BRUNEL UNIVERSITY

Department Of Computer Science

DOCTORAL THESIS

An Automated Method Mapping Parametric Features Between Computer Aided Design Software

Author:
Toby David BORLAND

Supervisors:
Dr. Andrea CAPILUPPI and
Dr. Stephen SWIFT

Year of Submission: 2019

Declaration of Authorship

Author Publications

Some of the research leading to this thesis has appeared previously in the following publication,

> Borland, T., 2012. Automated mapping between Computer Aided Design program formats. *From the Selected Works of Charles M. Schweik*, p.25. University of Massachusetts - Amherst

Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Andrea Capiluppi for his unremitting pragmatism and bonhomie. Additional thanks must also go to Dr. Stephen Swift for his contributions.
I should also acknowledge the succession of patient employers, friends and partners who I have taxed during the course of this research.

Abstract

Enterprise efficiency is limited by data exchange. A product designer might specify the geometry of a product with a Computer Aided Design program, an engineer might re-use that geometry data to calculate physical properties of the product using a Finite Element Analysis program. These different domains place different requirements on the product representation. Representations of product data required for different tasks is dependent on the vendor software associated with those tasks, sharing data between different vendor programs is limited by incompatibility of the vendor formats used. In the case of Computer Aided Design where the virtual form of an object is modelled, no standard data format captures complete model data. Common data standards transfer model surface geometry without capturing the topological elements from which these geometries are constructed. There are prescriptive data representations to allow these features to be specified in a neutral format, but little incentive for vendors to adopt these schemes. Recent efforts instead focus on identifying similar feature elements between different vendor CAD programs, however this approach relies on onerous manual identification requiring frequent revision.

This research develops methods to automate the task of mapping relationships between different data format representations. Two independent matching techniques identify similar CAD feature functions between heterogeneous programs. Text similarity and object geometry matching techniques are combined to match the data formats associated with CAD programs. An efficient search for matching function parameters is performed using a genetic algorithm that incorporates semantic data matching and geometry data matching. A greedy semantic matching algorithm is developed that compares with the Doc2vec short text matching technique over the API dataset tested. A SVD geometric surface registration technique is developed that requires fewer calculations than an equivalent Iterative Closest Point method.

Synopsis

Transferring the full details of design data between engineering programs is limited to the availability and accuracy of translations between model information. To date, only surface geometry is readily translated using neutral file formats, however modern CAD software uses vendor-specific parametric design features to capture model geometry and design. Translating models that retain these parametric design features is limited by the complexity and expense of mapping the functionality between the sets of native vendor features.
This difficulty of testing features for equivalence and mapping may be mitigated with the assistance of machine searching, testing and verification techniques.

This research develops a geometry matching technique coupled with a semantic matching technique to assist the process of unsupervised matching. Semantic text matching allows the search for potential function matches to be reduced to a fraction of the entire set of feature functions described in a CAD Application Programming Interface. Automated function testing requires that the output of functions can be tested for equivalence. This, in turn, demands that the geometry associated with a function operation can be compared with that of a function candidate match from a separate CAD program. A surface intersection technique is developed and tested to fulfil this requirement. A genetic algorithm incorporates these techniques to demonstrate the proposed automated function matching method.

Glossary of terms

For the sake of brevity, this list does not include any acronym which only appears once in the text next to its definition.

AAG	Attributed Adjacent Graph
AIA	Aerospace Industries Association of America
AM	Application Module
ANSI	American National Standard Institution
AP	Application Protocol
API	Application Programming Interface
AR	Application Resources
ASPI	Assured Product and Support Information
BIM	Business Information Modelling
BoM	Bill of Material
BRL	Ballistics Research Laboratory
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Automated Manufacturing
CAPP	Computer Aided Process Planning
CAx	Computer Aided x (a non-specific member of CAE)
CFD	Computational Fluid Dynamics
CIM	Computer Information Modelling
CML	Chemical Markup Language
CNC	Computer Numerical Control
COM	Component Object Model
COP	Constrained Optimisation Problem
CPD	Common Process Domain
CSG	Constructive Solid Geometry
CSP	Constraint Satisfaction Problem
DEAP	Distributed Evolutionary Algorithms in Python
DIFF	Domain Independent Form Feature
DNA	DeoxyRibonucleic Acid

DoD	Department of Defence (US)
DXF	Drawing Exchange Format
EOL	End Of Life
ERP	Enterprise Resource Planning
EXPRESS	(not an acronym)
FAG	Form Feature Adjacency Graph
FEA	Finite Element Analysis
FMS	Flexible Manufacturing System
FPMP	Function Parameter Matching Problem
GA	Genetic Algorithm
HDF5	Hierarchical Data Format 5
HKS	Heat Kernel Signature
HTML	HyperText Markup Language
IAR	Integrated Application Resource
ICAM	Integrated Computer-Aided Manufacturing
ICP	Iterative Closest Point
ICT	Information Communication Technology
IDEF	ICAM™ Definition Languages
IFC	Industry Foundation Classes
IGES	Initial Graphics Exchange Specification
ISO	International Organization for Standardization
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
NC	Numerical Control
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
NURB	Non Uniform Rational B-spline
OASIS	Organization for the Advancement of Structured Information Standards
OECD	Organisation for Economic Cooperation and Development
OEM	Original Equipment Manufacturer
OMG	Object Management Group
OWL DL	Web Ontology Language Description Logic
OWL LT	Web Ontology Language Lite

PCA	Principle Component Analysis
PDKM	Product Data & Knowledge Management
PDM	Product Data Management
PLM	Product Lifetime Modelling
PMI	Product and Manufacturing Information
PML	Product Modelling Language
RDF	Resource Description Framework
RMS	Reconfigurable Manufacturing Systems
RMSE	Root Mean Square Error
SAT	Boolean Satisfiability Problem
SIFT	Scale Invariant Feature Transform
SME	Small to Medium Enterprise
STEP	Standard for the Exchange of Product model data
SVD	Singular Variable Decomposition
SWRL	Semantic Web Rule Language
TF-IDF	Term Frequency, Inverse Document Frequency
UML	Universal Modeling Language
UPR	Universal Product Representation
VB	Visual Basic
VBA	Visual Basic Automation
XML	eXtensible Markup Language

Notes on fonts

Throughout the text, reference to computer code is printed in `Liberation Mono` font. This may be a reference to the code that accompanies this thesis, or short extracts of code from other referenced sources.

The other font that appears is Liberation Serif which is used in the text of tracts of psuedocode.

Where a term is introduced, it is italicised.

Table of Contents

Index of Figures

Index of Tables

1 Research Overview and Structure

Stating the research question at the outset presents an opportunity to unravel the form that this research has taken in answering it. This is as follows,

> *Given that it is possible to fully or partially recreate parametric CAD models between heterogeneous CAD programs using a mapped sequence of API feature functions, is it then possible to automate the process of mapping a functional equivalence between heterogeneous CAD feature libraries?*

This question must be qualified, the assumptions made explicit and the supporting research identified. This cannot be immediately satisfied in detail within an introduction to this research, but is addressed in the opening chapters of this thesis (Chapters 2, Chapter 3).

However, this research question may be inverted to define a research aim; to automate a task of identifying CAD program API functions that exhibit the same behaviour. This can be characterised as a search problem, where two functions that exhibit a specified degree of similar behaviour satisfy a goal criterion. There are three distinctive research areas that are reflected in the research objectives below. These three separate lines of inquiry may be justified by their individual contribution to the research aim.

Searching for equivalent functions between CAD API requires a comparison of the thousands of individual functions contained in a commercial API (Chapter 5.7, Parsing of API text data), the multiplicity of potential combinations of function parameters create a large combinational search space. A hybrid search strategy is proposed that separates the research into three separate and distinct strands. This hybrid search compares multiple aspects of function description and behaviour to reduce this matching search space. These function characteristics are common to all CAD API functions, namely semantic description and geometric behaviour. They may be introduced here as follows.

Semantic Text Matching

Feature functions are described by a function name, parameter names and descriptive text within API documentation or code libraries (Chapter 4.2, Descriptive text labels used within Application Programming Interfaces). These text descriptions may be compared using existing methods of semantic comparison. There is no published research describing the efficiency of semantic methods used to match CAD API function texts.

The first research strand tests the matching success of a range of methods on text data taken from commercial API documentation. These methods are introduced in Chapter 4 and the results are presented and examined in Chapter 5, (See Figure 1).

Surface Geometry Comparison

The second research strand allows surface geometric models to be compared, returning a numerical measure of similarity. These methods are presented in Chapter Error: Reference source not found and their subsequent evaluation in Chapter 7. This similarity measure can automate the comparison of API function operation as follows. A CAD feature function creates or modifies a geometric entity; if the two geometric entities that result from function operations within separate CAD programs produce a measurably similar shape, then it may be inferred that these two functions are functionally equivalent.

There is no universal algorithm to reliably compare geometry between heterogeneous CAD programs, this claim is elaborated in Chapter 6.5, Point cloud registration techniques. The second strand of research develops and tests a method that returns a geometric comparison between surface geometry models independently of affine transformation. This resultant metric facilitates a machine search for comparable functions and forms the basis of the robust GA parameter search method undertaken in the third strand of the research.

Equivalent Parameter Matching

> The third part to this thesis finds a mapping between function parameters using a Genetic Algorithm, this is a representative strategy applicable to this general class of problem. The method may be extended from finding a mapping between parameters of a pair of functions, to testing for mapping across a set of functions. This strand of research strand implements and tests a robust search function to find function parameters with equivalent behaviour. Rather than comparing a number of different methods over a common data set as is the case for semantic text matching, a relatively simple local search method is used to demonstrate how parameter combinational search space may be reduced to computationally efficient dimensions. The description of function parameters, and heuristics to reduce their search space are described in Chapter 8. This chapter introduces various non-deterministic search methods, the following Chapter 9 uses a Genetic Algorithm to demonstrate the parameter search space reduction heuristics.

1.1 Research Philosophy

The methods in this research have been selected according to a reductive strategy that adopts the most elementary apparent solution and subsequently determines the deficiencies of this method. The rationale behind this is twofold. Firstly, by demonstrating that a relatively simple solution has quantifiable performance gains, such as a basic genetic algorithm used in a combinational search problem, it justifies the value of an exploratory research without the overhead of excessive complexity or test variables. Secondly, it signposts obvious optimisation. For example the concept of a geometry model centroid works well as a basis for a unique model signature independent of rotation, but has limitations as the basis of a feature region search method that are readily resolved with a more complex schema.

From this general introduction to the research, it is possible to specify a more precise definition of the research question and corresponding research aims and objectives.

1.2 Research Objectives

This research examines practical methods to evaluate and test the similarity of features that generate CAD models within different programs. Two manual approaches that allow identification of CAD feature similarity are codified as machine tasks. These methods can be described as the recognition of equivalent CAD model geometry alongside the identification of semantic equivalence in the textual description of CAD features. With this in mind the objectives can be specified with greater clarity:

Objective I: *devise and test an algorithm capable of identifying two equivalent geometrical surfaces independent of scaling, rotation and translation while independent of vendor specific CAD programs.*

Preliminary investigation indicates that there is no method to compare the exact geometric representation of CAD model geometry between heterogeneous CAD program representation. The established method is to translate the surface boundary representation to a neutral common format. This approach is limited by the translation from the native representation to the neutral format and by the constraints of geometric representation within the format. Therefore a means to directly interrogate the geometric similarity of models with heterogeneous CAD model spaces is required.

An algorithm must be able to recognise a similar feature that is at a different orientation or location within a CAD model space. Any shape comparison must be invariant to affine transforms as there is no guarantee of consistent orientation or position within the model space of different CAD programs. If the geometric model representation generated by one CAD feature function can be compared against the model generated by a feature function in a different CAD program, it is possible to validate whether the two feature functions are geometrically equivalent. For an algorithm to make this comparison, it must operate within each CAD program to return geometric properties that can be numerically tested. As unknown function parameters may create objects of different orientation, scale or absolute position, this requires comparative testing methods to be invariant under spatial transforms that preserve relative geometry. An extension of this objective is the ability to quantify a measure of similarity between geometric shapes. The chapters dealing with this objective are shown in Figure 1, within the box titled “Surface Geometry Comparison”, it is also shown how this method is instrumental to the local search techniques to map parameters, shown in the box titled “Genetic Algorithm parameter search”.

Objective II: *devise and test a method capable of identifying the range of geometrical operations normally found within representative commercial CAD programs.*

Any method that can determine a geometric match between surface models within their native CAD environments must function with the full range of permissible shapes encountered within these environments. While the requirements to empirically prove a method for all possible shapes is impractical, it can be shown that complex shapes can be decomposed into a bounded set of simple geometries which can be uniquely identified. A complete geometric matching method should satisfy this requirement. A minimal set of identifiers is described in Chapter 6.25.

Objective III: *determine the applicability of semantic matching methods suited to identification of CAD software API function matches.*

Semantic matching methods are adapted to relatively long documents with comparatively sparse information. Generic schema matching methods use label syntax and structure matching heuristics. This approach is unsuited to the short texts used to describe function operations and parameters within program interface support. There is relatively little research on the effectiveness of semantic matching techniques for mapping API functions, semantic matching algorithms are not optimised for the short, terse phrases within function and parameter names or the descriptive text accompanying functions. A number of promising semantic similarity methods can be tested on a selected set of known API function documentation text matches and compared with conventional document retrieval metrics for a set of known API function matches. Examples of suitable methods include the WordNet corpus and the Word2vec method. Chapter 4 introduces the methods tested, these tests are described in the following Chapter 5. Figure 1 shows the research related to this objective within a box titled “Semantic matching”.

Objective IV: *demonstrate how a measure of surface boundary geometry similarity may be used to map features between heterogeneous CAD programs, where features are defined by interface library routines.*

Parametric CAD features are defined by function operations within the CAD program architecture, parametric features may be conceptual model artefacts such as a “flange” or geometry operations such as a “loft”. These features are specified by both explicit and implicit function parameters which manifesting as a determinate CAD model surface geometry.

While it is straightforward to determine that two functions that produce a similar model geometry are nominally equivalent, it is not a simple task to reverse this process and validate the similarity of two feature functions by a comparison of model geometry. Without a mapping between function parameters, equivalence validation becomes a combinational problem. To address the research question, it should be possible to show how the algorithm described in the first research objective allows an automated mapping of functional equivalence between CAD function libraries. Chapter 8 describes the heuristics and search methods used, this is followed by a demonstration of the techniques in Chapter 9. Figure 1 shows the relevant thesis section in as box marked “Genetic Algorithm parameter search”.

Note that these objectives are referenced again in the final Chapter 10.1, where the methods used to fulfil these objectives are listed, alongside their location within the thesis text.

1.3 Research Methodology

It is instructive to locate the methodology in this research within the framework of an established paradigm. A research methodology is an abstracted strategy of selecting and employing research methods to useful effect, generally following an uncontested and familiar format.

The natural and life sciences employ a long standing tradition of positivism, advancing a hypothesis determined from inductive reasoning and observation, which is subsequently proved or disproved using empirical observation or experiment. Theories are described in sufficient precision to allow them to be disproved. Social sciences generally uses a more qualitative observation to support theory. Computer Science and engineering differs from these established paradigms by virtue of a conceptual or physical artefact created to address an identified problem and the subsequent evaluation of this artefact in achieving its intended aims. Peffers describes this design research artefact as follows (Peffers *et al,* 2007),

> *"Conceptually, a design research artifact can be any designed object in which a research contribution is embedded in the design."*

This methodological approach, termed *Design Research,* has established a consistent approach to the principles, pursuit and presentation of research within the overlapping domains of Information Systems, Computer Science and Engineering (Hevner & Chatterjee, 2010). The Design Research process, or alternatively *Design Science Research* process is generally described as a sequential series of steps as follows,

- identify problem
- define solution objectives
- design and development

- demonstration
- evaluation
- communication

These steps may be iterated, or *circumscripted* to further knowledge generation (Eekels & Roozenburg, 1991; Nunamaker *et al*, 1990; Hevner *et al*, 2004; Vaishnavi *et al*, 2015). Note that this evaluation tends to ascertain how well an artefact works, rather than how or why it does so (March & Smith, 1995).

There is no published optimal mathematical procedure to effect the set of geometry operations within all CAD programs. In this research, the identification of an automated process to map equivalent operations between CAD programs serves as an artefact that embodies sufficient exploratory detail to justify selection of a Design System research methodology.

This research and thesis is structured according to the twofold activities of Design Science identified by March and Smith, namely *build* and subsequently *evaluate*. As there are three complementary concepts constructed as design artefacts, namely that of semantic comparison, geometric comparison and GA parameter searches, they appear in separate, sequential chapters. The evaluation of the three resultant instantiations, or artefacts, are described sequentially in three following chapters. See the "roadmap" describing the thesis structure in Figure 1.

The identification of a problem may be that of an issue that is well documented within the field, or the recognition of an issue that is not yet understood to pose a challenge to development within this field (Gregor & Hevner, 2013). The significance of a research question must be justified, as must the chosen approach to determine a useful resolution. A research question is a foundation to a conceptual framework that supports the defined research objectives. This conceptualisation lends structure to the architecture of a proposed artefact or method that addresses the research question, ideally providing a rationalisation for subsequent implementation and testing.

In Chapter 2.1 to 2.7, an economic case is made to support increased interoperability between heterogeneous engineering design software. The specific issue of data loss in transfers made between parametric Computer Aided Design programs is described in Chapter 2.10 - 2.11 and Chapter 3.6 - 3.7.
An overview of recent research approaches is presented in the remainder of Chapter 3, where the trend for a prescriptive approach to imposing interoperability is supplanted by methods to discover equivalence between software functions. The research described in this thesis continues this trend towards mapping CAD feature functions, describing and evaluating methods to automate this process.

The solution objectives identified are based on the most simple proposition to automate the task of manually identifying and mapping CAD API feature functions, namely that the methods used by a translator are automated. This may be broadly defined as three translator approaches, namely,

- The manual identification of similarities of text descriptions of CAD API functions, whether in API documentation, functions names or in stub libraries. Computational semantic and syntactic comparison methods are adapted to the short, information-dense descriptive strings associated with concatenated function names and the terse descriptions found in documentation.
- Two or more feature functions may be considered equivalent in operation if they may be configured to create geometric shapes or transforms that can be measured to have a specified degree of similarity. Without prior knowledge of function parameter mapping, a geometric comparison may create geometry that has been translated, rotated or scaled relative to one another. While a human operator might immediately recognise similarity between CAD function outputs, it is not a trivial problem for machine comparison. A proposed solution objective is the machine

recognition of similarity between geometric surfaces independently of affine transformation.

- Searching for parameters that create a comparable geometry between functions undergoing comparison is an additional action that a translator performs to support the geometric comparison validation described in the second method. This task of mapping parameters may be considered as a search problem that can use the two techniques outlined above. A directed search method is proposed that takes advantages of the measure of increasing similarity afforded by the geometric similarity method.

One advantage of automating methods used by a human operator is that there is no recognised threat to future validity of these methods. Vendors will continue to publish descriptions of API functions and these functions will continue to perform a measurable outcome within a Cartesian virtual model space.

The design and development of an artefact alone is not a contribution to research, however the embodiment of novel methods or technology within an artefact may serve as a basis for a contribution towards basic research. Nunamaker *et al* describe several criteria that validate the development of artefacts in support of IS research as follows,

- The purpose of this artefact addresses an important issue within Computer Science
- This contribution is not trivial and represents an improvement over existing systems.
- This artefact may be tested against the defined objectives.
- The methods employed in constructing the artefact may be used within a broader, more generalised application

(Nunamaker *et al*, 1990). Hevner and Chatterjee identify an important epistemological concept within Design Science, that of *iterative circumscription* (Hevner & Chatterjee, 2010). This concept formalises the intuitive computer science process of increasing knowledge through the refinement of a software implementation.

The three research objectives above are described in Chapter 4 - 5 for that of semantic similarity measurement, Chapter 6 for that of geometric similarity measurement and Chapter 8 - 9 for that of the evolutionary directed search technique. Each of these chapters describe the concepts used to create an artefact that embodies these objectives, in each case this manifests as a software implementation. During the course of development, further considerations arise as a consequence of the design approach and are described. For example, the helical point sequencing method described in response to adoption of fast SVD model registration (Chapter 6.17). This cyclical process of development, testing and refinement is what Vaishnavi *et al* describe as *iterative circumscription* (Vaishnavi *et al*, 2019).

Evaluation of the three concept instantiations may also be described within the Design Research paradigm. As the evaluation strategies differ over the three instantiations, they are described separately in detail below. In their description of Critical Realism applied to Mixed Methods Information Systems research, Zachariadis *et al* describe a *retroductive analysis* equivalent to a post hoc hypothesis of phenomena observed during experimental analysis. This approach is used in this research where experimental results merit further analysis (Zachariadis *et al*, 2013).

In the case of short text semantic comparison, existing techniques are supported by relatively few theoretical concepts. The distributional hypothesis (see Chapter 4.3), the term frequency, path length and information content account for most of the concepts supporting semantic similarity comparison measures. It is difficult to predict the efficiency of these techniques when used on short, information-dense technical phrases. Consequently, most relevant techniques are compared over several representative datasets, namely hand-compiled matches from commercial CAD API documentation, in what

Venables *et al* term a *naturalistic* evaluation strategy. A quantitative statistical evaluation allows direct comparison between methods and their combinations, such as the semantic similarity methods used for single words used in combination with a greedy optimisation for short phrase comparisons (see Chapter 5.3). The number of discrete variables such as word2vec parameters are limited by research scope, but a representative range of comparison methods over a narrow technical corpora highlights methods that show future potential.

Separated mixed methods are used for evaluation of the geometric similarity comparison method as follows. Firstly, quantitative analyses use an existing library compiled with the express purpose of benchmarking CAD shape matching methods. This approach allows a direct comparison against other methods (Bespalov *et al*, 2005). The validity of these tests are limited by the complexity of the primitive models used. A qualitative assessment is made of complex models from this same library. The mix of both methods is justified where there are constraints on the scope of variables that may be addressed by a quantitative survey in the time available, and where both contribute to a broader perspective on the relative merits and shortcomings of a design concept (Venables, 2016; Fielding, 2012).

The Genetic Algorithm search method described in Chapter 9.1 is evaluated using quantitative methods in Chapter 9.2.1. Two case studies are tested using several independent variables and the results directly compared against a default combinational limit.

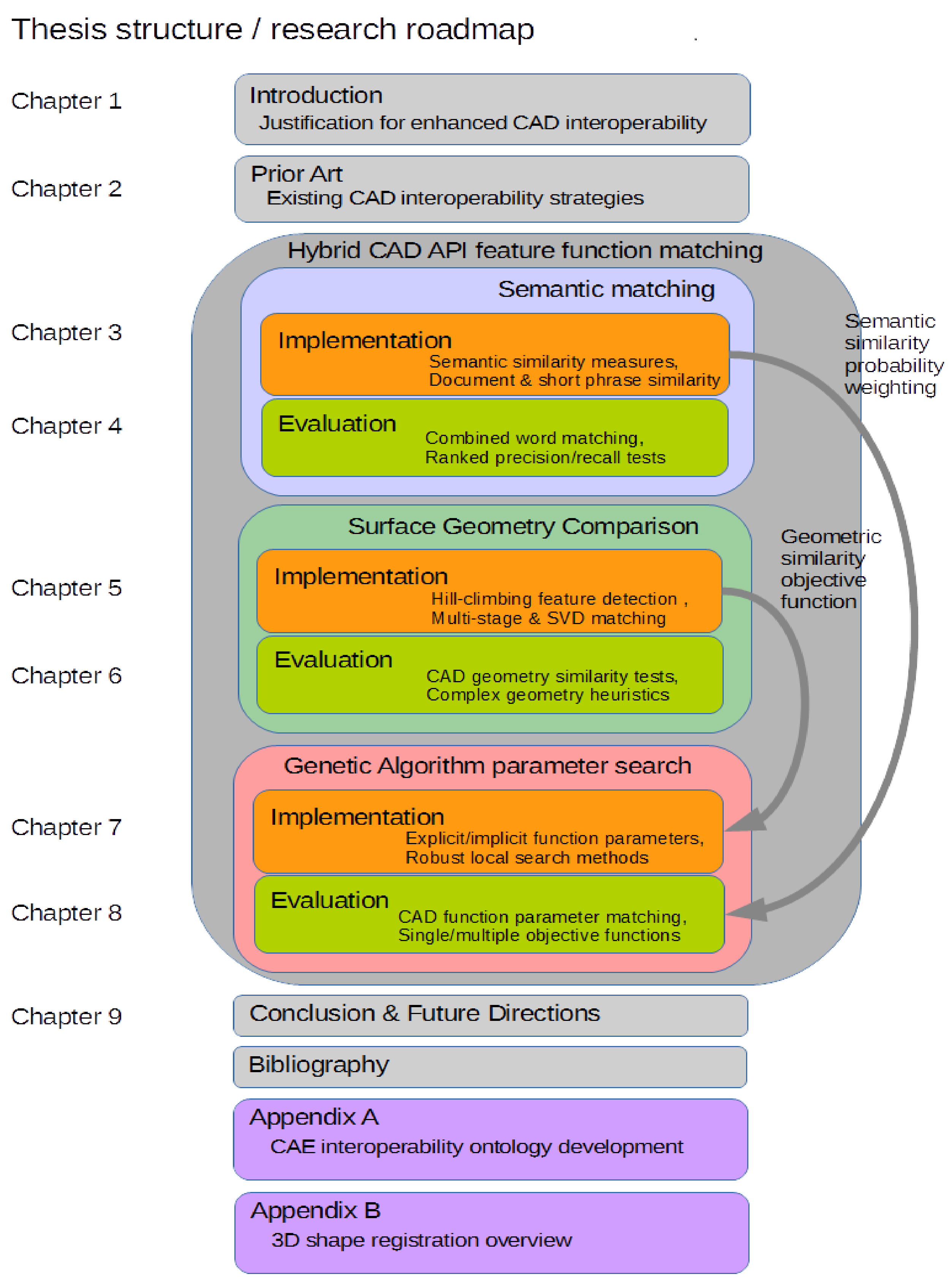

Thesis structure / research roadmap
Chapter 1
Introduction
Justification for enhanced CAD interoperability
Chapter 2
Prior Art
Existing CAD interoperability strategies
Hybrid CAD API feature function matching
Semantic matching
Chapter 3
Implementation
Semantic similarity measures,
Document & short phrase similarity
Chapter 4
Evaluation
Combined word matching,
Ranked precision/recall tests
Semantic similarity probability weighting
Surface Geometry Comparison
Chapter 5
Implementation
Hill-climbing feature detection ,
Multi-stage & SVD matching
Chapter 6
Evaluation
CAD geometry similarity tests,
Complex geometry heuristics
Geometric similarity objective function
Genetic Algorithm parameter search
Chapter 7
Implementation
Explicit/implicit function parameters,
Robust local search methods
Chapter 8
Evaluation
CAD function parameter matching,
Single/multiple objective functions
Chapter 9
Conclusion & Future Directions
Bibliography
Appendix A
CAE interoperability ontology development
Appendix B
3D shape registration overview

Figure 1: Thesis structure

1.3 Research Methodology

Content Overview

The introduction to this thesis (Chapter 2) deals with the general problem of interoperability between computer data formats within engineering disciplines and commercial enterprise. The methods and difficulties encountered with interoperability strategies are outlined alongside the context of organisational constraints. Interoperability is synonymous with progress in a range of industries, from multi-tier supplier networks in high value manufacturing to Building Information Modelling, the role of enhanced interoperability between engineering domains is examined. Computer Aided Engineering domains that share the physical, test and manufacturing data associated with product data are introduced, namely Computer Aided Design, Computer Aided Analysis and Computer Numerical Control machining. The concept of parametric CAD modelling is introduced to place the difficulties of CAD data interpretation between vendor products into context. Commercial methods to attain interoperability within Computer Aided Engineering are summarised, as are their respective shortcomings. This includes research efforts that adapt formal reasoning to identify semantic mapping between parametric CAD definitions, emphasising a need for machine searching and verification of prospective semantic matches.

Chapter 3 describes previous efforts to solve issues of interoperability between CAD programs. The difficulties encountered by the ongoing ISO10 303 standardisation initiative of parametric CAD formats are described. These problems, such as the so-called "persistent naming problem" highlight the intractability of semantic and procedural inconsistency between various closed-source commercial CAD programs. Alternative strategies and their relative success are described, such as the *Macro-Parametric approach*, which compiled a core set of universal CAD feature operations, the Universal Product Representation which uses a database of likely equivalent function candidates to test for compatibility between models, and the *Three-Branch model* which proposes a combination of semantic and geometric information to facilitate automated CAD mapping. Research efforts have moved from a prescriptive standardisation effort to automated methods of verifying geometric and conceptual equivalence between CAD model features. This chapter touches on the Hausdorff measure to determine geometric shape equivalency and the Hoffmann query protocol for direct numerical comparison of CAD model geometry.

Most of the formalisation efforts to create a universal semantic taxonomy for parametric CAD resulted in research ontologies. These constitute a diverse field in their own right, but also suffer from a limited adoption by vendors. The use of formal methods to determine interoperability is tangential to the central thesis and is listed in Appendix **A**.

Chapter 4 introduces the field of semantic matching within text documents. A brief description serves to introduce popular measures used to determine relationships between single words and texts. The methods used can be based on word relationships derived from manual or machine compiled corpora. The methods suited to matching short, dense texts are described, existing document retrieval methods are unsuited to the terse labels and descriptions associated with function API. The efficiency of these techniques is subsequently tested in Chapter 5.

Chapter 5 describes the requirements of text similarity evaluation for short texts associated with application programming interfaces. Documentation describing CAD API is parsed to yield sets of words. The similarity between all combinations of these words is calculated using several measures used with compiled corpora and similarity measures calculated using statistical models. These word similarity matrices are used in conjunction with the greedy method and compared with other document retrieval methods to test two sets of known matches between CAD API function descriptions. The relative performance of these techniques is evaluated with several document retrieval ranking measures.

Chapter 6 describes a CAD model geometry matching method suited to automated CAD feature function matching. A universal boundary surface query derived from intersection of a vector with a model surface is determined to be a widely implemented feature of commercial CAD programs. Comparison of geometric surfaces requires that surface sampling is invariant to model orientation or position, requiring that sets of point intersections between models may be accurately registered. A closed-form SVD solution is described that requires key feature points for model registration. These feature points also perform a role as a pose-invariant model signature for matching. A novel search method is described to identify several variants of surface features for model registration.

Chapter 7 tests the affine-invariant geometry matching method on a benchmark CAD shape comparison library. The precision and recall of this method is measured over simple geometry models that are randomly scaled, orientated and positioned. An example of a matched complex geometry model is given, using a partial complement of feature registration points.

Chapter 8 examines the constraints of parametric models in detail and proposes a genetic algorithm search method for efficient discovery of CAD function equivalence. This method uses a simplified variant of the geometry matching method as an objective function. A number of simplifications to feature function parameters are developed to reduce the combinational search space. A stochastic search using an evolutionary algorithm is developed for the purpose of mapping individual parameter relationships.

Chapter 9 demonstrates the function matching genetic algorithm proposed in Chapter 8 across a binary and trinary parameter representation. A single objective GA reduces geometric difference between the models created by independent CAD functions. A multi-objective test combines minimisation of geometric distance with a bias toward default zero-valued function parameters. A semantic similarity comparison is applied to the respective parameter labels and the determined values subsequently used to bias GA mutation.

Chapter 10 summarises the findings of the research directions and speculates on future directions that hybrid machine API mapping might take,

1.3 Research Methodology

Appendix 12 provides a historical summary of the use of ontologies to provide semantic interoperability between heterogeneous CAD and CAE functions. The various approaches to ontology specification are described alongside the difficulties of populating top-down prescriptive models against discovering bottom-up descriptive models.

Appendix B compiles popular methods of shape matching from the disparate fields of image matching, point cloud registration, 3D shape retrieval and CAD model comparison. An overview of view-based, histogram-based, transform-based and graph-based methods is given to justify the requirement of developing an accurate and rapid multi-stage comparison method for determining CAD model equivalence.

Appendix C is a table showing the test outcomes for single model matching results from the test described in Chapter 7.

1.3 Research Methodology

2 Introduction

The coordination of enterprise operations dictates the structure and efficiency of a traditional manufacturing firm. The quality of information transfer between operations is tantamount to coordination effort, this quality of information transfer is in turn dependant on the ability to seamlessly interpret information. The automation of data processing is transforming the design activities and specialised analysis integral to modern production methods where transfer of specialist information between domain experts has largely become the transfer of computer software data between diverse computer programs

Domain-specific software applications process the information generated within these knowledge domains and data exchange between these specialist programs constitutes a significant proportion of necessary communication between enterprise agents. Products or constructions of any complexity require coordination of multiple subcontractors and subcomponents, product data is shared between designers, engineers, sub-contractors, suppliers, marketeers, production process planners, logistics planners and others. While software vendors develop products aimed at these specialist domains, enterprises expend commensurate resources integrating these diverse programs within a cohesive system. Business analysis software and engineering software are considered to be instrumental in the efficiency of the services, manufacturing and construction economy. The range and complexity of communication between networked enterprises or enterprise agents determines the efficiency of participation within a market, in the form of lower transaction costs and wider market. The rapid evolution of cheap computational processing capacity has spurred development of intensive computational applications such as three dimensional modelling software and finite element analysis software used to predict physical behaviour of modelled objects.

The increasing complexity and diversity of these programs outstrips efforts to standardise the format of data exchanged leading to imperfect transfer of data between heterogeneous software (Panetto & Molina, 2008). Development of new functionality within an engineering software must necessarily create technical incompatibility with comparable legacy software. Software vendors release new versions of their products and withdraw support for older versions, requiring ongoing development to provide interoperability with competitor software. This trend is particularly burdensome for Small to Medium Enterprises that may have to interact with several different competing formats within a commercial network of several buyers (Le Duigou *et al*, 2011). In many cases the software and underlying operating system has a shorter lifespan than the product it is used to specify, in these cases data archival becomes challenging. (Ball *et al*, 2008; Heutlebeck *et al*, 2009; Peeling & Satchel, 2001).

Within engineering industries, the cost of data loss between different agents is variously estimated. The National Institute of Technology and Standards commissioned a study indicating that a billion dollars was lost annually within the US automotive chain (Tassey *et al*, 1999). A similar NIST study of interoperability-related issues within the US capital facilities industry revealed an annual loss of $15.8 billion, notably in the transferral of data between CAD and other engineering software (Gallaher *et al*, 2004). These surveys and others indicate that this expense is caused by translation and remodelling costs of geometry data files exchanged between manufacturers and suppliers. Estimation of imperfect interoperability is a cost that is commonly overlooked (Horst *et al,* 2010).

Standardisation has historically been seen as the primary means to provide a neutral basis for transaction between industries (Tassey, 2000). The standardisation of Computer Assisted Engineering software exchange formats have proceeded alongside that of CAE software development, but the process of standardisation consensus lags behind industry practice. The International Standards Organisation promotes a number of standards and a neutral exchange format, STEP, to address the compatibility issue between CAD programs (Pratt, 2001). Incomplete implementation of the STEP format by commercial vendors and dated geometrical definitions have led to slow industry adoption (Gielingh, 2008). This leads to improvised and sub-optimal procedures, for example the Aerospace Industries Association of America publishes recommendations on assessing errors in STEP data translations (AIA EDIG Guidebook, 2013), while the trend amongst automotive manufacturers is to dictate the software format that suppliers must use (Gerst & Bunduchi, 2005).

2.1 The economic case for enterprise interoperability

This research was originally motivated by the observation that there appears to be little equivalent to the Open Source computer science community phenomenon within the domain of mechanical engineering and manufacturing. Unlike the open source software environment of operating systems, compilers and allied tools, the programs used within the engineering community are relatively expensive commercial closed-source products that use proprietary data formats. Exploring issues of interoperability between engineering application software uncovered wider related problems across industry software. Interoperability itself has several definitions, within the context of data transferral between computer systems it is sufficient to define interoperability as,

> *the accurate and automatic interpretation of the meaning of information exchanged by two or more computer systems,*

(Paviot *et al*, 2009). This definition satisfies both technical and semantic interoperability concepts specified by Kosanke (Kosanke, 2004). Organizational interoperability is a correlated measure of business transactional cost that is related to technical and semantic interoperability (Paviot *et al*, 2011). Hoffmann introduces a useful definition of interoperability within the context of creating intermediate geometric model representations within defined precisions (Hoffmann *et al*, 2013). This interpretation is revisited in Chapter 3.12, Representative proxy model and query protocol.

Information Communication Technology is considered to lower the cost of market transactions (Malone *et al*, 1987). Transaction Cost Theory uses the market transaction costs model to predict that the size, structure and success of an enterprise is determined by the organisation configuration dictated by transactional costs. If the management overhead in communicating with outsourced transactions is more than that for equivalent in-house operations then the enterprise should expand to incorporate these operations.

Conversely, if Information & Communication Technology lowers the management communication costs of outsourcing, it is more efficient to adapt the enterprise size to those operations that still require detailed management and interpersonal interaction.

Enterprise Architecture and Enterprise Modelling theories formulate the optimal configuration of future enterprises. Virtual Manufacturing Enterprise organised around lowered communication costs predicts the replacement of interpersonal interaction within the traditional manufacturing plant with a distributed multi-agency model of specialist processes. These new paradigms can be seen in the proliferation of start-up enterprises based around the rapid crowd-funded design and manufacture of innovative consumer-led products. In many cases the engineering, prototyping and production of these products is entirely outsourced.

These initiatives are supported by a decoupling of the enterprise domains from the traditional integration within a firm. Within manufacturing, the change from Dedicated Manufacturing Lines associated with mass production to Flexible Manufacturing Systems and Reconfigurable Manufacturing Systems allows manufacturers to cheaply adapt production lines to different products. Cheap re-tooling allows manufacturers to solicit low-run production from external companies. Efficient data transfer underpins the development of a more distributed and agile production economy.

2.2 Small to Medium Enterprises

Small to Medium Enterprise market engagement can depend on the ability to provide a niche service or product. The efficiency of this process is dependent on the quality of coordination data available (Rullani *et al*, 2000). Capturing a specialist technological market requires tightly-coupled integration with supporting suppliers and buyers. In the case of tiered suppliers to Original Equipment Manufacturers, SME viability stems from coordinated Supply Chain Integration, where transactional costs are offset by niche specialisation. For SME to participate in the design process of product development, this can require native access to the engineering applications used by the coordinating enterprise. In the case of automotive subcomponent suppliers, the Original Equipment Manufacturer will mandate the use of specified PDM or CAE software to participate in the supply chain (Global Supplier Info Pack For FEDE-C3PNG Integration, 2017). Where a SME solicits business from several OEMs, there is a requirement to license several CAE or PLM systems to retain compatibility. As the licensing costs of industrial CAD programs represents significant capital expenditure for a SME, this presents a market barrier (Lomas & Matthews, 2007). A recommended method for aeronautical engineering translates CAE data to a neutral CAE standard format (ISO 10 303 STEP) and then advises the capture of mismatches between the proprietary CAE data and the neutral CAE formatted data. (Aerospace Industry Guidelines For Implementing Interoperability Standards For Engineering Data, 2013). Surveys also indicate that a significant percentage of data models received by SME engineering firms require rework or remodelling, translation between different CAD formats is an expensive process that introduces errors and loses data (Peruzzini *et al*, 2011). Note that the dataset of smaller SMEs in OECD statistics is under-represented, there is an administrative burden in obtaining data from companies with employees of five or fewer members, furthermore, there is generally a poor response from such enterprises that is attributed to the relative administrative overhead involved (Atkinson, 2004).

2.3 Integrated software and Product Lifecycle Modelling

Within larger companies, Product Lifecycle Modelling (PLM), is a platform to share data between engineering design and analysis applications; this addresses transfer of geometrical design data to models suited to numerical analysis and facilitates design cycle versioning (Assouroko *et al*, 2010; Le Duigou *et al*, 2011). PLM packages have developed from CAD and Product Data Management products representing production management tasks or Enterprise Resource Planning technologies that may span entire operational planning requirements for a business (see also Section 8, Decentralised Enterprise and other manufacturing paradigms). Until recently, these large and specialist platforms have been beyond the reach of SMEs owing to their high cost of purchase and integration

(Subrahmanian *et al*, 2005). Lately, off-the-shelf packages been adopted by SMEs. Studies of PLM adoption by French SMEs suggest that they are used as a cost effective means of CAE interoperability (Bidan *et al*, 2005).

Poor CAE application interoperability is being substituted by PLM platforms, however PLM systems that are not extended proprietary CAD products limit interaction with product data files to visualisation (Van Wijk *et al*, 2010). PLM vendors have evolved from Product Development platform vendors (e.g. UGS PLM Solutions, Tecnomatix, IBM-Dassault, Windchill), Enterprise Resource Planning platforms (Baan, SAP, Oracle) and more generic business ICT integration platforms (Microsoft, MatrixOne, Agile) (Terzi *et al,* 2006; Le Duigou *et al,* 2011). Integrated product design systems that provide the individual design, visualisation, numerical analysis and versioning applications on a single platform (normally by virtue of sharing a geometric kernel) are generally priced beyond the scope of SME.

2.4 Concurrent Engineering

While customary engineering practice has been to tackle product design and analysis in a sequential cycle, Concurrent Engineering defines a practice of running separate product design and analysis processes simultaneously to reduce product development time.

This approach obliges process planning and test analysis evaluations to be shared during early stages of the design phase. Productivity gains arising from concurrent engineering teams are offset against the greater transfer of engineering data between product domains (Yassine *et al*, 2003).

Successful concurrent and collaborative engineering practice places high interoperability demands on CAE application integration, one approach is to buy multiple seat licenses for a turnkey CAE integrated platform. The design of the Boeing 787 aircraft used Dassault Systems PLM platform comprising CATIA V5 CAD, DELMIA DMS visualisation and ENOVIA PDM. The thousands of engineers working on this global project used identical software and versions at $20k per seat licence. In 2006, the Airbus 380 design famously used two different versions of a CAD package (CATIA V4, CATIA V5) between partners, resulting in late stage interoperability problems and $6B losses (Ayubi, 2011). The integrated platform approach is expensive. It represents significant investment and subsequent risk of future vendor "lock-in" and it also reduces the availability of alternative CAE applications that may be more suitable (Tassey, 2010). Automotive enterprises tend to use a range of different CAE products for the required electromechanical integration, design and testing. Different software is specialised for designing drivetrain, electromechanical and styling aspects. The design process involves active participation with first and secondary tier supplier chains. As mentioned previously in the context of SME expense, it is common practice to mandate supply chains to use the same PDM or CAD systems (Global Supplier Info Pack For FEDE-C3PNG Integration, 2017). This requisite interoperability between supply chains, process designers and product designers places constraints on the viability of concurrent engineering.

2.5 Archival requirements

Engineering Informatics Archival is defined as the field of engineering information archival with particular reference to computer information. Archival of CAE application data presents the same issues as CAE application information interoperability. Once vendor support for legacy proprietary formats is lost, then all information that is inoperable with other applications is either lost or must be reverse engineered. Certain products such as aircraft, military hardware, medical hardware or public structures, have service lives that greatly exceed the lifespan of CAE application releases, or the underlying hardware and operating system. There may be legal, contractual or economic requirements to preserve CAE product data (Heutelbeck *et al*, 2009). While the recommended procedure is to utilise standard vendor-neutral formats such as ISO 10 303 STEP AP203, there are no suitable vendor-neutral programs to generate this data, and proprietary applications provide uneven implementation of recommended standards (Gielingh, 2008). Parametric features that capture design intent are not standardised across modern CAD programs, there is an uncertainty around the intellectual ownership of the definitions used to create design models (Patel & Ball, 2008). Where engineering information or design is defined in proprietary CAE applications and formats, the entire CAE system and underlying operating system and hardware may have to preserved. As many industrial CAE systems run on mainframes, this is an expensive proposition (Lubell *et al*, 2008; Peeling & Satchel, 2001).

2.6 Building Information Modelling

Building Information Modelling represents a coordination of the various agents required to complete a building, combining architecture with mechanical, structural and service engineering (Howard & Bjork, 2008). The Industry Foundation Classes, specified by the BuildingSMART Alliance is a widely used standard that captures semantic metadata on building modelling. The IFC specification works well for it's intended purpose, sharing visual data between stakeholders and contractors on a building project. However, the CAD programs that generate IFC metadata do not have consistent implementation of the protocol (Steel *et al*, 2012), and are reported lack validation processes for checking data exchange (Akinci *et al*, 2010). While some researchers envisage BIM as a future protocol for legal and contractual coordination on building projects, it appears that uneven CAD vendor support will limit this potential (Sebastian, 2010). A US survey from 2002 estimated an annual loss of $15.8B through interoperability inefficiency between stakeholders and contractors within the capital facilities industry. Studies indicate that incomplete interoperability is the major cost to the building industry (Gallaher *et al*, 2004).

2.7 Decentralised Enterprise and other manufacturing paradigms

To pursue profitability under increasingly global competitive pressure, manufacturing processes have been advanced to afford greater reconfiguration and a faster response to changing product specification (Sanchez & Nagi, 2001). The Agile Manufacturing paradigm is shifting towards mass customisation supported by Flexible Machining Centres. *Reconfigurable Manufacturing Systems* are emerging as a more efficient option to

FMS (Jørgensen *et al*, 2011). Where product line modularity and customer customisation is supported, there is an associated decoupling of manufacturing process from product. The emerging RMS manufacturing plant is not only more adept at fulfilling differing business orders, it is better placed to solicit orders of different businesses. Improved business and product data interoperability is seen as central to Agile manufacturing practice (Yusuf *et al*, 1999). These core tenets evolved into Enterprise Resource Planning, integrated business data processing and Product Lifecycle Management, the organisation of data generated from all aspects of product engagement. The models of responsive and reconfigurable manufacture-on-demand, coupled with projected technologies of design virtualisation and Collaborative Engineering give rise to speculative enterprise organisation.

Virtual Enterprise, or Virtual Manufacturing Enterprise are understood to be temporary consortia of existing enterprises that capitalise on market opportunity (Camarinha-Matos *et al,* 2003). These organisation arrangements are characterised by purely network, usually internet, coordination. *Cloud Manufacture* is a wholly decentralised virtual enterprise. If the ability to share design, test and production data between agents is taken to a logical extreme, it amounts to a completely virtual production process (Tao *et al*, 2011; Souza *et al*, 2006; Stark *et al*, 2010; Romero *et al*, 2010; Wang, 2012; Romero *et al*, 2012). Other prospective manufacturing arrangements include *Manufacturing-As-A-Service* (*MaaS*), *Distributed Manufacturing Systems*, *Peer Manufacture* and *CoDesign*, (Butala *et al*, 2013; Haythornthwaite, 2009).While there are a range of implementation details and motivating agents, all of these prospective organisations are variations on a theme of decentralised product design and manufacture. Enterprise decentralisation extends existing efficiency developments accumulated from specialist services, close-coupled supply chains and lowered transaction costs. Each of these visions without exception are reliant on networked data interoperability. Each envision seamless exchange between heterogeneous software applications.

Enterprise Resource Planning and Supply Chain Management adoption have eroded market transaction costs of searching, discovering and comparing services (Turna, 1998, Steinfield, 2011; Malone & Benjamin, 1987). This progress suggests that decentralised production enterprise might occupy niches that are uneconomic for other forms of enterprise organisation. Where this line of reasoning comes adrift is the difference between the information exchanged over Service Oriented Architectures based on simple, self-describing web interfaces and CAE domain data that is shared as part of an iterative design cycle. There are consortia-based standards and international standards that can theoretically capture this data, but these standards are not well supported by commercial software. The reasons for this imperfect standards implementation within commercial products are described in detail in Chapter 3.6, Standardisation of parametric features.

2.8 An illustrative overview of Computer Aided Engineering domains

Systems engineering, product design and manufacturing process domains within production enterprise are lumped under the term Product Lifecycle Modelling (PLM) (Sudarsen, 2005). This term is distinct from Product Life Cycle Modelling (PLCM) that covers the business perspective of production while Engineering Informatics (EI), is the computerisation and coordination of PLM activities.

It is helpful to illustrate the representation of an engineered object within several distinct CAE domains. The example is of a countersunk blind hole set in a circular plate.

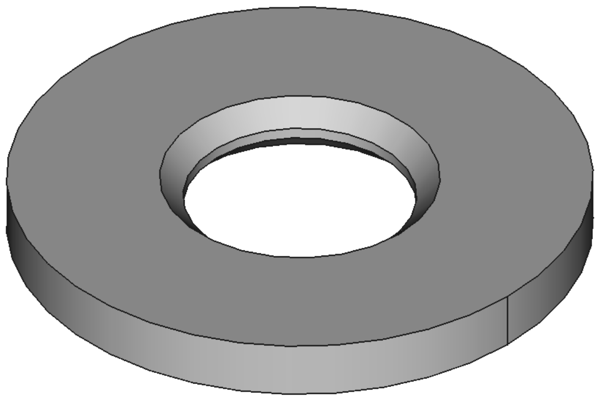


Figure 2: CAD model of plate with countersunk hole

A parametric *Computer Aided Design* program models the geometry and design constraints of the plate example as a cylindrical solid extrusion on a datum plane with a hole feature object of a countersunk variant subtype. Earlier three dimensional CAD programs may have explicitly modelled an oblong body intersecting a cylindrical void and a conical void. Typically CAD models serve as a master document that are referenced by other engineering, design and production teams. Recently these files tend to include non-geometric model data such as materials specification and design notes, referenced as a *Model Based Data* paradigm. Figure 2 shows an instance of this model created within the FreeCAD parametric CAD program (Riegel *et al*, 2019).

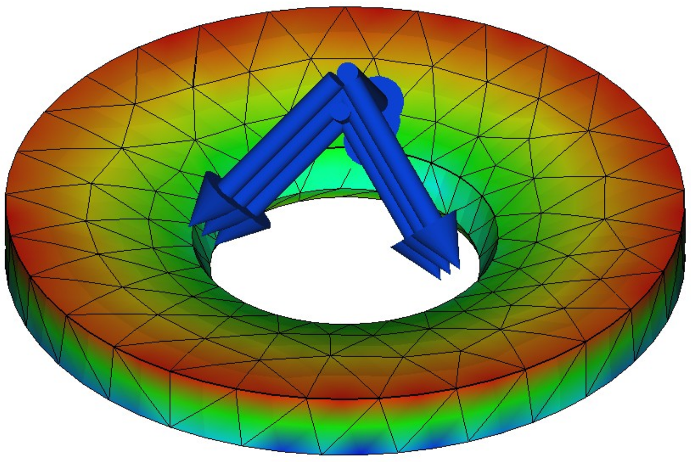


Figure 3: FEA model of plate under applied force.

Figure 3 show calculated stress and distortion of the earlier CAD plate model in Figure 2, reacting to an applied virtual force. A *Finite Element Analysis* uses the model surfaces to contain a generated lattice of space partitions, these finite model subdivisions then allow individual calculation of partial differential equations representing physical phenomena. This FEA model can then be used to test stress concentrations around this hole in the presence of applied force, or heat transfer through the plate if the model is subjected to a temperature differential. These virtual techniques can greatly reduce or eliminate the requirement of physical prototype testing. This analysis is part of the design phase of product engineering. Model analysis files are typically large and the process is numerically intensive, so it is customary to keep this data separate from geometry data. Examples of 3D CAD parametric software include Dassault CATIA, while ANSYS is a well-known Finite Element Analysis software, the analyses Figures 1, 2 & 3 are generated within FreeCAD, an open-source parametric CAD/FEA/CAM project (www.3ds.com, 2019; www.Ansys.com, 2019; Freecadweb.org., 2019).

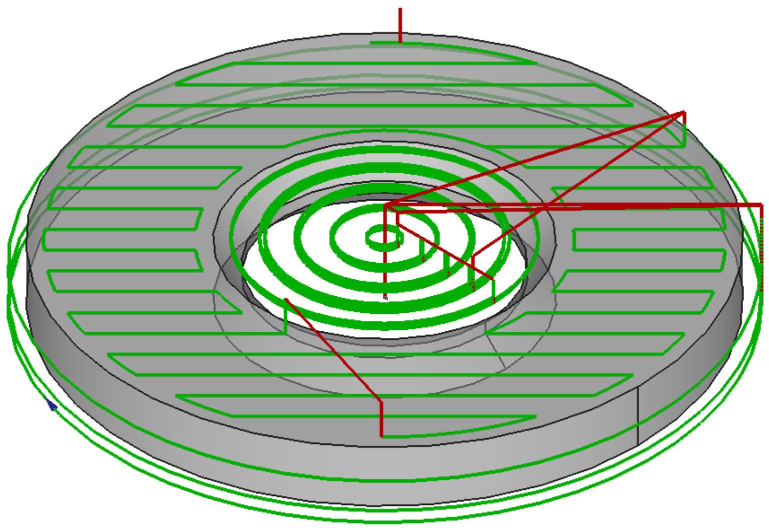


Figure 4: CAM software generates a toolpath for a milling operation.

Computer Numerical Control machines typically remove workpiece material as part of a manufacturing production process. The geometry defined in the CAD model is re-interpreted by a *Computer Aided Manufacturing* program according to the materials selection, the complexity of the geometry and the type of CNC machines available. These configuration selections might be further optimised by *Computer Aided Process Planning* in a large manufacturing plant.

Figure 4 shows the calculated machining paths required to create a physical instance of the CAD plate model in Figure 2 using a Computer Numerical Control mill. A CAM program takes the surface geometry and calculates the machining operations to create the CAD model on the requisite machine. A CNC mill might require instructions to move an end mill to the hole location and remove the cylindrical portion of the hole in a spiral motion, followed by selection of a ball end mill to remove the countersunk bevel. A **CNC** lathe might cut the geometry from cylindrical bar-stock, using a drill followed by an internal turning operation to remove the countersunk material. The finished piece might then be sliced from the bar with a parting tool. A 3D printer might build successive layers of laser-sintered metal or fused polymer filament to create the plate object. It becomes evident that the same piece can be manufactured in different ways on different CNC machines, requiring different toolpath planning operations. These operations typically take place as part of a production planning process, or may be carried out by independent subcontracting parts suppliers as CAM programs are frequently allied to CNC hardware.

A *Coordinate Measuring Machine* measures the geometry dimensions of a manufactured part for acceptable surface tolerance variation. This process may be integrated into a production line, or may be a stand-alone process for small batches of machined products. The extraction of *Geometric Dimensioning & Tolerance* model data from the CAD model is still a time-consuming manual requirement in most cases.

2.9 Parametric CAD feature modelling

A parametric feature-based CAD program will create a geometrical representation of a model that is generated from a topological model. This model topology is structured from a configuration of parametric *features*, modelling shapes based on a predetermined morphology which form the building blocks of model construction. These features are sequentially added and refined by an operator at a Graphic User Interface or by a series of Application Programming Interface commands. CAD parametric features embody engineering, functional or topological concepts with a local semantic definition (Hounsell & Case, 1998). Their advantage over a purely geometric or topological model definition lies in the parametric variables that give control over the quintessential feature characteristics.

CAD systems have evolved to use higher order representations of model aspects to reduce time spent in reworking models, geometrical changes can be generated by the application in response to changes in feature or topology definition. CAD systems possess functions to manipulate model feature parameters and organisation and there is generally a subset of command functions that are tightly coupled to the set of local CAD features.

Earlier methods to translate CAD models between heterogeneous applications relied on geometrical translation; holes and surface anomalies are inadvertently introduced into translated models by different CAD programs. This can be partly attributed to the difference in algorithms used to generate CAD geometry. The sequence of operations used to recreate a translated CAD model may also cause splits and slivers in the geometry surfaces. Different CAD software uses different numerical precision and different schema for geometry tolerance that frequently cause defects in model reconstruction (Gerbino & Brondi, 2004). If CAD features are mapped between different CAD applications, this allows native models with associated topology, geometry and design parameters to be generated within the target CAD application (Seo *et al*, 2005; Altidor *et al*, 2009).The problem is then to find features with equivalent semantics between heterogeneous CAD applications.

2.10 CAE data transfer methods and their limitations

The STEP neutral format provides a theoretical means of transferring surface geometry between CAE applications, in this case the STEP Application Protocol 203, but other data created by different CAE analysis is poorly supported or missing (De Sapio, 2010; Gielingh 2008; Goossenaerts, 2009). Certain high-level CAD vendors provide multi-analysis software that advertise seamless product data transfer between a suite of CAE analysis software. Some vendors create interoperability by sharing data from the proprietary geometry engine or kernel of their respective systems (Slansky, 2005).

These vendor products are aimed at large Original Equipment Manufacturers and typically represent significant capital investment. Third party translators exist that will translate models between different CAD packages, these programs (or services) are neither inexpensive nor infallible and generally require intervention to clean up errors (Gerbino &

Brondi, 2004). Some vendors now lease access to these suites of cloud services, but this still leaves issues of archival and data exchange between products of different vendors (Autodesk.co.uk, 2018). Commercial Product Data Management solutions have a relatively low uptake amongst SMEs, and are reported to be simply used to facilitate CAE software interoperability (Bidan *et al*, 2012). Interoperability of CAE systems is considered to be of particular economic benefit to SMEs, yet remains relatively inaccessible (OECD Workshop 2000; Le Duigou, 2012).

Much of the reasoning and data that is embodied within engineering design is not recorded within product geometry. Newer model formats attempt to capture this so-called “design intent”, other format additions capture qualities such as materials specifications, or product disposal recommendations. Efforts to adapt reasoning logic languages to capture product data semantic information have led to the development of an abundance of research models for information classification (see Part 12, Ontologies for CAE interoperability for details). Ontologies are structured specifications of domain information, and interpretation that define domain information relationships (Gruber, 1995). Because ontologies formalise the interpretation of domain data, these methods have been seen as a solution to communicating data without misinterpretation. Ontologies are subject to the same limitations of interoperability as data formats, they are by definition domain specific, implemented by domain-specific experts and are usually not derived from an overarching meta-ontology (Ciocoiu *et al*, 2001). However, the techniques used for merging or mapping ontologies have been used to find mappings that translate information between CAE programs. In the case of Computer Aided Design information, it becomes possible to automatically verify semantic matches using geometrical data queries.

2.11 Mapping ontologies

Feature semantics and defining parameters are local to individual CAD applications, mapping features and the functions used to access them is a laborious task requiring skilled intervention. Research efforts have been directed at directed mapping using techniques from ontology alignment and bridging. Individual CAD applications can be viewed as local ontologies composed of the semantic organisation of features concepts and the command function parameters that control these features (Kim & Han, 2007). Interoperability between CAD programs may be achieved via a semantic mapping of the CAD API feature functions and their associated parameters (Wang & Wang, 2014).

It would be a simple task if local CAD feature ontologies were derived from a universally accepted top-down product data ontology, many of which have been proposed (Ciocoiu *et al*, 2001). This is not the case, for the same reasons that prescribed product data meta-standards are not uniformly implemented among CAE vendors. The pragmatic approach has been to employ so-called *bottom-up* ontology mapping techniques to discover relationships between different CAD feature ontologies. These techniques determine probable relationships using discovered syntactic matches between parameters or feature labels and may also compare the organisational relationships between features and feature subtypes. General ontology mapping techniques are insufficient to allow unsupervised generation of bridging ontologies, but in the case of CAD API mapping where exposed API functions may number in the thousands, even partial matching success may significantly reduce human intervention. These ontology methods are covered in greater detail in Appendix 12.

Mapping efforts such as the TransCAD macro-parametric method construct a static library of mapped functions common to all CAD programs under consideration (The Macro-Parametric Approach). This approach does not reliably recreate model geometries constructed from long sequences of parametric feature operations which embody so-called *implicit constraints*, model specifications that are derived from complex sequences of operations. These context-dependent function behaviours may be captured using a *dynamic* mapping, where several functions are tested for their ability to replace a model construction sequence step in a different CAD system. Dynamic mapping requires repeated function mapping tests, creating a demand for an automated process. This approach is explained in greater detail in Error: Reference source not found, Error: Reference source not found.

2.12 A contribution to automation of feature mapping between CAD programs

The following Chapter 3 describes the most recent methods devised to approach the problem of CAD interoperability in detail, where the concept of mapping CAD API functions is used to recreate nominally equivalent models in heterogeneous CAD programs. This method is limited by the significant labour required to create a mapping between heterogeneous CAD API, summarised in Chapter 3.13. If a means to automate CAD API function mapping is found, this promises CAD data transfer that preserves a higher information content at a lower cost of manual intervention.

The research underpinning this thesis differs from previous semantic CAD API mapping research in the combined use of novel geometrical and semantic methods to increase the probability of unsupervised feature matching success. The task of translating models then becomes one of determining equivalent API function calls. This method can be outlined as follows.

If a method can query model geometry represented in different CAD programs, these models can be tested for geometric similarity. These techniques of model geometry matching are described in Chapter 6, (Boundary Surface Geometry Comparison).

A dynamic function mapping process will take each sequential parametric function operation from a CAD source model and determine a closest analogue function operation within a target CAD program. The API libraries of modern commercial CAD programs may contain several thousand distinct function operations, consequently this approach would be computationally intensive without a means to identify a shortlist of probable matches. This effort may be greatly diminished by ranking the search order of candidate functions according to their semantic similarity. Document retrieval methods are not suited to the short phrases encountered in function and parameter names, or their brief functional descriptions, instead a range of existing techniques and novel semantic matching techniques are compared in tests on several CAD API texts. A description of these methods can be found in Chapter 4, the outcomes of tests in Chapter 5. The limitations of documentation retrieval methods used for short phrases is covered in further detail in Chapter 4.2 - 4.3.

The process of function matching requires that individual function parameters are matched to their counterpart, should an equivalent exist. The geometric validation technique allows different parameter configurations to be tested to determine if they have a similar effect on the geometric output of a function operation. CAD functions typically specify a large number of parameters, reflecting the scope of parameter operation. An automated test that uses a combinational strategy to match parameter function will require an exponentially increasing number of tests with increasing number of function parameters. A more efficient search strategy is presented in Chapter 8 that uses a genetic algorithm optimisation technique to determine a function parameter configurations that create identical model geometry. From this point matching parameters may be readily discovered, this method is demonstrated in Chapter 8.7 (Function parameter type heuristics). This approach is shown to reduce the number of function parameter variations trialled by an order of magnitude.

3 Previous CAD data interoperability research

In this chapter, the concept of model-based computer aided engineering data organisation is introduced, in order to place the requirements of data interoperability within the context of manufacturing organisation. Efforts to standardise the representation of this data are reviewed alongside the technical pitfalls that beset a prescriptive standard applied to CAD parametric shape representation. The Macro-Parametric Approach and other methods identify equivalent feature functions between vendor software, allowing transfer of design constraints between programs. The identification and mapping of similar features between CAD software is an intensive task and methods such as the Three-Branch Hybrid Feature Model propose geometrical and semantic matches to automate identification of feature mapping. Computational geometry routines are described that allow comparison of model geometry between heterogeneous software.

3.1 Product Lifecycle Modeling & Computer Aided Engineering

The role of Product Lifecycle Modelling is to integrate all aspects of a product, from design, manufacture, budgeting, to end-of-life disposal into a single framework that promotes efficiency.

This is a perspective distinct from Enterprise Resource Management, which gives a centralised view of enterprise activity. PLM is instead focused around the product, referencing aspects of the physical product geometry with associated information, such as materials, machining operations, geometric dimensioning and tolerances, suppliers and so on. Operational efficiency is judged to be closely tied to the integration of product information held within separate domains. Decisions that modify product or process parameters profit from an instantaneous assessment of costs involved. A complex product such as an aircraft, a large building or a car requires coordination between several enterprises that supply or integrate sub-components. In a modern manufacturing paradigm, the coordination between enterprise subcontractors and departments is a limiting factor of operational efficiency (Subrahim *et al*, 2005). Product Lifecycle Modelling formalises this information management with the stated goal of production efficiency.

These efficiency savings are to be realised via:

- Avoidance of information duplication.
- Avoidance of information loss.
- Information structuring.
- Formalism of information interpretation.

Each domain discipline views product model data from a different perspective and consequently information that appears relevant from one domain, becomes superfluous in another. As an example, a stress analysis carried out using Finite Element Analysis takes a CAD boundary model, strips out detailed features and generates a second model within the CAD boundary volume composed of cellular elements. Computer Aided Machining might take the same CAD model and focus exclusively on the features that dictate the machining methods.

As different commercial vendors devise software around the perceived requirements of client groups, there are no overarching conventions for information naming, semantics or formats. Researchers have adopted several of the information modelling standards to define frameworks to capture this information, such as Express, RDF, UML and OWL.

3.2 CAD standardisation initiatives

Various industry consortia, Standards Setting Organisations and commercial vendors have devised XML extensions to represent particular domain vocabularies such as ebXML, BizTalk, cXML, CML, Bioinformatics Sequence Markup Language (BSML), MathML, MatML, etc. The first widespread coordination to standardise product model data was led by the International Organization for Standardization (ISO), who developed the STandard for Exchange of Product model data, or STEP, as the ISO10 303-1 standard. STEP has evolved within a number of different Application Protocols that reflect the requirements of specific industry sectors, the most common being AP203 (Configuration controlled 3D design of mechanical parts and assemblies), AP239 (Product lifecycle support) and AP214 (Core data for automotive mechanical design processes).

STEP defines an EXPRESS language (ISO10 303-11) for the purpose of geometry specification. It does not readily allow the description of non-geometrical associated product data, nor is it suited to integration with other aspects of the design process (Fenves *et al* 2008; Negri *et al* 2015). By the time the STEP format coalesced into a published standard, commercial CAD software had developed newer and more popular parametric methods of design modelling that were not specified within the STEP format. Several research initiatives sought to formalise the parametric modelling paradigm, but to provide some context it is necessary to describe the difference between parametric modelling and the model creation and drafting processes that it superseded.

3.3 Parametric feature modelling

Computer Aided Design programs evolved from technical drawing software to the main design interface used in computer aided engineering. While the original two-dimensional CAD drafting programs allow convenient editing of geometrical drawing detail, subsequent three-dimensional CAD programs capture the concepts and constraints that specify the geometry of an artefact. The advantage afforded by so-called *parametric* CAD design is that changes to model parameters can be automatically updated to model geometry of the model, dispensing with tedious editing labour. As parametric CAD software is aimed at engineers or architects, these parametric constraints are defined within a set of modelling objects or *features* that correspond with familiar engineering design concepts such as flanges, webs, bosses or pockets. A model is constructed through sequential application of features, recorded as a feature history.

These features accumulate design decisions, which in turn refine the specification of a parametric model. Unlike a STEP boundary model, the geometry is rarely explicit, it is generated by an interpretation of features and their interrelationship in the same fashion that a scripted computer language might generate an output. This is different to the static description of a STEP boundary model. While a boundary model may be specified from the edges, corners, points and radii that constitute a surface, a parametric model defines a surface as a conceptual feature using a minimal set of constraints and parameters. A sequenced assembly of these geometric features constitute the entire model. The immediate advantage is that a change to a feature parameter does not require a manual modification of the rest of the model to accommodate this change, but can be regenerated using a geometry constraint solver. This advancement allowed designers to capture the important defining concepts of a design, what is referred to as "design intent" (Choi *et al*, 2002). In most cases there is no proven optimal method with which to define features. As a result, different vendors have used different sets of features with differing parameters and constraints, some explicitly defined, some implicit. The problem of defining a standard format becomes a problem of capturing these variations. For a detailed description of the so-called implicit and explicit constraints that constitute parametric modelling, please see Chapter 8.2, Explicit and implicit CAD model constraints .

Parametric feature based CAD programs have enjoyed commercial success, but transferring a parametric model representation between different programs is fraught with difficulty. While the specification of a geometric surface model may conform to several common standards, there is no equivalent for parametric features. Researchers have published several models embodying parametric feature representation, but none form the basis of commercial software. Some of the early initiatives to create a model encompassing a standard for parametric modelling are described in the next section.

3.3 Parametric feature modelling

3.4 Procedural feature models

The ENGEN data model, EDM, extends the ISO10 303-21 standard for exchange of product model data incorporating parametric feature representation (Shih & Anderson, 1997; Anderson & Ansaldi, 1998). Form Feature Information Model is another prototype feature representation developed by the Product Data Exchange Specification committee, that captures both explicit and implicit feature parameters (Shah & Mathew, 1991). Editable Representation or E-REP established procedural models built entirely of feature operations, (Hoffmann & Juan 1992).

Middleditch and Reade describe a geometric kernel specified by a hierarchical feature architecture with relationships defined by geometrical constraints (Middleditch & Reade, 1997). Wang and Nnaji describe an extensible modelling language, UL-PML that captures feature representation with both implicit and explicit constraint relationships. UL-PML is tailored to capture design concepts (Wang & Nnaji, 2004).

3.5 ISO 10 303 standardisation

The ISO 10 303 standardisation effort, spearheaded by the National Institute of Standards and Technology (NIST) responded to the commercial adoption of parametric modelling by defining further standards to encapsulate these properties (Kim *et al* 2007; Kim *et al* 2008). ISO 10 303-111 describes a standard defining design features, ISO 10 303-108 defines the parametrisation and constraints that support parametric features and ISO 10 303-55 defines a construction history that supports recalculation of model geometry following alterations to feature parameters. These STEP standards are the best-known standardisation initiatives to address product design formats and the development of data exchange and management within product lifecycle engineering (Pratt, 2005). While STEP product geometry standards (ISO10 303-103) only supports model geometry and topology, they are extended in Application Protocol 224 to incorporate a parametric representation with the capacity to define CAD features (Pratt & Kim, 2006).

3.6 Standardisation of parametric features

Even if widely adopted, it appears unlikely that these standards might allow direct translation between commercial CAD programs for the following reasons.

3.6.1 Numerical accuracy

The AP224 parametric standardisation project revealed problems once it was trialled with commercial CAD programs. Inconsistencies in geometrical tolerances between different CAD programs accumulated numerical precision errors. Differing internal representations of geometric tolerances coupled with different methods of constraint evaluation created problems of numerical accuracy between model translations (Kim *et al*, 2008). This is less of an issue with more recent commercial CAD programs.

Different CAD programs were found to use different schemas of absolute Cartesian coordinates and local geometry coordinates to represent aspects of geometry such as sketch planes. Different CAD programs use different numerical tolerance schema, causing a variety of errors in model translation (Qi & Shapiro, 2006).

