

ABSTRACT

BHARADWAJ, AKSHAY GANESH. Driving Reasoning systems for Product Design and Flexible Robotic Manipulation using 3D Design-based Knowledge Graphs (Under the direction of Dr. Binil Starly and Dr. Ola Harrysson).

Product Design based Knowledge graphs (KG) aid the representation of product assemblies through heterogeneous relationships that link entities obtained from multiple structured and unstructured sources. This dissertation describes an approach to constructing a multi-relational and multi-hierarchical knowledge graph that extracts information contained within the 3D product model data to construct Assembly-Subassembly-Part and Shape Similarity relationships. This approach builds on a combination of utilizing 3D model meta-data and structuring the graph using the Assembly-Part hierarchy alongside 3D Shape-based Clustering. To demonstrate our approach, from a dataset consisting of 110,770 CAD models, 92,715 models were organized into 7,651 groups of varying sizes containing highly similar shapes, demonstrating the varied nature of design repositories, but inevitably also containing a significant number of repetitive and unique designs. Using the Product Design Knowledge Graph, we demonstrate the effectiveness of 3D shape retrieval using Approximate Nearest Neighbor search. We also illustrate the use of the KG for Design Reuse of co-occurring components, Rule-Based Inference for Assembly Similarity and Collaborative Filtering for Multi-Modal Search of manufacturing process conditions.

The application of robots in manufacturing environments has reached a high level of maturity, with advanced machine learning being increasingly used in conjunction with well-developed control systems. However, due to the specialized nature of applications such as robotic joining and assembly, repeatable but rigid programming-based control dominates industrial applications. Current applications in this domain driven by the latest trends in the Industry 4.0/ Smart Manufacturing paradigm requires robots to adapt to a variety of work

operations and environments, while maintaining accurate and efficient performance. However, there is a gap between the semantic understanding of the machines and the parts being manufactured. This work proposes a method to leverage autonomous object-level perception for flexible robotic manipulation and assembly operations by linking semantic information from CAD data to real-world scenes. By creating pixel-to-surface correspondences between the environment and the source CAD file, we demonstrate a method to create Scene Graphs based on 6D Pose estimates of the object and the hierarchical part data in combination with product manufacturing information (PMI). The application of this method is demonstrated through a sequential robotic manipulation and assembly planning task.

© Copyright 2023 by Akshay Ganesh Bharadwaj

All Rights Reserved

Driving Reasoning systems for Product Design and Flexible Robotic Manipulation using 3D
Design-based Knowledge Graphs

by
Akshay Ganesh Bharadwaj

A Dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Industrial Engineering

Raleigh, North Carolina
2023

APPROVED BY:

Dr. Binil Starly
Committee Co-Chair

Dr. Ola Harrysson
Committee Co-Chair

Dr. Yuan-Shin Lee

Dr. Emily Hector
External Member (Statistics)

Dr. Kemafor Anyanwu Ogan
Graduate School Representative

DEDICATION

To My Family;

With My Guru's blessings,

And the grace of the Almighty.

To My Dear Friends.

BIOGRAPHY

Akshay Ganesh Bharadwaj was born on 20th April 1994 and was schooled in Bengaluru, India. After completing his Bachelor of Engineering in Mechanical Engineering from Visvesvaraya Technological University (VTU) at Bengaluru, he moved to Raleigh to pursue graduate studies in Industrial Engineering at North Carolina State University. Post the completion of his MS in Industrial Engineering, he is currently pursuing a PhD in Industrial Engineering, with a focus on the utilization of data science for product design, robotics, and manufacturing automation.

ACKNOWLEDGEMENTS

I would like to convey my most sincere gratitude to my advisor, mentor and friend, Dr. Binil Starly. In over 5 years of working with him, I have come to admire not only his technical expertise and visionary ideas, but also his wonderful ability to engage people from multiple backgrounds, academic or otherwise. Besides this, however, I primarily attribute the results I have achieved under his guidance - and the skills I have learnt - to his patience and encouragement during the highs and lows of my research and professional life, particularly during the COVID-19 pandemic. I am thankful for the supportive and constructive environment that he created for students under his direction, feeding the student's zest for learning and innovation despite failures along the way. I hope to continue to benefit from his guidance, both personally and professionally.

I would like to thank Dr. Ola Harrysson for taking the time to be on my committee, as well as for involving me in the instruction of Product Design courses over the final 2 years of my PhD. It was an edifying experience, where Dr. Harrysson gave me significant freedom and the opportunities to pick up teaching, technical and professional skills. I would also like to thank committee member Dr. Yuan-Shin Lee, for his encouragement and guidance in my research as well as professional career. Many thanks to committee member Dr. Emily Hector, whose course provided valuable theoretical background that I continue to benefit from. Finally, many thanks to Dr. Kemafor Ogan for her feedback on the knowledge graph domain and for her valuable time.

I am grateful to my fellow researchers, past and present – Atin Angrish, Deepak Pahwa, Mahmud Hasan, Nabeel Mehdi, Aman Kumar, Connie Li, Pavel Koprov, Thomas Batchelder, and all others in the Manufacturing Lab - for their collaboration and openness to sharing their considerable knowledge and skills with me without hesitation. Many thanks to all the staff at NC State's Industrial Engineering Department and the Manufacturing Innovation Lab for their

kindness and assistance with a variety of technical and administrative issues, especially Jasmine Petway, Kendall Walker, Christina Pucci and Tim Coleman.

My sincere thanks to my friends in Bangalore, Raleigh, and elsewhere. Though they are too many to name, I am ever grateful for their unwavering backing and moral support through many years of friendship.

Lastly, my heartfelt gratitude to my parents Ganesh and Anuradha, and my brother Aniruddh, who have always gone above and beyond to unconditionally support me in my endeavors. I acknowledge the deep influence of my parents, grandparents, aunts and uncles, and my extended family; they have shaped my values and personality, and have provided me with valuable life lessons and direction in adversity. My work has been possible only with the support of my family and friends; I am thankful to them all.

LIST OF FIGURES

Figure 1: Construction Process & Schema of the Design Knowledge Graph	18
Figure 2: Graph Structure (a) For the Complete Graph: Histogram of Edge Weights: Only a small no. of edges are relevant; (b) After Weight Thresholding [0.85, 1]: Log-log Fit of the Dist. of Node Degree shows Exp.-Power-Law Fit	22
Figure 3: Construction of Optimal Retrieval Framework for Part Nodes	27
Figure 4: Performance of Community Detection Algorithms for Graphs based on (a) Modularity Density (b) Silhouette Score	29
Figure 5: Contextual Design Recommendation Process.....	36
Figure 6: (a) Distribution of Silhouette Score in the KG (b) Membership Func. for “highly_correlated” (Low, Medium and High).....	36
Figure 7: Recommended Co-occurring Parts with their corresponding Truth Values	37
Figure 8: Clusters of Assemblies based on individual part similarity	39
Figure 9: Recommendations for 6 parts: Recommendations for Spring and Triangular component are provided using Multimodal search using 3D Data + Text.....	41
Figure 10: Visual analysis of some clusters.....	45
Figure 11: Architecture for generating Surface Embeddings	53
Figure 12: Some example parts from the T-Less Dataset. Part 7 and 8 are assemblies of Part 6.....	56
Figure 13: Train and Validation Losses: YOLO V8 for T-Less dataset.....	57
Figure 14: Confusion Matrix for the T-Less Dataset.....	58
Figure 15: Performance on the T-Less Dataset: (a) Train and Validation set losses for the T-Less Dataset (b) Pose error (b) Prediction on the Test Dataset. Objects are superimposed on image in their predicted pose.	65
Figure 16: Qualitative Performance of the Pose Recognition Algorithm in cluttered scenes	66

Figure 17: Example of a scene graph for an Assembly Task 69

Figure 18: Final goal position for the Assembly Task..... 73

LIST OF TABLES

Table 1: Community Detection Algorithms: Complexity	24
Table 2: Comparing the two retrieval methods.....	32
Table 3: Class-wise mAP values for the T-Less Dataset.....	59
Table 4: Object Grasping Success Percentages	72
Table 5: Node Properties- Property Names and Classes/Text Descriptions.....	101
Table 6: Comparison of HNSW and KD-Tree for retrieval using Mean Average Precision (mAP) at $k = 1, 3, 5$: For each class	103

TABLE OF CONTENTS

LIST OF FIGURES	vi
LIST OF TABLES	viii
CHAPTER 1: INTRODUCTION	1
1.1. Objectives of the Dissertation	3
CHAPTER 2: THE CASE FOR KNOWLEDGE GRAPHS IN PRODUCT DESIGN.....	5
CHAPTER 3: RELATED WORK: KNOWLEDGE GRAPHS AND DATA-DRIVEN DESIGN	9
3.1. Knowledge Graphs and its Industrial Applications.....	9
3.2. Knowledge Extraction from 3D Data using Neural Networks	10
3.3. Data-Driven Design, Retrieval & Knowledge Graphs.....	10
CHAPTER 4: PRODUCT DESIGN KNOWLEDGE GRAPH: METHODOLOGY OF CONSTRUCTION, IMPLEMENTATION & STRUCTURING.....	15
4.1. Knowledge Extraction.....	15
4.2. Knowledge Graph Construction.....	16
4.3. Source Data & Annotation	18
4.4. Knowledge Extraction.....	20
4.5. Knowledge Graph Construction.....	20
4.5.1. Assembly Hierarchy Subgraph	20
4.5.2. Part Similarity Subgraph.....	21
4.6. Structuring the Knowledge Graph: Community Detection in the KG	22
4.7. Part Retrieval from the KG and Node Centrality	24
CHAPTER 5: EVALUATING COMMUNITY DETECTION AND RETRIEVAL.....	28
5.1. Community Detection	28
5.2. Top-k Retrieval Evaluation	30
5.3. Implementation Details	32
CHAPTER 6: RECOMMENDATION SYSTEMS USING THE PRODUCT DESIGN KNOWLEDGE GRAPH	33
6.1. Recommending Co-Occurring 3D Designs.....	34
6.2. Detecting Similarity of Assemblies.....	38
6.3. Design and Manufacturing Parameter Recommendations	40
6.4. Discussion	42
CHAPTER 7: IMAGE-BASED OBJECT POSE RECOGNITION FOR ROBOTIC MANIPULATION	46
7.1. Introduction.....	46
7.2. 6D Pose Recognition for Objects: Related Work	47

7.3. Scene Graphs and Applications: Related Work.....	50
7.4. Object Pose Estimation: Methodology	52
7.4.1. Pose estimation from 2D-to-3D Correspondences with Surface Embeddings	53
7.5. Training and Evaluation.....	55
7.5.1. Dataset.....	55
7.5.2. Object Detection with YOLO V8	56
7.5.3. Surface Embeddings & Pose Estimation	63
CHAPTER 8: SCENE GRAPHS FOR ROBOTIC MANIPULATION.....	67
8.1. Robot Programming and High-level Task Sequences	67
8.2. Scene Graph Generation: Methodology.....	68
8.3. Task Sequence Generation for Robotic Assembly	71
8.4. Discussion	73
CHAPTER 9: CONCLUSION	75
REFERENCES	77
APPENDICES	100
Appendix A: Available Part Annotations	101
Appendix B: Class-wise Mean Average Precision (mAP) Values	103
Appendix C: Examples of Retrieved Components for Standard Classes using HNSW+KNN Method (Bearings, Brackets, Pipes and Springs).....	105
C.1. Bearings: $mAP@5 = 1$ (Top row shows the query parts)	105
C.2. Brackets: $mAP@5 = 0.63$ (Top row shows the query parts)	106
C.3. Pipes: $mAP@5 = 0.58$ (Top row shows the query parts).....	106
C.4. Springs: $mAP@5 = 0.79$ (Top row shows the query parts)	107

CHAPTER 1: INTRODUCTION

Model-Based Enterprise (MBE) and its specific form, the Model-Based Definition (MBD), has produced an engineering workflow centered on 3D Computer-Aided Design (CAD) representation. In MBD, the CAD model ideally serves as the data source for all activities in the product lifecycle, with data from multiple workflow processes contained within Product Lifecycle Management (PLM) systems. However, there are challenges associated with using 3D data. Only an estimated 28% of surveyed engineers acknowledged that the majority of their designs are released with embedded product manufacturing information. Studies suggest that seamless methods for managing digital engineering information through the MBD paradigm can result in a potential \$9 billion in potential savings [1]. Due to the lacunae in efficient use of these existing frameworks (due to initial time & cost constraints), engineers often spend a significant amount of time in information retrieval tasks, much of which is repetitive in nature. In addition, much of downstream manufacturing data is semi-structured in nature, with a variety of formats and standards being used. For instance, systems modeling uses the Unified Modeling Language (UML) or the Systems Modeling Language (SysML), whereas process planning for manufacturing uses Computer-Aided manufacturing (CAM). To truly take advantage of this data, enabling interoperability through efficient links between a variety of domains, all using different data standards and transmission protocols, is a critical requirement. However, there are few uniform methods for linking data in the design and manufacturing domain. This means that while data formats have been standardized, the semantics of the data are often inaccessible.

A number of virtual assistants have been built for designers have utilized a compendium of formal geometric design rules organized in pseudo-code [2]; however, these are often organized manually at great effort. The vast corpus of existing data in organizations has the

potential to be utilized to change design from an iterative process towards providing pre-optimized design suggestions. Recent efforts in the manufacturing domain on context-aware design recommendations have proposed heterogeneous graphs as a method of linking design rules for proactive sequential recommendations [3]. This concept is part of the wider research in building recommendation systems takes the form of Knowledge Graphs, where extracted information is stored in the form of nodes and edges of a heterogeneous graph in a defined ontology, allowing for the use of reasoning systems to provide suggestions. Extracting data elements and the relationships between them goes hand-in-hand with Artificial Intelligence (AI) based techniques for information extraction. Specifically, the extensive research in natural Language Processing has been leveraged to create Knowledge Graphs in multiple domains to create successful recommendation systems, including manufacturing [4].

Recent research based on the Digital Thread concept has shown the advantage of data-associativity in PLM systems; they have been used to integrate traditional product manufacturing with newer technologies such as augmented reality [5], automation and robotics, and machine-readable multi-physics simulations [6] powered by AI. Vast improvements in computing and AI algorithms have been vital for organizations looking to streamline and integrate their design, manufacturing and supply chain systems [7]. This transformative potential of AI to improve design and manufacturing has long been recognized [8] [9], with robust cross-domain knowledge representations being crucial for leveraging this ability [10]. Knowledge graph representations are a powerful way of organizing this data for efficient reuse.

From a downstream perspective, CAD data plays an important role in manufacturing process planning and inspection. There is a dominant trend of manufacturing moving towards reduced production volumes and increasing variability, enabled by technologies such as Additive

Manufacturing. Simultaneously, the use of autonomy in manufacturing has increased, owing to dwindling specialized labor and the opportunity to lower costs. Cyber-Physical production systems based on Digital Twin technology is an important drivers of this increased autonomy in the age of Industry 4.0. They enable virtualization of machines and ease of data access to provide real-time actionable insights, efficient traceability and improved quality. With the increasing demands of flexible manufacturing environments, end-to-end digital integration must be leveraged to create self-perceptive machines to enable autonomous manufacturing task planning and implementation. The focus of work in this field so far has been on machine-to-machine communication, data models and predictive maintenance. However, there is a significant advantage to be gained by integrating the manufacturing components themselves into the cyber-physical mapping of the manufacturing system [11]. While this has been done through the use of RFID tags for parts, agents with self-perception must be able to recognize components in dynamically varying scenarios based on real-time sensor data.

1.1. Objectives of the Dissertation

Based on the motivations explained, the objectives of this work are described as follows:

Research Objective 1: Create a Knowledge Graph-based representation for 3D Product Design data to enable the recognition of patterns within designs for engineering decision making.

Data representations suitable for design-driven recommendation systems in the manufacturing domain often utilize textual approaches based on existing rules. However, these do not represent the full scope of design knowledge. By building a Knowledge graph based directly on 3D design data, this research aims to develop an integrated system for reasoning over designs, based on similarity of shape as well as design and textual context, for the purpose of providing data-driven engineering recommendations.

Research Objective 2: Develop a semantically rich scene graph-based method for robotic perception based on prior knowledge of product designs, for flexible and accurate robotic object manipulation.

Robotic systems are widely used in industrial applications. The Industry 4.0 paradigm enables flexible manufacturing through interoperability between machines and ease of data access. By linking Computer-Aided Designs with the real-time sensor data obtained during robotic manipulation tasks, this work aims to develop an autonomous reasoning system for manipulation, improving the capability for high-level sequential task planning.

CHAPTER 2: THE CASE FOR KNOWLEDGE GRAPHS IN PRODUCT DESIGN

We present a novel method for product design recommendation based on 3D shapes, using Knowledge Graphs predicated upon 3D CAD designs [12]. This chapter outlines our motivations and choice of architecture/representation to help solve the active field of design recommendations.

Product Design involves the optimization of multiple factors: performance, ease of manufacturing and assembly, cost and aesthetics. The parameters controlling these aspects of design are application-specific, either well defined or implicit, and requires domain expertise to apply towards good design practices. For example, Design for Additive Manufacturing requires concurrent evaluation of geometry, material, and mechanical properties during the design stage [13]. Present day commercial software has the capacity for optimized design suggestions through constrained design space exploration. They combine existing standard data representations such as STEP and statistical techniques such as design of experiments to provide optimal design parameters. However, these methods have scope for improvement, such as through automatic mining of pre-existing design data to provide more versatile recommendations. Considering the large amount of data generated by enterprises and those available in the open domain, it is important to create this linked data from existing semi-structured sources automatically as demonstrated recently in [14], [15] and [16].

At the enterprise level, data workflows need to include multiple sub-systems including electronic and software, each with their own ecosystems [17]. The tracking of product data thus spans multiple users, systems and domains. Clearly, data-driven methods to access and utilize multi-disciplinary engineering information requires organization and integration of data across multiple existing data-stores. Modern product management workflows contain the necessary data

and base level organization, but there is a need for methods to exploit this valuable data for decision making and potential reuse of design knowledge. The knowledge that is inherently present within the data-stores that contain vast amount of information on product data can only be extracted if the product data models are linked with attributes that define the model, and relationships that link any two digital 3D models.

Knowledge Graphs (KGs), which are graph representations of linked entities, can provide an efficient representation of connections among multiple structured and unstructured data sources. KG-based representations have been shown to mitigate issues in product development involving multi-disciplinary knowledge extraction and recommendation [18]. However, a large majority of work in this field focuses on textual KG representations. Generalized & open data sources for product design contain text classifications, but product specific descriptions in the open domain are scarce and non-standardized. This makes complex ontology-based methods that are commonly used to create KGs difficult to implement. Secondly, text alone does not completely cover all possible descriptions of diverse 3D data seen in designs, except in the case for standard components, which are readily available in existing databases. Finding relationships between non-standard components, be it parts, sub-assemblies, or even entire assemblies is more useful for design re-use in product development. One way to account for these issues is to use numerical representations of product shape as the core of the KG and creating a flexible representation of textual descriptions across the lifecycle built around these 3D model instances.

The following chapters demonstrate a hybrid method for construction of Knowledge Graphs (KG) from large Product Design repositories, implemented by leveraging existing product hierarchies and metadata from publicly available 3D data sources. In addition, 3D Shape Similarities obtained through Neural Networks are used in conjunction with Unsupervised Graph

Clustering to aid the construction of search indices within the design data repository, with the goal of linking data within the graph for efficient design search and recommendation. The resulting Product Design Knowledge Graph is released as a Linked Open Knowledge base for product design and manufacturing [19], [20]. Finally, use of the KG to provide efficient design recommendations from prior data is shown, along with its potential to use KGs for complex decision making. The use of 3D CAD models as the focus of the KG mirrors enterprise-level data representations; we believe that extensions of this framework with rich multi-disciplinary data available at the organizational level can enable better engineering driven multi-domain decision making including automation applications.

Automation has been applied to industrial assembly tasks with great success over the last few decades, leading to massive increase in productivity, revenue and safety. Due to the complexity of assembly tasks, the predominant method for automation in industry today is programmed robotic control, with pre-defined positions and goals for automated tasks. The lack of adaptability of fixed control to dynamic scenarios in industrial environments has led to the development of advanced control techniques based on machine learning techniques. These methods are often based on supervised learning techniques, which require significant human involvement for acquisition and annotation of data. Modern day applications require more flexible methods for automation that are safe, efficient and reduce human involvement in low-value tasks. Reinforcement Learning (RL) has shown great promise in integrating data sources used in current-day machine learning algorithms for training motor skills in automation systems with low effort. However, the sample inefficient nature of RL algorithms means that training these algorithms is often a complex process, with robust policies not guaranteed to be found. In this work, we propose a three-fold method to address these inefficiencies by (i) Training the RL

policy in simulation, (ii) Using prior knowledge of CAD data for part localization and to infer associations between parts to decipher assembly structure, and (iii) Transferring this trained policy to real-life applications. Using a combination of vision-based data and proprioceptive information from the robots trained in simulated environments, assembly tasks in industrial settings can be achieved at low cost with high potential for generalizability. The advantages of relatively low-cost simulated data and prior knowledge of geometric information can create adaptable and robust industrial assembly tasks, while minimizing human involvement in unsafe environments.

CHAPTER 3: RELATED WORK: KNOWLEDGE GRAPHS AND DATA-DRIVEN DESIGN

3.1. Knowledge Graphs and its Industrial Applications

Well-known Knowledge Graphs (KGs) for real world facts derived from linked-open data on the web have gained popularity for question-answering systems over the last decade, with notable examples being YAGO [21], Wikidata [22] and Freebase [23]. They are collections of ‘triples’ of the form (entity, relation, entity) represented by a directed graph structure, with nodes of multiple types representing entities and edges representing multiple relation types between them. There has been a growth in the development of specialized ontologies for KGs as part of a trend in the natural sciences for domain-specific KG applications. The biomedical field has seen significant work in knowledge graph construction techniques, with work by [24] and [25] being two recent examples of applying existing textual domain data to create a knowledge graph framework. Materials science has also seen advanced use of KG methods to connect material property relationships and relevant literature sources, connecting implicit and explicit knowledge with decision making [26].

It is observed that the overwhelming majority of knowledge graphs are created using Natural Language Processing (NLP) of existing textual data, for example in work by Zhou et al [4] and Myklebust et al [27]. This can also be seen in the exhaustive review by Li et al [18], where neural network-NLP approaches are used extensively to formalize knowledge for industrial products and services. This approach may not be able to fully represent data in the product design domains, leaving out relations that are not easily extracted in textual form. The large amount of multi-domain design data in enterprises means that automated knowledge extraction and codification is crucial for further applications.

