQL represents a significant step forward in AI-driven CAD generation, combining transformer architectures with geometric understanding and manufacturing constraints. The system's multi-modal approach and specialized attention mechanisms provide a robust foundation for future development in automated design and manufacturing.



References

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). "Attention Is All You Need"

Text2CAD: Generating Sequential CAD Models from Beginner-to-Expert Level Text Prompts Mohammad Sadil Khan∗†1,2,3, Sankalp Sinha∗1,2,3, Talha Uddin Sheikh1,2,3, Didier Stricker1,2,3, Sk Aziz Ali1,4, Muhammad Zeshan Afzal1,2,3 1DFKI 2RPTU Kaiserslautern-Landau 3MindGarage 4BITS Pilani, Hyderabad

OpenCASCADE Documentation (2024)

Appendix A: System Requirements

Software Dependencies

- PyTorch >= 1.12
- OpenCASCADE >= 7.6
- Transformers library
- NumPy, Rich (utilities)

Hardware Requirements

- CUDA-capable GPU recommended
- Minimum 16GB RAM
- 100GB storage for model files