3.6.2 Standardised feature taxonomy

STEP boundary representations rely on a common definition of geometric descriptors to represent surfaces, but there is no uniform or optimal definition of a feature (Bittner *et al*, 2005). Consequently complex commercial CAD features are rarely equivalent in either feature definition, or in their explicit or implicit parameters. There is no consistent semantic meaning to the features and associated parameters used between different CAD programs. There is no canonical standard of features allowing vendors to implement a palette of feature functionality that differentiates their product. The labels used to describe feature parameters and constraints do not have a consistent definition and frequently have subtle inconsistencies when used between differing CAD programs or even in different contexts (Maier & Stumptner, 2007). Researchers have responded to this ambiguity by adapting formal ontologies to capture explicit semantic meaning, Appendix B describes these efforts in detail.

3.6.3 Inconsistent definition of sequential and implicit feature constraints

A construction such as a surface boundary model may be entirely represented by explicitly defined values. In the case of a procedural model, constructed as a sequence of feature operations, the model parameters may be exclusively defined by the interaction of features with pre-existing features (Chapter 8.3, Sequential model defines explicit and implicit parameters used in parametric feature modelling, such as function dependence, prior selection and program architecture constraints). These interactions may lack any explicit or formal representation and may vary considerably between vendor programs. The most common implementation is a mix of both.

3.6.4 Inconsistent interpretation of sequential and implicit feature constraints

Existing commercial parametric feature architectures exhibit different behaviour interpreting multiple conflicting constraints. Where there is a combination of constraints that determine the geometry of a feature, the equations that use these constraints may not have unique solutions. The program heuristics used to select between multiple solutions represent an additional feature characterisation (Pratt & Anderson, 2001). Hoffmann and Juan observed that a procedural model might introduce constraints in a sequential manner during a modelling process, allowing single solutions to be found to parametric representations if there was a sequential modelling process with visual feedback. However a non-sequential model that embodies the same constraints may have several geometric solutions presenting the same issue of correct program selection (Hoffmann & Juan, 1992).

3.6.5 Inconsistent constraint combinations within generic features

Few features share the exact same parameters, semantics or functionality between differing CAD programs. Kim *et al* describe the *granularity* of feature semantics where the geometric modifications enacted by a single function in one program may require several sequential functions in another. A feature concept may be similar within two CAD systems, but one feature might encapsulate the functionality of two separate feature functions in these different CAD systems. There may not even be direct equivalence between certain CAD functions. Barber *et al* describe issues encountered with a limited subset of the most universally encountered features (Barber *et al*, 2010). As a consequence, CAD software vendor might diligently represent each of the functions that generate features within the ISO 10 303 standard, yet this representation may still be incompatible with the nearest function that another vendor has defined within the standard.

3.6.6 Unspecified semantic definition

The STEP EXPRESS language used to define the ISO 10 303 models is unsuited to capture of semantic detail required for feature function mapping, consequently it is unlikely that a logic reasoner might determine mappings between similar feature functions within the new parametric STEP standards. McKenzie-Veal *et al* experimented with the creation of ontologies for the purposes of CAD feature translation by extracting neutral STEP data from CAD programs for the purpose of geometric comparison (McKenzie-Veal *et al*, 2010). It was found that the STEP files created by nominally identical geometries within different commercial CAD programs were not equivalent, being either incorrectly parsed or having spurious data.

3.6.7 Unspecified labelling of feature entries

Bidarra and Bronsvoort describe the problems of maintaining and duplicating procedural feature model in greater detail, identifying problems relating to the chronological order in which features are created, where variations in sequence may create differing end models (Bidarra & Bronsvoort, 2000). The so-called *persistent naming problem* is also described: features may be defined relative to pre-existing features that have subsequently been deleted or modified. This unintuitive issue that arises from sequential geometric operations merits an explanatory paragraph, see The persistent naming problem.

The most up-to-date ISO standard for the exchange of parametric models, AP203 (second edition) requires a translator to convert from the CAD model to this neutral format. While most common commercial programs will export to this neutral format, the quality of models translated from one CAD system to another via STEP AP203 is unsuited to complex models at the time of writing (Ćuković *et al*, 2017).

The introduction of parametric capability to the ISO STEP standard has not yet led to a commercial adoption of the application protocol as a native feature standard. Nor has the availability of published feature ontologies encouraged widespread adoption of a prescriptive research model. Competitive market forces dictate that a commercial program can read the data formats of other vendor programs but avoids allowing its own format to be read by others (Katz & Shapiro, 1985).

3.7 The persistent naming problem

The *persistent naming problem* is a topological challenge faced by parametric CAD programs. Recall that the parametric procedural model is composed of an accumulation of feature operations. If the surfaces, or edges that are selected to form the basis of a feature creation operation are subsequently modified or deleted, the naming scheme to reference them becomes an issue. The persistent naming problem is exacerbated by the potential of multiple different procedural histories that construct identical models. In many cases there are more than one method to construct a desired model alongside more than one sequential order of operations to create a model. The Macro Parametric Approach described in the following section encountered errors with undefined combinations of functions that would result in an incorrect feature selection (a dependency issue). This highlights the intensive labour requirements in mapping combinations of functions to identical geometric models.

3.8 The Macro-Parametric Approach

The *Macro-Parametric* approach exploits existing CAD *Application Programming Interfaces* to avoided translating model features to a neutral feature format. This approach determined a common set of function commands, or "neutral modelling commands" to translate identical parametric CAD models between popular commercial CAD programs. Choi *et al* reasoned that the sequence of feature creation instructions used to create a parametric CAD model could be mapped to equivalent command sequences to create geometrically identical models within different CAD programs, while retaining the extra parametric information, see Figure 5 (Choi *et al*, 2002; Mun *et al*, 2003).

While all commercial parametric CAD programs are primarily designed to allow a user to model using a visual user interface, there is invariably a degree of access to the program internal data structures via a programming interface, an API. These interfaces are generally used to allow third-party applications interact with CAD programs, or to automate repetitive design tasks. The developers of the TransCAD macro-parametric approach use the internal scripting files generated by a CAD program (Choi *et al*, 2002).

The procedural sequence used to construct a model within a CAD program is recorded in a script using CAD API commands. An identical model can be recreated within the program via sequential execution of the script commands. The macro-parametric approach is to translate between the script representation of a model in one CAD program to a similar script in a different CAD program that will recreate an identical model. Consequently this approach requires that the macro commands of each CAD program are mapped for equivalence.

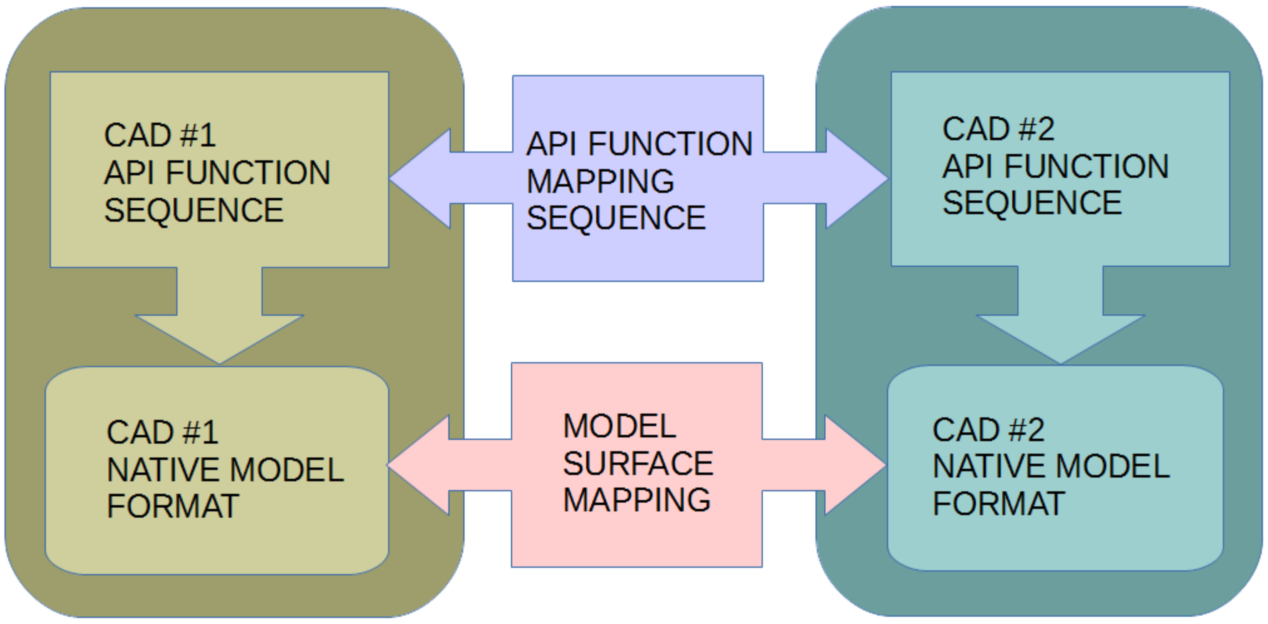


Figure 5: Macro-parametric feature mapping to generate equivalent CAD models.

The researchers identified a set of 167 feature commands common to six popular commercial CAD programs. It was found that implicit dependencies, such as a selection of features, surfaces or edges prior to a command operation required an extra routine to determine the associated explicit command. The researchers identified a number of these "indirect translation" requirements related to positioning with the CAD coordinate system and commands with no direct equivalent within the receiving system. Li *et al* discovered inconsistencies arising from topological errors, apparently from incompatible naming schemes (Li *et al*, 2010).

The TransCAD project revealed several subtle difficulties with the macro-parametric approach. The first version of the universal neutral command set used a topological scheme to reference the model surfaces. This approach was susceptible to the persistent naming problem (referenced in The persistent naming problem). CAD programs such as CATIA and Pro/ENGINEER use a topological naming scheme and label surfaces according to their relation to other surfaces. Other CAD programs, such as Solidworks and UG use a geometry-based naming scheme for model surfaces and features. Surfaces referenced by name may suffer ambiguity when split or merged in subsequent modelling operations, this is remedied with a geometry reference such as a Cartesian point known to be coincident with the surface (Song & Han, 2010). Farjana *et al* extend this scheme, introducing a name taxonomy that preserves name history with modelling history (Farjana *et al*, 2016).

The coordinates used to define features vary between different CAD programs, certain programs use screen coordinates, others use combinations of 2D sketch coordinates, feature entity names and 3D coordinates. The TransCAD project incorporates a CAD geometry kernel into the translation operation in order to allow conversion between topological and geometric coordinate references. A conversion from a CAD model script that records entity names to a CAD model script requiring geometric coordinates requires the translator to generate an internal CAD model representation that can compute the missing data (Choi *et al*, 2002). The Macro Parametric Approach incorporates a Geometric Modelling Kernel (pre-existing software routines to process geometry information) to determine the geometry of some of the feature parameters and constraints that are not explicitly defined within feature commands.

Macro-parametric research describes syntactic differences between the names of variables used in API command parameters. A function parameter terminology may have identical semantic meaning, but have dissimilar labels specified by disparate vendor API terminology. The TransCAD project relies on human intervention to determine identical API function semantics, as do many commercial CAD translators. The neutral intermediate format used in the TransCAD reflects the labour of creating a semantic mapping between each additional CAD program and the existing set of CAD programs. With a neutral intermediate format, a single translator is required, with peer-to-peer translation, a separate translation is required between a new CAD program and every other CAD program.

3.9 Universal Product Representation

Rappoport *et al* envisaged a *Universal Product Representation* method that addresses the practical difficulties of finding geometrical equivalence between geometry operations of different CAD programs. Universal Product Representation or UPR, is another Feature Based Data Exchange method that translates from one CAD program to another via a common representation (described as a *star architecture*), but unlike the TransCAD neutral feature set which represents the intersection of CAD feature functionality, UPR common representation is described as a union of CAD feature sets (Rappoport, 2003). This architecture is justified by the reduced labour involved in determining one-to-one function matches. The UPR method is also distinguished by trial-and-error methods that automate some function parameter matching tasks.

If a function that performs a geometric operation does not have an exact equivalence within the API of a second CAD program, the UPR architecture attempts different variations of the function parameters to reach a geometric equivalence between the operations within both programs. UPR is reliant on geometric checking for equivalence. Spitz and Rappoport detail three mechanisms whereby the geometric equivalence may be checked within the source and target CAD programs (Rappoport *et al*, 2005). Each of these methods rely on the pre-existence of specific operations within the CAD programs to allow this verification. Parametric feature operations are replicated with equivalent non-parametric geometric operations in a sequential procedure. If a parametric operation is completely subtracted from the equivalent geometric operation with a boolean operation, the parametric feature is judged to be functionally equivalent. This process allows for a model reconstruction via command script in a target CAD program. The UPR methodology forms the basis of a commercial CAD translation service (Iti-global.com, 2018).

While the TransCAD macro-parametric approach relies on manually identifying mappings between heterogeneous CAD API functions and parameters, the UPR ranks function matches according to probability. Rappoport observes that CAD API functions may not behave in a similar fashion in all modelling situations; the interaction between a feature operation and all pre-existing CAD models may be difficult to predict. Unpredicted feature behaviour can be rectified using substitute modelling options that preserve the model geometry but not the parametric definition. Rappoport describes methods to select correct edges and surfaces for feature operations in CAD models that may subdivide surfaces differently (Rappoport *et al*, 2005; Rappoport *et al*, 2006). Points are projected to the surfaces undergoing transformation in the CAD models to determine if there is an adequate geometrical correspondence between the selection in the target CAD model and the source CAD model. UPR architecture maintains a data structure recording a measure of geometric validity for trials with different functions. UPR forms the basis of the TranscenData Proficiency translation software and while the method is a commercially viable approach to CAD model translation, much of the implementation is withheld intellectual property.

3.10 Three-Branch Hybrid Feature Model

Tessier and Wang define an ontological data structure that captures explicit and implicit parameters derived from procedural models (described as reference attributes) and geometric verification data (Tessier & Wang, 2013). As with UPR, the research describes a machine learning method to identify similar features between CAD systems. Unlike UPR, which uses statistical machine learning based on a geometric validation of prior matches, the CAD feature ontology described employs semantic reasoning to find feature matches. This inference method relies on a discovery of set of rules that uniquely identify semantically equivalent features, but must be established by human observation. The inclusion of geometric data is limited to available B-rep surfaces, vertices and features available via a CAD API (Tessier, 2011). This approach is unique in using semantic and geometric feature characteristics in an effort to automate feature mapping between CAD systems.

3.11 Bidirectional Hausdorff metric

Zhang *et al* use bidirectional Hausdorff distance as a measure of geometric similarity (Zhang *et al*, 2016). In this instance this is the measure between discrete points on the surface of the source CAD geometry compared against a set of discrete points on the surface of the target CAD geometry. If each source point is compared against the set of all target points, and the minimum displacement is taken from this set, a Hausdorff measure is the maximum value of the set of minimum displacements.

This metric captures the relative orientation of source and target geometries as captured by discrete points. The Hausdorff distance has the unintuitive property of being asymmetric; the Hausdorff distance may change if the source and target points are exchanged. The bidirectional Hausdorff distance is the maximum value of both unidirectional Hausdorff distances. Zhang uses this metric as a basis of an iterative estimation of control points to match splines between two CAD programs.

Given point sets $A = \{a_1, a_2, \ldots a_n\}$ and $B = \{b_1, b_2, \ldots b_n\}$ in E^2, then the *one-sided Haussdorff* distance between A and B may be defined as

$$\delta_H(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

The *bidirectional Hausdorff* distance between A and B may be then be defined as,

$$\delta_H(A, B) = \max\left(\delta_H(A, B), \delta_H(B, A)\right)$$

3.12 Representative proxy model and query protocol

Hoffmann defines a proxy model based on the concept of the master model traditionally used as a reference for geometric dimensioning and tolerances (Hoffmann *et al*, 2014). This proxy model is defined by the semantic and geometric information of a reference model up to a specified geometric tolerance. A limited set of queries that possess a common definition between source and target CAD may generate this representative proxy model which is tailored to the requisite geometric tolerance of the target CAD. This approach avoids the limitations of translation methods, namely that,

1. The proxy model is limited to the queried information, there is no requirement for all aspects of the model to be represented in both systems. This avoids the difficulty of systems that have dissimilar model representation as only the queries must be interoperable.
2. The disparities of precision between CAD programs or other systems is explicitly defined.

Hoffmann formalises acquisition of model geometry around a minimal set of interoperable queries. These interoperable queries are defined in order of dependence as follows:

1. A query requesting model precision.
2. A nearest point query returning the model point or points closest to a specified point.
3. A point membership query, returning positive if all points within a region defined by the model precision lie within the reference model.
4. A point on or in the proxy model returns the differential information of the proxy model, such as the tangent curvature or torsion (defined as the k-dimensional submanifold of the proxy model).
5. A surface (or r-simplex) query that returns the collection of intersected surfaces (or r-simplices within defined model precision).
6. A point on or in the proxy model returns the CAD model label and characteristics at that point.
7. A point on or in the proxy model returns the CAD labels of all adjoining surfaces or parts.

3.12 Representative proxy model and query protocol

Hoffmann proposes that that the most computationally economic approach to model geometry interoperability is to probe CAD model geometry using intersection queries. An algorithm is outlined, sampling a proxy model with a grid of points to determine a form of voxellised model representation. Interoperable point membership queries determine whether points lie within or on the surface of the proxy model. This approach may be partially implemented across all the CAD programs that were tested during the course of this research. Every program will return the point value at the intersection of vectors or rays with model boundary surfaces (see Table 1). This common feature enables a statistical comparison of model geometry between heterogeneous CAD programs. This geometry testing method is described in detail in Chapter 6, Boundary Surface Geometry Comparison.

CAD program name and version	**API languages**	**Command**
Dassault Solidworks 2012	VB & Automation	`ModelDoc2.GetRayIntersectionPoints()`
McNeel Rhino 5.0	Automation Python	`Rhino.ProjectPointToSurface()`
Autodesk Inventor 5.3 2014	Automation Python	`SurfaceBody.FindUsingRay()` `partDoc.ComponentDefinition.WorkPoints.AddByCurveAndEntity()`
Siemens NXOpen Python 10	Python	`NXOpen.UF.Curve.Intersect(curve, entity, refPoint)`
Autodesk Fusion360 2018	Python	`line.worldGeometry.intersectWithSurface()`
Pro/ENGINEER Wildfire 2002	VBA	`ProSolidRayIntersectionCompute()`
CATIA V5R18 API	VBscripting /VBA	`CATIA.ActiveDocument.Part.HybridShapeFactory.AddNewIntersection()`
FreeCAD 0.17	Python Automation	`App.activeDocument().Common.Shapes`

Table 1: a selection of Common CAD program commands yielding surface coordinates via curve intersections.

3.12 Representative proxy model and query protocol

3.13 The state of the art

To recap, translating parametric CAD models while retaining design intent is complicated by semantic and structural heterogeneity between CAD programs. The API functions and parameters may use a different syntax, but a more intractable class of semantic heterogeneity is an indirect or singular mapping of feature operations or their defining functions between programs. In recent research efforts, these issues are addressed with substitutions of sequences of function operations that have an identical geometrical outcome.

The issue of structural heterogeneity is where an apparently successful semantic mapping results in disparate geometry. A common example is a blend, or a radiused fillet between model surfaces; different CAD programs will make different assumptions of blend behaviour where several surfaces meet. Each CAD program has several unique subtypes of fillet behaviour with further options that guide the creation of fillet geometry. Product differentiation will mean that the descriptions and varieties of fillet behaviour are only consistent across the most simple of geometries. The geometry created by a CAD fillet feature is not guaranteed consistent between all CAD programs for all configurations of model surfaces. As the algorithms that generate these feature geometries are proprietary, the only consistent means to check consistent geometry between different programs is a comparison of model geometry.

Geometric validation protocols are introduced to verify these substitutions and each mapped feature operation outcome. The Three-Branch Hybrid Feature model, the Universal Product Representation and the Domain Independent Form Features proposed by Gupta *et al* are representative examples of methods that test for geometric conformity between mappings using a variety of measures (Gupta & Gurumoorthy, 2008).

Prescriptive universal formats that provide a neutral standard intermediate representation between native CAD formats are limited to the adoption of these formats by commercial vendors. Translating an internal semantic model representation to a perpetually outdated standard creates unique difficulties and frequently results in incomplete representation (Barber *et al*, 2010). While commercial vendors compete on the basis of novel capability, intermediate formats have limited viability.

The Macro-parametric approach is another example of a neutral intermediary format, which suffers similar issues of semantic mismatch, coordinate scheme mismatch and procedural selection problems arising from persistent naming type errors (Li *et al*, 2011). While the so-called “star architecture” employed by a universal intermediary representation is an economic means of mapping translations between multiple systems, it represents an intersection of the set of common system functionality and thereby reduces the scope of accurately mapped features that a peer-to-peer mapping allows. The UPR system is described as an attempt to create an intermediate representation that is a union of system features (Rappoport, 2003). The labour required to match, validate or devise correction routines for feature function instances limits the application of existing translation approaches.

Further translation complexity arises from the constraints, selections and datums used to generate feature operations. These elements may be explicitly defined, or may be implicitly defined via the sequence of feature operations (Pratt, 2004). Other researchers variously reference these classes of implicit and explicit elements as procedural and declarative specification, first and second-order information, reference and parameter attributes.

While the Macro-parametric approach uses an internal geometric model to generate implicit references, other methods capture the requisite data during a sequential reconstruction of the source CAD model. Recent research has proposed machine reasoning to identify semantic equivalence of CAD features described within ontologies. To date, the protocols used to compare the geometry have been limited to the topological representations based on model faces and vertices commonly adopted for translation between CAD and CAM software.

A common feature of the research literature to date is an absence of tests carried out on existing commercial CAD programs that measure the efficiency of published CAD interoperability methods. The semantic and geometry comparative tests in this research address this omission, the demonstration of a Genetic Algorithm used for parameter matching uses the interface to two CAD programs to derive test results.

One significant area of prior research that has been deliberately omitted from this chapter is the development of formal ontologies that capture the semantics used in heterogeneous CAD programs. These formal descriptions of semantic relationships promise to automate the search and checking of CAD feature equivalency, however the task of defining commercial CAD feature semantics within these research ontologies has not been widely addressed by commercial vendors. Details of these forays are provided in Appendix A.

This chapter has introduced the techniques used for interoperability between data representations within different CAD software. The novel parametric modelling method removes the tedious requirement of updating model surfaces with design changes as models are generated from design constraints and relationships that are incorporated into the modelling process. Interoperability then becomes an issue of transfer of these constraints and design decisions between different vendor CAD programs. Standardisation between these design modelling architectures is difficult as each vendor chooses to encapsulate their feature operations in subtly different fashions. Prescriptive standardisation in the form of ISO specification and “top-down” ontologies have proved ineffectual to date. Contemporary research is now focused on creating mappings between feature operations by a process of discovery. The remaining open question is how the laborious process of determining function equivalence might be automated. The research presented in this thesis introduces a hybrid method to resolve this question.

4 Prior research on short text similarity measures

The following chapter examines the application of semantic matching to short texts, such as those used in the function names and descriptions found in the text accompanying application programming interface libraries. This technique is used to assign the probability of heterogeneous API libraries routines having an equivalent functional operation. The two broad approaches to identifying words of similar meaning are summarised, that of human-compiled and machine-compiled corpora based on statistical assumptions.

4.1 Semantic mapping within CAE systems

This research determines the feasibility of combining semantic and geometric comparison techniques to map similarities between the features in a Computer Aided Design program, and by extension, the functions in Application Programming Interface of a CAD program. Semantic matching, in this case is finding similarity in the meaning of the words used in function names and descriptions.

Research into mapping database entries has generated the field of ontology alignment, or ontology matching. Successful strategies rely on a combination of word meaning comparison, structural organisation matching and data type matching (Bernstein *et al*, 2011). These generic techniques have been appropriated for matching the product description databases used in Product Lifecycle Management. There is an information advantage to storing product data in a format that can be accessed by different production departments (e.g. design, engineering, accounting, subcontractors, suppliers) and a format that is accessible to different software. Product information databases do not have an over-arching standard or a dominant vendor format, so research efforts are directed towards integrating information stored on different files and databases.

It is helpful to distinguish between 'syntactic matching' and 'semantic matching' at this point. Syntactic matching is a means to correlate descriptive labels used in database schemas, semantic matching is the effort to identify identical meaning between items. Semantic mapping is commonly achieved by searching for a correlation between descriptive texts (Giunchiglia & Shvaiko, 2007).

Some researchers have used generic schema matching methods to identify mapping between articles in PLM databases (Dalianis & Hovy, 1998; Yeo, 2009), text descriptions of the geometric model features, feature function names and feature dependence are compared. The semantic text matching methods are derived from from simple syntactic matching. In other words, determining whether two words are identical or are recognised as synonyms. Broadly speaking, there are two approaches to finding semantically similar words or text. The first is a manually compiled corpus such as the Princeton WordNet® corpus (Miller, 1998). This particular corpus lists the synonyms, or so-called *synsets* of a word categorised by the usage of the word (a word may be used as a noun, verb, adverb, adjective, and so on, but not distinguished by any change of spelling).

The second is to generate semantic classifiers from statistical analysis of words in training texts. This is the basis of significant research effort in the field of document retrieval. The advantages of generating a corpus from a set of documents rather than using a pre-compiled and checked manual corpus is that a more precise corpus for a particular application may be created using a document set with a narrow range of topics. A machine-compiled corpus may also be readily updated or regenerated over a different set of documents (Senellart & Blondel, 2008).

This manual and machine compiled distinction between semantic matching methods each form the basis of of a multitude of semantic matching methods. These methods are described in the following sections, Figure 6 shows a diagram of the methods and their relationship to manual or machine-compiled corpora.

4.2 Descriptive text labels used within Application Programming Interfaces

In the case of matching program API functions, the text consists of short phrases. There is the function name, normally indicative of the action of the API function and composed of several words joined together, there are the function parameter names that follow the same conventions and there is generally a short explanatory text that provides a description of the function within an accompanying help file. These short texts contain more words than a label in a database schema, but less words than the average document size encountered in document retrieval methods. Function names are information dense, one cannot presume to disregard the relevance of any contained word, therefore these short phrases appear unsuited to methods such as *Latent Semantic Analysis*, that discard information in documents. Semantic matching methods are not optimised for matching short texts; the following description of the common methods reveals why.

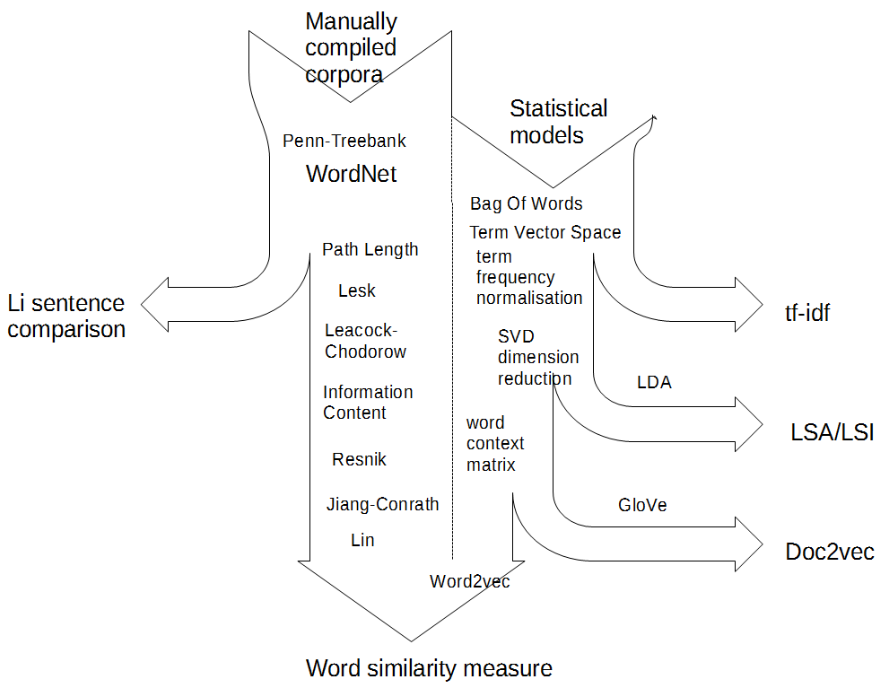


Figure 6: a taxonomy of semantic matching methods

4.3 Vector Space Models and statistical concept matching

Most Natural Language Processing strategies to discover relationships between words are based on the simple premise that words of similar meaning occur in close proximity to each other within a text. This heuristic, the *distributional hypothesis,* underpins algorithms used in document retrieval (Firth, 1957; Harris, 1954). Miller refines this idea with the observation that word similarity is proportional to context similarity within a document (Miller, 1991). Pederson further discriminates between a *micro-context* and a *macro-context* based on whether similar words are likely to be found on the same sentence or in the same document (Pederson, 2008). Simple implementations of these heuristics record the individual frequency of words within documents. A *Bag of Words* model is the unique set of words used in a document, merely recording the number of times each word occurs. Such a general measure is useful for tasks like email spam filtering where it is relatively efficient to compare these representations of documents against known spam documents (the degree of overlap between documents is the *Jaccard* coefficient). This numeric representation can also be represented as a *Term Vector Model* or *Vector Space Model*'. Typically a term is considered to be the most atomic unit, a word or a phrase. If each term is considered a dimension, then a document is a vector of terms within this multidimensional space. This representation allows the use of efficient vector comparison techniques over such large and sparse models, individual vectors that are parallel are considered to be most similar, where the cosine similarity measure is commonly used (Chen & Lynch, 1992).

The most common Vector Space Model uses another heuristic to further refine frequency based vector models; inverse document frequency weighting (Salton *et al*, 1975). The raw term frequency values are usually weighted and normalised to account for words that are most commonly encountered across all the documents and words that are encountered multiple times within a single document. Common words, such as "if", "and", "that" are less useful in identifying documents than uncommon words, their frequency is weighted according to an inverse power law and normalised within the local document in the *Term Frequency – Inverse Document Frequency* model (Crouch, 1990).

These weighted vector space models are large and sparse, Foltz *et al* have used Singular Value Decomposition to reduce the number of dimensions in a Vector Space model to create what is known as a *Latent Semantic Analysis* or *Latent Semantic Index* model (Foltz *et al*, 1998; Deerwester *et al*, 1990). SVD decomposition has the effect of creating abstract vectors that result in a more compact model with better defined concepts and quicker query times, but these vectors do not necessarily correspond to human-understandable semantic relationships.

All these common vector space methods rely on heuristics that do not work particularly well with short phrases or the text in API documentation. Vector Space models do not identify polysemy, the property of a word to have more than one meaning depending on context and part of speech (e.g. "crane" may be a bird, a tower for lifting objects, or the act of stretching one's neck). API names tend to follow a native convention; words are re-used in the same context and synonyms avoided to minimise confusion. These local schemes do not transfer across different API from different vendors, function names may be deliberately changed to avoid intellectual property infringement with a similar function of a competitor. Dimensional reduction, as in the case of LSA is seen as one method to reduce incidence of polysemy. Another approach has been to couple a corpora that documents known polysemous words with a vector space model. Passos and Wainer show that integrating compiled corpora such as WordNet to determine polysemy is ineffectual (Passos & Wainer, 2009).

TF-IDF or LSA methods identify concepts via co-occurrence of similar words. This approach is effective in long texts where there is the luxury of describing concepts with an abundance of words and their synonyms. Where a concept is represented in a single terse word, dimensional reduction can discard important data. Short sentences contain a higher information density, so standard techniques that discard words of lesser relevance are unsuitable (i.e. vector dimension reduction to a limited number of identified concepts). Methods that discard words or that disregard synonyms are unsuited to the short, information dense phrases as used in function naming convention. Approaches that would be considered computationally expensive on large documents are feasible for phrases of several words, such as the naming conventions used in descriptive API function names.

4.4 The WordNet corpus and word pair similarity

The word relationships in the WordNet corpus are designed to identify the relationships between words, but not that of sentences. Li *et al* use the WordNet corpus as a basis for matching short texts, the similarity metric used is the path length between words to a common ancestor word (Li *et al*, 2006). In the WordNet corpus, the information content of a word is determined from its relative precision. Words are organised into hierarchical member sets where a word is assigned as a specific instance of a more general word. WordNet path length calculates the number of hierarchies traversed from one word to another as a measure of similarity, (e.g. "finger" is a hyponym of "hand", as is "thumb", therefore the path length between "finger" and "thumb" is two). Li also calculates the absolute depth of the hierarchies of the words, similar to the Power Law weighting of TF-IDF, more precise words have a higher weighting. This method is the only one to assign a score to the similarity of the order of the words in the texts under comparison ('inflectional morphology'). The method does not account for a bipartite, one-to-one matching of semantically similar words. If an exact word match is not found between the two sentences under analysis, the semantically closest word is selected. The algorithm is a simple proof of concept and there is no provision to find the best overall combined semantic match. The selection of the highest ranked word is not related to word order and it is also unclear how the algorithm avoids multiple selection of the same word. This algorithm was one of a number that were tested for suitability of matching API texts.

Lakshmi and Mohanty describe a semantic matching method aimed at discovering compatible World Wide Web service functions. The most promising combination of words is formulated as a maximum weighted bipartite matching problem, the potential matches are found using a density based clustering algorithm, DBSCAN (Lakshmi & Mohanty, 2015). Dong *et al* use a similar density based algorithm.(Dong *et al*, 2004). Yeo references two other algorithms that determine the best combination of semantic matching scores in a short phrase (Yeo, 2009), the Gale-Shiply matching algorithm and the Munkres-Kahn, or Hungarian optimisation algorithm. Aguilera *et al* describe a similar semantic comparison between Web services, again using a unique method to combine a keyword based search with user and provider data using a matchmaker algorithm (Aguilera *et al,* 2007). Paik *et al* describe a Support Vector Machine classifier that aggregates a number of different semantic measures for matching WWW service functions (Paik *et al*, 2010). These and other research efforts are directed at functions distinguished by a small number of keywords such as those used in the benchmark test suite OWL-TC, which contains between three and seven words on average (Klusch & Kapahnke, 2019). There are no

4.4 The WordNet corpus and word pair similarity

comparative studies of contemporary semantic matching methods used over relatively long phrases within large collections. While some research is directed at efficient matching algorithms within large collections, it is not adapted to the exponential increase in computation required by comparatively long phrases.

For example, during the course of the research, it was found that the Li algorithm, while best suited to comparing short sentences, was impractically slow for comparing the volumes of text found in commercial program APIs. A more efficient algorithm was developed based on Greedy matching selection of promising semantic matches that is detailed in Chapter 5.2, Combined scoring for short text semantic comparison.

4.5 WordNet similarity measures

An introduction to the structure and similarity measures of the WordNet corpus ontology is required. The words in the WordNet corpus are organised in what is termed a *subsumption* hierarchy, *hypernyms* are a specific subset of their *hyponym* parent. The root concepts or *unique beginners* are the most general concepts from which each word is derived. Each word has several other relationships defined with other words besides synonymy; several *meronymy* relationships ('part of', 'component of', 'substance of') and *antonymy* relationships ('complement of').

Most of the metrics that determine the relationship between words are based on the relative path length traversed via a common conceptual ancestor and the absolute depth of this path within the WordNet corpus. To establish a path length between two words, a root concept, or *subsumer,* that is common to the concepts in both words is identified from the structured lexical hierarchy. The path length is then the count of edges between word nodes that lead between the words under consideration via the subsumer. Where there are more than two concepts, or synsets, embodied in the two words then there are multiple subsumers, giving rise to least common subsumer measures based on the shortest paths between words. As a word may have several meanings, depending on context or whether it is used as a noun, or another part of speech, it is more accurate to express semantic similarity as a comparison of concepts that have a unique meaning. Five of the highest performing measures are selected for evaluation (Budanitsky & Hirst, 2001). These are detailed in the following sections.

4.5.1 Leacock and Chodorow similarity measure

Leacock and Chodorow normalise the compared word concept path length (c_1, c_2) with the overall depth, $D_{WordNet}$ of the WordNet taxonomy (Leacock & Chodorow, 1998). This value is then weighted use the same logarithmic scaling. Intuitively, a linear descent into the hierarchy of increasing precision is matched by a logarithmic increase in the instances of precise word concepts.

$$sim_{LC}(c_1, c_2) = -log\frac{length(c_1, c_2)}{2D_{WordNet}}$$

4.5.2 Wu and Palmer similarity measure

Wu and Palmer's scaled measure combines the depth of the most specific common concept with the path length measure. It is the combined individual path distances of the shared word concepts normalised by the absolute depth of the specific common concept within the concept hierarchy. This distance measure is reformulated to give a similarity measure (Wu & Palmer, 1994).

$$sim_{WP}(c_1, c_2) = \frac{2 \times depth(lso(c_1, c_2))}{len(c_1, lso(c_1, c_2)) + len(c_2, lso(c_1, c_2)) + 2 \times depth(lso(c_1, c_2))}$$

4.5.3 Resnik similarity measure

The WordNet subsumer hierarchy does not capture the precision of a hypernym in its classification schema. Resnik introduced a separate value, the *Information Content* , $P(c)$, derived from the frequency of a term encountered in a corpus. A Power Law relationship of precision with respect to frequency is used, similar to the Term Frequency, Inverse Document Frequency weighting in Vector Spaces. The Resnik measure weights the value of the lowest common subsumer or the most precise concept common to the compared word concepts with a negative logarithmic value (Resnik, 1995). If the Information Content measure of a word, c, is defined as the frequency of this word within the corpus, N,

$$P(c) = \frac{freq(c)}{N}$$

Then the Resnik similarity measure can be defined as,

$$sim_R(c_1, c_2) = \max_{c \in S(c_1, c_2)} [-logP(c)]$$

Where $S(c_1, c_2)$ is the set of concepts that subsume c_1 and c_2.

4.5.4 Lin similarity measure

Lin refines the Resnik approach, using the Information Content value of each word undergoing comparison to normalise the Resnik value (Lin, 1998). The definition of word Information Content is refined, based on three assumptions:

1. Similarity of two concepts is proportional to commonality of the two concepts. An increase of related concepts common to both words is akin to an increased similarity between both words.

2. The inverse is also assumed, namely that the similarity of both concepts or word senses are inversely proportional to the number of differences they share.
3. The maximum similarity is the identity case and is defined as 1.

Lin extends Resnik's measure with the inclusion of the total amount of information that each concept or word represents alongside the number of common concepts shared. This gives a measure,

$$sim_L(c_1, c_2) = \frac{logP(common(c_1, c_2))}{logP(description(c_1, c_2))}$$

Where $description(c_1, c_2)$ is the measure of total information content of a concept. Using Shannon's information entropy definition that defines the information content of a message as the negative log of its probability allows the information content measure to be reformulated as $-logP(c_1) - logP(c_2)$, giving an alternative expression for the Lin similarity measure,

$$sim_L(c_1, c_2) = \frac{2 \times logP(lso(c_1, c_2))}{logP(c_1) + logP(c_2)}$$

Where $lso(c_1, c_2)$, the lowest shared ordinate, represents the union of the concepts shared by c_1 and c_2.

4.5.5 Jiang and Conrath distance measure

Jiang and Conrath also use the Information Content of each word from a corpus value, along with the lowest common subsumer derived from a hierarchic ontology (the Resnik value). The value is the sum of the respective information contents less the value of the most informative common subsumer. This metric is constructed on the premise that the semantic difference between a child concept and a parent concept is proportional to the difference in their information content (Jiang & Conrath, 1997).

$$\begin{aligned} dist_{JC}(c_1, c_2) &= IC(c_1) + IC(c_2) - 2 \times IC(lso(c_1, c_2)) \\ &= 2logP(lso(c_1, c_2)) - (logP(c_1) + logP(c_2)) \end{aligned}$$

4.5.6 Word Embedding and the word2vec similarity measure

The other word comparison measures derive from recent machine-compiled corpora based on the word2vec algorithm, one of a family of 'Word Embedding' techniques (or 'Deep Learning') that train a shallow neural network to represent word frequency found in a collection of training documents. This method is similar to the Latent Semantic Analysis method in that it creates a reduced dimensional representation of the word occurrence Vector Space.

Mikolov *et al* describe an asynchronous stochastic gradient back-propagation algorithm used to create the weighting of the two layer neural network (Mikolov *et al,* 2013). In essence, the neural net is trained to recreate the sampled word from a window of surrounding words. Two window sampling methods are described, the 'Continuous Bag Of Words Model', as with the Bag Of Words model, the word frequency in the moving word sampling window is used. With the alternative 'Skip-Gram' word window. the relative positions of the surrounding words contribute to the weighting (weighting is proportional to the distance from the word being modelled). A *softmax* log-linear classification model is used to weight the hidden Neural Network layer (Rong, 2014). The Word2vec algorithm creates a word vector matrix similar to the term vector space, and a word context vector matrix. Pennington *et al* identify this objective matrix as a co-occurrence matrix which is more explicitly defined in a comparative study of the Stanford GloVe algorithm (Pennington *et al*, 2014). On closer inspection, training a network to replicate the response of an input requires that each weight that is not part of the input is recalculated according to the log-likelihood normalisation of the softmax function. For the network to generate a convincing match, not only must it register a similarity to the surrounding concepts or words recorded in the vicinity of the word in question, it must be able to reject all words that are not considered a match. For a corpus of any appreciable size, this is a prohibitively expensive calculation. Mikolov uses a novel technique to reduce this calculation; the response of the network to a word is reduced to a relatively small sample of words chosen to represent the target word, alongside a sample of words that are determined to have no correlation with the target word, so-called *noise words*, used in *negative sampling*.

4.6 Distributed Memory Model of Paragraph Vectors

Le and Mikolov describe an extension to the learning vector representation of words used by the word2vec algorithm that allows comparison between paragraphs and texts (Le & Mikolov, 2014). This is termed the *Distributed Memory Model of Paragraph Vectors* (PV-DM), or more commonly, *Doc2vec*. The implementation is simple; a paragraph vector is introduced alongside the word vectors and trained alongside the word vector framework. The paragraph vector is identified by a tag referencing the individual document or paragraph token in the same manner as the word vector is labelled by the word string. While the word vectors are related to the contextual concepts encountered in the entire set of training documents, the paragraph vectors are only related to the concepts within the tagged paragraph. This can be seen as adding an extra dimension to the concept vector space, allowing retrieval of paragraphs related by similar concepts. Both the Word2vec and the document matching relative Doc2vec have efficient implementations in the Gensim library (Rehurek, 2010). In the case of the Word2vec algorithm, it allows a semantic similarity matrix to be constructed to test with the Greedy word combination.

4.7 Short text matching techniques to date

The STASIS short text semantic similarity measure described by Li *et al* uses a combination of structured lexical word relationships, word positional relationships and a word frequency metric to generate similarity scores between sentences (Li *et al*, 2006).

Relationships between words in sentences are scored according to whether they share a common synset, or if these word synsets share a common word, or failing that, the minimum path length between words. The relative depth of the subsumer words within the hierarchical semantic net also contributes to the individual matching scores as it is reasoned that words higher in a hierarchy are more general and have a lesser relationship. The word corpus frequency relates to how often words are encountered within a corpus. Infrequent words are assumed to have a more specialist meaning and are awarded a higher score when matched. Another measure is introduced to compare the similarity of the order of matched words within sentences under comparison. These different semantic and syntactic metrics are subsequently combined.

Islam and Inkpen extend this multi-factor semantic and syntactic comparison, introducing a measure of partial word string similarity (Islam & Inkpen, 2008). They also use a different method of semantic matching, based on the *Pointwise Mutual Information* measure, a different approach to the distributional hypothesis where the probability of two words appearing in a text together is normalised against the probability of each individual words appearing (Church & Hanks, 1990). Guo and Diab extend the limitations of the LSA model using a negative sampling technique (Guo & Diab, 2012). Other short text methods use combinations of techniques, combinations of path length heuristics derived from lexical databases, word frequency measures taken from corpus statistics and partial syntactic matching (Šarić *et al*, 2012; Corley & Mihalcea, 2005; Mihalcea *et al*, 2006). These methods tend to use the relatively small dataset described by Li for comparative testing, these tests give a similarity score for pairs of sentences (Li *et al*, 2006). None of these methods appear optimised for ranking the semantic similarity of a large set of phrases against a target phrase.

This chapter gives an overview of word and concept matching measures that are tested in comparative experimentation in 5, these common similarity metrics are implemented within WordNet and the NLP Python package. This is not an exhaustive compilation of semantic comparison methods available for individual words and short texts, further examples can be found in surveys such as Gomaa *et al*, Zhang *et al* (Gomaa

& Fahmy, 2013; Zhang *et al,* 2013). The methods sampled are selected on the basis of a broad representation of the available corpus and vector-based techniques applicable to short phrase matching, allowing a comparative evaluation. Out of all these techniques, only the relatively novel Doc2vec method is tailored to matching short phrases. In Chapter 5 this method is tested alongside a greedy algorithm that returns the highest combined value of word pair combinations between two phrases. The word pair semantic similarity measures are those described in this chapter. TF-IDF and LSI/LSA methods are also tested for comparative assessment.

4.7 Short text matching techniques to date

5 Comparison of semantic measures for API text matching

Methods to determine semantic similarity between short texts require a different approach to those methods customarily used for document retrieval. No existing methods are adapted to find semantic similarity between software descriptions found in documentation or parameter names. Two promising techniques are tested, the doc2vec statistical matching method and a greedy combination algorithm. This greedy algorithm combines individual word pair values between phrases to derive a maximum score, several common word-pair semantic match methods are compared. The semantic values returned by these methods are tested against known matches between three commercial CAD API datasets. It is found that both methods perform markedly better than conventional document retrieval methods which are tested for comparative purposes.

5.1 Matching texts associated with Application Program Interfaces

The overall set of words used by both API under comparison tend to be limited and precise. Literary conventions such as the use of synonyms to avoid repetition are discarded in favour of maintaining a unique meaning for a word within the API domain. In the three CAD API used for documentation (Solidworks 2012 API, RhinoScript 5 2013 API, AutoDesk Inventor API 2012) there was no evidence of synonyms used for technical terms. What is apparent is a tendency towards using a different vocabulary than that of a competitor, leading to synonyms encountered between API rather than within API. Lu describes a semantic similarity measure suited to text labels within CAD ontologies CAD to augment reasoning (Lu *et al* 2016). Min describes experiments to compare the effectiveness of shape file names and descriptions using existing text semantic matching techniques with geometric shape matching (Min *et al*, 2004). However no research to date has published comparative analyses of the efficiency of different semantic matching methods within the narrow domain of API text matching, despite an indication that semantic matching is more effective with algorithms tailored to the particular domain. The words used for a CAD feature function label, or within the text description of a function and its associated parameters and behaviour are relatively short and not well adapted to common methods used for document similarity comparison or retrieval, such as Latent Semantic Analysis, or Term Frequency-Inverse Document Frequency models.

Aside from the Mikolov Doc2vec model, the methods for matching short texts, such as the Li algorithm and its derivatives, are based on measures of semantic similarity between individual words and syntactic comparison. These short sentence matching methods, as described in the Chapter 4 use a range of different syntactic and semantic metrics to compare short phrases, but an implementation of the Li algorithm was found to be prohibitively expensive in computational time when used to search for the best match among the thousands of function candidates within a commercial CAD API[1]. This can be attributed to repeated calls to the lexical database, in this case a NLTK, or Natural Language Tool Kit Python interface to the Wordnet semantic net (Bird *et al*, 2009; Miller, 1995). It can be seen that text used in CAD API function description, or function names, has several qualities that distinguish it from short texts found in other media, such as social media texts or news headlines. There is a strong consistency in the use of language, a relatively small specialist vocabulary and a pronounced lack of synonyms. This style of technical language is also particularly terse, semantic comparison methods that rely on contextual concepts, such as LSA, may have difficulty where there are a high number of overlapping concepts within the short sentences. This prediction is supported by experimental observation in Section 5.8, Observations on semantic method comparison testing results.

The other drawback with most existing short text comparison metrics is the use of manually compiled corpora. As there are no geometry or CAD feature specific manually compiled corpora as of writing, general purpose corpora must instead be used. These non-specific general purpose corpora are considered to fare worse than specialised corpora when used for semantic matching (Dusserre & Padró, 2017; Crossley *et al*, 2017; Senellart

1 A version of an implementation published online by Pal was ported to Python 3 and adapted to the test set (Pal, 2014).

& Blondel, 2008). This interpretation may be due to the proportion of irrelevant documents in larger corpora that reduce the effectiveness of LSA trained models. It would seem that word2vec methods do not suffer equivalent degradation with larger corpora, possibly because of negative sampling (Altszyler *et al*, 2017). As these questions are not readily answered from existing literature, and there is no information on the narrow context of API text matching, a range of both manually compiled and machine compiled corpora are used for training models.

5.2 Combined scoring for short text semantic comparison

The Li algorithm is not practical for searching relatively large repositories for semantically similar sentences. A more computationally efficient method is developed using the properties inherent to the technical language of API texts, these properties are described below.

The combined vocabulary of both sets of API text has a comparatively small set of words when compared against the total number of word instances within their respective texts. There is a deliberate re-use of technical terms used to define the structure and function of the CAD API architecture. As an example, the Solidworks API word set combined with the Rhinoscript API word set make up only 1502 unique words once common adjectives, prepositions and articles are stripped out. The technical term for words that are too generic to be of much use in semantic comparison is *stopwords* and a list[2] is adapted from those found in the NLTK corpus which are in turn taken from the Penn Treebank (Taylor, 2003).

Consider the operation of a short text semantic algorithm used to find sentences similar to a chosen sentence (for clarity, the target sentences or texts are defined as those discovered to be most semantically matched to the chosen source sentence). Document retrieval methods such as LSA or other vector-space semantic similarity models are adapted to efficiently determine a number of matched target documents from a large

2 Found in parseXMLgensim.py

repository. This is not the case for short text algorithms that compare the semantic values of individual words. This class of algorithm is evaluated using a list of sentence pairs, as each word is tested for semantic similarity against all words in the candidate target set, this amounts to a large search space where the candidate target texts may number in the thousands. The same word pairs are likely to be evaluated multiple times when assessing a single source text against multiple candidate target texts.

The solution adopted is to create large matrices where each word pair within the shared vocabulary is pre-calculated for semantic similarity according to a specific method, e.g. word2vec, Resnik, etc. This leads to large sparse arrays that can be held in local program space memory, or which can be held in contiguous disc storage. In trials, the sparse matrix generated by the combined vocabulary of SolidWorks and RhinoScript was small enough to reside in the 2 Gigabyte program space allotted to a 32-bit process running on a Windows 7 operating system (Lionel, 2019). Equivalent matrices generated from the combined vocabulary of SolidWorks API and AutoDesk Inventor API (4037 elements) generated matrices that were too large to reside in local memory, so a PyTables implementation of a sparse HDF5 array representation was used to allow the larger matrices to be stored to disc, while permitting reasonably fast read/write access, Singer provides an overview of the method (Singer, 2019; Alted & Fernández-Alonso, 2003; Folk *et al*, 2011). SolidWorks and Inventor CAD API may be considered to be representative of fully-featured medium sized CAD applications commonly used within small to medium sized enterprise.

If the semantic similarity between word pairs is pre-calculated then it is relatively efficient to extract the set of semantic matches for a word pair that have a numeric value over a minimum threshold. This threshold operation gives a direct trade-off between an exhaustive search and a set restricted to the highest scoring matches. The disadvantage of this greedy method is that discarding low probability single matches carries the inherent risk that sentences with multiple low individual semantic word scores have a combined semantic word score that is higher than the threshold for consideration.

5.2 Combined scoring for short text semantic comparison

This process is repeated for each word in the source text, creating multiple sets of target words with semantic similarity values greater than the set threshold. These target words can identify the target texts in which they appear. While high scoring words may appear multiple times in the same sentence, the majority of candidate texts encountered will have a single instance of a high scoring target word. In paired texts with multiple words scoring over the threshold value, there is a challenge of determining the consistent semantic score where there may be several different possible combinations of value.

5.3 Greedy Matching over multiple word match combinations

If any pretence of actual semantic comparison is discarded in favour of a method returning a unique and deterministic solution biased towards the highest possible score, this evaluation may be represented as a combinational optimisation problem. The combination of potential semantic matches that produce the highest value may be represented as the classical *assignment problem*. In Figure 7 the comparison of two short texts based on function labels is represented as a bipartite graph. Note that the preposition “From” is excluded as it is a member of the stopword set of low-information words. The task is to maximise the summed value of semantic relation scores, as represented by the graph edges and their values, in such a way that there is a one-to-one correspondence between graph nodes (so-called *feasible labelling*). An exhaustive

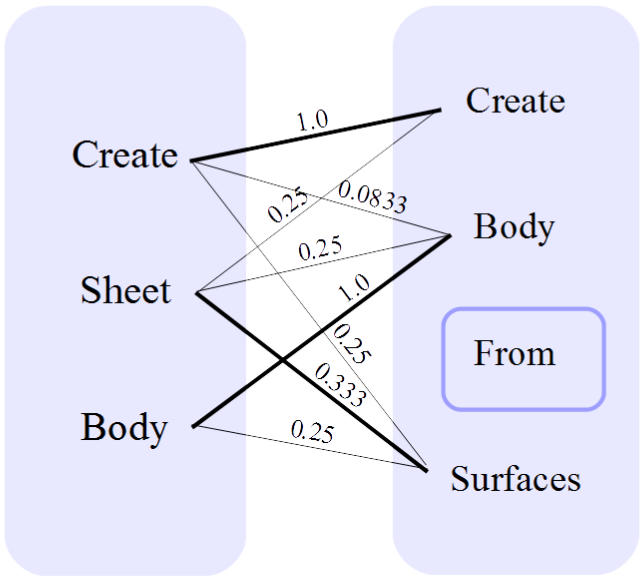


Figure 7: the highest word-pair match values are selected to form a word score, note stopwords such as "from" are excluded.

search for the optimum solution would complete in $O(n!)$, ruling out this method for sentences of any appreciable length.

Fortunately, a solution exists that is shown to solve the assignment problem in strongly polynomial time of $O(n^3)$. This is the *Hungarian method*, alternatively known as the *Munkres-Kahn* algorithm. An intuitive description of the algorithm is as follows (Kuhn, 1955; Pilgrim, 2019). Consider these graph edge values to be the values in a cost matrix where the rows represent the word nodes in one text and the columns the word nodes of the other text undergoing comparison.

The algorithm proceeds by subtracting the values of individual rows by the minimum row value to create one row element having zero value. This may reveal a minimum value solution, where each zero in the cost matrix occupies both a unique row and column. If there is no immediate solution, the same process is applied to each column and again the algorithm terminates if no zero is found for each unique row and column. Where there is still no solution, the cost matrix is adjusted again to create zeros cost values at new locations. This manipulation is applied to the region or regions of the cost matrix uncrossed by rows or columns containing zero values. The lowest value from this region is subtracted from all elements within the region, this lowest value is then added to the elements that appear on the rows and columns that contain zero values. There are a number of methods employed to determine the region for adjustment alongside the matrix elements that must be compensated for this adjustment which all generate the same outcome. This operation is repeated until distinct zeros are found for all rows and columns. The indices of these zero values may be used to extract the minimum values from the unaltered cost matrix. While this method finds a minimum value for the assignment problem, it has to be adapted to find the maximum value which can be simply achieved prior to the Hungarian calculation by subtracting the cost matrix from the largest element value.

Once the optimum combination of matches between singular or multiple words is determined, remaining unused words in each phrase are penalised as follows,

$$SemanticMatchScore = \frac{2 \times SemanticMatchScore}{length(phrase_A) + length(phrase_B)}$$

The probability of several semantically related words appearing within two texts under comparison is not related to the probability between individual words appearing, despite all machine compiled corpora relying on the distributional hypothesis. This is not the case where several semantically words share the same concept, but a greedy combinational algorithm will winnow out these relationships. This relationship is acknowledged in the WordNet *satellite* parts of speech, where two words that commonly appear together, such as "New York" are given a unique category. This categorisation allows the contribution of individual semantic matches to be independently registered. Research to date does not reveal similar evaluation, e.g. SyMSS (Oliva *et al*, 2011; Wang *et al* 2015; Li *et al*, 2011). This evaluation is considered tangential to the topic of comparing API texts, a simple arbitrary weighting is introduced to texts that have multiple semantic matches to introduce a bias to the overall Semantic Match Score value.

The advantage of this method is that it reserves computationally expensive combinational matching tasks to pairs of sentences that have known combinations. This tends to be a relatively small subset of the overall number of sentences. This is the limitation of the greedy algorithm, which is liable to miss relatively complex optima within a solution space. For comparison purposes, the Greedy algorithm using Leacock-Chodorow Similarity matching performed an order of magnitude faster over the same API dataset (Leacock-Chodorow similarity is the closest path length and depth matching metric to the Li algorithm) than the only other algorithm tested for short phrase matching, that of Li (Li *et al*, 2006). Figure 8 shows the proportion of phrases taken from a comparison of SolidWorks and Inventor CAD API that are matched by a single word with a score of 0.75 semantic similarity, against the number of phrases that have multiple words with this same match value. This semantic similarity figure is normalised to a range between 1.0 and 0.0 and is derived from a test of function parameter names, associated function text using a Wu-Palmer semantic similarity measure. This text parsing is described in Section 5.7.

5.3 Greedy Matching over multiple word match combinations

It is relatively simple to integrate the Natural Language Toolkit (http://www.nltk.org) which contains a version of the WordNet corpus accessible in the Python computer language (http://www.python.org) and a number of similarity measures based on ontology of the WordNet corpus (Bird *et al* 2009; Miller, 1995).

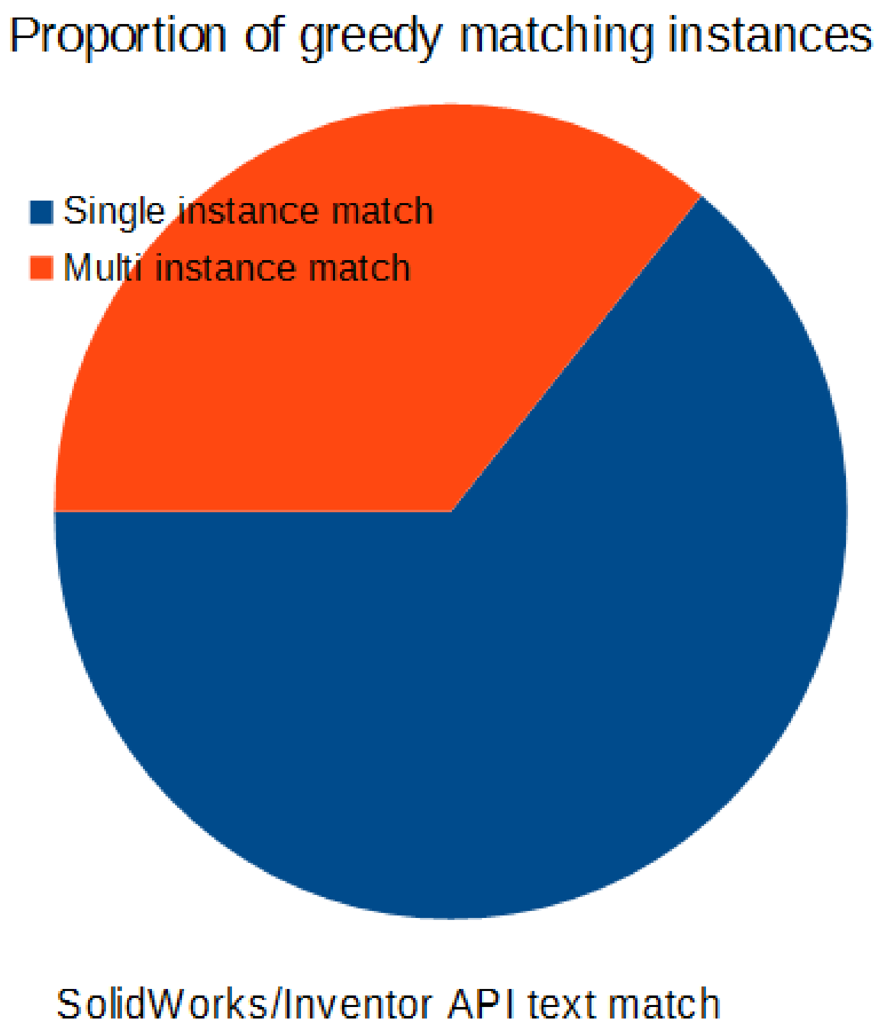


Figure 8: comparison of number of single text matches vs multiple text matches at a normalised similarity threshold of 0.75

5.3 Greedy Matching over multiple word match combinations

5.4 Trained corpus experimentation

Machine compiled corpora appear to be most successful when the document collection that informs the corpus is relatively specialised, as outlined in Chapter 4.1, Semantic mapping within CAE systems. There is also a requirement to have a large collection of training documents to reduce the negative influence of outliers. The semantic matching experiments have been carried out with Word2vec corpora compiled from document collections that range in size and specificity.

- A word matching corpus has been generated from the document collection within the English Wikipedia. This collection of 5 million documents represents a large and unspecialised corpus.
- A second corpus is generated from the entirety of documentation within both CAD API, totalling 6343 short documents.
- A third corpus is generated exclusively from the words used in function names. Function and parameter labels customarily employ a "camelcase" convention where separate descriptive words have whitespace removed to form compound labels such as "CreateBodyFromSurfaces".

A set of 38 pairs of matching functions were selected from the API of two commercial CAD programs, Dassault Solidworks(R) 2010 and McNeel RhinoScript 5. Another 67 pairs were identified between the same Solidworks 2010 API and Autodesk Inventor 2012 API. Each of these paired CAD functions may create identical geometries with appropriate parameters. Some functions had more than one equivalent function in the counterpart API, a source function would have two target functions that replicated the same functionality, or in some cases two source functions would have a single target. The source and target assignments of these pairs may then be reversed and the process repeated. In some cases there are more than one matching target text, where two functions may perform the same operation as the source function. For example, a function that creates a frustum in one API may be matched by a function that creates a cylinder given two radii and another that defines a conic section.

To try to evaluate the effectiveness of any method, rather than simply calculating the value of the semantic similarity, text matches that score over the threshold are ranked in order of semantic match values assigned. A perfect match would rank the known target function first in the set of returned target texts ranked by semantic match value.

Semantic matching effectiveness can be tested with a similar precision and recall metric as that used for document retrieval. The task of text semantic matching in this context is to arrange a ranked shortlist of likely matches of API functions.

5.5 Mean Average Precision, Mean Reciprocal Rank and Mean Rank metrics

The recall metric is the proportion of items correctly identified out of the total number of correct matches. Identifying equivalent, or semi-equivalent functions between the texts from two sets of API documentation places a tighter constraint on a recall metric. There are on average far fewer relevant matches than would be common in a document retrieval context, and the total number of functions to be matched would be smaller than most realistic document collections. In the case of manually identified matching functions, the recall is considered to be unity; all the matching functions are identified. With this constraint the precision then corresponds to the least probable ordered ranking of the set containing all the correct matches. In the case of a single equivalent function match between two API, the ranking of the correct match in the returned semantic match probabilities corresponds to the inverse of the precision measure, the rank measure, sometimes known as the “threshold” measure. This threshold measure is the proportion of CAD API functions out of the entire set that must be searched before a geometry match is successful.

Three measures are used, Mean Average Precision, Mean Reciprocal Rank and Mean Rank. These measures are adapted to ranked search results, the first two metrics commonly used for model comparative purposes. These measures are more readily presented in the context of more commonly used measures. *Precision* is defined as the proportion of documents retrieved by a query that are relevant to the given query.

$$Precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|}$$

Recall or *Sensitivity* is the complementary measure that normalises the fraction of relevant retrieved documents against the total number of relevant documents. This allows an assessment of the efficiency of the retrieval algorithm, or a normalised measure of true positives to false positives. In this case, all potential texts are tested and the probability ranking of the correct match is used, therefore recall is not relevant as all documents are retrieved.

$$Recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|}$$

In the case of retrieved documents that are ranked according to their relevance, it is helpful to see the variation of relevance across retrieved results. A *precision-recall curve* plots the precision against the recall $p(r)$, for each document in the ranked sequence, giving a visual estimate of the distribution of document query relevance across the ranked set of retrieved documents. This curve can be integrated into a single value, *Average Precision*, a coarser metric that gives a single figure useful for comparative assessments between multiple retrieval methods.

$$Average\ Precision = \int_0^1 p(r)dr$$

In practice, this figure is a summation of individual ranked precision values over the entire ranking interval.

5.5 Mean Average Precision, Mean Reciprocal Rank and Mean Rank metrics

$$Average\ Precision = \sum_{k=1}^{n} P(k)\Delta r(k)$$

Where k is the index of the set of returned ranked documents, N such that $P(k)$ is the precision of the document indexed by k and $\Delta r(k)$ is the difference in recall over the interval k to $k-1$. If the search algorithm finds a match on the highest ranked result, namely $k=1$, the local precision would be one. In the case where the highest rank was not a match this precision would equal zero. As the ranked probability decreases and k goes to N, these cumulative values of precision form the Average Precision.

$$Average\ Precision = \frac{1}{N}\sum_{k=1}^{N} P(k)rel(k)$$

Where $rel(k)$ is a binary measure of the relevance of the document at rank k, such as the document is judged either relevant, or irrelevant.

This measure gives a more general comparative metric if it can represent the results of several different queries. The *Mean Average Precision* takes the summation of multiple Average Precision results and normalises them to the number of queries, Q.

$$Mean\ Average\ Precision = \frac{1}{|Q|}\sum_{j=1}^{Q}\frac{1}{|N_j|}\sum_{k=1}^{N_j} P(k_i)rel(k_i)$$

Reciprocal Rank is the reciprocal of the kth rank of the first correctly identified match in the set of returned matches N, again if the highest probability ranking is correct, the value is $\frac{1}{1}$, if the second value is judged correct but not the first, this would give a value of $\frac{1}{2}$. The *Mean Reciprocal Rank* is the sum of these Reciprocal Rank values for a set of queries, Q normalised by the total number of correct results.

$$Mean\ Reciprocal\ Rank = \frac{1}{|Q|}\sum_{i=1}^{Q}\frac{1}{k_i}$$

This reciprocal ranking metric gives a reasonable numerical estimate of the benefit of using a semantic search to assist function matching. Semantic matching assistance increases the probability of finding a function match within a smaller search space, if the function text is compared using the matching algorithms described then a reciprocal rank measure indicates the number of functions that would have to be tested before a match is found.

5.6 Mean Rank

Consider that, unlike document retrieval testing, there are only one or two predetermined function text matches in a query over the entire API text dataset. Large document test sets are not matched against queries, instead a number of pooled search results are examined by several human assessors. This practise means that documents that may have escaped scrutiny are assumed as unmatched, potentially leading to a false assignment as a document with no relevance, a false negative. As the CAD API function text sets used for matching number in the thousands rather than the hundreds of thousands, the appearance of false negatives in retrieved matches is less likely. This in turn simplifies comparison between algorithms. Buckley and Voorhees describe a binary preference metric or *bpref* that gives figures suited to comparison of semantic match algorithms that return a ranked set of matches (Buckley & Voorhees, 2004), the metric described below is of similar format but is not subject to the false negatives above, for clarity it is named Mean Rank in this context.

Finding a single document in a set means that the retrieval algorithm will either correctly rank the predetermined match as the highest rank, or assign the correct match at a reduced ranking dependent on the accuracy of the algorithm. If the algorithm ranks the entire dataset and the correct match scores the lowest ranking, the outcome is perfectly inaccurate. If the numeric rank is divided by the length of the text dataset, this gives a metric metric that varies between zero for a perfectly accurate score, to one for a perfectly inaccurate score. If this figure is subtracted from one, it gives a measure of relative algorithm accuracy. This measure may be conducted over several queries and the summed outcome divided by the queries. Both Mean Reciprocal Rank and Mean Average Precision behave in a similar fashion when used with searches with a single judged match, this may be seen in the proportional similarity of graphic comparisons. Mean Rank is Mean Average Precision normalised to the size of the data set.

$$Mean\ Rank = \frac{1}{|Q|}\sum_{i=1}^{Q}\left(1 - \frac{k_i}{|R|}\right)$$

where Q is the set of queries, k is the known match in R where R is the set of texts undergoing comparison to k.

5.7 Parsing of API text data

The help files associated with CAD function API are invariably represented using Hypertext Markup Language files. These files may be converted into a free standing Compiled Help File which can be disassembled into its constituent HTML using Microsoft tools, or if the help files are not accessible as a single file, but are available from a website, they may be "scraped" using a webcrawler. These HTML descriptive files are parsed to extract text content as a series of strings using the BeautifulSoup library (Richardson, 2019) (`parseXMLgensim.py, API_SWKS_CHM_07.py`, etc). These text strings are parsed to remove stopwords as described in Section 5.2.

The words contained in these text strings is grouped into a single set, comprising the union of all words encountered in the API help data, these sets are of the words contained in the camelcase function names, along with the text in the function descriptions. Each combination of word pairs in this set is assigned an individual similarity value according to the Leacock-Chodrow, Jiang-Conrath, Lin, Resnik and Wu-Palmer measures that use the NTLK corpus. In the absence of other word context data, the Li method that selects the highest value synset for a word pair is used (`createWordNetSimMatrix`).

A slightly different approach is taken to construct a similarity matrix using the word2vec method. The short sentences created by the function filenames or descriptions are used to train an implementation of the Milolov's Word2vec model using the Gensim library (Rehurek, 2010) (`createWord2VecSimMatrix`). This model may then be queried for similarity rankings of individual word pairs as before to construct a similarity matrix (`createGensimWord2VecModel`). The Word2vec model is not restricted to CAD API texts for training purposes, the content of the Wikipedia web-based encyclopedia project is also used to create a Word2vec model to compare results against a more general corpus. The Li algorithm was abandoned for comparative testing of the API text dataset because of impractically long operation times.

These word pair similarity rankings are used with the greedy combinational method to determine a similarity measure for pairs of short texts. Each of these word pair similarity methods may then be trialled. Implementations of Doc2vec and LSI semantic similarity is also used for comparison. Both the LSI model and the term frequency – inverse document frequency model (TF-IDF) are trained from function names that have been converted from camelcase concatenations to short phrases, and API help text presented in the form of lists of stemmed strings. These semantic measures are included for comparison with the greedy methods intended for short strings, the Rehurek Gensim library allows rapid Bag of Words weighting of the API text corpus. In brief, a similar word pair co-occurence matrix is formed, then each word is weighted by a factor related to its frequency within the corpus, in this case the log of the inverse document frequency value. The Gensim library is highly optimised and calculations are performed an order of magnitude faster than the Python based greedy equivalent. This same model is then converted via SVD to a reduced abstract vector space, termed as Latent Semantic Indexing (or otherwise known as Latent Semantic Analysis), this model is more rapid but has a matching performance similar to the TF-IDF model.