3.2. Knowledge Extraction from 3D Data using Neural Networks

Over the last decade, the application of Neural Networks (NNs) for classification and retrieval applications has resulted in massive improvements in 3D shape representations. The use of image, point cloud and voxel data to characterize 3D shapes has seen great success. 3D Shapenets [28], VoxNet [29] and Pointnet [30] are some well-established techniques for 3D shape recognition using point cloud and voxel data. The maturation of computer vision for image recognition has meant that these methods have been successfully applied in the context of 3D shapes as well; the highly influential work in this field by Su et al [31] that uses multiple views of 3D shapes is a notable example of the view-based 3D descriptors. This was followed by work such as RotationNet [32] and our previous work on CAD model classification using multi-view images [33]. Easy availability of image data means that NNs pre-trained on well-established image data such as ImageNet [34] can be used to obtain accurate models trained on relatively small datasets much quicker. This is a critical advantage when automated knowledge extraction is applied at an industrial scale. In recent years, work on applying these techniques to CAD data for more specific feature recognition (as opposed to representing the whole part) and shape simplification has gained prominence as seen in work by Zhang et al [35] and Song et al [36]. These advantages of these techniques for automated knowledge acquisition from 3D shapes are crucial to knowledge representation of product designs. They provide a way to infer relations between currently unconnected data in the design and manufacturing domains and provide more opportunities to improve data-driven design.

3.3. Data-Driven Design, Retrieval & Knowledge Graphs

In contemporary industrial applications, standardization of components has been widely used to enable large-scale production while significantly reducing design effort. For example, in

the aerospace industry, there is an interest in reusing brackets from existing data for newer applications; in the work by Clark et al [37], the use of hierarchical clustering based on bracket features and dimensions is demonstrated to find representative parts from their entire dataset. Due to the restrictive nature of enterprise data, there has also been great interest in knowledge structuring outside industry which is crucial to drive automated knowledge acquisition for design [9]. Motivated by the same restrictions of enterprise data, there have been multiple open design repositories of varying size and functional descriptions proposed and used in [38]–[41] and the data used in this paper [42] to drive data science applications in product design.

The use of graphs and domain-specific ontologies for recording product data over the design and production cycle has gained popularity due to its potential for multi-dimensional decision making [43] [44] [45]. These include process modeling through MBE [46], automated assembly sequence generation [47], design decision support [48], capturing intent and data reuse for physics-based simulation [49] [50]. These ontologies can be leveraged to create Knowledge Graphs (KGs), which have great potential for performing search and inference on real-world data.

Text-based knowledge graphs, with data obtained from existing design documents have been a focus of research over the last few years. The use of domain specific text-driven semantic knowledge graphs has been explored for representing explicit and tacit design knowledge [51] and efficient process planning using a Process Knowledge Graph obtained from CAD/CAM systems [52]. Helping designers retrieve assemblies based on functional semantics was shown in the work by [53]. Multi-criteria topological similarity was used by [54] to enable similarity assessment, decomposing assemblies into a graph with common parts classified into node types and the kinematic assembly operation sequence denoting the relation between them. An

approach for applying graph-based knowledge reuse in product development was proposed by [55], with the potential to vastly increase knowledge reuse in the manufacturing industry. The problem of connecting multiple sub-systems containing associated data provides one major barrier to the realization of this goal; [56] provides a solution to link and trace data throughout the product lifecycle by creating digital threads across multiple system interfaces. Recently, a generalized solution to track products and their manufacturing dependencies using graph databases was shown by Martinez-Gil et al [57], which showed that product data in graph form outperformed relational databases. The advantages of graph-based approach for the product domain are clearly seen from the literature.

In conjunction with textual data, several Knowledge Graph implementations in the product domain include geometry as the basis for construction. These KGs use 3D data to inform further decision making in design and manufacturing. As seen in the work by Ferrero et al [41], these approaches have the ability to drive applications such as functional classification of components within assemblies, using data obtained from CAD models. CAD model KG based on decomposition of 3D point cloud to primitive shapes was applied for shape-based retrieval by [58], utilizing unsupervised methods to group 3D shape primitives. KG driven assembly planning was demonstrated by Zhou et al [59], where information from the CAD model and the assembly process document were used to generate assembly sequences. The authors demonstrate that the use of 3D shapes is key to be able to account for potential collisions, correct orientation, and associated manufacturing parameters. From the literature, systems for design intelligence clearly need to combine multiple modalities: textual data, implicit/explicit knowledge, design intent, simulation, manufacturing capabilities, and user requirements in order to enable true design intelligence.

An extensive search of existing KG literature in the design/manufacturing space by [60] shows that most knowledge graphs are created using a top-down approach, involving the creation of an ontology which is then populated with data to create the KG. However, an ontology-based approach is time consuming to create and update; automating the acquisition, processing and use of knowledge has high value when dealing with large amounts of diverse domain data [61]. Additionally, many manufacturing KGs are modeled as Resource Description Framework (RDF) graphs, where nodes and edges do not contain attributes. In contrast, the Labeled Property Graph (LPG) graph model can carry properties as key-value pairs and has the potential to outperform RDF for complex graph queries [62], while creating a more compact representation of properties. LPG and other ‘hyper-relational’ graph structures such as RDF* show possibilities for efficient representation of multiple relationship details and weights. This form of data representation is well-suited to exploratory approaches and sparse data.

Currently, many KGs in the product domain focus on one aspect of the product cycle such as design, simulation, manufacturing or assembly. Many of these KGs are constructed using concept mapping & classification of previous textual data as seen in [63]. However, the specialized nature of each of these domain leads to siloed data sources with little or no interaction across the product lifecycle that inherently will contain diverse textual descriptions. Work by Hao et al [64] answers this problem by the use of generalizable and domain-independent Decision Support Problem Technique (DSPT) to generate multiple KGs for domain-decision making processes, demonstrated on a supply-chain design domain. This method provides a common knowledge template for designers, but interrelated sectors continue to be isolated from another, which may lead to difficulty in multi-dimensional decision making.

Considering these limitations of text-based KGs, this paper describes a method using 3D shape & relevant metadata as the center of the product description and decision-making process. This can pave the way for proactive recommendations early in the product cycle, by using associations from prior data not explicitly linked to design. Importantly, for current applications involving large amounts of data, inter-connection between data must be created automatically and must be easily searchable. A method to achieve this is shown using numerical representations or ‘embeddings’ of design similarity to create a graph. Subsequently, unsupervised methods are applied to group parts into communities, and to account for the large variety of products that do not fit under standard part categories. At the enterprise level, unified product-based KGs thus created can be used for company-level data tracking, analysis and recommendation from diverse sources using data-driven reasoning. Textual data associated with design decisions using domain-specific ontologies can be linked to design documents, which can then generate automated suggestions.

CHAPTER 4: PRODUCT DESIGN KNOWLEDGE GRAPH: METHODOLOGY OF CONSTRUCTION, IMPLEMENTATION & STRUCTURING

Knowledge Graph generation and exploitation is generally divided into Knowledge Extraction and Construction, followed by Deduction of Patterns and Usage [18]. This section describes the methodology of Data Extraction and KG Construction. Two possible approaches exist for KG construction: modeling an ontology and then populating it with data (Top-Down), or by automatically inducing structure from the data (Bottom-Up) [60], [65]. Here, we propose a hybrid approach by extracting directly available textual data for Top-Down relations, as well as the use of latent shape data representations for Bottom-Up relationship linking.

4.1. Knowledge Extraction

Existing product design stores in the form of industry standard STEP files and their extensions are a primary source of data and are used for day-to-day design decision making. Individual part design data is represented in multiple formats with varying amounts of information, and the similarity between them can be computed based on their associated text data. The existing technical text data and hierarchical information from design files is the primary data extracted from the CAD files. However, these are not always complete; obtaining comprehensive textual descriptions and associated data of parts is a laborious process.

There has been a large body of work devoted to Neural Network Classification and Retrieval methods of 3D components based on shape ([28], [29]) as well as image-based methods, which have had success as seen in [31], [32], [66]. These methods can generate feature vectors or embeddings as global representations of the parts. Our method leverages these representations for further search and analysis. Thus, knowledge extraction is completed using a

combination of available textual data and vector representations of shape data using Neural Networks.

4.2. Knowledge Graph Construction

By restricting the base node data to product designs, we present a knowledge graph construction method utilizing three types of nodes: (i) Assemblies, (ii) Subassemblies, and (iii) Parts, in descending order of hierarchy. The edges or relations between these nodes of the Knowledge Graph are created as follows:

Existing assembly hierarchies obtained from CAD files are used for Assembly-Subassembly-Part relations, similar to the approach in [59]. This is termed as the *Assembly Hierarchy Subgraph*, and it contains 3 types of nodes: *part*, *subassembly*, and *assembly*, which are linked by the relation type *subset_of* in hierarchical order.

The second type of relations between individual parts are created using global similarity of feature-vectors representing each individual part. The subgraph containing nodes of type *part* and similarities between these parts given by the (weighted) relation *similar_to* is termed the *Part Similarity Subgraph*. These global similarity computations are performed using the respective vector embeddings of the *part* nodes.

Considering the intended application of this method for large design databases, there are an enormous number of part-to-part similarity relations that could potentially be created, with a wide range of edge weights which will affect the goal of efficient search within the KG. Previous work on complex networks has shown that making networks sparse by removing insignificant relations has the potential for huge improvements in speed for graph clustering or community detection, while maintaining the original nature of the graph [60]. This can be achieved for weighted graphs at the global scale using Edge-Weight Thresholding; work by Yan et al [67]

demonstrates the robust nature of network structure when weight thresholding is applied. In particular, weight-thresholded graphs with a Scale-Free degree distribution have been shown to maintain the overall topology of the complete graph. Inspired by similar approaches applied to brain networks in neuroscience as seen in [58] and [59], the construction method is fine-tuned by constructing a sparse yet meaningful graph through Global Weight Thresholding as described in the following paragraph.

Let the *Part Similarity Subgraph*, G , be described by its symmetric weighted adjacency matrix W , where the elements W_{ab} denote the weight of the edge connecting nodes a and b . Given a limiting value " α ", the weight thresholded graph G' then has an adjacency matrix W' , such that all weights below this value are discarded i.e., if $W_{ab} < \alpha$ then $W'_{ab} = 0$. This threshold α is chosen such that the resulting graph G' is a *scale-free network* with a heavy-tailed degree distribution. A scale-free network is one where the fraction of nodes $P(x)$ with degree x is close to a power law, i.e., $P(x) \propto x^{-\gamma}$. This phenomenon indicates that there is a small but significant proportion of nodes with a much higher degree than most other nodes. This process of simplifying the graph ensures a balance between representativeness and compactness at the construction level, which speeds up search and detection of structure within the Knowledge Graph.

Nodes and relations can be extended where data is available, for instance, adding Finite Element Analysis (FEA) or Kinematic Simulation data to create localized subgraphs to potentially predict the behavior of new data. This method provides a starting framework for such additions; the initial relationships are created considering the KG's objective of enabling optimal search and inference using a hybrid bottom-up construction method. Graph data is represented by the Labeled Property Graph model, which allows both nodes and edges of the graph to be

described by key-value pairs which can contain further metadata. This is advantageous in the case of a design graph, enabling a flexible schema representation for CAD parts and assemblies, which have different relation types and sets of properties. Fig.1 summarizes the process of constructing the Product Design Knowledge Graph.

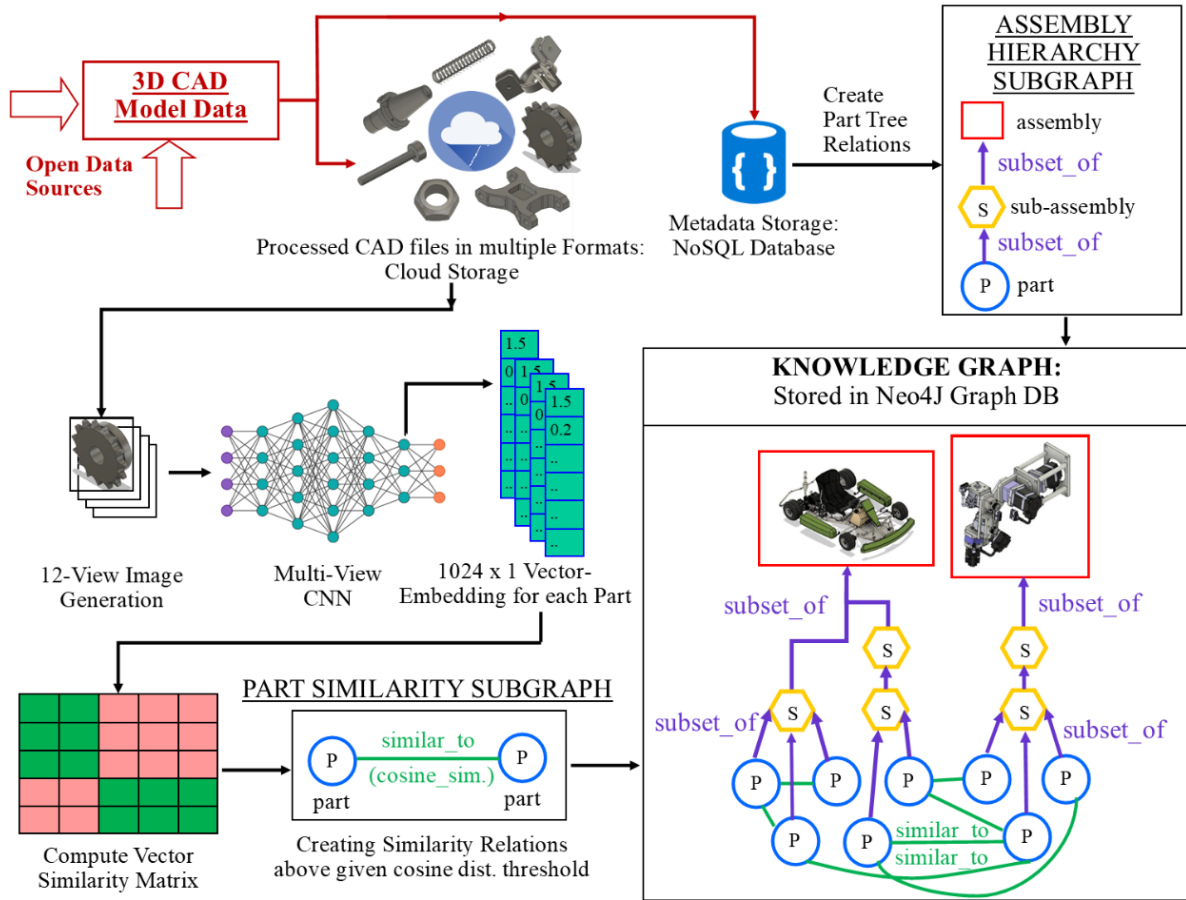


Figure 1: Construction Process & Schema of the Design Knowledge Graph

4.3. Source Data & Annotation

The KG construction is implemented with data from the FabWave repository described by Bharadwaj et al [42]. Created as a cyber-infrastructure to automate collection of CAD and related

manufacturing data from academic and open sources, it consists of over 3000 assemblies and over 110,000 parts. This data is divided the following subsets:

- (i) FabWave Categorized: These 4000+ 3D parts were obtained from student-generated data using design software add-ins and classified into 45 categories, based on both form and process [70].
- (ii) FW10C: A 10-class subset from the above dataset used for training classification algorithms.
- (iii) FabWave Open: Built from openly available sources of 3D data such as GrabCAD and Autodesk Fusion Gallery. Consists of 3000+ assemblies, processed into their individual parts (numbering over 110,000) in multiple data formats (STEP, STL, F3D etc.) along with component tree structure. Available basic metadata about the parts are recorded during file processing and from their original sources. [20]

The assemblies in the FabWave dataset were processed into individual parts and organized in the knowledge graph in accordance with the hierarchical structure of CAD assembly files. This dataset contains basic assembly categories and size meta-data; however, the individual parts lack detailed technical labels. The size of the dataset and variety of parts in the FabWave Open dataset means that a single label classification is a complex exercise and does not necessarily add value to the data. To account for this, a multi-description scheme was devised, including feature labels, free text descriptions, process/materials etc. These description categories are covered in Appendix A. With the help of human annotators with knowledge in design and manufacturing, 3118 of these parts were provided with descriptions based on this schema, which was combined with shape data to build the knowledge graph. Annotations are represented as node properties in the KG.

4.4. Knowledge Extraction

As described in section 3.1, global part representations were generated using neural network methods. Here, we leverage our previous work on a 12-image Multi-View Convolutional Neural Network approach, specifically modified for CAD data by the addition of dimension measurements for training [33]. Originally used for global classification of 3D shapes, this architecture uses 12 images from multiple views as input to ResNet CNNs terminating in 4096-dimension vectors, which are then aggregated and down sampled to a 10-class classifier along with a 1024-dimension vector. The concept of Transfer Learning enables a fast-paced training process for this neural network, by extending previously trained models on the ImageNet image classification database to 3D Shape classification and global feature extraction. Based on the pre-trained 10-label neural network classifier, transferrable features/embeddings are generated for other all other un-trained 3D models, as demonstrated by Angrish et al. Each of these embeddings is used as a representative of the corresponding part for calculating global similarity measures.

4.5. Knowledge Graph Construction

4.5.1. Assembly Hierarchy Subgraph

Our method translates the representation of Assemblies in tree-structure (commonly seen in standard assembly designs) to a graph. Applying this to the FabWave Open dataset, the Assembly Hierarchy Subgraph was generated. This subgraph has nodes of type *assembly*, *subassembly* and *part*, in order of descending hierarchy. The edge type *subset_of* is used to denote subassembly-assembly and the part-subassembly hierarchies, thus forming a directed graph.

4.5.2. Part Similarity Subgraph

Further extending the knowledge graph with CNN-based part similarities, the subset of 110,000+ individual part files is considered for similarity computation using their respective vector embeddings. Each of these parts is represented by a *part* node and the relations between the nodes are given by an undirected edge of type *similar_to*, with the edge weights denoting the cosine similarity (Eqn. 1) between the Part Embeddings.

$$\cos \theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (1)$$

This measure ranges from 0 (highly dissimilar) to 1 (highly similar) in the positive space. For this dataset, there exists 12 trillion possible similarity-based connection amongst the parts. Preliminary analysis of these cosine similarities (Fig. 2(a)) shows that a large majority of these connections denote a weak similarity between the parts. Constructing the KG using all these relations will result in a graph with a very high search complexity. However, closer inspection of the frequency distribution of edge weights of the complete graph in Fig. 2(a) reveals that there are a relatively small but significant number of these potential connections that denote a high level of part similarity. This phenomenon is likely to appear in databases with a small but significant fraction of highly similar 3D shapes, as is common in mechanical design. The interpretation of a “high” similarity level varies between different datasets; this is addressed by the threshold parameter " α " described in section 3.2. A much lower threshold will result in far more edges and a highly complex graph, which can provide better search results (at a far higher cost) than a sparser graph. The threshold parameter is empirically obtained to ensure the scale-free or heavy-tailed nature of the network, which will ideally ensure a representative yet sparse graph. Fig. 2(b) shows the observed degree distribution of the thresholded Part Similarity Graph at $\alpha = 0.85$. Based on an analysis of this graph’s empirical degree data, a Power-law distribution

($\gamma = 1.808$), with an exponential cutoff is found to be a better fit than the exponential distribution with a log-likelihood ratio $R = 18.1848$. This satisfies the minimum condition to exhibit the heavy-tailed nature of the degree distribution [71], ensuring a compact Scale-Free graph, maintaining the inherent structure of a complete graph. The hybrid construction method described ultimately results in 27 million relations.

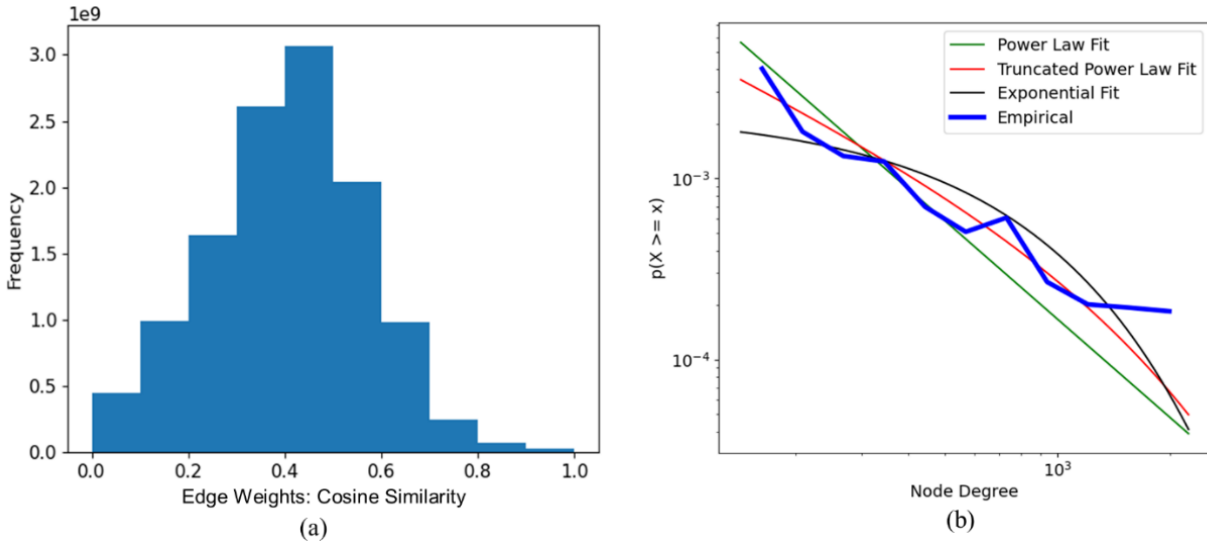


Figure 2: Graph Structure (a) For the Complete Graph: Histogram of Edge Weights: Only a small no. of edges are relevant; (b) After Weight Thresholding [0.85, 1]: Log-log Fit of the Dist. of Node Degree shows Exp.-Power-Law Fit

4.6. Structuring the Knowledge Graph: Community Detection in the KG

The most popular approaches to construct KG-based recommendation systems apply manual segmentation to the KG using classifying schemes, which can limit the results due to the large variety of possible labels. Recent KG frameworks apply unsupervised methods to group sub-domains in a larger topic to reduce time to generate and use the knowledge graphs, while still providing accurate recommendations. Some examples of this approach are seen in the use of local graph structure and hierarchy-aware systems for predictions [72], [73] and has been applied to fields such as the organization of scientific publications to aid researchers [74], [75]. In the

context of optimal product design search, community detection within the KG was implemented with the following aims:

- (i) Finding Part nodes that are very similar to each other in an unsupervised manner. This similarity detection reduces the search space.
- (ii) Using the internal structure of these communities to simplify searching the KG.
- (iii) Provide a framework for probabilistic label recommendations for un-labeled data, using node similarity within the communities. (Applicable to both Inductive reasoning for new incoming nodes, as well as Transductive reasoning for existing nodes)

For the thresholded *Part Similarity Subgraph* of the KG, the underlying structure of the graph is discovered by finding closely connected communities with highly correlated sets of relations of their respective nodes. For our application, community detection was applied to a single node type, and can thus be considered analogous to clustering methods used in data mining [76]. However, these methods can also be implemented considering multiple relation types if required, which ensures the ability of these techniques to be extended to more complex data sources. Two important assumptions were made in order to detect clusters/communities in the *Part Similarity Subgraph*: (i) Communities are disjoint/non-overlapping, and (ii) The network does not vary over time.

