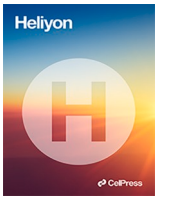
Heliyon 9 (2023) e19694

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Geometric feature extraction in manufacturing based on a knowledge graph

Tobias Köhler [a,b,*], Buchao Song [b], Jean Pierre Bergmann [b], Diana Peters [a]

[a] German Aerospace Center. (DLR), Institute of Data Science, Jena, Germany
[b] Technische Universität Ilmenau, Production Technology Group, Ilmenau, Germany

ARTICLE INFO

Keywords:
Knowledge graph
Feature technology
Ontology
Manufacturing
Geometry analysis

ABSTRACT

In times of global crises, the resilience of production chains is becoming increasingly important. If a supply chain is interrupted, a cost-effective solution must be established quickly. In the context of Industry 4.0, the concept of smart manufacturing offers a solution for fast and automated decision-making in production planning. The core idea of smart manufacturing is the digitalization of the product life cycle and the linking of individual phases of this cycle. Computer Aided Process Planning (CAPP) plays an important role as the connecting element between design and manufacturing. An important prerequisite for CAPP is the automated analysis of 3D models of components. The aim of this work is the development of an automatic feature recognition (AFR) -method to recognize geometric manufacturing features and their properties from 3D-models and then store them in a knowledge base. In that way, the result of the design can be automatically analysed and compared with manufacturing information afterwards in order to achieve an automated process planning. Geometric and topological information of a 3D model (STEP-AP242 format) generated by CAD systems is extracted by a Python-script developed and stored in an ontology-based knowledge base. The extracted product data is analysed using a Python-script to identify manufacturing features. To provide a comprehensive extensibility of the model, geometric features are defined according to a layered and hierarchical structure.

1. Introduction

In the context of Industry 4.0, smart manufacturing approaches offer a solution to reduce manufacturing costs and delivery times [1,2]. In addition to automated production, this concept also considers the digitisation and linking of all elements of supply chains. There are mainly three concepts for computer-aided approaches: (1) Computer Aided Design (CAD), (2) Computer Aided Process Planning (CAPP) and (3) Computer Aided Manufacturing (CAM). Process planning still requires human input to analyse the design of a product and to determine a suitable manufacturing process [3]. Since engineers often have incomplete data about manufacturing capacities the analysis of products cannot be done in real-time. Therefore, an automated solution for analysing and providing data is needed.

CAPP is an important link between Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM). A digital description of parts is required as inputs for CAPP systems. The Standard for the Exchange of Product model data (STEP) provides a good way to store and share information about a 3D models of parts. A STEP file (AP242) contains geometrical and topological descriptions of 3D

* Corresponding author. German Aerospace Center (DLR), Insitute of Data Science, Jena, Germany
E-mail address: tobias.koehler@dlr.de (T. Köhler).

https://doi.org/10.1016/j.heliyon.2023.e19694
Received 13 December 2022; Received in revised form 24 April 2023; Accepted 30 August 2023
Available online 4 September 2023

models, Model-Based Definitions (MBD) and Product Manufacturing Information (PMI), such as annotations of geometric dimensioning and tolerancing, fits or other dimensions. To perform an automated analysis of 3D parts, a semantic description can be derived from the information in the STEP file. The semantic description of geometric features and their properties of a 3D-CAD model can be used for (semi-)automated production planning. In Koehler et al. [4] the authors of this work presented a method for a Knowledge Graph-based description of manufacturing technology capabilities. The automated production planning can be achieved by matching the geometric properties of parts with capabilities of manufacturing machines. Due to the different types of information, a digital 3D model cannot directly be compared with manufacturing process data in order to decide about manufacturability. A 3D model is described by geometric and topological information. In order to compare this 3D model with process data, the part must be described semantically and in a machine-interpretable way. The semantic description of geometric elements of parts is called feature [5]. The derivation of these features from related geometry elements is called automatic feature recognition (AFR) [3,6,7].