The Doc2vec model is similar to the Word2vec model (see also Word Embedding and the word2vec similarity measure), but the model vectors are based on the *Paragraph Vector* model of Le & Mikolov, where each short text is a vector (Le & Mikolov, 2014). Because short texts appear most suited to the terse API help documentation, three different model variants are used, the first with a word sampling window as used in the Paragraph Vector model, the third with a larger word sampling window size. The second model does not retain the order of the words in the texts during sampling, but uses a distributed "Bag-of-Words" sample where word order is not preserved. Note that because there are at most three target matches for every source API text document, the customary Precision/Recall graphs are unsuited to comparison. In this study there are two sets of API text matches between different commercial CAD programs, for each source document the entire set or potential target documents is ranked according to the calculated match probability. The position of the known match on this ranked list gives a basis for comparing models, as described above in Mean Rank. These individual match figures for each model could be averaged for a single numerical comparison of each method, instead a violin plot is used which gives more visual information about the quantitative grouping of individual average precision values of each semantic similarity model and consequently a better indication of outliers. The column width of the violin plot is a continuous approximation of the probability density function generated from the individual values. This probability density function is smoothed using kernel density estimation. A violin plot displays the breadth of each plot column with a width proportional to the distribution of values. The point values are also superimposed on the plots as bars.

5.7 Parsing of API text data

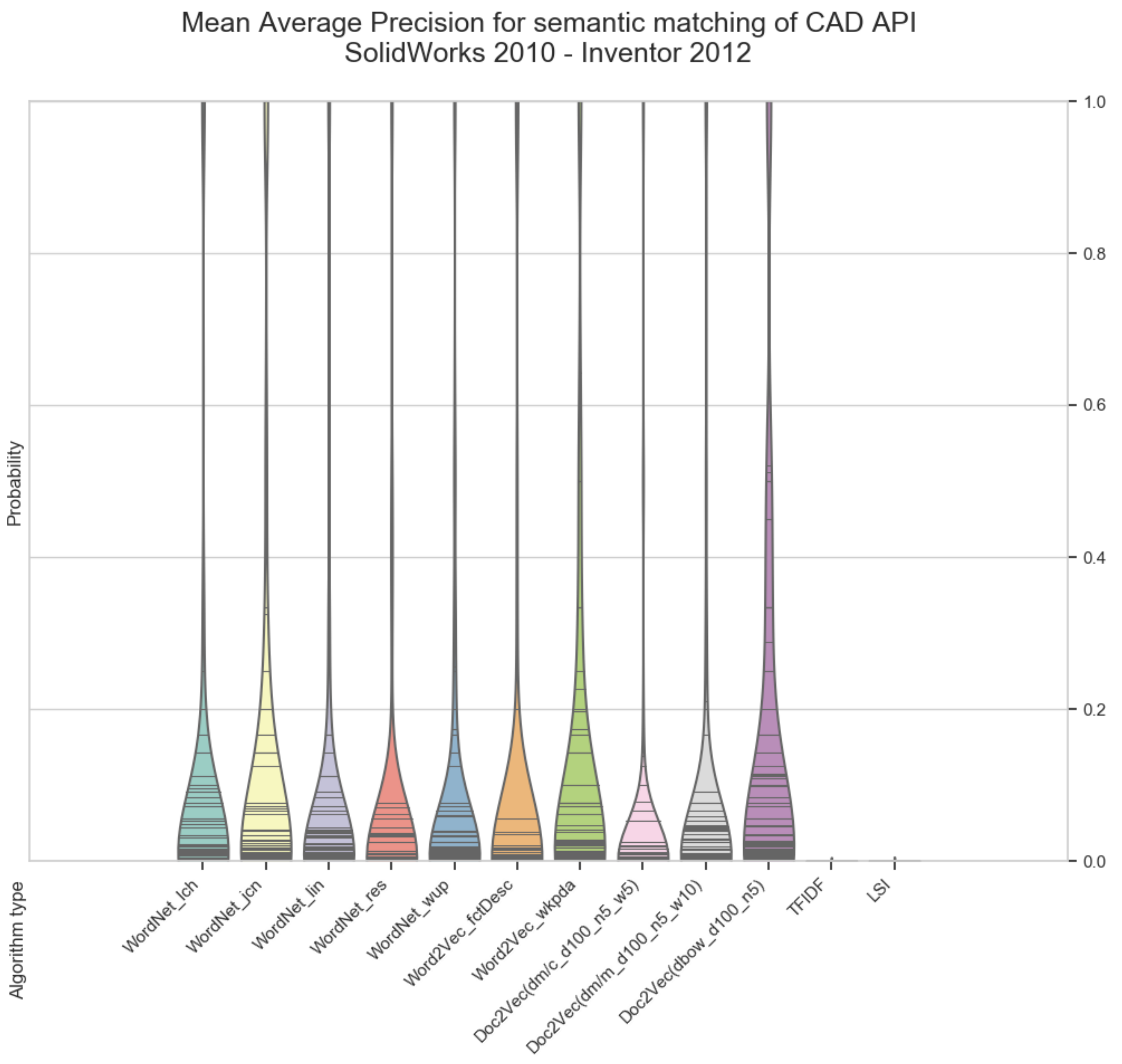


Figure 9: Mean Average Precision values for matched API function texts between Solidworks 2010 and Inventor 2012

5.7 Parsing of API text data

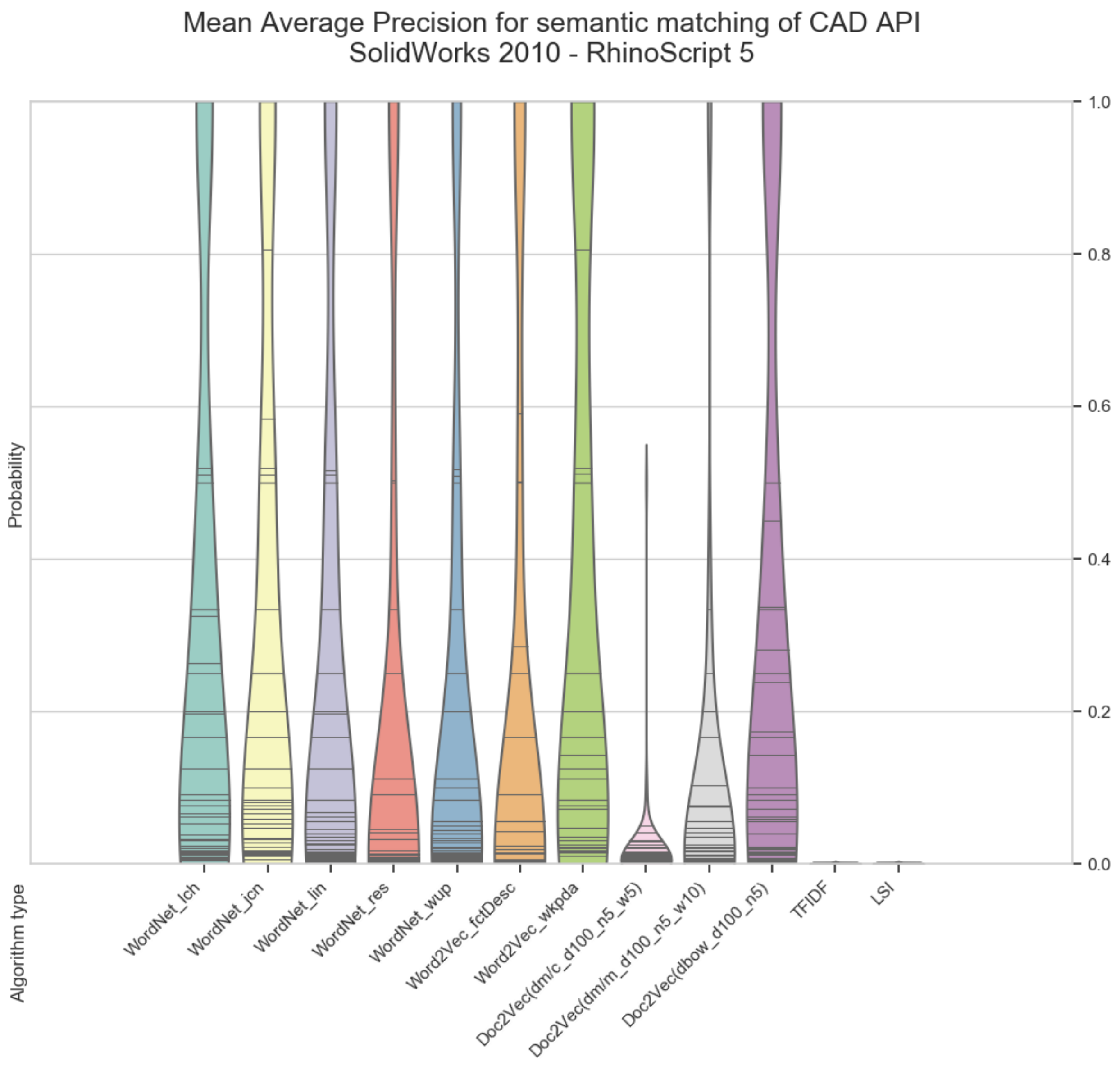


Figure 10: Mean Average Precision values for matched API function texts between Solidworks 2010 and RhinoScript 5

5.7 Parsing of API text data

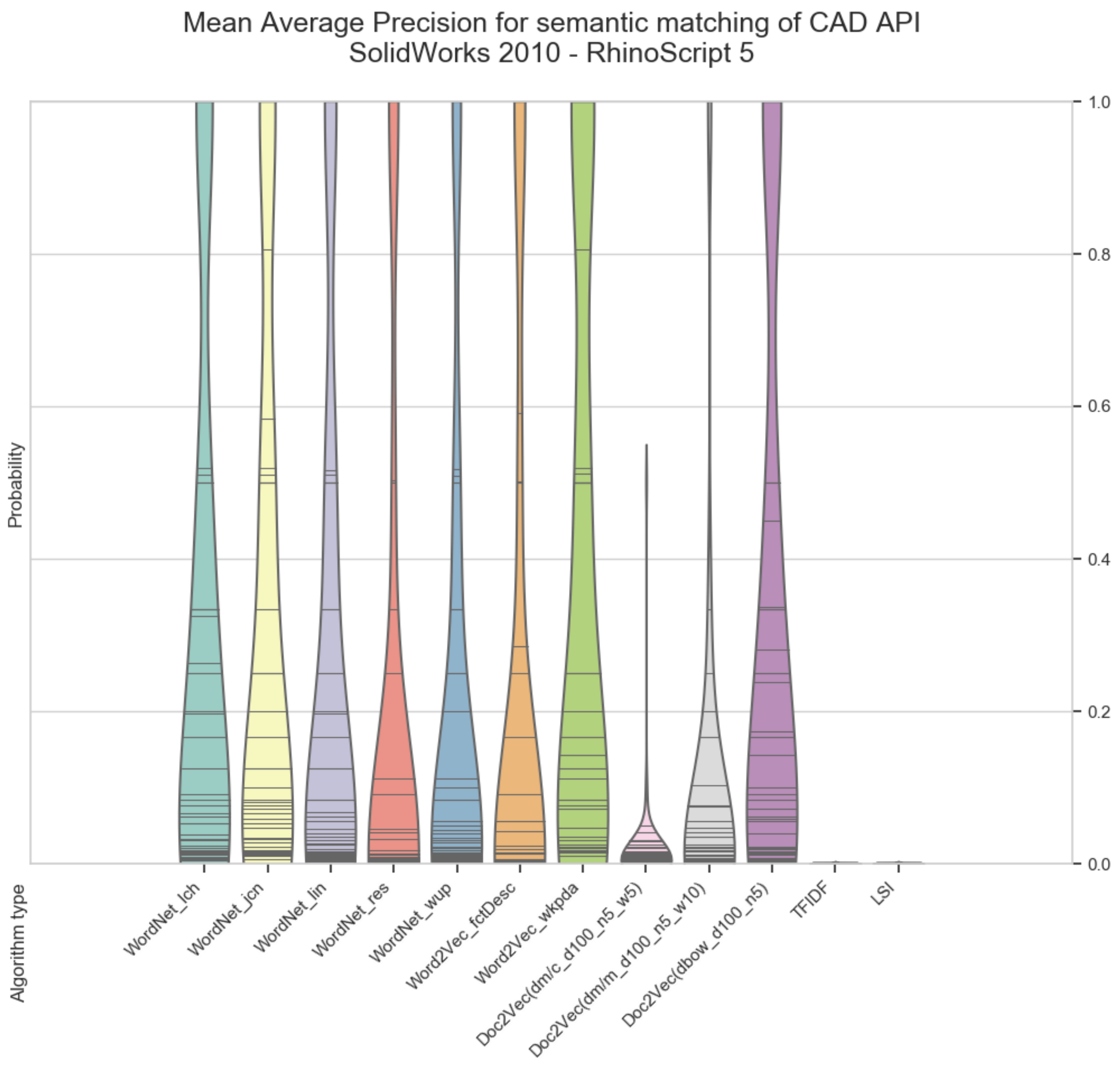


Figure 11: Mean Average Precision values for matched API function texts between Solidworks 2010 and RhinoScript5

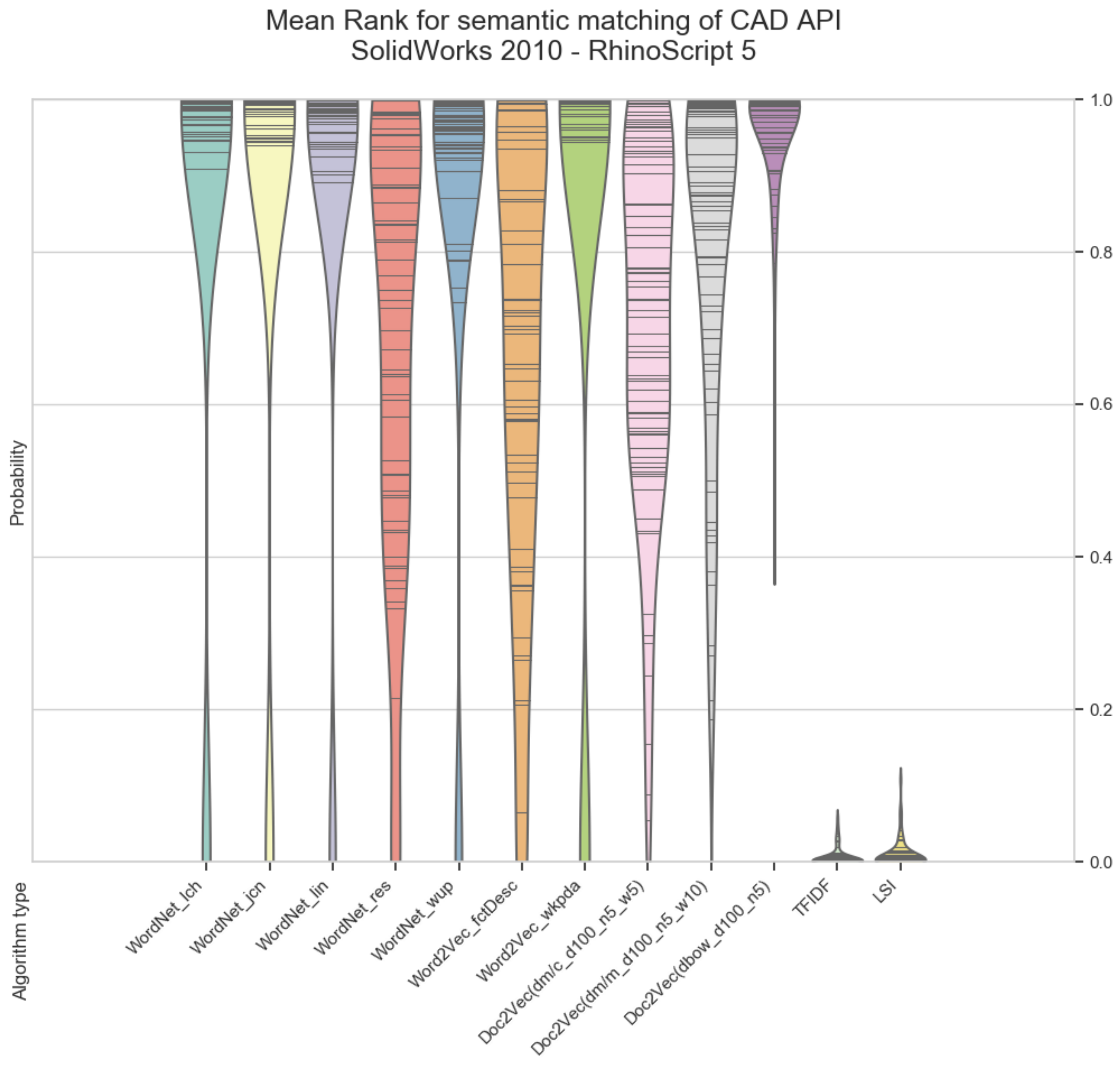


Figure 12: Mean Rank values for matched API function texts between Solidworks 2010 and RhinoScript 5

5.8 Observations on semantic method comparison testing results

The Mean Average Precision graphs for both sets of API texts show that the traditional text matching methods fare badly (Error: Reference source not found, Figure 11). This is related to the density of concepts within the short terse API descriptions. The Doc2vec methods show a surprising variation relative to the word sampling methods with the most basic Bag-of-Words method performing best. This indicates that elevating the importance of word order within a sampled phrase lowers accuracy. Virtually all short phrase matching techniques use word ordering (see Chapter 4.7). This anomaly may stem from the ordering of text within API HTML documentation. The HTML parser extracts readable text, adding paragraph titles, descriptive headings and sentences in the linear order in which they appear on the page, this does not guarantee grammatical sense in the short phrases generated.

As the greedy algorithm for the other models is identical, the only difference is in the methods used to select individual word pairs. While each of the WordNet based measures have similar performance, the two word2vec models show a different distribution and higher performance. The model combining a Word2vec similarity model derived from the Wikipedia corpus shows the best overall performance, but not by a notable margin despite having a far larger corpus size.

The violin graph showing the Mean Rank may be readily interpreted as the proportion of a ranked list of API texts that would have to be tested before arriving at the correct match. A low score indicates that the entire ranked API would have to be searched to find the correct match. In practice, a zero score for values determined by the greedy method indicate that no score was found. It can be seen that scores around the 0.5 mark are of similar quality as a random guess while a score of 1 means that the highest ranked semantic match coincidences with the correct target text. The rationale for this non-standard metric is that it gives a visual indication of how a semantic text match might partition a search space to a tractable fraction of an API.

Most search models perform better than average over the set of queries, this may in part be due to selected source-target pairs being indisputable matches rather than partial matches. The TF-IDF and LSI models perform worse than a random guess while the Resnik, word-sorted Doc2vec and API corpus word2vec models have a broad probability distribution. It may be seen that models that use a training corpus based on the respective API do not suffer from the instances of unrecognised words (Word2vec_fctDesc and all Doc2vec models; not visible with LSI and TF-IDF), unlike those based on a general purpose corpus such as the Semcor corpus of the WordNet path measures (Word2vec_lch, Word2vec_jcn, Word2vec_lin, Word2vec_res, Word2vec_wup), or the Wikipedia corpus (Word2vec_wkpda).

A second set of plots are made of the averaged queries to allow comparison between trials on different API. It can be immediately seen that the RhinoScript and SolidWorks API have a higher overall match success rate than do the Inventor and Solidworks. This may be due to the RhinoScript and SolidWorks sharing a more similar vocabulary to describe the same features. The Wikipedia corpus, Jiang-Conrath word2vec based greedy methods and Bag-of-Words Doc2vec method are seen to perform higher than average over the test sets, while the LSI and TF-IDF methods perform poorly. The Mean Rank graph indicates the proportion of each API that remains below the rank of the known source target match. These values indicate that each method bar the LSI and TF-IDF perform better than a random selection.

It may be concluded that the semantic matching techniques trialled here are not suited to determining feature function matches based on their API text descriptions alone. It is also apparent, particularly from the Mean Rank graphs, that several of the better techniques can reduce the search space to a small fraction of the entire API set for the source-target matches tested. A relatively rapid and general purpose method, such as the Doc2vec, or a greedy method combined with a word-pair semantic similarity method is shown to reduce the number of tests required to find a matching function. This technique may be used to increase the efficiency of specific function matching methods that are more computationally intensive, such as that described in Chapter 9.

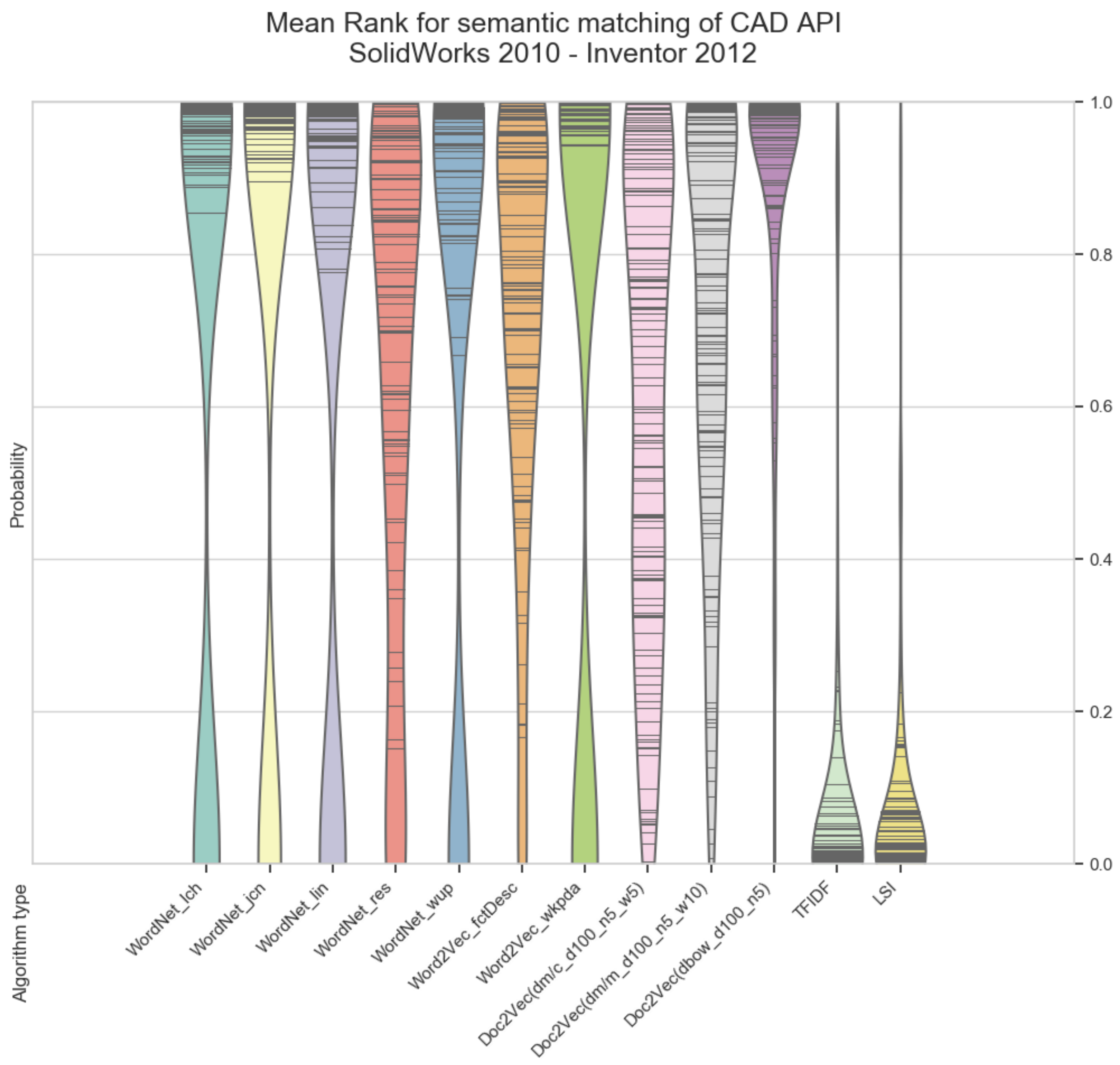


Figure 13: Mean Rank values for matched API function texts between Solidworks 2010 and Inventor 2012

5.8 Observations on semantic method comparison testing results

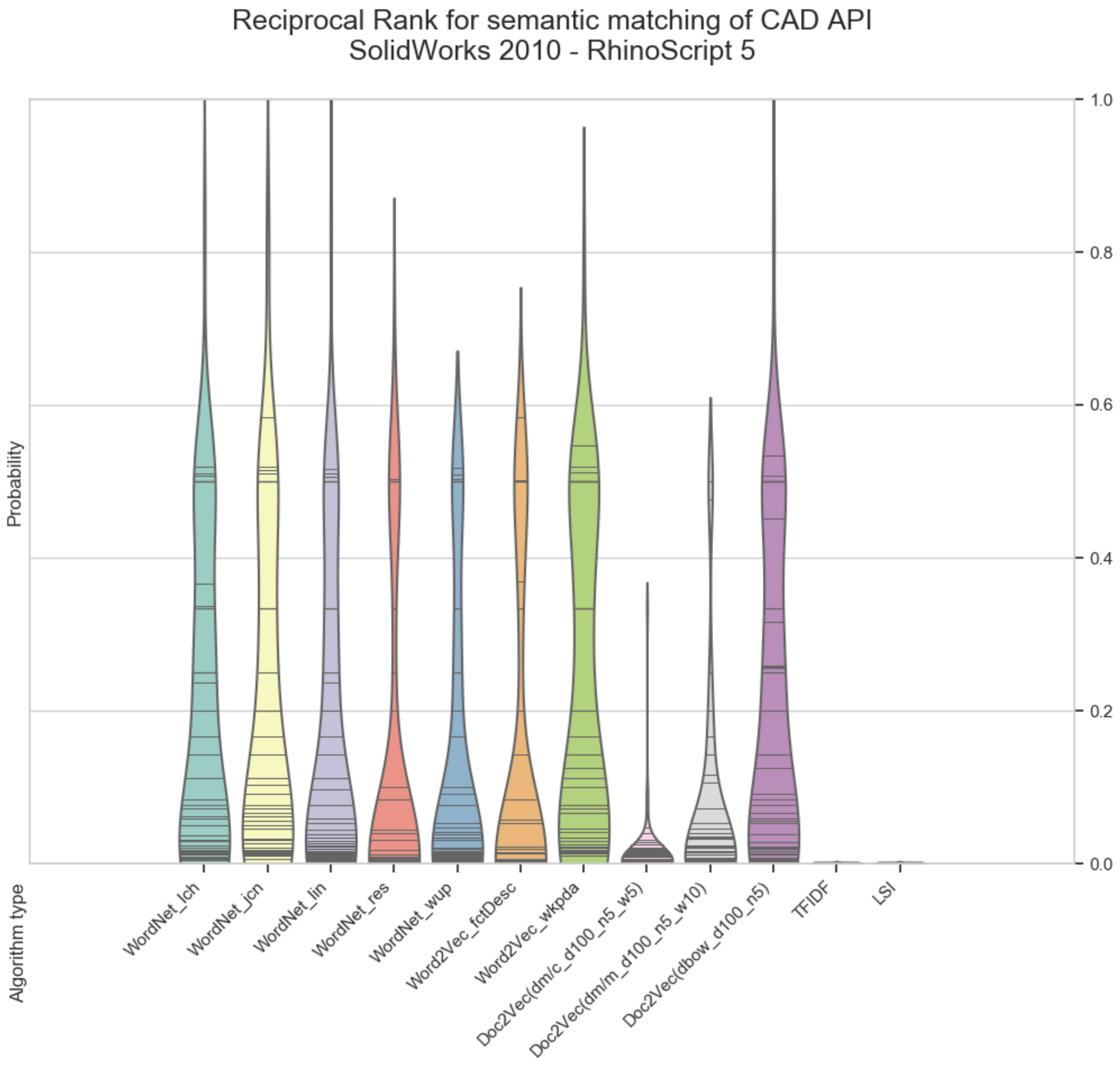


Figure 14: Reciprocal Rank values for matched API function texts between Solidworks 2010 and RhinoScript 5

5.8 Observations on semantic method comparison testing results

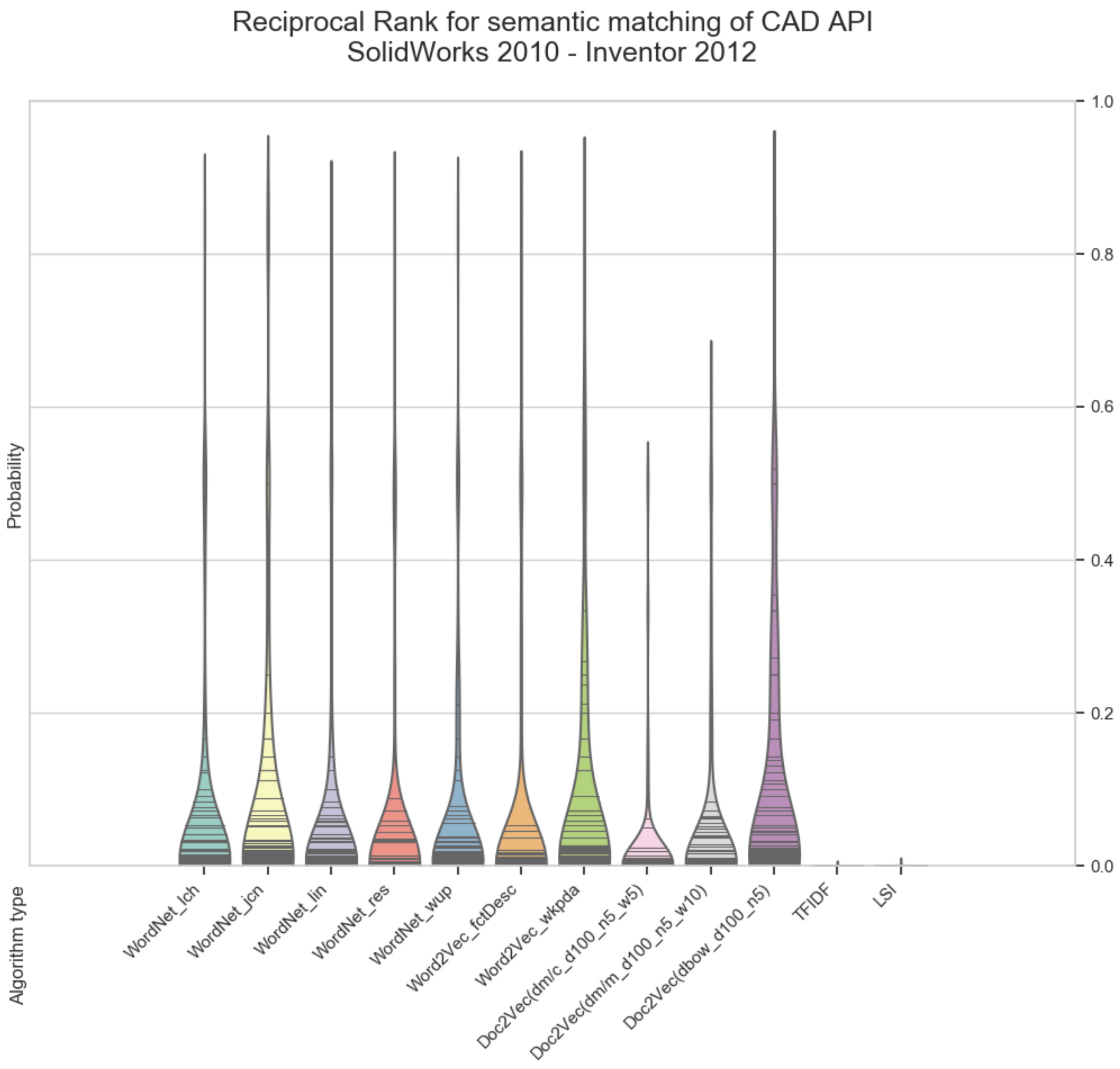


Figure 15: Mean Rank values for matched API function texts between Solidworks 2010 and Inventor 2012

5.8 Observations on semantic method comparison testing results

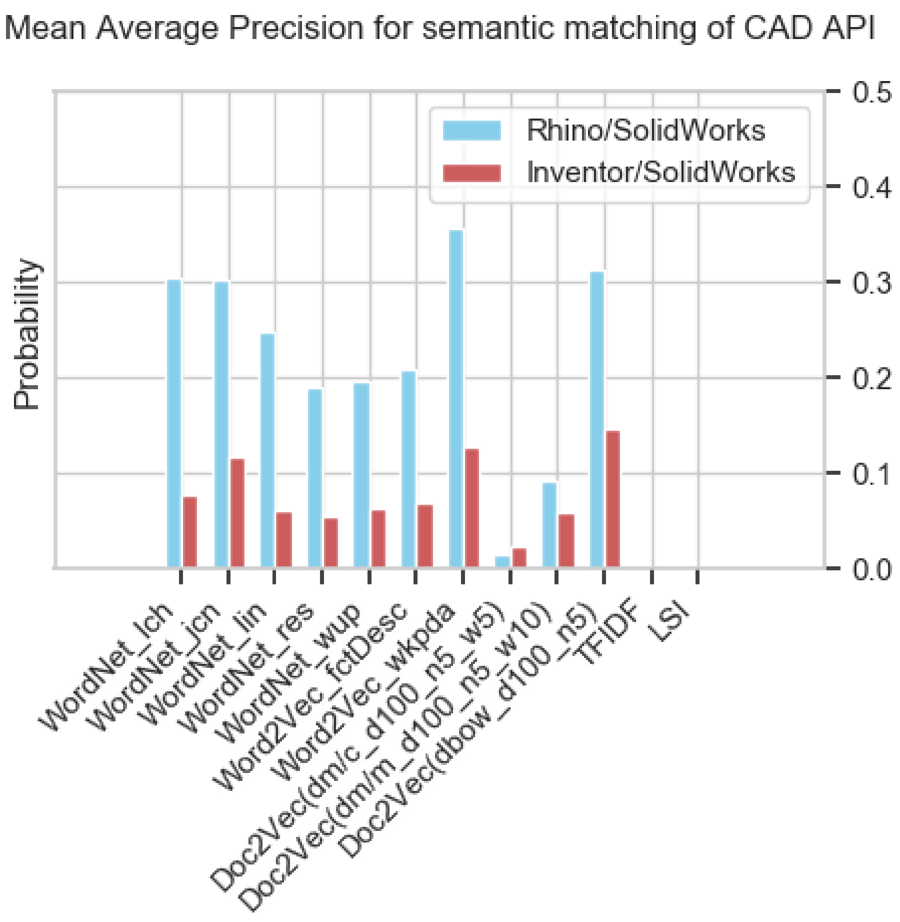


Figure 16: Mean Average Precision over both sets of CAD API samples [0, 0.5]

5.8 Observations on semantic method comparison testing results

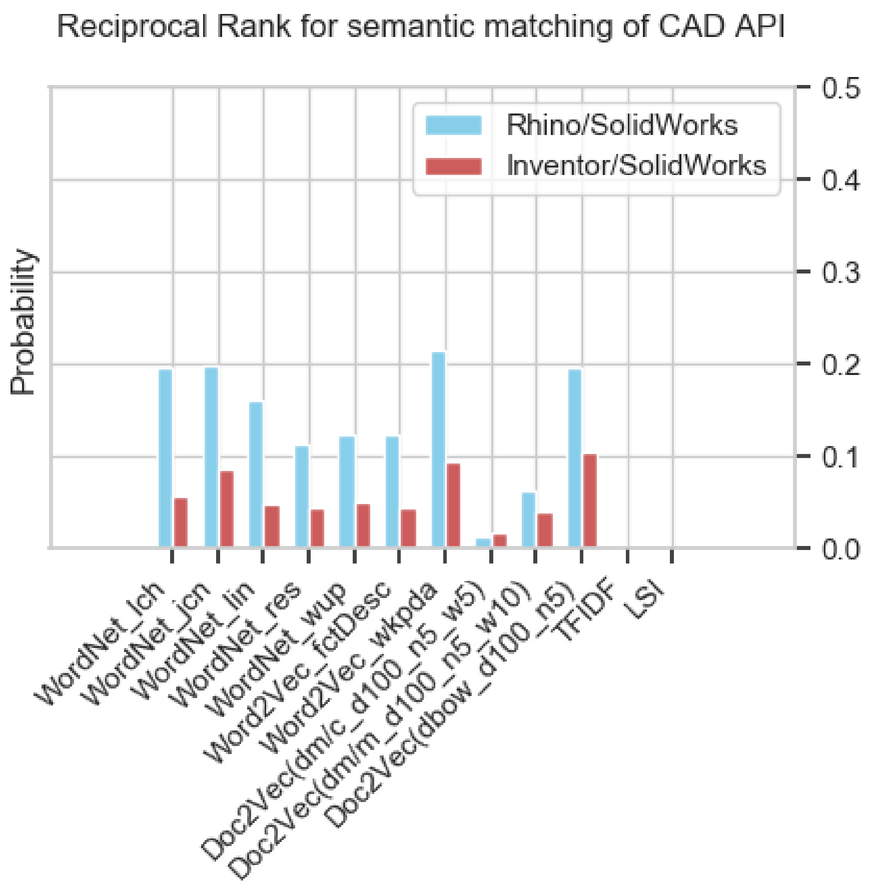


Figure 17: Reciprocal Ranking over both sets of CAD API samples [0, 0.5]

5.8 Observations on semantic method comparison testing results

5.8 Observations on semantic method comparison testing results

6 Boundary Surface Geometry Comparison

The following chapter describes a method to evaluate similarity between the boundary surface geometry of models within different CAD programs. This method is distinguished from similar surface registration algorithms by an accuracy that makes it suited to evaluating CAD feature function equivalence. The method determines a set of points on the boundary surface of a model that perform two functions. First, as a rotation-invariant model signature for model matching. Second, as registration points for deriving a rotation matrix and translation between geometrically similar models at different orientations. These feature points are identified via an iterative search to find a local maximum or minimum surface region relative to the model centroid. Additional operations determine the centre-point of ridges or grooves around a model axis. Points are sorted in a helical ordering that allows paired point matching between models.

6.1 Overview of a geometric matching method

Syntactic and semantic matching techniques applied to short API texts have a limited precision. They do not return an accurate ranked equivalence of the phrases undergoing comparison. The experiments in Chapters 5.4, Trained corpus experimentation support this assertion. While text matching techniques are useful in reducing a search for equivalent CAD feature functions to a smaller pool of higher probability matches, they are impractical as a stand-alone automated method for determining API function equivalence.

A more accurate API function matching technique can use information from a “black-box” comparison of function inputs and outputs, namely the API function is characterised as a transfer function without regard for its internal workings. In the case of CAD API functions, a combination of input parameters result in an output that modifies or creates a geometrical object within the model space of the CAD program.

If two CAD API functions have identical input and geometrical output, it can be inferred that the behaviour of the functions is equivalent. If two functions have dissimilar input yet identical geometrical output, it is apparent that the functions are related. Unlike generic text matching of function description, a geometric comparison can determine function relationship with a high precision, as measurement of space is better defined than measurement of concept in semantic or syntactic comparisons.

Many of the functions within a CAD API do not have a geometrical output, but comprise the routine housekeeping functions such as saving files or altering the user interface. As a translation between CAD geometric models generally only concerns events within the CAD model space, these ancillary functions can be disregarded. They may be readily filtered out of the set of candidate matching functions by identification of their input and output types. The proportion of CAD API functions relevant to geometric operations is shown in Figure 19, where the relevant fraction is highlighted. This chart shows a chart of all API calls within the McNeel Rhinoscript API for the Rhino CAD program, version 5 (developer.rhino3d.com., 2017). The functions have been manually sorted into six categories, four of which describe API functions that either directly or indirectly affect model geometry, a category of functions where the function relationship to model geometry is not readily specified and the largest category of methods that are not associated with geometric operations.

6.1 Overview of a geometric matching method

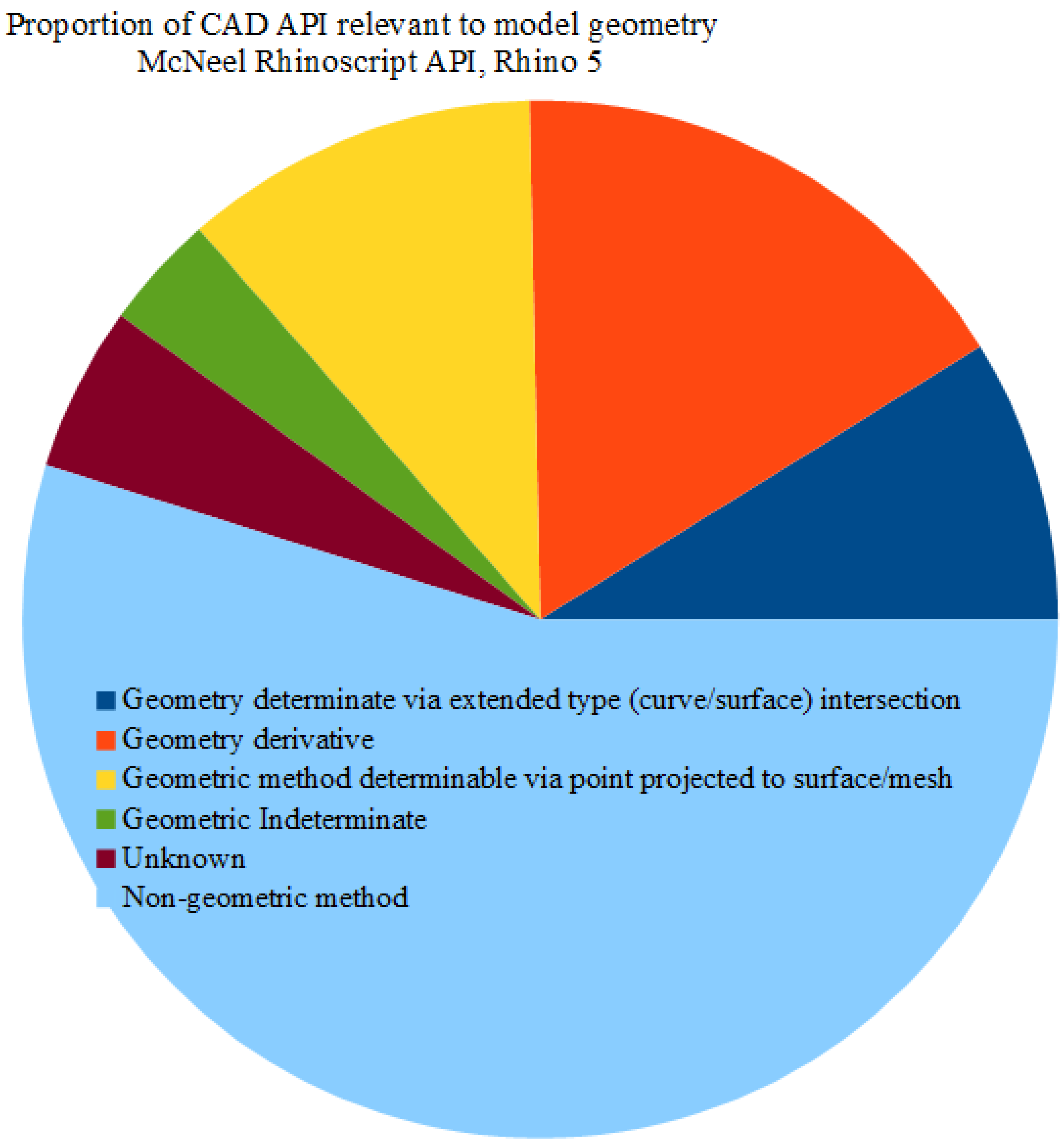


Figure 19: proportion of CAD API functions directly applicable to model geometry in RhinoScript 5.

6.2 CAD model geometry comparison

If the output of CAD API functions are to be measured for equivalence, this necessitates a geometric comparison of CAD geometry models within their respective CAD programs.

The instinctual solution of importing one CAD model into the model space of the second CAD program for direct surface comparison would be subject to two limiting constraints.

- Firstly, this approach relies on this import facility existing within the second CAD program.
- Secondly, should this import facility exist, it would be subject to precision limitations inherent to potentially inequivalent definitions and numerical tolerances. See Chapter 3.6.1, Numerical accuracy for detail on incompatible representation issues between different commercial CAD software.

An alternative solution might propose that the respective geometric outcomes of the two function outputs undergoing comparison are exported to a standardised neutral intermediary format such as ISO10 303-21, then allowing a direct comparison of model boundary surfaces. This proposition is again subject to the constraints of precision and interpretation imposed by the CAD export implementation. There is a second issue where it is not evident how the format of a neutral geometry representation might undergo a numerical geometric comparison. The summary of Chapter 3.2, CAD standardisation initiatives provides more detail on the role of neutral formats.

A universal query routine must work with any CAD software undergoing comparative testing and must also be relatively immune to geometry inconsistencies between the internal geometric model representation of different CAD software. The Hoffmann proposition of Chapter 3.12, Representative proxy model and query protocol describes an explicit routine to determine relative precision of CAD programs respective to a hypothetical reference model (Hoffmann *et al*, 2014). The following method describes a query-based numerical comparison of CAD models.

6.2 CAD model geometry comparison

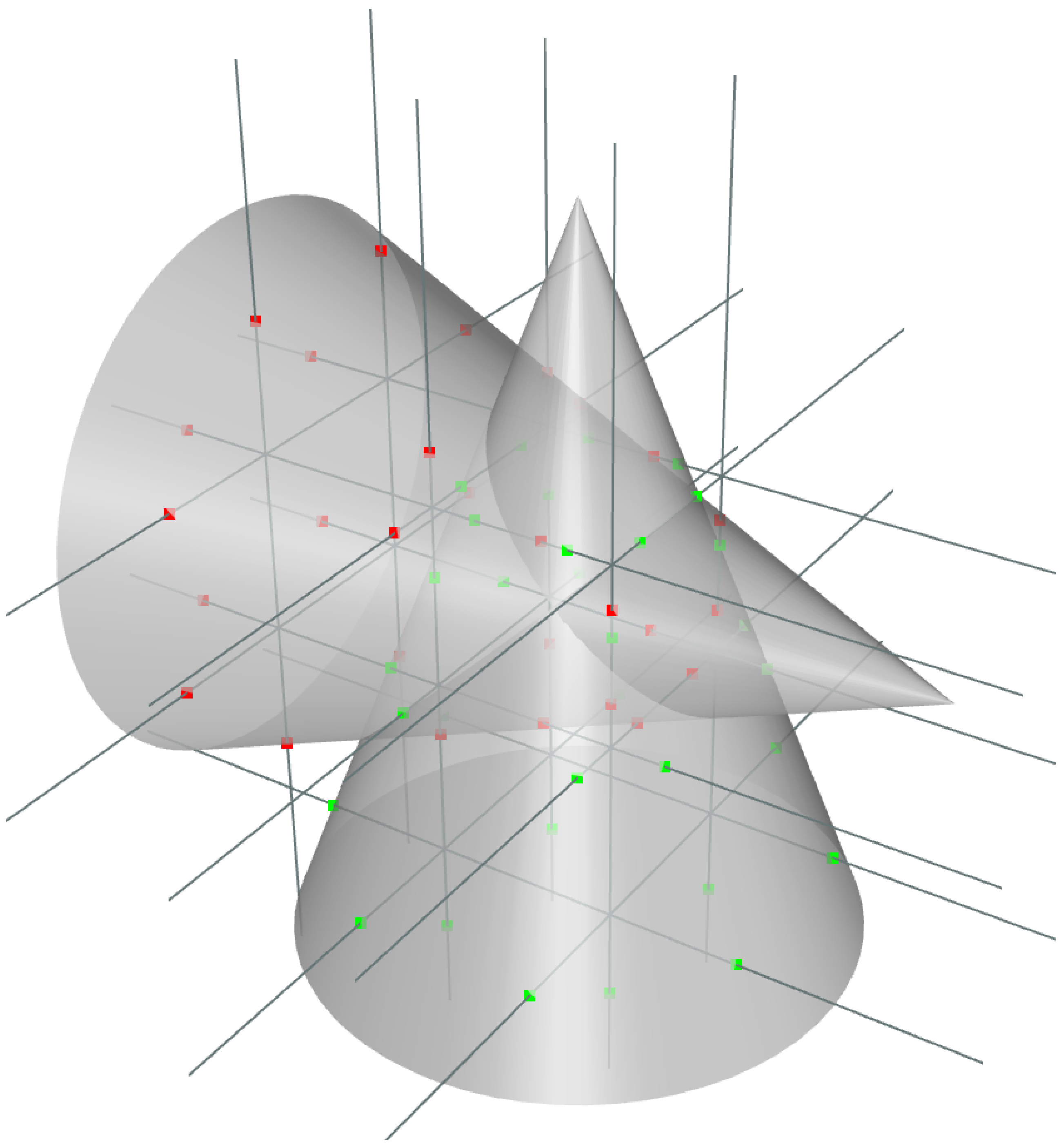


Figure 20: affine dependent intersection of a Hoffmann grid with cone object and rotated equivalent cone object.

6.3 A boundary surface intersection query

In all the commercial CAD systems tested during this research, it is possible to obtain the Cartesian point at which a projected line, vector, or ray intersects a boundary surface of a CAD model, see 3.12, Table 1: a selection of Common CAD program commands yielding surface coordinates via curve intersections. for details. Some of these CAD programs, such as FreeCAD, require several commands to return the Cartesian coordinates of an intersection point.

If many intersections are made with a native CAD geometry model, a point cloud is generated that describes the model surface. Two geometric models within their respective CAD environments can be compared using the similarity of the point clouds that they create. If the intersections of each model are identical and both models have the same orientation in Cartesian space, then each intersection point will have an equivalent in the other model that can be compared. A simple measure of geometric distance between these points will give an indication of the similarity of both models. The greater the distance separating equivalent point pairs, the more dissimilar the CAD models are in terms of an absolute geometric measure. The resultant two sets of point clouds will be within a defined tolerance if the intersecting mesh is identical and if the scale, position and orientation of the geometrical objects are identical. The method can identify a match between geometric models that are identical in both CAD model spaces, but fails on models differing in scale, orientation or position relative to an origin point. In other words, a simple comparison of intersection points will fail on identical models that differ by an affine transformation. Figure 20 shows a simple example of two identical cones, one of which is rotated relative to the other. It can be seen that a uniform grid will return different Cartesian points where it intersects the cone surfaces.

Hoffmann describes a generic method of querying a model surface with an intersected point. This is to create a regular orthogonal mesh of rays within a CAD model space and intersect them with a boundary surface model, the same operation is repeated in a second CAD model space with a nominally identical geometrical model (Hoffmann *et al*, 2014). The Hoffman intersection query is formally supported by a number of other queries on model precision. It is found that CAD programs do not typically enjoy a uniform precision but return differing values dependent on the scale and geometry of models. These variations are waived for the purposes of testing two models for similarity.

Setting aside the discrepancies of Cartesian point coordinates arising from numerical rounding errors and noise, this method returns a positive identification when used to compare two CAD feature functions for geometric similarity. For the sake of clarity, it will be assumed that a high level of geometric similarity corresponds to a minimal bidirectional Hausdorff distance between the returned point sets, this is referred to as *equivalent geometry* for brevity in the following section (see Chapter 3.11, Bidirectional Hausdorff metric for a description of this metric). Supposing two feature functions are selected from two heterogeneous CAD API, and that these feature functions are known to create features with functionally equivalent geometry. If these feature functions are then compared using a Hoffmann mesh intersection method, they will only return a result of geometric similarity if they share identical parameters and constraints that are assigned identical values. If this method is used to test for geometric equivalence between unknown feature functions, it is unlikely that the selected functions will generate equivalent geometries. It is possible to repeat the same test using different combinations of parameters until an equivalent geometry is detected, but this test would require an impractical number of match tests for feature functions requiring multiple parameters. To demonstrate how this method might provide a basis for a more practical approach, consider the following example.

Given two feature functions f_A, and f_B selected from two heterogeneous CAD programs, C_A and C_B. These two functions each have a set of parameters $\{a_1, a_2 \cdots, a_n\}$ and $\{b_1, b_2, \cdots, b_m\}$ respectively. In the case of an exact function equivalence, the parameters of the functions f_A, and f_B have exact same properties and order in both functions. In this instance, if these functions are tested with identical parameter values so that $a_1 = b_1,\ a_2 = b_2, \cdots\ a_n = b_m$, then the point comparison results from a Hoffman test will indicate an equivalent geometry.

Consider a subsequent instance where both selected functions again possess an equal number of parameters of the same type and property but the order of the parameters differs between functions. Unless all parameters are given the same value, the simple Hoffmann test is unlikely to return an equivalent geometry. This disparity can be assigned to two factors:

- Firstly, there is no guarantee that the size, position and location of geometric objects created by a CAD API function are equivalent, a Hoffmann test does not recognise affine transforms of scale, translation or rotation.
- Secondly, the geometric shape of the models produced by functions f_A, and f_B are unlikely to be similar without exactly equivalent parameter values, which are in turn dependent on an equivalent effect of the parameters. A circle function that is defined as tangential to three lines differs from a circle function that is defined as a centre point and radius value, or even a circle function defined as a centre point and a diameter value.

However, if the topologies of both geometric models are compared, the difference in function input parameters becomes a tractable problem. Once a point cloud matching technique can recognise topographical similarity between geometric models that have undergone an affine transformation, this allows geometric models to be tested for equivalence within their respective CAD programs. Feature functions that deform or modify existing models can similarly be compared using point cloud matching of model topologies. It must be noted that not all function parameter values may result in models that have recognisably similar topologies, these issues are explored further in Chapter 8.6, Parametric variables, where heuristics for function parameters are outlined.

6.4 Rigid body registration

A similar active research field already exists, that of scanned point cloud registration. In order to digitise the geometry of physical artefacts, the surfaces of objects are scanned to create a virtual representation. This process creates large sets of overlapping point clouds that require orientation and alignment. Successive scans of objects taken from different perspectives are knitted together to form a smooth unbroken virtual model. A laser or infra-red scanner generates multiple sets of 3D Cartesian point tuples as point clouds with a coherent relative orientation, but no absolute orientation values. To take these successive scans taken at different angles and merge overlapping scan portions correctly with respect to one another requires point cloud rotation, translation and possibly scaling.

Registration techniques divide into two categories, rigid body registration and non-rigid body registration. As comparisons are made to distinguish equivalent topologies in this application, only rigid body registration is relevant. Matching non-rigid shape geometries, such as articulated multi-part CAD models is beyond the scope of this research. For the purposes of determining similarity between CAD model geometries, it is sufficient to limit registration to that of rigid shapes so that 6 degrees of freedom describe the transformation. Distortions of CAD model geometries (e.g. stretching, bending, lofting) are inevitably the outcome of a geometric feature operation and may be identified as CAD features in their own right.

Laser scanners use rotating mirrors to scan across surfaces, this has the effect of creating relatively unpredictable sample point positions. Points from randomly peppered surfaces do not allow one-to-one point correspondences between adjacent scans. Attempting to minimise the difference between the point sets is not a simple linear problem with an exact solution[3]. To qualify this assertion, the number of parameters that could define a cost function of a 6 Degree-Of-Freedom system is much smaller than the number of sample points involved and thus the number of equations defining a reorientation.

6.5 Point cloud registration techniques

Affine transformations pose an equal problem for CAD model identification methods as they do for point cloud or polygon mesh similarity matching. The Cartesian coordinates

3. An exact solution is alternatively referred to as a closed-form expression.

that characterise geometric model data must be normalised, scaled and orientated before they can be directly compared.

The methods detailed in the following section describe the most salient techniques used for matching point cloud objects. The subject of geometry matching, geometry registration and matching shape models is particularly diverse. A representative description and evaluation of these techniques is tangential to this thesis and so has been moved to 3D Shape matching methods overview. None of the methods described in Appendix B promise a sufficiently accurate similarity evaluation required to verify CAD surface boundary similarity.

As stated above, much of the research in registration of Cartesian point clouds arises from the challenges of processing data points measured by 3D scanners. Laser scanners use rotating mirrors to scan across surfaces, meaning that the positioning of sample points is relatively unpredictable. Sample points from randomly peppered surfaces do not allow one-to-one point correspondences. Attempting to minimise the difference between the point sets is not a simple linear problem with an exact solution.

If the Euclidean distance between pairs of points is minimised, the best solution has the least overall global error. This Least Squares approach forms the basis of the most widely used methods such as *Iterative Closest Point*, an iterative descent method. In the most basic ICP point-to-point manifestation, one point cloud is manoeuvred relative to another by minimising the displacement between pairs of points between the source point cloud and the target point cloud (Besl & McKay, 1992). This approach assumes that the points closest to one another are corresponding points. For an iterative Least Squares approach to work the two point clouds must be comparatively well aligned to start with and the two distribution of points over the surface must have a relatively similar distribution. More robust variations of the ICP algorithm match points to local surfaces generated on the target point cloud, or discriminate against improbable point pair matches (Masuda & Yokoya, 1995; Trucco *et al,* 1999).

The other common registration method is *Principle Component Analysis*. PCA methods use point cloud mean values to solve translation, as this mean value should correspond to a centroid. Rotation is determined from the eigenvectors of the covariance matrix, which in geometric terms is akin to alignment with the longest axis of the scanned surface (Jolliffe, 1986). The accuracy of point cloud registration techniques are limited by the accuracy of point-to-point correspondences.

The PCA algorithm is a relatively intuitive concept of finding a dominant shape axis, this method may fail is in cases of symmetric shapes where the covariance matrix eigenvalues are equal. This can lead to an undetected reversal of the principal axis. Symmetric objects such as cylinders or spheres create similar issues, giving no clear dominant axis. Funkhouser *et al* observe that small differences in model extremities have a disproportionate effect on principle axes (Funkhouser *et al*, 2003).

The accuracy of point cloud registration techniques are limited by the accuracy of point-to-point correspondences. Most point cloud data is generated by surface scans, the position of sampled points is random and the density of sampled points depends on the orientation of the surface relative to the direction of the scanner beam. Therefore many techniques generate an approximation of a surface which is used for alignment. Xiong represents a point cloud as a probabilistic distribution function derived from a Gaussian Kernel Density Estimate (Xiong *et al*, 2013). Rotational Invariant Feature Transforms extends the 2D Scale Invariant Feature Transform to 3D, creating local histograms of gradients, similar to the histograms described in Rusu (Skelly & Sclaroff, 2007; Rusu *et al*, 2008). Histograms of shape geometry, statistical shape representations or shape representations independent of absolute position such as Spherical Fourier Transforms lose geometric data which limits their effective precision. See Appendix B, Section B for a more comprehensive overview of registration and matching methods.

Each of these techniques to match object geometry is limited by the accuracy of point clouds derived from scans of physical objects or tessellated surfaces. None are suited to the purpose of validating CAD model geometry equivalence as they lack the requisite precision for discrimination of complex surfaces. Properties unique to CAD geometry yield a novel method suited to accurate validation, these are described in the following section.

6.5 Point cloud registration techniques

Joshi and Chang defined an *Attributed Adjacency Graph* or AAG, an abstract structure of bounded CAD surfaces that could yield specific discriminatory features, this concept was extended to the *Multi-attributed Adjacency Graph* (MAAG) which includes the relative angle of adjacent surfaces to permit discrimination of a wider variety of model surfaces (Joshi & Chang, 1988). These methods are subject to the same limitations as common neutral formats, namely that they are reliant on the host CAD program implementation to export a uniformly interpreted representation. This technique is also referenced in Appendix B.16 - B.17.

6.6 Registration features search using surface intersection queries

Geometry registration techniques balance the requirements of efficiency against reliable identification of invariant geometry features (as opposed to CAD parametric features). The most simple and intuitive invariant features are regions of highest curvature, or greatest differentiation from surrounding regions (Bae *et al*, 2006; Sharp *et al*, 2002). Calculating areas of highest curvature is limited by the accuracy and distribution of sampling data. For a scattered point cloud, the density of points near distinctive areas is inadequate to allow them to be reliable. A second issue is that these small regions of interest are susceptible to sensor noise when derived from physical scans. There is a trade-off between robustness to noise and discrimination of small features in sampled data, but noise is far less of a consideration in points derived from CAD models.

Several important differences exist between points extracted from CAD surface geometry models and the points returned form physical scanners.

1. Unlike laser range scanners, CAD software returns points with a relatively consistent geometric error associated with the numerical noise and internal data representation. Range scanners return a proportion of incorrect points, requiring robust algorithms to discard outliers. The caveat is that CAD software is liable to produce errors of large magnitude on occasion. Tiny

surface holes, slivers and algorithmic artefacts will generate errors, although none were encountered during the course of this research.

2. Intersected surfaces are ‘transparent’ within a CAD model environment. A projected ray will return all points at which it intersects a surface. Scanned physical objects obtain surface points within a line of sight from the scanner, leading to ‘data holes’ where surface data from the rear of an object is missing.

3. CAD API intersection operations are relatively expensive and slow compared to the data arriving from range scanners. Each CAD API operation requires a negotiation via the API interface, usually the veteran Microsoft Component Object Model interface as the majority of CAD programs are supported on the Microsoft Windows series of Operating Systems (Docs.microsoft.com, 2018).

4. Points intersection operations within CAD environments can be directed in space. Point cloud generation via physical scanning is undirected, the distribution of points samples are dictated by the mechanism of the scanner. Within a CAD environment, the orientation of a ray or vector that can yield an intersection point is well defined.

6.7 Directed CAD feature point search

It is possible to project a multitude of evenly spaced rays from a model centroid to define a point cloud on the surface of a CAD geometry object. Hoffmann describes an evenly spaced grid or mesh to achieve the same purpose (Hoffmann *et al*, 2014). Leifman describes a similar method that records deviation from a sphere enclosing the object (Leifman *et al*, 2003). The representational accuracy of any intersected surface is proportional to the spacing of the rays, in a similar fashion as a graphics bitmap is limited to the resolution of the image. This represents a computationally expensive approach as the query efficiency is limited by the process of accessing the CAD program via an API interface.

A second approach might make use of existing CAD API functions to return lines or areas of high curvature, but this would rely on a less generally applicable approach. Bearing in mind that a querying a CAD boundary surface with intersections is a directed search rather than a pre-ordained point cloud, feature determination can proceed in a different manner using a minimal number of points, this is described below.

If points can be guaranteed to be evenly distributed on a boundary surface, then their mean Cartesian value forms a good approximation to the centroid of this geometric model. Translation transformation of two point cloud can be determined as the displacement of their relative centroids. Centroid displacement is unreliable where only partially overlapping point clouds are available or where deformed non-rigid models are compared, neither of which limit this particular application. Given a point P that represents a Cartesian tuple, the centroid of a dataset X containing N point values, the centroid can be determined as follows,

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$centroid_X = \frac{1}{N} \sum_{i=1}^{N} P_X^i$$

6.8 Global Registration via Singular Variable Decomposition

The global registration problem (alternatively known as the *Procrustes matching problem*) can be shown to have a closed form solution for a translation and rotation. Given one set of points A and a second set B that corresponds to a rotated and translated equivalent of point set A, registration can be expressed as a combination of rotation $\mathbf{R}$ and translation t,

$$B = \mathbf{R}A + t$$

Singular Variable Decomposition or SVD will decompose (or factorise) a matrix $\mathbf{E}$ into three matrices, it is useful to describe the matrices in terms of geometric operations. Using the mathematical notation given in Umeyama, $\mathbf{V}$ is an initial rotational matrix operation, $\mathbf{S}$ is a scaling matrix operation and $\mathbf{U}$ is a final matrix rotation operation (Umeyama, 1991).

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = SVD\left(\mathbf{E}\right) \qquad \text{where}$$

$$\mathbf{E} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

Two triplets of points, P_A, and P_B, can yield a rotation matrix via the SVD properties once they are corrected for translation. Again, translation is the euclidean distance between point cloud centroids, $centroid_A, centroid_B$. If the covariance matrix $\mathbf{H}$, is accumulated from these translation-normalised point clouds, it can be decomposed to give a rotation matrix $\mathbf{R}$

$$\mathbf{H} = \sum_{i=1}^{N=2} (P_A^i - centroid_A)(P_B^i - centroid_B)^T$$

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = SVD(\mathbf{H})$$

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T$$

This process determines the optimal orthogonal rotation matrix $\mathbf{R}$, however this matrix may represent reflections of the point set as well as rotations. If the determinant of $\mathbf{V}\mathbf{U}^T$is negative, this indicates that the rotation matrix contains a reflection. It can be shown that the next optimal rotation matrix value is attained as follows (Sorkine-Hornung &

Rabinovich, 2017). Note that the subsequent research suggests that this step is redundant if there is a consistent ordering of matched point pairs that respects chirality (see Section 6.17, Helical point sequencing).

$$\mathbf{M} = \mathbf{V}^T\mathbf{R}\mathbf{U} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & -1 \end{pmatrix}$$

giving,

$$\mathbf{R} = \mathbf{V}\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & -1 \end{pmatrix}\mathrm{U}^T$$

This may be generalised for rotation matrices with and without reflection as,

$$\mathbf{R} = \mathbf{V}\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & det(\mathbf{V}\mathbf{U}^T) \end{pmatrix}\mathrm{U}^T$$

The translation operation, t, may be subsequently derived from the rotation matrix and centroid values such that,

$$t = -\mathbf{R} \times centroid_A + centroid_B$$

It may be observed that the *Least-Squares* approach is a generalisation of the SVD solution for cases where point clouds A and B contain more than three points (Besl & McKay, 1992). In this case there are taken to be N corresponding points between both sets. The affine transform is defined as the least-squares minima of the overall error between corresponding points,

$$error = \sum_{i=1}^{N} \left\| \mathbf{R} P_A^i + t - P_B^i \right\|^2$$

This manifestation still requires that each point P_A^i in the set A has an analogue P_B^i in set B. This constraint rules out SVD matrix calculation registration for point clouds which lack pointwise correspondence between scans.

6.9 Object Point Cloud Registration and Object Recognition

The ICP, PCA and SVD algorithms described above are generally used in applications where point data sets under comparison are assumed to derive from the same model. Object recognition differs, in that the task is to identify a target object from dissimilar objects, particularly in cases when the object may be partly hidden or rotated in orientation. Discrimination between objects requires that multiple geometry models must be matched for equivalence, consequently an efficient algorithm should uniquely and minimally define a geometry model to allow rapid searching for equivalents within large sets of models.
The approach taken in this proposed identification method is to reduce the CAD models undergoing comparison to a minimal set of features that will reliably discriminate between models. The difficulty with this approach is determining the optimal minimisation of model details that returns both an acceptable comparison accuracy and comparison time. An associated problem is the reliable identification of relevant model features, for example if two cubes are being compared it is important that all corners of these cubes have been identified. The following section describes the aspects of geometric model features suited to comparison.

6.10 Geometric registration feature types

Equivalence comparison of CAD geometric models representations may be simply stated as determining similar models that may have undergone affine transformations and discriminating against dissimilar models. If these geometric models are represented by a set of identifying features, these features must invariant under transform while allowing a comparison of model topology. Identifying features might be defined as any point, curve or region that possesses unique characteristics. The term “feature” is used within CAD literature to represent parametric features that provide the defining building blocks of a CAD model (see Chapter 3.3, Parametric feature modelling), while the term “feature” within the domain of 3-D surface registration is simply a topological characteristic that permits accurate transformations.

Registration features are required to be both minimal and unique. Audette lists several feature categories, points, curves and regions and determines curvature to be the defining property (Audette *et al,* 2000). Several schema of registration features suited to matching CAD models are possible. As models are statistically more likely to have fewer regions of high curvature, these are obvious candidates. Curvature based methods are briefly described in Appendix B 12, Curvature based descriptors. Regions of highest curvature such as corners and edges may be taken from the convex hull that encloses the body, or from an API query that returns edges and corners. It is also a simple matter to extract this data from neutral file formats such as ISO10 303-23, or STEP. However it is necessary that this method is independent of any particular format in order to map feature functions between CAD API.

Distinguishing matching features must be determined by the minimal interface required by the point sampling methods described in CAD model geometry comparison, namely that these features can be determined via the Cartesian points returned by intersections with projected rays. As this method is intended to be a proof of concept rather than a provably efficient schema, the features that can be determined with a simple search method are chosen. In this instance, these are the furthest points situated from centroid of the shape. These points may be determined using an iterative search over the surface of the model to find the highest local maxima, this search method is explained in detail in Section 6.20, Registration feature search strategy. In Section 6.25, A minimal set of registration features types, further distinguishing features are described that allow a broader spectrum of shapes to be distinguished.

6.11 A progressive search refinement strategy

The use of unique discriminating feature points allows geometric models to be represented in progressively detailed description as follows.

6.11.1 Matching distinguishing feature sets:

In the simple case of using model corners to represent point features, an oblong shape of eight corners may be distinguished from a tetrahedral shape of four corners. However this rapid comparison will not distinguish a rectangular cuboid from a cube as both possess the same number of corners.

6.11.2 Displacements of each feature point from the shape centroid:

The distance from shape centroid to shape corners constitute another set of descriptors that are invariant to rotation and translation. These displacement descriptors are not invariant to scale transformations and must be normalised before comparison. For example each distance may be divided by the longest distance. These descriptors will distinguish a cube from a parallelepiped, but not from a rectangular cuboid. See Figure 21 which represents a match between two sets of centroid-feature displacements ordered by type. These displacements represent the geometry shown in Figure 22, Figure 23, Figure 26.

6.11.3 Transformation between sets of feature points:

The rotation, translation and scaling between sets of points can be carried out relatively quickly using a closed-form solution such as the SVD based rigid motion transform given in Section 6.8, Global Registration via Singular Variable Decomposition**.** Once the point sets are transformed to share the same centroid, scale and orientation, the sum of displacements between feature points indicates whether the points or corners share the same relative coordinates. Figure 22 shows two shapes with matching feature point sets. This test will distinguish between a rectangular cuboid and a cube, but it will provide no information about the surfaces that are not at the feature points. For instance a cube with radiused edges and corners will return a match for a cube with orthogonal edges and corners.

6.11.4 Checking via random transformed surface points:

Once the transformation rotation matrix and translation has been generated for feature registration points, it may be used to transform any intersection ray or point between source and target shapes.

In this case a point created by a ray emanating from the centroid of a source shape that generates an intersection with the shape surface may be transformed to its relative position on the surface of the target shape. If this transformed point is not tangential to the surface then it can be inferred that the shape surfaces are not equivalent. The orientation of the ray used to create the source point may be transformed to the target shape and an equivalent point generated on the surface of this target, rather than querying the tangency of a transformed point on the target surface, it is possible to use the displacement between the transformed point and the point created by the transformed ray to determine whether the shape surfaces coincide at the point. Figure 23 shows three random points transformed between two identical shapes. This measure is not suited to the unusual circumstance where the shape surface is coincidental with a ray emanating from the shape centroid.