Extensive reviews of the best community detection/clustering algorithms for graphs are given in [77] and [78], along with an in-depth performance comparison on benchmarks. Based on these reviews, we implement Louvain, Leiden, Label Propagation, Walktrap, and Infomap algorithms for Graph Clustering on the Part Similarity Subgraph of the KG. Importantly, these algorithms do not require prior information such as the number or size of communities in the

network. This makes them well-suited for unsupervised detection of communities, with the only hyper-parameter being the strength of relations in the graph. This is an intuitive parameter to control and can be evaluated by the effectiveness of the graph partitions. The complexity of each of these algorithms are briefly summarized in Table 1, for a graph with n nodes, e edges and average degree k . Performance of these algorithms are evaluated using intrinsic measures described in section 6.

Table 1: Community Detection Algorithms: Complexity

Louvain [79]	Has a complexity of $O(e)$.
Leiden [80]	Demonstrated to be faster and to find higher quality clusters than Louvain. $O(n \log \log (k))$ complexity. [81]
Label Propagation [82]	Depends on community structure alone; $O(e)$ complexity.
Walktrap [83]	Has $O(n^2 \log (n))$ complexity for sparse networks.
Infomap [84]	Simulates the flow of information in weighted networks. $O(e)$ complexity.

4.7. Part Retrieval from the KG and Node Centrality

Intelligent retrieval methods using knowledge graphs have been well-studied in multiple domains. For Knowledge Graph constructed with vector embeddings, we chose an approach that leverages the detected communities for subsequent greedy nearest-neighbor search. Assuming that communities have been detected effectively, a single descriptor node for each cluster can massively reduce the complexity of nearest neighbor search, while maintaining accuracy of search results. We implement this using node centrality measures within communities. While communities themselves are evaluated using extrinsic measures, the combined community-retrieval method is evaluated manually using retrieval measures described in Section 6.

Node-based centrality measures for networks such as Degree, Closeness, Eigenvector, Katz centrality and PageRank scores have had success for social network analysis and web search results as seen in [85] and [86]. In particular, the PageRank algorithm [87] has been extensively used in weighted graphs to rank webpages by order of the most effective at transmitting information to its neighbors. For this design-based KG, the priority for selecting central nodes is its effectiveness in representing the nodes within its own community. Using just the network structure to find nodes that have high control over the flow of information, the PageRank technique was adopted to find central nodes of the communities in the Part Similarity Subgraph, which were designated as Level-1 Centers. These L-1 center nodes have the highest centrality scores within each community and are subsequently used to simplify retrieval of data within KG.

Information retrieval from large databases is a well-studied problem, with the use of Approximate Nearest Neighbor (k-ANN) Search suitable for high-dimension metric spaces. Well studied approaches to solving this problem include Space Partitioning/Tree methods (e.g. KD-Trees). Another popular approach to large-scale ANN tasks is the Proximity-Graph method which generates graphs based on vector similarity metrics, such as the Navigable Small-World Graphs (NSW) method [88]. By maintaining a proximity graph with both long-range links as well as short links to closer neighbors, NSW graph methods perform greedy search with poly-logarithmic complexity by starting at low degree nodes and progressing to higher degrees. Hierarchical NSW (HNSW) [89] improves this technique to logarithmic search complexity, separating links by magnitude into multiple layers and performing greedy search on this multi-layer index. In the case of a similarity-based graph, the lowest layers contain relations with high similarity weights, with decreasing relation weights in higher levels. Benchmark studies on ANN search demonstrate that

graph-based algorithms such as HNSW outperform Tree and Hashing-based algorithms at high recall values, while maintaining a relatively compact index size even for large datasets [90].

Leveraging these advantages of the algorithm and the community structure of the KG as described in section 5.1, the part retrieval technique was implemented as follows:

- (i) The HNSW algorithm was used to index the previously defined L-1 Centers/cluster representatives for Approximate Nearest Neighbor search. (L-1 Centers number 25,706 of the 110,770 total Part nodes, and this reduces the time required to create the index.)
- (ii) Vector-embedding were generated for the CAD model of interest to the user, and subsequently Top-k search was performed on the KG L-1 Centers using the HNSW index.
- (iii) This process was then followed by an exhaustive Nearest Neighbors search within the Communities that the top-k L-1 Centers belong to, thus providing the best matches within the top candidate communities. The process is visualized in Fig. 3.

The effectiveness of the hybrid ANN and exhaustive search was compared to Exact Nearest Neighbor Search on all 110,770 Part nodes using the popular KD-Tree method.

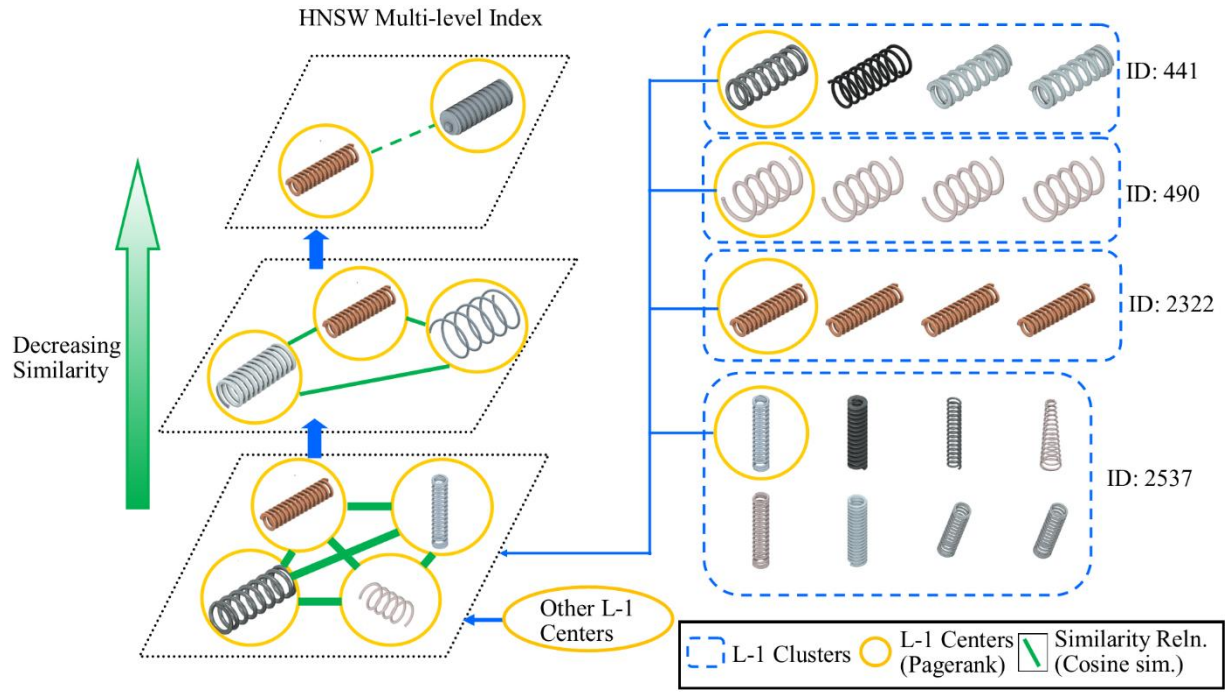


Figure 3: Construction of Optimal Retrieval Framework for Part Nodes

CHAPTER 5: EVALUATING COMMUNITY DETECTION AND RETRIEVAL

5.1. Community Detection

The quality of Community Detection and Clustering can be evaluated using External or Intrinsic measures. External measures include Adjusted Rand Index and Normalized Mutual Information (NMI). However, external measures require knowledge of ground truth communities, which makes them unsuitable for evaluating our approach which is exploratory in nature. Two intrinsic measures are used in place of these external metrics: Modularity Density and Silhouette Score, which is another popular method used to evaluate clustering effectiveness. Modularity Density is a graph specific method which uses node degree to evaluate graph partitions and is well correlated with the ground truth-based NMI metric. Both these scores rely on simultaneously finding the level of cohesion within clusters, and the level of separation between clusters. They are described as follows.

For a community C in a graph partition S , Modularity Density (D) is given by:

$$D = \frac{1}{n_c} \sum_{i \in C} (2\lambda k_i^{int} - 2(1 - \lambda)k_i^{out}) \quad (2)$$

where i is a node within cluster C , n_c is the number of nodes in cluster C , k_i^{int} is the degree of node i within cluster C , and k_i^{out} is the degree of node i external to cluster C . This measure is then averaged over all clusters to find the score for the entire graph. λ is a tuning parameter; 0.5 is the standard value used.

Silhouette Score for a graph with n nodes is given by:

$$S = \frac{1}{n} \sum_{i \in n} \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad S \in [-1, 1] \quad (3)$$

Here, for a node i belonging to cluster C , $a(i)$ is its mean intra-cluster distance (weighted). $b(i)$ is the mean (external) weighted distance from i to all nodes in the cluster X ,

which is the nearest cluster to the center of C ($X \neq C$). The best possible Silhouette Score for a clustering is 1.

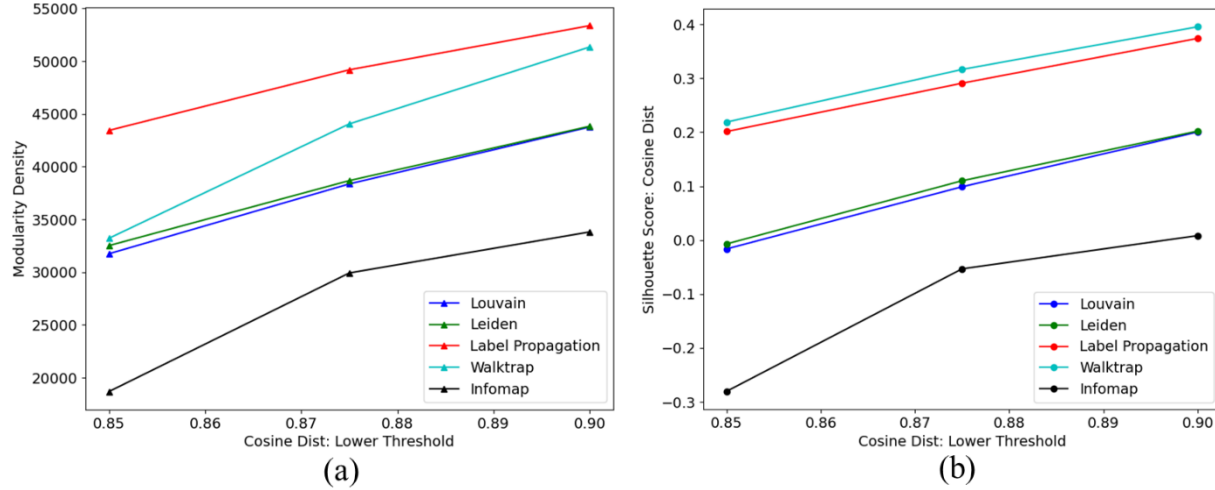


Figure 4: Performance of Community Detection Algorithms for Graphs based on (a) Modularity Density (b) Silhouette Score

Implementing the community discovery algorithms on a range of thresholds from $\alpha = 0.85$ to $\alpha = 0.9$, Label Propagation and Walktrap algorithms at a lower weight threshold level of $\alpha = 0.9$ shows the best performance in terms of Modularity Density and Silhouette Score respectively as seen in Fig. 4. The communities generated by the Label Propagation algorithm were chosen as the best representation of structure at the lowest level of granularity among parts, since Modularity Density has the highest correlation to the extrinsic NMI metric for graphs [91]. This algorithm produces 25,706 communities. On closer examination, 18,055 of these communities contain only a single part. The largest cluster contains 4295 parts; this confirms our previous intuition regarding a significant number of repetitive part shapes within the database. In summation, through community detection, 7,651 highly similar groups are identified, incorporating 92,715 out of the 110,770 parts.

Selecting central nodes for each of these communities using the PageRank algorithm gives 25,706 nodes, one for each at the Level-1 communities. This was considered the first level of the hierarchy in the knowledge graph or L-1 Centers. This reduces the detection space significantly, allowing for a much simpler graph representation. The L-1 Centers embeddings are indexed for search using Hierarchical Navigable Small-World Graphs (HNSW) method for Approximate Nearest Neighbor search.

5.2. Top-k Retrieval Evaluation

Retrieval is evaluated for 29 standard part categories from the FabWave Categorized dataset described in Section 3.1. Following the method for retrieval described in Section 5.2, an Approximate Nearest Neighbor (ANN) search on the 25,706 L-1 Center node vectors is used to find top-k clusters, followed by exhaustive k-Nearest Neighbor (NN) search on the top clusters. This hybrid method is compared with exhaustive k-NN search on all 110,770 nodes using the popular KD-Tree method. Retrieval is evaluated using Mean Average Precision (mAP) metric, defined as follows:

$$mAP@k = \frac{1}{num_query} \left(\frac{1}{GTP} \sum_{i=1}^k \frac{TP_{seen}}{i} \right) \quad (4)$$

Here, num_query refers to the number of queries made for each class, GTP refers to the total number of ground truth positives for the query, and TP_{seen} is the number of true positives seen till k . 20 query parts are evaluated for each class, with each search returning the top-5 neighbors of the queried part. Table 2 shows the performance comparison of the HNSW Index-based Approximate Nearest Neighbor (ANN) search for knowledge graph part nodes, compared to the KD-Tree based kNN search. Using the advantage of L-1 centers of the graph communities, the HNSW ANN search index is constructed in far lesser time in comparison to the KD-Tree

method. Searching within the graph using the hybrid HNSW-kNN method is also achieved in a fraction of the time taken by KD-Tree. This can be attributed to two factors: the reduction of number of node vectors to index for search, and the decrease in performance of KD-Tree with high vector size.

As seen in from the results in Table 2, better mAP can be achieved in much lower time using ANN search, due to a combination of graph community detection and the advantages of the ANN algorithm. Class-wise mAP values are given in Appendix B, and Appendix C, which shows examples of retrieval for certain example classes. An analysis of these results show that the ANN approach performs better for Bushing, Collets, Gasket, Headless Screws, Hex Headed Screws and Washers; in comparison to this method, the KD-Tree approach performs better for Brackets and Machine Keys. This is attributed to errors in the design data used for querying (FabWave Categorized); some incomplete part designs in this dataset lead to substandard part matches for these two classes in both cases. However, the results of Hybrid ANN + kNN approach are generally better or comparable to that of the baseline KD-Tree method. These results represent the combined effect of node clustering in the graph and hybrid ANN search, demonstrating the overall effectiveness of the KG and ANN-based retrieval workflow. In enterprise databases, this combined method has the potential to account for the tradeoff between speed and accurate representation of large amounts of data.

Table 2: Comparing the two retrieval methods

Method	Search Index		Query Time (seconds)		Retrieval: 29 Std. Categories		
	No. of Indexed Nodes	Constr. Time (sec)	<i>Mean (SD)</i>		mAP@1	mAP@3	mAP@5
HNSW followed by kNN	25,706	17 sec	HNSW	KNN	0.6913	0.7242	0.7278
			0.2362 (0.0152)	0.0649 (0.1015)			
KD-Tree based kNN	110,770	76.61 sec	KD-Tree		0.6439	0.6706	0.6760
			2.7859 (0.9022)				

5.3. Implementation Details

The 3D CAD model processing and image capture was implemented using Fusion 360 design software and Blender 3D graphics tool. The Knowledge Graph and all algorithms were implemented on a Windows 10 machine with Intel Xeon W-2225 processor and 64GB RAM. The KG is implemented on Neo4J, queried with the Cypher language. Visualizations are provided using Graphlytic and Matplotlib. Part and Assembly metadata are stored on MongoDB, with file storage on Google Drive. Graph Representation and Clustering were achieved using NetworkX [92], Neo4J's Graph Data Science extension, CDLib [93], Infomap [84], Scikit-learn [94], and Scikit-Network [95] packages on Python. The Powerlaw package [71] was used for testing truncated power-law nature of the degree distribution. hnsplib package [89] is used for final retrieval using approximate nearest-neighbor (ANN) search. The Fuzzy-Rule system in section 7.1 is implemented using the Simplful library [96]. Mlxtend [97] was used to implement the FP-Growth algorithm for recommendation systems shown in Section 7.3.

CHAPTER 6: RECOMMENDATION SYSTEMS USING THE PRODUCT DESIGN KNOWLEDGE GRAPH

Search and classification for Product Databases is an active field of study, with multiple potential applications such as design reuse [37], manufacturer selection [98], function prediction [41], design recommendation and improving sustainability in design [99]. The effectiveness of the KG for retrieval of similar components was demonstrated in section 6, which can be used for design reuse by engineers or Numerically Controlled (NC) process plan reuse. Motivated by such problems, this section shows the potential to apply recommendation systems to associated problems for engineers, using the advantage of richer data associativity within the KG.

Retrieving similarly shaped components is one common application for product data stores. Design efficiency can also be greatly improved by helping designers obtain ‘associated’ standard or non-standard components, which are likely to be a part of the same subassembly as similar parts of interest. These parts can be ordered based on the frequency of occurrence in other similar subassemblies, using the part cluster IDs. Section 6.1 describes a Fuzzy Rule-Based method to obtain these relevant components from the KG. Using an approach similar to those used in [100], [101] and [54], section 6.2 expands on the use of the KG for component-based assembly similarity evaluation using shape similarity. Efficient retrieval of CAD models in large databases consisting of combinations of hundreds of thousands of components is an industry problem that has primarily relied on PLM/PDM based textual search. Multi-modal search using 3D data in conjunction with associated textual data has the potential to increase effectiveness of the required suggestions; this is demonstrated in section 6.3.

6.1. Recommending Co-Occurring 3D Designs

Design engineers spend a significant percentage of their time in locating and sorting information, often comparable to the time spent on solving actual design problems [102]. Recommendation systems can be powerful tools to improve designers' productivity; standardized components are often listed in design software libraries for use by engineers. In the context of new designs, engineers often spend their time designing critical components which have multiple engineering constraints in an iterative manner. The KG based recommendation system can improve designer productivity by providing both standard and non-standard parts associated with high-priority critical components, with multiple variants to choose from. Components that occur in the same subassemblies as the closest matches to the query part are found through a combination of ANN search amongst the parts, followed by graph traversal for the Assembly hierarchy structure in the KG. The parts associated with these Nearest Neighbors (parts in the same subassembly or assembly) were then analyzed and provided as recommendations to the designer, ranked by relevance.

From the existing relations of the graph, there are two relations that can be manually inferred: *co_occurring* and *highly_correlated*. Parts that belong to the same subassembly are defined as *co_occurring*. Formally, for two parts A and B and a given subassembly SA , one fact that can be inferred from the graph is:

$$(A, subset_of, SA) \wedge (B, subset_of, SA) \Rightarrow (A, co_occurring, B)$$

Parts that belong to the same cluster are defined as *highly_correlated*. Thus, for parts A & B in the same cluster,

$$(A, same_cluster, B) \Rightarrow (A, highly_correlated, B)$$

We define the following rule for obtaining potentially relevant co-occurring parts R , for a given query part Q :

$$(Q, nearest_neighbor, A) \wedge (A, co_occurring, R)^* \Rightarrow (Q, co_occurring, R)^*$$

$(A, co_occurring, R)^*$ is inferred through the rule:

$$(A, co_occurring, B) \wedge (B, highly_correlated, R)^* \Rightarrow (A, co_occurring, R)^*$$

Parts A and B are “transitory” parts obtained from the Knowledge Graph, in the process of finding relevant parts R . Fig. 5 illustrates a method to use the KG for this application, considering an Unthreaded Flange as the query part.

The consequence of the above rules is true only to a certain degree; thus, it is denoted by a *. This is due to the potential inaccuracies induced by the clustering method due to the use of weighted similarity relations. We approach this rule-based inference problem using a Generalized Fuzzy Rule method, which has seen success in associated fields, most recently in manufacturing performance measurement [103]. Using these rules, we obtain parts that can potentially be used in conjunction with a given 3D part of interest. The final measure of truth of the chained rules is calculated using the well-studied Fuzzy Inference System described by Mamdani and Assilian [104]. This is a linguistic logic modeling technique where rules are stated using “IF-THEN” rules, with both the prior and consequent of the rule represented as fuzzy sets. The degree of membership of each element in the fuzzy set is determined by Membership Functions (range of 0 to 1) heuristically obtained from source data.