In the state-of-the-art, a lot of AFR models are described and presented [3,6]. However, there are still some challenges in that field, that have not been fully solved. One of the most critical challenges is the extensibility of an AFR-model [3,8]. It is nearly impossible to define all possible features generally applicable at once. Thus, there must be the option to continuously add new features to the model. In their work, Wang and Yu [8] presented an AFR model based on an ontology to achieve effective extensibility. Although this model shows great potential, there is still space for improvement. For example, this model cannot optimally handle or recognize interactive features [8]. The combination/interaction of two features can often disrupt the original definitions of the features and cause false recognition. Consequently, the ability to handle interactive features is also another main requirement for AFR systems [3].

This is where this work follows up with the development of an ontology-based AFR-model. With the help of our approach, geometric features of digital 3D models can be determined and stored in a machine-interpretable, easily extendable knowledge base. The goal of this work is the development of a method, which enables the extraction of geometric component features and to insert them afterwards into a knowledge base (Knowledge Graph (KG)). The required knowledge base is to be developed in order to store information of components and recognized features, thus preparing them for further use. The method and the enriched KG should be easily extendable and modifiable.

The base of this work is a layered and hierarchical structure of the feature library. This enables the separation of the geometric description of components and the linking of these with manufacturing processes. Furthermore, new features can be added easily in both independent hierarchical layers. In the first step, the faces of a part are assigned to predefined Basic Features (BF) according to their topological and geometrical properties. By combining the BF among each other and adding additional geometric restrictions, manufacturing process-related Manufacturing Features (MF) can be defined. With the help of this layered structure, both the BFs and the MFs can be easily extended. Furthermore, the layered description of the geometry of the part also allows it to handle interactive features.

2. Related work

By using MFs, the digital model of a part can already be prepared for subsequent processes during design. According to Weber [5] and Haasis [9], a feature is an additional semantic descriptive element of a product that describes non-geometric and geometric properties. Examples are holes, threads, slots and properties of surfaces. Semantic feature descriptions can be divided into function-oriented, geometry-oriented, and technology-oriented semantics [10,11]. This division leads to the distinction of different feature types made in the literature. A distinction is made between shape features, design features, manufacturing-related features, and compound features [12,13]. Shape features embody the geometric properties of the components as well as specific areas of the components [13]. Design features also contain the geometric properties of the components and additionally contain information about the intended task of a component, or a subarea of the component [14]. Manufacturing-related features represent, in addition to the geometric properties, semantic descriptions which can be used for the manufacturing of the product [15].

In order to be able to use the features defined in the design for production or for evaluation of components, they have to be extracted from 3D models. In general, systems for feature extraction and recognition from different CAD files combine information which is collected at a relatively low level (points, lines and curves) and convert them into features (holes, chamfers, slots, cylinders) [7,16,17]. Feature recognition methods can be divided into five areas [3,7,18,19]: (1) syntactic pattern recognition [20–26], (2) graph-based recognition [6,19,27–35], (3) logic rule-based recognition [36–43], (4) hint-based recognition [32,38,44–46] and (5) artificial neural nets [18,32,47–50].

The syntactic pattern recognition (SPR) method describes the recognition of features based on matching with previously defined geometric patterns [51]. Depending on the representation type, the pattern can be a string or a graph. At the beginning of the recognition process, patterns representing the features are predefined. During the recognition process, the geometry of a part is then compared to predefined patterns. When a geometrically matching pattern is found on the part, this geometry is recognized as a feature [3,6,23]. While this method can successfully detect some features, it has also a weakness: only 2D profiles of a 3D model is considered in this method. It lacks a holistic consideration of the model, which is why information about 3D profiles of the model can be lost. For this reason, the SPR method is mostly suitable for symmetrical components [3,6,20].