6.12 Model difference measure via mapped points

Given a centroid point c_S on a source model and a point q on the surface of this model, a vector $\vec{q}$ can be defined from the source model centroid to q. This vector might then be scaled, rotated and translated to an estimated position on the target model,

$$\vec{p} = \mathbf{R}\vec{q} + t$$

via a rotation matrix $\mathbf{R}$ and a translation t (the scalar operation of scaling is omitted for clarity). If this vector $\vec{p}$ is based at the corresponding target centroid c_T, there is an equivalent surface intersection point p' that lies on $\vec{p}$ that must be coincident if the two models have identical geometrical features. A point p' may be created at the intersection of the target surface with $\vec{p}$ projected from c_T, giving a vector $\vec{p'}$. The scalar Euclidean displacement between $\vec{p}$ and $\vec{p'}$ is the error measurement between the source and target models at the point q.

$$\delta_q = \left| \vec{p} - \vec{p'} \right|$$

A selection of points qi give a higher probability of model surface equivalence according to the binomial probability distribution described in the next section. This random selection N of points q_i from the source model give a set of error displacements, which may be summed to provide a measure of similarity. Root-Mean-Square-Error is common method to normalise these error displacements to give a single error metric. This metric penalises outliers to a greater extent than measures such as Mean-Absolute-Error, making it more suited to registering anomalous δ_q values. This metric is used in preference to the bidirectional Hausdorff measure which is that of the most extreme outlier. In practice there is little difference when assigning an arbitrary threshold to indicate a match.

$$err_{RMSE} = \left[\frac{1}{N} \sum_{i=1}^{N} \delta_i^2 \right]^{\frac{1}{2}}$$

Selection of several random surface points cannot, by definition, guarantee that both shapes are equivalent for all surface points. It is however possible to assign a probability of both shapes being similar, proportional to the number of samples tested. This can be described in simple terms as the proportion of surface that differs between both shapes. If 50% of the surface area of the target shape differs from the source shape, there is a 50% chance of discovering this difference in a single sample. This may be generalised as an expression of the probability mass function where n is the number of samples, p is the proportion of differing surface area,

$$F(k;n,p) = \binom{n}{k} p^k (1-p)^{n-k}$$

For example, if $\frac{1}{6}$ of the target surface differs from the source surface and 6 samples are taken then the probability that no difference is encountered is,

$$f_{PMF}(0) = \frac{6! \times 1 \times \left(\frac{5}{6}\right)^6}{0! \times 6!} = \left[\tfrac{5}{6}\right]^6 = 0.33$$

Throughout this research, probability values are based on a random selection of point values from the surface, this approach lends itself to optimisation, as detailed at the end of this chapter.

These four tests for shape equivalence are complementary, a shape difference not detected by the first test may be detected by the second, if not by the second, then by the third, if not by the third, then by the final test. Performing these tests sequentially over shapes allows efficient sorting such that computationally expensive tests are reserved for shape discrimination tasks that cannot be resolved using more rapid techniques.

6.12 Model difference measure via mapped points

6.13 Software resolution and machine precision

One aspect of point matching that is briefly mentioned in Chapter 3.6.1, Numerical accuracy is the discrepancy in tolerance inherent within different CAD programs. Absolute geometric values must be approximated on finite precision computers and each CAD program has variations in the accuracy of spatial representation. This accuracy or resolution is generally expressed as an absolute value δ that represents the boundary within which a CAD program Cartesian representation and a similar queried value may be considered coincident.

Hoffmann references a *bounds query*, that returns the resolution ϵ_c integral to a CAD program alongside the separation distance, δ_0, between an ideal geometric model M_0 and the CAD representation M. In the experiments described in Chapter 6, Test configuration for single and multiple model matches, the tolerances of the CAD programs used are several orders of magnitude smaller than the scales of the models used for test evaluation, a simple minimal threshold distance validates point coincidence. This validation is summed over all mapped point displacements, similar to the minimisation of least-squares error.

$$\delta_{error} = \frac{1}{N}\sum_{i=1}^{N} \left\|(P_A^i - P_B^i)\right\|$$

The mean deviation of a set of registration features, or random points that have been mapped are tested against an absolute threshold value to determine whether the two point sets can be considered as coincident. This threshold value may vary between different CAD programs, or even over varying surface topology. A measurement of these variations is considered outside the scope of this research.

Source and target shape models are assigned a measure of error distance, ranging from zero, where a source and target model are deemed to be geometrically identical independent of any affine transformation, to infinity, where there is insufficient similar elements identified to allow further determination of geometric shape equivalence. In practice this value is composed of the RMSE value test points transformed from the source model to the target model surface and infinity for values that do not have sufficient matching feature registration points to create a transform.

$$err = \begin{cases} err_{RSME}, & \text{if } N \geq N_{min} \\ \infty, & \text{if } N < N_{min} \end{cases}$$

Once the registration features of the geometry model are located, the relative displacements from the model centroid to these features form a unique shape signature that is invariant under rotation and translation, and is proportional under a scaling transform. These maxima-centroid displacements can be represented as bin values within a histogram. The immediate advantage of using an affine-invariant histogram (or vector, depending on domain terminology) to represent a shape configuration lies is the ability to rapidly compare a source histogram against multiple target histograms (see Figure 21). This can be achieved with a cosine measure or a specific histogram distance measure such as the intersection distance used in the implementation (`distanceRank`)(Swain & Ballard, 1991). Zehtaban *et al* describe a similar alphanumeric shape signature for CAD models with orientation-invariant numerical values comparable via a distance measure (Zehtaban *et al*, 2016).

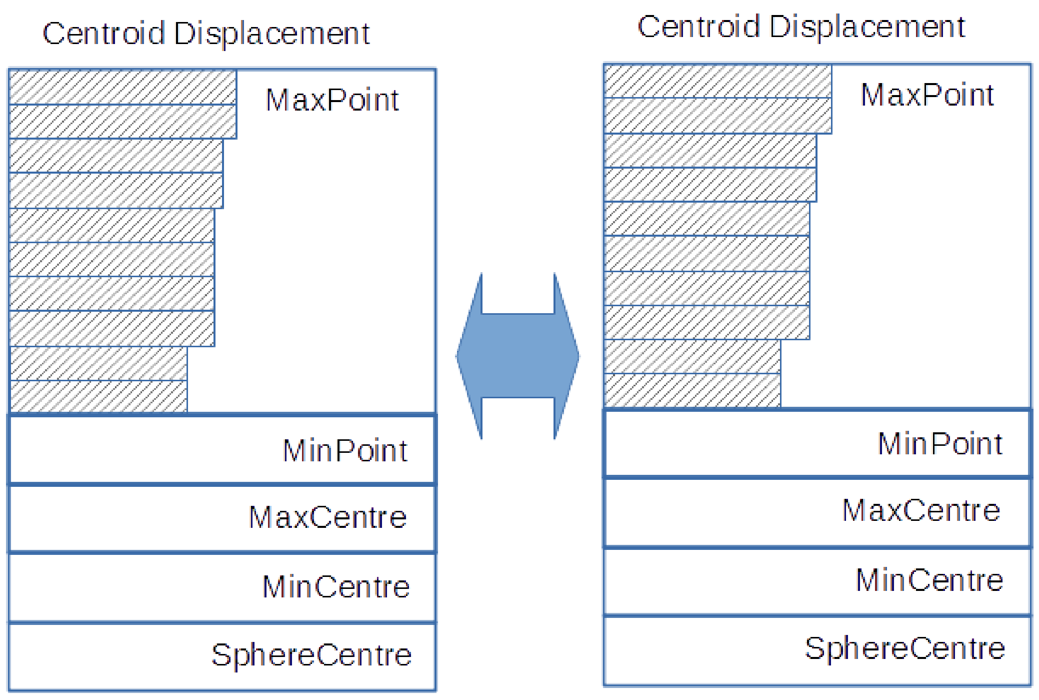


Figure 21: comparison of categories of registration feature displacements from a model centroid form an orientation-neutral histogram for rapid comparison.

After the search for registration features on an identical source and target model has terminated, the number of these features should be identical and the euclidean displacement from each feature to the model centroid should be equivalent. If these displacements are summed between identical geometry models the histogram bins will contain even numbers. Should a high proportion of histogram bins contain even numbers, it then becomes worthwhile testing the missing point values of the bins of uneven number. Note that the comparison between CAD models can be directed, unlike a physical scan cloud. If a point exists in a source CAD model but has not been found in the target model, the target model can be searched at the corresponding point. If no corresponding point is detected within a predicted location, the comparison test can be terminated, yielding a known mismatch.

6.13 Software resolution and machine precision

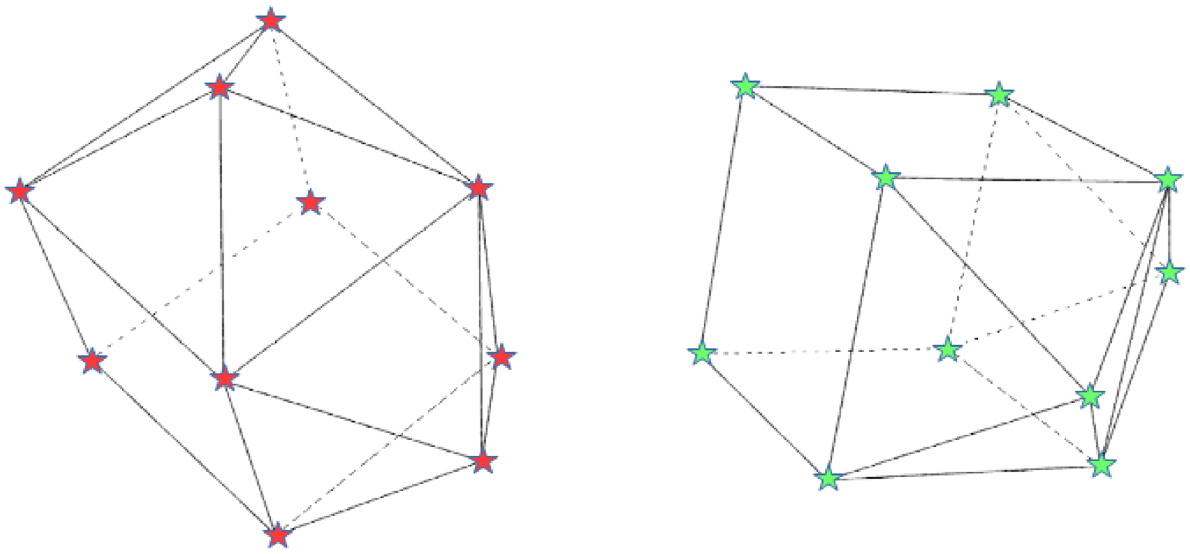


Figure 22: Singular Vector Decomposition of registration feature points yields a rotation matrix and translation between source and target models.

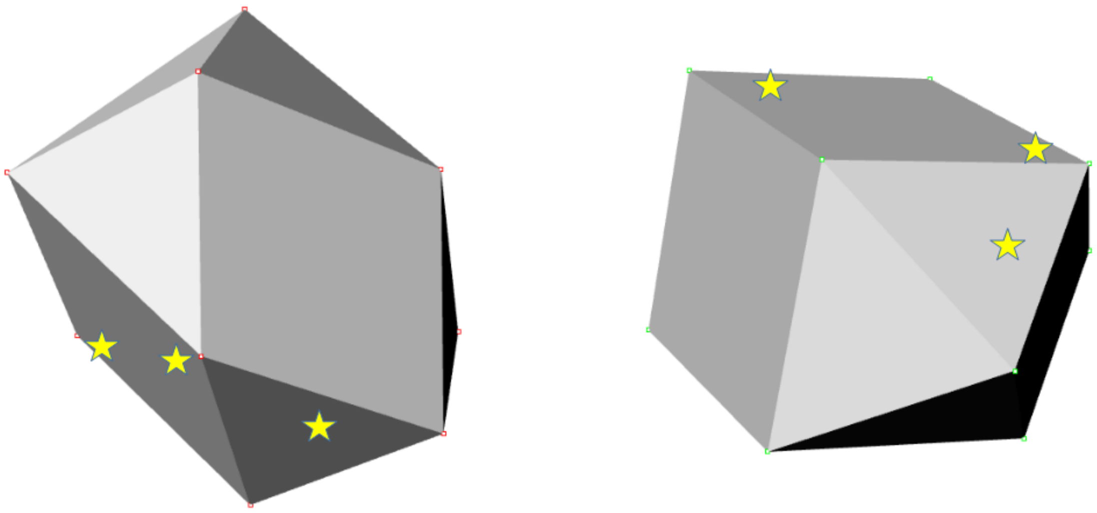


Figure 23: random points are transformed from source to target model, summed deviations from estimated surface intersections give a model similarity value.

6.13 Software resolution and machine precision

6.14 Registration feature search reliability

The number of extrema features is an unknown initial quantity. Determining registration features using a directed search rather than an exhaustive search carries the risk of missing features that have a low probability of being detected using a hill-climb algorithm. To describe the issue in detail requires a description of local search reliability.

A simple registration feature search scheme subdivides the geometry model into search regions. If multiple extrema searches within the same region terminate at the same point or points, then it is more likely that all extrema points within the search location have been identified. To use a simple probability example, if a coin is tossed 10 times, there is only a 2^{10} probability that heads appears 10 times in a row. In a similar fashion, if there are two features of identical geometry within a search region on which searches reliably converge such as in the image above (see Figure 24), then there is the same 1 in 2^n chance of missing a feature in n searches. The problem with this method is that only platonic solids have search regions of equal size where a hill-climb search terminates with an equal probability. Figure 24 shows a surface with equal probability and the adjacent image, Figure 25, a surface with unequal probability of hill-climb searches terminating in both surface features.

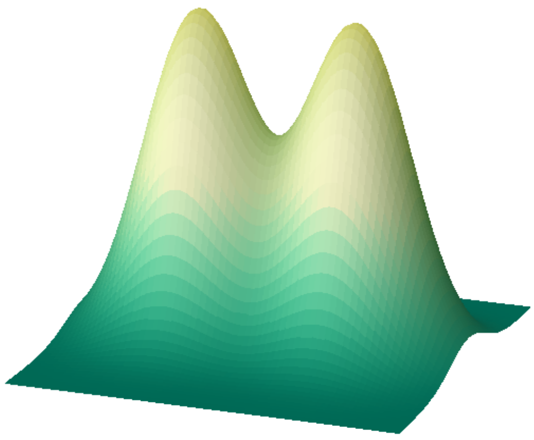


Figure 24: surface representation containing two regions with an equal probability of discovery.

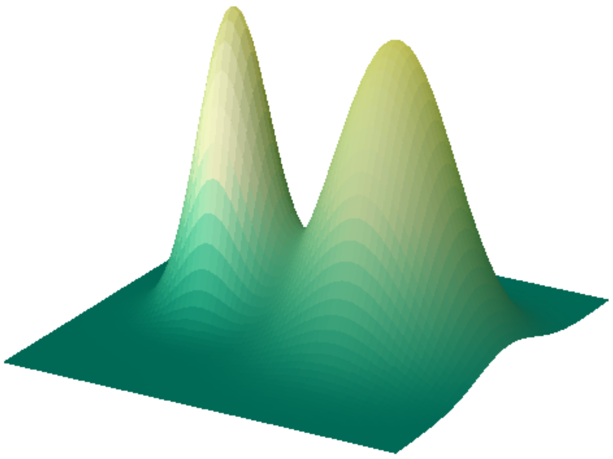


Figure 25: surface representation containing two regions with unequal probability of discovery.

6.15 Registration feature search repeatability

Search reliability is one factor, search repeatability is the second. If the same algorithm is used within both CAD programs to determine registration features, are the results repeatable on similar geometries? This factor illustrates a weakness in steepest-gradient searches, if the gradient undergoes a sharp discontinuity, a random search might not encounter it and terminate. To illustrate this concept with an example, supposing a search for maximum extrema features is performed on a plane with a few sparsely distributed thin needles protruding from the surface. A search starting from a level surface does not move towards a protuberance and may eventually terminate without ever encountering a sharp protrusion. This shape cannot be reliably distinguished from a level plane using this technique.

The unknown probabilities of registration feature discovery signify a weakness inherent in a simple feature counting method for matching shape models. If the number of registration features are unknown in advance, and a termination condition for a feature search is based on the number of times that independent searches discovers the same feature, then repeated searches on the same shape will not reliably yield a deterministic number of features.

This problem is not applicable to the other three search sequential techniques, therefore a robust matching heuristic must afford some latitude to the number of feature points discovered. Rather than all registration feature points being exhaustively discovered, it is a sufficient condition that features with a high probability of discovery are found. The attendant issue of incomplete sets of matching point pairs may be resolved by removing a counterpart feature from a target shape that is missing in the source shape. The tests that match centroid displacements provides the basis for a first assessment to sift likely matches.

Tests on the benchmark CAD shape libraries reveal the general utility of CAD matching algorithms, but have limited application for feature testing algorithms. The variety of simple shapes allow random search techniques to excel, yet the majority of CAD models are of complex, asymmetric geometry. The issues of model complexity are elaborated further in Chapter 7.5, Complex model matching.

6.16 SVD exhaustive feature search requirement

The advantage of SVD methods to find affine transforms is that it is an order of magnitude faster than other geometry registration methods such as ICP which iterate transformation calculation to minimise error between source and target models. The disadvantage of SVD is that the exact transformation calculation is dependent on paired registration feature points mapped between the source model and the target model. If the shapes undergoing match testing have registration point sets that do not have pairwise correspondences, a SVD algorithm is unlikely to find an existing mapping, consequently this method is inadequate for determining shape matching using unordered sets of source and target geometry coordinates. However, if identical registration features can be found on both models and these features can be paired between models, the closed form SVD calculation may be used.

As the hill-climb search for registration features is likely to finds regions with a similar probability of detection with a similar frequency, it is possible to create sets of registration features common to both source and target models. Features that do not lie within the intersection of both sets may be found by matching the scalar value of centroid-feature displacement values common to features on both models (`listIntersection`). Both sets of feature displacement values need first to be normalised to a common scale, dividing each value within by the largest value of the respective set will accomplish this. If this operation leaves sufficient feature pairs to perform SVD calculation, the next step is to find a sequential arrangement of these feature pairs that is verified to exist on both source and target model, as described in the next section.

6.17 Helical point sequencing

The method used is to order all points on each model according to a helical arrangement, this is accomplished as follows.

1. The first step is to determine a registration feature that has a unique point-to-centroid displacement value to serve as a starting point. In the case where there is no registration feature with a unique centroid displacement value, the task is then to determine the smallest set of registration feature displacements.
2. Once a start point is chosen from this set, the registration features are translated so that this start point forms a new coordinate origin.
3. The registration features are rotated so that the model centroid lies co-linear with the Z-axis emanating from the start point. This may be carried out via a rotation through the axis generated by the cross product of a vertical unit vector and a normalised vector constructed from the starting point and the centroid. The rotation angle may be determined from the arc-cosine of these same vectors, there are several exceptions for small values and start points located on the Z-axis. This translation and rotation allows the registration feature points to be converted to cylindrical coordinates using a signed arctangent function and Z-axis displacement.
4. Once registration features are represented by an angular value and a displacement value from the start point, they may be sorted by cylindrical coordinates to determine a sequence of features that lie in a helical path. If two points share the same Z-axis coordinate value, the point with the lower angular value takes precedence, otherwise the point with the lower Z-coordinate takes precedence in the sequence. This algorithm will generate identical sequences of registration points from asymmetric matching target and source models that share the same start point. It is necessary that both sequences share the same chirality, or "handedness" of generative helix. This procedure generates a known order of feature registration points, as shown in the accompanying diagram, Figure 26. This illustration indicates

the sequence of extrema feature points discovered from a shape orientated such that the point A is co-linear with the Z-axis.

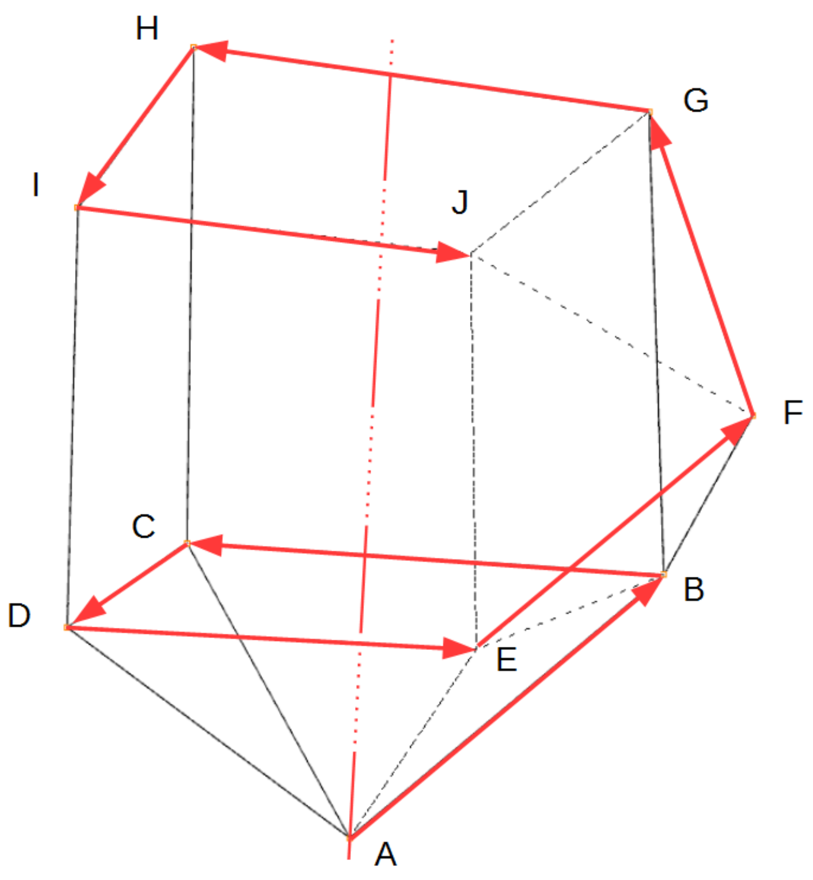


Figure 26: a sequence of points is generated from an initial feature point A on a model geometry to form the series ABCDEFGHIJ

In the case of models with no unique registration feature displacements, there is no option but to generate multiple sequences from the smallest set of points that share an identical displacement value from a model centroid. This gives several candidate helical sequences of feature points for both source and target models. These sequences may then be tested for an identical set of centroid displacement values to determine a sequence pairing. The implementation calculates a single registration point sequence (`leftHandSpiralOrder5`) for a source model with the associated sequence of point displacements, target model sequences are then generated until one matches the displacement sequence associated with the source model sequence. This process may be formalised as follows.

6.17 Helical point sequencing

Algorithm 1: Ordered registration point sequence

```
Given:    $F_S = \{(\boldsymbol{F_S})_l\}_{l=1}^{N_S}$ and $F_T = \{(\boldsymbol{F_T})_l\}_{l=1}^{N_T}$, two sets of registration
          features on model surfaces $S$ and $T$ respectively
Generate: $D_S = \{(\boldsymbol{D_S})_l\}_{l=1}^{N_S}$ and $D_T = \{(\boldsymbol{D_T})_l\}_{l=1}^{N_T}$, two sets of scalar
      displacements between each value in $F_S$ and $F_T$ and their respective
      centroid point values $c_S$, $c_T$.
For each set $D_M$ in {$D_S, D_T$}, $F_M$ in {$F_S, F_T$} and $c_M$ in {$c_S, c_T$},
    Find the set $G_M$ of registration features $f_M$ corresponding to the minimum values in $D_M$
        such that $\left|\overrightarrow{f_M, c_M}\right| \subset \min(D_M)$ and that $\forall \{d_i, d_j\} \in D_M, d_i = d_j$
    For each initial point $g$ in $G_M$,
        Translate $F_M$ point values by $\vec{g} = \overrightarrow{c_M, -g}$ such that $g$ is centred at the
            coordinate system origin.
        Rotate all points in $F_M$ such that the rotated centroid, $c_M$,is co-linear with the
            positive Z axis.
        For each $f$ in $F_M$, convert the point Cartesian coordinates to Cylindrical
            coordinates.
        Sort $f_i(z_i, \alpha_i)$, $f_j(z_j, \alpha_j)$for $i$, $j$ in $F_M$ and corresponding $d_i$, $d_j$ in $D_M$ such
            that,
            If ( $z_i = z_j$ and $\alpha_i > \alpha_j$) or ($z_i > z_j$)
                $k \leftarrow i, i \leftarrow j, j \leftarrow k$
        Add each ordered set $(F_M)_g$ to $F_M^G$ such that $F_M^G = \left\{(F_M)_g\right\}_{g=1}^{G_M}$
        Add each ordered set $(D_M)_g$ to $D_M^G$ such that $D_M^G = \left\{(D_T)_g\right\}_{g=1}^{G_M}$
For each ordered set $(D_S)_g$ in $D_S^G$ and $(D_T)_g$ in $D_T^G$,
    If $d_i = d_j$ for $d_i$ in $(D_S)_g$ and $d_j$ in $(D_T)_g$,
        Add respective feature sets $\{(F_S)_g, (F_T)_g\}$ to paired match set $P_{FS}$
```

6.17 Helical point sequencing

6.18 Order ambiguity arising from rotational symmetry

This schema will generate unique sequences for an asymmetric model, but not in cases of radially symmetric models. In the simple case of a regular octahedron that has 4-fold rotational symmetry, four individual paths are possible, each of which share an identical centroid displacement sequence. This ambiguity is caused by arbitrary absolute angular values created in the conversion to cylindrical coordinates. A target model in an unknown orientation cannot provide a datum point for rotationally symmetric feature registration values. Therefore, while the sequence of symmetric feature points arranged around an axis of rotation co-incidental with a model centroid will generate a sequence ordered by relative angular position, the first feature registration point in this sequence may be any one of the rotationally symmetric points. This ambiguity does not pose a problem for registration feature point matching, but where there are further model matching details that are not captured by registration features, this property may indicate a false negative.

This registration feature matching schema is sensitive to the position of the model centroid. Model centroids are generated from the mean values of registration feature points, covered in more detail in Section 6.27. If, however, centroids are assigned to rotation or reflection axes, the schema above will fail on models with rotationally symmetric registration features combined with asymmetric registration features. A centroid centred on a rotational axis will introduce the path ambiguity described, but asymmetric registration features will then add a rotational orientation matching constraint. A centroid created from mean values of points incorporates the asymmetrical perturbation of points that are not rotationally symmetric, allowing the registration feature matching algorithm to discriminate between helical path sequences. The 10-point shapes used to illustrate helical point sequence ordering exhibit this property (Figure 22, Figure 23, Figure 26). This shape can be described as a cube with additional points in the centre of two adjacent faces. If the centroid is arbitrarily placed at the position of the cube centroid and the order of points is selected from one of these two additional extrema as in Figure 26, the initial order of points might progress via any of the paths, e.g. AB, AC, AD, AE. This may result in a sequence $\overline{\underline{\mathbf{A}}}$CDEB$\overline{\underline{\mathbf{F}}}$HIJG where the underlined point labels have a different vertex-centroid displacement than the other point labels. Once this is compared with $\overline{\underline{\mathbf{A}}}$BCDE$\overline{\underline{\mathbf{F}}}$GHIJ, it can be seen that the displacement values for both sequences are equivalent but they do not record the same sequence of points.

In cases where there are several sets of paired feature set matches, $\{(F_S)_g, (F_T)_g\}$ that share the same sequence of displacement values, this indicates a pair of models that are both symmetrical around a plane.

In the case of models where there is a high confidence that all feature points have been determined, only one of these mapped pairs are required. However in cases where this certainty does not exist, such as the complex bracket described later in Chapter 7.5, Complex model matching, the registration feature points on the convex hull of the model do not account for undetected feature points within the model. In this case the only the discovered feature points are symmetrical about a plane. This uncertainty requires that all paired sets of feature points are tested and then subsequently verified using random points transformed from the source model to the target model (this algorithm is implemented in `leftHandSpiralOrder2` where the minutiae of vector calculation and rotation is contained in the subroutine `LHHelixOrder`).

6.19 Correction for SVD reflections about a plane

A final observation on this SVD matching schema is the detection and rectification of reflection within rotation matrices derived from the SVD algorithm, detailed in Section 6.8, Global Registration via Singular Variable Decomposition**.** To recap, if the determinant of $\mathbf{VU}^T$is negative, this registers the presence of an unwanted reflection in the rotation matrix. The Umeyama method that reverses the sign of the outermost diagonal in the identity matrix is found to cause an unwanted reflection in the rotation matrix. However, if the indicated presence of an unwanted reflection were ignored, the generated rotation matrix was correct for all test examples trialled. This unusual observation was not pursued further.

6.20 Registration feature search strategy

The search used to find feature points is based on a steepest ascent hill-climbing algorithm. For every point found to have the maximum displacement from the model centroid, a ring of neighbouring points is generated, and from this ring, the neighbouring point with a greater centroid displacement is chosen as the next point. As this iterative search for local maxima happens in a local search sector, the displacement between search steps starts at a value that distributes neighbouring points over the most of the search sector and reduces the step size with each iteration to converge at a resolution suited to CAD geometry comparison.
To describe this process in more detail, it can be dismantled into a series of operations as follows.

1. A model centroid is estimated, starting from the CAD model space origin.
2. A number of evenly distributed rays are projected from the model centroid to the boundary surface of the model. The intersection of these rays with the model boundary form the seed points of independent registration feature searches.
3. Each point seed generates a ring of neighbours, from which the next most suitable search point is selected.
4. The search starting from each seed point terminates after a set number of cycles, or once a feature is identified.

6.21 Preliminary centroid identification

A model queried for shape similarity is of unknown geometry, at an unknown position and orientation within a CAD model space. To identify the model centroid, the model must first be detected. As the methods available for model discovery are limited to point intersections with projected rays, an array of rays are projected in all directions from the origin to intersect the model surface. The Deserno regular method used is described in Section 6.22, **Equidistant spherical projection**. This method is sensitive to the density of projected vectors and the scale of the geometry model. Any surface intersections indicate the presence of a model, the centroid of this model may then be estimated.

If the Cartesian values of existing point intersections are averaged to form a mean point, this position forms a closer estimate to the model shape centroid, a similar spherical projection of rays is likely to intersect more of the model surface, generating a correspondingly more accurate estimate for a centroid. This process is iteratively repeated until the distance between centroid estimate adjustments falls below a set threshold value (`centroidTranslation`). Where the target model is a number of separate model geometries, the centroid will represent the mean points of intersection for all surfaces.

Algorithm 2: Centroid detection for unknown geometric surfaces

```
Given: a geometric surface model $M$
Given: assigned constant values $N_0$, $N_{max}$, $\delta_{min}$
Given: an initial number $N_0$ of points, $p$ on a unit Deserno sphere, $P_D = \{(\boldsymbol{P_D})_i\}_{i=1}^{N_0}$
Assign the initial centroid $c_0$ position to the Cartesian origin, $O$
Generate projection vectors, $\vec{p_i}$ from $c_0$ and $p_i$ for $p_i$ in $P_D$, such that $\vec{p_i} = \overrightarrow{p_i, c_0}$
While $P_D \cap M = \emptyset$ and $N \leq N_{max}$ do,
    $N \leftarrow N + 1$
    Generate $N$ points on the unit Deserno sphere, $P_D = \{(\boldsymbol{P_D})_i\}_{i=1}^{N}$
    Generate projection vectors, $\vec{p_i}$ for $p_i$ in $P_D$, such that $\vec{p_i} = \overrightarrow{c_0, p_i}$
    Test for intersection points, $q_i$ between vector $\vec{p_i}$ and model surface $M$
If $P_D \cap M \neq \emptyset$ then,
    While $\delta \leq \delta_{min}$ do,
        $c_{mean} = \frac{1}{M} \sum_{i=1}^{M} q_i$        # get mean value of intersections
        $\delta \leftarrow |c - c_{mean}|$     # \delta is displacement between centroid & mean value
        $c \leftarrow c_{mean}$    # centroid position is updated to mean value
```

6.21 Preliminary centroid identification

6.22 Equidistant spherical projection

Initial model detection and search initialisation seeding routines require evenly distributed vectors emanating from a point in 3D space. This distribution problem is equivalent to creating equidistant points on a unit sphere.

This problem is approached in two ways. There is the *Tammes problem*, which is to determine an arrangement of a fixed number of points on a sphere which maximizes the minimum distance between any two points.

The second method is the *surface Coloumb problem*, finding a stable distribution of a fixed number of mutually repellent point charges in the surface of a sphere, which equates to the same problem (Erber & Hockney, 1991). Two existing methods are tested, the spiral-point algorithm developed by Rahkmanov and modified by Thomsen, and the method described by Deserno that places points on evenly spaced rows on a sphere (Rakhmanov, 1994; Thomsen, 2012; Deserno, 2004). Both these methods arrange points with even linear spacing, either as a helix or as rows. Tests indicate that the Deserno regular distribution gives a more regular point distribution near the poles of these distributions.

6.23 Model search regions

The search for registration features commences from individual search seed points. Ideally the surface of the geometric model would be divided into search regions of equal area, allowing each individual search an equivalent probability of finding a local registration feature. However the surface geometry of a random model is not known in advance, so a search heuristic must divide up regions according to regions defined by the angle between rays projected from the centroid (`randomDistIntersects`). The Deserno regular distribution of projected rays defines these arcs according to the number of points specified on a unit sphere (Section 6.22). An array of projections emanating from a single point will cause the local search regions to be smaller on surfaces close to the centroid than on surfaces relatively distant from the centroid. This results in a lower probability of discriminating between nearby features that are relatively distant from the centroid.

Elongated shapes give rise to registration features that have varying probabilities of detection, similar to the variations in probability of detection in Section 6.14.

This may be considered the same problem as detecting when all relevant search points have been found; when all points have the same statistical significance of being found, such as points on Platonic solids with corners that are equidistant from a centroid, then a search may be terminated once all registration features have been located more than a set number of times. Once a model contains features that have a different probability of being found then there is a choice between attempting repeated searches to exhaustively find all points or searches that detect all points likely to be found within a set number of searches. Without prior knowledge of the shape geometry, or the associated probability of finding all shape registration features, an exhaustive search is not guaranteed to complete.

If a surface is subjected to multiple searches and the same features are discovered each time, then it is likely that an adequate number of searches have been conducted. If, on the other hand, the same number of searches discover different points each time, it is less likely that all registration features have been discovered. This may be described again as a binomial probability distribution, as a simple example two searches in a region with a single feature will always return the same feature, however two searches in a region with ten features each having an equal probability of discovery will only have a 0.1 chance of encountering the same feature twice in a row. In the implementation used for testing, a ratio of the number of features discovered multiple times against the total number of features found is used to evaluate a termination condition (`pointsToBins2`, `pointsToCurves2`).
This approach is sensitive to the values of termination condition used, models may have features that are only revealed at a higher number of searches. When models are evaluated for registration features using differing parameters such as higher density of seed points or a higher confidence threshold, they may present an unmatched number of features that preventing simple matching. This issue is ameliorated by the use of preliminary searches that use both registration feature type and centroid displacement to detect a potential

shape match. This method of selecting the set intersection of matching registration features between both models is covered in more detail in Section 6.28 (`listIntersection, listTypeOrderMatch`).

6.24 Hill-climbing search for registration features

The method used to search for the most simple type of registration feature, a local maximum point may be described here. In practice, the search algorithm is similar to a hill-climbing algorithm with adaptive step size. Each individual search for a maximum registration feature starts from a seed point and progresses in the direction of steepest relative gradient. Unlike steepest gradient descent, the function describing the CAD model surface is unknown, therefore the surface derivative is also unknown. This means that the relative gradient must be sampled from neighbouring points. In the implementation used, these neighbouring points are a ring of eight points sampled around the initial point.

The initial ray intersecting the seed point may be rotated in eight cardinal directions to create these new surface intersection points similar to a compass rosette (`neighbourGrid`, `rotationMatrix`). This pattern is shown in Figure 27. The initial angle is half that of the angle between vectors emanating from the Deserno projection, allowing each seed point to encompass the entire area of a local search region within two iterations (`getModelFeatures2, searchFeatures, getMaxMin`).

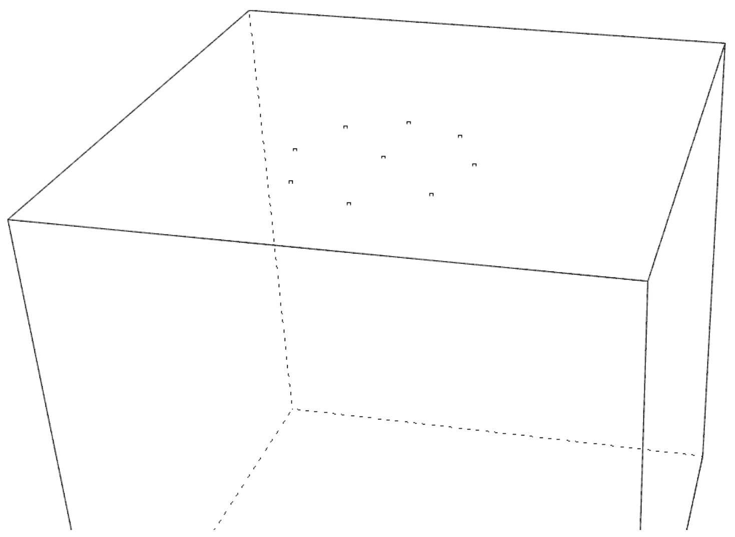


Figure 27: initial search point surrounded by rosette of generated neighbouring points.

This angle at which surrounding points are projected is halved at each iteration to reduce the diameter of the search pattern when the central point has a higher value than the surrounding rosette of points. Where one of the neighbouring points has the highest value, it is assigned as the next central point, but the angle is not subdivided for the next rosette of points. This strategy is combined with a rotation of the points rosette around an axis co-linear with the projected ray through the centre point on each rosette generation (`rotateCluster`). The rosette is rotated by $\phi/8$, which has the effect that the rosette points always occupy a novel rotational angle on each iteration. This additional operation allows the search to progress along sharp edges and at corners where the limited resolution of eight surrounding points might otherwise miss the highest local gradient. These iterations continue until either a feature is located, or the angle between points is lower than a set minimum. An absolute limit of iterations is set to prevent excessively long search sequences where the search undergoes repetitive inconclusive cycles. Figure 28 shows these maximum displacement points in a number of iterations as the search converges on a model extrema.

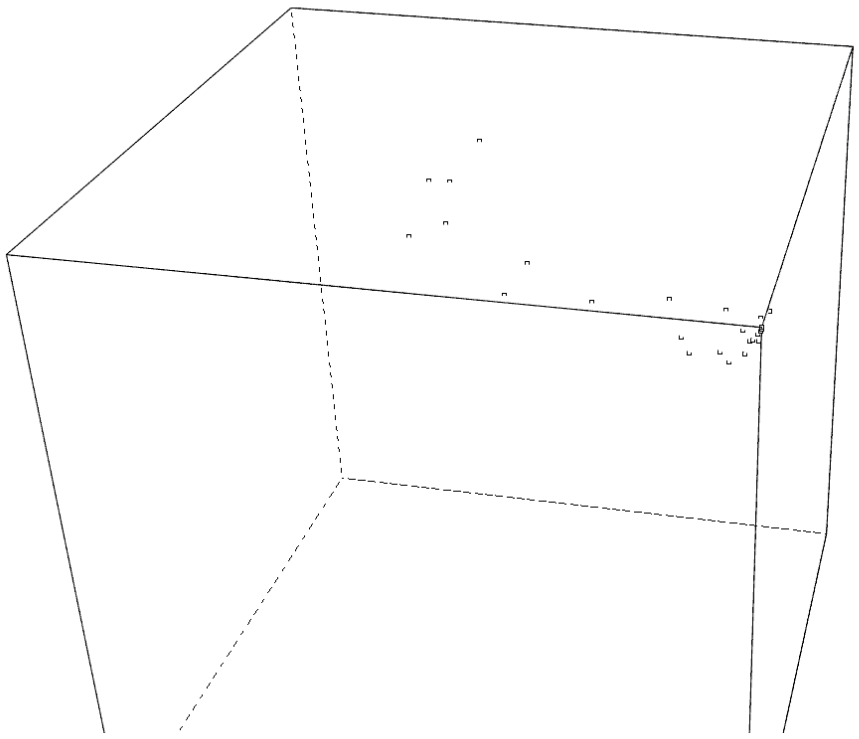


Figure 28: a series of iterated points selected for a maximum displacement from the model centroid as they converge on a local maximum corner region.

Algorithm 3: Maxima Feature detection for unknown geometric models

```
Given: a geometric surface model $M$
Given: a point $s_0$ at the intersection of the model, $M$ and a vector, $\vec{p}$ projected from the
    model centroid, $c_0$
Given: assigned constants $N$, $\alpha_0$, $\alpha_{min}$, $iter_0$, $iter_{max}$
Given: a plane, $B_0$, passing through the point $c_0$, $s_0$ and any orthographic axis,
Generate N vectors $\vec{q}$ at an angle $\alpha_0$ from $\vec{p}$ passing through angles of $\frac{2\pi}{N}$ arrayed radially around
an axis formed by $\vec{p}$ with one member, $\vec{q_0}$ co-linear with plane $B_0$
Get points $s_i$ formed at intersections between $Q_N \cap M$ such that $S_N = \{s_1, s_2 \ldots s_N\}$
While $\alpha_i \leq \alpha_{min}$ and $iter \leq iter_{max}$ do,
    Get displacements $D_R$ of $\{s_1, s_2, \ldots s_N\}$ from centroid, $c_0$ such that
    $D_R = \{|s_1, c_1|, |s_2, c_2| \ldots |s_N, c_N|\}$
```

```
Get displacement $d_0$ of $s_0$ from centroid, $c_0$ such that $d_0 = |s_0, c_0|$
Get $D_{peak}$ such that $d_i > d_0$ for each element $d_i$ of $D_{peak}$
    $D_{peak} = \{d_i : d_i > d_0 \text{ for all } d_i \in D_R\}$
If $d_0 = max(D_N)$ then,    # centre point has greatest displacement
    $\alpha \leftarrow \frac{\alpha}{2}$                  # reduce angle between central point and offset points
Else
    Case: $card(D_{peak}) \in \{1, 3\}$,
        $s_0 \leftarrow s_i$ ,               # central search point updated to highest point
    Case of $card(D_{peak}) = 2$,
        $s_0 \leftarrow s_i$        such that $d_i = max(D_N)$
        $\alpha \leftarrow \frac{\alpha}{2}$          # reduce angle between central point and offset points
    Case of $card(D_{peak}) < 3$,
        $\alpha \leftarrow \frac{\alpha}{2}$          # reduce angle between central point and offset points
$iter \leftarrow iter + 1$              # increment search counter
$\alpha_0 \leftarrow \alpha_0 + \frac{\phi}{N}$             # increment start angle $\alpha_0$ for radial vector array $Q_N$
Generate N vectors $\vec{q}$ at an angle $\alpha_0$ from $\vec{p}$ passing through angles of $\frac{2\pi}{N}$ arrayed
radially around an axis formed by $\vec{p}$
```

Finally there is a branching termination condition that directs the search for different types of registration features if there are inadequate registration features to perform a SVD based mapping, these differing classes of registration features are introduces in Section 6.26. These different classes of registration feature points, such as surface minima and arc centre points require a more complex decision criteria than simple comparison of the displacements between points and the model centroid.

6.24 Hill-climbing search for registration features

6.25 A minimal set of registration features types

Testing surface models for equivalent geometry requires several registration feature points for an SVD transform, as described in Section 6.8. If a CAD models contain sufficient facets and corners to reveal three registration points for SVD matching, this is generally sufficient for determining the rotation matrix and translation component of an affine transform between these two models. If these registration features appear relatively close together, or all three points are nearly co-linear, the derived rotation matrix may be comparatively inaccurate. If only two registration points are obtained from a search, this is only adequate for rotationally symmetric models where these two points lie on the axis of rotation. A single point only allows matching spheres by their centroid. Simple rotationally symmetric shapes such as toruses and cylinders do not have local maxima that yield registration feature points using the algorithm described in Section 6.20, Registration feature search strategy.

In the case of a cylinder, a search may readily find one of the two edges, but may not converge at a maximum point as the cylinder edge forms a ridge of constant displacement from the centroid. One solution in the case of a cylinder is to search for a local minima rather than a local maxima, allowing searches to converge in the centre of the discs at each end of the cylinder. This approach is easy to implement, but is not robust in practice. The ends of tall, narrow cylinders have a low probability of a search being seeded nearby, and a high probability of any local minimum search converging at a band around the centroid. The search for a local minimum will simply converge anywhere on the inner surface of a toroid. Local minima registration features allow detection of certain unusual classes of shapes, such as spheres with dimples, but they do not create points with the same robust precision on a flat surface as a local maximum feature search will return at a corner.

If the search termination condition is expanded to include ridges of constant displacement from the centroid, this allows detection of rotationally symmetric ridges that have an axis of rotation that passes through the centroid. This condition may be detected if two of the search rosette points have a similar centroid displacement as the search pattern central point. The ability to detect a circular edge does not contribute a singular point suited to the closed form SVD method used for matching registration features, instead searches may terminate at any point on a ridge radius.

If a ridge is identified at three points it becomes possible to calculate the ridge centre point (`pointsToCircleRadiusCentre`), see Figure 29. The additional complexity of this method lies in determining three points that accurately intersect the ridge and are broadly spaced in order to return a centre point with reasonable accuracy (`rotSymTest5`). One subroutine uses an iterative search method similar to the rosette search to find points on ridge maxima that flank a discovered ridge point (`getRadialEdges`), a second subroutine refines the accuracy of the point centred on the discovered ridge (`refineRadialEdgesMidpoint`).

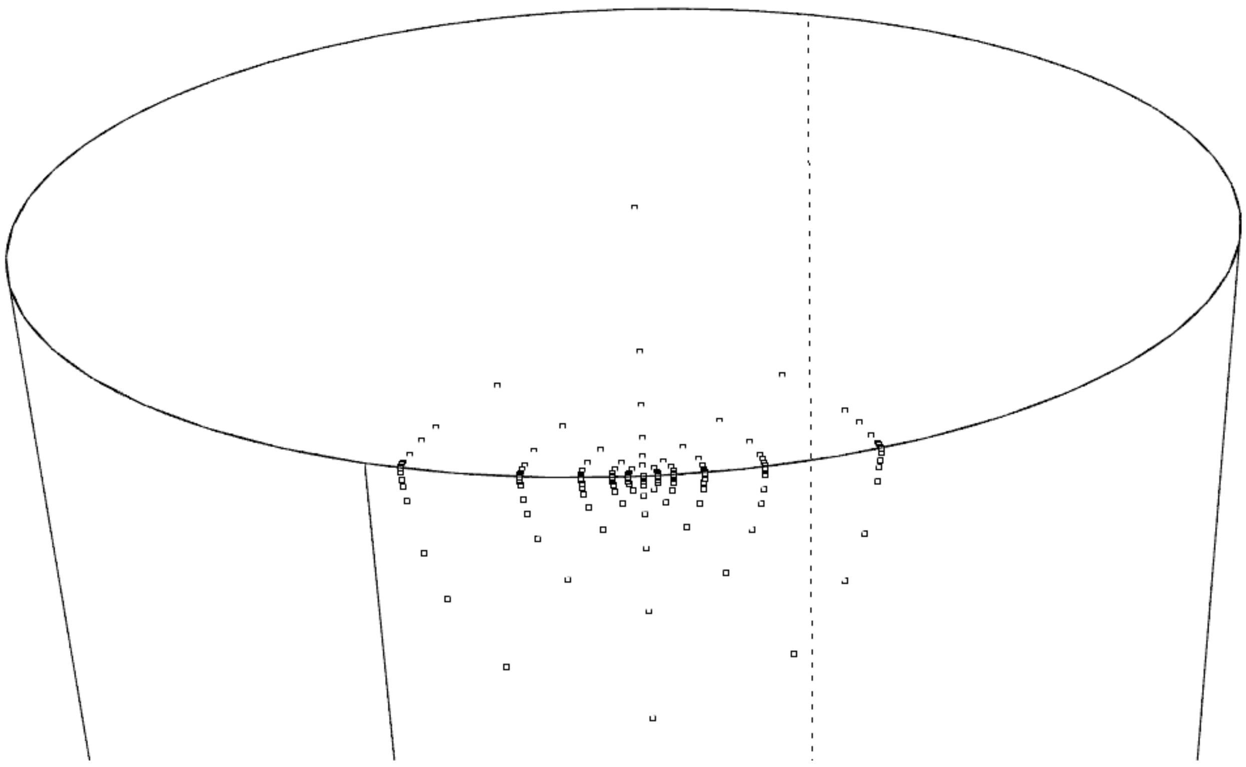


Figure 29: detail of point search to detect ridge edges and determine cylinder centre-point.

6.25 A minimal set of registration features types

This method may also be used to search for curved grooves using a minimum point displacement criterion, allowing the centre of cylinders or the inner surface of toruses to be represented by a single central point.

The final class of registration feature that may be detected using a nearest neighbour search with a rosette of points is that of a spherical surface. If the surrounding points all share the same centroid displacement values as the centre point, across a range of scales then the surface is spherical with the sphere centre sited at the centroid. In summary, five types of registration feature are specified as shown in Table 2.

Singular points	Local maximum points with respect to model geometry centroid
	Local minimum points with respect to model geometry centroid
Circle centre points	Ridge centre-points that lie on an axis passing through the model geometry centroid
	Groove centre-points that lie on an axis passing through the model geometry centroid
Sphere centre points	Spherical surfaces with centres co-incidental with model centroid

Table 2: classification of feature registration point types.

6.26 A minimum set of feature registration classes for all geometric shapes

Five types of registration feature are specified; the associated question is whether these five registration feature varieties allow representation of all possible CAD models. CAD geometries are very frequently assemblages of simple geometrical features, or of projections of two dimensional designs into a three-dimensional space. There are also CAD surfaces that are created from patches defined by control points, such as turbine blades or the sculpted surfaces of modern consumer products, as described in Chapter 8.1, An incomplete match of registration features. Each of these geometries may be subdivided into more simple regions. A feature search method that finds local maxima of regions is required to do two things.

- Firstly to find sufficient points to allow a rotation matrix to be generated between similar models undergoing matching.
- Secondly to create a sufficiently distinctive signature to allow comparison with signatures from other models.

Note that different models may produce identical registration feature points, but the final step of matching random points mapped from the surface of a target model to that of a source model is liable to detect these instances (Section 6.11.4). For example, an algorithm that only uses local maximum points is unable to distinguish between a frustum, sphere or a torus, see Table 3.

	Local Maximum	Local Minimum	Ridge Centre	Groove Centre	Sphere
Bowling ball		3			1
Cone	1		1		
Cylinder			2	1	
Torus			1	1	
Elongated torus	2	2			
Tetrahedron	4				
Tube			2	2	
Helical tube	2	2			
Elliptic Cylinder	4				

Table 3: examples of feature registration signatures.

Certain registration feature types are computationally expensive, such as detecting the centre-points of ridges or grooves around a central axis. If a model has a sufficient number of local maxima points to perform transforms, the search is completed. For example, where there are inadequate distinct points for a transform as in the case of circular edges perpendicular to axes passing through the centroid in a lenticular shape, it is necessary to combine the search for local maximum points with a search for local minimum points to find sufficient unique registration features.

6.27 Centroid sensitivity

All registration feature points are defined from the geometric centroid of the model, therefore the accuracy of registration features are dependent on the accuracy of the model centroid.

The position of the centroid is refined from the increasing number of surface points identified during the course of registration feature searches. While the centroid is initially located, the mean value of the Cartesian coordinates of all points defines the centroid position (see Section 6.21, Preliminary centroid identification). The method to determine model location radiates a spherical array of vectors from the origin to intersect the model surface. This method performs well for initial centroid estimation. Subsequent searches for registration features generate numerous surface intersections, however the distribution of these points on the geometry surface is not even. Points are concentrated in the vicinity of registration features and at a higher density on surfaces closer to the estimated centroid. Consider that a centroid measure is an approximation of the barycentre of a model with a uniform mass associated with each surface region. An uneven surface distribution of points creates inaccuracy in this centroid approximation. Gope and Kehtarnavaz prove that a centroid derived from a convex point set is affine-invariant, this finding can be extended to encompass any consistent set of point based registration features (Gope & Kehtarnavaz, 2007).

In the implementation used, the remedy is to change to a centroid defined by mean values of model surface extrema once the model has undergone searches for registration features. The extreme surface features correspond to corners on the convex hull of the geometry model, or the local maximum points located. While this method is sensitive to maximum point features that are not identified, tests indicate that a median calculation provide a reliable centroid location.

6.28 An incomplete match of registration features

A centroid determined from the mean value of registration feature points is sensitive to cases where there is an incomplete sets of registration features, where a complex geometrical model may contain feature regions with a low probability of detection. These missing points result in an altered mean centroid position. While two sets of displacement histograms formed from representations missing exact same registration feature points will be correctly recognised, in cases where these sets miss different points the method may return a false negative geometrical similarity score. The median value of the discovered feature registration points returns more robust centroid measure as corners at the model extrema have a relatively high probability of discovery (`medianCentroidCorrection`). A robust centroid value allows partial histograms to be matched, where centroid displacement values that do not have counterparts may be pruned from the source and target sets (`listIntersection`).

6.29 Geometric transformation and matching algorithm overview

Figure 30 illustrates the sequence of processes used to accomplish geometric matching. The first operation detects the location of a model via the method described in Section 6.27, Centroid sensitivity. If the model is not detected during the initial projection of rays from the model space origin, the number of projected rays is increased. This process repeats until a limiting resolution is reached or surface model is located. This located model yields a centroid as an average value of intersections.

The second operation uses the nearest neighbour hill search technique based on the method described in Section 6.24, Hill-climbing search for registration features to determine a set of feature points. This search may yield one of several feature point types based on the model geometry which provides both a distinct model signature and a set of registration points suited for determining any rotational difference between the source and target model. This set of feature types is fully described in Section 6.25, A minimal set of registration features types and Section 6.26, A minimum set of feature registration classes for all geometric shapes. The search for model feature points cannot be exhaustive if the total number of potential feature point regions is unknown, therefore this search is repeated until the same feature point regions have been detected a statistically significant number of times. This requirement is also covered in Chapter 7.5, Complex model matching.

The third stage in Figure 30 uses the feature point type, centroid-displacements and number to create an affine-invariant model signature that might be rapidly compared against the signatures of all other models that have been processed in a similar fashion. Once a likely match is found, the rotation, scaling and translation variances between the source model and target model undergoing comparison is calculated. This rotation is determined using the SVD method described in Section 6.8, Global Registration via Singular Variable Decomposition. The individual steps to the comparison algorithm are described in more detail in the first two steps of Section 6.11, A progressive search refinement strategy.

The final verification stage shown in Figure 30 involves the latter two of the sequential method described in Section 6.11.3, and Section 6.11.4. The statistical likelihood of geometric model similarity is generated from the transform of model feature points and random model surface points using the calculated rotation, scaling and translation. The deviation from the predicted location of feature points, and the distance from the model surface provide a single numeric attribute of similarity.

This chapter has covered the implementation of a geometric model comparison method. A boundary surface intersection method returns a numerical estimate of geometry similarity that is not subject to model position and orientation. A characteristic set of distinctive surface features relative to a model centroid provides a simplified model representation. These point features allow several progressive stages of geometry matching discrimination, namely a match of point feature types, a match of feature displacements, a transformed mapping of features from one model to another and finally the transformation of random points from one model surface to another. If point features may be paired with their counterpart between models, shape registration may be carried out using a computationally efficient closed form SVD calculation. A minimal set of potential

6.29 Geometric transformation and matching algorithm overview

pairing configurations is determined from a match of point-centroid displacements between models, ordered in a helical fashion along a select model axis. The robustness of this method is based on reliable detection of the majority of model characteristic point features. This is in turn governed by the topological complexity of a boundary surface model. The accuracy and efficiency of this proposed method is tested on a benchmark CAD shape dataset in Chapter 7.

6.29 Geometric transformation and matching algorithm overview

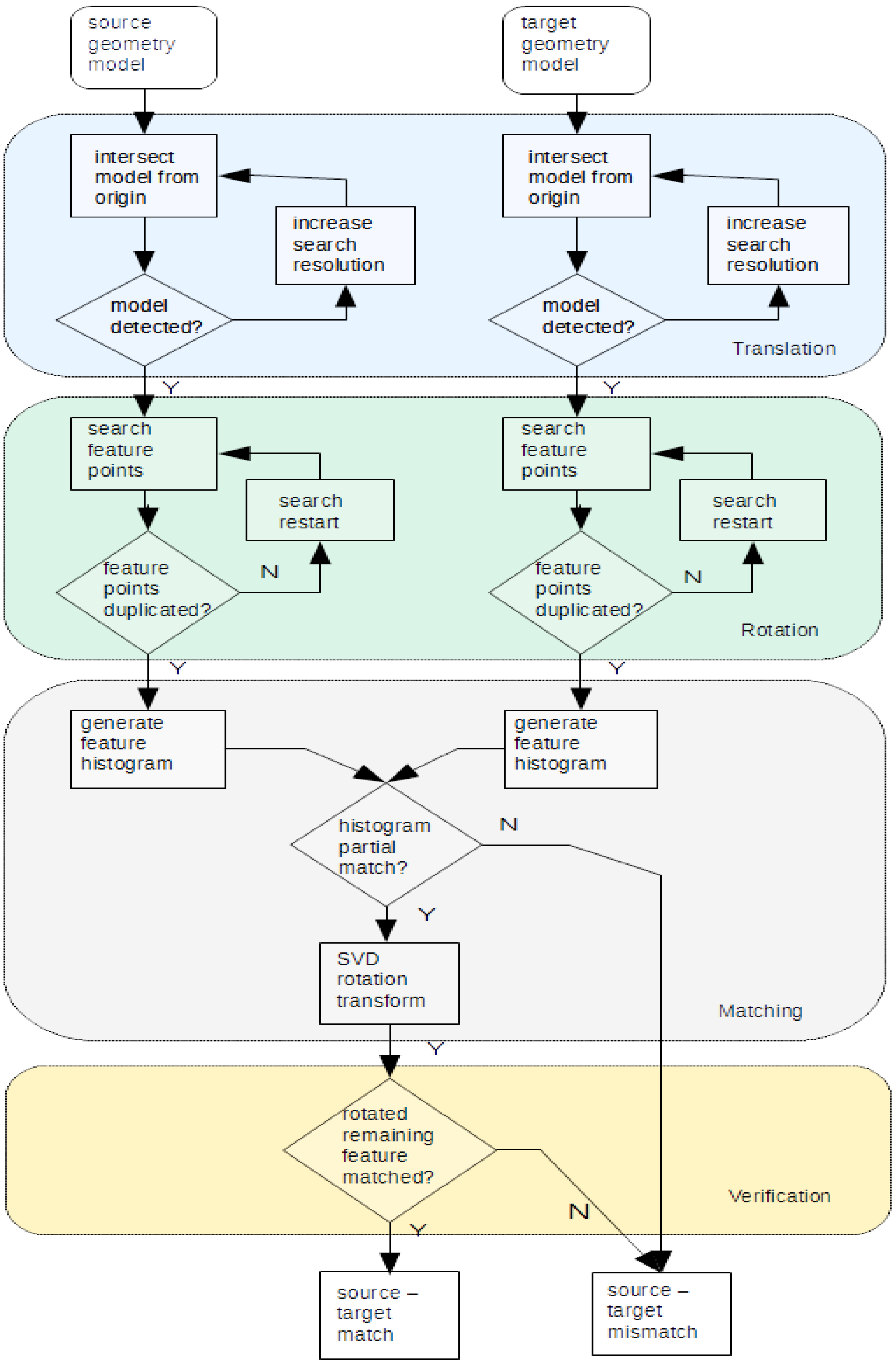


Figure 30: overview of transformation and matching algorithm.

6.29 Geometric transformation and matching algorithm overview

6.29 Geometric transformation and matching algorithm overview

7 Geometry matching method tests and results

The geometry matching method described in Chapter 6 is tested on a benchmark CAD shape dataset. Two tests are carried out. The first test determines the degree of similarity between a random shape selected from a benchmark library and a rotated, translated and scaled version of the same shape, a second comparison is made against a non-matched shape. The second test returns the values of a randomly chosen shape tested against all model representations generated in the first test. It is found that the method is sensitive to scale and identification of features with a proportionally large radius of curvature. Limitations of simple feature type comparison are presented, as is the method of matching relatively complex geometry.

7.1 Drexel CAD shape benchmark library

Testing the search and match algorithm described in Chapter 6 may be carried out in several ways, the anticipated implementation would run two CAD model programs independently from a third independently running comparison program. This configuration allows projection of vectors and retrieval of point intersection with geometry surfaces using minimal CAD program interfacing. These interfaces permit source geometry model testing in a source CAD instance with a target geometry model in a target CAD instance to determine model similarity in an automated process. It is reasonably straightforward to create interfaces in the Python, or C++ languages to communicate via the Component Object Model interface to most CAD programs running on Microsoft Windows, however this is not required for testing the effectiveness of the matching algorithm. Source and target models may be created within the same model space of a single CAD program and tested for similarity.

To obtain sets of models on which to test algorithm efficiency, it is possible to randomly generate models or to randomly select existing models from a library. Commercial libraries and repositories contain three-dimensional models, many of them complex models from parts catalogues. Others such as the Princeton Shape Benchmark are not stored as CAD files (Shilane *et al*, 2004). The US National Institute of Standards and

Technology created a benchmark repository of three dimensional CAD models in a variety of neutral formats ranging from simple to relatively complex representations, which was subsequently hosted at Drexel University (Regli & Gaines, 1997; CAD Models Dataset, 2004). This repository is subdivided into several categories. The Primitive Models Classification set contains 300 variations of cubes, cylinders, spheres and toruses. The testing was arranged so that,

- A random source model is selected from this set and placed within the CAD model space at a fixed orientation and position.
- The same model is placed within the CAD model space again, but this time at a randomly generated orientation and at a random position and scaled to a random factor.
- A third model that is guaranteed different from the source model is added to the CAD model space, again placed at a random position, orientation and scale.

This arrangement allowed testing for the percentage similarity against a known match and also against that of a known mismatch. This format gives a figure for model similarity, along with a figure for model rejection. Tests were conducted on a Dell Latitude E6540 laptop running 64bit Windows 7. The CAD program used to host the model comparisons was McNeel Rhino 5, accessed via the COM Automation Rhinoscript API. The Python comtypes library allows external Python language scripts to access this API. The programs used for this purpose are written in the Python scripting language and are not optimised for speed, but demonstrate proof of concept.

7.2 Test configuration for single and multiple model matches

It was found that the method used to determine shape matches had a comparably high success rate when compared against other shape matching methods using the same library of shape primitives (Bespalov *et al*, 2005). Precision-recall plots give an indication of the number of correct shape matches which rank at the highest probability, there are defined in more detail in Chapter 5.5, 62. Again, as with the semantic matching tests in Chapter 5.5 the entire set of shapes is tested and ranked by probability. The recall metric determines the percentage of correct matches in the retrieved set, but in this case it is more accurate to describe the returned set as the selection of models which surpass a threshold of similarity. The returned distance measure is based on the RMSE value derived from the accumulated error values of points transformed between the source and target models. In the event that there are inadequate feature registration points to derive a closed form solution to the relative orientation of the source and target models, the distance value is set at infinity,

$$err = \begin{cases} err_{RSME}, & \text{if } N \geq N_{min} \\ \infty, & \text{if } N < N_{min} \end{cases}$$

where N is the number of identified registration features on both source and target models, and N_{min} is the minimum number of features required for SVD determination of affine transformation. For the purposes of a probability metric this value is transformed via,

$$P(err) = \frac{1}{1 + err}$$

where $P(err)$ is the probability of a match in the range [0, 1]. Results are tabled in Appendix C. Comparison of the matching metric against existing shape classifier methods is limited in utility, the method is intended to verify model similarity rather than identify similarity, however the method performed comparably well against other object matching methods tested using the same CAD model library (Bespalov *et al*, 2005). This apparent success is explained by the use of model signature matching that proved to be sufficient for the set of models used.

Each model pair is tested using progressively refined discrimination as follows,

- sets of registration feature types,

- sets of relative registration feature centroid displacements,
- sets of registration feature coordinates
- sampled surface points transformed between models

Consequently most early tests will reject considerably different shapes and the test sequence may be halted. Model signatures that have a widely differing number of points and displacements are winnowed out using a correlation distance metric. The models that use rotation and sampled point tests only constitute a small percentage of overall tests. A full description of this method may be found in Chapter 6.11, A progressive search refinement strategy.

In 57 tests between pairs of randomly sampled shapes at random orientation, scale and position, the method correctly identified each class of shape, plus each shape variant.

7.3 Observations on algorithm performance

Comparing shape signatures is comparatively rapid with individual tests generally taking under a tenth of a second to complete, but generating model representations is several orders of magnitude slower. Simple shapes such as cubes would take 20 seconds, while shapes with radiused curves take up to ten times longer. The search for model feature points involves a large exchange of coordinate data with the CAD program API undergoing testing using the Microsoft Component Object Model. In cases where there are large numbers of potential registration features at interior and exterior model locations that have a low probability of discovery, the search process takes several minutes to terminate. While the primitive shapes allow rapid discovery of registration feature points, complex forms commonly encountered in CAD design are not so tractable.

It was found that the algorithm is relatively sensitive to scale. The matching routine is relatively independent of scale, but the methods used to identify registration features are susceptible to numerical noise. The routines to determine feature points and ridges require comparison of point-centroid displacements, these comparative measurements are sensitive to the absolute size of the models tested. One example is finding the outermost ridge of a large torus by comparing displacements of the surface points from the model centroid, where the variation in surface curvature produces relatively small differences in displacement compared to the absolute displacement value. Accordingly scaling of the target model was limited in range to between 0.5 and 2.5 times the scale of the source model. This sensitivity to numerical noise becomes a greater problem when relying on the precision of different CAD programs. While normalising CAD model scale via program API is a relatively trivial task, this is not envisaged in the minimal interface afforded by the Hoffmann query method (see Chapter 3.12).

The tests described created a database of 303 variations of the primitive shape library in random orientations, positions and scales. Each of these models were subsequently tested against all other models within this database to evaluate match accuracy. Two instances were found of models that did not have the same class of shape as closest match, this was subsequently found to be caused by malformed feature registration points registering an equivalent infinite distance in similarity between these models and all others. The Precision-Recall plots in Figure 31 - Figure 34 show that shape category matching, based on the configuration of registration features used for the set of primitive models adequately describes all shape classes. The plots are more interesting where the method discriminates between models distorted along various axes, showing a decrease in accuracy. The worst shape discrimination performance occurs between models that have radiused edges. A radiused edge has finer differences between point-centroid displacement values, leading to a loss of corner or edge accuracy. Where the radius has the same diameter as the point-centroid displacement, the surface is indistinguishable from that of a sphere. These three aspects of model similarity are plotted on precision-recall plots for several class of primitive shape tested, (Figure 31, Figure 32, Figure 33, Figure 34). As the tetrahedron and dodecahedron shapes only exist in a single variants that was added to the Drexel benchmark CAD repository data, comparative tests only provided shape discrimination, which is displayed and commented in Figure 35. Note that the sphere and torus shapes do not have blended or radiused versions, all model shape distortions are uniformly applied along the chosen axes.

7.3 Observations on algorithm performance

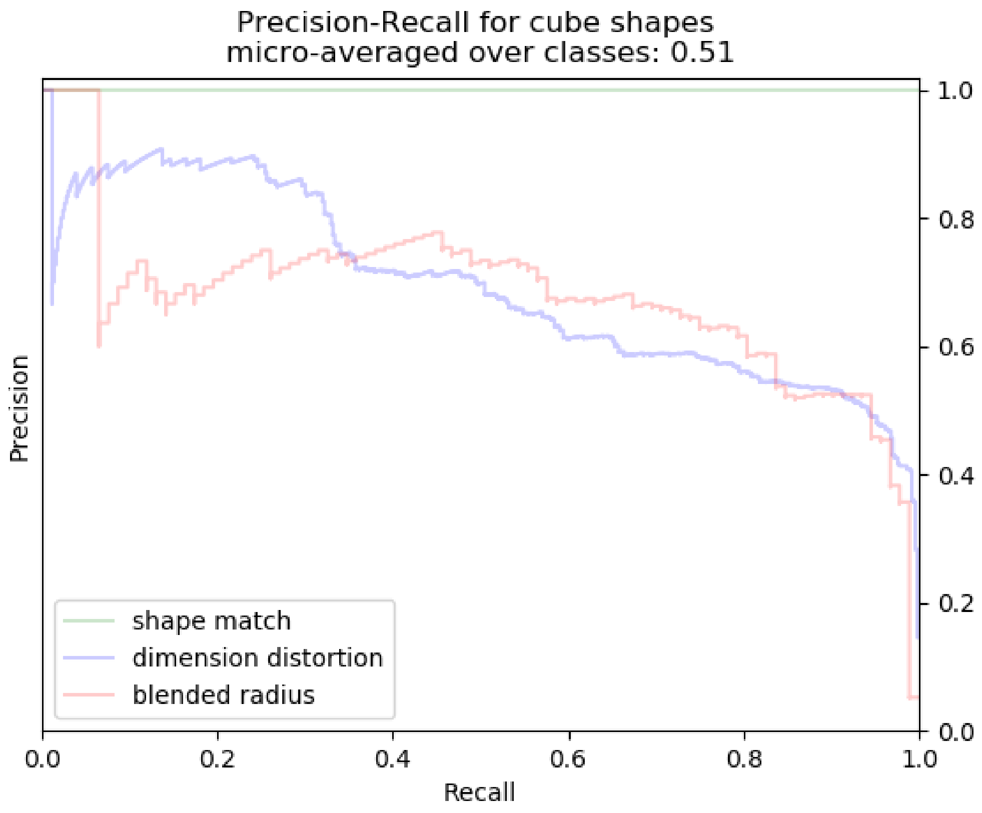


Figure 31: Precision-recall for Cube shape class similarity determined via transformed point sampling method.