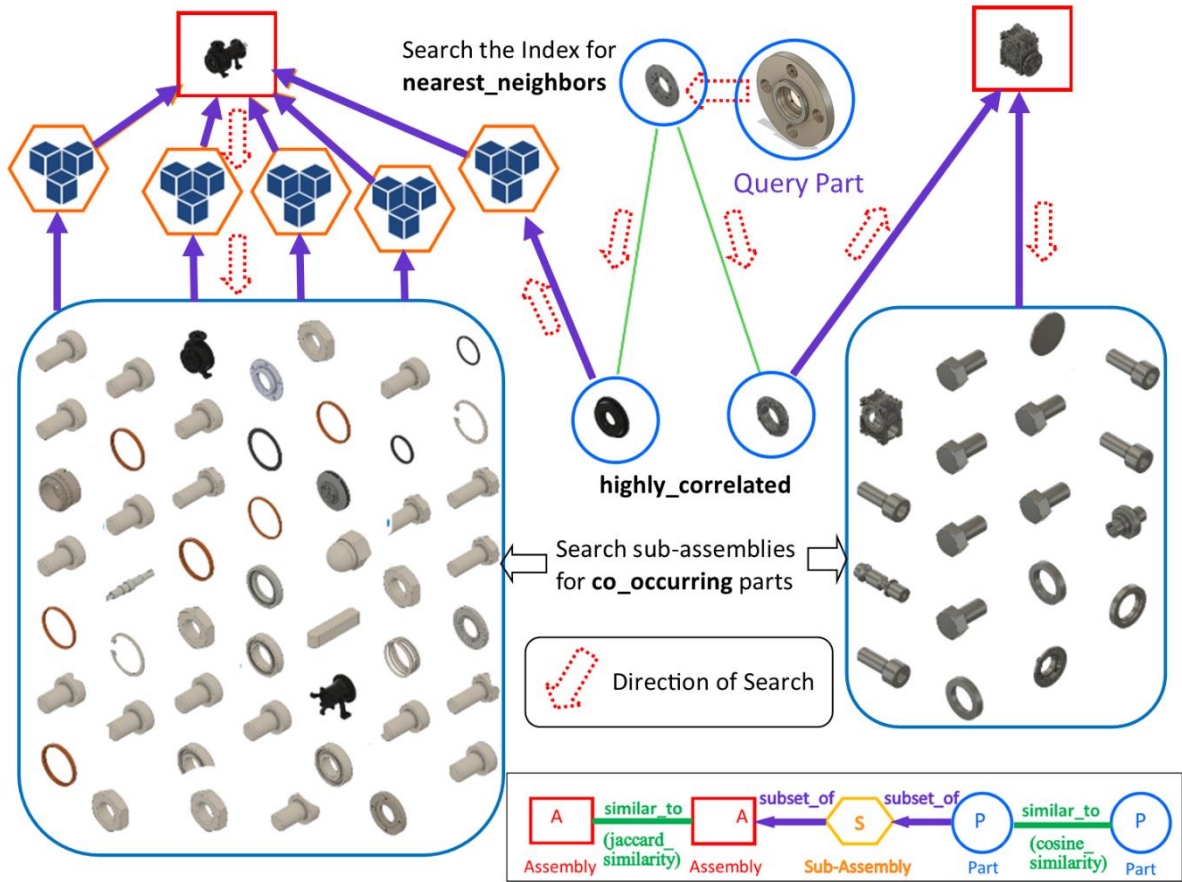


Figure 5: Contextual Design Recommendation Process

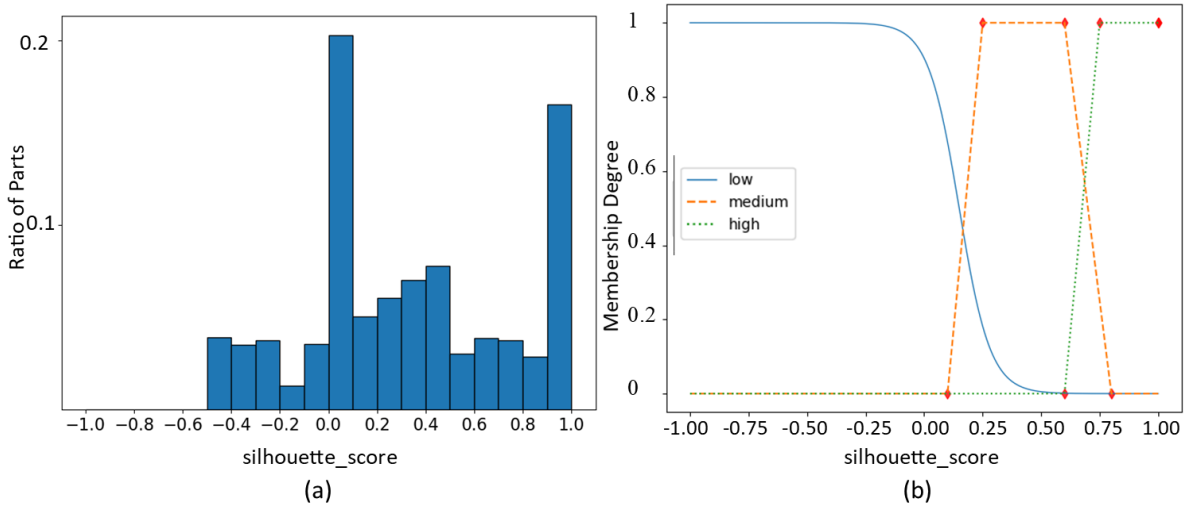


Figure 6: (a) Distribution of Silhouette Score in the KG (b) Membership Func. for "highly_correlated" (Low, Medium and High)

The uncertainty is introduced in the rule-based system through the component $(B, highly_correlated, R)$. “*highly_correlated*” is based on parts belonging to the same cluster; based on the results in section 6.2, the effectiveness of this property can be measured using the Silhouette Score of the cluster containing B and R . Since a lower silhouette score in the range $[-1, 1]$ indicates a low or negative correlation between the parts, 3 different fuzzy sets are utilized to describe the property “*highly_correlated*” denoted as Low, Medium or High. Based on prior knowledge of the silhouette score as seen in Fig. 6, the sets are defined with the following membership functions: Sigmoid (Low) and Trapezoidal (for Medium and High). The property “*co_occurring*” is represented by High or Low; since it is a fact derived from the graph in $(A, co_occurring, B)$, this would correspond to a True/False value. Based on these definitions, the linguistic rules for the fuzzy inference system are thus re-defined as follows:

IF (A, co_occurring, B) IS High AND (B, highly_correlated, R) IS High THEN (A, co_occurring, R) IS High
IF (A, co_occurring, B) IS High AND (B, highly_correlated, R) IS Medium THEN (A, co_occurring, R) IS High
IF (A, co_occurring, B) IS High AND (B, highly_correlated, R) IS Low THEN (A, co_occurring, R) IS Low
IF (A, co_occurring, B) IS Low THEN (A, co_occurring, R) IS Low

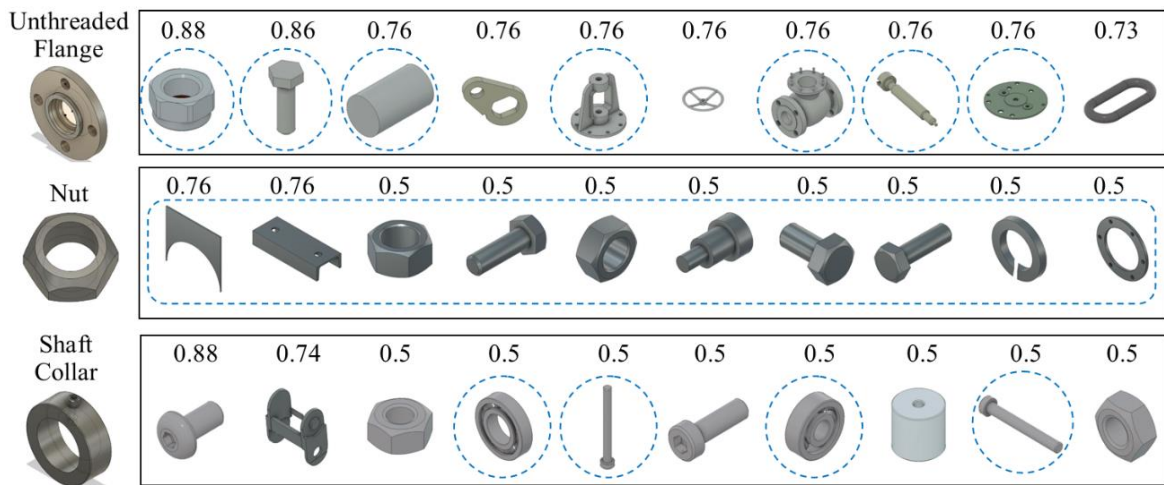


Figure 7: Recommended Co-occurring Parts with their corresponding Truth Values

The final truth value is obtained from the conjunction of rules as shown previously in this section, and denotes its actual relevance based on the structure of the knowledge graph. Fig. 7 shows results of the fuzzy inference system with relevance scores. A manual check of the results shows a few relevant co-occurring parts for 3 example queries, which are circled in blue. Since the truth values of the recommendations are based on the graph structure (which is not perfect), the ordering and accuracy of the results is not ideal; further investigation can reveal more complex and accurate rule-based systems for part recommendation from the knowledge graph.

6.2. Detecting Similarity of Assemblies

To compute similarity of assemblies, the clusters of their constituent parts in the KG can be used as a factor. Every assembly was represented by a multiset containing each of its constituent parts' community IDs at Level-1. The rule for similarity between two assemblies X & Y can thus be defined as follows:

$$\begin{aligned} (p, \text{subset_of}, X) &\Rightarrow \exists A(A, \text{subset_of}, X) \\ (q, \text{subset_of}, Y) &\Rightarrow \exists B(B, \text{subset_of}, Y) \\ \exists A \exists B (A, \text{same_cluster}, B) &\Rightarrow (X, \text{similar_to}, Y)^* \end{aligned}$$

The *similar_to* relation in the above rule between these assemblies X & Y was quantified using the Jaccard index for the similarity of sets. A modified version of the Jaccard Index (Eqn. 4) which rewards repeated values in a multiset was used:

$$J(X, Y) = \frac{|XX \cap YY|}{|XX| + |YY|} \quad J \in [0, 0.5] \quad (4)$$

where XX and YY are the multiset collection of part L-1 cluster IDs for two assemblies X & Y respectively.

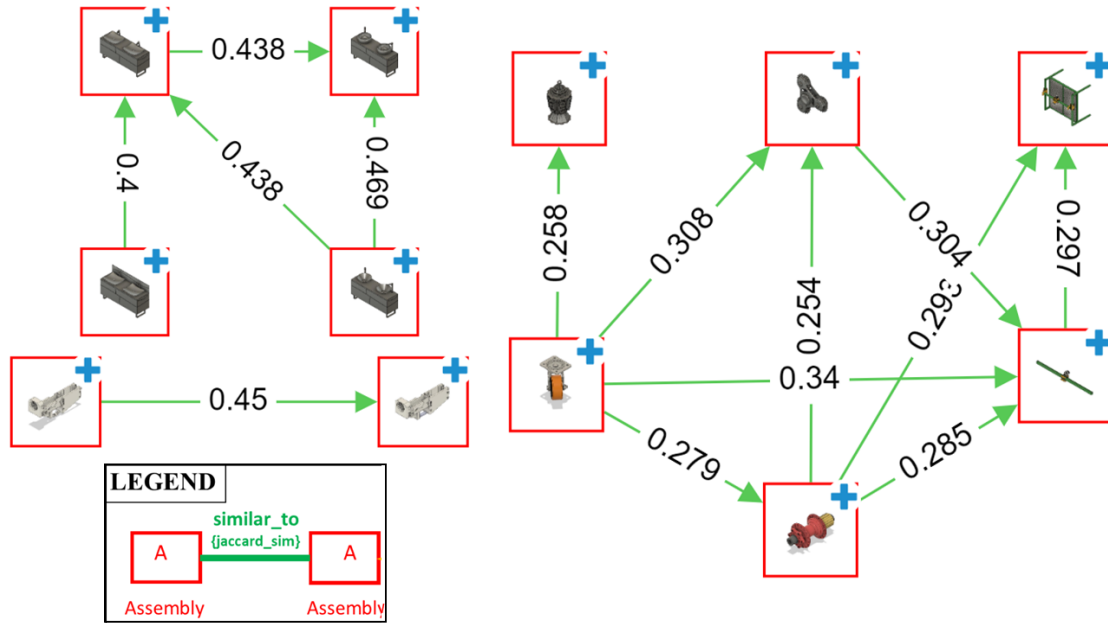


Figure 8: Clusters of Assemblies based on individual part similarity

This similarity measure can be used to compute clusters among assemblies, based on the similarities of their individual parts. Fig. 8 shows some examples of community structure for assemblies. This can easily be extended to sub-assembly similarity search by the same method. The practical implication of building this data in the KG is future improvement in modular product design, enabling flexible production and low volume customization. Using these advantages, the best practices previously followed for design-for-assembly and design-for-manufacture can be adopted by designers and further refined to ultimately create better products. The flexibility of the KG approach can be used to include data pertaining to electromechanical or hydraulic systems, which can provide valuable information to designers and simplify communication across domains, with types of node clusters detected based on multiple relation types.

6.3. Design and Manufacturing Parameter Recommendations

As described in Section 3.1, the Knowledge Graph contains annotations from users on various design and manufacturing parameters for parts in the database. Although these annotations cover only a fraction of the KG nodes, the concept of collaborative filtering can be applied based on the clusters of parts detected within the KG. Collaborative Filtering (CF) is a popular method used in web-based recommender systems to provide fast and accurate recommendations to users, based on the preferences of similar users who have previously interacted with the system. In the context of this KG, the recommendations provided depend on the subset of previously annotated nodes which are the most similar to a search node. Using a combination of the clusters in the graph and pattern mining, a method that can be described as Clustering-Model Based Collaborative Filtering technique [105] was used to provide recommendations.

Based on the annotations for the parts given by users described in Appendix A, design and manufacturing parameter recommendations are generated using the Frequent-Pattern Tree Growth (FP-Tree) approach [106]. This approach decomposes pattern mining into smaller sub-tasks for more efficient search and is faster than the well-known Apriori algorithm for the same task. For each part node that has been annotated, the set of individual descriptions are considered for parameter search. This algorithm evaluates the relevance of parameter recommendations using the probability of co-occurring parameters, given primarily by two parameters: Support and Lift. Support is the measure of how often a parameter A occurs in the relevant nodes of the KG i.e, its frequency of occurrence in the set of node parameters. Lift is an extension of this parameter, which describes whether two parameter sets X and Y are independent. It is given by Eqn. 5:

$$Lift(X \Rightarrow Y) = \frac{Support(XUY)}{Support(X) \times Support(Y)} \quad (5)$$

A Lift value of 1 suggests that the parameter sets X and Y are independent of each other.

Higher the lift value, the stronger the degree of dependence of the parameter sets X & Y.

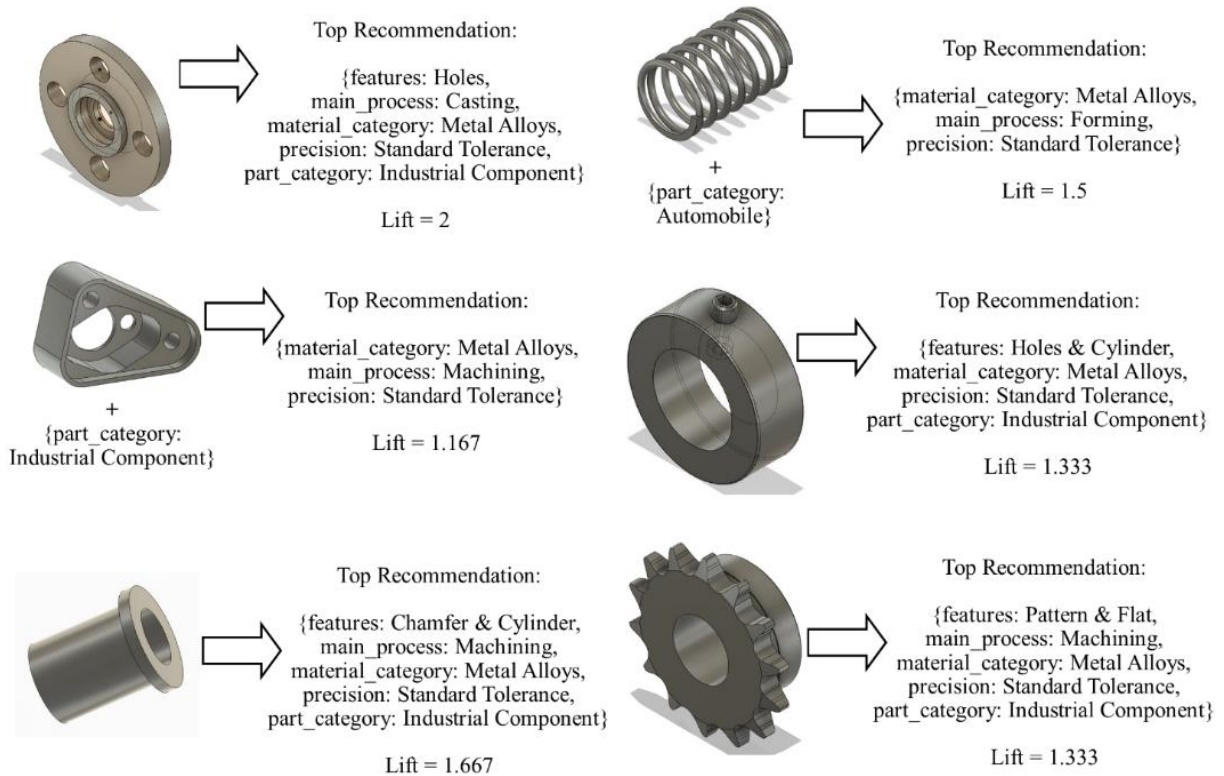


Figure 9: Recommendations for 6 parts: Recommendations for Spring and Triangular component are provided using Multimodal search using 3D Data + Text

The recommended parameters for the Unthreaded Flange include the feature commonly seen in associated parts in the database ('Holes'), along with its material and process suggestions ('Metal Alloys', 'Casting'). Fig. 9 shows the recommendations for multiple parts. Parameters for the Spring component are queried with a part category ('Automobile') along with shape and dimensions, providing parameter recommendations relevant to automotive spring components. Similarly, the triangular component is queried with a part description denoting 'Industrial

Component'. These results are dependent on the amount of data available; currently, the annotations are sparse relative to the size of data. Further investigation on graph-recommendation systems can yield superior recommendations.

6.4. Discussion

Structuring product data in clusters reveals the nature of this design data store created from open sources: a mix of unique designs and a significant fraction of repeated non-standard parts, along with standardized parts. Similar data in enterprise siloes can serve an important purpose for future designs: either through design reuse or by enabling data-driven parameter suggestions, in contrast to manual retrieval by text-based search or advanced image only visual search. With minimal human intervention, existing patterns detected within the corpus of designs provide a rich source of engineering decisions. The ability of an automatically structured KG for efficiently implementing recommendations is demonstrated, despite minimal availability of annotated data.

We note a few potential weaknesses to the KG construction approach and the data used:

- (i) Local relation density differences in large KGs are not considered for structure detection and may lead to issues with accuracy and scalability.
- (ii) The assumption of disjoint communities, and the static nature of these communities will not hold with extensive additions to the database over time.
- (iii) The current approach uses shape similarity alone to both prune relations and detect local structure in the graph. This method does not currently consider other relation types that can potentially be created, particularly with the significant amount of associated data with product models available within enterprise stores.

- (iv) The relative lack of rich annotations or textual descriptions in our dataset limits the types of relations and recommendations that can be provided to designers.

Future improvements to the data must focus on improved representations of parts through a combination of automated feature/process recognition, and connecting existing data to design documents, manufacturing parameters, simulations and other geometric and textual part data. With improved data connectivity, the KG can be used for more complex KG-based recommendation applications such as mining design rule updates from downstream manufacturing and quality inspection data, similar to the work by Ko et al [107].

The construction of the KG has a large bearing on the quality of recommendations that can be provided using this technique. For the cosine similarity relations created using a bottom-up method, the thresholding parameter described in section 3.2 and implemented in 4.3.2 affects the quality of edges in the Part Similarity Subgraph. A lower threshold could provide better recommendations at the cost of greater complexity. However, if unweighted methods are used for graph traversal and recommendation systems, it is possible that low-quality recommendations could be obtained. In this semi-supervised framework, it is thus critical to choose a threshold that is suitable for downstream tasks. An incremental edge addition method that optimizes search results is an area for future exploration.

The assumption of disjoint and static communities relies on the large size of the database relative to new additions. Future additions to the knowledge graph require these issues to be addressed to ensure scalable solutions, with dynamically responsive KG structuring methods. The problem of localized graph structure variation is visualized in Fig. 10. Basic inspection reveals that while there are very dense clusters in the graph with high internal part similarity, there are also multiple clusters that can be merged. There are also some relatively large clusters

that can be divided into distinct subsets. Fine-tuning these clusters can be achieved by Incremental Cluster Repairing of existing clustered similarity graphs, using newer methods such as n-depth re-clustering as demonstrated by Saeedi et al [108]. However, this is not an issue with effectiveness of clustering alone; the exclusive use of similarity relations for clustering is another contributor to variability. Using multiple relations and node features to find structure in the KG is the next area of focus to improve the KG. The interrelated aspects of clustering efficiency and encoding multiple relation types can be addressed by graph construction techniques that consider the dynamically varying nature of different neighborhoods and multiple-relation types of the graph, as seen in the work by Fakhraei et al [109]. Alternately, the graph structure can be encoded in rule-based learning or graph embedding methods to combine structuring and reasoning.

The size of enterprise data is often several orders of magnitude larger than the data corpus used in this paper; scalable approaches for multi-modal search are an important priority. Future research directions in intelligent product design using KGs necessitate larger and more complex graphs. The Approximate Nearest Neighbor and Collaborative Filtering methods used for reasoning over the KG have limitations: the ANN method currently indexes only a single relation type, and collaborative filtering suffers from cold-start issues with prediction due to data sparsity. The techniques must eventually be replaced by multi-hop inference using latest knowledge reasoning methods, most popular of them being Graph Embeddings. While embeddings work well on large-scale graphs, they tend to lack interpretability. To create recommendation systems in the design domain, Inductive Subgraph Reasoning methods show more promise, since they learn the underlying relational semantics of the graph. Work by Teru et al [110] demonstrates the strong performance of this technique in comparison to embedding

based methods on standard benchmarks. In the future, we plan to investigate similar methods for more complex decision-making problems, focused on design parameter recommendation, manufacturer search and data driven automation.

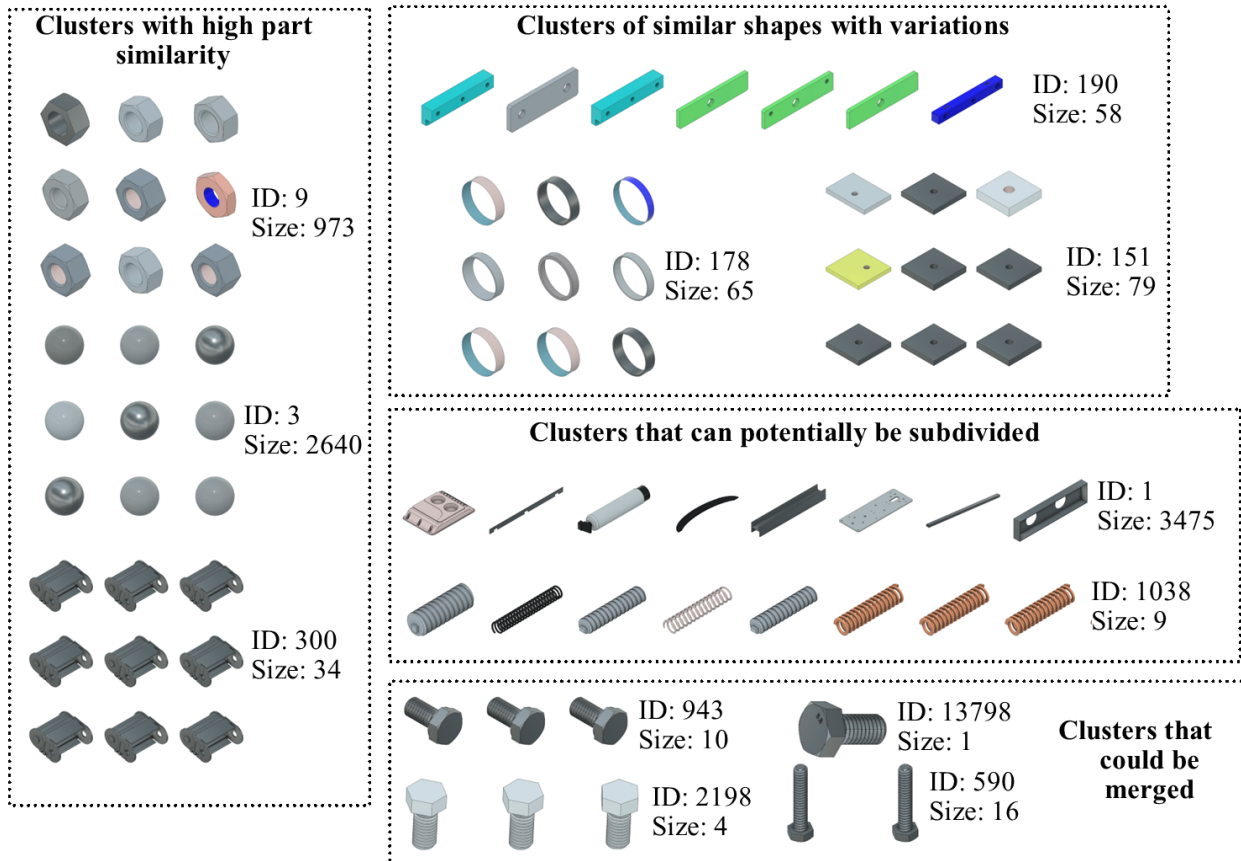


Figure 10: Visual analysis of some clusters

CHAPTER 7: IMAGE-BASED OBJECT POSE RECOGNITION FOR ROBOTIC MANIPULATION

7.1. Introduction

Automated systems in manufacturing environments have traditionally relied on fixed programming based on well-defined positions. While there has been a rise in RGB-based image perception systems for robotic applications and the use of image recognition algorithms such as edge or color detection, these systems are often fine-tuned for specific parts and highly specific scenarios such as conveyor pick-and-place. In a scenario where general-purpose robots are becoming ubiquitous, their use in flexible manufacturing scenarios require increased adaptability. This improved flexibility must be twofold: perception systems must be able to recognize different objects in a variety of environments with high accuracy. In addition to this, the automated system must be able to take decisions in different scenarios with reduced human intervention.