In graph based recognition (GBR), faces, dependencies and features are represented graphically using nodes and edges and evaluated using graph theory methods [29,51]. Features to be recognized are defined according to connection types (concave or convex). At the beginning of the process, an Attributed Adjacency Graph (AAG) is generated from a part. This AAG contains information about connections and connection types of all faces of a component. By comparing the AAG with predefined definitions of features, it is possible to recognize the features of a component [3,6]. An advantage of this method is its comprehensive and expedient extensibility. The features' simple and modular definitions facilitate the extension and modification. However, there is a problem with the sensitivity

of recognition in a purely GBR-based model. In most cases, only connections between faces are considered when defining features, but other geometric information is lost in this representation. This problem can be solved by using a combination with other AFR methods [52]. This problem is particularly noticeable when representing features that interact with each other [3,6].

Logic rule-based algorithms (LRBR) operate on decision logic rules. By those, certain shapes are recognized and sorted accordingly [8,51,53]. Actually, logic rules are used in all feature recognition methods. The characteristic of the LRBR method is that in the other methods, the logic rules are only applied in feature recognition process, while in this method logic rules are also used for the interpretation of a part. Here, geometric and topological information is directly analysed with logic rules to extract high-level information [3,6–8].

With the help of logical rules, the model has high stability and a high potential to detect complex features. However, due to the embedding of logical rules in the representation of a component, it is difficult to consider the representation phase and recognition phase separately. Therefore, this method is characterized by relatively low flexibility and extensibility [3,7].

Finally, features can also be recognized and classified using machine learning methods based on artificial neural networks (ANN). An ANN is a network of nodes and links between those nodes. Each node is characterized by input, an arithmetic operation and weighting. Input is either the output data or one or more outputs from other nodes. An ANN can learn from using sample data and apply the learned rules to previously unseen data. The inputs and the weights of nodes are constantly changed during the learning process. Thus, the database for training the networks is essential. If the learned rules are to be extended to new geometric features, the network must be trained again [3,18]. Since the extensibility of the feature library is one of the main challenges to be solved in this work, ANNs will not be considered further on.

As presented, there are two main challenges for AFR models: expandability of feature libraries and handling of interactive features. To achieve good extendibility and proper handling of interactive features, this work focuses on solving the following challenges: (1) expandability of feature library, (2) integration of modules, (3) handling of complex features, and (4) robustness of recognition. The different AFR methods presented are evaluated and compared below based on the above challenges.

2.1. Expandability of feature library

One of the main requirements of an AFR method is a high detection rate. Ideally, AFR models should be able to recognize all features of a component. It is relatively easy to recognize primitive features that are separated from each other, such as holes, linear grooves, or simple pockets. In reality, these features often do not occur alone or independently of each other. Primitive features are often interconnected and are then referred to as interactive features. This interaction of features can disrupt the original definitions of features and lead to false detections. The ability to handle interactive features can therefore be used as a benchmark for AFR models [3]. In the past, attempts have been made to define each interactive feature separately to solve the problem of recognizing interactive features. But the possible combinations of features are almost innumerable. Due to the lack of standards, the combinations of features are mostly dependent on the intended functions of the design and the habit of engineers. Therefore, it is very difficult to enumerate all possible combinations of features. A practical solution to this is to develop an AFR model with an easily extensible feature library so that the model can always be updated and extended. For this reason, extensibility is also noted as an important benchmark in recent research [3,8].

Ontologies offer a good solution for the quick extensibility of feature libraries. With a hierarchical structure and the opportunity to connect individual features among themselves over so-called triple relations, it is possible to use characteristics of existing features for the creation of new features. Thus, an easy extensibility of the library is given. Selected studies on ontology-based feature extraction are presented in more detail below:

In their work, Wang and Yu 2014 present an ontology-based automatic feature recognition framework. The authors aim at a standardized description of features using an ontology. The use of the presented ontology offers advantages due to a high level of flexibility, maintainability and traceability. The authors show that it is possible to represent features by means of an ontology on the one hand and to recognize features on the other hand. In the model of Wang and Yu connection types (e.g. concave/convex connections) of the faces are extracted first. Using logical rules, the faces that are connected in a concave way to each other are recognized as a face set (concave connected face set). This process is called *decomposition* in their paper. When the *decomposition* is done, face sets are compared with predefined rules to detect features. It is assumed in this model that each face set represents exactly one feature [8].