7.3 Observations on algorithm performance

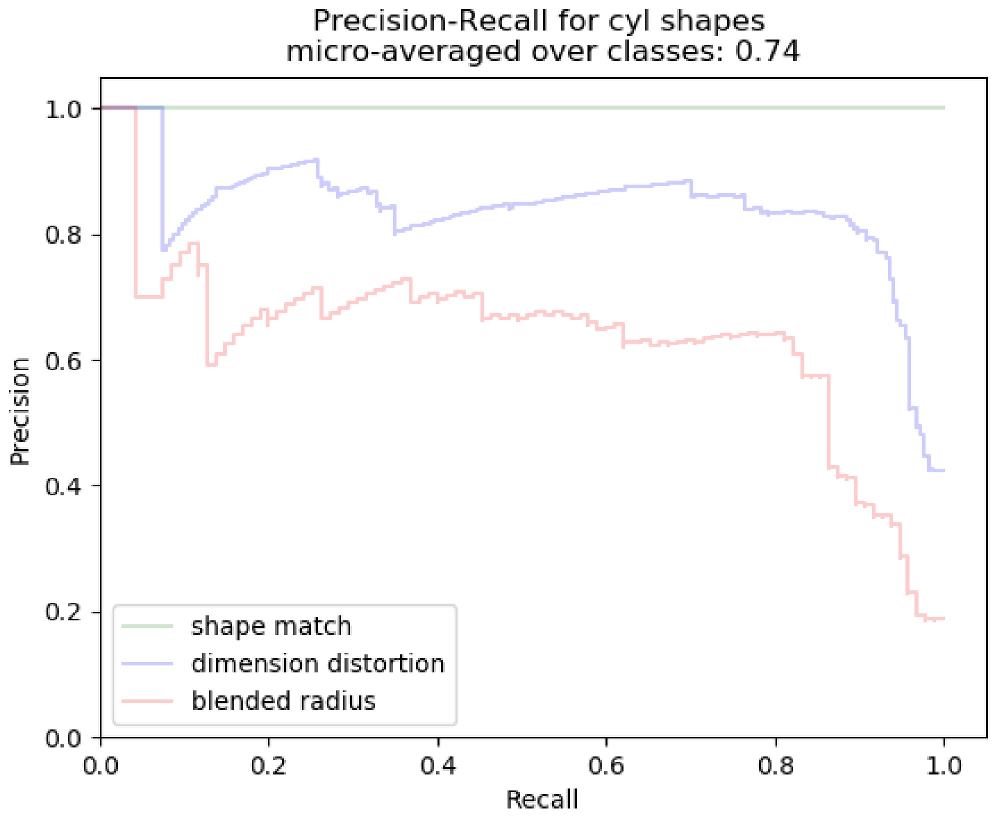


Figure 32: Precision-recall for Cylinder shape class similarity determined via transformed point sampling method.

7.3 Observations on algorithm performance

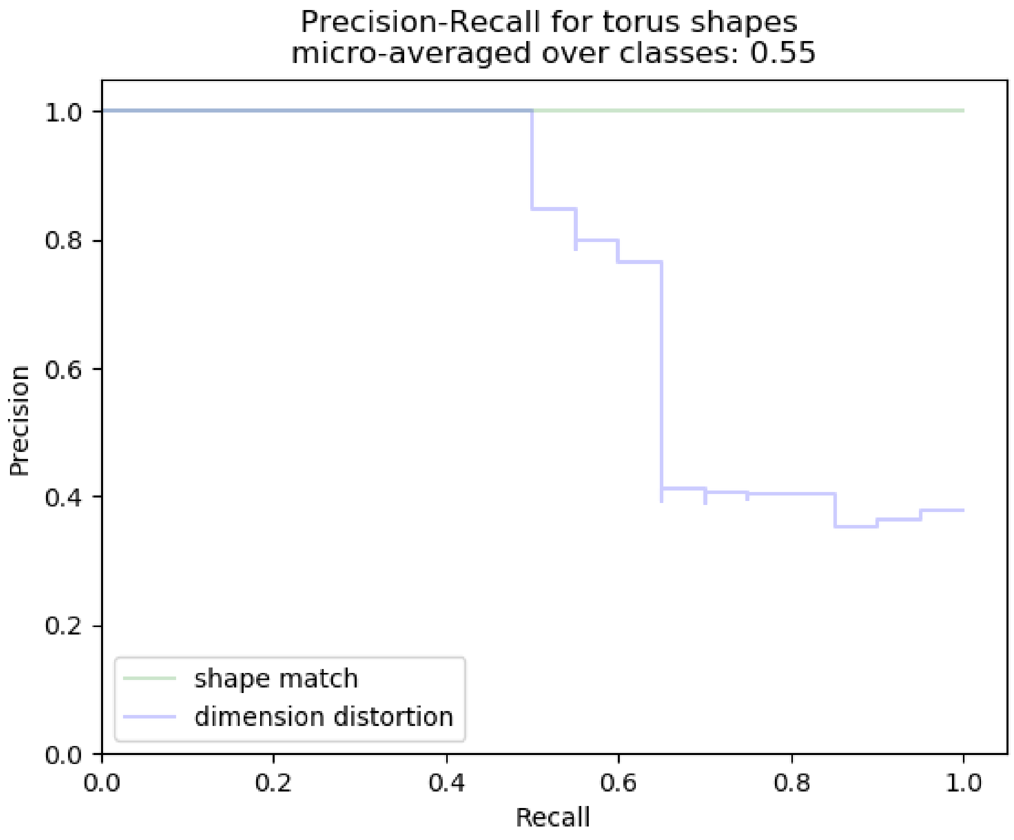


Figure 33: Precision-recall for Torus shape class similarity determined via transformed point sampling method.

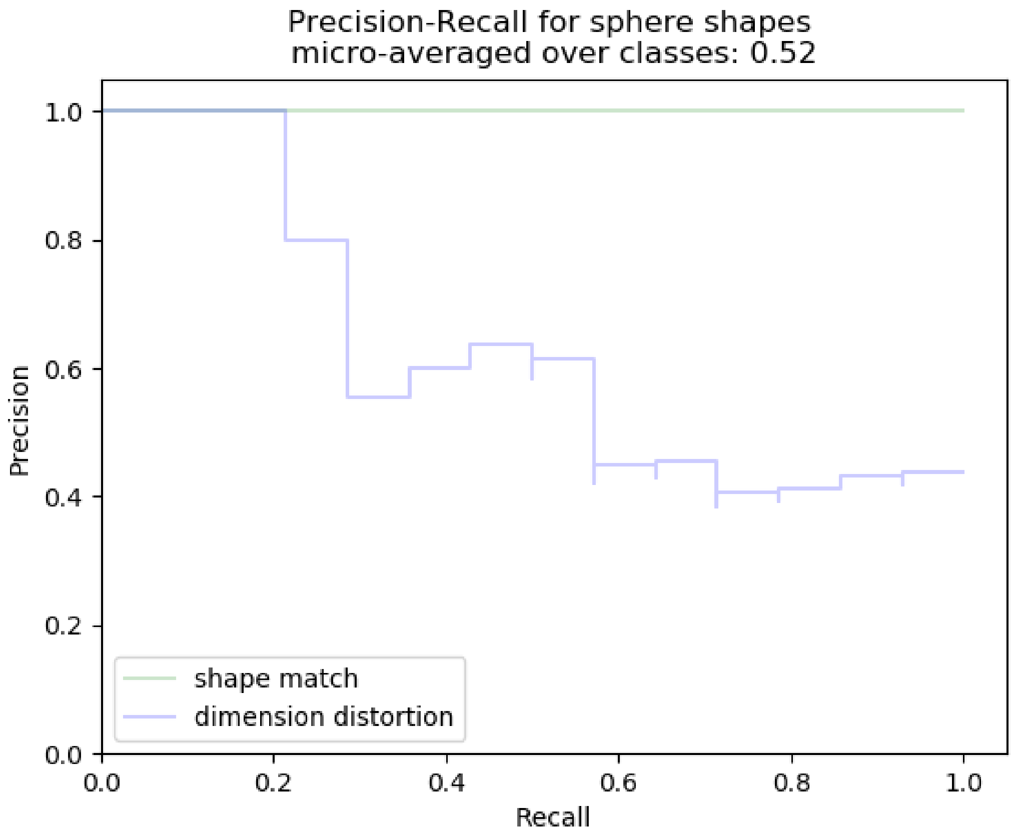


Figure 34: Precision-recall for Sphere shape class similarity determined via transformed point sampling method.

7.4 Instances of registration feature mismatch

A further graph, Figure 35, extracts information on perfectly matched and slightly mismatched shapes that are assigned a probability score, these correspond to true positive and a false positive values. The split violin plot reveals inaccuracy within the sphere and torus categories. This appears to be related to elongated versions of spheres and torus that are both distinguished by two point maxima at the apex of each elongated axis, and two point minima which are at the innermost points of torus rings, but on the outside of the shortest axis of the elongated sphere. As a result, both models pass the minimum test requirements and are evaluated for surface point deviations between models.
A similar mismatch exists between tetrahedrons and elliptic cylinders, both are

distinguished by four feature maxima. These shapes are distinguished by the accumulation of error in the predicted position of points mapped between the two surfaces as described in Chapter 6.11.4, Checking via random transformed surface points.

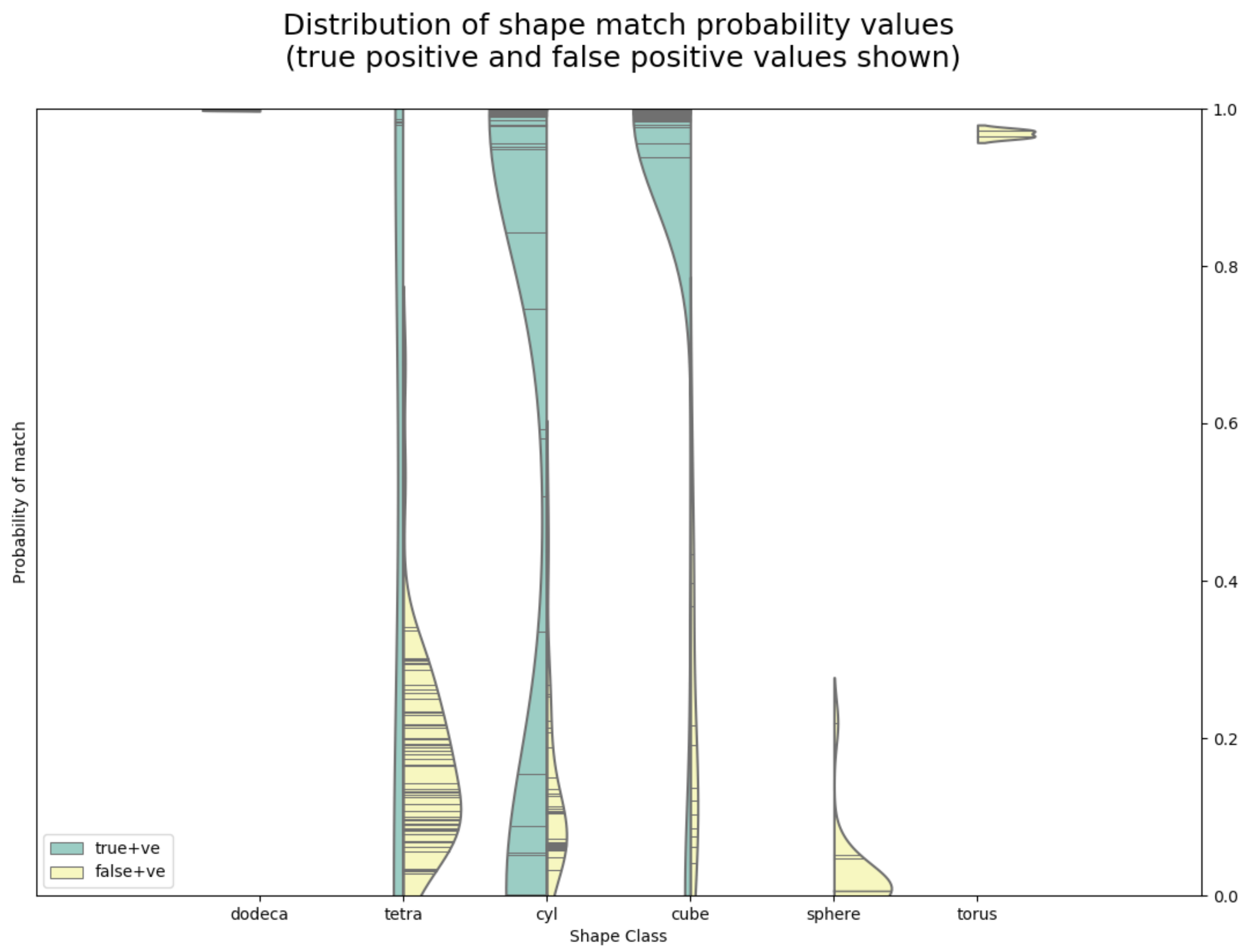


Figure 35: True positive and false positive probability distributions over all shape classes determined using transformed point sampling method.

7.4 Instances of registration feature mismatch

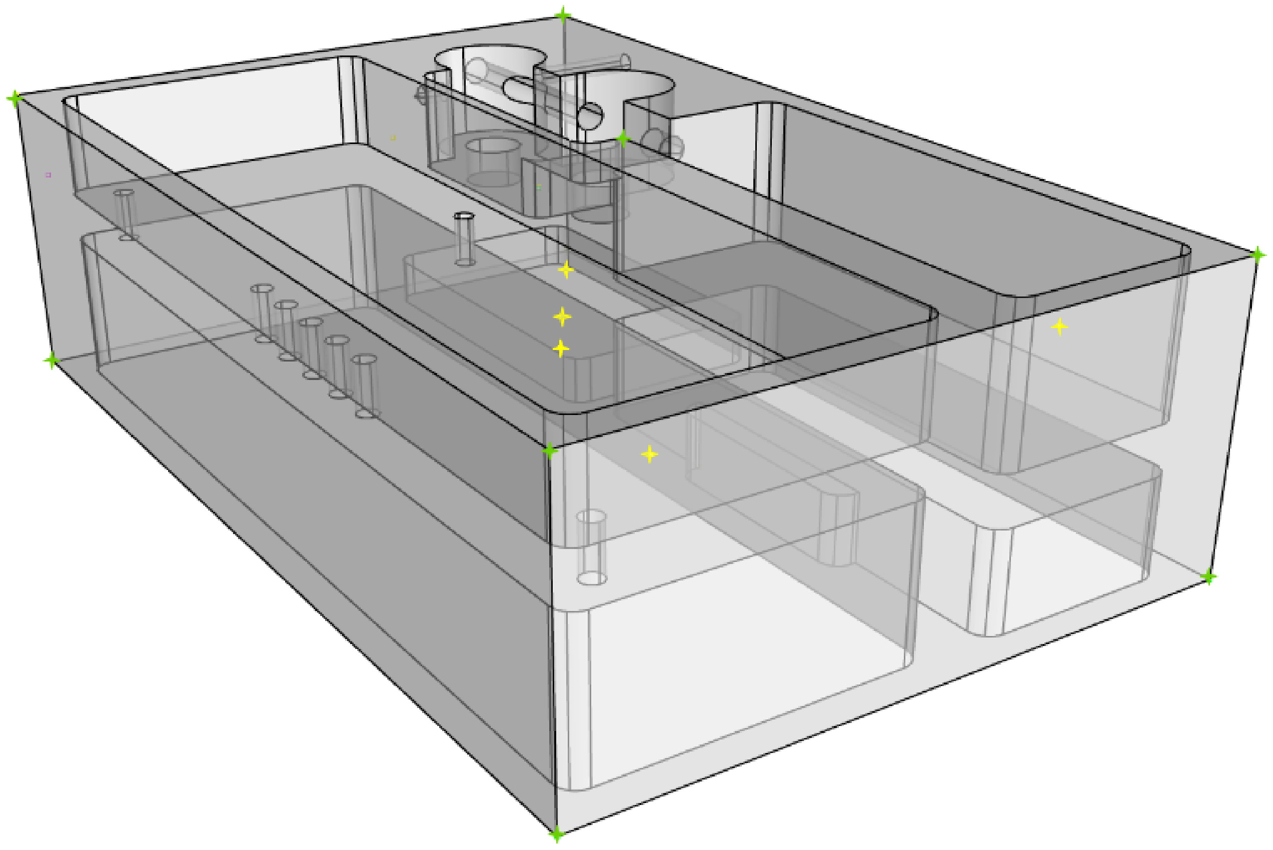


Figure 36: Transformed and matched complex asymmetrical bracket model, green points represent points used for SVD registration, yellow points are mapped between models to verify model similarity.

7.5 Complex model matching

Registration feature identification is effective for simple shape geometry where it is probable that all surface regions that can serve as registration features may be found in an exhaustive fashion. Where there are a large number of potential registration feature sites on a complex model, with widely differing probabilities of discovery, exhaustive searching becomes impractical. In this event, matching models relies on finding roughly the same number of features between identical models. The drawback with an incomplete set of registration feature points is the subsequent inaccuracy of a model centroid estimated to be at the position given by the mean values of all discovered registration features. If this value is substituted for the position given by the median of discovered model registration features, it allows models that do not share exact complements of features to be matched. A subsequent problem encountered with geometries with incomplete sets of registration features is a potential ambiguity of feature point ordering. As detailed in Chapter 6.17, Helical point sequencing, the set of features must be matched pairwise for successful SVD transformation. If the ordering of features is reliant on matching sets of centroid displacements, this may fail where readily accessible features may allow several model orientations, such as the corners of a block, but where undetected features are asymmetrical. In this instance the solution is to subsequently test all possible model orientations using mapped points taken at random from the source model and transformed to the target. Figure 36 shows a complex machined block which has been transformed in this fashion. The green points represent detected registration features on the model surface, the yellow points are random sampled test points transformed between the two models.

7.6 Points and vector detection

The method described to intersect 3D CAD model surfaces is not effective on geometry elements with two dimensional representation such as vectors, rays, lines, or like points, having zero dimension. However the same principle of intersection may be used to return a Cartesian point. In the case of a line described in three-dimensional space, a projected plane or surface may be substituted for a projected ray. The line may then be identified via an intersection with the projected plane. Further planes or surfaces may be then generated to identify registration features on a 2D object, such as line ends or centres of curvature. In the case of points, a 3D volume will register intersections.

7.7 Future directions

The method of projecting points from the estimated model centroid is effective for relatively compact and simple models. There are however surface geometries that are relatively intractable. Any geometry with surfaces that are either at a constant radius from the centroid, such as regions of a sphere, or are parallel to any vector emanating from a model centroid produce ambiguous feature descriptors. Projecting rays to intersect a model surface from a fixed centroid has diminishing search effectiveness on surface regions relatively distant from the model centroid. A more efficient method might use several judicious locations from which to direct searches rather than a single centroid point.

The search for registration feature points takes a scrupulous approach that rules out errors associated with model export to a neutral format. Yet the search for features is computationally expensive. Registration data may be extracted from neutral export files which can be subsequently verified within the CAD model space. Registration feature points may be extracted from these common file formats, see example taken from the Drexel benchmark CAD dataset in Table 4. Several of these formats such as the ISO10 323-21 STEP format, or the Autodesk DXF format explicitly describe many of the Cartesian coordinates that constitute the feature registration representations used, such as vertices or the centre points of arcs (Autodesk, 1997, 10 303-21:2016, 2016).

```
/* ISO 10303-21 file written by STEP Caselib, ProSTEP GmbH, Germany */
DATA;
#1=CARTESIAN_POINT('POINT1',(5.0E-01,5.0E-01,-5.0E-01));
#2=VERTEX_POINT('VERTEX1',#1);
```

Table 4: example of Cartesian point information taken from a Drexel CAD benchmark library STEP file.

Parsing these file formats to extract model geometry information is more efficient and potentially more accurate than registration feature search methods that use surface intersections. A model geometry comparison system may export geometry data from an internal CAD program representation to a neutral file format, which can be subsequently compared using the method described in Chapter 6.11, A progressive search refinement strategy. The drawback of this strategy is the possibility of incorrect translation of native model geometry to neutral file format. Where the model details are relatively simple non-parametric surface data, it is likely that STEP translators work as specified. In the case of more complex data there are documented instances of CAD programs that do not export STEP model data in compliance with the ISO standard (McKenzie-Veal *et al*, 2010; Ćuković *et al*, 2017).

The feature registration regions classifications used in this implementation are limited to spherical surfaces alongside point surfaces, ridge surfaces and their inverse, dimples and groove surfaces. These features are sufficient to identify the models tested, but there is scope to include saddle surfaces and co-linear surfaces.

A co-linear surface is an edge or surface that is parallel to the projected vector used to retrieve surface intersection, this surface will return ambiguous results unless a separate test identifies it.

8 Automated feature function mapping

This chapter distinguishes the existing methods of model generation in CAD programs. Explicit and implicit constraints are defined with respect to sequential modelling operations on parametric model descriptions. The influence of prior modelling decisions on succeeding geometry may be captured by a "dynamic" mapping of intermediate modelling stages. This approach does not rely on a stored function mapping, but tests several potential matches for geometric equivalence, demanding an automated or assisted method of function matching. Geometry matching techniques can identify functions that create similar surfaces, mapping function parameters may be defined as a stochastic search problem. A technique is described to conduct efficient parameter mapping using a genetic algorithm with a restricted parameter set.

8.1 Overview of Computer Aided Design modelling methods

The development of commercial three dimensional modelling software has been influenced by earlier two dimensional drafting programs. It is difficult to manipulate three dimensional designs using a 2D Graphics User Interface and CAD software uses a range of techniques to allow accurate model generation that is relatively intuitive. Some early modelers used an assemblage of geometric "primitives", such as cones, oblongs and toroids to generate complex shapes. These Constructive Solid Geometry use boolean operations of union, intersection and difference on primitives to sculpt surfaces (Deiz & Appllin, 1993). Systems that exclusively use this technique such as BRL-CAD are cumbersome to use for defining complex geometry and impossible to use for free-form geometric surfaces that are curved in several directions.

While modern CAD programs will incorporate boolean operations and a range of primitive geometric objects, a more common generative technique is to define two dimensional profiles on a 2D plane and then extend this profile into three dimensions. A simple circle may be extruded, or projected along an axis perpendicular to the profile plane to form a cylinder. This circle profile may follow a curved path to form a ring or a helix. A two dimensional plane may be described on the surface of an object to allow a hole or a pocket to be projected into the shape.

Many CAD model geometries are composed of planar surfaces or surfaces with curvature limited to one plane, there is a class of operations based on the interaction of adjacent surfaces. The edges of adjacent surfaces may be rounded or blended to form a *fillet* or a *chamfer*, or if these surface edges do not meet they may have an additional surface to patch the gap. Complex free-form surfaces are defined using a two dimensional surface equivalent of a spline. These surface patches are mathematically defined as Non-Uniform Rational B-splines (Piegler & Tiller, 1987), discrete Coons patches (Farin & Hansford, 1999), or equivalent. Surfaces may then be modified by moving "control points", similar to the control points that guide the path of a spline. They can also be draped over a series of 2D cross-sectional profiles analogous to an aircraft wing surface formed from the internal structure of aerofoil sections (Figure 37).

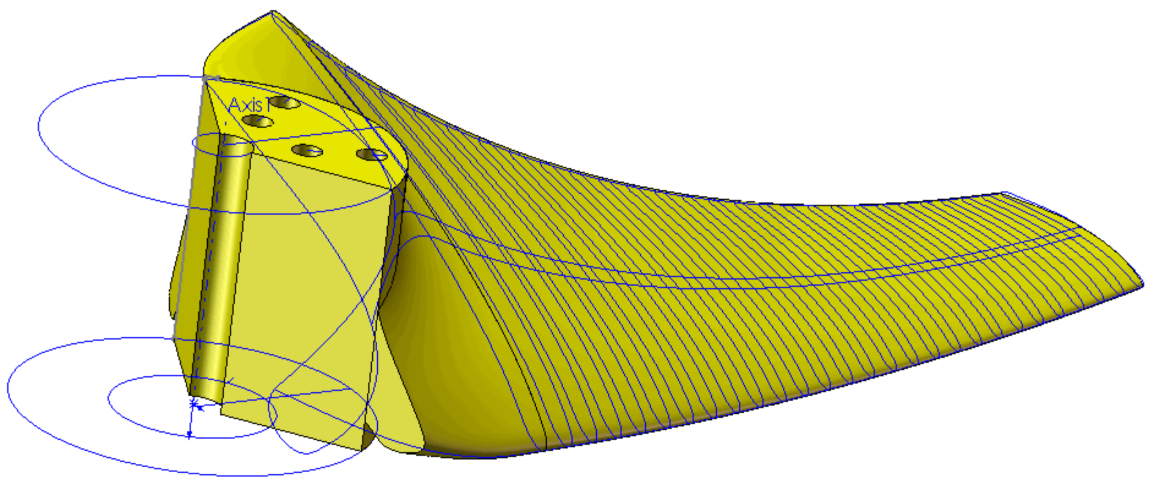


Figure 37: CAD model of hydrodynamic turbine blade showing combination of extruded surfaces, blended surface patches and surfaces patches draped across hydrofoil sections.

8.2 Explicit and implicit CAD model constraints

These four approaches, that of boolean operations on geometric primitives, of modifications made to profiles defined on drawing planes in 3D space, and of operations on model surfaces and free-form surface creation form the majority of all 3D model creation techniques used in CAD programs. Each of these four methods of model geometry manipulation are dependent on interaction with prior model geometry. The CSG process may create a stand-alone geometric primitive, but a useful CSG model requires an assembly of these geometric primitives. Again the extrusion of a 2D profile along a path creates a limited range of geometries, these extrusion features generally occur from a drawing plane defined on the surface of an existent model geometry. Even complex free-form geometries that are generated from numerical calculation such as turbine blade profiles are invariably wedded to a composite geometry created by conventional means, such as the geometry of the turbine blade root. This relationship of constraints and parameters that influence the outcome of a feature geometry operation has been described as *implicit constraints*, in contrast with the explicit constraints that are detailed in the feature function parameters.

Implicit constraints are specified in the sequence of prior feature operations that compose a CAD model geometry, therefore access to these constraints may be required in subsequent modification operations. Parametric kernels, the geometry engines of CAD programs, allow designers to revisit the parameters of previous feature operations in the model generation sequence and regenerate the model geometry without having to manually rebuild subsequent feature operations. For the purpose of creating distinctions between implicit and explicit constraints, geometry constraints may be defined as a subset of feature parameters. These deterministic constraints on feature behaviour attach importance to the order of feature operation sequence, as well as the relationship between each model feature. Software vendors have introduced graphical tree representations to represent the sequence of feature operations and to represent circumstances where multiple features may branch out from a single feature node, such as several holes defined on a single plate.

8.3 Sequential model generation

A complex model is an outcome of a deterministic sequence of user selections. A parametric CAD program will allow a modification to a modelling decision within the history tree to propagate changes made to the final model, in similar fashion to a time-traveller altering the present by making small changes to the past. These selections have been defined as implicit geometric constraints, they arise from a choice taken from a set of possible modelling options. This chain of modelling decisions is distinct from explicit specification of function parameters and geometric constraints used to specify a parametric CAD feature. To give an example of an explicit function parameter, a torus feature might require a cross-sectional radius, a central point and a perimeter radius to be defined. These explicit parameters are set by the user, while implicit parameters arises from the model context. As an example, a fillet is a rounded surface applied along a model edge. This fillet feature is dependent on the prior geometry of the edge, which may in turn be defined by the surfaces adjacent to the edge.

In a parametric CAD program, any modification to the adjacent surfaces will update the geometry of the dependent feature. The feature may contain inherent geometric constraints, in the example of the fillet feature there is an implicit geometric constraint that defines one axis of the fillet feature collinear to the referenced model edge. Another inherent geometric constraint of the fillet feature defines the fillet edges to be co-tangent with the surface edges. Inherent geometric constraints are axiomatic to the definition of the parametric feature, yet these implicit and explicit feature constraints are distinct from explicit constraints that may be defined between CAD features. Parametric CAD programs make provision for explicit geometric relationships such as co-linearity and concentricity to be specified between independent features. An assembly or mechanism composed of several independent models commonly requires that the individual parts are related to one another by geometric constraints. Implicit constraints create a potentially complex interaction of features. While the method is a powerful and intuitive modelling paradigm, the behaviour of features in response to pre-existing feature configurations becomes more difficult to fully define. A conceptual hierarchy of the relationship between explicit and implicit constraints with regard to models of increasing complexity is shown in Table 5.

	Explicit constraints	Implicit constraints
Features	Feature parameters (both optional and non-optional) Defined geometric constraints	Referenced precursor features
Parts (composed of features inheriting from a single feature)	Defined geometric constraints Set properties (colour, density, etc)	Interaction of composite features
Assemblies (composed of parts inheriting from multiple features)	Defined geometric constraints	Not defined

Table 5: conceptual hierarchy of implicit and explicit feature constraints.

The elementary method of feature mapping described in Chapter 6 tests the effect of a feature operation on a purposefully basic model geometry. This process is duplicated within the two CAD programs undergoing geometric comparison of feature manifestations. If the geometric outcome of a feature operation is modified by a sequence of functions characterised by a complex model, this extends the number of parameters that must be tested to determine the full mapping of feature equivalence. It is impractical to test the response of feature operations on the unbounded set of complex geometric models. This limitation is addressed by a revision of the translation process.

Static and dynamic geometry matching

Conventional translation methods seek to establish a mapping between set elements, ideally returning a simple one-to-one match between elements of the respective sets. In the case of feature matching of CAD programs, a match is identified from the class-level feature libraries of the CAD programs undergoing comparison. The Macro Parametric Approach described in Chapter 3.8 is an example of this method. Once all features of a library have been matched with their equivalent counterpart, this static set of mappings allow a model generated from a sequence of CAD features to be recreated in another CAD program using directly mapped substitutes for these features.

The drawback with this approach is that it presupposes an exact one-to-one equivalence between features of heterogeneous CAD programs. Commercial CAD programs are found to only have a very small proportion of functions that are direct equivalents to functions in other CAD programs. While the conceptual framework of geometry features may appear virtually identical, the scope of each feature operation is bounded by different parameters. This leads to features having a different granularity, where the geometric outcome of a single feature operation might only be replicated by several sequential feature operations in a different CAD program. Or features might have a one-to-many correspondence, where a single feature operation in one program may be geometrically replicated by several distinct feature operations in another. These complications are referenced in greater detail in Chapter 3.6.5, Inconsistent constraint combinations within generic features.

There are two approaches described that address the limitations of a relatively small set of static matches used for translation. Tessier and Wang propose a method of dynamic matching (Tessier & Wang, 2013). Rather than using a static match for every feature within a model to construct an identical model in a second CAD program, each model feature operation is tested against the entire library for geometric conformity. The method employs semantic matching and rule-based identification, using the inherent identifying constraints of a feature. This approach would allow the identification of functions most suited to create an identical geometrical outcome, rather than relying on the translation supplied from a static mapping. Feature mappings that satisfy constraint requirements and geometric validation would be added to an ontology for future reference.

A similar approach is taken by Rappoport as the basis of the Proficiency CAD translation software (Rappoport, 2003). The geometric difference created by each feature operation in a model is tested against a set of features in the target CAD program likely to recreate the geometry, this process is known as “rewrites” in the Universal Product Representation method described. In instances where no satisfactory match is found, the geometric difference is recreated using surface patches. Testing for optimum substitutions from a range of known likely candidate features can be described as a one-to-many mapping between the features of two CAD programs, however, this description may also apply to a mapping that requires several distinct operations in a source CAD to replicate the action of a single operation in a target CAD.

The second value of the dynamic matching approach is the capacity to incorporate implicit constraints from prior feature operations. As a completed feature model is generated from a series of feature operations, this sequence may be reversed to reveal the intermediate stages of model geometry. This deconstruction process can then return the difference in model surface geometry before and after any single feature operation. As Rappoport explains, unknown mappings may be substituted by a surface patch, that replicates the geometric difference. Likewise, it is possible to substitute the entire feature-based geometry with multiple surface patches to recreate the model geometry prior to a feature operation. This substitution of intermediate model construction geometry can address the difficulty of determining the response of a proxy function to the implicit constraints inherent to a specific model geometry. Rappoport *et al,* identify the shortcomings of surface substitutions; the behaviour of functions is dependent on the selection of model geometry faces and edges, which in turn, is innate to the vendor CAD program rather than an agreed convention (Rappoport *et al,* 2006). If an intermediate boundary model representation does not have the constituent edges and faces that would be normally generated in the target CAD program, the implicit constraints that direct feature behaviour are not represented. It would be expected that the substituted boundary model of a cube would have consistent edges and faces, but in cases like a cylinder, the curved surface may comprise one or two faces depending on the CAD program. The procedure underlying the Proficiency software identifies and maps these inconsistencies, substituting functions and splitting faces to provide workarounds. In summation, the technique of dynamic mapping partially solves the problems of implicit constraints and one-to-many mappings at the expense of further complexity. It appears evident that a method that can discriminate between CAD geometric models might determine whether a mapping matches, or fails to match a recreated model in a target CAD system, but does not necessarily identify the correct feature function required to create a successful mapping.

8.4 Indirect feature operations

The general form of feature operation adds or modifies an element within a parametric CAD model. Feature function parameters indicate the type of input required to implement the feature operation. If the effect of the feature operation cannot be detected in the subsequent modified geometry, then the method of comparing geometric feature output between CAD programs to verify function mapping cannot determine a match. This limitation of geometric comparison may be extended to features that have an indirect effect on geometry behaviour. Two important cases may be described. The first is in cases of granularity mismatch between feature operations. The outcome of a particular feature operation in a source CAD system may require more than one feature operations in a target CAD systems. This is another manifestation of a one-to-many mapping, as distinct from several functions in a source CAD that have the same effect as a single function in a target CAD. Here again, the distinction may not be tractable.

As an example, a single filleting operation may provide several optional variants within one function, in another CAD the same filleting operations is unlikely to have the same set of variants and would require extra modification to achieve an equivalent geometric outcome. Yet if both feature functions shared a subset of matching variants, this provides a partial match. A second category of potentially intractable feature operations require multiple steps to effect a geometric model modification but do not create incremental or intermediate geometry differences during these steps. One example is that of a complex selection operation that requires an iterative process to select a particular face or edge from a linked set of geometry elements. Another feature operation that may have an elusive effect on subsequent geometry is that of explicit geometric constraints.

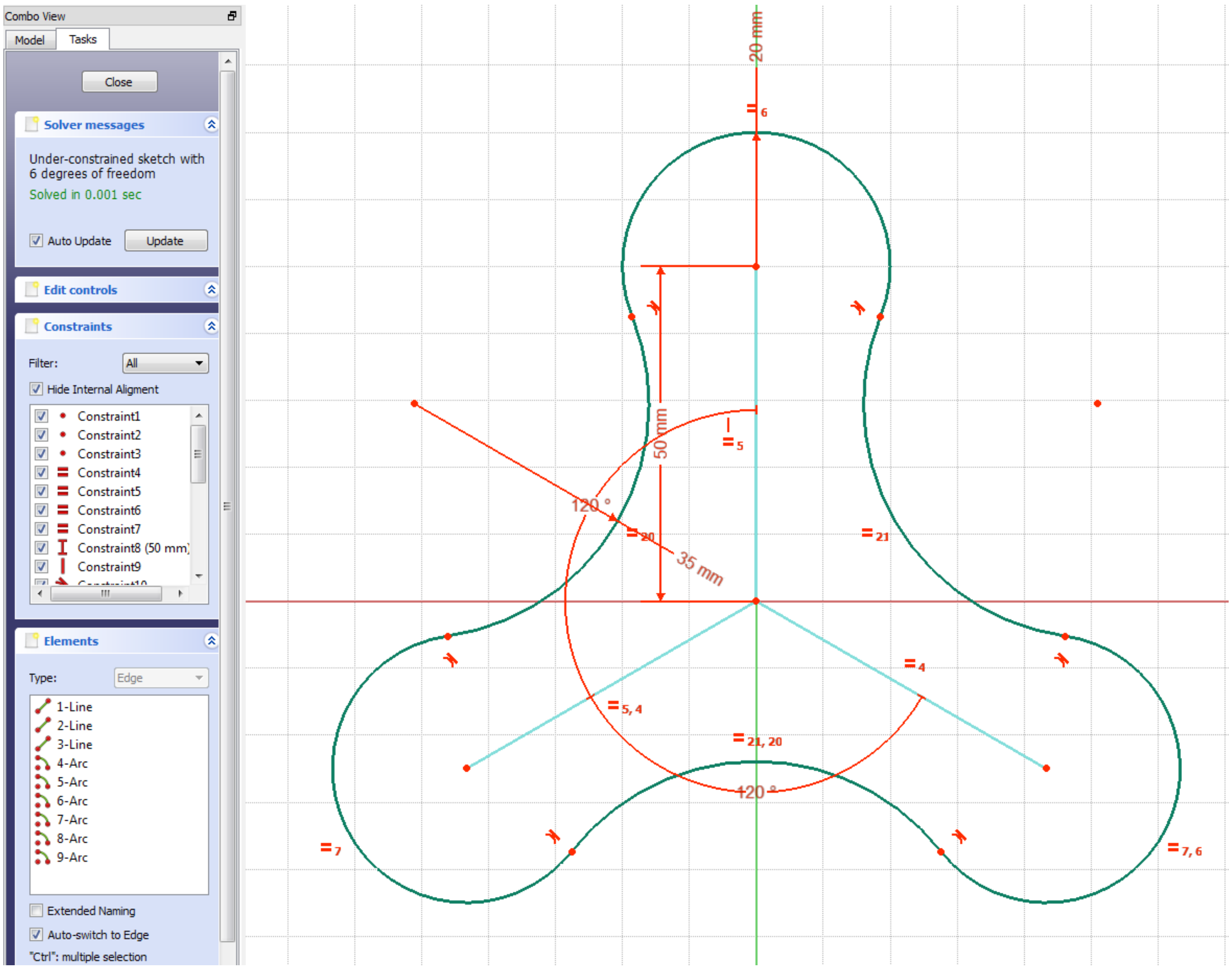
8.5 Explicit geometric feature constraints

The relationship between geometric entities within a 2D or 3D representation is explicitly defined by constraints, these constraints are either parameter values relative to the model space or relationships defined between independent features. It is common for the two dimensional elements in parametric sketches, such as lines, curves and ellipses to be defined relative to other sketch elements, three-dimensional models composed of multiple individual elements may use explicit geometric constraints to describe the relative positions of these assembly elements. Explicit geometric constraints present a potential difficulty with an approach that compares model surface geometry between feature operations. While an explicit geometric constraint is not manifested as a boundary surface geometry, the existence of an explicit constraint may be determined in the behaviour exerted on geometric elements. It may not be immediately intuitive how an explicit constraint is geometrically determined. If a constraint relationship is created between two 2D or 3D elements, these elements may move to conform with the applied constraints. For example if a parallel constraint is applied to two straight lines, one will move to become aligned in a parallel relationship to the other, unless both lines are already parallel. This illustrates the difficulty of geometric testing for operations that are not guaranteed to respond with a measurable geometric outcome. There are heuristic test geometry configurations that may be used to identify particular explicit constraints while avoiding false positives from other constraints with a similar effect. To extend the parallel line example, two lines used to detect a parallel relationship constraint may fail to discriminate the outcome from a co-linear relationship constraint.

The considerations of matching geometries determined by implicit feature constraints and explicit geometric constraints between features have been addressed, the third type of feature constraint is that of explicit feature parameters. These constraints are equivalent to the function parameters that represent feature operation in the CAD application programming interface. The next section deals with methods to reduce the search space associated with multi-variable features for the purposes of geometric matching.

8.6 Parametric variables

Modern feature-based parametric CAD programs employ a mix of implicit and explicit constraints in feature definition. Explicit constraints are equivalent to the parametric variables used to define a function, directly accessible via a graphic user interface or the feature function API. Mapping CAD feature functions requires that the parameters specific to each function are also mapped. This task is complicated by the differing scope between nominally similar functions, there may be a parameter configuration that cause two functions to create an identical geometry, but this may be a small intersection of the combined sets of shape geometries that these functions can create.



Under-constrained sketch with 6 degrees of freedom
Solved in 0.001 sec

☑ Auto Update

Filter: All
☑ Hide Internal Aligment

☑ Constraint1
☑ Constraint2
☑ Constraint3
☑ Constraint4
☑ Constraint5
☑ Constraint6
☑ Constraint7
☑ Constraint8 (50 mm)
☑ Constraint9

Type: Edge
1-Line
2-Line
3-Line
4-Arc
5-Arc
6-Arc
7-Arc
8-Arc
9-Arc

☐ Extended Naming
☑ Auto-switch to Edge
"Ctrl": multiple selection

Figure 38: 2D CAD geometry generated using explicit constraints and parameters (FreeCAD 0.18)

In Figure 38, a two-dimensional sketch shows a geometry partly defined by constraining relationships between sketch elements and partly by fixed sketch element parameters. The tri-lobed outline is composed of six arc-segments. The inner arcs are constrained to the same length, as are the outer three arcs. Each arc is constrained such that the end is tangential to the end of the adjacent arc. The centres of these arcs are placed at the ends of construction line segments constrained to equal length. The radius of a single outer arc, inner arc, the length of a single construction line and the angle between construction lines is sufficient to define the dimensions of the entire 2D geometry.

As another example, a frustum or a truncated cone may be created by a cone feature, or a cylinder feature, or a cone that is subsequently cut to form this geometrically identical feature. To make matters worse, there is generally several operations within any single CAD system that will create the same geometry. Certain CAD programs have their own recommended methodology to generate shapes, but there is no settled orthodoxy between different vendor programs (Camba *et al*, 2016). To date, any manual mapping between similar CAD feature functions has required expert and laborious intervention, the issue of unmatched feature subtypes, unmatched constraints such as datum-points and end-conditions are not trivial problems (Barber *et al*, 2010). These efforts have tried to create exhaustive static translations between independent functions within CAD feature libraries. As referenced earlier (Section 8.3, Sequential model generation), the implicit constraints inherent to sequential parametric model creation make it impossible to anticipate all potential feature configurations within a static mapping library.

The dynamic mapping processes envisaged by Tessier, and in the UPR described by Rappoport avoid this limitation. A dynamic mapping process will search for a geometric mapping from one feature model state to a subsequent model state. This dynamic mapping process may search the entire CAD library for a suitable function or trial a shortlist of probable feature functions. This technique has a higher probability of determining suitable feature matches where implicit constraints would have an unquantifiable effect on an equivalent feature selection determined by a static match. A practical implementation of this dynamic feature matching would retain a selection of suitable feature candidates, with the attendant requirement that there must be a database of static function matches from which to draw this selection (the UPR rewrite method hints at this strategy). Creating a database or an ontology and populating it with known function matches is a relatively straightforward task, finding matches between CAD features is not. Within each CAD program, there may be several hundred functions that create or manipulate the geometry of a CAD model. Each of these functions may in turn have many parameters that alter the behaviour of the function. The number of permutations of function operations possible make a brute-force combinational search impractical. The task of feature matching becomes a challenge of partitioning the combinational search space into manageable proportions.

8.6 Parametric variables

Geometry matching may be independent of relative model rotation, translation or scaling, but a method that may accurately compare surfaces cannot discriminate models that differ by varying proportions. A topological comparison may reduce the set of potentially equivalent features, but it cannot determine features that are functionally equivalent in geometric behaviour.

If the geometric manifestation of a feature operation is dependent on the associated feature parameters, it is necessary to find the combination of parameters for a feature operation in a source CAD model environment that replicates an identical geometry of an equivalent feature in a target CAD environment. If both features have multiple parametric constraints, then a successful geometric match will require a search for two sets of feature parameters that both produce an identical model geometry. As the number of parameters associated with each feature increases, the possible permutations of the operation increases. An exhaustive and undirected combinational search has no prior information on parameter order.

As a trivial example, consider two hypothetical features undergoing matching, the first containing three boolean variables from a source CAD program S, the second from a target CAD program T, again containing three boolean parameters,

$$S\left\{s_1, s_2, s_3\right\}, T\left\{t_1, t_2, t_3\right\}$$

If these parameters govern equivalent configurations of each feature geometry then for any combination of parameter values of T, they are guaranteed to be matched within 2^3 trials, the number of permutations with repetition allowed. This is a combinational search problem with a worst case of exponential complexity, which is not amenable to a practical function mapping application.

Two feature functions undergoing comparison may not encapsulate the same functionality, it is correspondingly unlikely that these functions will share the same number of parameters. This can be illustrated by extending the first example, where a function containing four boolean parameters from a source CAD is matched to a function containing three boolean parameters within a target CAD as before.

$$S\left\{s_1, s_2, s_3, s_4\right\}, T\left\{t_1, t_2, t_3\right\}$$

If a geometric match between both functions is independent of the extra boolean parameter within S, then the maximum number of trials is 2^4, if a match is not independent the maximum number of trials required to find a match is 2^3. In an undirected search it is unknown whether a parameter is required for a match. A brute-force combinational parameter matching approach does not appear optimal for several reasons:

1. Real CAD feature functions may contain a substantial number of variables, leading to an exponential increase in search space. While certain commercial CAD API such as AutoCAD Inventor ® use a feature object paradigm that embody the set of methods and parameters, most CAD API tend to represent feature variants as additional function parameter options. This leads to a high combinational space when comparing functions.
2. A combinational approach is impractical for an unbounded parameter type, such as the set of integers, or floating point numbers.
3. A commercial CAD API may contains many hundreds of feature properties and methods. An undirected combinational search would require that a feature function in a source CAD be tested against every function in a target CAD. The potential for a one-to-many match, where the source function has a positive match with several target functions implies that feature searches must be exhaustive.
4. An exhaustive search cannot benefit from additional data, such as semantic matches of function parameter names, or the proximity of the function in the API data model to known functions.

It is helpful to illustrate the claims above with data taken from the three commercial CAD programs tested for semantic similarity in Chapter 5.7. The documentation and text associated with the API of Rhino 5 Rhinoscript library, Solidworks 2010 API and Autodesk Inventor 2012 is parsed to yield several XML files of uniform format. These files capture the function names, parameters, parameter data types and whether a parameter is required or optional.

Figure 39 shows the proportion of functions relative to the number of associated function parameters. This graph has a logarithmic vertical axis to represent the number of functions of a certain number of parameters because of a wide variation in the distribution within each API, and also between each API. Several observations may be made with respect to the distribution of functions according to their number of parameters.

- The Inventor API follows an object model, many functions are descended from a "parent" function and are an object property that simply returns an object state, other functions are methods which might configure or determine the properties of an object. As an example the `Arc2D` object contains a method `Arc2D.PutArcData`, that sets the parameters defining an arc, it also contains a property `Arc2D.Radius`, that may be used to determine or configure a single object parameter. This style of API architecture creates a large number of single parameter functions.
- Both the Inventor and Solidworks API contain a larger number of functions than the Rhinoscript interface.
- The distribution of functions ranked according to number of parameters is an approximation to an inverse frequency relationship, this has important ramifications for the tractability of an automated search where the combinational search space is an exponent of the number of function parameters, this concept is covered in greater detail in Section 8.7.

8.6 Parametric variables

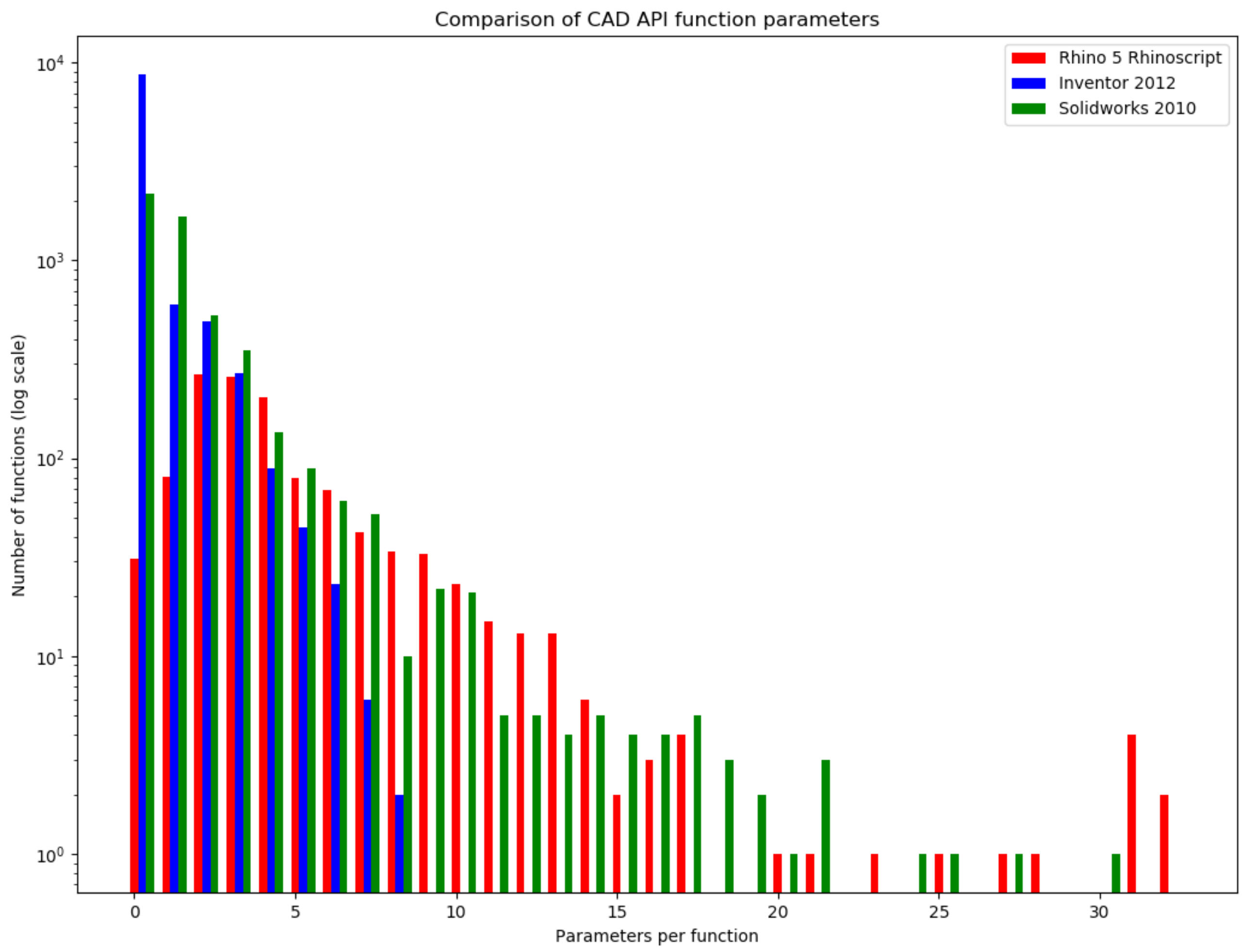


Figure 39: distribution of CAD API functions ranked according to number of parameters

Geometric matching is a class of problem that can be solved within Non-deterministic Polynomial time or NP-hard, the intuitive verifier-based definition is satisfied by the polynomial complexity of proving two surface models are geometrically identical, while an exhaustive combinational parameter search is of exponential complexity, as indicated in the simple example above. No simple or unique solution exists to direct a feature function parameter search for the purposes of feature matching. However, a combination of techniques can divide the potential search space into tractable partitions. These techniques are outlined as follows.

8.6 Parametric variables

8.6.1 Function parameter label matching

Semantic matching is the use of the text associated with the API feature function description or type library to estimate relationships between similar concepts. This text may be drawn from help files, from the feature function names or from the embedded text in object files. These methods are more fully described in Chapters 4 and 5, where the relative effectiveness of various techniques are tested on the short texts found in function descriptions. While the tests in Chapter 5 show the accuracy of the different techniques over a set of functions, the same methods can be used to detect syntactic or semantic similarities between the names or descriptions of function parameters undergoing testing.

8.6.2 Object model inference

The architecture of CAD program API tend to reveal conceptual relationships between feature functions. In all CAD programs examined (see table on page 35), there is a relationship between the API data model and the CAD feature taxonomy. While all major vendor software are based on an object-orientated data structure, API architecture is also shaped by the customs and constraints of the interface language. Some API structures such as RhinoScript 5, a native Python language interface to the Rhino 5 CAD program have a relatively flat object model with several object type categories and simple object handles and data structures. Other programs, such as CATIA V5R21 Automation interface have several layers of parent-child hierarchy that group feature functionality by inheritance, allowing structural relationships between functions to be deduced based on path length between function positions within the object model. Examples of similarity measures are Wu-Palmer, Resnik, Jiang-Conrath, Lin & Leacock-Chodorow. A full description of these methods is to be found in Chapter 4.5, WordNet similarity measures.

8.6.3 Object model type inference

The API object data model generally has a structure that reflects the inheritance of feature objects. There is a more fundamental set of data types that are readily recognisable across different CAD API. A typical example is that of the floating point triplets that describe points in 3D space. These points are invariably floating point numbers commonly defined within an array structure containing three values. Another frequent parameter is a pointer to an object, or in some cases a reference string that is the handle for a feature object type, such as a model feature, or a surface face, or an entire body. Integers will invariably reference iterations rather than geometric values.

8.7 Function parameter type heuristics

Certain heuristics may be used to reduce function parameter search space in a search for function equivalence. What follows is a typical sample from the Solidworks 2010 C++ API documentation that was introduced in Solidworks 2001. The first three parameters establish the position and orientation of the feature, a conical surface, the two parameters that follow establish the proportions of the feature. The function returns a pointer to the created conical surface object.

```
CreateConicalSurface2(center, direction, refDirection, radius, semiAngle)
```

	Parameters	**Description**
Input:	(double*) center	Pointer to an array of 3 doubles, XYZ location which represents the center of the bottom
Input:	(double*) direction	Pointer to an array of 3 doubles, XYZ direction of the axis of the conical surface
Input:	(double*) refDirection	Pointer to an array of 3 doubles, XYZ direction of the axis of the conical surface
Input:	(double) radius	Radius at the center
Input:	(double) semiAngle	Half angle of the cone in radians
Return:	(LPSURFACE) retval	Pointer to the resulting Surface object

Table 6: parameter description of SolidWorks CreateConicalSurface2 function.

8.7 Function parameter type heuristics

Reverting to the simple illustrative notation used earlier in Section 8.6, this functions may be modelled as a source CAD function, where this time the parameter values in Table 6 are floating point numbers such that,

```
retval = CreateConicalSurface2(center.x,
                               center.y,
                               center.z,
                               direction.x,
                               direction.y,
                               direction.z,
                               refDirection.x,
                               refDirection.y,
                               refDirection.z,
                               radius,
                               semiAngle)
```

Reformulating this function where parameters are grouped by Cartesian triplets then gives,

$$s_1 = S\left\{\left\{s_2, s_3, s_4\right\}, \left\{s_5, s_6, s_7\right\}, \left\{s_8, s_9, s_{10}\right\}, s_{11}, s_{12}\right\}$$

Once the assumptions that floating point triplets represent spatial coordinates are factored in, the number of independent variables are halved. Recognising that the returned type s_1 is a pointer handle to a data object and most likely the feature associated with the function reduces the number of unknown parameters further.

$$S\left\{s_1, s_2, s_3, s_4, s_5\right\}$$

8.7 Function parameter type heuristics

While s_4, s_5 relate to the morphology of the feature, s_1, s_2, s_3 determine the positioning and orientation of the feature within model space. Two of these three coordinates specifies the axis of the feature object, meaning that if the exact same coordinates are used, the function will return an error. Any other coordinate groups will create a feature that can be normalised with respect to absolute position, scale and orientation. Consequently the function has been reduced from twelve to two parameters using these heuristics.

The scope of the data type heuristics described above are demonstrated with a comparison against a function with an equivalent geometric behaviour from the Autodesk 2012 Inventor COM API Visual Basic reference. Note the slight differences in data type declaration convention. The same assumptions as used above reveal two coordinate data types, two floating point types that determine cone proportions and a boolean value that determines cone orientation.

```
Sub PutConeData( ByRef BasePoint As SAFEARRAY(double),
                 ByRef AxisVector As SAFEARRAY(double),
                 Radius As double,
                 HalfAngle As double,
                 IsExpanding As VARIANT_BOOL)
```

	Parameters	**Description**
Input:	BasePoint	Input/output Double that specifies the base point of the cone.
Input:	AxisVector	Input/output Double that specifies axis vector.
Input:	Radius	Input Double that specifies the radius of the cone.
Input:	HalfAngle	Input Double that specifies the half-angle value for the cone.
Input:	IsExpanding	Input Boolean that specifies whether the radius of the cone is expanding or not, in the direction of the axis vector.

Table 7: parameter description of AutoDesk Inventor PutConeData function.

The two parameters governing the cone proportions are identical in both the Solidworks and Inventor functions. The convention for establishing a principle axis, or a direction vector differs, yet two the geometric models within the source and the target CAD programs will produce similar geometry for models that are normalised in scale, position and orientation.

The metric of similarity in this illustrative case can be the bidirectional Hausdorff measure, or a minimal affine-invariant feature description of extrema and symmetrical curves, in this case a single maxima at the tip of the cone, a single minima at the centre of the base and the single centre of the cone rim which coincides with the minima. These methods are described in detail in Chapter 6.25, A minimal set of registration features types, Explicit and implicit CAD model constraints . The task of identifying a similarity between two functions calls for a comparison between a potentially large number of geometric shapes. An efficient approach is to use a minimal representation of each shape, followed by a more detailed comparison of surface geometry. Hence the use of affine-invariant feature representations which store a relatively small number of identifiers and which may be compared in two stages. In the schema introduced in Chapter 6, several features types are extracted from the geometric model, the number and class of each feature type provides a descriptive signature for each shape.

A comparison of affine feature categories will indicate a match in this example, both functions will generate a cone with a single minima, maxima and curve centre. These points are further distinguished by their relative displacement from the cone centroid. If random numbers are used as parameters to generate the cone model, it is unlikely that the respective signatures or vectors of point displacements will match. If the same values are used for both function parameters governing cone proportions, both cones will match, despite having a different parameter order. Similarly, if the values used to specify Cartesian values for points or vectors in the function parameters are taken from a minimal n-ary set of values, this reduces the base of the exponential combinational search space. For instance, a function requiring eight or 2^3 unique Cartesian triplets of floats may be specified in a psuedo-binary alphabet, [0.0, 1.0]. A function requiring 27 or 3^3 unique Cartesian triplets might use a trinary alphabet, etc. This heuristic is used for the parameter coding of the genetic algorithm solution presented in Section 8.12.

In Chapter 6, a method to efficiently match geometric shapes is presented. This method can determine whether two shapes are geometrically similar, independent of orientation, position or scale. Such a method may act as an objective function to allow automated testing of feature functions for geometric output equivalence. While such an oracle may indicate whether the shape created by two feature functions is similar, it cannot determine the required parameter combinations of functions undergoing comparison.

8.8 Parameter mapping problem formulation

If this mapping process is described as a combinational search for feature function parameters which minimise a measure of geometric difference of the function outputs, then the problem can be formulated as a Constraint Satisfaction Problem. The Constraint Satisfaction Problem is defined as a triple $\langle X, D, C \rangle$, such that,

$X = \{X_1, \ldots, X_n\}$ is a set of Constraint Satisfaction Problem variables, each of which may assume values within individual domains,

$D = \{D_1, \ldots, D_n\}$, subject to the set of constraints $C = \{C_1, \ldots, C_m\}$.

These domains represent the set of possible values that each X_j may assume. Each constraint C_j specifies a subset t_j of the variables X with a k-ary relation R_j on the corresponding subset of associated domains D_j. This relation can be expressed as the tuple $\langle t_j, R_j \rangle$ such that assigning variables to t_j satisfies the relations R_j. A vector of variables assigned to each member of X which satisfy all members of C is a solution. A CSP may also have a solution that maximises or minimises an objective function.

Unlike the generic representatives of the Constraint Satisfaction Problem, (such as Sudoku puzzle generators and 8-queen puzzles), the constraints inherent to CAD feature functions are not known prior to mapping. The search for parameters which generate identical geometries may be considered to have two distinct types of constraint, hard constraints and soft constraints. Hard constraints are equivalent to inviolable rules which, if broken, result in no solution. In this case, hard constraints are combinations of parameters that cause a CAD feature function to return an error rather than an instance of a shape. In the cone example presented later in Section 8.6, a FreeCAD `makeCone` function specifies a radius for the base and another radius for a frustum geometry, naturally if both these radii are equal the function produces a cylinder, but as this configuration is too far from the semantic conception of a cone or frustum, this combination of parameters is impermissible and does not output a shape. A soft constraint may be considered as a suboptimal solution according to a designated metric, which in this case is represented in the geometric similarity measure. An optimal solution has a minimal difference between generated shapes, which may be determined by a measure of geometric similarity.

While individual constraints C_j may only reference a subset of parameters within a single function, the minimisation of shape difference is a global cost function (also known as a criterion function or objective function). A Constraint Optimization Function describes problems that require determination of variable assignments X, which satisfy the hard constraints C and return an optimal value for soft constraints, in this case the minimisation of a global objective function. This representation can be given as,

$$F\left(S\left(x_1,\ldots,x_n\right),T\left(x_1,\ldots,x_m\right)\right)=\Delta_H$$

Where S represents the source function of n parameters from one CAD system undergoing mapping to a target feature function T of m parameters in a second CAD system. Solving this formulation gives a parameter configuration for functions S and T which minimises the difference between function geometric output. However, there may be several solutions that minimise the objective function. Both functions may generate equivalent geometric shapes, but in different orientations. Equivalent parameters that alter the proportions of output shapes may generate entire families of geometrically equivalent shapes. The discovery of a single configuration of parameters that minimise the objective function does not, by itself, reveal a mapping between individual function parameters.

A further objective is required to identify a solution with the most minimal values of parameters. This aspect is covered in the description of the heuristics used in Section 8.12.3, at present it is sufficient to state that solutions that incorporate a high percentage of zero-valued parameters tend to have reduced dimensionality. Finding a set of parameters that satisfy hard feature function constraints while minimising a measure of similarity and parameter value may be classified as a Multi-objective Optimisation Problem incorporating both hard and soft constraints, respectively a combination of a CSP and COP. The hard constraints of the function parameters are unknown, the soft constraints of geometric similarity and minimisation of parameter values cannot be considered as continuous, linear functions.

The continuity of the feasible parameter space may be characterised by the general categories of parameter, for instance, related feature constraints that define feature geometric proportions tend to be linear. Using a cone example, the radius values of two cone features will cause a geometric similarity measure to vary proportionally to the difference of their relative values. However other parameters, such as a boolean or Cartesian array values that reverses the orientation of a feature shape will cause a non-linear effect on a geometric similarity measure. The combination of unknown parameter constraints and multiple non-linear objective functions limit the methods that may be employed to solve this class of problem.

If this mapping problem is described as a decision problem, the combination of parameters tested give an answer to whether the parameters are a legal operation for both functions. They may also answer whether the geometric difference of the respective function outputs is below a numerical bound. The complexity of a problem is related to the time and space required for a particular algorithm to solve it. In combinational problems where the algorithms tend to be search methods, this is defined as the worst-case asymptotic time complexity.

Two noteworthy complexity classes are those of $\mathcal{P}$, the set of problems that can be solved by a deterministic machine in polynomial time, and $\mathcal{NP}$, the set of problems that may be solved by a *nondeterministic* machine in polynomial time. A *deterministic* algorithm will always retrace a determinable execution path, while the progress of a nondeterministic algorithm cannot be predicted in advance, even with identical input. Conceptually, a nondeterministic algorithm is not necessarily a random process, but one which may take different execution paths, or may “guess” a process step. The equivalent so-called *verifier* definition of $\mathcal{NP}$-class problems is the set of problem instances where a positive solution may be verified in polynomial time using a deterministic algorithm.

8.9 Problem complexity classification

This conceptualisation of problem complexity allows a useful classification. $\mathcal{NP}$-hard problems are those which are at least as hard as any $\mathcal{NP}$ problem, but are not themselves necessarily a member of $\mathcal{NP}$ problems. A simple problem may be readily mapped to a more complex problem, but there is a distinct set, $\mathcal{NP}$-complete, to which any $\mathcal{NP}$

problem can be transformed to within polynomial time (this process is also termed *polynomial reduction*). There are a number of well-researched $\mathcal{NP}$-complete problems that shed light on similar decision problems.

Returning to the case of identifying a mapping between two CAD feature functions, it may be assumed that the verification of both soft constraints and hard constraints are deterministic. The hard constraints are defined by the legal input of the respective feature functions. While the operation of the function response may be indeterminate, obfuscated by a compiled executable, the task is to determine whether the function input is of correct type and range, which is readily accomplished by a deterministic algorithm in polynomial time. The soft constraints of the mapping are also partly reliant on assumptions regarding the operation of the CAD software.

If it is assumed that a deterministic algorithm is used to find the intersection of a vector and a surface within the CAD program then the process of checking the geometric difference between CAD models may be verified within polynomial time. The second soft constraint, that of overall input value minimisation, is a summation which is readily calculated. These independent verifications of each parameter combination solution occupy polynomial time, with the implication that the function mapping problem is a member of the set of $\mathcal{NP}$ problems.

This feature mapping problem can be specified as $\mathcal{NP}$-complete by comparison against the so-called *knapsack* problem. In brief, the knapsack problem is a classic multivariate optimisation problem. A selection of items of different costs and volumes must be chosen to fill a knapsack of finite volume, these items must be selected to maximise the value of the contents of this knapsack. Karp proved that this problem could be mapped to a known $\mathcal{NP}$-complete problem (Karp, 1972). This can be formally stated as a finite set U of elements, each possessing two properties, one defining the volume of each element, $s(u) \in \mathbb{Z}^+$, the other defining the value of each element, $v(u) \in \mathbb{Z}^+$ for each $u \in U$ where $\mathbb{Z}^+$ is the set of positive integers. The knapsack problem is the search for a subset $U' \subseteq U$ such that,

$$\sum_{u \in U'} s(u) \leq B \qquad \text{and} \qquad \sum_{u \in U'} v(u) \leq K$$

where there is given a volume constraint $B \in \mathbb{Z}^+$ and a value threshold $K \in \mathbb{Z}^+$.

The knapsack problem can be shown to map to the feature function mapping problem if the set U is taken to be the set of all legal input parameters for both functions. Define the similarity function $\Delta_H \leq E_H$ to be satisfied if is below a discrimination threshold E_H, and the summation of parameter values $\sum_{i=n} x_i + \sum_{j=m} x_j \leq S_X$ to be satisfied if less than an arbitrary value S_X. Then a set U' can represent a solution instance that may satisfy a linear combination of these objective functions. This representation can be mapped to the knapsack problem formulation, meaning that for simple examples missing hard constraints, the feature mapping problem is $\mathcal{NP}$-complete. Functions with parameters that dictate hard constraints are as least as complex as the knapsack problem, with parameter subset solutions that can be verified in polynomial time, meaning that they too are $\mathcal{NP}$-complete. Certain $\mathcal{NP}$-complete problems such as the knapsack problem have approximate solutions for limited subsets in $\mathcal{P}$, known as polynomial-time approximation schemes (PTAS) (Hromkovič, 2013). These work in polynomial time for an approximation ratio $1 + \epsilon$, where the relative error ϵ is greater than zero, within a time limited by a polynomial function of an order equivalent to that of the problem instance. Where this time has an upper bound of $\frac{1}{\epsilon}$, these approximation schemes are known as fully-polynomial-time approximation schemes (FPTAS).

There are several approximate solutions to the related simple knapsack problem, e.g. (Johnson, 1974). To improve readability, the function parameter matching problem is abbreviated to FPMP, it must be remembered that the search for a set of parameters that result in a solution optimising soft geometric and value constraints does not in itself solve the identification of equivalent parameters. It is, however, the starting point for method that does so. While this FPMP may be at least $\mathcal{NP}$-complete in complexity, this does not rule out practical solutions. $\mathcal{NP}$-hardness reflects the worst case complexity, yet for many problems the majority of solutions may be found in a comparatively short time. This may be shown empirically for certain algorithms such as the Simplex Algorithm for linear optimisation, which has an exponential complexity in the theoretical worst case but returns solutions in polynomial time in the average case (Klee & Minty, 1972). Other successful practical solutions to $\mathcal{NP}$-hard problems identify a problem subclass that is more tractable. The signed SAT problem is $\mathcal{NP}$-complete, yet some of its subclasses are polynomially solvable (Beckert *et al*, 2000).

Where the order and behaviour of function parameters is unknown and there is a strong possibility of discontinuous or multimodal function response to function input parameters, it is unlikely that a deterministic method would be successful for matching a broad range of function types. An enumerative approach may be effective for functions with few parameters, but does not allow additional problem information or information derived from prior enumerations to guide the sequence of parameters to test. Stochastic search methods have been developed to tackle these particularly intractable classes of problems. In the next section, a brief overview of local stochastic search and optimisation methods serves to justify selecting an evolutionary algorithm to tackle the FPMP.

8.10 Stochastic Local Search methods

SLS methods are briefly introduced as a common approach to hard combinational problems. Information from a candidate solution is used to guide the search direction. This information may be simply a tally of search attempts for a restart algorithm, or data on recent search candidates used in a *tabu* search that avoids repetitively exploring the same search region. Local searches may be divided into a local search over a complete solution space, traversing the entire solution set, or local searches which find a path to the optimal value in a partial solution set. As complete searches, such as tree searches are impractical for $\mathcal{NP}$-hard problems, they are not covered further.

A local search may be qualified by several common features, namely,

- a search space, S which is the set of all possible parameter variations

- a solution set, $S' \subseteq S$ which is the subset of candidate solutions to the problem
- a neighbourhood relation, $\mathcal{N} \subseteq S \times S$ defining the relationship by which candidate solutions are considered search neighbours
- an evaluation function, $g : S \mapsto \mathbb{R}_0^+$ whereby the response of a solution candidate to the problem may be determined

The local search starts from an initial position s_0, then uses information from the evaluation function to determine to which neighbouring position the search should progress. *Constructions heuristics* may start from an initial point and then extend a solution based on a neighbourhood evaluation heuristic. The *first fit* or *best-first algorithm* uses a greedy approach to select the highest-scoring neighbour for staging the next iteration. *Bounded backtrack algorithms* use heuristics to determine the scope of candidate solutions from neighbours reached by previous branching search path positions. Examples of backtracking heuristics include *credit-based* algorithms that employs an initial credit parameter which determines the breadth of the initial search paths, the distribution of credit among these created search paths as a measure of liberty to explore path depth, and finally a backtracking measure to determine the scope of the search neighbour cluster at the termination of individual search paths. The issue with local search techniques are that they tend to get stuck in local minima rather than finding a global minima, or maxima. The first fit algorithm is an example of a hill-climbing algorithm that suffers from this shortcoming. Adding random selection to the uphill neighbourhood moves available will improve global performance at the expense of speed.