The challenge of perception in dynamically varying conditions is in obtaining reliable and accurate data about the state of the environment, which requires scene reconstruction. Typically, scene reconstruction for robotics focuses on creating accurate occupancy maps using methods such as Simultaneous Localization and Mapping (SLAM). This is used in path planning and collision avoidance for robotic applications. However, due to a lack of specific semantic information about the environment, these methods are only suitable for relatively simple navigation tasks. Recently, there have emerged improved methods for accurate detection of object states in 3D space [111], [112] termed 6D pose recognition, as well as efficient representations of their states suitable for high-level task planning [113]. In particular, visual scene graphs have emerged as a holistic representation for hierarchical modeling of objects, kinematics and human-robot interactions. In the case of robotic manipulation, object locations as

well as their interrelationships are represented in 3D scene graphs, which have been demonstrated to be highly efficient for high level and large-scale task planning [114].

This work proposes a framework for robotic object manipulation in dynamic scenarios, using richly available CAD data that is commonplace in the engineering industry. The contributions of this work are:

- (i) A methodology to generate 6D object poses for industrial objects from RGB image data in an unsupervised fashion, based on a Pixel-to-Surface correspondence technique
- (ii) A methodology for sequential robotic manipulation using Scene Graphs, based on a combination of image perception and semantic information obtained from CAD data.

7.2. 6D Pose Recognition for Objects: Related Work

Learning-based approach to Object Pose Recognition from image data has been an active field of interest over the last decade. Common applications for localization in 6 degrees-of-freedom (6 DoF) include outdoor camera localization and object pose detection from RGB images. The approaches for these tasks build on established methods such as Simultaneous Localization and Mapping (SLAM), and closely related methods in image segmentation and object detection. A number of datasets have been introduced to benchmark these approaches. Notable ones include Linemod [115], HOPE [116], YCB-Video [111] for household objects; similar ones for manipulation in warehouse environments [117], [118], and other datasets that replicate objects seen in industrial environments. Two significant examples are the T-Less dataset [119], which includes texture-less objects relevant to electrical assemblies, and the ITODD dataset [120] containing objects from industrial manufacturing setups. Earlier datasets

such as the Linemod and YCB-Video contain images from real environments with significant time spent on human annotations. However, with the introduction of the standardized format of Benchmark on 6D Pose Estimation (BOP) [121] and Blender-based automated pipelines [122] for data generation, creation of simulated photorealistic renderings of objects (including ground-truth data) has become the dominant method of dataset generation. This simulated data forms the core of the training data for the latest object pose detection methods. Typical pose detection methods rely on a cropped image input, which is generated by an object detection algorithm; these methods are covered in the next few paragraphs.

Within the area of object pose estimation, many of the best-performing techniques use Convolutional Neural Networks (CNNs) with either some combination of 2 approaches: Pose regression-based, or 2D-to-3D correspondence-based techniques for 6D pose estimation. A notable example of a direct regression-based approach for 6 DoF camera localization is PoseNet [123], which is focused towards localizing monocular camera pose for outdoor scenes. Work by [124] uses the PoseNet architecture for geometric understanding from a scene; this forms the basis for regressing monocular depth estimation from videos. A related approach based on Single-Shot Detection (SSD) was taken by [125], relying on keypoint-based object bounding boxes to infer 3D translation, and scoring the rotation by evaluating a large number of discrete viewpoints from a given scene. However, this has some drawbacks: symmetric objects are considered special cases during data generation. It also provides an approximate pose only, and relies on ICP refinement to achieve final accuracy. Improving on this SSD-based concept, [126] uses only the bounding boxes of shapes for training to obtain accurate estimates without fine-tuning. A regression-based approach accounting for symmetric objects was demonstrated in PoseCNN by [111] for estimating rotation, where the translation is calculated with respect to an

object center obtained through Hough voting. PoseCNN's formulation accounts for symmetric objects through a modified loss function, as well as using semantic mask labels instead of bounding boxes, resulting in improved performance in cluttered scenes.

Correspondence-based methods commonly apply 2 stages: (i) Establishing pixel-to-3D point correspondences, followed by (ii) a method to minimize the object projection difference with respect to the image (usually by PnP). One such approach is PVNet [127], which extends the PoseCNN concept through an approach of detecting keypoints by projecting a vector field over the image pixels. This creates a spatial distribution for each keypoint, which is subsequently used to obtain the 6D object pose using the PnP method. Point based methods have shown success in dealing with the two significant challenges of object symmetry and occlusion which hamper generalizability. Two methods to solve these problems are given in [128], [129], where correspondences are established through latent representations. These methods, as well as approaches based on surface patches [130] focus on removing the need for symmetry exceptions in training data generation, improving generalizability. While the bulk of correspondence methods use the PnP method for the final pose estimation step, there are methods such as [131], which aim to substitute the non-differentiable PnP method with an end-to-end workflow for 6D pose estimation. They simultaneously predict segmentation masks and regress 6D pose indirectly using 2D-3D correspondences, integrating the regression and pixel-to-pose matching methods. Another notable example of this end-to-end approach is shown by Wang et al [112], where dense correspondences and surface patch maps are used as intermediate representations to ultimately regress the 6D pose of the object with state of the art results. This is also a notable example of generating explicit surface-level representations, which have shown good performance in generating 6D pose estimates. These surface-based representations for shapes with defined

canonical orientations has also been demonstrated in [132], where pointwise vector representations are modeled a function of the 3D object manifold. This provides an advantageous representation where partial portions of a surface can be estimated from images, showing potential for fine-grained object feature-based position/orientation estimation. Two recent examples of similar detailed surface representations are shown in Surfemb [133] and ZebraPose [134] with comparable accuracies. Su et al generate a hierarchical groupings for surface points and use this to generate embeddings and 6D pose, whereas Haugaard et al demonstrate a method based on a point key-image query model in [133]. The latter work performs better on texture-less objects commonly found in industrial scenarios, in addition to being more suitable to partial surface matching.

7.3. Scene Graphs and Applications: Related Work

Scene graphs are structured representations of a space of interest, typically derived from sensor data in image or point cloud form (denoted a ‘scene’). They are a natural way of symbolizing extracted semantic information from image space, and are well suited for higher level visual understanding. First proposed as a method for image retrieval [135], detailed scene graphs have shown great potential for a number of tasks such as visual question answering [136], visual reasoning [137], [138] and 3D scene understanding for robot-human interaction [139]. The typical workflow for construction typically involves Feature Extraction (detecting and encoding entities), Contextualization (associating different entities in the scene) followed by Link Prediction and Inference.

Recently, there has been significant interest in scene graphs, driven by their use as topological representations of objects and their interrelations within a given scene. This has resulted in the introduction of 3D Scene Graphs ([140]–[142], [113], [143]) for spatial

perception, representing multiple levels of semantics in 3D world space. Early work by Armeni et al [140] provides a hierarchical framework for organizing multi-level knowledge, along with the earliest real world 3D semantic scene graph dataset containing data ranging from object to building level. This was followed by 3DSSG [142], which provides a large-scale real world scene graph based on RGB-D sequences of the 3RScan indoor scene dataset [144]. These scene graphs are closely related to the evolution of classic topological mapping and visual SLAM to include Semantic representations [145], [146], where prior semantic knowledge of the environment has been demonstrated to improve the estimation of geometric navigation maps. Research by Kim et al [141] demonstrates an integrated method of scene graph construction using a combination of surfel-based dense SLAM [147] and Instance segmentation by Faster-RCNN. Building on these earlier concepts, recent efforts focus on incremental, real-time systems for Semantic SLAM and Scene Graph generation. Rosinol et al [113] provide one such automated method and library (Kimera) for scene graph construction, including aspects of object recognition & localization, human pose estimation for collaborative scenarios in addition to metric-semantic SLAM. Tian et al [148] extend its application to distributed multi-robot metric-semantic mapping. Hughes et al [143] describe a method for hierarchical scene graph construction, geared towards loop closure detection for SLAM. Notably, the focus of these approaches is on large-scale mapping. This means that low-level object recognition is implemented with unsupervised clustering and CAD-based point cloud registration, which are vulnerable to inaccuracies in keypoint and local feature detection. For low-level object manipulation, dense representations in conjunction with have the potential to improve estimation of object state.

There is recent focus of research within robotic task planning, where 3D scene graphs have been used as an intermediate stage for automatic generation of goal states [149]. For instance, approaches geared towards the goal of searching for objects within a given environment use these 3D scene graphs [150]. They apply joint reasoning over the semantic as well as geometric details of objects during the search phase. Similar approaches for learning symbolic operators in task planning use sets of named relations between objects as constraints in solving heuristic task planning guided by shortest-path methods [151]. Motivated by these applications, we base our method of CAD-guided sequential task planning on 3D scene graphs. These are generated by a high fidelity 6D object pose estimation method based on surface embeddings; this provides the advantage of including generating ideal object grasping positions based on the part interactions from assembly CAD files.

7.4. Object Pose Estimation: Methodology

This section describes the methodology for object pose estimation, which is the initial requirement for robotic manipulation. Given an RGB-D image of a scene, the objective of pose estimation is to find the transformation of the objects of interest in the scene with respect to the camera optical reference frame. This transformation in $SE(3)$ Euclidean space is represented by 3D Rotation R (commonly represented by the relative rotation about the X, Y and Z axes of the reference frame) and a 3D Translation T . Together, R and T represent the full 6D pose of the object.

This section provides the model description, the training data generation pipeline and the method of generating pose estimates from the generated 2D-to-3D correspondences. The pose estimation algorithm requires a large amount of ground-truth pose and mask data for training. Taking advantage of the latest realistic image rendering tools, we use synthetic object rendering

to effectively generate low-cost image training data, as well as ground truths for training these algorithms. This is generated using the data generation pipeline & renderer Blenderproc [122]. This makes use of Physically-based Rendering (PBR) to generate highly realistic synthetic images, with the ultimate goal of estimating accurate poses for real-world scenes. The objects are rendered in a variety of sampled poses in 3D space with varied backgrounds. The recorded ground-truth data contains:

- (i) Binary Object Masks
- (ii) Bounding Boxes of the object (in full and occluded versions), and
- (iii) The Transformation matrix of the object relative to the camera.

7.4.1. Pose estimation from 2D-to-3D Correspondences with Surface Embeddings

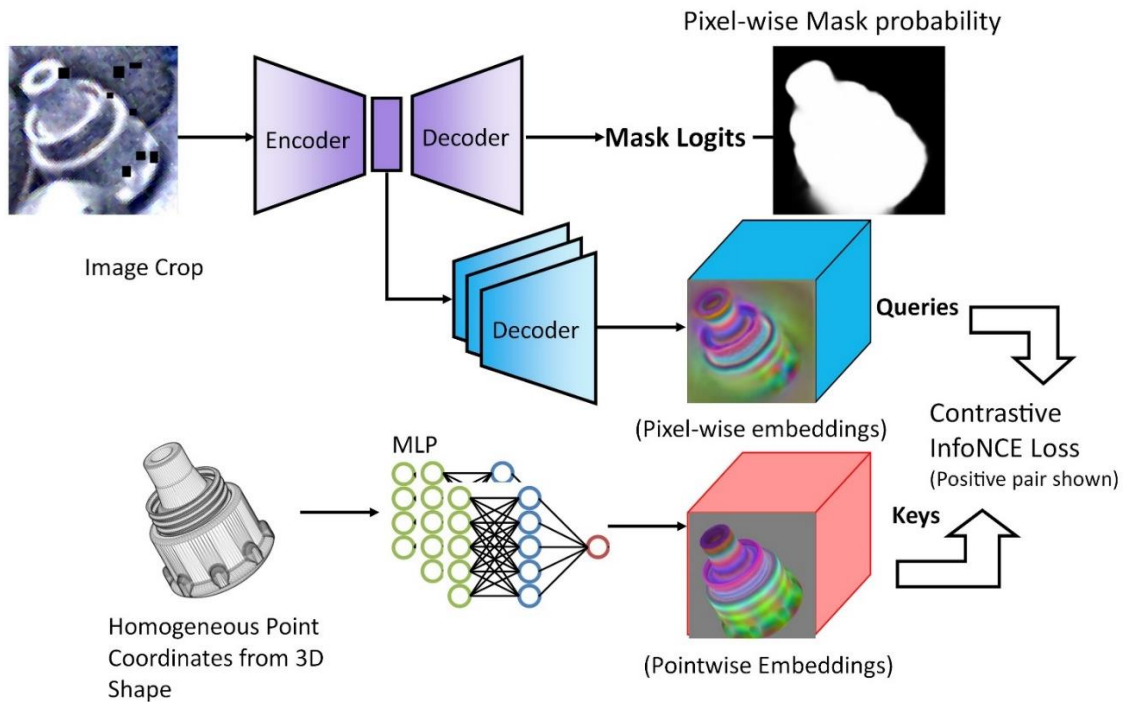


Figure 11: Architecture for generating Surface Embeddings

The implementation of the 6D Pose estimation algorithm is based on the work on Surface Embeddings by Haugaard et al [133]. The model is trained in contrastive manner to create

correspondences, using positive and negative pairs of image-point data. Contrastive learning has become a powerful method for learning representations from of paired data; in this case, the aim is to learn paired correspondences between images of objects and their corresponding points in 3D space.

Fig. 11 shows the network structure which consists of 2 main branches, each generating representations for a different type of input: (i) Image Crops and (ii) Point Coordinates of the 3D model respectively. The first branch is an Encoder-Decoder type CNN which generates representations of color image crops (generated by an object detection algorithm), denoted ‘queries’ (q). These are pixel-wise embeddings corresponding to the object coordinates within the image. This branch also predicts the predicted binary mask of the object in the image, which ensures that only the relevant pixels are encoded. The mask portion of this branch is trained with the Binary Cross Entropy loss as given in eqn. 6. Here, y_n is the ground truth label and x_n the predicted score of this label, for an image coordinate n .

$$L_M = \{l_1, l_2, \dots, l_n\}; l_n = -[y_n \cdot \log x_n + (1 - y_n) \cdot \log (1 - x_n)] \quad (6)$$

The second branch computes the representations of surface points, which are denoted as ‘keys’ (k). The queries from the first branch and keys from the second are then projected into a shared embedding space; the distribution of surface embeddings is then given by the softmax of the dot product between queries and keys. This is represented by the Noise Contrastive Estimation (NCE) loss for contrastive learning tasks, given in eqn. 7. For 2 sets of points S and U sampled uniformly from an object surface and the pixel-mask respectively, the InfoNCE loss [152] is given by:

$$L_E = -\frac{1}{|U|} \sum_{u \in U} \log \frac{\exp(q_u^T k_u)}{\sum_{c_i \in S \cup C_u} \exp(q_u^T k_i)} \quad (7)$$

where q_u is the query value at image coordinate u and k_u is the key value of an object coordinate present at u , given by the homogeneous 3D points at a location u . k_i is the key value for any one of a sampled set of contrastive points. (These are generated by sampling all points that are out-of-view.) The final loss is a sum of the mask loss and the embedding loss, given as $L_T = L_M + L_E$.

The correspondences thus generated are used to gather 6D pose estimates based on the Perspective-n-Point method. The pose estimate is obtained by solving the following equation for the transformation matrix $[R | T]$: $sp_c = K[R | T]p_w$, where p_w is the homogeneous world point, p_c the image point (with scale factor s), T the translation vector and R the rotation matrix. K is the matrix representing the camera's intrinsic parameters.

7.5. Training and Evaluation

7.5.1. Dataset

The method is trained and evaluated on the T-Less dataset, consisting of 30 unique objects. Fig. 12 shows some example parts from the dataset. Based on the available CAD models, different scenes consisting of a combination of parts are constructed within Blender. Subsequently for each scene, Blenderproc's PBR rendering pipeline is used to capture images from 25 distinct camera positions. The dataset consists of 1200 such individually rendered scenes with ground truth labels, resulting in a total of 30000 images. The algorithm is trained with a subset of 10000 images (400 unique scenes) for data efficiency. A number of augmentations are implemented in the training pipeline to account for this, including Gaussian blurring and Color jitter for the images, and Point coordinate noise. These also help in dealing with sensor artifacts during inference.

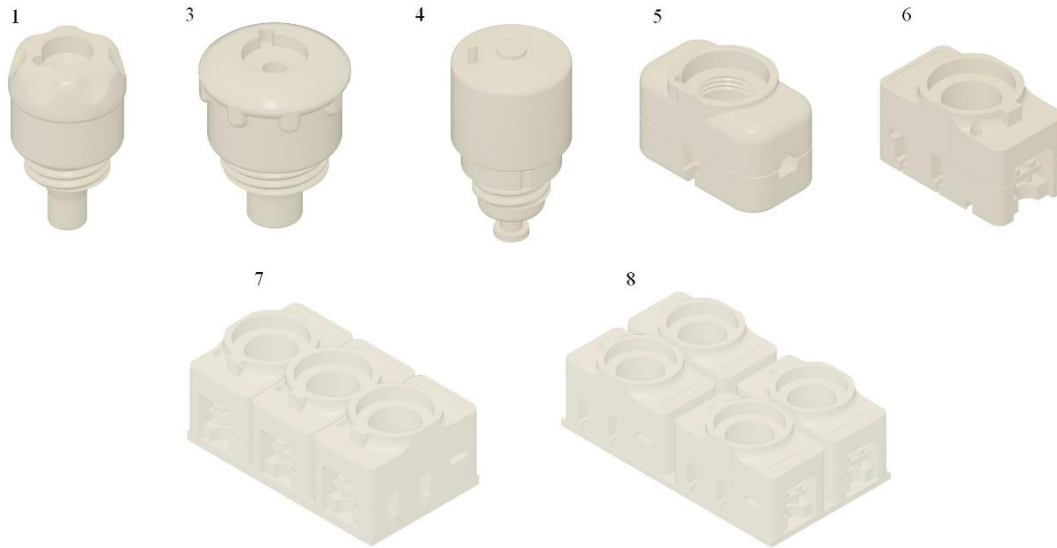


Figure 12: Some example parts from the T-Less Dataset. Part 7 and 8 are assemblies of Part 6.

The overall training process consists of 2 stages: an object detection algorithm to detect and classify objects from the full image, followed by training of the surface embedding algorithm on the individual image crops of the object. Following sections describe the training process.

7.5.2. Object Detection with YOLO V8

The object detection method is used to obtain image crops of interest from a given scene for the inference step, which are then used as input to the surface embedding algorithm. We use the state-of-the-art method for object detection, YOLO V8 [153]. It is based on the highly influential You Only Look Once (YOLO) algorithm, where object localization (in the form of bounding box in image space) and classification are handled in a single step within the network. The high speed and accurate performance of this class of object detection methods means that they have been used in a variety of applications, including manufacturing. The YOLO V8 method predicts the center of a given object directly, instead of basing the object detection on rectangular stencils of pre-defined shape, known as “anchors”. In addition, it makes use of a

variety of augmentations including “mosaic” type augmentations, where training images are overlaid on top of each other to help the model improve detection in occluded scenarios. Fig. 13 shows the training and testing losses for the T-Less dataset, over 250 epochs of training.

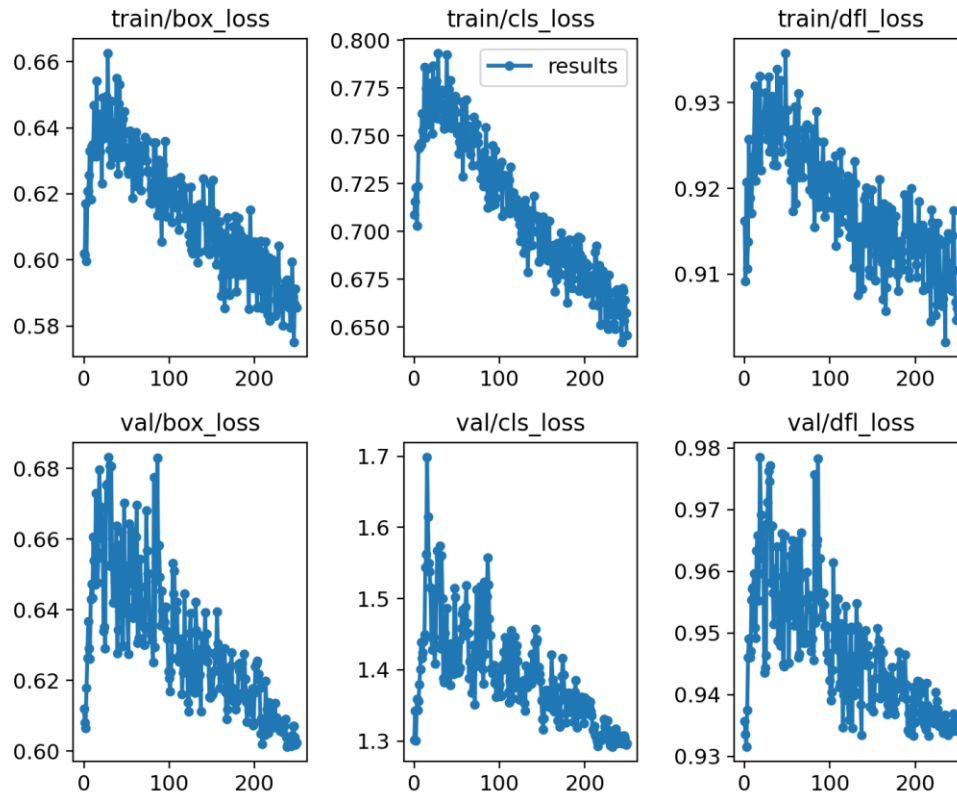


Figure 13: Train and Validation Losses: YOLO V8 for T-Less dataset

Fig. 14 shows the classification performance of the YOLO algorithm on the T-Less dataset through a normalized confusion matrix, comparing true labels with the predicted labels.