Ma et al., 2020 are using a graph-based feature recognition method in combination with an ontology. In their paper, the authors represent recognized features of a turning process in an ontology in order to use them as input for an automated production planning approach. Since the ontology is not used for feature detection in this case, there are no suitable means of extensibility for the method [51].

Due to the similarities between the structure of STEP schema and the structure of an ontology, it is possible to map the STEP schema

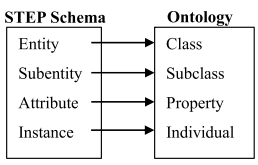


Fig. 1. Basic principle of mapping STEP schema into ontologies.

into ontologies. In previous research work, approaches have already been developed to represent the STEP schema in ontologies. In 2009, a model *OntoSTEP* was presented in the article of Ketan and Yaqoub [20], which allows to represent information of a STEP file in an ontology. In 2015, *OntoBRep* was presented in the paper by Perzylo et al. [54]. This model allows not only mapping STEP information into an ontology, but also further analysis of this information. In 2014, a similar model was presented in the paper of Wang and Yu [8]. This model focuses on feature recognition using information from STEP schema, which was previously mapped into an ontology.

The mapping concepts in the above contributions are similar. The basic principle for this can be illustrated as in Fig. 1.

Objects of the STEP schema are represented as entities, subentities, attributes and instances. Similarly, objects in an ontology can be represented as classes, subclasses, properties and individuals. Furthermore, the classes of an ontology and entities of the STEP schema can be hierarchically classified as super/subentities and super/subclasses. Therefore, it is possible to represent entities and instances from the STEP schema as classes and individuals in an ontology without changing their hierarchy. Objects of the STEP schema can be linked by attributes to other objects or other data such as strings, Booleans or numbers. In an ontology, Data and Object Properties exist, that can connect classes and individuals to other classes and individuals or to other data [8,20,54].

2.2. Integration of modules

When different modules of a software program are highly integrated with each other, it is difficult to modify and manage them. In this paper, a method to separate the representation of 3D models and the detection of features is presented, so that in future work one can extend the feature library without modifying other parts of the model. In LRBR (Logic Rule-Based Recognition), there is usually no separate representation module. Geometric and topological information is directly compared with logical rules to recognize features [3,7]. Due to overly complex logical rules, it is time-consuming to extend the model. In HBR (Hint Based Recognition), a representation module exists, but this module is highly integrated with the recognition module. Shapes of hints directly influence the definitions of features. Moreover, this representation module is highly dependent on manufacturing processes [3,6,46]. The GBR (Graph-Based Recognition) method provides a structure for separating the representation module from the recognition module. In this method, a 3D model is first represented by AAGs (Attributed Adjacency Graph). A separate recognition module has the task of comparing these AAGs with predefined patterns of features [29].

2.3. Handling of complex features

Most models have no issue with recognizing simple features such as slots, pockets, or steps. In reality, these features often do not appear as single features, but combined together as interactive features. Some combinations of simple features form reasonable patterns for manufacturing processes, therefore these combined simple features should be recognized collectively as one complex interactive feature. Due to the loss of information, the GBR method can only handle interactive features to a limited extent [3]. Moreover, the AAG (Attributed Adjacency Graph) can be very complex when features are connected with one another [3]. In the LRBR method, it is possible to correctly identify interactive features. However, required logical rules are costly and complex. The LRBR method focuses on the geometries of faces and the relationships between faces. For example, the definition of *Cross-Slot* requires at least 9 faces to be linked (8 side faces and 1 base face), while the definition of a simple *Slot* only requires constraints on 3 faces (2 side faces and 1 base face) [3,7] (see Fig. 6). The HBR method can simplify the definition of interactive features because it does not require a holistic consideration of all faces, but only consideration of hints [3,6,46].