Stochastic Local Search algorithms add a measure of randomness, or use other metaheuristics to find optimal solutions. The *random-restart* or *shotgun* hill-climbing algorithm performs a sequence of the search paths from random initial positions, increasing the likelihood of detecting a global extrema among local extrema. *Simulated Annealing* combines the exploration of a random neighbourhood search with the exploitation of local neighbourhood information. The metallurgical annealing analogy refers to the availability of transitional system states according to the level of free energy remaining in the system. The selection of the next iterative position in a simulated annealing search is based on an *acceptance probability function*, that changes neighbour selection from a relatively random candidate, to the most optimal candidate according to a varying "temperature" parameter. The choice of neighbour is further determined by the heuristic to select neighbouring candidate positions. The original formulation described by Kirkpatrick used a Metropolis function to determine the candidate selection probability (Kirkpatrick *et al,* 1983). This algorithm is generated as follows; for each candidate neighbouring state, a random energy difference $\triangle E$ is generated, representing the transition from the present state to the candidate state. If $\triangle E$ is negative, the candidate is accepted. Otherwise the probability of transitioning to a prospective state is given by,

$$P\left(\triangle E\right) = \exp\left(\frac{-\triangle E}{k_B T}\right)$$

where k_B is the Boltzmann constant and T is the temperature.

This probability decays exponentially with decreasing temperature, meaning that the search will converge to an optimum, but that it has less likelihood of becoming trapped in a local optimum in an early iteration of the algorithm. The other influences over the search pattern is the selection and qualification of suitable neighbouring candidates from the search space.

Tabu search is another metaheuristic to circumvent searches becoming trapped in local optima. Where a simple hill-climb search will not select a neighbouring candidate if the position has a lower value, the tabu search will make exploratory detours to lower valued positions in the chance of finding a higher valued optimum. As this strategy regularly leads to repetitive cyclic search path behaviour, the tabu search will store the values of recently visited search positions on a taboo list, avoiding the same path twice.

The search methods described are based on a single search path in the solution space at any time. This is distinct from *population-based* searches which use several concurrent search processes. A process such as an *Ant Colony Search* models each search path process as a foraging virtual ant that lays a trail "pheromone" for subsequent ants.

These trail pheromones degrade with time, or search iterations, so that short paths appear stronger. The technique combines elements of a tabu search, each ant agent recollects recent paths, a global heuristic, which is the relative strength of path pheromones, and a local heuristic, which may be a greedy search for nearby optimal candidates.

Particle Swarm Optimisation is another multi-agent search for a global optimum that combines a global heuristic, the transmission of information between particle agents, and a local heuristic such as a greedy search of the neighbouring candidates. These population-based searches are commonly hybridised with other search techniques for particular applications.

The search method used to find a mapping between feature function parameters is another population-based method, a *genetic* or *evolutionary* algorithm. In brief, these methods use concepts from genetics, such as a pool of individual solution candidates that are interbred and randomly mutated to evolve a search solution. The next section outlines the structure of evolutionary algorithms and the rationale for implementation of an evolutionary algorithm as the basis of a feature function parameter mapping search.
A Genetic Algorithm, or GA, is selected for three reasons,

A GA is arguably the most simple implementation of a local search method, the philosophy of selecting the most simple method to for exploratory research is described in Chapter 1.1.

A GA allows a combination of several search heuristics, in this case a semantic match score and a zero-valued parameter are trialled. This satisfies the requirement of a hybrid method search strategy described in Chapter 1.

A Genetic Algorithm can be considered as a robust search method. Consider pairs of function parameters that have a linear, proportionate effect on the geometry they create. A search method can readily determine a relationship using the response provided by an objective function measuring surface differences. However, once parameters with a non-linear effect on output geometry are introduced, the discontinuities in the solution space cause difficulty for traditional search methods. This is discussed in greater detail in Section 8.11, Genetic Algorithm overview.

8.11 Genetic Algorithm overview

A Genetic Algorithm is used to map unknown function parameters. The general properties of evolutionary algorithms are based on the concept of Darwinian natural selection, taken from evolutionary biology. A problem is formulated so that successive sets of promising candidate solutions are intermingled and refined to maximise an objective function. What distinguishes this method from other non-deterministic population search methods is the creation of new candidate solutions; rather than simply using enumerative or random methods to identify neighbouring candidate solutions from a search space, the candidates are assembled from the components of high-scoring previous candidates using processes that mimic genetic adaptation to evolutionary pressures.

This genetic metaphor extends to the representation of a candidate solution, the properties of the solution are encoded as an n-ary string similar to the base pairs of DNA. This representation enables the solution to be subdivided into smaller chunks, which may then be exchanged in *crossover* processes between parent solution strings. There are several schema that combine chance and fitness in selecting which parents to combine, as there are in selecting which portions of the parent strings to swap in order to generate offspring. The other important modification to solution candidates is the introduction of string *mutations*, analogous to the random mutations introduced into chromosomes by meiosis or mutagens. These mutations may simply be a random process to reverse the value of a bit in a binary coded string.

In a simple GA, a randomly generated population of candidate strings is tested against the objective function to reveal the relative fitness of each solution. These solutions are combined in a crossover scheme and subjected to random mutation, then evaluated again. This cycle is repeated for a predetermined number of generations, or it may be terminated should any individual reach a fitness threshold criterion. Genetic

Algorithms lend themselves to diverse combinations of random and probabilistic methods to mutate, modify or select individuals from a population. Candidates may be selected for reproduction based on a *roulette wheel* strategy, a random selection of individuals weighted by respective fitness values, or *tournament* selection where a random selections of candidates are reduced tyo their fittest members

For a GA to converge, there must be some indication of an improving fitness, this technique will not work with an entirely discontinuous or random solution space. What stands out is the ability of GA to converge to optimal solutions despite discontinuities in the search space, the robustness of this method is attributed to the survival of successful partial patterns, or *schemata,* in solutions with higher fitness values. These patterns are accumulate in successive generations, if a population is large enough to offset the destructive actions of crossover and mutations. A schemata is formalised as the same alphabet that comprises the chromosome coding, with an extra "wildcard" character that may represent any string value. This allows partial representations of solutions with irrelevant values ignored by the wildcard character. Holland used this definition to calculate the lower bound of the numbers of schema surviving crossover (Holland, 1975).

One observation that can be made from the analysis of partial solutions within the schemata is their durability across generations. Holland's schema theorem suggest *single-point crossover* that cuts both strings at the same site tend to sever linkages that arise from broad schemata, or in other words, where there is a relationship between a value near the beginning of a string and near the end, they are liable to be separated, while two adjacent values are more likely to survive a single-point crossover process. However, the findings of the *Exact Schema Theorem* of Stephens and Waelbroeck suggests that attempting to order the sequence of values in any string is futile (Stephens & Waelbroeck, 1998). The genetic algorithm used for the method demonstration use a single-point crossover, this is justified in Chapter 9.1.

In the following section, a genetic algorithm is described for matching parameters of two similar functions from heterogeneous CAD programs. This technique is a qualitative assessment of this search technique as compared with a brute force combinational search equivalent.

8.12 Representing a function comparison as a Genetic Algorithm

To represent a candidate solution of two functions undergoing geometric comparison, a string is generated. This string contains coded values representing the parameters of the functions, with the values representing the parameters of one function concatenated to the values representing the parameters of the other function.

The required parameters of both functions are coded as a psuedo-binary alphabet in the range [0.0, 10.0]. The choice of values for an n-ary coding is chosen to be the minimum number of default values available over the union of parameters of the two functions. This gives a string, or chromosome, to use the GA parlance,

$$\left[x_1^S, \ldots x_n^S, x_1^T, \ldots, x_m^T\right] \longmapsto \Delta_H(S(x_1^S, \ldots x_n^S,), T(x_1^T, \ldots, x_m^T))$$

Where $\left(x_1^S, \ldots x_n^S\right)$ are coded values mapped to the parameters of the source function, S, and $\left(x_1^T, \ldots x_m^T\right)$ are coded values mapped to the parameters of the target function, T. This formulation is used to search for a permutation of parameter values over both functions undergoing comparison such that the geometry created is identical within the Cartesian model space of the two respective CAD programs. This outcome does not identify equivalent parameters between functions, but serves as a "ground state" from which parameters may be readily identified. Three heuristics are proposed to reduce the combinational space of the potential solutions, these are described below.

Figure 40 shows an overview of the cyclical process used to determine this "ground state" that permits a mapping to be found between individual function parameters, details of this subsequent mapping process are given in Section 8.14. The diagram in Figure 40 shows the psuedo-binary alphabet representation of independent CAD parameter values, where the list of numbers at the top of the diagram is a fragment of a generated population. These individual candidates can be seen as a concatenation of the two sets of required parameters to generate function outputs in the respective CAD programs undergoing function mapping. In this case both functions generate a cone geometry. Both these cones might then be probed for similarity within their respective CAD model spaces. In the lower part of Figure 40, these cones are shown superimposed on one another within a common Cartesian model space. This superimposition is intended to illustrate the intersection of radiating vectors from the common model space origins with each cone instance. The distance between the points at which the rays intersect with the cone surfaces constitute the geometric difference between the respective cone surfaces as sampled at each ray. This measured disparity is highlighted in red for clarity. These surface difference measures may be summed to produce a single number that represents the sampled similarity between cone surfaces. If the parameters that generate the respective cone geometries produce two nominally identical cones, this surface deviation value will be reduced to the error applicable to the relative precision of both CAD programs (Chapter 3.12 describes a proxy model that encompasses these errors of program geometric precision). This surface deviation value is then suited to use as a fitness value that represents the success of each GA candidate. A bidirectional Hausdorff measure (Chapter 3.11) using the same set of intersection values may also be used in this application.

8.12.1 Restricted range parameter coding

Zero is a commonly used default parameter across CAD programs, either for binary values or for Cartesian points. Another example is the default value of 360 used in the *angle* parameter of the Part.makeCone FreeCAD function (see Chapter 9.6, Published parameters of the functions used in tests for complete API details), in one test configuration, a [0.0, 360.0] binary was used but did not converge to relatively small values, suggesting that this method is sensitive to scaling. As each additional parameter within a function introduces an exponential increase in potential parameter combinations, the coding of parameter values is reduced a minimum of bases. In practical terms this means that if two values may be used as legal input to both functions, the coding for generating candidate solutions should be restricted to these two values, a psuedo-binary, three values imply a psuedo-trinary and so on. While a binary or boolean normally refers to a data type in the context of function parameters, a restricted coding alphabet may contain floating point, integer or string values. The number of values, or codes used in the alphabet set corresponds to the base of the exponential combinational search space. The strength of a genetic algorithm is the ability to retain and combine partial solution fragments in the search for an optimal solution value. Consequently continuous unbounded parameters such as unsigned integers, or floating point representations that have a base-10 representation may be discretised and re-encoded as a base-2 representation to allow the preservation of longer string fragments (Goldberg, 1991).

8.12.2 Default function configuration

The second heuristic employed in feature function matching is the minimisation of function options. CAD API functions commonly use a multitude of optional parameters that are set at a default value. Certain CAD API use a hierarchical API data model that represents feature subvariants as individual child functions, or child object functions (e.g. AutoDesk Inventor). However the trend is for most commercial CAD programs to add optional parameters to extend the behaviour of the feature concept, as an example, the SolidWorks API `FeatureManager.FeatureFillet` function contain 25 independent variables which combine to produce 144 different variants of edge fillet feature. The strategy adopted is to match a minimum representation of a feature that reduces the number of parameter combinations available within the search space for a geometric match. This minimisation is justified by the customary practice of using the default values of a feature functions optional parameters to represent the most simple and most widely accepted representation of the feature geometry that the function controls.

8.12.3 Zero-valued parameter assumption

The third heuristic introduced is a minimisation of the sum of coded value states, so that candidate solutions with more zeros are evaluated as more fit. Consider that the first stage in matching function parameters to be the search for a parameter state that results in a minimal geometric difference between respective CAD models. There may be several different parameter states that generate identical geometric models. An additional optimisation goal is introduced to minimise the summed values of the coded parameter states, giving priority to solutions that contain more zeros, as zero is a common default value in CAD functions. The GA may then be formulated as follows

$$[x_1^S, \dots x_n^S, x_1^T, \dots, x_m^T] \longmapsto A\Delta_H(S(x_1^S, \dots x_n^S,), T(x_1^T, \dots, x_m^T)) + B\left(\sum_{i=n} x_i^S + \sum_{j=m} x_j^T\right)$$

where A and B are weights applied to the objective functions to give a summed fitness value. Details of the penalty values used are given in Chapter 9.1, A Genetic Algorithm configuration demonstrating CAD function matching.

This representation of soft constraints is similar to that commonly used on the so-called knapsack problem, where both the summed value and the summed volume of knapsack contents contribute to the fitness of a solution. In this instance, the soft constraints are accompanied by hard constraints, namely illegal parameter combinations that result in an error state when applied to their respective function. These combinations are recorded and candidate solutions that incorporate these combinations are awarded a low fitness value, as this optimisation search is to minimise geometric difference this results in a high number.

In the tests conducted in Chapter 9, a configuration using dual objective functions as described is compared for efficiency against a configuration with a single objective function that minimises the geometric distance between respective models.

Michalewicz defines the set of hard constraints limiting a search space as two disjoint subsets of feasible and infeasible subspaces, there are a number of approaches taken to incorporate this information into Genetic Algorithm searches (Michalewicz, 1995). Certain techniques remove infeasible candidates from GA populations, other ascribe a variable fitness penalty dependent on the level of transgression. In this particular application, the domain of illegal candidate solutions is hard to model. The CAD functions queried act as an oracle, either returning an instance of the desired geometric operation or registering an error. Therefore it is impractical to assign varying penalty values to illegal candidates.

In the trials run in Chapter 9, it was found that feasible candidate solutions comprised a relatively small subset of populations, while the coding of illegal candidates may have been close to an optimal solution. In this instance, a large fixed penalty value was awarded to illegal candidates. Illegal candidate solutions are integral to the automated mapping of feature functions, therefore these values are retained, allowing detection of similarly illegal solutions to be assigned a penalty fitness without having to undergo a computationally expensive fitness evaluation. Selecting a predetermined number of candidates ranked by fitness score allows identification of candidates suited to generating subsequent populations. This approach permits the few legal solutions to be included on every cycle. The selection of crossover is described in Chapter 9.1, A Genetic Algorithm configuration demonstrating CAD function matching, while the mutation parameters are described in Chapter 9.2, Elevated mutation constant.

Part.makeCone	radius1
	radius2
	height
Rhino.AddCone	arrBase.x
	arrBase.y
	arrBase.z
	arrHeight.x
	arrHeight.y
	arrHeight.z
	dblRadius

Table 8: genetic algorithm string values mapped to concatenated function parameters.

8.13

8.14 Individual parameter mapping

Once such a "ground state" is identified, then functional relationships between individual parameters may be detected via perturbation of individual parameters in one function against individual parameters in the second function to identify bijective mapping. If an additional base is added to the code alphabet used for parametric testing, individual parameter matches may be identified via the geometric similarity measure with a minimum number of search attempts. In the cone example presented below, a string with a solution of

$$\boxed{\mathrm{A}}\boxed{0}\boxed{\mathrm{A}} \mid \boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{\mathrm{A}}\boxed{\mathrm{A}}$$

generates identical cone models in both CAD programs with a maximum number of zero values. This string is based on the required values of the FreeCAD `Part.makeCone` function concatenated with the required parameter values of the RhinoScript `Rhino.AddCone` function where A is set to a non-zero value. The sequence of required parameters is mapped to their respective function names in Table 8, documentation excerpts from the relevant APIs are given at the end of the chapter (Developer.rhino3d.com, 2017; Riegel, 2017). These functions reflect the common practice of using zero as a default value.

Once a configuration is found that returns a model match, the search for individual matching parameters is a comparatively simple sequential search using an additional base, [0, A, B]. A parameter search may be as simple as a Gray code combinational search of the non-zero parameters, as demonstrated in the following example.

The sequence between $\boxed{\mathrm{A}}\boxed{0}\boxed{\mathrm{A}} \mid \boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{\mathrm{A}}\boxed{\mathrm{B}}$ and $\boxed{\mathrm{A}}\boxed{0}\boxed{\mathrm{B}} \mid \boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{\mathrm{A}}\boxed{\mathrm{B}}$ are omitted as there is no change to the three codes in the first part of the chromosome mapped to the first function parameters.

$\boxed{\mathrm{A}}\boxed{0}\boxed{\mathrm{B}} \quad \boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{\mathrm{A}}\boxed{\mathrm{B}}$:FAIL -no match for `Part.makeCone` height parameter

$\boxed{\mathrm{A}}\boxed{0}\boxed{\mathrm{B}} \quad \boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{\mathrm{B}}\boxed{\mathrm{A}}$:SUCCEED -height parameters matched

$\boxed{\mathrm{B}}\boxed{0}\boxed{\mathrm{A}} \quad \boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{\mathrm{A}}\boxed{\mathrm{B}}$:SUCCEED - base radius parameters matched

Identified parameter relationships are used to "mask" sequences of the concatenated string

$$\boxed{\mathrm{A}_1}\boxed{0}\boxed{\mathrm{A}_2} \mid \boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{\mathrm{A}_3}\boxed{\mathrm{A}_4}$$

8.14 Individual parameter mapping

which may be represented as,

$$\boxed{A_1}\boxed{A_2} \mid \boxed{A_3}\boxed{A_4}$$

This representation allows the function parameter search space to be partitioned over a smaller range of combinations. A Gray code is used in the example above simply to minimise the number of parameter changes between permutations, but any combinational method including a text-directed genetic algorithm may be used to perform a search for individual parameters over this reduced combinational space. If all individual parameters are identified, the zero values may then in turn be exchanged for suitable code alphabet values and subjected to a combinational search to identify related parameters common to both CAD API functions. After searches have been performed for individual parameters, it is then possible to attempt to match the optional parameters to these functions which have so far been excluded. In the example above no match would be found for the zero values in the second function in the chromosome until the optional parameters of the first function are tested as shown (the optional parameter description is listed in tabular form in Chapter 9.6, Published parameters of the functions used in tests).

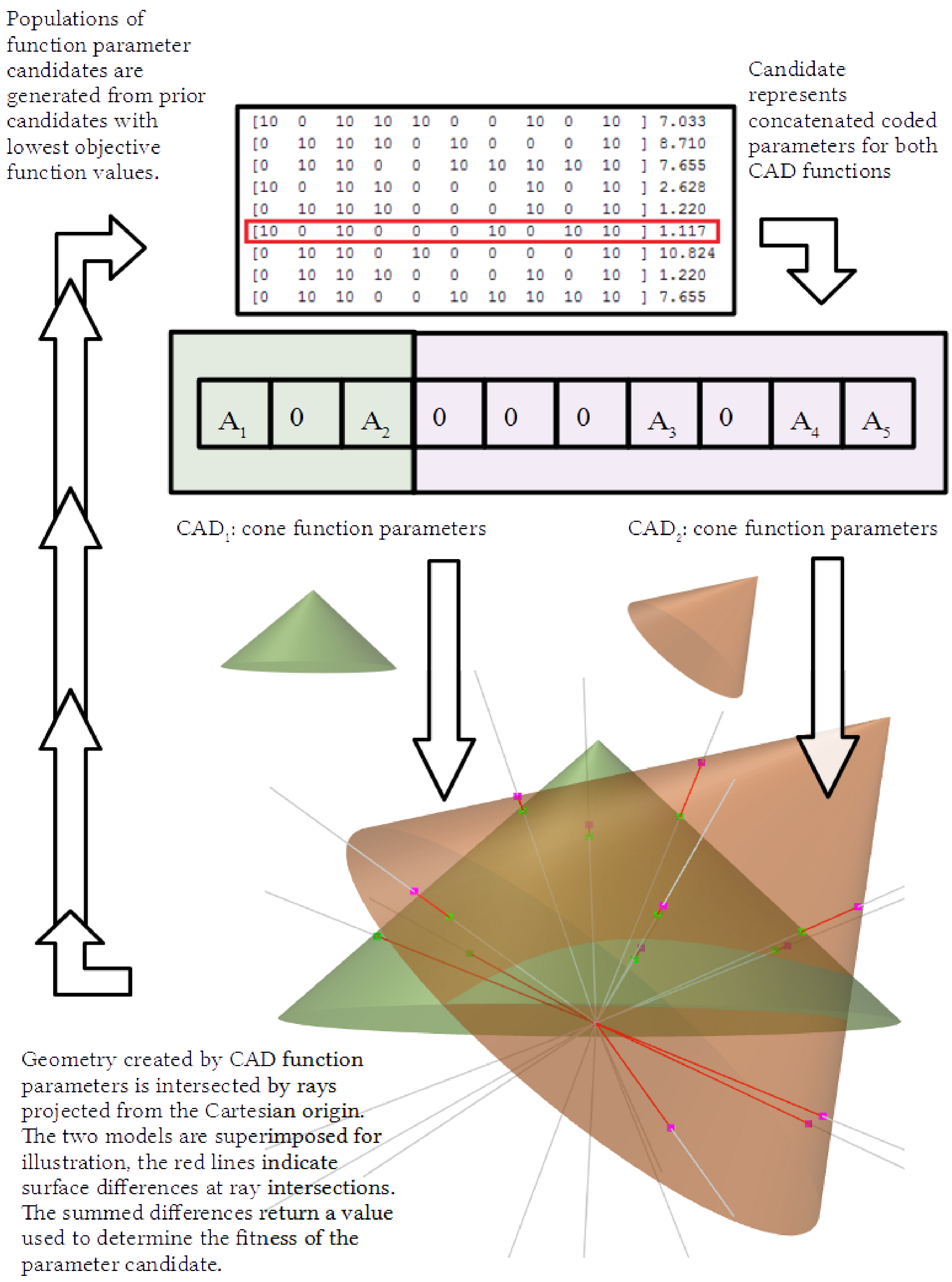
$$\boxed{A}\boxed{0}\boxed{A}\boxed{x}\boxed{y}\boxed{z} \mid \boxed{0}\boxed{0}\boxed{0}\boxed{x}\boxed{y}\boxed{z}\boxed{A}$$

The example here shows that a directed search may identify relationships more rapidly than an exhaustive search, but this method will find relatively simple matches of single parameters between functions, the efficiency of this approach is limited to the independence of model parameters. In cases where model parameters are not independent, the exclusion of prior mapped parameter discoveries may have to be abandoned to determine a potential mapping.

This chapter has described the common methods used to construct CAD model geometry. Construction of parametric feature-based geometry using a sequence of explicit and implicit constraint parameters is defined. Dynamic feature mapping is proposed as a solution to the challenge of detecting implicit constraints set by feature operations outside the scope of the feature under test. Testing function equivalence using dynamic mapping is a labour intensive task that requires automated function checking to be practical.

The task of automating a search for parameter combinations to detect a function mapping is described. A genetic algorithm parameter search method is described in detail where several heuristics are introduced to partition the combinational search space. This proposed method is tested in Chapter 9.

8.14 Individual parameter mapping



Populations of function parameter candidates are generated from prior candidates with lowest objective function values.

```
[10  0   10  10  10  0   0   10  0   10  ] 7.033
[0   10  10  10  0   10  0   0   0   10  ] 8.710
[0   10  10  0   0   10  10  10  10  10  ] 7.655
[10  0   10  10  0   0   0   10  0   10  ] 2.628
[0   10  10  10  0   0   0   10  0   10  ] 1.220
[10  0   10  0   0   0   10  0   10  10  ] 1.117
[0   10  10  0   10  0   0   0   0   10  ] 10.824
[0   10  10  10  0   0   0   10  0   10  ] 1.220
[0   10  10  0   0   10  10  10  10  10  ] 7.655
```

Candidate represents concatenated coded parameters for both CAD functions

A_1 | 0 | A_2 | 0 | 0 | 0 | A_3 | 0 | A_4 | A_5
CAD_1: cone function parameters
CAD_2: cone function parameters

Geometry created by CAD function parameters is intersected by rays projected from the Cartesian origin. The two models are superimposed for illustration, the red lines indicate surface differences at ray intersections. The summed differences return a value used to determine the fitness of the parameter candidate.

Figure 40: Overview of genetic algorithm process to generate CAD function parameter mapping

8.14 Individual parameter mapping

9 An automated search for function equivalence

Chapter 8 describes a method of performing a genetic algorithm based search for matching function parameters using a restricted set of parameter values and a geometry matching method similar to that described in Chapter 6. This algorithm configuration is described, and the results of tests on functions are presented. Several algorithm configurations are compared for relative efficiency.

9.1 A Genetic Algorithm configuration demonstrating CAD function matching

An implementation of a Genetic Algorithm scheme for matching CAD API functions was created using the DEAP evolutionary algorithm framework (Fortin & Rainville, 2012). Two CAD programs with relatively trouble free API access were selected, Rhino 5 via the Rhinoscript 5 Automation API and FreeCAD 0.17 via a native Python interface. The Python comtypes library is used to negotiate Automation access to the Rhinoscript interface, while a FreeCAD instance can be accessed as a native Python object (Heller *et al*, 2019; Riegel *et al*, 2017; Baer, 2011).

The algorithm used takes the form of the most simple genetic algorithm implementation (Bäck *et al*, 2018).

- An initial population of solution candidates is generated.
- These candidates are evaluated against a measure of fitness.
- While there is no overall optimum best solution candidate, or other termination criterion,
    - The most successful candidates are recombined
    - This pool of recombined candidates are subjected to random mutation.

- This pool is re-evaluated for fitness and the elite candidates are selected for the next generation.

Each of these steps merits further description. To recap, each chromosome string is the concatenation of parameters from two CAD API functions undergoing parameter mapping or functional similarity detection. The function parameter type, formatting and optionality are stored in a format that allows ready generation of legitimate variants. An initial population is seeded with randomly created individuals.

The fitness of each individual is expressed as a weight attribute. This may be a single weight that represents the similarity of the CAD models that the individual candidate represents, or it may include further objectives.

There are several variants of GA trialled. In a the multi-objective implementation (`GAP_Match07_multiRun.py`) each individual candidate has two objectives, the first is to minimise the geometric difference between CAD models that these candidates represent. The second objective to minimise is the value of numeric parameters, this heuristic sets available numeric parameters to zero with the aim of determining a viable geometric model configuration with a maximum number of default zero-valued parameters.

The so-called Knapsack problem is a simple representation of a multi-objective GA (see also Chapter 8.9). Selections from a set of items of differing size and value are to be fitted within the finite volume of a knapsack in such a way that the value of the contents is maximised, but the sum volume of contents is less than the knapsack capacity. The solutions to this problem may be represented as individual collections with a summed volume approaching an optimum value of a maximum summed worth. This can be represented as two weights attributes, one which maximises value, the second which approaches the maximum volume value.

Once an initial population is generated, the fitness of each of these individuals is tested. The chromosome is split into the segments that map to the individual parameters of both functions under test and the functions generate CAD geometry models in their respective CAD programs using the parameter data.

To test geometric equivalence, a basic implementation of the method detailed in Chapter 6 is used to sample points from the surfaces of both these geometries. A series of evenly distributed points on a unit sphere are generated using the method described by Deserno (Deserno, 2004). These points direct the angle of vectors emanating from the model space centroids. Where these vectors intersect the generated function models, the points are returned for comparison. Unlike the complex method described in Chapter 6.11, there is no search for feature registration points nor affine transformation between models. Consequently both models register a minimum deviation in point values once they occupy the same absolute Cartesian model space with respect to the sampling vectors. In the case of optimisation for zero-valued codes, the code values of the individual candidate are summed.

These tests form the basis of candidate fitness. In the frequent instances of parameter combinations that do not return any value, such as an instance of all parameters set at a zero value, the geometric fitness weight is set to a large penalty value (10^3). These values are recorded so that there is no requirement to re-evaluate these combinations. The evolutionary algorithm method used for all the trials generates an entirely new population from a recombination of relatively successful instances from the prior generation population (eaSimple). It was found that other models tested, such as versions that would add a fraction of unmodified elite individuals from the prior generation showed comparatively little effect on this class of partially continuous matching problem (`eaMuPlusLambda`). The `eaSimple` strategy is used as a baseline for all models. There are a range of established methods to cut and recombine individual candidates, in this instance the most simple technique is used, the *one point crossover*. As the name suggests, both parents are cut at the same random location and subsequently spliced to form a new individual (`deap.tools.cxOnePoint`).

This new generation of individuals is then subjected to mutation, where each code in each individual may change to another code value dependent on a probability derived from a psuedorandom value (`deap.tools.mutFlipBit`). This function is modified to operate with trinary and *n*-ary code bases.

This process continues until either the maximum iteration of population generations is reached, or if the fitness of any particular candidate instance is found to surpass a threshold. In this case, this equates to the average displacement between compared model intersection points falling below a threshold.

9.2 Elevated mutation constant

The directed guessing technique employed by a genetic algorithm is reliant on the results of the objective function decreasing as both CAD models approach a uniform configuration. A GA is more robust than other methods in tackling discontinuities in the output of the objective function, but without any discernable relationship between parameters it will perform no better than a brute force combinational search. All GA variants used relatively high probability values of mutation to compensate for a discontinuous objective function. Only a subset of parameter adjustment leads to a smoothly varying objective function, consequently additional randomness appears to lead to faster solutions.

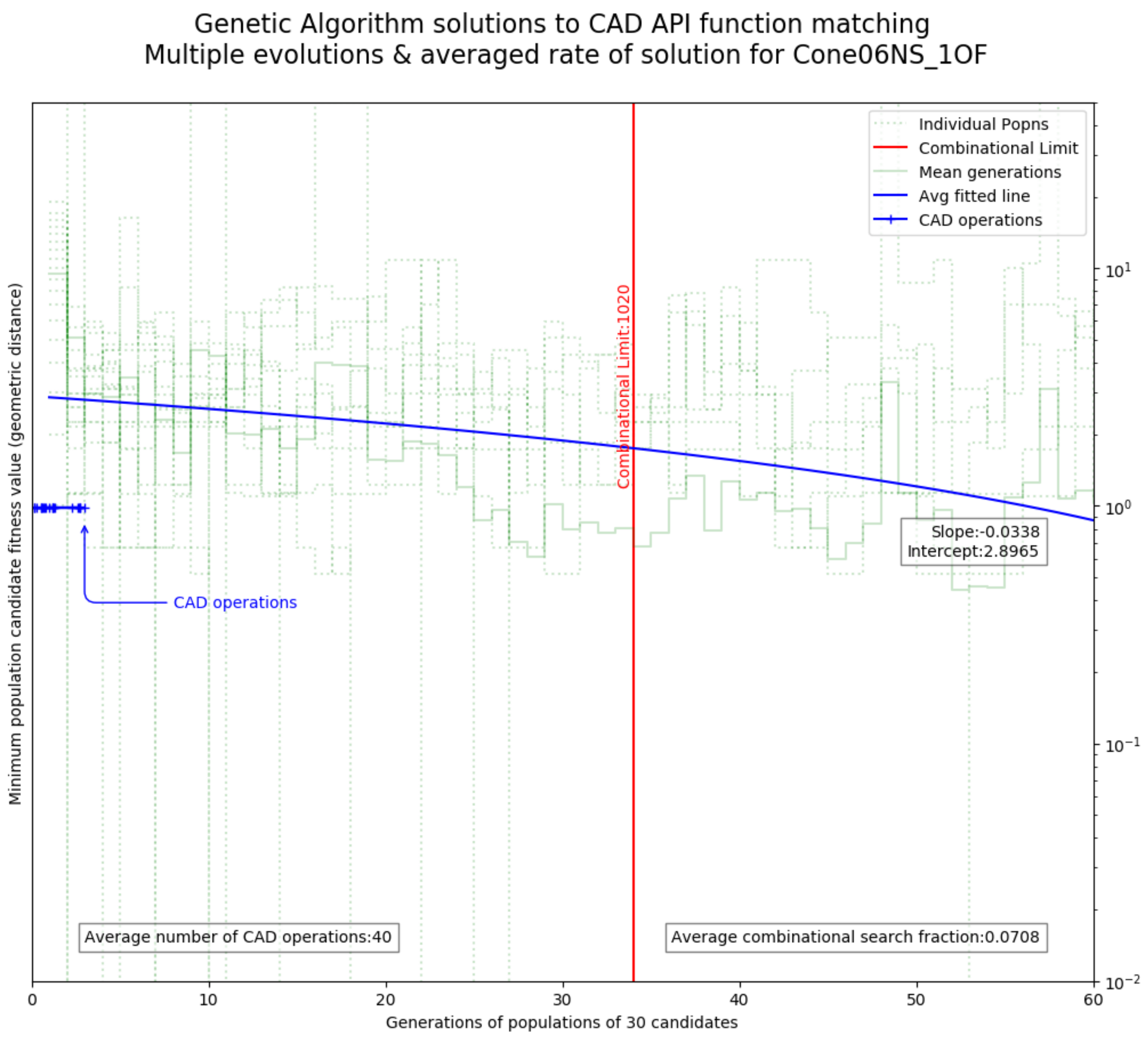


Figure 41: Single objective function GA performance without semantic match assistance.

9.2.1 Comparative testing of multi-objective and single objective fitness functions

The success of the GA technique may be measured against the exhaustive number of parameter combinations available for the base of the code set. In the following graphs these are represented as red vertical bars and can be considered as a limit of method efficiency. If any method takes, on average, fewer operations than this limit value, it may be considered to be more efficient than a brute force method. Two measures of GA operation are superimposed on each of the graphs (Figure 41 to Figure 46), the green lines indicate the value of the objective function to be minimised, the blue crosses the number of CAD operations. Both measures indicate the progress towards a solution, the inclusion of CAD operations is a measure of algorithm efficiency against a brute force solution.

The minimum geometric values attained within each generation are displayed for 20 different separate trials. The average value of these minima is plotted and these values taken to construct a linear least squares regression curve. From a comparison of the multi-objective fitness variant, (Figure 45) against that of the single objective fitness variant (Figure 42), biasing a solution towards zero code values appears to reduce the number of generations taken to reach a threshold model equivalence value. The caveat with this finding is that it is only tested on four representative API functions.

The number of generations of populations are not indicative of the number of times that the represented function parameters are used to test comparative CAD models before a solution is reached. The number of actual CAD comparative tests performed is marked by a blue cross at the number of evaluations at which a solution is found. It is observed that these computationally expensive tests involve a comparatively small number of tests compared to the exhaustive combinational test benchmark[4]. The number of CAD search operations required for the GA solution is divided by the number of combinations required for a brute force solution to the particular pair of CAD functions (note that there are several correct solutions). This figure is averaged over a series of 20 runs to give an estimate of advantage that the GA solution has over the combinational solution for each function pair. The representative GA configuration used will regularly regenerate combinations that have already been tested. In the implementation shown, each legitimate CAD operation outcome is recorded to avoid expensive duplicate geometric matching (see Table 9).

4 The combinational limit represents the required number of parameter code permutations to guarantee a solution is found, provided that one exists. Note that the highest probability of determining a correct solution using random, unrepeated guesses corresponds to half that figure. For simplicity, the maximum value is used.

	Cone Function Pair	Torus Function Pair
Single objective function	0.071	0.073
Dual objective function	0.062	0.050
Single objective function with semantic match assist.	0.084	0.062
Dual objective function with semantic match assist.	0.096	0.061

Table 9: averaged value of CAD match operations of GA variants as a fraction of required CAD match operations for combinational search.

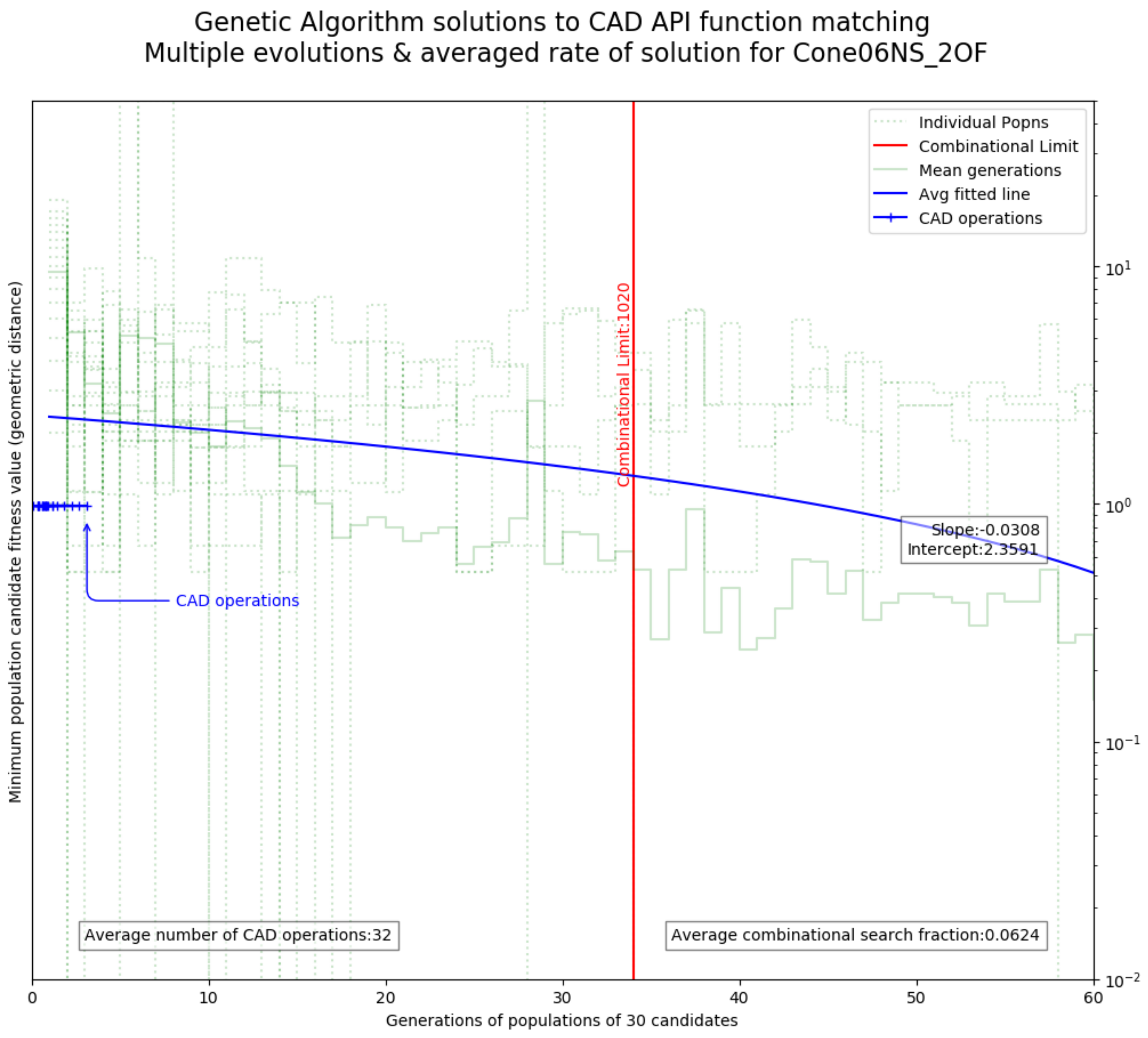


Figure 42: Multi-objective function GA performance using binary coding, no semantic match assistance

9.2 Elevated mutation constant

9.3 Semantic match assisted Genetic Algorithm trials

The final set of tests add semantic matching information determined from the labels given to the function parameters. A large matrix of word pair similarity matches is used to establish semantic match probabilities between words (the Word2vec model derived from the Wikipedia corpus, Chapter 4.5.6, Word Embedding and the word2vec similarity measure). Unlike the greedy method described in Chapter 5.2, the matching process simply finds the matching word pair with the highest value. The semantic similarity of parameter names is used to influence the creation of new chromosome individuals, or generate a new population during the initialisation of the genetic algorithm. For a pair of functions undergoing analysis, groups of semantically-related parameters may be identified. This corresponds to parameter labels that return a score of high semantic similarity when matched together.

A rule is introduced, that at least one member of an identified group within each function has at least one non-zero assigned value. In practice this imposes a requirement that there are at least two non-zero values per group, one of the parameters of the first function and one of the parameters of the second function. Groups that are unique to only one function are discarded. Random chromosomes are generated from the permissible set of parameter values and instances that do not conform to the semantic group rule as defined above are discarded.

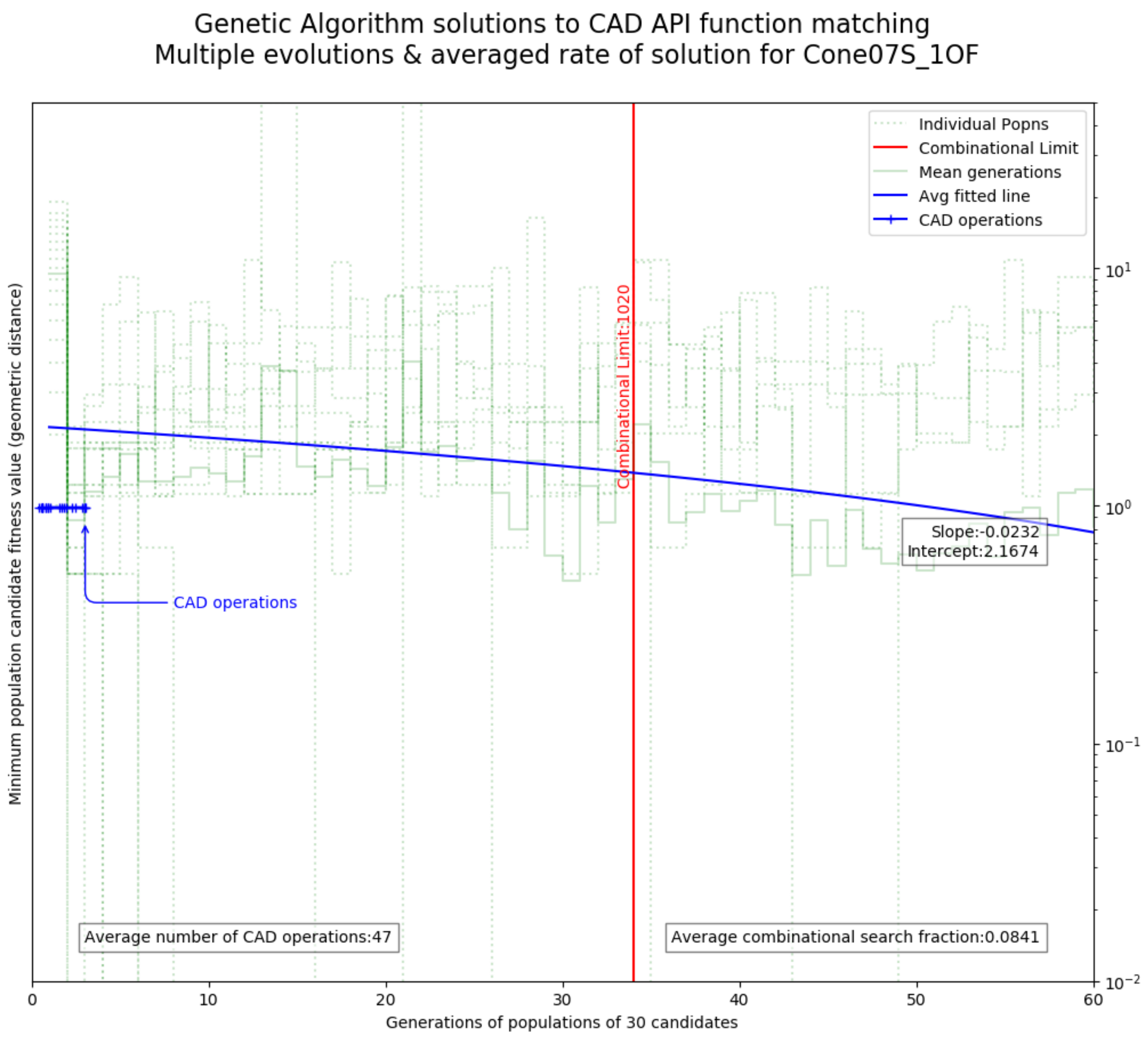


Figure 43: Single-objective function GA performance using binary coding, with semantic match assistance

Surprisingly, this additional information led to reduced GA efficiency. A consistently higher number of search attempts was recorded for both CAD function pairs tested. If the parameter names are compared with the semantic matching ratios, it seems that there is ambiguity arising from inadequate discrimination between categories. For example, three of the four functions tested had more than one “radius” parameter. A semantic matching would determine a relationship between the identical syntax, but not yield any finer discrimination. The limited range of functions tested restricts the scope of conclusions that can be drawn.

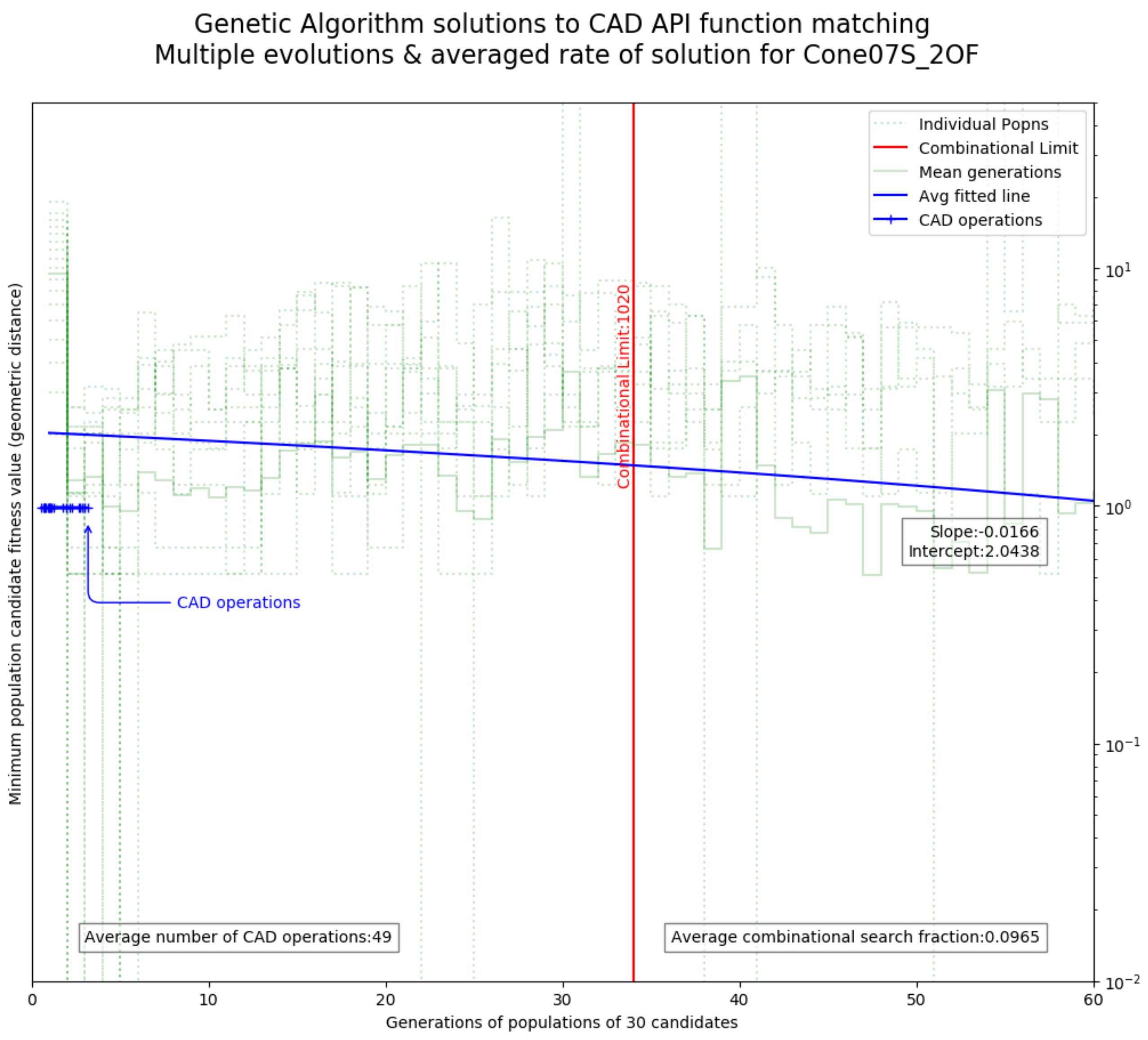


Figure 44: Multi-objective function GA performance using binary coding, with semantic match assistance

9.4 Tests extending beyond psuedo-binary coding

The example presented here is a simple known match. While there is some parameter mismatch between the two API functions, both functions will produce valid models with a minimum of two code values, in this case [0, 10].

A function of similar apparent complexity, the RhinoScript `AddTorus` command, is an example of a function that must take three codes to generate minimum valid output. This function places a constraint on parameters such that the radius of the torus about its centroid must be larger than the sectional radius of the torus. This requirement demands a minimum of three codes, say [0, 10, 20]. However this requirement may only be determined after an exhaustive search of the parameters, in this case 2^3 combinations. See Figure 45 and Figure 46.

If the number of independent parameters required to create an identical geometry model are not known in advance, this would indicate that coding the problem as a genetic algorithm would require a minimum of $(k-1)^n$ operations, where k is the minimum number of parameter states required to find a solution to two functions with n independent parameters between them.

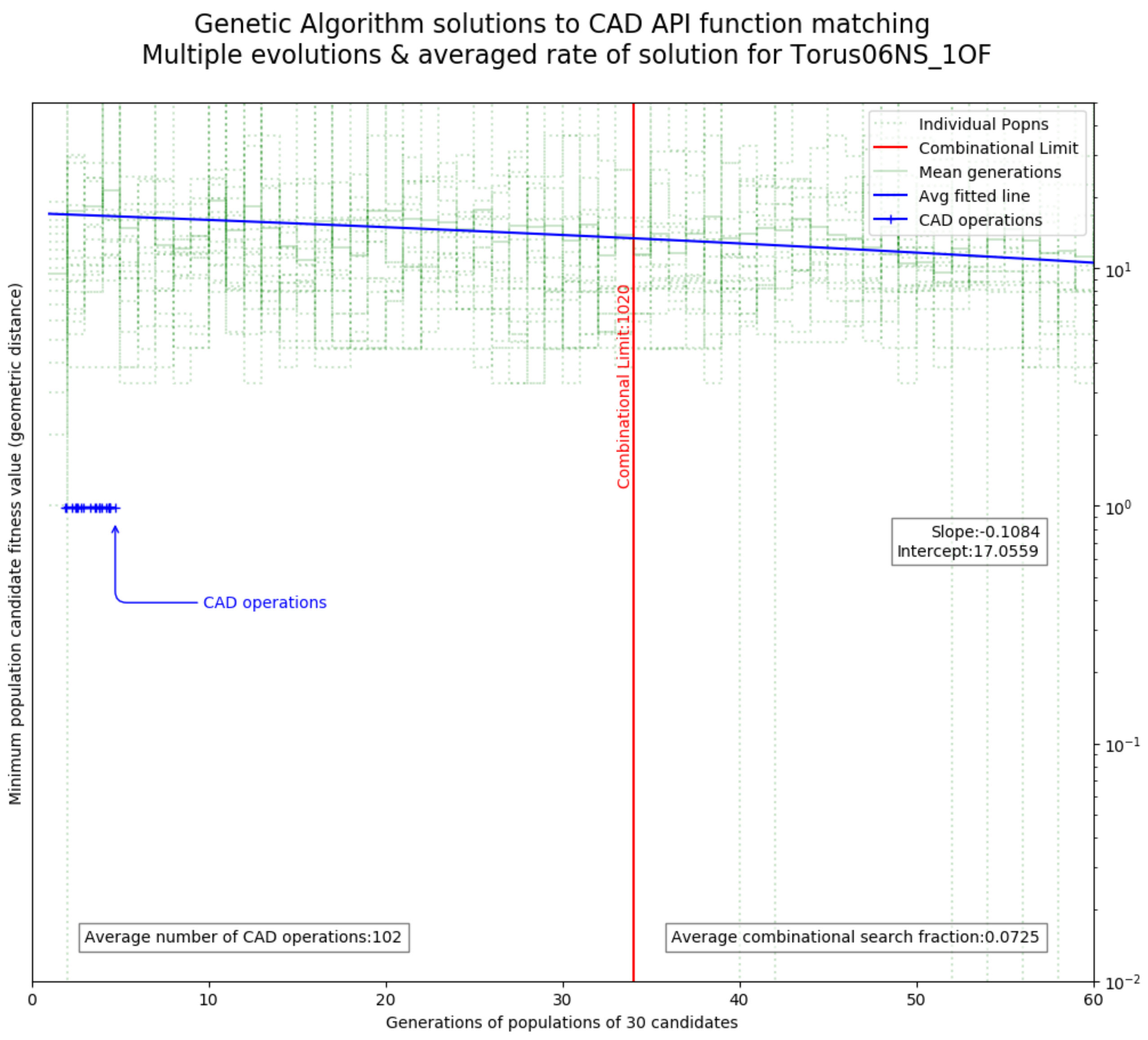


Figure 45: Single objective function GA performance using trinary coding, no semantic match assistance

9.4 Tests extending beyond psuedo-binary coding

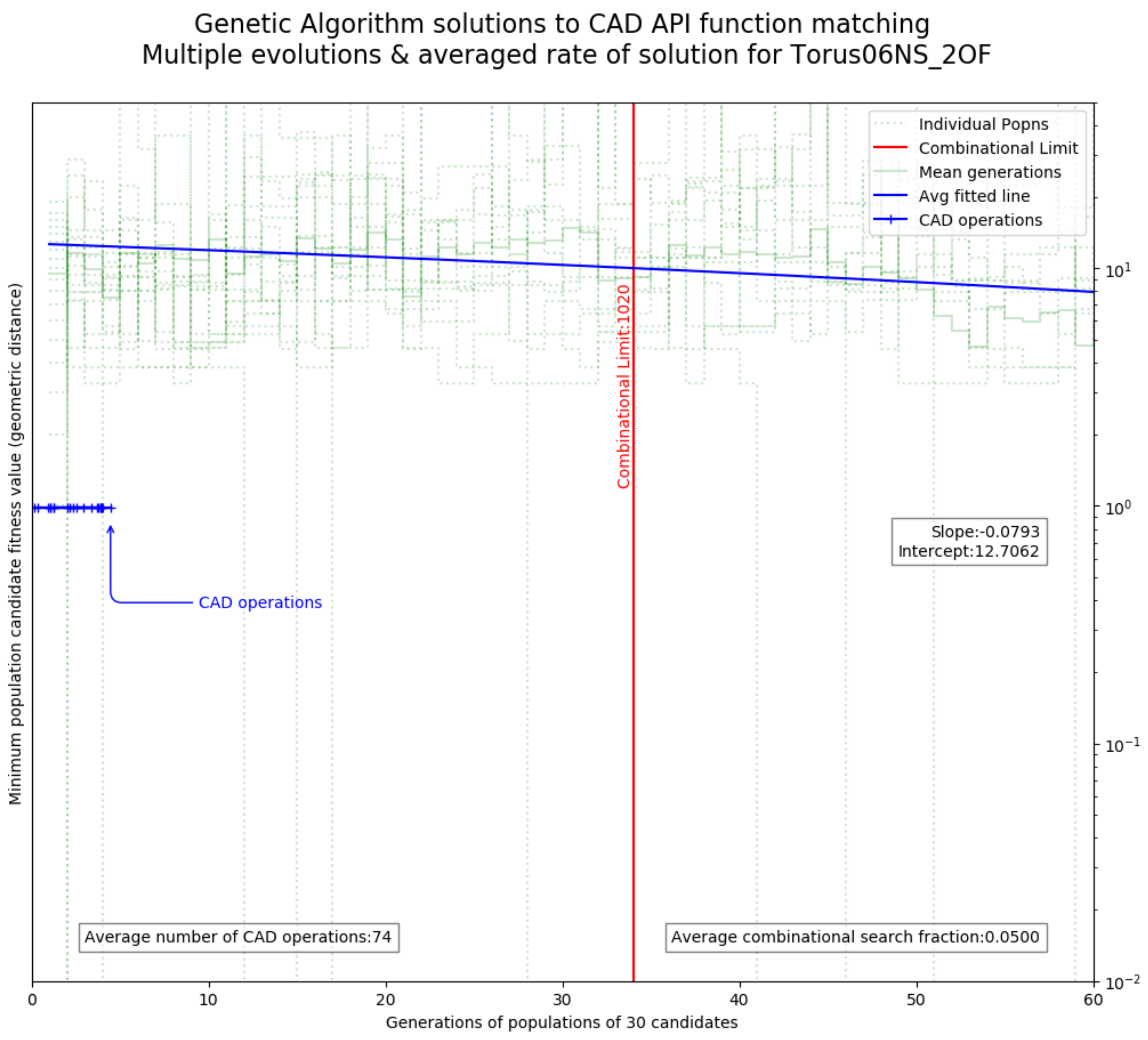


Figure 46: Multi-objective function GA performance using trinary coding, no semantic match assistance

9.5
Results summary

This chapter demonstrates the viability of stochastic local search techniques to map the functionality of parameters. In the illustrative examples a geometry comparison indicates whether model parameters create an identical model within Cartesian model space.

This automated search uses a genetic algorithm with an objective function based on the absolute geometric difference between CAD models generated by each function parameter configuration. This geometric difference is a summation of numerical differences between model surface boundary points sampled at identical locations on both models. The sampled points correspond to intersections with vectors radiating from each model centroid.

This approach requires that parameters with an X, Y, and Z axis component is solved for each axis. For instance, each vector value contains three independent Cartesian parameters. The method of comparing CAD model surface boundaries described in 6 is independent of model orientation, location and scale. Employing this method as an objective function is more complex, but reduces the number of independent variables within a search for model parity. For the cone example, this represents a reduction from ten to six independent parameters

This chapter demonstrates two pairs of CAD functions that are solved using a restricted set of parameter values is adequate for determining a correspondence between CAD functions. On average, the simple genetic algorithm used returns a solution in under a tenth of all possible permutations used in an exhaustive search (see Table 9). Part of the reason for this efficiency is that illegal CAD states are recorded for each function alongside solutions that return a model. This search method is found to be improved by an additional search objective, the minimisation of search values.

Finally, the method of generating candidate solutions is modified to include semantic relationships between individual function parameter text labels. In the function pairs tested this resulted in a slight degradation of performance. Additional semantic information may lead to faster solutions between parameters with a larger number of parameters, but this is not covered in these exploratory tests.

In this chapter, a stochastic local search method has been used to determine a parameter state required for model parity. In Chapter 8, the method is presented as a means to map individual parameters between functions. The same technique may be used to determine whether unknown functions share a capacity to create similar geometrical models.

9.6 Published parameters of the functions used in tests

```
Part.makeCone (radius1,  radius2,  height,  [pnt,  dir,  angle])
Description: Makes a cone with given radii and height. By default pnt is
Vector(0,0,0), dir is Vector(0,0,1) and angle is 360
```

Parameters

Name	**Optional**	**Type**	**Description**
radius1	Required	Number	Radius of the arc or circle defining the lower face
radius2	Required	Number	Radius of the arc or circle defining the upper face
height	Required	Number	The height of the Part Cone
pnt	Optional	Number	By default point is Vector(0,0,0).
dir	Optional	Number	By default dir is Vector(0,0,1).
angle	Optional	Number	The default 360 creates circular faces, a lower value will create a portion of a cone as defined by upper and lower faces each with edges defined by an arc of the number of degrees and two radii.

Returns

Object	The created shape object reference.
N/A	If not successful, or on error.

```
Rhino.AddCone (arrBase, arrHeight, dblRadius [, blnCap])

Rhino.AddCone (arrPlane, dblHeight, dblRadius [, blnCap])
```

Parameters

Name	**Optional**	**Type**	**Description**
arrBase	Required	Array	The 3-D origin point of the cone.
arrPlane	Required	Array	The cone's base plane. The apex of cone is at plane's origin and the axis of the cone is plane's Z axis.
arrHeight	Required	Array	The 3-D height point of the cone. The height point defines the height and direction of the cone.
dblHeight	Required	Number	The height of the cone. If *arrPlane* is specified, then the center of the *arrPlane* is height * the plane's Z axis.
dblRadius	Required	Number	The radius at the base of the cone. Note, tan(cone_angle) = *dblRadius / dblHeight*.
blnCap	Optional	Boolean	Cap the base of the cone. The default is to cap the cone (True).

Returns

String	The identifier of the new object if successful.

Null	If not successful, or on error.

```
Part.makeTorus(radius1,radius2,[pnt,dir,angle1,angle2,angle]).
By default pnt=Vector(0,0,0),dir=Vector(0,0,1),angle1=0,angle2=360 and
angle=360.
```

Parameters

Name	**Optional**	**Type**	**Description**
radius1	Required	Number	Radius of the circle around which the disc circulate.
radius2	Required	Number	Radius of the disc defining the form of the torus.
pnt	Optional	Number	The center of torus. By default pnt is Vector(0,0,0).
dir	Optional	Number	By default dir is Vector(0,0,1).
angle1	Optional	Number	1st angle to cut / define the disc of the torus
angle2	Optional	Number	2nd angle to cut / define the disc of the torus
angle3	Optional	Number	3rd angle to define the circumference of the torus

Returns

Object	The created shape object reference.
N/A	If not successful, or on error.

```
Rhino.AddTorus(arrBase, dblMajorRadius, dblMinorRadius[, arrDirection])
Rhino.AddTorus(arrPlane, dblMajorRadius, dblMinorRadius)
```

Parameters

Name	**Optional**	**Type**	**Description**
arrBase	Required	Array	The 3-D origin point of the torus.
arrPlane	Required	Array	The base plane of the torus.
dblMajorRadius	Required	Number	The major radius of the torus.The major radius must be larger than the minor radius.
dblMinorRadius	Required	Number	The minor radius of the torus.The minor radius must be greater than zero.
arrDirection	Optional	Array	A point that defines the direction of the torus.If omitted, a torus that is parallel to the world XY plane is created.

Returns

String	The identifier of the new object if successful.
Null	If not successful, or on error.

10 Conclusions and future directions

Computer Aided Engineering has evolved from simple drafting programs to the nexus of design, simulation and production information. There has been a proliferation of commercial offerings that have improved the capture, specification and transfer of product information, however the profusion of vendor systems has not coalesced around a de facto representation of product data. The conceptual vocabulary of vendor design elements have no universally agreed semantics or architecture.

This thesis outlines the efforts to agree, impose and deduce interoperability across these different varieties of vendor software. When design data was little more than fixed model boundary geometry, it was comparatively easy to standardise the formats that captured this data. The introduction of parametric design features led to significantly more complex interpretations of user-specified parameters that frustrate standardisation efforts. Transfer of data between CAE software must now include more design concepts, or "design intent", than the boundary surfaces of designed objects, but there are no common standards to permit this transfer.

There have been significant efforts to formalise the semantic definitions of the concepts and terminology used within parametric feature CAD programs. A formal ontology capturing the specification of features and their constraints for several programs should allow machine checking for equivalence between CAD features. This in turn promises to facilitate the reconstruction of models from features that have been mapped between CAD programs. The "top-down" approach of determining a universal ontology that might be subsequently endorsed by vendors has not led to a widely-adopted common ontology. The "bottom-up" approach is a pragmatic effort to discover existing relationships between CAD concepts and capture these mappings within an ontology. This concept has been extended from a static mapping between CAD feature libraries, to a "dynamic mapping" that determine feature equivalence for each instance within a model based on formal type checking of a shared ontological description.

Dynamic mapping methods allows the capture of decisions applied during sequential feature operations that create a model, where a static mapping method might return an ambiguous interpretation of these sequential feature model decisions accumulated through design choices, termed “implicit constraints”. Methods that use this dynamic feature mapping approach require a that either a large body of potential mapping data exists, or that a large number of candidate features are trialled to determine an accurate map. If an ontology is used to search or validate a candidate function mapping, this will still require that the ontology holds sufficient detail on CAD feature libraries to allow accurate machine reasoning. These requirements limit the practicality of this exploratory technique.

An automated means to test CAD functions for equivalence under explicit and implicit parameter configurations dispenses with the costly requirements of expert feature checking. To do so requires an efficient means to search for feature similarity and mapping validation. This thesis has developed techniques for both retrieval and validation of candidates for CAD feature function mapping between heterogeneous CAD programs. It is shown that there is a tractable approach to automated testing and matching of CAD geometric functions between heterogeneous programs. This research has been exploratory, testing the viability of different methods that reduce the combinational search for matches.

10.1 Fulfilment of research objectives

Three different research strands are developed to address the research question, namely,

1. Semantic matching of API text, existing and novel matching methods are tested on the short texts accompanying CAD function descriptions.
2. An affine invariant geometry matching method is described and tested on a range of benchmark CAD shapes.
3. A search for matching function parameters is coded as an evolutionary algorithm.

These three strands are assessed with respect to the research objectives given in (REF). The short descriptions of the research objectives are repeated below.

Objective III: *determine the applicability of semantic matching methods suited to identification of CAD software API function matches.* (Chapter 1.2)

There are hybrid techniques that are adapted to identifying semantic similarity between the short, terse texts associated with CAD API libraries. Only the Doc2vec method is adapted to ranking a text against the comparatively large number of texts found in a CAD API. A greedy method is developed to combine semantic similarity scores of individual word pairs. This method is tested using a variety of word pair measures on two sets of known function matches from commercial CAD API documentation. None of the text similarity techniques tested demonstrate sufficient accuracy to merit a stand-alone CAD function matching technique. However all of the short text similarity tests reduced the function search space to a third of the API sets indicating that semantic text matching has utility as part of a hybrid matching technique.

The methods defined and tested in Chapters 4 and 5 are all tested on the same test data allowing a measure of comparison. While it is relatively easy to predict the comparatively poor performance of methods designed for larger documents, such as LSA and TF-IDF, there is a surprising disparity between the other methods tested. This is further compounded by large performance variations between similar methods using different parameter settings, notably the importance of word order in the doc2vec method. The initial objective is fulfilled if one were to consider that there is strong evidence from test results that semantic matching may significantly reduce the search space of automated API matching. However this variability suggests that further performance optimisation may be attained from a more judicious selection of methods and parameters.

Objective I: *devise and test an algorithm capable of identifying two equivalent geometrical surfaces independent of scaling, rotation and translation while independent of vendor specific CAD programs.* (Chapter 1.2)

The geometric similarity technique introduced is distinct from other CAD model matching techniques by virtue of using registration feature regions. While registration regions such as areas of high curvature are not a novel technique within the broad field of 3D object matching and registration, they do not appear within CAD model matching, which tends to use graph based representations of surface face connectivity (AAG, MAAG, see Appendix 12, CAD graph methods). A set of registration feature types are defined that can serve as points to allow a closed-form calculation of a rotation matrix and translation between shapes undergoing comparison, while also serving as a distinctive model signature to allow rapid similarity searching within a model database. The geometric matching algorithm validates model geometry similarity using a transform of registration points between two models alongside a transform of random sampled points. This approach returns a correlation between model similarity confidence and the number of sampled points tested.

A robust method must identify registration feature points on CAD models without recourse to neutral formats or interface code reliant on API functions that identify these regions. This is accomplished with a hill-climb algorithm that is solely dependent on points returned from the intersection of a ray with the model boundary surface. The efficiency of this method is tested against the Drexel CAD benchmark library of primitive shapes, returning sufficiently high scores to be considered applicable for a geometric surface similarity and verification method suited to CAD feature mapping.

This method of determining feature point signatures for models, then using a multi-stage process for testing similarity provides an accuracy that outperforms other methods on the same benchmark data set. This accuracy is proportional to the complexity of the compared models and the number of surface samples taken to verify equivalence. This approach fulfils the criteria of the first research objective, namely to devise a method to determine the similarity of two CAD model surfaces independent of affine transformation.

Objective II: *devise and test a method capable of identifying the range of geometrical operations normally found within representative commercial CAD programs.* (Chapter 4)

The second research objective is addressed in Chapter 6.25 , 6.26 and 7.5. It is found that the set of feature identification points (Table 6.25) when used in conjunction with the multi-stage geometry matching process (Chapter 6.11, A progressive search refinement strategy) will uniquely identify each shape within the Drexel benchmark CAD library of primitive shapes and variants (CAD Models Dataset. 2004).

The original second objective is resolved using several methods in sequence. It can be seen that no individual method used is suited to an accurate comparison, for example the method that compares point signatures will return a false positive result for instances of elongated spheres and torus shapes (Chapter 7.4, Instances of registration feature mismatch). This hybrid approach is justifiable within a Design Research methodology where an artefact may include exceptions to a rule. The proposed hill-walking feature detection algorithm is an example, where shapes that cannot be uniquely defined by corners such as cylinders and spheres require the set of identifying features to be extended to allow their unique identification.

Once an object has more than the minimum number of feature points required to allow a rotation transform, there is an opportunity to change to a selective set of features that have a high probability of detection using the hill-walking method (Chapter 6.24). This strategy encompasses models of a higher geometric complexity than the primitive models used for the tests in Chapter 7.1. It may be observed that a methodology that permits a hybrid of partial solutions to address a problem may return a “good-enough” solution to perform subsequent analysis, such as as objective function as used for GA parameter matching in Chapter 9.1. Simon refers to this concept as “satisficing” (Simon, 1956).

Objective IV: *demonstrate how a measure of surface boundary geometry similarity may be used to map features between heterogeneous CAD programs, where features are defined by interface library routines.* (Chapter 1.2)

The fourth research objective specifies a demonstration of how a geometry comparison test may be used to search for matching CAD feature functions. This is pursued in the third strand of the thesis that uses an evolutionary algorithm as a local search method. This third section explores heuristics to partition the combinational search space associated with CAD feature mapping using a geometric verification technique. CAD API functions typically feature large numbers of parameters. It is unlikely that all parameters have a bijective mapping between functions. Once a parameter configuration is found for both functions that returns an equivalent geometrical model output, then subsequently identifying individual parameter mapping uses a reduced combinational space. Functions may be matched using an exhaustive search for parameters, but this is liable to be computationally intensive without a reduction of parameter values to a minimum set of possible states. A Genetic Algorithm is used to search for a set of function parameter values that return an equivalent geometric output. The geometric difference between models is calculated using a simplified variant of the described geometry matching schema and set as the objective function. Simultaneous tests between functions within different CAD programs returns a solution within a fraction of the CAD operations required by an exhaustive search. A multi-objective Genetic Algorithm variant using geometric distance and a minimised parameter value is found to arrive at a solution in fewer operations. Semantic similarity information is added to the GA tests but is found to increase the average time to arrive at a solution.

10.2 Contributions to knowledge

Several contributions to knowledge are made during the course of this research, these are summarised here for reference.

- A novel greedy algorithm for fast short phrase matching is developed and tested against a representative range of contemporary text matching methods. Unlike other short text semantic comparison method, this greedy approach permits comparison over large collections of phrases within a useful computational time, see Chapter 5.3.