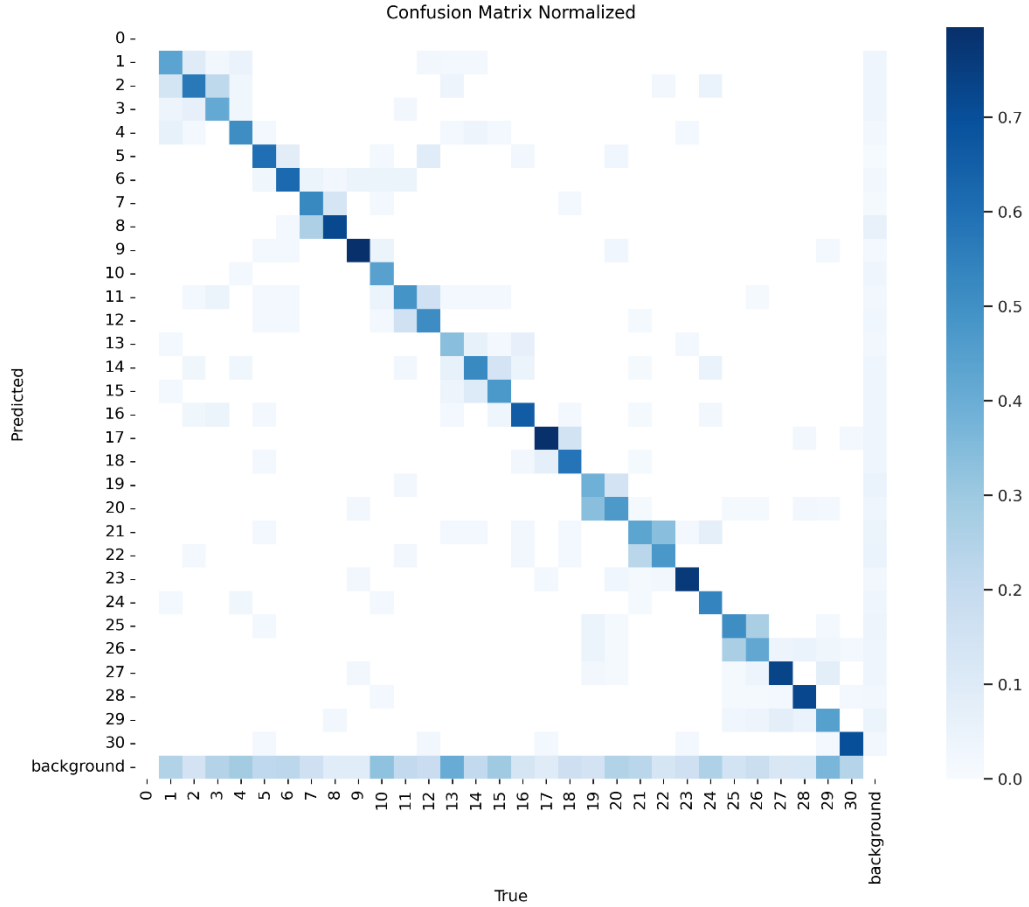


Figure 14: Confusion Matrix for the T-Less Dataset

The performance of the algorithm for object bounding box detection is evaluated using the Intersection-over-Union (IoU) metric, also referred to as the Jaccard similarity coefficient.

For a ground truth bounding box A and predicted bounding box B, IoU is given by: $\frac{A \cap B}{A \cup B}$. This value is calculated for a number of thresholds in the range 0.50 to 0.95, and is then summed up

through the Mean Average Precision metric (mAP). (Average precision $AP =$

$$\frac{\text{True Positives}}{\text{No. of True Predictions}})$$

The mAP over all the classes of the T-Less dataset is 0.548. Table 3 shows the class-wise mAP values.

Table 3: Class-wise mAP values for the T-Less Dataset

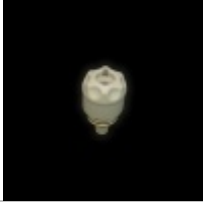

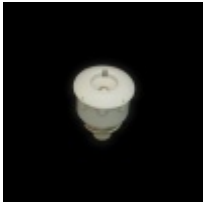



Object	Class	No. of Instances	mAP @ 0.50	mAP @ 0.50:0.95
	1	48	0.517	0.449
	2	61	0.644	0.552
	3	41	0.469	0.411
	4	59	0.614	0.522
	5	60	0.662	0.616
	6	58	0.62	0.547

Table 3 (continued)






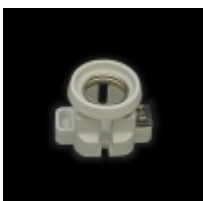
	7	42	0.667	0.566
	8	43	0.732	0.649
	9	44	0.835	0.703
	10	61	0.606	0.467
	11	43	0.559	0.445
	12	43	0.665	0.565

Table 3 (continued)

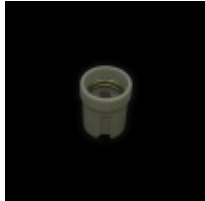
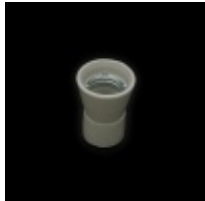




	13	47	0.405	0.343
	14	48	0.492	0.443
	15	57	0.568	0.501
	16	44	0.675	0.618
	17	53	0.855	0.808
	18	46	0.718	0.632

Table 3 (continued)

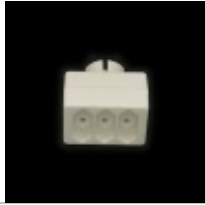
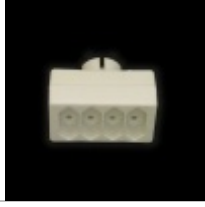










	19	41	0.52	0.443
	20	68	0.521	0.485
	21	65	0.48	0.434
	22	44	0.537	0.473
	23	60	0.819	0.7
	24	39	0.612	0.546
	25	67	0.517	0.451

Table 3 (continued)

	26	71	0.506	0.441
	27	49	0.778	0.728
	28	40	0.815	0.725
	29	60	0.55	0.501
	30	50	0.739	0.674

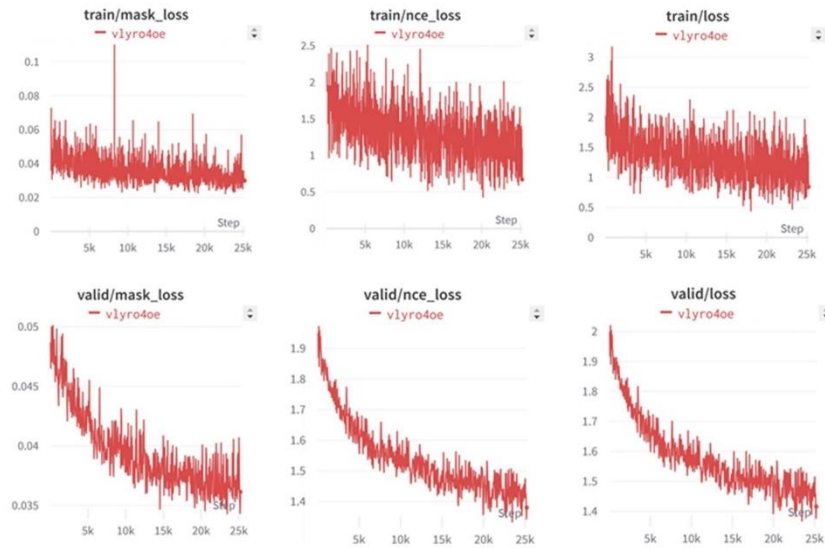
7.5.3. Surface Embeddings & Pose Estimation

The learning process for this second step is implemented in Pytorch, trained for 500,000 steps with the Adam optimizer; the CNN Encoder-Decoder and the MLP are trained with learning rates of 1×10^{-4} and 3×10^{-5} respectively. The 2D-3D correspondences obtained after

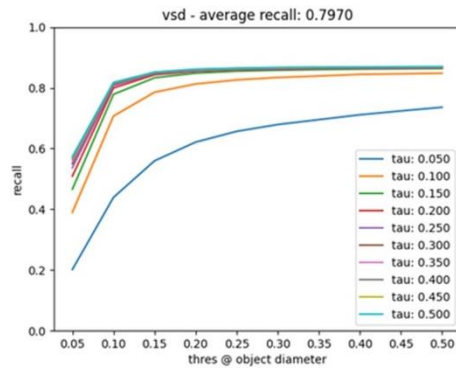
training this algorithm are subsequently used to compute 6D pose. The pose estimates are evaluated based on the Visible Surface Discrepancy (VSD) error [154] given by:

$$(e_{VSD} = \text{avg}_{p \in \hat{V} \cup \bar{V}} \{0\} \rightarrow p \in \hat{V} \cup \bar{V} \wedge |\hat{D}(p) - \bar{D}(p)| < \tau) \wedge (e_{VSD} = \text{avg}_{p \in \hat{V} \cup \bar{V}} \{1\} \rightarrow \text{else}) \quad (8)$$

Here, \hat{D} and \bar{D} are distance maps obtained by rendering the object model in estimated pose \hat{P} and ground-truth pose \bar{P} respectively. (τ is a misalignment tolerance given based on a percentage of the object diameter.) Previously, the common evaluation metrics used were: (i) Translational + Rotational error and (ii) Average Distance of Model Points (predicted point positions vs. ground truth). However, the VSD metric has been shown to be invariant to ambiguities in indistinguishable poses, for instance, in equivalent views of symmetric objects. Fig. 15 (a) show the loss curves for the train and validation sets during the training process.



(a) Loss Curves for Training and Validation



(b)



(c)

Figure 15: Performance on the T-Less Dataset: (a) Train and Validation set losses for the T-Less Dataset (b) Pose error (b) Prediction on the Test Dataset. Objects are superimposed on image in their predicted pose.

An estimated pose is considered to be correct, if the value of the VSD error is less than a given threshold θ_e . The Average Recall of the VSD error rates AR_{VSD} is calculated based for multiple combinations of τ and threshold θ_e , as shown in Fig. 15(b). The Average Recall for VSD over all values of τ and θ_e is 0.7970, indicating robust performance. Examples of estimated pose on real-world scenes are shown in Fig. 15(c) and Fig. 16.

CHAPTER 8: SCENE GRAPHS FOR ROBOTIC MANIPULATION

8.1. Robot Programming and High-level Task Sequences

Flexibility in robotic manipulation requires adaptation not only to a variety of scenarios and different objects through perception, but also to variable task and manipulation sequences. This often requires time-intensive re-programming of the robotic systems, which leads to increased costs and inefficiency. The reason for this is the fact that robotic programs contain not only the position and action/path sequences required for manipulation, but also intrinsically contains knowledge of the robotic system and its surroundings, expected object locations, and the encoded semantic knowledge of the human programmer. Programming robots to perform challenging tasks requires significant expertise, and is a challenge for non-experts users of robots in the fields of manufacturing.

Improvements in automation technology means that robots are often used alongside humans in unstructured environments. Such applications provide improvements in energy and time-intensive manipulation tasks, leading to substantial improvements in efficiency and safety of users. While these systems are driven by sensors which ensure safety of workers, successful deployment hinges on the ability for high-level programming of tasks, and not on position-based rigid control. This leads to a limited versatility in human-robot collaboration tasks, or increase pressure on the humans to adapt to the robot itself.

Generating sequences of high-level tasks has recently been implemented based on the Programming-by-Demonstration paradigm, where non-experts are able to program robots based on procedural demonstration of low-level robot trajectories in robot space. Alternately, kinesthetic position-based programming uses perception data to create manipulation plans.

The next generation of improvements in task-level robotic programming requires high-level task planning, which abstracts away the semantic knowledge of the human user from the low-level program. The use of perception data for this purpose requires the linking of semantic knowledge of the task with spatial knowledge of the workspace, which can enable Decision-making at the micro-level (eg. obstacle avoidance) and macro-level (object-level task planning) for automated systems. This chapter describes a method to generate “scene graphs” for robotic workspaces, where ubiquitous CAD data from manufacturing environments is leveraged to encode semantic knowledge of object relationships in the form of a graph. This is subsequently linked to the result of perception data inference, specifically the object detection and pose results within the workspace.

8.2. Scene Graph Generation: Methodology

We consider the task of sequential robotic manipulation as related to rearrangement and assembly tasks. By extracting the data contained in product designs in the Model-based Definition (MBD) paradigm, sequential task plans can be generated directly from CAD and implemented in the real world.

The semantic scene graph is defined by a set of nodes N with attributes A , containing edge triplets $G = (N, E)$. The scene graph has three main types of nodes:

- (i) *world_node*: representing a fixed frame of reference with respect to the robot
- (ii) *part_world*: representing the 3D object in the real world
- (iii) *part_cad*: representing the 3D CAD model

The nodes are the graphical embodiment of a given object’s canonical frame. These nodes contain a set of ‘affordances’, which are a property representing the possible actions that can be performed using this node; these can be based on the state of a given scene. These graph

nodes also contain any associated properties of a node such as its class/ID, material, color or any physical properties.

Three types of edges are created to represent the spatial and hierarchical relations between the nodes:

- (i) *child_of*: representing the hierarchical sequence of parts to be manipulated
- (ii) *current_link*: representing the current state of transformation between two parts
- (iii) *goal_link*: representing the final state of one object with respect to another
- (iv) *instance_of*: linking a real object to a scene object

These edges contain attributes describing the rotations and translations between the nodes, either current or ideal.

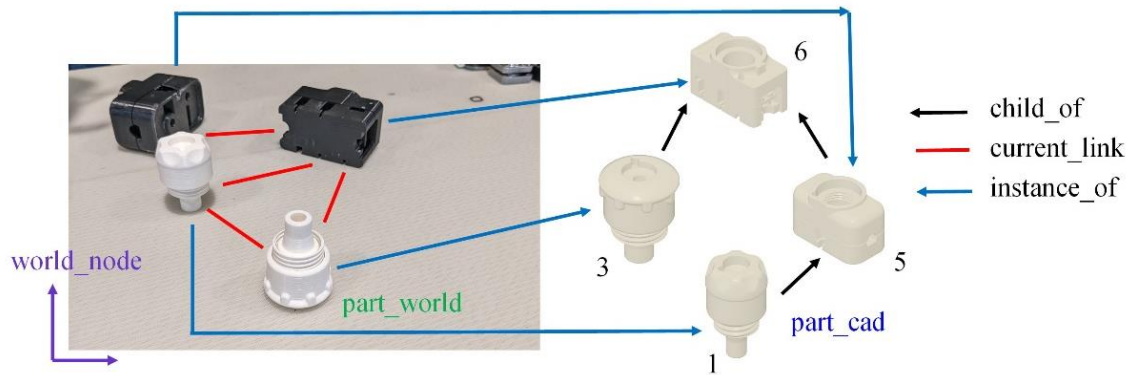


Figure 17: Example of a scene graph for an Assembly Task

Node data is obtained directly from the associated 3D designs, where the canonical base co-ordinate frames represent the nodes *part_world* and *part_cad*. This facilitates the creation of the *instance_of* edges, where corresponding *part_world* and *part_cad* nodes are linked together. The *world_node* is a frame of reference within the 3D space, typically the robot base axes.

The edge data *current_link* is obtained through the pose recognition algorithm. Given *part_world* nodes A and B, the pose recognition algorithm estimates the 6D transformation with

respect to nodes W and C of type *world_node*, representing the robot base frame (W) and the camera optical frame (C) respectively.

Consider a generic transformation T_X^Y representing the transformation between two nodes X and Y, the transform between the two *part_world* nodes A and B is given by the equation 9. The edge *current_link(A, B)* is thus represented by the transformation T_A^B .

$$T_A^B = T_A^C T_C^W T_W^C T_C^B \quad (9)$$

There are two possible methods of obtaining edge data of type *goal_link*:

- (i) Declarative programming by users, where users specify where and how each object must be placed with respect to the *world_node* W. The other relative object transformations are obtained by chaining the transforms with respect to *world_node* type nodes, as shown in eqn. (9).
- (ii) The use of obtained ground-truth transformations between each of the given objects, specified within the Assembly CAD designs in the STEP files. This allows for the specification of hierarchical relationships along with the relative transformations.

Other attributes of the nodes are extracted from CAD designs, such as the type of fit between the two parts, and the type of features being assembled. These are included as node properties in the graph, and are treated as affordances, representing specific actions that can be performed by the relevant object represented by the node. The following section explores the second approaches, and evaluation of the two manipulation tasks.

8.3. Task Sequence Generation for Robotic Assembly

We consider a simple bottom-up assembly task involving 4 parts from the T-Less dataset as described earlier. The final assembly requires Parts 5 and 6 to be placed side-by-side. Part 3 is to be inserted into Part 6, and finally Part 1 is assembled with Part 5. The MBD data obtained from STEP AP242 format files contain a number of assembly constraints; in this case, we focus on the following: Fixed Constituent Assembly (fixing a ‘base’ part within the assembly) and Parallel Assembly Constraint with Dimensions (defining distance between parallel features). Based on an assembly obtained from CAD software in addition to a hierarchy defined by the user, the scene graph is created with the 4 types of relationships, as shown in Fig. 17.

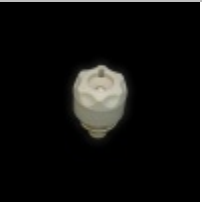



The part sequence is defined by the assembly hierarchy constraint of the *part_cad* nodes, as represented by the edge *child_of*. These nodes form a directed acyclic graph representing a series of tasks with no loops. The order of operations is as follows: starting from the “base” node (which has no parent), to the final node which has no children. The geometric constraints extracted from CAD are used as the target to be achieved. Connecting the results of the perception system (which return the object ID and its pose of the *part_world*) with their relevant *part_cad* nodes through the *instance_of* edge provides a pipeline to plan the assembly task. The overall manipulation task is modeled as follows:

- (i) Grasp the base part from the top and manipulate it to ensure Fixed Assembly rotation constraint is satisfied; place it in the same position.
- (ii) Grasp the first child node using its detected 6D pose.
- (iii) Rotate it to its goal orientation relative to its parent part.
- (iv) Perform a translation operation to minimize the distance between the object’s current position in 3D space to the goal position.

- (v) Repeat till the end of the tree.

Implementing such a sequential task for grasping using a subset of objects from the T-Less Dataset (30 trials for each object, for grasping and manipulation to goal position), the results are shown in Table 4. This shows the percentage of successful grasping and goal manipulation, where an object is to be grasped along its canonical Z Axis and manipulated to a goal position successfully.

Table 4: Object Grasping Success Percentages

Object Image and ID		Percentage of Successful Grasp-and-Move
	Object 2	83% success
	Object 3	90% success
	Object 12	67% success
	Object 6	30% success

We implement an assembly process using a UR5e robot with a Proportional controller operating in Cartesian space. Through initial trial and error, we set gains of 0.05 for translation, and 1.50 for rotation. The result of an ideal assembly process is shown in Fig. 18.



Figure 18: Final goal position for the Assembly Task

8.4. Discussion

The proposed approach is a flexible approach for automated reasoning in robotic systems, and can be modified to suit a variety of tasks in the manufacturing domain. There are only two prior requirements for human intervention: CAD data for the Objects of Interest, and a Hand-Eye calibration between the camera and robot gripper. However, there are some disadvantages of the pose recognition approach used:

- (i) The method is data-heavy. Though the simulated data does not require annotations, data generation does take a significant amount of time.
- (ii) The architecture used for pose recognition is multi-step and complex. While it shows high accuracy in real-world scenarios, the raw output of the Surface Embedding network is not interpretable; it requires an additional step to convert it to 6D pose results.

- (iii) Adaptation to new objects requires re-training of the network, which may be time consuming in high variety manufacturing.

Future work must address these concerns; the adaptability of the algorithm can be improved by modifying it to a class-level or few-shot recognition algorithm. These are open research areas, with a large scope for future improvement. The types of relationships provided by this method continues to be limited to spatial data; inferring more complex relationships from scenes such as object features can provide a greater depth of information and improve the reasoning systems. Another approach to creating more relationships is to integrate the scene graph with existing prior language models and/or domain-specific Knowledge Graphs, further improving generalizability and creating a new language-based task programming framework for manufacturing automation.

CHAPTER 9: CONCLUSION

In this dissertation, a novel method of constructing Product Design Knowledge Graphs (KG) is presented, with 3D product designs at the core, to enable intelligent design & automation recommendations. In addition, the use of 3D design data to create semantic links for reasoning between real-world spatial data and CAD data is demonstrated, based on a scene graph construction method driven by object pose recognition. The contributions are summed up as follows:

- (i) We demonstrate a Hybrid Knowledge Graph construction methodology where Top-Down relations are constructed using hierarchical assembly-part relations obtained from technical descriptions of designs. Subsequently, an efficient technique for Bottom-Up construction of Part Similarity relations is demonstrated, by applying Neural-Network generated Shape Vectors and Global Weight Thresholding to create a sparse yet representative graph.
- (ii) Using our dataset collected from openly available sources, we demonstrate the effectiveness of a KG constructed by this method for recommendation of similar shapes. This is implemented using a combination of Unsupervised Graph Community Detection and Approximate Nearest Neighbor search on the knowledge graph.
- (iii) Three potential applications of this method are illustrated: (a) Rule-Based retrieval of potentially associated components, (b) Similarity Detection for Assemblies, and (c) Contextually relevant design parameter recommendation using Collaborative Filtering. Similar approaches can be implemented for multi-modal search in enterprise databases, combining both 3D components and textual data obtained from prior design-associated tasks. The addition of further product lifecycle data obtained from diverse enterprise data

sources have the potential to enable multi-domain decision making. Future work aims to improve the 3D-driven KG recommendation systems in design and manufacturing automation, particularly in autonomous robotic assembly.

- (iv) Using Product Design data, a method for semantic reasoning over spatial perception data is demonstrated, as applied to autonomous robotic manipulation. This Scene Graph method is based on the recognition of object pose and semantic links created between CAD and real-world objects. Its use is demonstrated for sequential task programming for autonomous robotic manipulation. Future work aims to develop a few-shot method for simultaneously creating object representations and semantic scene understanding.

REFERENCES

- [1] D. Thomas, “The model based enterprise:: a literature review of costs and benefits for discrete manufacturing,” National Institute of Standards and Technology, Gaithersburg, MD, NIST AMS 100-26, Jul. 2019. doi: 10.6028/NIST.AMS.100-26.
- [2] M. Hodkiewicz, J. W. Klüwer, C. Woods, T. Smoker, and E. Low, “An ontology for reasoning over engineering textual data stored in FMEA spreadsheet tables,” *Computers in Industry*, vol. 131, p. 103496, Oct. 2021, doi: 10.1016/j.compind.2021.103496.
- [3] A. Huet, R. Piquié, P. Véron, A. Mallet, and F. Segonds, “CACDA: A knowledge graph for a context-aware cognitive design assistant,” *Computers in Industry*, vol. 125, p. 103377, Feb. 2021, doi: 10.1016/j.compind.2020.103377.
- [4] B. Zhou, J. Bao, Y. Liu, and D. Song, “BA-IKG: BiLSTM Embedded ALBERT for Industrial Knowledge Graph Generation and Reuse,” in *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, Warwick, United Kingdom: IEEE, Jul. 2020, pp. 63–69. doi: 10.1109/INDIN45582.2020.9442198.
- [5] T. Vernica, A. Hanke, and W. Bernstein, “Leveraging standard geospatial representations for industrial augmented reality,” National Institute of Standards and Technology, Gaithersburg, MD, NIST AMS 100-29, Apr. 2020. doi: 10.6028/NIST.AMS.100-29.
- [6] J. Szarazi and C. Bock, “Machine-Readable Physics to Improve Collaboration and Process Management for Design Simulation,” National Institute of Standards and Technology, Gaithersburg, MD, NIST AMS 100-29, Apr. 2020. doi: 10.6028/NIST.AMS.100-29.
- [7] R. Geissbauer, J. Wunderlin, S. Schrauf, J. H. Krause, J.-T. Morr, and A. Odenkirchen, “Digital Product Development 2025.” PricewaterhouseCoopers GmbH, Mar. 2019.