- Several instances of commercial CAD API library documentation are converted to short texts and used to compare the efficiency of a broad range of semantic matching methods for determining function similarity, see Chapter 5.4.
- A novel method is developed and tested to determine similarity between surface geometry models independent of position, orientation and scaling. This method is found to perform significantly better over a benchmark library of CAD shapes than existing methods. This method is described in Chapter 6.
- A “hill-climbing” nearest neighbour search method that allows detection of surface boundary geometry feature points relative to a model centroid. This method is modified to describe a set of simple and unique feature point classifications. See Chapter 6.24.
- A novel helical feature point ordering algorithm allows the use of a Kabsch algorithm to solve the optimal rotation matrix between two geometry surfaces represented by registration feature points. Detailed in Chapter 6.17.
- A robust local search method is used to demonstrate an efficient function parameter matching method using a minimal parameter representation This approach is shown to perform significantly better than a combinational search, see Chapter 9.2.1.

10.3 Program requirements for a production environment

This research is an exploration of the feasibility of automated feature mapping. The implementations of the algorithms described are not optimised for speed nor efficiency, several improvements appear evident, listed at the end of Chapter 7. This thesis probes the feasibility of several novel or repurposed techniques to address the research question of whether a CAD API feature mapping could be automated. This in turn would allow a translation between CAD models that captures a greater proportion of the design information embodied within CAD models (Chapter 3.3).

The code is scripted in Python for rapid development, there is a potential for performance gains if refactored in a faster language such as C++ and subsequently profiled for a production environment. There are several approaches that may reduce the number of samples to determine geometric similarity, in turn reducing the number of calls made to the CAD API via the COM interface. Details of these potential optimisations are outlined in Chapter 7.7. The research is presented as three separate experiments over a representative data set, but there is little detail of what an efficient production system might require, this may be outlined as follows. Consider that there are two separate requirements,

- a program that takes a CAD model in source CAD program format and replaces it with a functionally equivalent model in a target CAD program format,
- a program that searches for functions within heterogeneous CAD programs that test positive for geometrical equivalence.

These two distinct operations may be described in more detail. In the case of the first task, each CAD model to be translated would require that the sequential operations used for construction are retained in the form of native API function calls. If access to the native CAD system is not available, the intermediate model geometry produced at each step of a sequential process would be needed. Assuming that a CAD mapping exists with an equivalent function operation in a target CAD program, each stage of a model may be mapped to its target counterpart and tested against the geometric intermediate form.

The task of determining mapping between heterogeneous CAD is essentially a search problem that may be conducted between any agent with access to a copy of the source or target CAD programs. Each CAD program requires a minimal interface program that directs the projection of vectors through CAD model surfaces and returns the absolute Cartesian points of intersections with these vectors. This search may be

10.3 Program requirements for a production environment

conducted between multiple agents who share the results of tests, which may then contribute to a search space database of completed tests.

10.4 Potential stakeholders and relevant groups

For a company trying to compete in a market of shorter product life-cycles, the integration of heterogeneous systems that make up Product Lifecycle Management is crucial. Off-the-shelf PLM products require expensive adaptation to an existing manufacturer development cycle, including an adoption cost. Devising a PLM solution around existing company software systems is prohibitively expensive and generally only feasible for the largest of companies (Chapter 2.3). Large manufacturers and their value chains place demands of interoperability on their CAE software products, and of the software products used by their value chain.
There are significant switching costs changing from one CAD system and ancillary software to another, these costs may be compounded by frequent software product upgrades, where support for design stored within legacy software becomes an additional expense. These costs present a market barrier to small to medium enterprises who might wish to supply several top-tier manufacturers demanding different CAD systems. The appearance of methods to reduce labour and expense in developing API mapping for improved data translation promises to lower additional costs arising from inefficient data transfer.

It is worthwhile briefly speculating on the groups that might adopt and contribute to such a research program. The research presented here immediately lends itself to furthering open-source CAE projects, such as the FreeCAD CAD program (Riegel *et al*, 2019), where the utility of open-source software is increased by the ability to transfer data to and from commercial equivalents. This facility is equally attractive to CAD vendors who may wish to reduce the labour required to add a translation functionality to their software.

Automated function-mapping techniques also lend themselves to public-funded efforts to preserve and coordinate data between industries. As an example, the US Department of Defence invests significant resources in standardisation efforts (Rachuri *et al*, 2006). More recently the DoD has promoted a Model Based Engineering approach (Duncan, 2019), placing further demands on commercial CAD and PLM software interoperability.

Large to medium companies that rely on efficient communication between different departments and value chains have a financial incentive to develop cheaper automated methods to enhance interoperability between their preferred choice of business and engineering software, these arguments are detailed further in Chapter 2.3. This technology will also appeal to companies with a business based around transfer of models between representations in commercial systems.

10.5 Observations and Future Directions

The hybrid method proposed for identifying and mapping API functions between heterogeneous CAD programs uses techniques that are broadly applicable to mapping functions between API libraries. The semantic similarity measures used for short descriptive text requires no adaptation to function for all API documentation. The genetic algorithm used to find sets of parameters producing a matching function output requires an objective function that can determine a measure of similarity between the output of functions undergoing comparative testing. As an example, the API of two image-processing programs undergoing comparison might use screen capture bitmaps to create an objective function for minimisation. The bitmaps of functions that produce an identical screen image may be compared to yield a numerical measure of similarity.

The development of a feature function matching technique to map similarities between programs presents a novel perspective on the creation of associated semantic ontologies. An automated mapping cannot ascribe semantic meaning to discovered matches, yet it is possible to determine functional equivalence of parameters according to a geometric measure. In the event that one CAD API parameter has a purely geometric semantic definition, can mapping this parameter to another CAD API to another yield a semantic match? A feature function ontology will typically use a reasoner to infer relationships, however there is an unexplored possibility of constructing geometric semantic relationships via a combination of mapping and inference.

10.5 Observations and Future Directions

11 Bibliography

3ds.com. 2019. CATIA™ 3DEXPERIENCE® - Dassault Systèmes® 3D Software. [online] Available at: https://www.3ds.com/products-services/catia/ [Accessed 22 Mar. 2019].

10 303-21:2016, I. (2016). ISO 10 303-21:2016. [online] ISO. Available at: https://www.iso.org/standard/63 141.html [Accessed 9 Mar. 2019].

Abdul-Ghafour, S., Ghodous, P., Shariat, B. and Perna, E., 2007, November. A common design-features ontology for product data semantics interoperability. In Proceedings of the IEEE/WIC/ACM international conference on web intelligence(pp. 443-446). IEEE Computer Society.

Abdul-Ghafour, S., Ghodous, P., Shariat, B., Perna, E., and Khosrowshahi, F., 2014, "Semantic Interoperability of Knowledge in Feature-Based CAD Models," Comput.-Aided Des., 56(11), pp. 45–57.

Aerospace Industry Guidelines For Implementing Interoperability Standards For Engineering Data. (2013). 1st ed. [ebook] Aerospace Industries Association. Available at: http://www.aia-aerospace.org/wp-content/uploads/2016/05/AIA_EDIG_Guidebook.pdf [Accessed 10 Apr. 2018].

Aguilera, U., Abaitua, J., Diaz, J., Bujan, D. and de Ipina, D.L., 2007, September. A semantic matching algorithm for discovery in UDDI. In International Conference on Semantic Computing (ICSC 2007) (pp. 751-758). IEEE.

Ahmed, F., and Han, S., 2015, "Interoperability of Product and Manufacturing Information Using Ontology," Concurrent Eng., 23(3), pp. 265–278.

Akinci, B. and Lipman, R.R., 2010. Semiha Kiziltas, Fernanda Leite. CAD and GIS Integration, p.73.

Alted, F. and Fernández-Alonso, M., 2003. PyTables: processing and analyzing extremely large amounts of data in Python. PyCon2003. April, pp.1-9.

Altidor, J., Wileden, J., Wang, Y., Hanayneh, L. and Wang, Y., 2009, January. Analyzing and implementing a feature mapping approach to CAD system interoperability. In ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. 695-707). American Society of Mechanical Engineers.

Altszyler, E., Sigman, M. and Slezak, D.F., 2017. Corpus specificity in LSA and Word2vec: the role of out-of-domain documents. arXiv preprint arXiv:1712.10054.

Anderson, B. and Ansaldi, S., 1998. ENGEN Data Model: a neutral model to capture design intent. PROLAMAT98.

Ankerst, M., Kastenmüller, G., Kriegel, H.P. and Seidl, T., 1999, July. 3D shape histograms for similarity search and classification in spatial databases. In International Symposium on Spatial Databases (pp. 207-226). Springer, Berlin, Heidelberg.

Ansary, T.F., Daoudi, M. and Vandeborre, J.P., 2007. A bayesian 3-d search engine using adaptive views clustering. IEEE Transactions on Multimedia, 9(1), pp.78-88.

Ansys.com. 2019. Engineering Simulation & 3D Design Software | ANSYS. [online] Available at: https://www.ansys.com/en-gb/ [Accessed 22 Mar. 2019].

Anumba, C.J., Siemieniuch, C.E. and Sinclair, M.A., 2000. Supply chain implications of concurrent engineering. International Journal of Physical Distribution & Logistics Management, 30(7/8), pp.566-597.

Assouroko, I., Ducellier, G., Belkadim, F., Eynard, B. and Boutinaud, P., 2010. Improvement of engineering design and numerical simulation data exchange based on requirements deployment: a conceptual framework. In Proceedings of the 7th International Product Lifecycle Management Conference, Bremen.

Atkinson, P.E., 2004, June. Strengths and Weaknesses of SME Statistics Systems: The Users' Perspective'OECD Presentation of Identified Key Issues. In *Special Workshop on'SME Statistics: Towards a More Systematic Statistical Measurement of SME Behaviour', 2nd OECD Conference of Ministers responsible for SMEs, Istanbul* (pp. 3-5).

Audette, M.A., Ferrie, F.P. and Peters, T.M., 2000. An algorithmic overview of surface registration techniques for medical imaging. Medical image analysis, 4(3), pp.201-217.

Autodesk.co.uk. (2018). 360-cloud. [online] Available at: https://www.autodesk.co.uk/360-cloud [Accessed 22 Jun. 2018].

Autodesk.com. (1997). General DXF File Structure [DXF - DXF Reference]. [online] Available at: https://www.autodesk.com/techpubs/autocad/acad2000/dxf/general_dxf_file_structure_dxf_aa.ht m [Accessed 9 Mar. 2019].

Ayubi, H.H., 2011. Advanced skills required for engineering leaders in global product development (Doctoral dissertation, Massachusetts Institute of Technology).

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., 2010, The Description Logic Handbook: Theory, Implementation and Applications, 2nd ed., Cambridge University Press, Cambridge, UK

Babic, B., Nesic, N. and Miljkovic, Z., 2008. A review of automated feature recognition with rule-based pattern recognition. Computers in Industry, 59(4), pp.321-337.

Bae, K.H. and Lichti, D.D., 2006. Automated registration of unorganised point clouds from terrestrial laser scanners. Curtin University of Technology..

Bae, K.H., 2006. Automated registration of unorganised point clouds from terrestrial laser scanners (Doctoral dissertation, Curtin University).

Baer, S., 2011. RhinoPython Scripts. [online] GitHub. Available at: https://github.com/mcneel/rhinoscriptsyntax/blob/rhino-6.x/README.md [Accessed 14 Mar. 2019].

Barbau, R., Krima, S., Rachuri, S., Narayanan, A., Fiorentini, X., Foufou, S., and Sriram, R. D., 2012, "OntoSTEP: Enriching Product Model Data Using Ontologies," Comput.-Aided Des., 44(6), pp. 575–590.

Barber, S.L., Junankar, A., Maitra, S., Iyer, G. and Devarajan, V., 2010. Experience in Development of Translators for AP203 Edition 2 Construction History. Computer-Aided Design and Applications, 7(4), pp.565-578.

Beckert, B., Hähnle, R. and Manya, F., 2000. The SAT problem of signed CNF formulas. In Labelled Deduction (pp. 59-80). Springer, Dordrecht.

Berners-Lee, T., Hendler, J., and Lassila, O., 2001, "The Semantic Web," Sci.Am., 284(5), pp. 28–37.

Bernstein, P.A., Madhavan, J. and Rahm, E., 2011. Generic schema matching, ten years later. Proceedings of the VLDB Endowment, 4(11), pp.695-701.

Besl, P.J. and Jain, R.C., 1986. Invariant surface characteristics for 3D object recognition in range images. Computer vision, graphics, and image processing, 33(1), pp.33-80.

Besl, P.J. and McKay, N.D., 1992, April. Method for registration of 3-D shapes. In Sensor Fusion IV: Control Paradigms and Data Structures (Vol. 1611, pp. 586-607). International Society for Optics and Photonics.

Bespalov, D., Ip, C.Y., Regli, W.C. and Shaffer, J., 2005, June. Benchmarking CAD search techniques. In Proceedings of the 2005 ACM symposium on Solid and physical modeling (pp. 275-286). ACM.

Bidan, M., Rowe, F. and Truex, D., 2012. An empirical study of IS architectures in French SMEs: integration approaches. European Journal of Information Systems, 21(3), pp.287-302.

Bin, W., Kaimo, H., Dong, L. and Hui, Z., 2017. Solid model edit distance: a multi-application and multi-level schema for CAD model retrieval. Visual Computing for Industry, Biomedicine and Art, 30(01), p.3.

Bird, S., Klein, E. and Loper, E., 2009. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc.".

Bittner, T., Donnelly, M. and Winter, S., 2005. Ontology and semantic interoperability. Large-scale 3D data integration: Challenges and Opportunities, pp.139-160.

Björk, B.C. and Laakso, M., 2010. CAD standardisation in the construction industry—A process view. Automation in Construction, 19(4), pp.398-406.

Blum, C., Puchinger, J., Raidl, G.R. and Roli, A., 2011. Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing, 11(6), pp.4135-4151.

Bronstein, M.M. and Kokkinos, I., 2010, June. Scale-invariant heat kernel signatures for non-rigid shape recognition. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (pp. 1704-1711). IEEE.

Bronsvoort, W.F. and Bidarra, R., 2000. Semantic feature modeling. Computer-Aided Design, 32, pp.201-225.

Buckley, C. and Voorhees, E.M., 2004, July. Retrieval evaluation with incomplete information. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 25-32). ACM.

Budanitsky, A. and Hirst, G., 2001, June. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In Workshop on WordNet and other lexical resources (Vol. 2, pp. 2-2).

BuildingSMART: Industry Foundation Classes Edition 3 Technical Corrigendum 1. http://www.buildingsmart.com (July 2007)

Burel, Gilles, and Hugues Hénocq. "Three-dimensional invariants and their application to object recognition." Signal Processing 45.1 (1995): 1-22.

Bustos, B., Keim, D.A., Saupe, D., Schreck, T. and Vranić, D.V., 2005. Feature-based similarity search in 3D object databases. ACM Computing Surveys (CSUR), 37(4), pp.345-387.

Butala, P., Vrabič, R. and Oosthuizen, G., 2013. Distributed manufacturing systems and the internet of things: a case study.

Bäck, T., Fogel, D.B. and Michalewicz, Z. eds., 2018. Evolutionary computation 1: Basic algorithms and operators. CRC press.

CAD Models Dataset. 2004, [online] Available at: https://web.archive.org/web/20181127121430/http://edge.cs.drexel.edu/repository/ [Accessed 27 Feb. 2019].

Camarinha-Matos, L.M., Afsarmanesh, H. and Rabelo, R.J., 2003. Infrastructure developments for agile virtual enterprises. International Journal of Computer Integrated Manufacturing, 16(4-5), pp.235-254.

Camba, J.D. and Contero, M., 2016. Parametric CAD modeling: An analysis of strategies for design reusability. Computer-Aided Design, 74, pp.18-31.

Cao, Y. and Petzold, L., 2006. Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems. Journal of Computational Physics, 212(1), pp.6-24.

Cardone, A., Gupta, S.K. and Karnik, M., 2003. A survey of shape similarity assessment algorithms for product design and manufacturing applications. Journal of Computing and Information Science in Engineering, 3(2), pp.109-118.

Chen, D.Y., Tian, X.P., Shen, Y.T. and Ouhyoung, M., 2003, September. On visual similarity based 3D model retrieval. In Computer graphics forum (Vol. 22, No. 3, pp. 223-232). Blackwell Publishing, Inc.

Chen, H. and Lynch, K.J., 1992. Automatic construction of networks of concepts characterizing document databases. IEEE Transactions on Systems, Man, and Cybernetics, 22(5), pp.885-902.

Choi, G.H., Mun, D. and Han, S., 2002. Exchange of CAD part models based on the macro-parametric approach. International Journal of CAD/CAM, 2(1), pp.13-21.

Choi, N., Song, I. Y., & Han, H. (2006). A survey on ontology mapping. SIGMOD Record, 35(3), 34-41.

Christoph, M.H., Robert, J.A. and Arinyo, J., 1998. CAD and the product master model. Computer-Aided Design, 30(11), pp.905-918.

Chu, C.H. and Hsu, Y.C., 2006. Similarity assessment of 3D mechanical components for design reuse. Robotics and Computer-Integrated Manufacturing, 22(4), pp.332-341.

Chungoora, N. and Young, R. I. M., 2008, Ontology Mapping To Support Semantic Interoperability In Product Design & Manufacture, Proceedings of the 1st International Workshop on Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) in Conjunction with the CAiSE'08 Conference, 340, pp. 1-15.

Church, K.W. and Hanks, P., 1990. Word association norms, mutual information, and lexicography. Computational linguistics,16(1), pp.22-29.

Cicirello, V.A. and Regli, W.C., 1999, June. Resolving non-uniqueness in design feature histories. In Proceedings of the fifth ACM symposium on Solid modeling and applications (pp. 76-84). ACM.

Cicirello, V.A. and Regli, W.C., 2002. An approach to a feature-based comparison of solid models of machined parts. AI EDAM, 16(5), pp.385-399.

Ciocoiu, M., Nau, D.S. and Gruninger, M., 2001. Ontologies for integrating engineering applications. Journal of Computing and Information Science in Engineering, 1(1), pp.12-22.

Cordella, A., 2006. Transaction costs and information systems: does IT add up?. Journal of information technology, 21(3), pp.195-202.

Corley, C. and Mihalcea, R., 2005, June. Measuring the semantic similarity of texts. In Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment (pp. 13-18). Association for Computational Linguistics.

Costa, C.A., Harding J.A., Young R.I.M., (2001) "The application of UML and an open distribute process framework to information systems design" Computers in Industry Vol 46, pp33-48.

Crossley, S., Dascalu, M. and McNamara, D., 2017, May. How Important Is Size? An Investigation of Corpus Size and Meaning in both Latent Semantic Analysis and Latent Dirichlet Allocation. In The Thirtieth International Flairs Conference.

Crouch, C.J., 1990. An approach to the automatic construction of global thesauri. Information Processing & Management, 26(5), pp.629-640.

Ćuković, S., Devedžić, G., Fiorentino, M., Ghionea, I., Anwer, N., Qiao, L. and Rakonjac, B., 2017. A comparative study of CAD data exchange based on the STEP standard. U.P.B. Sci. Bull., Series D, Vol. 79, Iss. 4.

Dalianis, H. and Hovy, E., 1998, August. Integrating STEP schemata using automatic methods. In Proceedings of the ECAI-98 Workshop on Applications of Ontologies and Problem-Solving Methods (pp. 54-66).

Daras, P. and Axenopoulos, A., 2009, June. A compact multi-view descriptor for 3D object retrieval. In Content-Based Multimedia Indexing, 2009. CBMI'09. Seventh International Workshop on (pp. 115-119). IEEE.

Daras, P. and Axenopoulos, A., 2010. A 3D shape retrieval framework supporting multimodal queries. International Journal of Computer Vision, 89(2-3), pp.229-247.

Dartigues, C., 2003. Product data exchange in a collaborative environment. Lyon: PhD Thesis, University of Claude Bernard-Lyon.

Dartigues, C., Ghodous, P., Gruninger, M., Pallez, D. and Sriram, R., 2007. CAD/CAPP integration using feature ontology. Concurrent Engineering, 15(2), pp.237-249.

De Sapio, V., 2010. Long-term archival and retrieval of engineering data: implications for the DART workbench (No. SAND2010-2477). Sandia National Laboratories.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R., 1990. Indexing by latent semantic analysis. Journal of the American society for information science, 41(6), pp.391-407.

Deitz, P.H. and Appllin, K.A., 1993. Practices and Standards in the Construction of BRL-CAD Target Descriptions (No. ARL-MR-103). Army Research Lab Aberdeen Proving Ground MD.

Deserno, M., 2004. How to generate equidistributed points on the surface of a sphere. P.-If Polymerforshung (Ed.), p.99.

developer.rhino3d.com., 2017. RhinoScript Programmer's Reference. [online] Available at: https://developer.rhino3d.com/api/rhinoscript/index.html [Accessed 14 Mar. 2019].

Docs.microsoft.com. (2018). Component Object Model (COM) - Windows applications. [online] Available at: https://docs.microsoft.com/en-gb/windows/desktop/com/component-object-model--com--portal [Accessed 8 Feb. 2018].

Dong, W., Wang, Z., Charikar, M. and Li, K., 2008, October. Efficiently matching sets of features with random histograms. In Proceedings of the 16th ACM international conference on Multimedia (pp. 179-188). ACM.

Dong, X., Halevy, A., Madhavan, J., Nemes, E. and Zhang, J., 2004, August. Similarity search for web services. In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30 (pp. 372-383). VLDB Endowment.

Dorador J.M., Young R.I.M. (2000). The Application of IDEF0, IDEF3 AND UML Methodologies in the Creation of Information Models. International Journal of Computer Integrated Manufacture, Vol.13, No.5, pp430-445 ISSN 0951-192X

Dublincore.org. (2012). DCMI: DCMI Metadata Terms. [online] Available at: http://dublincore.org/documents/dcmi-terms/ [Accessed 4 Jun. 2018].

Duncan, D. (2019). DoD as a Model-Based Enterprise. Defense Standardization Journal, [online] October/December 2015. Available at: https://www.dsp.dla.mil/Portals/26/Documents/Publications/Journal/151 201-DSPJ-04.pdf [Accessed 22 Aug. 2019].

Dusserre, E. and Padró, M., 2017. Bigger does not mean better! We prefer specificity. In Iwcs 2017—12th international conference on computational semantics—short papers.

Dutagaci, H., Sankur, B. and Yemez, Y., 2005, June. Transform-based methods for indexing and retrieval of 3d objects. In 3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on (pp. 188-195). IEEE.

Eddy, D., Krishnamurty, S., Grosse, I., Liotta, A. and Wileden, J., 2012, August. Toward Integration of a Semantic Framework With a Commercial PLM System. In ASME 2012 International Design

Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. 1247-1261). American Society of Mechanical Engineers.

Edelsbrunner, H., Letscher, D. and Zomorodian, A., 2000. Topological persistence and simplification. In Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on (pp. 454-463). IEEE.

Eekels, J. and Roozenburg, N.F., 1991. A methodological comparison of the structures of scientific research and engineering design: their similarities and differences. Design studies, 12(4), pp.197-203.

Egenhofer, M.J. and Franzosa, R.D., 1991. Point-set topological spatial relations. International Journal of Geographical Information System, 5(2), pp.161-174.

El Kadiri, S., and Kiritsis, D., 2015, "Ontologies in the Context of Product Lifecycle Management: State of the Art Literature Review," Int. J. Prod. Res., 53(18), pp. 5657–5668.

El-Mehalawi, M. and Miller, R.A., 2003. A database system of mechanical components based on geometric and topological similarity. Part I: representation. Computer-Aided Design, 35(1), pp.83-94.

El-Mehalawi, M. and Miller, R.A., 2003. A database system of mechanical components based on geometric and topological similarity. Part II: indexing, retrieval, matching, and similarity assessment. Computer-Aided Design, 35(1), pp.95-105.

Elinson, A., Nau, D.S. and Regli, W.C., 1997, May. Feature-based similarity assessment of solid models. In Proceedings of the fourth ACM symposium on Solid modeling and applications (pp. 297-310). ACM.

Erber, T. and Hockney, G.M., 1991. Equilibrium configurations of N equal charges on a sphere. Journal of Physics A: Mathematical and General, 24(23), p.L1369.

Euzenat, J., Le Bach, T., Barrasa, J., Bouquet, P., De Bo, J. and Dieng, R., 2004. State of the art on ontology alignment. KnowledgeWeb deliverable D2. 2.3. Karlsruhe, Germany: University of Karlsruhe.

Fankam, C., Jean, S., Pierra, G., Bellatreche, L. and Ameur, Y.A., 2009, March. Towards connecting database applications to ontologies. In Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA'09. First International Conference on (pp. 131-137). IEEE.

Farin, G. and Hansford, D., 1999. Discrete Coons patches. *Computer Aided Geometric Design*, *16*(7), pp.691-700.

Farjana, S.H., Han, S. and Mun, D., 2016. Implementation of persistent identification of topological entities based on macro-parametrics approach. Journal of Computational Design and Engineering, 3(2), pp.161-177.

Fielding, N.G., 2012. Triangulation and mixed methods designs: Data integration with new research technologies. Journal of mixed methods research, 6(2), pp.124-136.

Fellbaum, C., 1998. A semantic network of English verbs. WordNet: An electronic lexical database, 3, pp.153-178. Available at https://wordnet.princeton.edu/

Fenves, S.J., Foufou, S., Bock, C. and Sriram, R.D., 2008. CPM2: a core model for product data. Journal of computing and information science in engineering, 8(1), p.014501.

Ferri, M., Frosini, P. and Landi, C., 2011. Stable shape comparison by persistent homology. Atti Semin. Mat. Fis. Univ. Modena Reggio Emilia, 58, pp.143-162.

Firth, J.R., 1957. A synopsis of linguistic theory, 1930-1955. Studies in linguistic analysis.

Folk, M., Heber, G., Koziol, Q., Pourmal, E. and Robinson, D., 2011, March. An overview of the HDF5 technology suite and its applications. In Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases (pp. 36-47). ACM.

Foltz, P.W., Kintsch, W. and Landauer, T.K., 1998. The measurement of textual coherence with latent semantic analysis. Discourse processes, 25(2-3), pp.285-307.

Fortin, F.A., Rainville, F.M.D., Gardner, M.A., Parizeau, M. and Gagné, C., 2012. DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research, 13(Jul), pp.2171-2175.

Fortineau, V., Paviot, T., and Lamouri, S., 2013, "Improving the Interoperability of Industrial Information Systems With Description Logic-Based Models— The State of the Art," Comput. Ind., 64(4), pp. 363–375

Freecadweb.org., 2019. Part Cone - FreeCAD Documentation. [online] Available at: https://www.freecadweb.org/wiki/Part_Cone [Accessed 14 Mar. 2019].

Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D. and Jacobs, D., 2003. A search engine for 3D models. ACM Transactions on Graphics (TOG), 22(1), pp.83-105.

Gallaher, M.P., O’Connor, A.C., Dettbarn Jr, J.L. and Gilday, L.T., 2004. Cost Analysis of Inadequate Interoperability in the US Capital Facilities Industry. 2004. National Institute of Standards and Technology: Gaithersburg, Maryland, p.210.

Gao, S. and Shah, J.J., 1998. Automatic recognition of interacting machining features based on minimal condition subgraph. Computer-Aided Design, 30(9), pp.727-739.

Gao, Y., Dai, Q. and Zhang, N.Y., 2010. 3D model comparison using spatial structure circular descriptor. Pattern Recognition, 43(3), pp.1142-1151.

Garey, M. R. and Johnson, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York, 1979.

GCR, N., 2004. Cost analysis of inadequate interoperability in the US capital facilities industry. National Institute of Standards and Technology (NIST).

Gelfand, N., Mitra, N.J., Guibas, L.J. and Pottmann, H., 2005, July. Robust global registration. In Symposium on geometry processing (Vol. 2, No. 3, p. 5).

Gerbino, S. and Brondi, A., 2004. Interoperability issues among CAD systems: a benchmarking study of 7 commercial MCAD software. In DS 32: Proceedings of DESIGN 2004, the 8th International Design Conference, Dubrovnik, Croatia.

Gerst, M. and Bunduchi, R., 2005. Shaping IT standardization in the automotive industry–the role of power in driving portal standardization. Electronic Markets, 15(4), pp.335-343.

Ghafour, S.A., Ghodous, P., Shariat, B. and Perna, E., 2006. An Ontology-Based Approach for" Procedural CAD Models" Data Exchange. Frontiers In Artificial Intelligence And Applications, 143, p.251.

Gielingh, W., 2008. An assessment of the current state of product data technologies. Computer-Aided Design, 40(7), pp.750-759.

Giunchiglia, F., Yatskevich, M. and Shvaiko, P., 2007. Semantic matching: Algorithms and implementation. In Journal on data semantics IX (pp. 1-38). Springer, Berlin, Heidelberg.

Global Supplier Info Pack For FEDE-C3PNG Integration. (2017). [ebook] Lindsay Larcombe. Available at: https://web.c3p.ford.com/start/Global_Info_Pack_version_1.4.5_English.pdf [Accessed 10 Jan. 2018].

Goldberg, D.E., 1991. Real-coded genetic algorithms, virtual alphabets, and blocking. Complex systems, 5(2), pp.139-167.

Gomaa, W.H. and Fahmy, A.A., 2013. A survey of text similarity approaches. International Journal of Computer Applications, 68(13), pp.13-18.

Goossenaerts, J., Dreverman, M., Smits, J.M. and van Exel, P.W., 2009. Plant lifecycle data standards in the process industry: diagnosis and resolution of collective action failure.

Gope, C. and Kehtarnavaz, N., 2007. Affine invariant comparison of point-sets using convex hulls and hausdorff distances. Pattern Recognition, 40(1), pp.309-320.

Gregor, S. and Hevner, A.R., 2013. Positioning and presenting design science research for maximum impact. MIS quarterly, pp.337-355.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge acquisition, 5(2), pp.199-220.

Gruber, T.R., 1995. Toward principles for the design of ontologies used for knowledge sharing?. International journal of human-computer studies, 43(5-6), pp.907-928.

Gruninger, M. and Menzel, C., 2003. The process specification language (PSL) theory and applications. AI magazine, 24(3), p.63.

Guarino, N. ed., 1998. Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy (Vol. 46). IOS press.

Guo, W. and Diab, M., 2012, July. Modeling sentences in the latent space. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-volume 1 (pp. 864-872). Association for Computational Linguistics.

Gupta, R. K., and Gurumoorthy, B., 2008, "A Feature-Based Framework for Semantic Interoperability of Product Models," J. Mech. Eng., 54(6), pp. 446–457.

Han, J., Pratt, M. and Regli, W.C., 2000. Manufacturing feature recognition from solid models: a status report. IEEE Transactions on Robotics and Automation, 16(6), pp.782-796.

Hanayneh, L., Wang, Y., Wang, Y., Wileden, J.C. and Qureshi, K.A., 2008, January. Feature mapping automation for CAD data exchange. In ASME 2008 International Design Engineering

Technical Conferences and Computers and Information in Engineering Conference (pp. 1257-1266). American Society of Mechanical Engineers.

Harris, C. and Stephens, M., 1988, August. A combined corner and edge detector. In Alvey vision conference (Vol. 15, No. 50, pp. 10-5244).

Harris, Z., (1968), "Mathematical Structures of Language," New York: Wiley.

Haythornthwaite, C., 2009, January. Crowds and communities: Light and heavyweight models of peer production. In System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on (pp. 1-10). IEEE.

He, L., Ming, X., Ni, Y., Li, M., Zheng, M., and Xu, Z., 2015, "Ontology Based Information Integration and Sharing for Collaborative Part and Tooling Development," Concurrent Eng., 23(3), pp. 199–212.

Healy, D.M., Rockmore, D.N., Kostelec, P.J. and Moore, S., 2003. FFTs for the 2-sphere-improvements and variations. Journal of Fourier Analysis and Applications, 9(4), pp.341-385.

Heller, T., Farrow, C., 2006-2019, comtypes (Version 1.1.4) [Software] Available from https://github.com/enthought/comtypes

Henderson, M.R. and Anderson, D.C., 1984. Computer recognition and extraction of form features: a CAD/CAM link. Computers in industry, 5(4), pp.329-339.

Hendler, J. and McGuinness, D.L., 2000. The DARPA agent markup language. IEEE Intelligent systems, 15(6), pp.67-73.

Heutelbeck, D., Brunsmann, J., Wilkes, W. and Hundsdörfer, A., 2009, June. Motivations and challenges for digital preservation in design and engineering. In Proceedings of First International Workshop on Innovation in Digital Preservation.

Hevner, A. and Chatterjee, S., 2010. Design science research in information systems. In Design research in information systems (pp. 9-22). Springer, Boston, MA.

Hevner, A., March, S.T., Park, J. and Ram, S., 2004. Design science in information systems research. MIS quarterly, 28(1), pp.75-105.

Hilaga, M., Shinagawa, Y., Kohmura, T. and Kunii, T.L., 2001, August. Topology matching for fully automatic similarity estimation of 3D shapes. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques(pp. 203-212). ACM.

Hoffmann, C., Shapiro, V. and Srinivasan, V., 2014. Geometric interoperability via queries. Computer-Aided Design, 46, pp.148-159.

Hoffmann, C.M. and Juan, R., 1992. EREP An editable high-level representation for geometric design and analysis.

Holland, J.H., 1998. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to biology, control and artificial intelligence. MIT Press, ISBN 0-262-58 111- 6. (NB original printing 1975).

Horrocks, I. and Tobies, S., 2000. Reasoning with Axioms: Theory and Pratice. arXiv preprint cs/0 005 012.

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M., 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 21, p.79.

Horst, J., Hartman, N. and Wong, G., 2010, September. Metrics for the cost of proprietary information exchange languages in intelligent systems. In Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop (pp. 77-81). ACM.

Hounsell, M.S. and Case, K., 1998. Feature-based designer's intents. Manufacturing Strategies for Europe, The Proc of the 15th Conf of the Irish Manuf Cttee, IMC-15, (eds C Hepburn and D M J Harris), University of Ulster

Howard, R. and Björk, B.C., 2008. Building information modelling–Experts' views on standardisation and industry deployment. Advanced Engineering Informatics, 22(2), pp.271-280.

Hromkovič, J., 2013. Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics. Springer Science & Business Media.

Huang, R., Zhang, S., Bai, X., Xu, C. and Huang, B., 2015. An effective subpart retrieval approach of 3D CAD models for manufacturing process reuse. Computers in Industry, 67, pp.38-53.

Iti-global.com. 2018, Proficiency. [online] Available at: https://www.iti-global.com/proficiency [Accessed 4 Jun. 2018].

Jiang, J.J. and Conrath, D.W., 1997. Semantic similarity based on corpus statistics and lexical taxonomy. arXiv preprint cmp-lg/9 709 008.

Jiayi, P., Cheng, C.P.J., Lau, G.T. and Law, K.H., 2008. Utilizing statistical semantic similarity techniques for ontology mapping—With applications to AEC standard models. Tsinghua Science & Technology, 13, pp.217-222.

Johnson, A.E. and Hebert, M., 1999. Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Transactions on pattern analysis and machine intelligence, 21(5), pp.433-449.

Johnson, A.E., 1997. Spin-images: a representation for 3-D surface matching (Doctoral dissertation, Carnegie Mellon University).

Jolliffe, I.T., 1986. Principal component analysis and factor analysis. In Principal component analysis (pp. 115-128). Springer, New York, NY.

Joshi, S. and Chang, T.C., 1988. Graph-based heuristics for recognition of machined features from a 3D solid model. Computer-Aided Design, 20(2), pp.58-66.

Jørgensen, S., Nielsen, K. and Jørgensen, K.A., 2011. Reconfigurable manufacturing systems as an application of mass customisation. International Journal of Industrial Engineering and Management, 1(3), pp.111-119.

Kalfoglou, Y. and Schorlemmer, M., 2003. Ontology mapping: the state of the art. The knowledge engineering review, 18(1), pp.1-31.

Karp, R.M., 1972. Reducibility among combinatorial problems. In Complexity of computer computations (pp. 85-103). Springer, Boston, MA.

Katz, M.L. and Shapiro, C., 1985. Network externalities, competition, and compatibility. The American economic review, 75(3), pp.424-440.

Kazmi, I.K., You, L. and Zhang, J.J., 2013, August. A survey of 2d and 3d shape descriptors. In Computer graphics, imaging and visualization (cgiv), 2013 10th international conference(pp. 1-10). IEEE.

Khuri, S., Bäck, T. and Heitkötter, J., 1994, April. The zero/one multiple knapsack problem and genetic algorithms. In Proceedings of the 1994 ACM symposium on Applied computing (pp. 188-193). ACM.

Kim, B. and Han, S., 2007. Integration of history-based parametric translators using the automation APIs. International Journal of Product Lifecycle Management, 2(1), pp.18-29.

Kim, J., Pratt, M.J., Iyer, R. and Sriram, R., 2007. Data exchange of parametric CAD models using ISO 10 303-108. NIST intergovernmental report. Gaithersburg (MD, USA): National Institute of Standards and Technology.

Kim, J., Pratt, M.J., Iyer, R.G. and Sriram, R.D., 2008. Standardized data exchange of CAD models with design intent. Computer-Aided Design, 40(7), pp.760-777.

Kim, K.Y., Manley, D.G. and Yang, H., 2006. Ontology-based assembly design and information sharing for collaborative product development. Computer-Aided Design, 38(12), pp.1233-1250.

Kim, O., Jayaram, U., Jayaram, S. and Zhu, L., 2009, January. An ontology mapping application using a shared ontology approach and a bridge ontology. In ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. 431-441). American Society of Mechanical Engineers.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., 1983. Optimization by simulated annealing. science, 220(4598), pp.671-680.

Klusch, M. and Kapahnke, P. (2019). SemWebCentral: OWL-S Service Retrieval Test Collection: Project Info. [online] Projects.semwebcentral.org. Available at: http://projects.semwebcentral.org/projects/owls-tc/ [Accessed 16 Feb. 2019].

Koenderink, J.J. and Van Doorn, A.J., 1992. Surface shape and curvature scales. Image and vision computing, 10(8), pp.557-564.

Kosanke, K., 2006. ISO Standards for Interoperability: a comparison. In Interoperability of enterprise software and applications (pp. 55-64). Springer, London.

Krima, S., Barbau, R., Fiorentini, X., Sudarsan, R. and Sriram, R.D., 2009. Ontostep: OWL-DL ontology for step. National Institute of Standards and Technology, NISTIR, 7561.

Kubotek USA (2006), "The 2006 CAD interoperability survey results: a Kubotek USA study of the design and manufacturing marketplace," (online)
http://www.kubotekusa.com/company/interopsurvey/index.asp (retrieved September 30, 2010)

Kuhn, H.W., 1955. The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2), pp.83-97.

Kuzminykh, A. and Hoffmann, C., 2008. On validating STEP product data exchange. Computer-Aided Design, 40(2), pp.133-138.

Kyprianou, L.K., 1980. Shape classification in computer-aided design (Doctoral dissertation, University of Cambridge).

Lakshmi, H.N. and Mohanty, H., 2015, February. A Preprocessing of Service Registry: Based on I/O Parameter Similarity. In International Conference on Distributed Computing and Internet Technology (pp. 220-232). Springer, Cham.

Landauer, T.K., Foltz, P.W. and Laham, D., 1998. An introduction to latent semantic analysis. Discourse processes, 25(2-3), pp.259-284.

Le Duigou, J., Bernard, A. and Perry, N., 2011. Framework for product lifecycle management integration in small and medium enterprises networks. Computer-Aided Design and Applications, 8(4), pp.531-544.

Le Duigou, J., Bernard, A., Perry, N. and Delplace, J.C., 2011. Application of PLM processes to respond to mechanical SMEs needs. In Global Product Development (pp. 319-326). Springer, Berlin, Heidelberg.

Le Duigou, J., Bernard, A., Perry, N. and Delplace, J.C., 2012. Generic PLM system for SMEs: Application to an equipment manufacturer. International Journal of Product Lifecycle Management 7, 6(1), pp.51-64.

Le, Q. and Mikolov, T., 2014, January. Distributed representations of sentences and documents. In International Conference on Machine Learning (pp. 1188-1196).

Leacock, C. and Chodorow, M., 1998. Combining local context and WordNet similarity for word sense identification. WordNet: An electronic lexical database, 49(2), pp.265-283.

Leifman, G., Katz, S., Tal, A. and Meir, R., 2003, February. Signatures of 3D models for retrieval. In Proceedings of the 4th Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics (pp. 159-163).

Li, B., Lu, Y., Li, C., Godil, A., Schreck, T., Aono, M., Burtscher, M., Fu, H., Furuya, T., Johan, H. and Liu, J., 2014. SHREC'14 track: Extended large scale sketch-based 3D shape retrieval. In Eurographics workshop on 3D object retrieval (Vol. 2014).

Li, J., Han, S., Shin, S., Lee, S., Kang, Y., Cho, H., Kim, H., Song, I., Kim, I. and Rathore, P.S., 2010. CAD data exchange using the macro-parametrics approach: an error report. International Journal of CAD/CAM, 10(2).

Li, M., Zhang, Y.F. and Fuh, J.Y.H., 2010. Retrieving reusable 3D CAD models using knowledge-driven dependency graph partitioning. Computer-Aided Design and Applications, 7(3), pp.417-430.

Li, X., He, F., Cai, X. and Zhang, D., 2012. CAD data exchange based on the recovery of feature modelling procedure. International Journal of Computer Integrated Manufacturing, 25(10), pp.874-887.

Li, Y., McLean, D., Bandar, Z.A. and Crockett, K., 2006. Sentence similarity based on semantic nets and corpus statistics. IEEE Transactions on Knowledge & Data Engineering, (8), pp.1138-1150.

Li, Z., Zhou, X. and Liu, W., 2015. A geometric reasoning approach to hierarchical representation for B-rep model retrieval. Computer-Aided Design, 62, pp.190-202.

Lin, D., 1998, July. An information-theoretic definition of similarity. In Icml (Vol. 98, No. 1998, pp. 296-304).

Lionel, S. 2019. Memory Limits for Applications on Windows*. [online] Software.intel.com. Available at: https://software.intel.com/en-us/articles/memory-limits-applications-windows [Accessed 17 Jan. 2019].

Liu, Q., 2012. A survey of recent view-based 3d model retrieval methods. arXiv preprint arXiv:1208.3670.

Liu, Y., Zha, H. and Qin, H., 2006, June. The generalized shape distributions for shape matching and analysis. In Shape Modeling and Applications, 2006. SMI 2006. IEEE International Conference on (pp. 16-16). IEEE.

Lomas, C. D. L. and Matthews, P. C. (2007) 'Meta-design for agile concurrent product design in the virtual enterprise.', International journal of agile manufacturing., 10 (2), pp.77-87.

Lu, W., Qin, Y., Qi, Q., Zeng, W., Zhong, Y., Liu, X. and Jiang, X., 2016. Selecting a semantic similarity measure for concepts in two different CAD model data ontologies. *Advanced Engineering Informatics*, *30*(3), pp.449-466.

Lubell, J., Rachuri, S., Mani, M. and Subrahmanian, E., 2008. Sustaining engineering informatics: Toward methods and metrics for digital curation. International Journal of Digital Curation, 3(2).

Ma, L., Huang, Z. and Wang, Y., 2009, August. Common design structure discovery from CAD models. In Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics' 09. 11th IEEE International Conference on (pp. 363-366). IEEE.

Ma, Y.S., Chen, G. and Thimm, G., 2008. Paradigm shift: unified and associative feature-based concurrent and collaborative engineering. Journal of Intelligent Manufacturing, 19(6), pp.625-641.

Maier, F. and Stumptner, M., 2007, October. Enhancements and ontological use of ISO-10 303 (STEP) to support the exchange of parameterised product data models. In Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on (pp. 433-440). IEEE.

Malone, T.W., Yates, J. and Benjamin, R.I., 1987. Electronic markets and electronic hierarchies. Communications of the ACM, 30(6), pp.484-497.

March, S.T. and Smith, G.F., 1995. Design and natural science research on information technology. *Decision support systems*, *15*(4), pp.251-266.

Marefat, M. and Kashyap, R.L., 1990. Geometric reasoning for recognition of three-dimensional object features. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(10), pp.949-965.

Masuda, T. and Yokoya, N., 1994, February. A robust method for registration and segmentation of multiple range images. In CAD-Based Vision Workshop, 1994., Proceedings of the 1994 Second (pp. 106-113). IEEE.

Masuda, T. and Yokoya, N., 1995. A robust method for registration and segmentation of multiple range images. Computer vision and image understanding, 61(3), pp.295-307.

McGuinness, D. L., and Harmelen, F. V., 2004, “OWL Web Ontology Language Overview,” last accessed Jan. 18, 2016, http://www.w3.org/TR/owlfeatures/

McKenzie-Veal, D., Hartman, N.W. and Springer, J., 2010. Implementing ontology-based information sharing in product lifecycle management. In 65th Midyear Meeting Proceedings, Houghton, MI, Oct (pp. 3-6).

Miao, H.K., Sridharan, N. and Shah, J.J., 2002. CAD-CAM integration using machining features. In International Journal of Computer Integrated Manufacturing, 15(4), pp.296-318.

Michalewicz, Z., 1995. A survey of constraint handling techniques in evolutionary computation methods. Evolutionary programming, 4, pp.135-155.

Middleditch, A. and Reade, C., 1997, May. A kernel for geometric features. In Proceedings of the fourth ACM symposium on Solid modeling and applications (pp. 131-140). ACM.

Mihalcea, R., Corley, C. and Strapparava, C., 2006, July. Corpus-based and knowledge-based measures of text semantic similarity. In AAAI (Vol. 6, pp. 775-780).

Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

Miller, G.A., 1995. WordNet: a lexical database for English. Communications of the ACM, 38(11), pp.39-41.

Miller, G.A., 1995. WordNet: a lexical database for English. Communications of the ACM, 38(11), pp.39-41.

Min, P., Kazhdan, M. and Funkhouser, T., 2004, September. A comparison of text and shape matching for retrieval of online 3D models. In International Conference on Theory and Practice of Digital Libraries (pp. 209-220). Springer, Berlin, Heidelberg.

Ming, X.G., Yan, J.Q., Lu, W.F. and Ma, D.Z., 2005. Technology solutions for collaborative product lifecycle management–status review and future trend. Concurrent Engineering, 13(4), pp.311-319.

Mostefai, S., Bouras, A., Batouche, M. Effective collaboration in product development via a common sharable ontology. International Journal of Computer Intelligence, vol. 2, no. 4, 2005, p. 206-212.

Mun, D. and Han, S., 2005. Identification of topological entities and naming mapping for parametric CAD model exchange. In Intl. J. of CAD/CAM.

Mun, D., Han, S., Kim, J. and Oh, Y., 2003. A set of standard modeling commands for the history-based parametric approach. Computer-aided design, 35(13), pp.1171-1179.

National Design Repository, 2005, National Institute of Standards and Technology, http://edge.mcs.drexel.edu/repository/frameset.html, [Accessed 8 Feb. 2018].

Negri, E., Fumagalli, L., Garetti, M., and Tanca, L., 2015, "Requirements and Languages for the Semantic Representation of Manufacturing Systems," Comput. Ind., 81(9), pp. 55–66.

Nunamaker Jr, J.F., Chen, M. and Purdin, T.D., 1990. Systems development in information systems research. Journal of management information systems, 7(3), pp.89-106.

OECD, 2000, Enhancing the Competitiveness of SMEs In The Global Economy: Strategies And Policies, Bologna: OECD

Ohbuchi, R., Osada, K., Furuya, T. and Banno, T., 2008, June. Salient local visual features for shape-based 3D model retrieval. In Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on (pp. 93-102). IEEE.

Ohbuchi, R., Otagiri, T., Ibato, M. and Takei, T., 2002. Shape-similarity search of three-dimensional models using parameterized statistics. In Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on(pp. 265-274). IEEE.

Oomori, S., Nishida, T. and Kurogi, S., 2016. Point cloud matching using singular value decomposition. Artificial Life and Robotics, 21(2), pp.149-154.

Osada, R., Funkhouser, T., Chazelle, B. and Dobkin, D., 2002. Shape distributions. ACM Transactions on Graphics (TOG), 21(4), pp.807-832.

Paik, I., Fujikawa, E. and Kim, S., 2010, November. Aggregating Web Service matchmaking variants using web search engine and machine learning. In *2010 2nd International Symposium on Aware Computing* (pp. 191-195). IEEE.

Pal, S., 2019. Computing Semantic Similarity for Short Sentences. [online] Sujitpal.blogspot.com. Available at: http://sujitpal.blogspot.com/2014/12/semantic-similarity-for-short-sentences.html [Accessed 13 Jan. 2016].

Panetto, H. and Molina, A., 2008. Enterprise integration and interoperability in manufacturing systems: Trends and issues. Computers in industry, 59(7), pp.641-646.

Paquet, E., Rioux, M., Murching, A., Naveen, T. and Tabatabai, A., 2000. Description of shape information for 2-D and 3-D objects. Signal processing: Image communication, 16(1-2), pp.103-122.

Passos, A. and Wainer, J., 2009, September. Wordnet-based metrics do not seem to help document clustering. In International Workshop on Web and Text Intelligence (WTI-2009).

Patel, M. and Ball, A., 2008. Challenges and issues relating to the use of representation information for the digital curation of crystallography and engineering data. In International Journal of Digital Curation, 3(1).

Patel, M., Ball, A. and Ding, L., 2008, October. Curation and preservation of CAD engineering models in product lifecycle management. In Conference on Virtual Systems and Multimedia Dedicated to Digital Heritage (VSMM'08) (pp. 59-66). University of Bath.

Patil, L., Dutta, D. and Sriram, R., 2005. Ontology-based exchange of product data semantics. IEEE Transactions on automation science and engineering, 2(3), pp.213-225.

Paviot, T., Cheutet, V. and Lamouri, S., 2009, July. Design and logistics IT federation through Product Lifecycle Support standard. In PLM09, IFIP WG 5.1 (pp. pp-139).

Paviot, T., Lamouri, S. and Cheutet, V., 2011. A generic multiCAD/multiPDM interoperability framework. International Journal of Services Operations and Informatics, 6(1-2), pp.124-137.

Pease, A. 2018. The Suggested Upper Merged Ontology (SUMO) - Ontology Portal. [online] Adampease.org. Available at: http://www.adampease.org/OP/ [Accessed 4 Jun. 2018].

Pedersen, T., 2008. Computational approaches to measuring the similarity of short contexts: A review of applications and methods. arXiv preprint arXiv:0806.3787.

Peeling, N. and Satchell, J., 2001. Analysis of the impact of open source software. QinetiQ Ltd. QINETIQ/KI/SEB/CR010 223.

Peffers, K., Tuunanen, T., Rothenberger, M.A. and Chatterjee, S., 2007. A design science research methodology for information systems research. Journal of management information systems, 24(3), pp.45-77.

Pennington, J., Socher, R. and Manning, C., 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Peruzzini, M., Mengoni, M. and Germani, M., 2011, July. PLM benefits for networked SMEs. In The proceedings of 11th International Conference on Product Lifecycle Management. Edited by JH, Pels et al., PLM 11, 11-13 July, 2011, Eindhoven University of Technology. 14 pp.

Piegl, L. and Tiller, W., 1987. Curve and surface constructions using rational B-splines. Computer-Aided Design, 19(9), pp.485-498.

Pilgrim (2019). Munkres' Assignment Algorithm. [online] Csclab.murraystate.edu. Available at: http://csclab.murraystate.edu/~bob.pilgrim/445/munkres_old.html [Accessed 21 Jan. 2019].

Pilli, S.I.S.S., 2017. Algorithms for Recognition of Geometric and Parametric Features in prismatic parts using IGES. International Journal Of Engineering And Computer Science, 6(4).

Pratt, M.J. and Anderson, B.D., 2001. A shape modelling applications programming interface for the STEP standard. Computer-Aided Design, 33(7), pp.531-543.

Pratt, M.J. and Kim, J., 2006, June. Experience in the exchange of procedural shape models using ISO 10 303 (STEP). In Proceedings of the 2006 ACM symposium on Solid and physical modeling (pp. 229-238). ACM.

Pratt, M.J., 2001. Introduction to ISO 10 303—the STEP standard for product data exchange. Journal of Computing and Information Science in Engineering, 1(1), pp.102-103.

Pratt, M.J., 2004, June. Extension of ISO 10 303, the STEP standard, for the exchange of procedural shape models. In Shape Modeling Applications, 2004. Proceedings (pp. 317-326). IEEE.

Pratt, M.J., 2005. ISO 10 303, the STEP standard for product data exchange, and its PLM capabilities. International Journal of Product Lifecycle Management, 1(1), pp.86-94.

Pratt, M.J., Anderson, B.D. and Ranger, T., 2005. Towards the standardized exchange of parameterized feature-based CAD models. Computer-Aided Design, 37(12), pp.1251-1265.

Qi, J. and Shapiro, V., 2006. Geometric interoperability with epsilon solidity. Journal of computing and information science in engineering, 6(3), pp.213-220.

Qin, Y., Lu, W., Qi, Q., Liu, X., Zhong, Y., Scott, P.J. and Jiang, X., 2017. Status, comparison, and issues of Computer-Aided Design model data exchange methods based on standardized neutral files and Web Ontology Language file. Journal of Computing and Information Science in Engineering, 17(1), p.010801.

Rachuri, S., Foufou, S., Kemmerer, S. and Rachuri, S., 2006. Analysis of Standards for Lifecycle Management of Systems for US Army: A Preliminary Investigation. US Department of Commerce, National Institute of Standards and Technology.

Rahm, E. and Bernstein, P.A., 2001. A survey of approaches to automatic schema matching. the VLDB Journal, 10(4), pp.334-350.

Rakhmanov, E.A., Saff, E.B. and Zhou, Y.M., 1994. Minimal discrete energy on the sphere. Math. Res. Lett, 1(6), pp.647-662.

Ramesh, M., Yip-Hoi, D. and Dutta, D., 2001. Feature based shape similarity measurement for retrieval of mechanical parts. Journal of Computing and Information Science in Engineering, 1(3), pp.245-256.

Ramos, L., 2015, "Semantic Web for Manufacturing, Trends and Open Issues: Toward a State of the Art," Comput. Ind. Eng., 90(12), pp. 444–460.

Rappoport, A., 2003, June. An architecture for universal CAD data exchange. In ACM Symposium on Solid and Physical Modeling: Proceedings of the eighth ACM symposium on Solid modeling and applications (Vol. 16, No. 20, pp. 266-269).

Rappoport, A., Spitz, S. and Etzion, M., 2005, June. One-dimensional selections for feature-based data exchange. In Proceedings of the 2005 ACM symposium on Solid and physical modeling (pp. 125-134). ACM.

Rappoport, A., Spitz, S. and Etzion, M., 2006, July. Two-dimensional selections for feature-based data exchange. In International Conference on Geometric Modeling and Processing (pp. 325-342). Springer, Berlin, Heidelberg.

Reed, S.L. and Lenat, D.B., 2002, July. Mapping ontologies into Cyc. In AAAI 2002 Conference Workshop on Ontologies For The Semantic Web (pp. 1-6).

Regli, W. and Gaines, D. (1997). A National Repository for Design and Process Planning. [online] NIST. Available at: https://www.nist.gov/publications/national-repository-design-and-process-planning [Accessed 27 Feb. 2019].

Regli, W.C., Gupta, S.K. and Nau, D.S., 1995. Extracting alternative machining features: An algorithmic approach. Research in Engineering Design, 7(3), pp.173-192.

Rehurek, R. and Sojka, P., 2010. Software framework for topic modelling with large corpora. In In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks.

Resnik, P., 1995. Using information content to evaluate semantic similarity in a taxonomy. arXiv preprint cmp-lg/9 511 007.

Richardson, L. (2019). Beautiful Soup: We called him Tortoise because he taught us.. [online] Crummy.com. Available at: https://www.crummy.com/software/BeautifulSoup/ [Accessed 28 Jan. 2019].

Riegel, J., Mayer, M., van Havre, Y., 2001-2017, FreeCAD (Version 0.17.13522) [Software] Available from http://www.freecadweb.org

Romero, D., Rabelo, R.J. and Molina, A., 2012, June. On the management of virtual enterprise's inheritance between virtual manufacturing & service enterprises: Supporting "dynamic" product-service business ecosystems. In Engineering, Technology and Innovation (ICE), 2012 18th International ICE Conference on (pp. 1-11). IEEE.

Romero, D., Rabelo, R.J., Hincapie, M. and Molina, A., 2009. Next generation manufacturing systems and the virtual enterprise. IFAC Proceedings Volumes, 42(4), pp.630-637.

Rong, X., 2014. word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.

Rullani, E., 2000. Enhancing the competitiveness of SMEs in the global economy: the strategies and policies. In Bologna 2000 SME Conference, June.

Rusu, R.B., Marton, Z.C., Blodow, N. and Beetz, M., 2008. Persistent point feature histograms for 3D point clouds. In Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany (pp. 119-128).

Sadjadi, F.A. and Hall, E.L., 1980. Three-dimensional moment invariants. IEEE Transactions on Pattern Analysis and Machine Intelligence, (2), pp.127-136.

Sakurai, H., 1995. Volume decomposition and feature recognition: Part 1—polyhedral objects. Computer-Aided Design, 27(11), pp.833-843.

Salton, G., Wong, A. and Yang, C.S., 1975. A vector space model for automatic indexing. Communications of the ACM, 18(11), pp.613-620.

Sanchez, L.M. and Nagi, R., 2001. A review of agile manufacturing systems. International Journal of Production Research, 39(16), pp.3561-3600.

Sanfilippo, E.M. and Borgo, S., 2016. What are features? An ontology-based review of the literature. Computer-Aided Design, 80, pp.9-18.

Šarić, F., Glavaš, G., Karan, M., Šnajder, J. and Bašić, B.D., 2012, June. Takelab: Systems for measuring semantic text similarity. In Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (pp. 441-448). Association for Computational Linguistics.

Sawtell, M.A., 2002. A Study of the Communication of Design Engineering Information by Way of Computer Systems within the Automotive Manufacturing Community.

Sebastian, R., 2010, May. Breaking through business and legal barriers of open collaborative processes based on building information modelling (BIM). In W113-Special Track 18th CIB World Building Congress May 2010 Salford, United Kingdom (p. 166).

Senellart, P. and Blondel, V.D., 2008. Automatic discovery of similar words. In Survey of Text Mining II (pp. 25-44). Springer, London.

Seo, T.S., Lee, Y.S., Cheon, S.U., Han, S.H., Patil, L. and Dutta, D., 2005. Sharing CAD models based on feature ontology of commands history. International Journal of CAD/CAM, 5(1), pp.39-47.

Shah, J.J. and Mathew, A., 1991. Experimental investigation of the STEP form-feature information model. Computer-Aided Design, 23(4), pp.282-296.

Shah, J.J., Anderson, D., Kim, Y.S. and Joshi, S., 2001. A discourse on geometric feature recognition from CAD models. Journal of computing and information science in engineering, 1(1), pp.41-51.

Sharp, G.C., Lee, S.W. and Wehe, D.K., 2002. ICP registration using invariant features. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1), pp.90-102.

Shih, C.H. and Anderson, B., 1997, May. A design/constraint model to capture design intent. In Proceedings of the fourth ACM symposium on Solid modeling and applications (pp. 255-264). ACM.

Shilane, P., Min, P., Kazhdan, M. and Funkhouser, T., 2004, June. The princeton shape benchmark. In Proceedings Shape Modeling Applications, 2004. (pp. 167-178). IEEE.

Shvaiko, P. and Euzenat, J., 2005. A survey of schema-based matching approaches. In Journal on data semantics IV (pp. 146-171). Springer, Berlin, Heidelberg.

Simon, H.A., 1956. Rational choice and the structure of the environment. Psychological review, 63(2), p.129.

Singer., P. 2019. Handling huge matrices in Python. [online] Available at: https://medium.com/@ph_singer/handling-huge-matrices-in-python-dff4e31d4417. [Accessed 17 January 2018].

Sipiran, I. and Bustos, B., 2011. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. The Visual Computer, 27(11), p.963.

Skelly, Luke J., Stan Sclaroff., 2007, Improved feature descriptors for 3D surface matching. Optics East 2007. International Society for Optics and Photonics, (2007).

Slansky, D., Interoperability and Openness across PLM: Have We Finally Arrived? sl: ARC Strategies, OCTOBER 2005. Copyright© ARC Advisory Group.

Song, I. and Han, S., 2010, September. Parametric CAD data exchange using geometry-based neutral macro file. In International Conference on Cooperative Design, Visualization and Engineering (pp. 145-152). Springer, Berlin, Heidelberg.

Sorkine-Hornung, O. and Rabinovich, M., 2017. Least-Squares Rigid Motion Using SVD. Computing, 1, p.1.

Soumaya El Kadiri & Dimitris Kiritsis (2015) Ontologies in the context of product lifecycle management: state of the art literature review, International Journal of Production Research, 53:18, 5657-5668, DOI: 10.1080/00 207 543.2015.1052155

Souza, M.C.F., Sacco, M. and Porto, A.J.V., 2006. Virtual manufacturing as a way for the factory of the future. Journal of Intelligent Manufacturing, 17(6), pp.725-735.

Stark, R., Krause, F.L., Kind, C., Rothenburg, U., Müller, P., Hayka, H. and Stöckert, H., 2010. Competing in engineering design—The role of Virtual Product Creation. CIRP Journal of Manufacturing Science and Technology, 3(3), pp.175-184.

Steel, J., Drogemuller, R. and Toth, B., 2012. Model interoperability in building information modelling. Software & Systems Modeling, 11(1), pp.99-109.

Steinfield, C., Markus, M.L. and Wigand, R.T., 2011. Through a glass clearly: standards, architecture, and process transparency in global supply chains. Journal of Management Information Systems, 28(2), pp.75-108.

Stephens, C.R. and Waelbroeck, H., 1998. Effective degrees of freedom in genetic algorithms. Physical Review E, 57(3), p.3251.

Subrahmanian, E., Rachuri, S., Fenves, S.J., Foufou, S. and Sriram, R.D., 2005. Challenges in supporting product design and manufacturing in a networked economy: A PLM perspective. PLM, 5, pp.11-13.

Subrahmanian, E., Rachuri, S., Fenves, S.J., Foufou, S. and Sriram, R.D., 2005. Product lifecycle management support: a challenge in supporting product design and manufacturing in a networked economy. International Journal of Product Lifecycle Management, 1(1), pp.4-25.

Subramani, S. Feature mapping, associativity and exchange for feature-based product modeling. PhD thesis, Indian Institute of Science, Department of Mechanical Engineering, Bangalore, India, 2005

Sudarsan, R., Fenves, S.J., Sriram, R.D. and Wang, F., 2005. A product information modeling framework for product lifecycle management. Computer-aided design, 37(13), pp.1399-1411.

Sun, J., Ovsjanikov, M. and Guibas, L., 2009, July. A concise and provably informative multi-scale signature based on heat diffusion. In Computer graphics forum (Vol. 28, No. 5, pp. 1383-1392). Blackwell Publishing Ltd.

Sundar, H., Silver, D., Gagvani, N. and Dickinson, S., 2003, May. Skeleton based shape matching and retrieval. In Shape Modeling International, 2003 (pp. 130-139). IEEE.

Swain, M.J. and Ballard, D.H., 1991. Color indexing. *International journal of computer vision*, *7*(1), pp.11-32.

Szykman, S., Fenves, S.J., Keirouz, W. and Shooter, S.B., 2001. A foundation for interoperability in next-generation product development systems. Computer-Aided Design, 33(7), pp.545-559.

Tan, C.F., Kher, V.K. and Ismail, N., 2013. Design of a feature recognition system for CAD/CAM integration. World Applied Sciences Journal, 21(8), pp.1162-1166.

Tangelder, J.W. and Veltkamp, R.C., 2004, June. A survey of content based 3D shape retrieval methods. In Shape Modeling Applications, 2004. Proceedings (pp. 145-156). IEEE.

Tao, F., Zhang, L., Venkatesh, V.C., Luo, Y. and Cheng, Y., 2011. Cloud manufacturing: a computing and service-oriented manufacturing model. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 225(10), pp.1969-1976.

Tassey, G., 2000. Standardization in technology-based markets. Research policy, 29(4-5), pp.587-602.

Tassey, G., Brunnermeier, S.B. and Martin, S.A., 1999. Interoperability cost analysis of the US automotive supply chain. Research Triangle Institute, Report, (7007-03).

Taylor, A., Marcus, M. and Santorini, B., 2003. The Penn treebank: an overview. InTreebanks (pp. 5-22). Springer, Dordrecht.

Terzi, S., Cassina, J. and Panetto, H., 2006. Development of a metamodel to foster interoperability along the product lifecycle traceability. In Interoperability of Enterprise Software and Applications (pp. 1-11). Springer, London.

Tessier, S. and Wang, Y., 2013. Ontology-based feature mapping and verification between CAD systems. Advanced Engineering Informatics, 27(1), pp.76-92.

Tessier, S., 2011. Ontology-based approach to enable feature interoperability between CAD systems. PhD, Georgia Institute of Technology.

Thomsen, K. 2012. 'Generalized spiral points': further improvement - sci.math | Google Groups. [online] Web.archive.org. Available at: http://web.archive.org/web/20121103201321/http://groups.google.com/group/sci.math/browse_thread/thread/983105fb1ced42c/e803d9e3e9ba3d23#e803d9e3e9ba3d23%22%22 [Accessed 23 Feb. 2019].

Trucco, E., Fusiello, A. and Roberto, V., 1999. Robust motion and correspondence of noisy 3-D point sets with missing data. Pattern recognition letters, 20(9), pp.889-898.

Tuma, A., 1998. Configuration and coordination of virtual production networks. International Journal of Production Economics, 56, pp.641-648.

Umeyama, S., 1991. Least-squares estimation of transformation parameters between two point patterns. IEEE Transactions on Pattern Analysis & Machine Intelligence, (4), pp.376-380.

Vaishnavi, V.K. and Kuechler, W., 2015. Design science research methods and patterns: innovating information and communication technology. CRC Press.

Van Wijk, D., Etienne, A., Guyot, E., Eynard, B. and Roucoules, L., 2010. Enabled virtual and collaborative engineering coupling PLM system to a product data kernel.

Vandenbrande, J.H. and Requicha, A.A., 1993. Spatial reasoning for the automatic recognition of machinable features in solid models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(12), pp.1269-1285.

Venuvinod, P.K. and Wong, S.Y., 1995. A graph-based expert system approach to geometric feature recognition. Journal of Intelligent Manufacturing, 6(3), pp.155-162.

Vranic, D. and Saupe, D., 2001. 3D shape descriptor based on 3D Fourier transform. In EURASIP (pp. 271-274).

Vranic, D.V. and Saupe, D., 2002. Description of 3D-shape using a complex function on the sphere. In Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on (Vol. 1, pp. 177-180). IEEE.

Vranic, D.V., 2003, September. An improvement of rotation invariant 3D-shape based on functions on concentric spheres. In Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on (Vol. 3, pp. III-757). IEEE.

W3.org. 2016) OWL/Implementations - Semantic Web Standards. [online] Available at: https://www.w3.org/2001/sw/wiki/OWL/Implementations [Accessed 5 Jun. 2018].

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Hübner, S., 2001, August. Ontology-Based Integration of Information-A Survey of Existing Approaches. In OIS@ IJCAI.

Wang, B., Li, D., Hu, K. and Zhang, H., 2010, May. Shape similarity assessment approach for cad models based on graph edit distance. In Short Paper Proceedings of EUROGRAPHICS 2010 Workshop on 3D Object Retrieval(pp. 13-16).

Wang, E. and Kim, Y.S., 1998. Form feature recognition using convex decomposition: results presented at the 1997 ASME CIE Feature Panel Session. Computer-Aided Design, 30(13), pp.983-989.

Wang, J. and Wang, H., 2014. Product Design Ontology System Based on FBS-API. JCP, 9(3), pp.543-550.

Wang, M., Lu, Z., Li, H. and Liu, Q., 2015. Syntax-based deep matching of short texts. arXiv preprint arXiv:1503.02427.

Wang, X., 2012. Development of an Interoperable Cloud-based Manufacturing System (Doctoral dissertation, ResearchSpace@ Auckland).

Wang, Y. and Nnaji, B.O., 2004. UL-PML: constraint-enabled distributed product data model. International Journal of Production Research, 42(17), pp.3743-3763.

Wang, Y. and Nnaji, B.O., 2006. Document-driven design for distributed CAD services in service-oriented architecture. Journal of Computing and Information Science in Engineering, 6(2), pp.127-138.

Wu, Z. and Palmer, M., 1994, June. Verbs semantics and lexical selection. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics (pp. 133-138). Association for Computational Linguistics.

Xiong, H., Szedmak, S. and Piater, J., 2013, June. A study of point cloud registration with probability product kernel functions. In 3D Vision-3DV 2013, 2013 International Conference on (pp. 207-214). IEEE.

Yang, Q. Z., and Miao, C. Y., 2007, "Semantic Enhancement and Ontology for Interoperability of Design Information Systems," IEEE Conference on Emerging Technologies and Factory Automation, IEEE, New York, pp. 169–176.

Yassine, A. and Braha, D., 2003. Complex concurrent engineering and the design structure matrix method. Concurrent Engineering, 11(3), pp.165-176.

Yeo, I., 2009. Matching methods for semantic interoperability in Product Lifecycle Management.

Yusuf, Y.Y., Sarhadi, M. and Gunasekaran, A., 1999. Agile manufacturing:: The drivers, concepts and attributes. International Journal of production economics, 62(1-2), pp.33-43.

Zachariadis, M., Scott, S. and Barrett, M., 2013. Methodological implications of critical realism for mixed-methods research. MIS quarterly, pp.855-879.

Zaharescu, A., Boyer, E., Varanasi, K. and Horaud, R., 2009, June. Surface feature detection and description with applications to mesh matching. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on(pp. 373-380). IEEE.

Zaharia, T. and Preteux, F.J., 2001, May. 3D-shape-based retrieval within the MPEG-7 framework. In Nonlinear Image Processing and Pattern Analysis XII (Vol. 4304, pp. 133-146). International Society for Optics and Photonics.

Zaharia, T. and Prêteux, F., 2002. Shape-based retrieval of 3D mesh models. In Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on (Vol. 1, pp. 437-440). IEEE.

Zehtaban, L., Elazhary, O. and Roller, D., 2016. A framework for similarity recognition of CAD models. *Journal of Computational Design and Engineering*, *3*(3), pp.274-285.

Zhan, P., Jayaram, U. and Jayaram, S., 2006, July. A Method of Capturing Product Data Semantics by Building Engineering Ontology. In Proceedings of NSF Design, Service, and Manufacturing Grantees and Research Conference (pp. 24-26).