[Online]. Available: <https://www.pwc.de/en/digitale-transformation/digital-product-development-2025.html>

- [8] S. K. Sim and A. H. Duffy, “A foundation for machine learning in design,” *AI EDAM*, vol. 12, no. 2, pp. 193–209, 1998.
- [9] S. K. Chandrasegaran *et al.*, “The evolution, challenges, and future of knowledge representation in product design systems,” *Computer-aided design*, vol. 45, no. 2, pp. 204–228, 2013.
- [10] A. K. Goel, S. Vattam, B. Wiltgen, and M. Helms, “Cognitive, collaborative, conceptual and creative—four characteristics of the next generation of knowledge-based CAD systems: a study in biologically inspired design,” *Computer-Aided Design*, vol. 44, no. 10, pp. 879–900, 2012.
- [11] K. Ding, F. T. S. Chan, X. Zhang, G. Zhou, and F. Zhang, “Defining a Digital Twin-based Cyber-Physical Production System for autonomous manufacturing in smart shop floors,” *International Journal of Production Research*, vol. 57, no. 20, pp. 6315–6334, Oct. 2019, doi: 10.1080/00207543.2019.1566661.
- [12] A. G. Bharadwaj and B. Starly, “Knowledge graph construction for product designs from large CAD model repositories,” *Advanced Engineering Informatics*, vol. 53, p. 101680, 2022.
- [13] I. Gibson, D. Rosen, B. Stucker, and M. Khorasani, “Design for Additive Manufacturing,” in *Additive Manufacturing Technologies*, Cham: Springer International Publishing, 2021, pp. 555–607. doi: 10.1007/978-3-030-56127-7_19.
- [14] X. L. Dong *et al.*, “AutoKnow: Self-Driving Knowledge Collection for Products of Thousands of Types,” in *Proceedings of the 26th ACM SIGKDD International*

Conference on Knowledge Discovery & Data Mining, Virtual Event CA USA: ACM, Aug. 2020, pp. 2724–2734. doi: 10.1145/3394486.3403323.

[15] C. Meilicke, M. W. Chekol, D. Ruffinelli, and H. Stuckenschmidt, “Anytime Bottom-Up Rule Learning for Knowledge Graph Completion.,” in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 3137–3143.

[16] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, “COMET: Commonsense Transformers for Automatic Knowledge Graph Construction,” *arXiv:1906.05317 [cs]*, Jun. 2019, Accessed: Sep. 25, 2021. [Online]. Available: <http://arxiv.org/abs/1906.05317>

[17] P. Z. Chądzyński and V. McQueen, “The MBE Vision needs MBD to reach outside its current MCAD and PMI comfort zone,” *Model-Based Enterprise Summit (MBE 2020)*, p. 70, 2020.

[18] X. Li, M. Lyu, Z. Wang, C.-H. Chen, and P. Zheng, “Exploiting knowledge graphs in industrial products and services: A survey of key aspects, challenges, and future perspectives,” *Computers in Industry*, vol. 129, p. 103449, Aug. 2021, doi: 10.1016/j.compind.2021.103449.

[19] Bharadwaj, Akshay Ganesh and Starly, Binil, “FabWave Product Design Knowledge Graph (FPD-KG).” Zenodo, Feb. 14, 2022. doi: 10.5281/ZENODO.6083697.

[20] B. Starly, Bharadwaj, Akshay Ganesh, and A. Angrish, “Fabwave-3D Part Repository and Product Design Knowledge Graph,” *Fabwave- 3D Part Repository*. <https://www.dimelab.org/knowledge-graphs> (accessed May 18, 2022).

- [21] T. P. Tanon, G. Weikum, and F. Suchanek, “Yago 4: A reasonable knowledge base,” in *The Semantic Web. ESWC 2020. Lecture Notes in Computer Science*, Springer, 2020, pp. 583–596. doi: https://doi.org/10.1007/978-3-030-49461-2_34.
- [22] D. Vrandečić and M. Krötzsch, “Wikidata: a free collaborative knowledgebase,” *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.
- [23] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” *Advances in neural information processing systems*, vol. 26, 2013.
- [24] J. Yuan *et al.*, “Constructing biomedical domain-specific knowledge graph with minimum supervision,” *Knowl Inf Syst*, vol. 62, no. 1, pp. 317–336, Jan. 2020, doi: 10.1007/s10115-019-01351-4.
- [25] J. Dörpinghaus, A. Stefan, B. Schultz, and M. Jacobs, “Towards context in large scale biomedical knowledge graphs,” *arXiv:2001.08392 [cs]*, Jan. 2020, Accessed: Jun. 22, 2021. [Online]. Available: <http://arxiv.org/abs/2001.08392>
- [26] D. Mrdjenovich *et al.*, “propnet: A Knowledge Graph for Materials Science,” *Matter*, vol. 2, no. 2, pp. 464–480, Feb. 2020, doi: 10.1016/j.matt.2019.11.013.
- [27] E. B. Myklebust, E. Jimenez-Ruiz, J. Chen, R. Wolf, and K. E. Tollefsen, “Knowledge Graph Embedding for Ecotoxicological Effect Prediction,” *arXiv:1907.01328 [cs]*, vol. 11779, pp. 490–506, 2019, doi: 10.1007/978-3-030-30796-7_30.
- [28] Z. Wu *et al.*, “3d shapenets: A deep representation for volumetric shapes,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

- [29] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 922–928. doi: 10.1109/IROS.2015.7353481.
- [30] C. R. Qi, H. Su, K. Mo, and L. 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*, 2017, pp. 652–660.
- [31] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.
- [32] A. Kanazaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5010–5019.
- [33] A. Angrish, A. Bharadwaj, and B. Starly, "MVCNN++: Computer-Aided Design Model Shape Classification and Retrieval Using Multi-View Convolutional Neural Networks," *Journal of Computing and Information Science in Engineering*, vol. 21, no. 1, p. 011001, Feb. 2021, doi: 10.1115/1.4047486.
- [34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
- [35] Z. Zhang, P. Jaiswal, and R. Rai, "Featurenet: Machining feature recognition based on 3d convolution neural network," *Computer-Aided Design*, vol. 101, pp. 12–22, 2018.

- [36] Y. Song, F. He, Y. Duan, Y. Liang, and X. Yan, "A Kernel Correlation-Based Approach to Adaptively Acquire Local Features for Learning 3D Point Clouds," *Computer-Aided Design*, vol. 146, p. 103196, 2022.
- [37] E. Clark, A. Vincent, J. N. Kutz, and S. L. Brunton, "Bracketing brackets with bras and kets," *Journal of Manufacturing Systems*, vol. 58, pp. 384–391, Jan. 2021, doi: 10.1016/j.jmsy.2020.12.018.
- [38] S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani, "Developing an engineering shape benchmark for CAD models," *Computer-Aided Design*, vol. 38, no. 9, pp. 939–953, Sep. 2006, doi: 10.1016/j.cad.2006.06.007.
- [39] S. Koch *et al.*, "Abc: A big cad model dataset for geometric deep learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9601–9611.
- [40] S. Kim, H. Chi, X. Hu, Q. Huang, and K. Ramani, "A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks," in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, Springer, 2020, pp. 175–191.
- [41] V. Ferrero, K. Hassani, D. Grandi, and B. DuPont, "Classifying Component Function in Product Assemblies with Graph Neural Networks," *arXiv:2107.07042 [cs]*, Jul. 2021, Accessed: Sep. 14, 2021. [Online]. Available: <http://arxiv.org/abs/2107.07042>
- [42] A. Bharadwaj, Y. Xu, A. Angrish, Y. Chen, and B. Starly, "Development of a pilot manufacturing cyberinfrastructure with an information rich mechanical CAD 3D model repository," in *International Manufacturing Science and Engineering Conference*, American Society of Mechanical Engineers, 2019, p. V001T02A035.

- [43] K.-Y. Kim, D. G. Manley, and H. Yang, "Ontology-based assembly design and information sharing for collaborative product development," *Computer-Aided Design*, vol. 38, no. 12, pp. 1233–1250, Dec. 2006, doi: 10.1016/j.cad.2006.08.004.
- [44] L. Qiao, Y. Qie, Z. Zhu, Y. Zhu, U. K. uz Zaman, and N. Anwer, "An ontology-based modelling and reasoning framework for assembly sequence planning," *Int J Adv Manuf Technol*, vol. 94, no. 9–12, pp. 4187–4197, Feb. 2018, doi: 10.1007/s00170-017-1077-4.
- [45] Z. Ming, G. Sharma, J. K. Allen, and F. Mistree, "An Ontology for Representing Knowledge of Decision Interactions in Decision-Based Design," *Computers in Industry*, vol. 114, p. 103145, Jan. 2020, doi: 10.1016/j.compind.2019.103145.
- [46] N. Wan, R. Mo, L. Liu, and J. Li, "New methods of creating MBD process model: On the basis of machining knowledge," *Computers in Industry*, vol. 65, no. 4, pp. 537–549, May 2014, doi: 10.1016/j.compind.2013.12.005.
- [47] Y. Zhong, C. Jiang, Y. Qin, G. Yang, M. Huang, and X. Luo, "Automatically generating assembly sequences with an ontology-based approach," *AA*, vol. 40, no. 2, pp. 319–334, Nov. 2019, doi: 10.1108/AA-12-2018-0271.
- [48] R. Wang *et al.*, "Ontology-Based Representation of Meta-Design in Designing Decision Workflows," *Journal of Computing and Information Science in Engineering*, vol. 19, no. 1, p. 011003, Mar. 2019, doi: 10.1115/1.4041474.
- [49] F. Boussuge *et al.*, "Capturing simulation intent in an ontology: CAD and CAE integration application," *Journal of Engineering Design*, vol. 30, no. 10–12, pp. 688–725, Dec. 2019, doi: 10.1080/09544828.2019.1630806.
- [50] H. Cheong and A. Butscher, "Physics-based simulation ontology: an ontology to support modelling and reuse of data for physics-based simulation," *Journal of*

Engineering Design, vol. 30, no. 10–12, pp. 655–687, Dec. 2019, doi:
10.1080/09544828.2019.1644301.

[51] Z. Wu *et al.*, “Semantic hyper-graph-based knowledge representation architecture for complex product development,” *Computers in Industry*, vol. 100, pp. 43–56, Sep. 2018, doi: 10.1016/j.compind.2018.04.008.

[52] X. Li, S. Zhang, R. Huang, B. Huang, C. Xu, and B. Kuang, “Structured modeling of heterogeneous CAM model based on process knowledge graph,” *Int J Adv Manuf Technol*, vol. 96, no. 9–12, pp. 4173–4193, Jun. 2018, doi: 10.1007/s00170-018-1862-8.

[53] Z. Han, R. Mo, H. Yang, and L. Hao, “CAD assembly model retrieval based on multi-source semantics information and weighted bipartite graph,” *Computers in Industry*, vol. 96, pp. 54–65, Apr. 2018, doi: 10.1016/j.compind.2018.01.003.

[54] K. Lupinetti, F. Giannini, M. Monti, and J.-P. Pernot, “Content-based multi-criteria similarity assessment of CAD assembly models,” *Computers in Industry*, vol. 112, p. 103111, Nov. 2019, doi: 10.1016/j.compind.2019.07.001.

[55] C. Zhang, G. Zhou, Q. Lu, and F. Chang, “Graph-based knowledge reuse for supporting knowledge-driven decision-making in new product development,” *International Journal of Production Research*, vol. 55, no. 23, pp. 7187–7203, Dec. 2017, doi:
10.1080/00207543.2017.1351643.

[56] T. D. Hedberg Jr, M. Bajaj, and J. A. Camelio, “Using graphs to link data across the product lifecycle for enabling smart manufacturing digital threads,” *Journal of computing and information science in engineering*, vol. 20, no. 1, p. 011011, 2020.

[57] J. Martinez-Gil *et al.*, “General Model for Tracking Manufacturing Products Using Graph Databases,” in *Data-Driven Process Discovery and Analysis*, P. Ceravolo,

M. van Keulen, and M. T. Gómez-López, Eds., in *Lecture Notes in Business Information Processing*, vol. 379. Cham: Springer International Publishing, 2020, pp. 86–100. doi: 10.1007/978-3-030-46633-6_5.

[58] W. Nie, Y. Wang, D. Song, and W. Li, “3D Model Retrieval Based on a 3D Shape Knowledge Graph,” *IEEE Access*, vol. 8, pp. 142632–142641, 2020, doi: 10.1109/ACCESS.2020.3013595.

[59] B. Zhou, J. Bao, Z. Chen, and Y. Liu, “KGAssembly: Knowledge graph-driven assembly process generation and evaluation for complex components,” *International Journal of Computer Integrated Manufacturing*, pp. 1–21, Mar. 2021, doi: 10.1080/0951192X.2021.1891572.

[60] G. Buchgeher, D. Gabauer, J. Martinez-Gil, and L. Ehrlinger, “Knowledge Graphs in Manufacturing and Production: A Systematic Literature Review,” *arXiv:2012.09049 [cs]*, Dec. 2020, Accessed: Jun. 22, 2021. [Online]. Available: <http://arxiv.org/abs/2012.09049>

[61] R. Piquié, P. Véron, F. Segonds, and T. Zynda, “A Property Graph Data Model for a Context-Aware Design Assistant,” in *Product Lifecycle Management in the Digital Twin Era*, C. Fortin, L. Rivest, A. Bernard, and A. Bouras, Eds., Cham: Springer International Publishing, 2019, pp. 181–190.

[62] A. J. Donkers, D. Yang, and N. H. Baken, “Linked Data for Smart Homes: Comparing RDF and Labeled Property Graphs,” in *Proceedings of the 8th Linked Data in Architecture and Construction Workshop*, Dublin, Ireland: CEUR-WS. org, Jun. 2020, pp. 23–36.

[63] A. Huet, F. Segonds, R. Piquie, P. Veron, J. Guegan, and A. Mallet, “Context-aware cognitive design assistant: Implementation and study of design rules

recommendations,” *Advanced Engineering Informatics*, vol. 50, p. 101419, Oct. 2021, doi: 10.1016/j.aei.2021.101419.

[64] J. Hao, L. Zhao, J. Milisavljevic-Syed, and Z. Ming, “Integrating and navigating engineering design decision-related knowledge using decision knowledge graph,” *Advanced Engineering Informatics*, vol. 50, p. 101366, Oct. 2021, doi: 10.1016/j.aei.2021.101366.

[65] Z. Zhao, S.-K. Han, and I.-M. So, “Architecture of knowledge graph construction techniques,” *International Journal of Pure and Applied Mathematics*, vol. 118, no. 19, pp. 1869–1883, 2018.

[66] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, “GVCNN: Group-view convolutional neural networks for 3D shape recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 264–272.

[67] X. Yan, L. G. S. Jeub, A. Flammini, F. Radicchi, and S. Fortunato, “Weight Thresholding on Complex Networks,” *arXiv:1806.07479 [physics]*, Oct. 2018, doi: 10.1103/PhysRevE.98.042304.

[68] C. R. Buchanan *et al.*, “The effect of network thresholding and weighting on structural brain networks in the UK Biobank,” *NeuroImage*, vol. 211, p. 116443, May 2020, doi: 10.1016/j.neuroimage.2019.116443.

[69] O. Civier, R. E. Smith, C.-H. Yeh, A. Connelly, and F. Calamante, “Is removal of weak connections necessary for graph-theoretical analysis of dense weighted structural connectomes from diffusion MRI?,” *NeuroImage*, vol. 194, pp. 68–81, Jul. 2019, doi: 10.1016/j.neuroimage.2019.02.039.

- [70] B. Starly, A. Bharadwaj, and A. Angrish, “FabWave CAD Repository Categorized Part Classes - CAD 1 through 15 Classes (Part 1/3).” Dec. 04, 2019. doi: 10.13140/RG.2.2.31167.87201.
- [71] J. Alstott, E. Bullmore, and D. Plenz, “Powerlaw: a Python package for analysis of heavy-tailed distributions,” *PLoS ONE*, vol. 9, no. 1, p. e85777, Jan. 2014, doi: 10.1371/journal.pone.0085777.
- [72] Z. Yang and S. Dong, “HAGERec: Hierarchical Attention Graph Convolutional Network Incorporating Knowledge Graph for Explainable Recommendation,” *Knowledge-Based Systems*, vol. 204, p. 106194, Sep. 2020, doi: 10.1016/j.knosys.2020.106194.
- [73] Z. Zhang, F. Zhuang, M. Qu, F. Lin, and Q. He, “Knowledge graph embedding with hierarchical relation structure,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3198–3207.
- [74] C. Carusi and G. Bianchi, “Scientific community detection via bipartite scholar/journal graph co-clustering,” *Journal of Informetrics*, vol. 13, no. 1, pp. 354–386, Feb. 2019, doi: 10.1016/j.joi.2019.01.004.
- [75] M. D. L. Tosi and J. C. dos Reis, “SciKGraph: A knowledge graph approach to structure a scientific field,” *Journal of Informetrics*, vol. 15, no. 1, p. 101109, Feb. 2021, doi: 10.1016/j.joi.2020.101109.
- [76] R. Guidotti and M. Coscia, “On the Equivalence Between Community Discovery and Clustering,” in *Smart Objects and Technologies for Social Good*, B. Guidi, L. Ricci, C. Calafate, O. Gaggi, and J. Marquez-Barja, Eds., in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 233. Cham: Springer International Publishing, 2018, pp. 342–352. doi: 10.1007/978-3-319-76111-4_34.

- [77] M. A. Javed, M. S. Younis, S. Latif, J. Qadir, and A. Baig, “Community detection in networks: A multidisciplinary review,” *Journal of Network and Computer Applications*, vol. 108, pp. 87–111, Apr. 2018, doi: 10.1016/j.jnca.2018.02.011.
- [78] Z. Yang, R. Algesheimer, and C. J. Tessone, “A Comparative Analysis of Community Detection Algorithms on Artificial Networks,” *Sci Rep*, vol. 6, no. 1, p. 30750, Aug. 2016, doi: 10.1038/srep30750.
- [79] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.
- [80] V. Traag, L. Waltman, and N. J. van Eck, “From Louvain to Leiden: guaranteeing well-connected communities,” *Sci Rep*, vol. 9, no. 1, p. 5233, Dec. 2019, doi: 10.1038/s41598-019-41695-z.
- [81] V. A. Traag, “Faster unfolding of communities: Speeding up the Louvain algorithm,” *Physical Review E*, vol. 92, no. 3, p. 032801, 2015.
- [82] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” *Physical review E*, vol. 76, no. 3, p. 036106, 2007.
- [83] P. Pons and M. Latapy, “Computing communities in large networks using random walks,” presented at the J. Graph Algorithms Appl, Citeseer, 2006.
- [84] M. Rosvall and C. T. Bergstrom, “Maps of random walks on complex networks reveal community structure,” *Proceedings of the national academy of sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

- [85] A. Landherr, B. Friedl, and J. Heidemann, “A critical review of centrality measures in social networks,” *Business & Information Systems Engineering*, vol. 2, no. 6, pp. 371–385, 2010.
- [86] F. A. Rodrigues, “Network Centrality: An Introduction,” in *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*, E. E. N. Macau, Ed., in *Nonlinear Systems and Complexity*, vol. 22. Cham: Springer International Publishing, 2019, pp. 177–196. doi: 10.1007/978-3-319-78512-7_10.
- [87] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: Bringing order to the web.,” Stanford InfoLab, 1999.
- [88] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, “Approximate nearest neighbor algorithm based on navigable small world graphs,” *Information Systems*, vol. 45, pp. 61–68, Sep. 2014, doi: 10.1016/j.is.2013.10.006.
- [89] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.
- [90] M. Aumüller, E. Bernhardsson, and A. Faithfull, “ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms,” in *Proceedings of the 2017 International Conference on Similarity Search and Applications*, Springer, 2017, pp. 34–49.
- [91] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly, “Metrics for Community Analysis: A Survey,” *arXiv:1604.03512 [physics]*, Apr. 2016, Accessed: Jul. 02, 2021. [Online]. Available: <http://arxiv.org/abs/1604.03512>

- [92] A. Hagberg, P. Swart, and D. S Chult, “Exploring network structure, dynamics, and function using NetworkX,” Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
- [93] G. Rossetti, L. Milli, and R. Cazabet, “CDLIB: a python library to extract, compare and evaluate communities from complex networks,” *Appl Netw Sci*, vol. 4, no. 1, p. 52, Dec. 2019, doi: 10.1007/s41109-019-0165-9.
- [94] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
- [95] T. Bonald, N. de Lara, Q. Lutz, and B. Charpentier, “Scikit-network: Graph Analysis in Python.,” *J. Mach. Learn. Res.*, vol. 21, pp. 185–1, 2020.
- [96] S. Spolaor, C. Fuchs, P. Cazzaniga, U. Kaymak, D. Besozzi, and M. S. Nobile, “Simpful: a user-friendly Python library for fuzzy logic,” *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 1687–1698, 2020.
- [97] S. Raschka, “MLxtend: Providing machine learning and data science utilities and extensions to Python’s scientific computing stack,” *Journal of open source software*, vol. 3, no. 24, p. 638, 2018.
- [98] A. Angrish, B. Craver, and B. Starly, “‘FabSearch’: A 3D CAD Model-Based Search Engine for Sourcing Manufacturing Services,” *Journal of Computing and Information Science in Engineering*, vol. 19, no. 4, p. 041006, Dec. 2019, doi: 10.1115/1.4043211.
- [99] V. J. Ferrero, “Data-Driven Environmentally Sustainable Product Design: A Shift Toward Increased use of Sustainable Design Activities in the Early Design Phase,” Doctoral Dissertation, Oregon State University, 2021.

- [100] A. S. Deshmukh, A. G. Banerjee, S. K. Gupta, and R. D. Sriram, "Content-based assembly search: A step towards assembly reuse," *Computer-Aided Design*, vol. 40, no. 2, pp. 244–261, Feb. 2008, doi: 10.1016/j.cad.2007.10.012.
- [101] P. Wang, Y. Li, J. Zhang, and J. Yu, "An assembly retrieval approach based on shape distributions and Earth Mover's Distance," *Int J Adv Manuf Technol*, vol. 86, no. 9–12, pp. 2635–2651, Oct. 2016, doi: 10.1007/s00170-016-8368-z.
- [102] M. A. Robinson, "How design engineers spend their time: Job content and task satisfaction," *Design Studies*, vol. 33, no. 4, pp. 391–425, Jul. 2012, doi: 10.1016/j.destud.2012.03.002.
- [103] E. Pourjavad and R. V. Mayorga, "A comparative study and measuring performance of manufacturing systems with Mamdani fuzzy inference system," *J Intell Manuf*, vol. 30, no. 3, pp. 1085–1097, Mar. 2019, doi: 10.1007/s10845-017-1307-5.
- [104] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, Jan. 1975, doi: 10.1016/S0020-7373(75)80002-2.
- [105] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009, doi: 10.1155/2009/421425.
- [106] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, vol. 8, no. 1, pp. 53–87, 2004.
- [107] H. Ko, P. Witherell, N. Y. Ndiaye, and Y. Lu, "Machine Learning based Continuous Knowledge Engineering for Additive Manufacturing," in *2019 IEEE 15th*

International Conference on Automation Science and Engineering (CASE), Aug. 2019, pp. 648–654. doi: 10.1109/COASE.2019.8843316.