Zhan, P., Jayaram, U., Jayaram, S., 2007, "A Semantic Approach for CAD/CAE Integration", Proceedings of 2008 NSF Engineering Research and Innovation Conference.

Zhan, P., Jayaram, U., Jayaram, S., Kim, O., Zhu, L., 2008, "Knowledge Representation and Ontology Mapping Methods For Product Data In Engineering Applications", Proceeding of 2008 ASME IDETC/CIE Conference.

Zhan, P., Jayaram, U., Kim, O., and Zhu, L., 2010, "Knowledge Representation and Ontology Mapping Methods for Product Data in Engineering Applications," ASME J. Comput. Inf. Sci. Eng., 10(2), p. 021 004.

Zhang, C. and Chen, T., 2001. Efficient feature extraction for 2D/3D objects in mesh representation. In Image Processing, 2001. Proceedings. 2001 International Conference on (Vol. 3, pp. 935-938). IEEE.

Zhang, D.J., He, F.Z., Han, S.H. and Li, X.X., 2016. Quantitative optimization of interoperability during feature-based data exchange. Integrated Computer-Aided Engineering, 23(1), pp.31-50.

Zhang, L., da Fonseca, M.J., Ferreira, A. and e Recuperação, C.R.A., 2007. Survey on 3D shape descriptors. FundaÃgao para a Cincia ea Tecnologia, Lisboa, Portugal, Tech. Rep. Technical Report, DecorAR (FCT POSC/EIA/59 938/2004), 3.

Zhang, Z., Gentile, A.L. and Ciravegna, F., 2013. Recent advances in methods of lexical semantic relatedness–a survey. Natural Language Engineering, 19(4), pp.411-479.

Zhu, K.P., Wong, Y.S., Lu, W.F. and Loh, H.T., 2010. 3D CAD model matching from 2D local invariant features. Computers in industry, 61(5), pp.432-439.

Zhu, L., Jayaram, U., Jayaram, S. and Kim, O., 2009, January. Ontology-driven integration of CAD/ CAE applications: strategies and comparisons. In ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. 1461-1472). American Society of Mechanical Engineers.

Zimmermann, J.U., Haasis, S. and Van Houten, F.J., 2002, January. Applying Universal Linking of Engineering Objects in the Automotive Industry: Practical Aspects, Benefits, and Prototypes. In ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference pp. 65-74). American Society of Mechanical Engineers.

11 Bibliography

12 Ontologies for CAE interoperability

A.1 Introduction

To start, it is helpful to broadly define an ontology with respect to Information Science, which is a formal description of the types, properties and relationship of information within a specific domain; the representation of information structure, semantics and relationships in an ontology preserves a consistent interpretation of data. Guarino gives a useful definition of an ontology (Guarino, 1998). Here, a conceptualisation covers the intended meaning of a formal vocabulary, rather than real-world, ad-hoc usage that language undergoes.

"An ontology is a logical theory accounting for the *intended meaning* of a formal vocabulary, i.e. its *ontological commitment* to a particular *conceptualization* of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualisation) by approximating these intended models"

Ontologies can be separated into two broad categories (Fankam, 2009). The first are storage-oriented ontologies that define the elements and their position within a domain architecture using a common vocabulary, what Fankam *et al* define as a *conceptual canonical model* where a class may only contain a single datum.

The second category of ontologies, or *non conceptual canonical model* allows machine reasoning on the domain content, where elements may be shared between multiple classes. The latter category is used within CAE ontological modelling where the intent is to create interoperability between heterogeneous systems and consequently is of interest to modelling CAE systems. Subsequent references to ontologies are considered to be exclusively of this class.

A.2 Layered ontologies

Guarino describes four distinctive types of ontology, which later give an indication of the limits of their applicability (Guarino, 1998).

A *top-level* ontology may define very fundamental concepts that encapsulate a frame of reference, such as space, time, events, actions, objects. Such a top-level ontology would be distinguished by organising the most basic taxonomy from which other ontologies might then be derived, other terms used are foundation ontology and common sense ontology. Dublin Core, SUMO and CYC are examples of top-level ontologies (Dublincore.org, 2012; Pease, 2018; Reed & Lenat, 2002).

A *domain* ontology is distinguished by representing the subset of objects, with a specific domain while a *task* ontology is distinguished by actions or processes associated within a domain such as medicine or agriculture. An *application* ontology encompasses both concepts and operations associated with a domain. There is a trade-off between the generality of an ontology, as in the breadth of concepts that it addresses, and the utility of this same ontology, or the ability to perform useful reasoning. Gruber refers to this balance as the problem of portability (Gruber, 1993). The drawback of a single ontology approach, where each application shares the same ontology, is the difficulty in specifying a global shared vocabulary sufficiently general to represent diverse domain associations, yet economical enough to be computationally effective (Wache *et al*, 2001).

A.3 The Core Product Model

The *Core Product Model* of the National Institute of Standards and Technology was another project devised to address the deficiencies of STEP models, adding the capacity to represent function and behaviour (or as a concept familiar to engineers, intended behaviour and observed behaviour). NIST augmented this model with a Design-Analysis Integration Model, a Product Family Evolution Model and an Open Assembly model to define assembly, geometric tolerancing, kinematics and engineering analysis structured within the associated relationship hierarchy (Sudarsan *et al*, 2005).

A.4 Product Specification Language, Product Semantic Representation Language

Product Specification Language was developed at NIST as a production process ontology, limited to geometry and related manufacturing processes (Gruniger & Menzel, 2003). Product Semantic Representation Language took the NIST Core Product Model and defined an ontology aimed at semantic interoperability between CAD geometry terminology and Computer Aided Process Planning terminology (Patil *et al*, 2005). This research encoded the ontology syntax in the newly developed DARPA Agent Markup Language, an emerging standard from the field of Semantic Web research (Hendler & McGuinness, 2000).

Dartigues *et al* extended the NIST Core Product Model to provide a sufficient ontology to allow interoperability between a CAD software (Pro-Engineer) and a Computer Aided Process Planning software (PART) using the Knowledge Interchange Format (KIF). The researchers note that a further challenge to domain interoperability arises from the use of heterogeneous languages to defining ontologies. This research demonstrated that a feature entirely described in semantic terms could be transferred from one domain to another without a geometrical model (Dartigues *et al*, 2007).

Chungoora *et al* describe a heavyweight manufacturing ontology based on a foundation layer modelled with a Common Logic based formalism. The research intentionally captures the semantics of multiple domain perspectives of a common product artifact (Chungoora & Young, 2008). OntoSTEP was introduced to add descriptive semantic references to STEP Application Protocol Models, but did not capture the conceptual data associated with product models (Barbau *et al*, 2011; Krima *et al*, 2012).

A.5 Ontologies based on the semantic web, OWL, RDF

The readability, extensibility and self-documenting properties of the Extensible Markup Language, or XML, gave rise to a family of languages to extend the scope of internet web pages. The so-called semantic web incorporates a layer of self-description that allows machine reasoning. An embedded semantic page description allows automated searches across internet web pages that can contextualise page data without human intervention. Resource Description Framework is one of these languages that allow in-line definition of web page metadata in a machine-interpretable format. Data can be ordered within classes allowing the expression of subject – predicate – object relationships. For example,

subject: Jack, *predicate*: isBrotherOf *object*: Jill.

Web Ontology Language or OWL takes this metadata triplet further, allowing axioms to specify relationships between RDF annotations. OWL retains the advantages of XML in being relatively flexible, readable and reusable (McGuinness & Harmelen, 2004; Berners-Lee *et al*, 2001). OWL also enjoys widespread web adoption with a correspondingly mature tool-chain, consequently there have been numerous research prototypes integrating Computer Aided Engineering domain semantics using ontologies based on OWL (W3.org, 2016). What follows are research developments that give an indication of the scope of these efforts, for an exhaustive catalogue of CAE interoperability research using OWL ontologies see (Soumaya *et al* 2015; Qin *et al* 2017).

Yang and Miao create an ontology for the semantic integration of a prototype Design Information System, essentially an architecture to allow interaction between different processes of an engineering design cycle (Yang, Miao, 2007). The prototype integrates AutoCAD CAD software along with OptiCAD optical CAD software. OptiCAD and AutoCAD are both AutoDesk products, there is no published detail of CAD interoperability.

Mostefai *et al* create a prototype ontology that encompasses part detail, manufacturing process and assembly design (Mostefai *et al*, 2005). This ontology is based on feature representation, describing both geometric and semantic aspects.

Domain Independent Form Feature, or DIFF, is an ontology structured around a geometric classification of CAD form features defined by surface faces (Gupta & Gurumoorthy, 2008). The geometric characteristics of features used in proprietary CAD systems are abstracted as classes of faces inherent within the feature, further distinguished by the interrelation between adjacent faces. This abstract feature representation is adapted from Subramani and used to represent the construction history of CAD models (Subramani, 2005). The DIFF ontology research identifies several problems unique to semantic interoperability between CAx systems that can be identified and resolved using a non-proprietary geometric representation of features. These problems can be summarised as,

1. Different syntactic labels referencing identical feature geometries
2. Different construction histories that generate identical model geometries.

Ahmed and Han create a platform to integrate CAD/CAM operations using a common OWL ontology to augment parametric CAD data with non-parametric Geometry, Dimensioning and Tolerance PMI data and machining data (Ahmed and Han, 2015). This platform employs the macro-parametric approach, or MPA, to define a set of neutral modelling commands for the purpose of CAD interoperability (Choi, Han, Mun, 2002). The macro-parametric method is previously described in detail in Chapter 3.8.

He *et al* propose an OWL based ontology to facilitate collaborative part and tooling development, a process requiring interaction between the domains of design, manufacturing process planning and tooling development (He *et al*, 2015).

These ontologies have been developed to test the potential for knowledge sharing between computer aided engineering domains. As traditionally a product model would have to be reconstructed or adapted to be transferred to a different design cycle process, the ability to share a single model between different domains offers better efficiency and data integrity (Gupta & Gurumoorthy, 2008). At a higher level, ontologies may be used to standardise the interpretation of semantic information used within application syntax of heterogeneous domains (Altidor, 2009).

A.6 Top-down ontologies and interlingua

Anticipation of the required scope of an upper ontology is a difficult proposition, as it presupposes future development of applications. One example of a top-down ontology that has been subsequently adopted by commercial vendors is the Egenhofer and Franzosa formalisation of topological relationships that is supported by all mainstream Geographical Information Services software (Egenhofer and Franzosa, 1991). It is also incorporated within the relevant standard, the Simple Feature Specification of the Open Geospatial Consortium. This contrasts with the domain of Computer Aided Engineering where most innovation has been initiated by commercial vendors and no top-down ontologies or published semantic models are used to represent data.

In the absence of an adopted top-down CAD or CAE ontology, researchers have explored methods to generate or extract a shared ontology from heterogeneous domain ontologies. One alternative is a *bottom-up* approach, creating a bridging ontology or *interlingua* based on a mapping between CAD systems (Uschold & Gruniger, 1996).

Zhan and Kim broach the requirement of mapping similar domain ontologies (Zhan & Kim, 2010). The research identified a means to map application ontologies rather than attempting to define a unique ontology that defines all use cases and domains. The proposition takes an overarching *General Domain Ontology* that specifies the broadest of concepts shared by all conceivable sub-ontologies. From this universal ontology they model *Domain Specific Ontologies* that encapsulate the semantics of terminology within domains such as product design and assembly simulation. A third layer of more specialised ontologies, *Application Specific Ontologies* inherit the concepts of the Domain Specific and General Domain ontologies. These hierarchical ontologies can be seen to correspond with the Gerbino definition of a top-level ontology, a domain ontology and application ontologies respectively. The example applications are given as instances of commercial software ontologies and covered in greater detail in Zhu (Zhu *et al*, 2009).

Zhan proposes several techniques for mapping the semantics between these application specific ontologies using a set of heuristics. Similarity is based on the proportion of syntactic matches within an element. Zhan presents an example that defines geometrical matching between elements, but these matches are based on syntactic matching of the features described in the referenced CAD ontologies rather than a comparison of geometry. Each atomic element within these application specific ontologies is defined as a *Basic Design Entity*, a triplet capturing attributes of function and behaviour (similar to the NIST CPD ontology). The rules of establishing an entity match are derived from the relationship to known matches, whether this is matches of associated attributes or a matching position within both ontologies under scrutiny.

Jiavy *et al* examine the use of statistical methods to determine similarity between commonplace ontologies used within the building industry, (the Industry Foundation Class (IFC), CIMsteel Integration Standards (CIS/2), OmniClass Construction Classification System). The ontologies are considered as text corpora that can be compared under Jaccard similarity coefficient and Cosine similarity measures. This research also experiments with a Market Basket Analysis to derive a comparative measure between ontology elements. This approach is functionally identical to the comparative assessment of associated ontology element attributes used in the matching rules of Zhan (duck typing).

Ciocoiu *et al* adopt the top-down layered ontology approach described in Zhan *et al*, but append *bottom-up* ontologies in cases of pre-existing heterogeneous application ontologies which are to be integrated within this layered ontology structure. In a top-down layered ontology structure, the application ontologies inherit concepts from the upper ontologies. Real-world applications require that predefined ontologies or data models are bridged with a shared upper ontology. The mappings, or *axioms*, that define identical or related concepts between a source ontology and the shared upper ontology require the intervention of a domain expert.

Seo *et al* state that the macro-parametric approach, or MPA, captures inadequate semantic detail to allow automated translation between CAD systems. The OntoSmart system described in the research uses the macro-parametric CAD script input that is translated to a target CAD script, but instead of using the routines described in Cho *et al*, the OntoSmart uses ontology reasoning to achieve the same outcome. Note that the OntoSmart system uses an F-Logic based ontology rather than the more widespread OWL basis. Seo *et al* use a similar ontology structure to determine semantic matching between features. Unlike prior research that matched feature semantics via syntactic mapping between parameter labels (macro-parametric), a shared base ontology mapped or *bridged* to local CAD ontologies captures sufficient semantic detail to allow recognition of semantically similar features via pre-determined rules, described as *axiom bridges* (Seo *et al*, 2005).

Wang and Nnaji describe an XML/RDF triplet-based domain ontology that captures the semantics of features within different CAE programs to allow mapping and machine reasoning.

Jayaram *et al* describe a method that extracts product model metadata via a CAD program API and creates a corresponding semantic model by matching this metadata with concepts from a pre-existing ontology.

Eddy *et al* describe a process to extract the semantic content of a CAD model. A commercial Product Lifecycle Modelling software can derive a Bill of Materials from the CAD model, which comprises a detailed textual description of model features. Part relationships are identified by the vocabulary used.

Altidor *et al* develop algorithms to automate mappings between semantic descriptions of CAD features (Altidor *et al*, 2009, Hanayneh *et al*, 2008). A CAD feature is represented as a directed, labelled and attributed graph capturing explicit and implicit parameters. Further layers of description such as topological relationship of the feature to the CAD model and the individual attributes of the feature are combined to describe a *Hybrid Semantic Feature*. Two forms of matching are used, graph matching and type matching to compare semantic structure and labels between hybrid semantic features. The graph matching uses an unspecified subgraph isomorphism algorithm over the feature graph and those within another CAD feature library.

Altidor describes the conventional function mapping between source and target CAD programs as *static mapping*, where a single hybrid semantic feature instance in the source CAD is mapped to one or more equivalent instances in the target CAD class-level library. A system of *dynamic mapping* is proposed where a hybrid semantic feature instance is dynamically mapped to a feature equivalent within the set of target CAD features. Every time a translation is made for an individual feature, the mapping is determined by a search process rather than a look-up process. This approach has the primary advantage of only requiring access to the target CAD function library.

Tessier and Wang use four distinct categories to define CAD feature elements within an ontology, parameter attributes, reference attributes, feature type and geometric surface data (B-rep data). The feature type stores previous successful function matches. The other three categories are of interest here. While parameter attributes are defined as the explicit parameters referenced in the feature API function description, reference attributes are defined as the prior selections, datums and other references that are implicitly required of a function operation. Geometric surface data is included as a feature representation independent of the syntactic description of parameter and reference attributes. An independent geometry description allows verification of feature matches in instances where the parameter and reference attributes are insufficient for unambiguous mapping. This geometric description is derived from a rule-based analysis of basic geometric features made accessible via an API. The derivation of feature definitions is similar to rule-based automated feature recognition that depend on formalised descriptions of geometric rules associated with particular features (Henderson & Anderson, 1984). This approach describes *dynamic* matching between feature instances with the feature library of the target CAD program, rather than the customary static matching requiring access to the class-level libraries of both CAD systems undergoing a feature mapping.

Existing software has an internal architecture that may not conform to a published ontology. A fixed ontology hampers innovation as there is no opportunity to experiment with novel taxonomies of features. There is no defined optimal set of features that might be universally adopted, consequently an ontology sufficiently general to allow the representation of disparate feature hierarchies would be of limited utility.

The allure of defining CAE features as ontologies stems from the promise of formal reasoning to automatically determine the relationships between heterogeneous CAE systems. Though many theoretical efforts exist, only a few subsets of commercial CAD programs have been translated into ontologies (Qin *et al*, 2017). Defining CAD features within an ontology is labour intensive, limiting the practicality of this approach.

A.7 Ontology mapping: OWL DL

Semantic mapping between CAD ontologies generally uses one of three methods, OWL inference, SWRL inference or semantic mapping (Shvaiko & Euzenat, 2005).
In the case of OWL, the most expressive variant OWL DL is commonly used. Of the other two varieties of OWL, OWL Lite is limited in expressiveness while OWL Full is undecidable and does not work with reasoners. Inference reasoning in OWL DL is based on the *satisfiability decision algorithm* or *Tableau algorithm.* A hypothetical feature class may be further defined using a number of rules or axioms. For example,

"a hole feature has an axis of symmetry"
"a hole feature has only one axis of symmetry"

An OWL DL ontology allows the classes and properties to be posed as propositional logic theorems and tested for veracity. A tableau algorithm will construct each class instance as a node within a tree and test axiom propositions until it encounters either a false result or completes. These axioms may be tested on the classes and properties of CAD domain ontologies that describe the set of features of specific CAD programs. If both feature classes are semantically equivalent, all instances of the source class are recreated within the target class.

To describe this process further it is helpful to adopt the terminology of the OntoSTEP OWL ontology, used to model STEP EXPRESS CAD data (Krima *et al*, 2009). An ontology has a set of classes, these classes correspond to instances of objects or artefacts, in the case of OntoSTEP, the classes correspond to STEP *entities* which represent model instances. A class will normally have an identifying name and an associated description. A class may also have a subclass, where another class inherits the characteristics of the class, but possessing a more specific definition. Within EXPRESS an *attribute* defines the relationship between entity and an instance of data, or a relationship defined between entities. A useful ontology will possess a set of *properties,* for example OntoSTEP ontology properties map the STEP attributes. These properties also contain a description and a name associated with the attribute. Where this becomes relevant, is the ability of OWL reasoners to distinguish semantically matched classes, but not matched properties. In other words, it is possible to compare instances, but not qualities**.**

The Common Design Feature Ontology (CDFO) of Kim *et al,* Assembly Relation Model (ARM) of Abdul-Ghafour *et al*, Ontology-Based Information Integration and Sharing (OBIIS) framework of He *et al* the Three-Branch Hybrid Feature Model of Tessier and Wang and the PRO-AO, VADE-AO and CAT-AO layered ontologies of Zhu *et al* all adopt SWRL reasoning (Kim *et al*, 2006; Abdul-Ghafour *et al*, 2007; He *et al*, 2015; Tessier & Wang, 2013; Zhu *et al*, 2009).

As mentioned above, OWL DL can test for semantic equivalence between classes, but not properties. Deriving valid axioms that might encapsulate the essence of CAD features or models is a difficult task, requiring skilled intervention and an in-depth knowledge of both CAD feature concepts and propositional logic. Consequently several researchers address the property shortcomings of OWL DL using *Semantic Web Rule Language* or SWRL (Horrocks *et al*, 2004). SWRL is an extension of OWL representation that conveniently allows ontology properties to be assessed for semantic equivalence.

A.8 OWL SWRL

SWRL retains the same drawback as OWL DL, axioms used to infer semantic equivalence must be carefully conceived by a skilled practitioner expert in conceptual definitions of

CAD features. OWL descriptions logic uses relatively simple terminological axioms, or so-called *T-box*, generally involving set membership or equivalence relations, e.g. “banana is a kind of fruit”. SWRL, on the other hand, uses assertional reasoning, requiring *A-box* axioms (Horrocks & Tobles, 2000). Asserting facts from ontology concepts adds a level of difficulty in determining reliable universal truths about CAD features, which are in themselves more of a conceptual commercial construct than a consistently defined system. Sanfilippo and Borges exhaustive review of the CAD features literature draws several conclusions that make ontological models of features difficult (Sanfilippo & Borges, 2016).

The last method used to match ontologies is semantic matching. The labels used in ontologies are compared for semantic or syntactic similarity. Patil *et al* describes a method that can replace exact equivalences (Patil *et al*, 2005).

A.9 Limitations of ontologies and semantic inference

Semantic feature representations and explicit research ontologies have not been retrospectively adopted by commercial CAD vendors. The semantic models, ontologies or interlingua described propose specific architectures, or prescriptive concepts which are then verified as a necessarily limited research experiment.

There is no general theory supporting feature representation. Ad hoc feature design has led to a balkanisation of specifications. What feature representation exists, tends to have unspecified implicit properties inherent to the domain in which they are defined. Feature-based models describe the parameters and constraints that define features, but they do not describe a semantic quality that defines a feature. Nor do ontology models define interrelation of features within a model in a non-geometric sense, a hole might exist within a model, but is only anchored in place by geometrical constraints.

Objects referenced within CAD & CAE do not have universally agreed semantic description, moreover the different terminologies are associated with geometrical identity rather than a pure language description. Research efforts focus on the ability of ontologies to structure descriptive terms, but there is no equivalent approach to create non-verbal taxonomies of geometries. Mapping semantic similarity is only a partial solution to a requirement to map a geometric similarity.

12 Ontologies for CAE interoperability

B 3D Shape matching methods overview

B.1 3D Shape matching introduction

The challenge of determining a geometric similarity between virtual three dimensional models has attracted different approaches within different fields. Point cloud registration, image recognition, model retrieval and CAD geometry feature extraction all share a similar requirement to recognise a match between shape geometry. The nature of the available data from which a comparison is drawn influences the methodologies used, images have pixels of differing intensities or colours, point clouds are arrays of 3-D point coordinates defining a surface, most 3-D models are models defined by triangular facets, CAD models may have several means to define boundary model surfaces. Central to the task of shape recognition is the ability to detect similarity despite changes in orientation, scale or position of shapes under comparison. This task may be further complicated by recognising shapes that have been distorted, or matching incomplete shapes.

A robust method to detect similarity between two CAD surface boundary models must be insensitive to differences of model orientation, position or scale. This can be described as a rigid body transformation problem (the more complex case of recognising a deformed instance of a non-rigid body is not required for this application).

Prior research that has compared CAD models from different software for similarity has relied on an export to a neutral format, resulting in difficulties encountered with the uneven commercial implementations of STEP AP203 Edition 2 (McKenzie-Veal *et al,* 2010). Other research is reliant on an existing CAD API that can export the model surfaces (Tessier & Wang, 2013).

Three-dimensional shape descriptors are an active research area, drawing methods from related disciplines such as image matching, point cloud registration, 3D shape retrieval. Tangelder *et al* have categorised the methods within four basic approaches as follows (Tangelder *et al,* 2007; Kazmi *et al*, 2013).

View-Based	the shape description is extracted from multiple 2D images taken of the 3D shape.
Histogram-Based	the discriminating features of the shape are represented as a unique histogram signature comprising a vector of numerical identifiers.
Transform-Based	the 3D shape is transformed to a non-geometric mathematical domain where the defining characteristics are unaffected by the orientation or position of the geometric shape.
Graph-Base	a 3D shape is transformed to a simplified topological representation.

Table 10: shape matching method categories

It is helpful to give an overview of the representative techniques within each of this taxonomy, but distracting to attempt an exhaustive categorisation of this broad field. There are several surveys that cover 3D shape matching and shape description (Kazmi *et al*, 2013; Cardone *et al*, 2003*;* Zhang e*t al*, 2007).

B.2 View-Based shape matching methods

View-Based shape matching methods benefit from extensive research matching 2D images and from the ability to operate without explicit reference to the virtual shape model data. Most methods take an image of the shape model from several angles and combine them to form a characteristic shape descriptor.

The *Light Field Descriptor* described by Chen *et al* which takes the shape silhouettes from images taken at ten evenly spaced angles and characterises them via a *Zernike moment descriptor* or *Fourier descriptor* to provide an orientation-invariant signature (Chen et al, 2003).

Adaptive Views Clustering uses 320 images represented as *Zernicke moments* and then creates a series of K-means clusters of these Zernicke moments defined with a different number of clusters. The number of clusters that represent the optimal level of shape information are selected via a Bayesian Information Criteria calculation. As each shape is represented by a the optimal number of image signatures, the researchers specify a probabilistic Bayes matching approach to rank shape similarity according to the most similar of associated image signatures.

The *Compact Multi-View Descriptor* takes two image descriptors, the shape outline and the pixel brightness intensity and generates three image descriptors based on the rotationally invariant Polar-Fourier transforms, Zernicke moments and *Krawtchouk moments*. The orientation of each shape model is normalised using Principle Component Axis alignment, allowing direct matching between the images to sum to a match probability.

Ohbuchi *et al* describe an image-based 3D shape matching algorithm based on the well-known *SIFT feature matching* algorithm used in image matching applications (Ohbuchi *et al,* 2008). This method is based around the Bag-of-Features method used in semantic matching. Multiple images are taken from a shape normalised in scale and pose orientation. The Scale Invariant Feature Transform identifies image regions that can be identified at different scales and image rotations. This gives a number of features such as corner points or distinctive marks. These features are clustered using a k-means technique and the SIFT features are ordered into histogram bins depending on their proximity to the cluster barycentres. These barycentres must be pre-computed. Using vector quantisation terminology, the histograms are vectors that identify the shape features. The distance between the vectors representing the shape models is calculated using the *Kullback-Leibler divergence* measure.

Gao *et al* present *Spatial Structure Circular Descriptors* as a projective image-based 3D shape comparison method (Gao *et al*, 2010). A 3D shape is pose normalised using PCA, each surface point is projected to a minimal bounding sphere. This sphere is then mapped to a flat circular image. As several surfaces of the 3D shape may be coincident with a ray projection from the sphere origin to the sphere surface, multiple surfaces data are mapped to a separate circular images. These images are segmented into regions for the purposes of creating bins suited to histograms. As each shape will produce several histograms, the best matches between shapes are detected using the *Munkres-Kuhn* method. For a more exhaustive catalogue of View-Based shape matching methods see Liu (Liu, 2012).

B.3 Histogram-Based shape matching methods

Histograms are a common representation of a unique shape signature that allows rapid comparison with other shapes. Descriptive characteristics are sorted by a quantisation of a property range (commonly known as *bins*), or by distinct categories to form a numerical signature intended to be independent of shape position, orientation and scale. Histograms may also be characterised as feature vectors within a dimensional space defined by the number of bins, or quanta (Bustos *et al*, 2005).

B.4 Spatial map-based methods

The *spatial map-based approach* segments the shape into regions in order to generate a histogram from the proportion of each region occupied by shape volume. Ankerst *et al* describe a method that partitions the shape model within an encompassing sphere internally divided by radial and angular subdivisions (Ankerst *et al*, 1999). The limited resolution of the sphere subdivisions becomes apparent using a Euclidean distance metric between the histograms generated by the approach. A quadratic form distance function is used to circumvent this issue, essentially weighting the values of sectors in close proximity.

Vranic *et al* record shape model boundary intersections with a series of concentric spheres around the shape barycentre (Vranic & Saupe, 2002), (Vranic, 2003). These points are computed to determine the discrete Legendre transforms that comprise a Spherical Fourier Transform, allowing the shape to be approximated as a spherical harmonic function (Healy *et al*, 2003). While this method is insensitive to shape orientation, it is also incapable of discriminating shape distortions that preserve radial dimensions.

Ohbuchi *et al* describe an approach that measures the moment of inertia of a slice of a shape taken along it's longest axis of inertia (Ohbuchi *et al*, 2002). Each vertex is assigned a weight and the eigenvalues of the derived covariance matrix determines the principle axis of inertia. The weighted vertices of the model are divided into sections along this axis, each of which yield a characteristic inertial moment and barycentre. These values give a histogram.

B.5
Local feature based methods

The *3D Shape Spectrum Descriptor* uses the instances of a range of surface geometry features to define a histogram (Zaharia & Prêteux, 2001). These features are defined as the properties of surface curvatures along two axes, or the *Koenderink shape index* (Koenderink & Van Doorn, 1992). This ranges from a local surface indentation at a point, to a groove, a saddle inflection, a ridge to a surface peak. A polygon mesh surface is smoothed by a parametric continuous representation, regions that conform to the shape index are identified and are accumulated in bins defined by a division of the shape index.

The *2D Hough transform* may map a line to a curve via a radius and angle, in the 3D equivalent, this is achieved via spherical coordinates. The *3D Hough Transform* polls a shape model for polygon mesh planes using a radius, azimuth and elevation basis centred at the shape model barycentre (Zaharia & Prêteux, 2002). Dividing a spherical sampling volume into meridian and parallel regions creates unequal volumes, a 3D Hough transform that compensates for this by re-sampling along the three axes of inertia of the model is computationally expensive. A Canonical 3D Hough Transform is described that maps the spherical sampling partitions to an octagonal partition schema.

B.6 Point signatures

Surface curvature methods define local features, other methods determine non-local general features. Zhang and Chen describe a means to tessellate 2D and 3D polygon mesh models to calculate volume (Zhang & Chen, 2001). These discrete volumes can be integrated to derive a moment characteristic that is independent of shape orientation. The Fourier transform of these volumes provides a distinct set of coefficients that also serve as a unique shape signature. The second order moments may also be used to derive the principle axes of a shape, allowing for pose normalisation.

Paquet *et al* describe general shape descriptors that identify 2D or 3D shapes (Paquet *et al*, 2000). A *bounding box* is the smallest box that encloses a shape, this data and the absolute position and orientation of this box constitute a coarse discrimination between shapes. Further discriminants may be derived from the volume of the box occupied by the shape. A cord technique that describes a ray from the shape barycentre to the centroid of shape surface polygons gives a set of angles that can form a signature histogram. The cords concept also allows the moment of the shape volume to be calculated. A wavelet transform of these cords vectors gives a set of signature coefficients.

Vranic and Saupe extend the idea of a bounding box, subdividing it into cubes or voxels (volumetric pixels) and retaining those voxels that intersect the shape model (Vranic & Saupe, 2001). The 3D discrete Fourier transform of the voxels representing the shape model give a unique set of coefficients to form a characteristic histogram.

B.7 Variant models

Dutagaci *et al* generate a voxel representative of a shape model and trial a Discrete Fourier Transform alongside a Radial Cosine Transform function to minimise the influence of shape orientation on the derived representation (Dutagaci *et al*, 2005). A spectral energy representation of shape boundaries is also tested to determine improvement over the discrete binary voxel model. This Radial Cosine Transform of the 3D function $v(x)$ is:

$$V_{RCT}(m) = \sum_x v(x)\phi_m(|x|)$$

where $\phi_m(r)$ are radial cosine transform basis functions defined as follows:

$$\phi_m(r) \begin{cases} 1, & \text{if } r = 0 \\ 2\cos(\pi m r) & \text{otherwise} \end{cases} \text{ for m} = 0,\ 1,\ 2,\ \ldots \text{M}$$

B.8 Shape distribution signatures

Burel and Hénocq decompose a point cloud object into eigenvectors of the angular momentum to generate invariant tensor descriptions. If point clouds are represented as angular momentum, eigenvector decomposition can be used to represent the clouds as spherical harmonic invariants (Burel & Hénocq, 1995).

Sadjadi and Hall describe an invariant moment features that create rotationally invariant signatures based on the enclosed space within a point cloud or polygon mesh (Sadjadi & Hall, 1980).

Osada *et al* explore a range of shape functions that return a value within a single parameter range, allowing a characteristic histogram shape signature to be generated (Osada *et al*, 2002). The functions tested are chosen to provide a robust metric for shapes represented by a polygon mesh, a representation that may suffer from discontinuities, duplicate polygons, missing polygons, and irregularly sampled meshes. These unintuitive shape functions embody general statistical properties of the mesh, for example the angle between three random vertices on the mesh, or the root of the area between three random surface vertices. The discriminative quality of each metric is tested by adding noise to the test models. The best overall measure is found to be the distance between two random sampled vertices on the surface.

B.9 Curvature based descriptors

Two common curvature based surface descriptors are defined by the *principle curvatures* at a point, namely the curves of maximum and minimum curvature that can be projected on to a surface normal through the point. While the principle curvatures are defined by the Euclidean space that the surface is embedded in (or possessing an extrinsic quality), the product of these curvatures, the *Gaussian curvature* is independent of this space (or possessing an intrinsic quality). This gives Gaussian curvature the quality of invariance under isometric transforms, hence it's utility in defining feature descriptors.

Besl and Jain take the Gaussian curvature, κ , and the mean value of the maximum and minimum curvatures (H) to formulate a metric that identifies a surface point region as a ridge or groove, or as a saddle ridge, or saddle valley, a concave or convex ellipsoid (Besl & Jain, 1986).

Koenderink and van Doorn take a different combination of the principle curvatures to define a shape index, which describes the curve inflection type within a single parameter S (Koenderink & Van Doorn, 1992). S varies from concave $[-1 < S < -0.5]$, to hyperbolic $[-0.5 < S < 0.5]$, to convex $[0.5 < S < 1]$. S, the shape index, is defined as:

$$S = \frac{2}{\pi} \arctan \frac{\kappa_2 + \kappa_1}{\kappa_2 - \kappa_1} \quad \text{where } \kappa_1 \geq \kappa_2$$

The degree of curvedness, C, is represented in a second expression:

$$C = \sqrt{\frac{\kappa_1^2 + \kappa_2^2}{2}}$$

B.10 Spin images

The problem of shape orientation within to Euclidean reference frames has led to the development of local reference frames generated on geometric shapes. The Spin Image (Johnson & Hebert, 1999, Johnson, 1997) is the best known instance. A spin-image is an image that acts as a local vertex signature or a local basis. It is composed of the individual images of vertex points encountered by a plane spun around the axis of the normal vector passing through the vertex. The density of accumulated points on this revolved plane are then coded as pixel-wide histogram bins, the darkness of an image pixel corresponds to the accumulated point density. Each of these local point spin-images may be readily compared with point images from point spin-images on a comparative shape. This is similar in concept to comparing two panorama images to determine if they were taken from the same spot. Efficient image matching requires image compression, Principle Component Analysis is used to create a compressed representation of multiple images that can be directly compared.

In the search for rotationally-invariant shape discrimination, Liu *et al* derive a coordinate system based on surface characteristics (Liu *et al*, 2006). Spin images are used as local descriptors and evenly sampled over the surfaces of all models to be compared. These spin images are clustered into a smaller set of representative local descriptors. The Osada shape distribution measure between two surface points is augmented with coordinates relative to the nearest generated local descriptors, avoiding the orientation sensitivity of absolute euclidean coordinates.

B.11 Integral volume

An alternative approach to determining intrinsic surface curvature descriptors is to sample the proportion of shape volume captured within a sphere at a point.

Gelfand *et al* describe a method based on the determination of volume within a sphere centered on a point (Gelfand *et al*, 2005). An integral volume descriptor is defined at each vertex p on the model as described:

$$V_r(\mathbf{p}) = \int_{B_r(\mathbf{p})\cap\overline{S}} \mathrm{d}x$$

The integration kernel $B_r(\mathbf{p})$ is a sphere of radius r centered at vertex point $\mathbf{p}$.

$\overline{S}$ represents the model surface boundary, so that $V_r(p)$ is the volume enclosed by the intersection of the interior of the sphere with the model surface. The volume of each surface point is aggregated within histogram bins, the least common value of volume associated with a point is selected from these bins. Nearby point values are winnowed from this point to avoid indistinct feature point sets. The scale-space of each point is readily adjusted by varying the size of the sphere sampling the point volume. The described method incorporates data for feature points matched at varying scales.

One approach examines the deviation of a mesh vertex from it's surrounding neighbours. The difference in the position of a vertex from the mean value of it's immediate neighbouring vertices gives an intrinsic surface descriptor. The *MeshDOG* feature detection described by Zaharescu *et al* convolves a surface function (possibly curvature or texture) with a radially symmetrical Gaussian kernel across the ring of vertices surrounding the vertex undergoing evaluation (Zaharescu *et al*, 2005). This function is applied to successive concentric rings of vertices, which is then subtracted from the value of the previous inner ring to form a “Difference of Gaussian” scale space representation. All points are tested for this measure and those over a threshold value are extracted. This set of points is further filtered to find those that exhibit corner characteristics. This is determined using the Hessian operator and setting a threshold ratio for the difference of the minimum and maximum eigenvalues.

The *Harris operator* is a simple and effective point detector that captures distinctive pixels from an image (Harris & Stephens, 1988). This edge and corner detector smooths the pixel intensity contrast around the source pixel with an analytic expansion similar to the autocorrelation function. Using the notation of Sipiran and Bustos, the difference in overall intensity $e(x, y)$ of a region under analysis W, defined as a Gaussian function is derived from small shifts of $(x_i + \Delta x, y_i + \Delta y)$ around x_i, y_i as follows:

$$e(x,y) = \sum_{x_i,y_i} W(x_i,y_i) \left[I(x_i + \Delta x_i, y_i + \Delta y_i) - I(x_i,y_i) \right]^2 \tag{HO1}$$

The window W is defined as a Gaussian function and $I(x_i, y_i)$ describes the image function.
A Taylor expansion to first order terms gives an autocorrelation patch matrix with the subscripted I values representing the partial derivatives.

$$\begin{aligned} e(x,y) &= [\Delta x \Delta y] \begin{bmatrix} \sum_{x_i,y_i} W.I_x^2 & \sum_{x_i,y_i} W.I_x.I_y \\ \sum_{x_i,y_i} W.I_x.I_y & \sum_{x_i,y_i} W.I_y^2 \end{bmatrix} [\Delta x \Delta y]^{\mathrm{T}} \\ &= [\Delta x \Delta y] \, E(x,y) \, [\Delta x \Delta y]^{\mathrm{T}} \end{aligned} \tag{HO2}$$

Harris avoids eigenvalue calculation with the following approximation: [Eq 3]

$$h(x,y) = \det(E) - k.(\mathrm{tr}(E))^2 \tag{HO3}$$

To apply this method to a 3D polygon-mesh introduces additional complications of potentially irregularly spaced vertices and topology, causing difficulty calculating derivatives. Sipiran and Bustos use PCA to fit a plane over the region under analysis, the vertices within the region are rotated to local tangent plane coordinates $f(u,v)$ over this plane and fitted with a quadratic surface (Sipiran & Bustos, 2011). A second order parametric equation of the form in Equation.4 is considered sufficient to capture the most complex shapes encountered, yet allows relatively simple differentiation. The quadratic patch parameters are represented by $\mathbf{a}^{\mathrm{T}}$, such that:

$$f(u,v) = \mathbf{a}^{\mathrm{T}} \left(u^2, uv, v^2, u, v, 1 \right) \tag{HO4}$$

Yielding a 3D Harris expression of the form:

$$\mathsf{E} = \frac{1}{\sqrt{2\pi\sigma}} \int_{\mathbb{R}^2} \mathrm{e}^{-\frac{u^2+v^2}{2\sigma^2}} \begin{pmatrix} f_u^2(u,v) & f_u(u,v)f_v(u,v) \\ f_u(u,v)f_v(u,v) & f_v^2(u,v) \end{pmatrix} \mathrm{du}\ \mathrm{dv} \tag{HO5}$$

Note that the discrete Gaussian function W representing the region is replaced with a continuous Gaussian function. The 3D Harris Operator is calculated as before via (HO4) and those of the highest value are selected or those over a threshold.

$$\mathsf{H}(\mathsf{x}) = \det(\mathsf{E}) - 0.04\mathrm{tr}^2(\mathsf{E}) \tag{HO6}$$

B.12 Heat diffusion features

The *Heat Equation* defines heat diffusion across a surface from a point. As the rate of diffusion is tied to the curvature of the surface, but not dependent on the Euclidean orientation of the point this makes it a basis for an intrinsic feature detector. This heat diffusion equation over a compact Riemannian manifold is defined by:

$$\left(\Delta_X - \frac{\partial}{\partial t}\right) u(x,t) = 0 \tag{HKS1}$$

where $u(x,t)$ is heat distribution with respect to a point, x, at a time, t, and Δ_X is the Laplace-Beltrami operator. The Laplace-Betrami operator, can be considered here as the divergence of heat from the point source, but in more general terms is the divergence of a gradient on functions defined on surfaces in Euclidean space. The Laplace-Beltrami operator is defined as:

$$\int f\Delta_X h \, d\text{vol} = -\int g_x(\nabla_{Xf}, \nabla_X h) \, d\text{vol} \tag{HKS2}$$

for smooth scalar fields $f, h : \mathbb{X} \to \mathbb{R}$ where $d$vol represents the differential area or volume of the manifold. For a Euclidean metric, where $g_{ij} = \mathbf{I}$, the Laplace-Beltrami operator reduces to:

$$\Delta_f = -\left(\frac{\partial^2 f}{\partial u_1^2} + \frac{\partial^2 f}{\partial u_2^2}\right) \tag{HKS3}$$

The heat kernel is a solution to an initial point condition $f(0,x) - \delta(x - x')$ representing the heat transferred from x to x' in time t via diffusion. If ϕ_i are the eigenfunctions and $\lambda_i$the eigenvalues of the Laplace-Beltrami operator such that $\Delta\phi_i = \lambda_i\phi_i$ the heat kernel can be represented as:

$$k_t(x, x') = \sum_{i \geq 0} e^{-\lambda_i t}\phi_i(x)\phi_i(x') \tag{HKS4}$$

The most attractive feature of this heat kernel is that it is intrinsic, or invariant to isometric deformation. This allows recognition of articulated or crumpled geometric models. Because the analytical geometry of shapes presents difficulty in calculation, heat kernels are calculated from the discrete form of the Laplace-Beltrami operator characterised as:

$$(\Delta_X\, f)_i = \frac{1}{a_i}\sum_j w_{ij}(f_i - f_j) \tag{HKS5}$$

where a_i are the normalisation coefficients and w_{ij} are the weights. Reformulating to matrix notation gives~~:~~

$$\Delta_{\hat{X}} f = A^{-1} L f \tag{HKS6}$$

where $A = \mathrm{diag}(a_i)$ and $L = \mathrm{diag}(\sum_{l \neq i} w_{il}) - (w_{ij})$

The eigenvalues and eigenvectors are calculated over surface mesh of N vertices using the finite element method or the *generalised eigendecomposition* of discrete Laplace operators over the model mesh surface. If the eigenvalues $\lambda_0, \cdots, \lambda_k$ are ordered by size within a $(k+1) \times (k+1)$ diagonal matrix Λ and the corresponding eigenfunctions $\phi_0, \cdots, \phi_k$ within a $N \times (k+1)$ matrix Φ this gives a formulation:

$$A\Phi = \Lambda L \Phi \tag{HKS7}$$

Sun *et al* introduce a Heat Kernel Signature (HKS) based on the Heat equation for the purposes of determining distinctive features on mesh shape models (Sun *et al*, 2009). As computing the diffusion between the point of interest and all other mesh points is prohibitively expensive, only the time parameter is calculated and calculation over the spatial domain is dropped.

$$\mathsf{HKS}_x(t) = k_t(x, x) = \sum_i e^{-\lambda_i t} \phi_i^2(x) \tag{HKS8}$$

The analytical solution to the underlying manifold is generally unknown, so approaches are based around discrete mesh-based methods. A discrete Laplace operator may be constructed from a sparse matrix A, representing the area associated with each mesh vertex as $A(i, i)$ on the matrix diagonal and W, represents a symmetric semi-definite matrix:

$$L = A^{-1} W \tag{HKS9}$$

This arrangement guarantees that the general eigenproblem description $W\phi = \lambda A\phi$ is composed of real eigenvalues and eigenvectors. The Laplace mesh can then be defined as

$$L = \Phi\Lambda\Phi^{\mathsf{T}} A \tag{HKS10}$$

such that Φ is a matrix of eigenvector columns and Λ is a diagonal matrix of eigenvalues. The local maxima of the HKS function at large time values is used to determine salient feature points. The authors introduce a multiscale variation of the HKS by altering the time parameter in a logarithmic fashion (see Eqn. HKS8), reflecting the exponential decay of the heat diffusion equation.

The Heat Kernel Signature is invariant with respect to Euclidean coordinates and possesses a robust invariance to moderate geometric distortion, yet the Heat Equation is not independent of scale. Some approaches normalise the entire shape model to similar dimensions (global pre-normalisation). The Scale Invariant Heat Kernel Signature or SI-HKS is an approach that normalises the local scale, using a logarithmic factor to correct the exponential form of the heat kernel.

B.13 Graph methods

Topological persistence is a concept introduced by Edelsbrunner *et al*, different topological shape features appear at different values of spatial resolution, otherwise known as persistent homology (Edelsbrunner *et al*, 2000). The appearance, and disappearance of topological features such as voids, connections, tubes for varying values of the scale parameter constitute a numerical signature independent of orientation, (though not of scale). A continuous non-negative scalar function, such a heat kernel, is defined on the surface of a shape, this single parameter *filtering function* reveals the disappearances of geometrical local maxima and corresponding changes in the level sets defining the shape topology. Ferri *et al* report that a vector-valued filtering function is stable with regard to function perturbations and geometrical space perturbations (Ferri *et al*, 2011). The interval between appearances and annihilation of topological features are characterised as *persistent Betti numbers* or represented on a *persistence diagram.*

To briefly introduce a number of connected mathematical concepts, a *level set* is the set where a real-valued function holds the same constant value. For a function of two variables, this is a *level line* or *contour line*, for a three variable function, a *level surface*, or *isosurface*, for a higher number of variables, a *level hypersurface.* A *quotient topology* is the representation of level sets defined as equivalent classes within a topological space. A *Reeb graph* is a mapping of level sets of a function within a quotient topology.

For a continuous, real-valued functions that describe a surface and are free of degenerate critical points, the corresponding Reeb graph is more readily defined: vertices of the Reeb graph correspond to critical points, arcs to connected components of level sets and level sets are contracted to points. As the topological connectedness of a shape is independent of orientation, it is a candidate for a numerical signature identifying an object. A function based on object height in one dimension is adequate for a Reeb graph on a two-dimensional manifold, but is sensitive to shape orientation.

Hilaga *et al* describe a function to allow determination of mesh-based shape topology independent of shape orientation (Hilaga *et al*, 2001). Each vertex point is defined relative to the summed distance from all other vertex points. These individual displacements are calculated from a geodesic distance edge length metric based on Dijkstra's algorithm. These values are normalised for scale as the geodesic distance measure is not scale invariant. Mesh polygons may be subdivided for further accuracy. This Reeb graph is then used to generate a multi-resolution shape signature, the graph is divided by the finest resolution and adjacent nodes within the same graph division are subsumed into one another. This process is repeated for coarser graph divisions until a set of graphs are generated at multiple scales.

B.14 Greedy matching method

Skeletal graphs produce a visually similar graph as Reeb graphs. Rather than determining critical points at connecting level sets, a skeletal graph will thin the volume of a shape along an insignificant axis to produce a stick-figure representation of a shape volume. Sundar *et al* describe one implementation where a shape mode is transformed to a voxellised representation (Sundar *et al*, 2003). For each voxel, its minimum distance to a nearest boundary surface is compared against the mean minimum boundary distance of that of its 26 neighbours. If this value is over a specific threshold, and the voxel is not too close to the boundary surface, the voxel can be used as a skeletal point. A *Minimum Spanning Tree* algorithm is subsequently used to connect these points to form an undirected acyclic graph, which may be further refined.

B.15 Hybrid mesh methods

Researchers have found that combining several distinct polygon mesh matching methods leads to higher overall accuracy. Daras and Axenopoulos describe a Compact Multi-View Descriptor, a combination of three 2D view matching techniques, namely Polar-Fourier Coefficients, Zernike Moments and Krawtchouk Moments (Daras & Axenopoulos, 2010). The 2014 Eurographics Workshop on 3D Object Retrieval event tests refined shape matching techniques against a prepared dataset. Four of the five methods entered combine several methods to boost overall accuracy (Li *et al*, 2014).

B.16 CAD graph methods

CAD programs retain geometry description within internal proprietary program formats, or exported to external neutral formats such as ISO 10 303 (See Chapter 3 for further details). Efforts to compare geometry models defined within CAD programs have extracted descriptive data from available program API, or from the geometry exported to a neutral format representation (Miao *et al*, 2002). CAD model matching methods are limited by the available access to shape data. API access to CAD program internal geometry model representation is limited to the proprietary CAD programs that support equivalent API access. External comparisons can be made from CAD model representations that are exported to a documented neutral file format such as IGES, STEP or DXF. Other research will take two-dimensional representations of 3D CAD models such as a screen grab and use SIFT image matching techniques to derive a match (Zhu et al, 2010).

Some use heuristics to determine the feature representation from descriptive semantics within neutral formats (Tan *et al*, 2013). The approach that has attracted most research attention extracts the model as a collection of distinct boundary surfaces including a description of interconnection with adjoining surfaces. Each surface is represented as a node within an *Attributed Adjacent Graph*, while the edge between surfaces is represented as a connecting arc, (Joshi & Chang, 1988). The distinction between an internal angle and an external angle is given by a binary value attached to the arc. AAG graphs generated via different model geometries can be compared to determine similarity using techniques of *subgraph isomorphism*, namely identifying similarities between graph subsets.

The basic AAG graph is limited to polyhedral shapes rather than curved faces and was adapted to work in cases where a single surface contacts more than one other edge, such as the cap of a cylinder adjoining cylinder walls defined by several surfaces.

B.17 Multi-attributed adjacency graph

The *multi-attributed adjacency graph* or MAAG addresses several of the shortcomings of the AAG, describing the angle between mating surfaces in greater resolution. Venuvinod and Wong present a more detailed representation of surface characteristics within an *enhanced winged-edge data structure*, capturing extra geometric data available in the common AutoCAD Drawing eXchange Format or the predecessor to the STEP format, IGES (Venuvinod & Wong, 1995). Much of the impetus behind developing geometric shape comparison within CAD derives from the utility of transferring an engineering design to a

Computer Aided Machining tool without human intervention. As geometric forms within modern CAD programs are defined as a set of "features", CAD graph techniques attempt to extract geometric regions that correspond to CAD features, such as recognising a boss, or a slot (Henderson & Anderson, 1984).

An extra complication is the different set of features inherent to CAM manufacturing operations; features may be defined by machining operations rather than geometries relevant to engineering form concepts. For example, a manufacturing feature might be determined to be a volume removed by a tool in a single cutting operation, yet a form feature could describe a volume swept by a referenced face. Elinson *et al* describe *classification trees*, hierarchical graphs of manufacturing features for the purpose of identifying model similarity (Elinson *et al*, 1997).

Attempts to create rule-based systems or algorithms that can categorise CAD or CAM features from model geometries are limited by the lack of a canonical set of feature types and by the difficulty of identifying intersecting features in a geometric definition (Han *et al*, 2000), (Marefat & Kashyap, 1990). One common heuristic adopted extends identified features to determine the largest coherent feature volume (Regli et al, 1995).

To briefly observe in passing, most form features depend on prior features. For example, a hole feature requires a pre-existing solid feature. This dependency is covered in more detail in Chapter 5. Parametric feature CAD programs commonly use hierarchical tree structures to graphically represent the interdependency of form features. These relationships are described in a *Model Dependency Graph*, capturing the interdependence of identified machining features (Cicirello & Regli, 1999). There may be several different possible configurations of feature hierarchy, consequently the Model Dependency Graphs of identical geometrical models may not have unique feature orders, this property is defined as *D-morphism*. Part similarity is determined via the largest common subgraph using *Ullmann's algorithm* or a *Greedy Subgraph Isomorphism Checker*, fully solving D-morphism uniqueness requires additional model geometric or topological data (Cicirello & Regli, 2002).

El-Mehalawi and Miller, describe a graph matching method that extracts model face connectivity from an exported neutral STEP AP203 geometry file. This method extracts topological and vertex geometry information, but does not describe a matching process that uses this geometric information (El-Mehalawi & Miller, 2003a), (El-Mehalawi & Miller, 2003b). Ma *et al* group topological graph features by the extra geometrical information of nodes (Ma *et al*, 2009). Wang *et al* optimise the subgraph isomorphism calculation for similarity matching by pruning inconsequential surfaces and weighting more salient surface nodes (Wang *et al*, 2010). Li *et al* introduce a feature hierarchy, each identified feature is assigned a parent-child relationship within a *Hierarchical Partition Graph* that allows the use of a more efficient *Greedy Matching* algorithm for model comparison (Li *et al*, 2015).

Bin *et al* adopt an attributed graph with nodes based on primitive geometric surfaces (presumably extracted from the STEP neutral file format data strings) and edges defined by connecting model faces (Bin *et al*, 2017). The graph edit distance, or the number of graph edits required to transform one model graph to another between two graphs is defined as the model similarity.

Zehtaban *et al* introduce a method converting a CAD graph model into an Opitz alphanumerical code for ready comparison of Computer Aided Machining objects with other similar shapes, this method is notable in that the shape signatures may be numerically compared using cosine similarity or similar metrics (Zehtaban *et al*, 2016).

B.18 Volumetric CAD methods

Convex hull volume decomposition methods and cell based volumetric decomposition approaches are other methods for CAD feature identification, but are primarily developed for the recognition of manufacturing operations rather than creating a representation suited to similarity matching.

Convex hull volumetric decomposition approach starts with the convex hull of the model shape, then removes select prismatic or cylindrical volumes to arrive at the final shape. While the concept is simple, the implementation is difficult with a mixture of curved and flat surfaces. Kyprianou described the original concept, Wang and Kim developed practical techniques using a sequence of boolean addition and subtraction operations to represent the model geometry, described as the *Alternating Sum of Volumes with Partitioning* method (Kyprianou, 1980), (Wang & Kim, 1982). The differences between a geometric model and its convex hull can be recursively decomposed until they become convex. This process generates a structured hierarchy of model boundary faces that allows an association with volumetric representations. Cylindrical or curved surfaces are replaced by equivalent planar volumes, to be re-inserted into the model after decomposition. This approach is hampered by a limited capacity to tackle free-form surfaces, or curved surfaces that do not follow the principle axes of the model.

Cell-based volumetric decomposition has a similar approach to shape decomposition from a CAM machining perspective. The negative voids in the convex hull of a geometry model are voxelised and then regrouped to correspond with the most efficient toolpath volume. As a brute force comparison of multiple cellular voids is computationally expensive, several heuristics are introduced to partition the search space (Sakurai, 1995). While these methods are not directly used for geometric comparison, volumetric decomposition has been used to determine the presence of CAD features within models (Pilli, 2017; Ramesh *et al*, 2001).

B.19 Hint based matching

Vandenbrande and Requicha describe a hint-based method to guess the feature composition of a model (Vandenbrande & Requicha, 1993). Where previous feature recognition methods used a minimum set of geometrical and topological properties to identify machinable features, the hint-based approach relaxes these constraints to allow partial representations of feature boundaries or topologies, allowing intersecting features to be recognised. One common heuristic is to extend the characteristic boundary surface of a machinable feature to test if it coincides with other faces of the model geometry. The range of hints has also been extended to non-geometric manufacturing attributes such as design features, design attributes and tolerances.

As testing all surface geometry to identify potential feature membership is computationally expensive, the process is optimised by ordering matching candidates within a *priority queue* ranked by measure of matching probability, a form of greedy matching. More in-depth reviews of these approaches are given in Shah, Babic, Han (Shah *et al*, 2001; Babic *et al*, 2008; Han *et al*, 2000).

B.20 Interacting Multiple Methods

Gao and Shah adapt the hint-based approach to encompass several types of graph matching in an *Extended Attributed Adjacency Graph* method (Gao & Shah, 1998). The requirement of hint-based methods to amalgamate information from multiple topological relationships is formalised as *Concave Adjacency Graphs*, *Manufacturing Face Adjacency Graphs*, *Minimal Condition Subgraphs*, *Partly Concave Adjacency Graphs* and *Concave Adjacency Graphs*. Inference rules determine feature type from configurations of the extended graph information. Gao and Shah describe their approach as a hybrid method, combining hint-based matching with graph matching.

B.21 Hybrid methods

Chu and Hsu introduce a hybrid method that combines elements of three disparate CAD model representations (Chu & Hsu, 2006). Form Feature Adjacency Graph or FAG, is composed of feature volumes combined via additive or subtractive operations. Each node of this FAG contains a topological graph of its corresponding feature volume, similar to an Attributed Adjacency Graph. The similarity of individual FAG nodes are determined. A similar method compares topographical graph data. The resulting similarity matrices can yield an optimal solution as an assignment problem solved via the Hungarian algorithm.

The FAG and topological representation of a model does not contain distinguishing geometrical information and may yield matches for differing model geometries. A third model analysis creates a D2 statistical histogram from sampled points on the model surface, allowing a further level of model discrimination.

Li *et al* use a similar combination of a form feature graph combined with a D2 statistical measure (Li *et al*, 2010). The interdependencies between features are represented with a *Feature Dependency Directed Acyclic Graph*, relying on a CAD API for feature hierarchy information. This graph representation allows sub-components to be extracted from the model, given as independent graph branches from a root node. It also permits "de-featured" simplified representations stripped of minor surface features, the outermost nodes of this graph.

Huang *et al* extend this method to match model sub-parts in more detail, incorporating model characteristics such as axial and radial geometric dimensioning, part tolerances and a measure of the relative angles between characteristic machining feature data (referenced as "tool access direction") (Huang *et al*, 2015).

These hybrid methods illustrate a recent trend towards combination of several matching methods to achieve higher matching scores. Techniques are drawn from the formerly divergent domains of 3D polygonal mesh matching and feature matching performed on data extracted from CAD programs or output files.

This appendix has briefly outlined the approaches to surface geometry model matching within the broad areas of shape matching, shape registration and feature recognition using the four distinct approaches of view-based matching, histogram-based matching, transform-based matching and graph-based matching.

C Single model geometric matching test

+ve target name	+ve target gen time	+ve target insertion pc	+ve target scale	+ve target rotation	+ve target RMSE match	-ve target name	-ve target gen time	-ve target scale	-ve target rotation	-ve target RMSE match
unit_torus_z2.0.stp	39.7137581103	Point(x=0.180087140148161	1.7800715208	258.9365217997	inf	unit_cyl_yz4.0_blend.01.stp	19.3392398625	1.7846706225	19.2503027947	inf
unit_torus_z2.0.stp	333.0471164047	Point(x=0.180087140148161	1.7800715208	258.9365217997	0.0009467013	unit_cyl_yz4.0_blend.01.stp	19.3392398625	1.7846706225	19.2503027947	inf
unit_sphere_xy1.5.stp	40.4736870914	Point(x=0.235314571307478	2.9480444526	130.0669988063	0.0007854441	unit_cube_x1.5_blend.01.stp	21.3815919112	6.348338601	229.2861091327	inf
unit_sphere_z2.0.stp	30.7176357002	Point(x=4, y=7, z=7)	3	246	0.0004366373	unit_cube_xz4.0_blend.03.stp	41.1745807359	1	184	inf
unit_cube_yz1.5_blend.05.stp	31.3971981907	Point(x=0, y=8, z=2)	2	61	0.0007901559	unit_sphere_yz1.5.stp	32.5692439562	10	172	inf
unit_cube_x1.5.stp	31.2476360803	Point(x=2, y=3, z=5)	2	228	0.0002502628	unit_cube_x1.5_blend.01.stp	21.1061851908	3	294	0.0095419462
unit_cube_xz2.0_blend.06.stp	27.2522544473	Point(x=7, y=9, z=9)	15	88	0.0014994855	unit_cube_yz4.0_blend.05.stp	87.4976326593	7	31	7.8400170077
unit_cube_inc1.5_blend.01.stp	47.2122069333	Point(x=7, y=2, z=1)	8	180	0.0008541827	unit_cyl_y4.0_blend.01.stp	16.2191002737	7	218	inf
unit_cube_yz8.0_blend.05.stp	113.6132731489	Point(x=0, y=0, z=6)	4	341	0.0008526973	unit_cyl_xy1.5_blend.06.stp	64.3592101262	2	50	inf
unit_cyl_yz1.5_blend.03.stp	25.1167743652	Point(x=1, y=3, z=2)	14	10	0.0068035635	unit_cyl_xy4.0_blend.06.stp	186.9959848715	12	120	inf
unit_cube_yz4.0_blend.03.stp	115.7963558613	Point(x=3, y=2, z=7)	12	270	0.0006179216	unit_cube_yz8.0_blend.05.stp	72.140584613	11	187	24.4893052907
unit_torus_xy2.0.stp	16.6745807359	Point(x=1, y=5, z=8)	4	286	0.0044845181	unit_cyl_xz4.0.stp	94.755928995	12	52	63.5526788635
unit_cyl_xz8.0.stp	19.1086118291	Point(x=4, y=5, z=6)	5	266	0.6122820025	unit_cube_yz2.0_blend.05.stp	67.1895590695	8	86	inf
unit_cyl_xy1.5_blend.03.stp	57.5333655162	Point(x=5, y=2, z=0)	1	304	0.0003368171	unit_cube_inc1.5_blend.03.stp	20.2865535198	1	97	inf
unit_cyl_xz4.0.stp	79.6016390452	Point(x=3, y=8, z=7)	14	328	0.0146435488	unit_torus_y1.5.stp	26.2492462369	18	302	inf
unit_cube_inc1.5_blend.01.stp	106.8164318838	Point(x=0, y=5, z=5)	12	318	0.0009347494	unit_cyl_xy4.0_blend.01.stp	79.779361411	3	167	inf
unit_cube_inc8.0.stp	11.5492344534	Point(x=6, y=0, z=3)	9	165	0.0103911574	unit_sphere_yz8.0.stp	117.116246769	7	195	inf
unit_cube_xy1.5.stp	12.4770803558	Point(x=4, y=1, z=8)	15	254	0.0019652096	unit_cyl_inc8.0_blend.05.stp	153.5342401551	1	125	inf
unit_cube_yz4.0_blend.06.stp	151.482329329482	Point(x=3, y=2, z=7)	19	140	0.0492605528539	unit_cube_inc1.5_blend.06.stp	28.5175824844154	12	177	6.0268339089
unit_cube_inc8.0.stp	11.5492344533982	Point(x=6, y=0, z=3)	9	165	0.0170182818151	unit_sphere_yz8.0.stp	117.116246769044	7	195	inf
unit_cyl_inc4.0_blend.01.stp	204.973529967427	Point(x=7, y=9, z=2)	10	351	0.0358331253775	unit_cyl_xy4.0_blend.03.stp	71.6208034978122	4	90	inf
unit_cube_xz1.5_blend.03.stp	23.5626522186528	Point(x=5, y=8, z=2)	19	273	0.0104541719378	tetrahedron.scaled.stp	8.11443193433772	6	79	inf
unit_cyl_xz1.5_blend.01.stp	17.5869881965566	Point(x=7, y=1, z=9)	8	223	0.0148690155142	unit_cube_xy4.0_blend.01.stp	52.238067597151	15	165	inf
unit_cube_xz2.0_blend.05.stp	24.6191807191544	Point(x=0, y=9, z=7)	4	349	0.00278070674117	unit_cube_yz8.0.stp	47.8948801314339	6	358	22.6425263458
unit_torus_xy4.0.stp	37.0538531394818	Point(x=9.0, y=8.0, z=0.0)	4	320	0.00149816779644	unit_cube_yz4.0.stp	17.7436180779596	4	188	inf
unit_cube_inc4.0.stp	9.36607476489857	Point(x=2, y=8, z=9)	1	211	1.27382490328	unit_cube_xz8.0_blend.03.stp	68.0409469363923	14	332	41.5475251391
unit_torus_xz1.5.stp	65.3982786914223	Point(x=7.0, y=9.0, z=5.0)	5	239	0.320736885394	unit_cyl_z1.5_blend.03.stp	132.644710307971	16	194	inf
unit_cube_xy1.5_blend.06.stp	57.4374580192977	Point(x=5, y=5, z=7)	12	18	0.00661077156934	unit_torus_x2.0.stp	15.3549772244593	1	60	inf
unit_torus_inc4.0.stp	237.418900030524	Point(x=3.0, y=5.0, z=1.0)	15	198	0.0100690846715	unit_torus_inc1.5.stp	153.978585707916	6	67	0.00200294625604
unit_cube_x2.0_blend.01.stp	21.0876182634634	Point(x=3, y=5, z=9)	2	85	0.729623677492	unit_cyl_xy8.0_blend.05.stp	191.559778429322	7	253	inf
unit_cube_inc1.5_blend.01.stp	104.711082677323	Point(x=8, y=0, z=1)	14	30	0.00363373348527	unit_cyl_z1.5_blend.03.stp	1280.11889923225	2	280	inf
unit_cyl_inc2.0_blend.03.stp	183.020692423574	Point(x=8, y=5, z=6)	10	55	0.00565485513796	unit_torus_y1.5.stp	38.5498151993805	17	165	inf
unit_cube_inc2.0.stp	14.1386348664391	Point(x=8, y=2, z=0)	3	99	0.00124003609203	unit_sphere_xy4.0.stp	79.6740788383809	7	92	inf
unit_cube_xz1.5_blend.01.stp	46.8101052468323	Point(x=4, y=0, z=9)	2	125	0.001367358144	unit_cyl_xy2.0_blend.03.stp	190.489483109455	15	67	inf
unit_cyl_x2.0.stp	58.9267837545753	Point(x=6, y=4, z=8)	12	289	0.0140252500004	unit_cube_x8.0_blend.01.stp	47.070758319259	7	205	inf
tetrahedron.scaled.stp	8.79842268499363	Point(x=6, y=5, z=9)	12	244	0.0207446776016	unit_cube_yz8.0_blend.01.stp	115.292015009087	15	35	inf
unit_cyl_inc1.5_blend.01.stp	88.7516874959374	Point(x=3, y=5, z=3)	8	125	0.00255444885233	unit_cube_inc2.0_blend.01.stp	96.6343410982665	11	265	inf
unit_cube_xz4.0.stp	35.6773719120545	Point(x=0, y=9, z=2)	23	185	inf	unit_cyl_inc4.0_blend.05.stp	102.014513776513	21	352	inf
unit_cube_inc1.5_blend.03.stp	21.4554271853628	Point(x=0, y=5, z=7)	11	54	0.00625075529923	unit_cyl_z1.5_blend.01.stp	195.205562987114	15	48	inf
unit_cube_inc2.0.stp	14.1386348664391	Point(x=8, y=2, z=0)	3	99	0.00113026838328	unit_sphere_xy4.0.stp	79.6740788383809	7	92	inf
unit_cube_xz1.5_blend.03.stp	27.7889946793083	Point(x=2, y=1, z=3)	18	45	0.00717375446365	unit_cyl_z2.0_blend.05.stp	1675.7303526512	21	318	inf
unit_cube_xz4.0.stp	32.6235674293232	Point(x=0, y=9, z=2)	23	185	0.0322571685624	unit_cyl_inc4.0_blend.05.stp	108.692227806345	21	352	inf
unit_cube_inc8.0_blend.05.stp	155.189135251702	Point(x=8.0, y=8.0, z=8.0)	19	188	0.0225497483775	unit_cyl_inc4.0_blend.03.stp	2552.95978063408	24	352	inf
unit_cube_inc8.0.stp	6.28264478329308	Point(x=2, y=8, z=7)	10	334	0.0172894129138	unit_cyl_y8.0.stp	211.837803482227	17	170	inf
unit_cyl_xy1.5_blend.06.stp	86.5866080656928	Point(x=9, y=6, z=6)	20	348	0.0153979514085	unit_cyl_x8.0.stp	86.3169192826322	13	179	inf
unit_cube_yz4.0.stp	18.5827885107727	Point(x=8, y=2, z=4)	13	197	0.0143826204041	unit_sphere_inc2.0.stp	15.424560939349	18	251	inf
unit_cube_x4.0_blend.03.stp	153.528529011397	Point(x=2, y=9, z=7)	17	237	0.00292022172177	unit_cube_x4.0_blend.01.stp	48.1081738399262	12	323	0.173837766789
unit_cyl_inc1.5.stp	657.466058685363	Point(x=0, y=7, z=6)	8	155	0.0114905035688	unit_cube_inc2.0.stp	5.81135359658917	5	343	inf
unit_cube_xz2.0_blend.06.stp	27.2522544473168	Point(x=7, y=9, z=9)	15	88	0.0079805286846	unit_cube_yz4.0_blend.05.stp	87.4976326592669	7	31	8.60023987198
unit_cube_yz4.0_blend.06.stp	111.623026455831	Point(x=0.0, y=0.0, z=0.0)	10	90	0.0050583648446	unit_cube_inc1.5_blend.06.stp	28.5175824844154	12	177	5.5185358576
unit_cube_yz4.0_blend.06.stp	151.482329329482	Point(x=3, y=2, z=7)	19	140	0.0492605528539	unit_cube_inc1.5_blend.06.stp	28.5175824844154	12	177	6.0268339089
unit_cyl_inc4.0_blend.03.stp	90.7728258130144	Point(x=8, y=6, z=4)	5	295	0.00247623503715	unit_cyl_xy2.0_blend.05.stp	172.327903544074	5	35	inf
unit_cyl_z4.0_blend.03.stp	196.465629517618	Point(x=4, y=9, z=3)	13	111	0.00546070767158	unit_cyl_xy4.0_blend.03.stp	100.28377453222	23	28	inf
unit_cube_inc2.0.stp	6.16422489606272	Point(x=3, y=7, z=4)	13	295	0.0063628949913	unit_cyl_xy2.0_blend.03.stp	95.2426504548456	22	8	inf
unit_cube_inc2.0_blend.01.stp	22.4446573177282	Point(x=6, y=0, z=9)	6	264	0.00244026519714	unit_cube_xy4.0_blend.01.stp	84.5927798703982	6	273	9.6985562581
unit_cyl_xy1.5.stp	45.8452369412694	Point(x=5, y=0, z=3)	11	18	0.0256148744242	unit_cube_inc8.0_blend.03.stp	62.2471732518637	13	188	inf
unit_cyl_inc1.5_blend.05.stp	148.823428453479	Point(x=5, y=6, z=3)	10	41	0.00112803362712	unit_cyl_y2.0_blend.01.stp	21.2237343827986	18	184	inf