[108] A. Saeedi, E. Peukert, and E. Rahm, “Incremental Multi-source Entity Resolution for Knowledge Graph Completion,” in *The Semantic Web*, A. Harth, S. Kirrane, A.-C. Ngonga Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase, and M. Cochez, Eds., in *Lecture Notes in Computer Science*, vol. 12123. Cham: Springer International Publishing, 2020, pp. 393–408. doi: 10.1007/978-3-030-49461-2_23.

[109] S. Fakhraei, D. Sridhar, J. Pujara, and L. Getoor, “Adaptive neighborhood graph construction for inference in multi-relational networks,” *arXiv preprint arXiv:1607.00474*, 2016.

[110] K. Teru, E. Denis, and W. Hamilton, “Inductive relation prediction by subgraph reasoning,” in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, 2020, pp. 9448–9457.

[111] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes.” *arXiv*, May 26, 2018. Accessed: Apr. 17, 2023. [Online]. Available: <http://arxiv.org/abs/1711.00199>

[112] G. Wang, F. Manhardt, F. Tombari, and X. Ji, “GDR-Net: Geometry-Guided Direct Regression Network for Monocular 6D Object Pose Estimation.” *arXiv*, Mar. 09, 2021. Accessed: Apr. 30, 2023. [Online]. Available: <http://arxiv.org/abs/2102.12145>

[113] A. Rosinol *et al.*, “Kimera: From SLAM to spatial perception with 3D dynamic scene graphs,” *The International Journal of Robotics Research*, vol. 40, no. 12–14, pp. 1510–1546, Dec. 2021, doi: 10.1177/02783649211056674.

- [114] C. Agia *et al.*, “Taskography: Evaluating robot task planning over large 3D scene graphs,” presented at the Conference on Robot Learning, PMLR, 2022, pp. 46–58.
- [115] S. Hinterstoisser *et al.*, “Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes,” presented at the Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5–9, 2012, Revised Selected Papers, Part I 11, Springer, 2013, pp. 548–562.
- [116] S. Tyree *et al.*, “6-DoF pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark,” presented at the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2022, pp. 13081–13088.
- [117] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, “A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1179–1185, 2016.
- [118] C. Mitash *et al.*, “ARMBench: An object-centric benchmark dataset for robotic manipulation,” *arXiv preprint arXiv:2303.16382*, 2023.
- [119] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, “T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects,” presented at the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2017, pp. 880–888.
- [120] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger, “Introducing mvtec itodd-a dataset for 3d object recognition in industry,” presented at the Proceedings of the IEEE international conference on computer vision workshops, 2017, pp. 2200–2208.

- [121] T. Hodan *et al.*, “Bop: Benchmark for 6d object pose estimation,” presented at the Proceedings of the European conference on computer vision (ECCV), 2018, pp. 19–34.
- [122] M. Denninger *et al.*, “BlenderProc2: A Procedural Pipeline for PhotorealisticRendering,” *JOSS*, vol. 8, no. 82, p. 4901, Feb. 2023, doi: 10.21105/joss.04901.
- [123] A. Kendall, M. Grimes, and R. Cipolla, “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile: IEEE, Dec. 2015, pp. 2938–2946. doi: 10.1109/ICCV.2015.336.
- [124] Z. Yin and J. Shi, “GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 1983–1992. doi: 10.1109/CVPR.2018.00212.
- [125] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, “SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again,” presented at the Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1521–1529. Accessed: Apr. 17, 2023. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2017/html/Kehl_SSD-6D_Making_RGB-Based_ICCV_2017_paper.html
- [126] B. Tekin, S. N. Sinha, and P. Fua, “Real-Time Seamless Single Shot 6D Object Pose Prediction,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA: IEEE, Jun. 2018, pp. 292–301. doi: 10.1109/CVPR.2018.00038.

- [127] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, “PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 4556–4565. doi: 10.1109/CVPR.2019.00469.
- [128] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, “Implicit 3D Orientation Learning for 6D Object Detection from RGB Images.” arXiv, Jul. 17, 2019. Accessed: Apr. 30, 2023. [Online]. Available: <http://arxiv.org/abs/1902.01275>
- [129] Z. Li, G. Wang, and X. Ji, “CDPN: Coordinates-Based Disentangled Pose Network for Real-Time RGB-Based 6-DoF Object Pose Estimation,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 7677–7686. doi: 10.1109/ICCV.2019.00777.
- [130] T. Hodan, D. Barath, and J. Matas, “Epos: Estimating 6d pose of objects with symmetries,” presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11703–11712.
- [131] Y. Hu, P. Fua, W. Wang, and M. Salzmann, “Single-stage 6d object pose estimation,” presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 2930–2939. doi: <https://doi.org/10.48550/arXiv.1911.08324>.
- [132] N. Neverova, D. Novotny, V. Khalidov, M. Szafraniec, P. Labatut, and A. Vedaldi, “Continuous Surface Embeddings.” arXiv, Nov. 24, 2020. Accessed: Apr. 30, 2023. [Online]. Available: <http://arxiv.org/abs/2011.12438>
- [133] R. L. Haugaard and A. G. Buch, “SurfEmb: Dense and Continuous Correspondence Distributions for Object Pose Estimation with Learnt Surface Embeddings.”

arXiv, Apr. 04, 2022. Accessed: Apr. 30, 2023. [Online]. Available:

<http://arxiv.org/abs/2111.13489>

[134] Y. Su *et al.*, “ZebraPose: Coarse to Fine Surface Encoding for 6DoF

Object Pose Estimation.” arXiv, Mar. 29, 2022. Accessed: Apr. 30, 2023. [Online]. Available:

<http://arxiv.org/abs/2203.09418>

[135] J. Johnson *et al.*, “Image retrieval using scene graphs,” presented at the

Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3668–3678.

[136] V. Damodaran *et al.*, “Understanding the Role of Scene Graphs in Visual

Question Answering.” arXiv, Jan. 16, 2021. Accessed: Apr. 28, 2023. [Online]. Available:

<http://arxiv.org/abs/2101.05479>

[137] K. Tang, H. Zhang, B. Wu, W. Luo, and W. Liu, “Learning to Compose

Dynamic Tree Structures for Visual Contexts.” arXiv, Dec. 05, 2018. Accessed: Apr. 28, 2023.

[Online]. Available: <http://arxiv.org/abs/1812.01880>

[138] J. Shi, H. Zhang, and J. Li, “Explainable and explicit visual reasoning over

scene graphs,” presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 8376–8384.

[139] S. Li, P. Zheng, Z. Wang, J. Fan, and L. Wang, “Dynamic Scene Graph

for Mutual-Cognition Generation in Proactive Human-Robot Collaboration,” *Procedia CIRP*,

vol. 107, pp. 943–948, 2022, doi: 10.1016/j.procir.2022.05.089.

[140] I. Armeni *et al.*, “3d scene graph: A structure for unified semantics, 3d

space, and camera,” presented at the Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 5664–5673.

- [141] U.-H. Kim, J.-M. Park, T.-J. Song, and J.-H. Kim, “3-D scene graph: A sparse and semantic representation of physical environments for intelligent agents,” *IEEE transactions on cybernetics*, vol. 50, no. 12, pp. 4921–4933, 2019.
- [142] J. Wald, H. Dhano, N. Navab, and F. Tombari, “Learning 3d semantic scene graphs from 3d indoor reconstructions,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3961–3970.
- [143] N. Hughes, Y. Chang, and L. Carlone, “Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization,” *arXiv preprint arXiv:2201.13360*, 2022.
- [144] J. Wald, A. Avetisyan, N. Navab, F. Tombari, and M. Nießner, “Rio: 3d object instance re-localization in changing indoor environments,” presented at the Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7658–7667.
- [145] C. Cadena *et al.*, “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016, doi: 10.1109/TRO.2016.2624754.
- [146] T. Lai, “A Review on Visual-SLAM: Advancements from Geometric Modelling to Learning-based Semantic Scene Understanding.” arXiv, Sep. 12, 2022. Accessed: Apr. 28, 2023. [Online]. Available: <http://arxiv.org/abs/2209.05222>
- [147] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, “ElasticFusion: Real-time dense SLAM and light source estimation,” *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.
- [148] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone, “Kimera-Multi: Robust, Distributed, Dense Metric-Semantic SLAM for Multi-Robot

Systems,” *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2022–2038, Aug. 2022, doi:
10.1109/TRO.2021.3137751.

[149] Z. Jiao, Y. Niu, Z. Zhang, S.-C. Zhu, Y. Zhu, and H. Liu, “Sequential Manipulation Planning on Scene Graph,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan: IEEE, Oct. 2022, pp. 8203–8210. doi:
10.1109/IROS47612.2022.9981735.

[150] A. Kurenkov, R. Martin-Martin, J. Ichnowski, K. Goldberg, and S. Savarese, “Semantic and Geometric Modeling with Neural Message Passing in 3D Scene Graphs for Hierarchical Mechanical Search,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi’an, China: IEEE, May 2021, pp. 11227–11233. doi:
10.1109/ICRA48506.2021.9560736.

[151] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Perez, “Learning Symbolic Operators for Task and Motion Planning,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic: IEEE, Sep. 2021, pp. 3182–3189. doi: 10.1109/IROS51168.2021.9635941.

[152] A. van den Oord, Y. Li, and O. Vinyals, “Representation Learning with Contrastive Predictive Coding.” arXiv, Jan. 22, 2019. Accessed: Apr. 30, 2023. [Online]. Available: <http://arxiv.org/abs/1807.03748>

[153] G. Jocher, A. Chaurasia, and J. Qiu, “YOLO by Ultralytics.” Jan. 10, 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>

[154] T. Hodaň, J. Matas, and Š. Obdržálek, “On evaluation of 6D object pose estimation,” presented at the Computer Vision–ECCV 2016 Workshops: Amsterdam, The

Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14, Springer, 2016, pp. 606–619.

APPENDICES

Appendix A: Available Part Annotations

Table 5: Node Properties- Property Names and Classes/Text Descriptions

Description Name	Type	Classes or Text Description
component_type	Single Label	{‘Standard Part’, ‘Non-Standard Part’, ‘Assembly’}
features	Multi Label (x3 unique features)	{‘Holes’, ‘Flat’, ‘Cylinder’, ‘Fillet’, ‘Slot’, ‘Other’, ‘Counterbore’, ‘Chamfer’, ‘Pocket’, ‘Threads’, ‘Pattern’, ‘Sweep’, ‘Freeform’, ‘Loft’}
feature_descriptions	String x3 (describing each of the ‘features’)	Eg: ‘lengthwise semi-circular grooves’ Eg: ‘Two slots on the cover attached to the motor drill’
main_process	Single Label	{‘Joining’, ‘Casting’, ‘Machining’, ‘Forming’, ‘Additive Manufacturing’, ‘Molding’, ‘Other’, ‘Engraving’}
material_category	Single Label	{‘Metal Alloys’, ‘Plastics & Polymers’, ‘Composites’, ‘Ceramics’, ‘Other’}
model_type	Single Label	{‘Solid Model’, ‘Surface Model’, ‘Invalid 3D Model’}
part_category	Single Label	{‘Home Goods’, ‘Industrial Component’, ‘Automobile’, ‘Oil & Gas’, ‘Electronics’, ‘Electrical’, ‘Construction’, ‘Children Toy’, ‘Paper Products’, ‘General Transportation’, ‘Cosmetics & Jewelry’, ‘Wood & Furniture’, ‘Aerospace’, ‘Educational’, ‘Defense’, ‘Medical Device’, ‘Agriculture, Fisheries, Fishing’}

Table 5 (continued)

part_description	String	Eg: 'hexagonal extruded part with two concentric holes one of which is through the part' Eg: 'irregular shaped block with C shaped pocket and two cylinders attached on one end'
precision	Single Label	{'Standard Tolerance', 'High Tolerance', 'Unknown'}
quantity	Single Label	{'Mass Production', 'One Quantity', 'Limited Quantity'}

Appendix B: Class-wise Mean Average Precision (mAP) Values

Table 6: Comparison of HNSW and KD-Tree for retrieval using Mean Average Precision (mAP) at $k = 1, 3, 5$: For each class

Class	HNSW + Exhaustive K-NN			KD-Tree		
	mAP@1	mAP@3	mAP@5	mAP@1	mAP@3	mAP@5
Bearings	1	1	1	1	1	0.9918
Brackets	0.55	0.6583	0.6311	0.8	0.8083	0.8068
Bushing_Damping_Liners	0.3	0.35	0.3772	0.25	0.2833	0.3033
Bushing	0.7	0.7375	0.7421	0.525	0.5583	0.5703
Collets	0.35	0.4	0.3875	0.25	0.3	0.3
Gasket	0.95	0.95	0.9342	0.8	0.8166	0.7758
Grommets	0.8	0.8	0.8	0.75	0.75	0.75
HeadlessScrews	0.55	0.6083	0.6344	0.35	0.3416	0.3502
Hex_Head_Screws	0.85	0.875	0.8879	0.65	0.725	0.7463
Keyway_Shaft	0.5	0.5167	0.5239	0.6	0.5916	0.5902
Machine_Key	0.6	0.6167	0.6152	0.85	0.85	0.85
Nuts	1	1	1	1	1	1
O_Rings	1	1	1	1	1	1
Pipe_Fittings	0.85	0.925	0.9057	0.35	0.5791	0.6096
Pipe_Joints	0.1	0.3208	0.4232	0.15	0.3666	0.4481
Pipes	0.5	0.5583	0.5788	0.5	0.5458	0.5433
Push_Rings	0.9	0.9417	0.8761	0.9	0.9291	0.9422
Retaining_Rings	1	0.9917	0.9902	1	1	1
Rollers	1	1	1	0.9	0.9	0.9

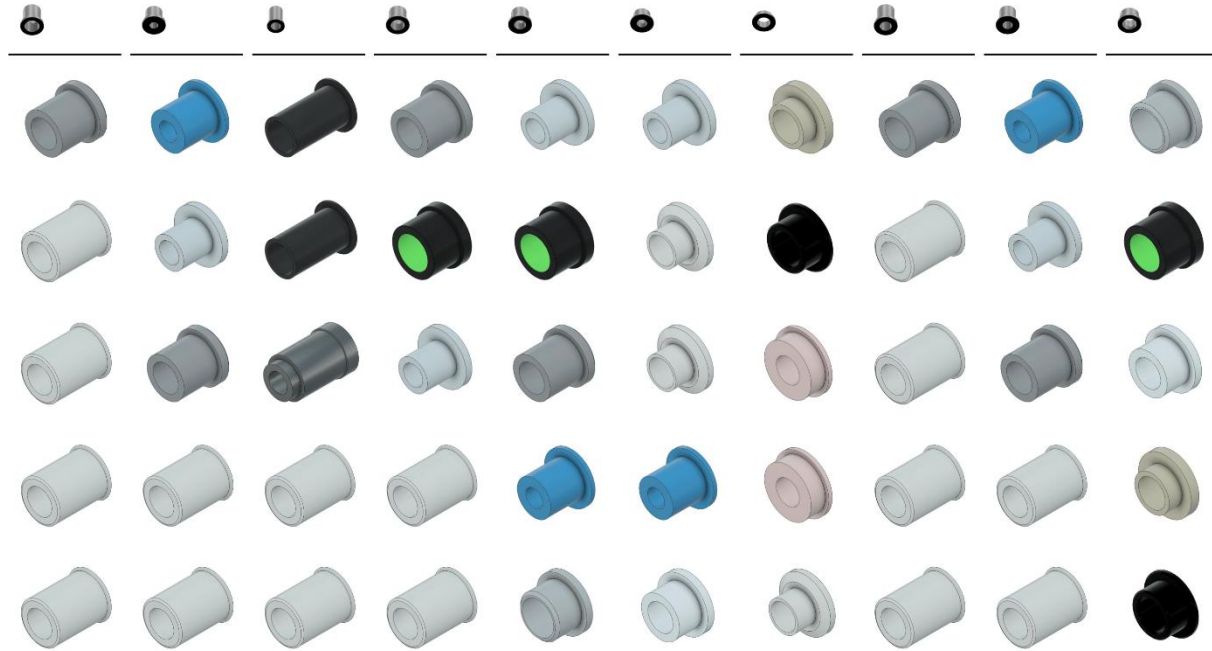
Table 6 (continued)

Rotary_Shaft	0.6	0.6	0.6	0.65	0.65	0.65
Shaft_Collar	1	1	1	0.95	0.95	0.95
Slotted_Flat_Head_Screws	0.1	0.1	0.1	0	0	0.01
Socket_Head_Screws	1	1	1	1	1	1
Springs	0.8	0.7917	0.7902	0.85	0.85	0.8537
Sprockets	0.95	0.9416	0.9255	0.9	0.9083	0.8925
Thumb_Screws	0.05	0.1167	0.1592	0.05	0.0666	0.0891
Unthreaded_Flanges	1	1	1	1	1	1
Washers	0.65	0.6958	0.7078	0.3	0.3	0.3
Socket_Head_Screws	0.95	0.95	0.95	0.95	0.9333	0.9223

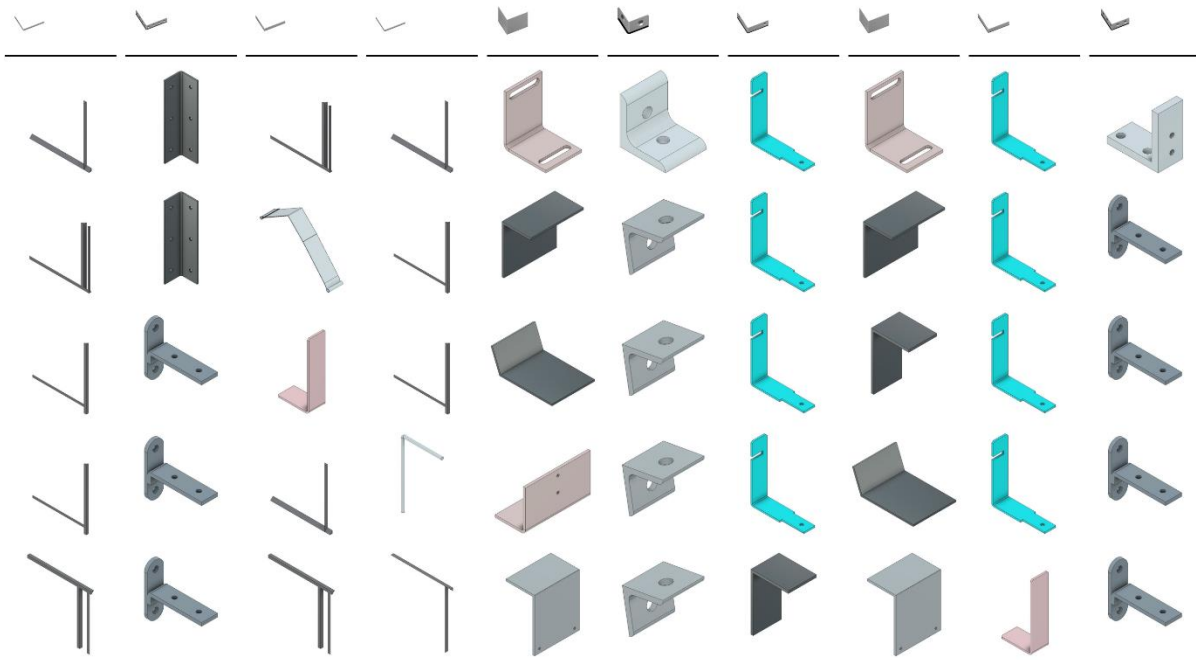
Appendix C: Examples of Retrieved Components for Standard Classes using HNSW+KNN

Method (Bearings, Brackets, Pipes and Springs)

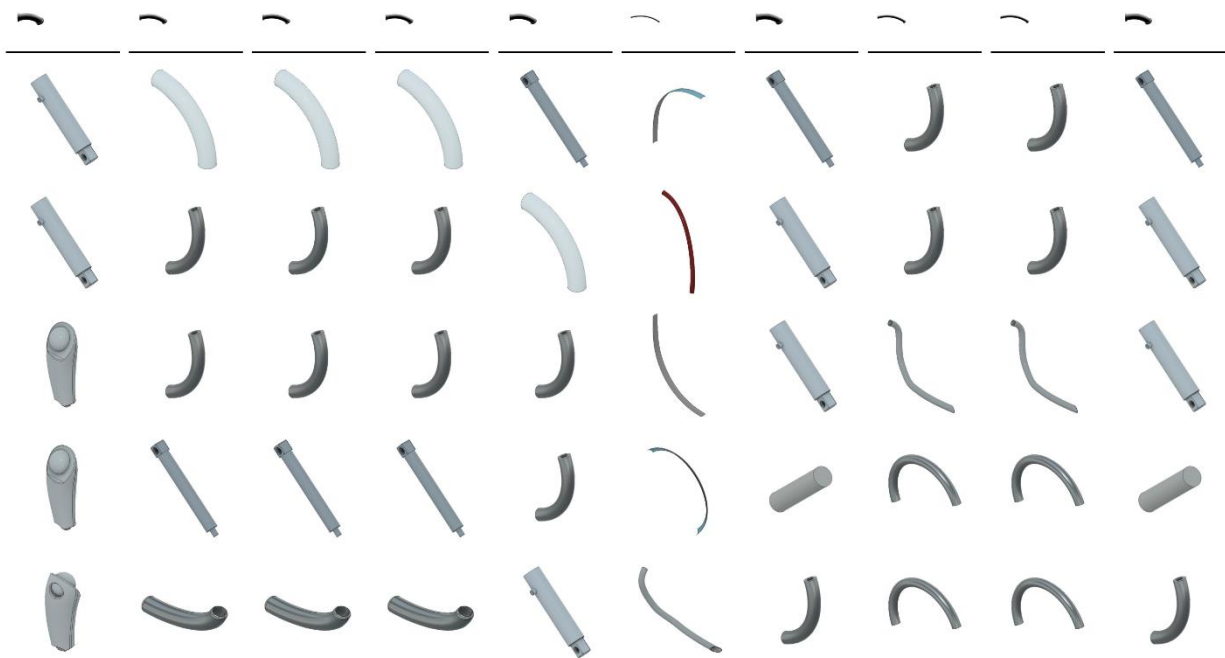
C.1. Bearings: $mAP@5 = 1$ (Top row shows the query parts)



C.2. Brackets: $mAP@5 = 0.63$ (Top row shows the query parts)



C.3. Pipes: $mAP@5 = 0.58$ (Top row shows the query parts)



C.4. Springs: $mAP@5 = 0.79$ (Top row shows the query parts)