2.4. Robustness of recognition

The interaction of two nearby features can suppress the definition of individual features, so that these features cannot or only be recognized incorrectly. A complete definition that considers all relevant information can improve robustness of detection. Due to loss of information, it is difficult to create a complete definition using GBR methods [3]. At this point, the LRBR method and the HBR method have similar characteristics and therefore both are suitable for a complete definition [3].

Four state-of-the-art conventional AFR methods exist, but since the SPR (Syntactic Pattern Recognition) method is only suitable for symmetric 3D models, this method is not considered. The GBR method can solve the integration between modules well, but due to loss of information, it is difficult to handle interactive features with it. The HBR method enables high robustness and can handle interactive features well. However, the HBR method is highly related to manufacturing processes by nature, since this method is based on hints of manufacturing processes. Which makes it difficult to improve extensibility of the HBR method in order to use the method for different manufacturing processes. In this work the LRBR method was chosen, since this method has high robustness and allows treatment of interactive features. The two major drawbacks of this method are complexity of the logical rules and strong integration from the representation module and recognition module. Due to these drawbacks, a model based on LRBR method has weak extensibility.

3. Materials and tools

For the investigation in this paper, the STEP format for storing geometric information of 3D models was used. STEP is a widely used exchange standard for product data, for which different Application Protocols (APs) are defined for different use-cases. In this work AP242 according to ISO 10303-AP242:2020 [55] was used, which allows to store MBDs and PMI as it is typically found in technical drawings (geometric dimensions and tolerances, surface specifications, etc.). Product information that is not represented by the

geometry and would be lost in other STEP APs can be annotated in the 3D model and exported by STEP AP242. Currently, the function of exporting thread dimensions is not supported (yet) by this protocol.

In addition to the analysis of product data, a knowledge representation method is required to represent and store extracted information, geometric features, and other product data and their relations. In this work, due to the richness of modelled knowledge, an ontology as knowledge representation method is used. The ontology data format used is OWL (Web Ontology Language) [56]. Despite the possibility of modifying and writing an OWL file directly, a graphical user interface application can greatly simplify this process. In this work, the application "Protégé" [57] from Stanford University is used to manage the ontology.

Furthermore, a Python script for STEP file analysis and extraction process is developed in this work. The Python package Owlready2 is used in this work to exploit contents of the knowledge base and process them using Python code. Owlready2 is an open-source Python package introduced and developed by Jean-Baptiste Lamy [58]. This package provides the possibility to load and modify objects of an ontology as Python objects (such as classes, instances and attributes). Furthermore, with help of Owlready2 it is possible to load several ontologies at the same time in Python, which allows to exploit the content of ontologies and Knowledge Graphs.

4. Methodology

After explaining materials and tools used in this work, the methodology and their implementation is described in this chapter. The proposed model can be divided into two parts. In the first part, the entities of the STEP schema (e.g. faces and edges) are translated into a KG. Here, existing approaches from the state-of-the-art of the authors Ketan and Yaqoub 2010 [20], Perzylo et al., 2015 [54] and Wang and Yu 2014 [8] are used and implemented in self-developed software and KGs. In the second part, semantically described features are derived from the elements stored in the KG using ontology-based logical rules. As described in chapter *Related Work*, this method has the disadvantages that on the one hand a low extendibility is given and on the other hand interacting features cannot be recognized sufficiently. The compensation of these disadvantages takes place in this work in two steps: (1) a hierarchical structure of feature libraries, according to Wang and Yu [8], described in a taxonomy, and (2) by a layered model of different geometry- and manufacturing-related feature types. The novelty of this work consists in the combination of the hierarchical approach of Wang and Yu [8], with a novel, ontology-based, layered feature recognition model. The underlying ontologies, taxonomies, KG, logical rules, and software source code were developed by the authors of this work themselves and hence, represent a scientific novelty. Furthermore, the feature libraries were developed following current standards of manufacturing technology, which is also to be considered a novelty.

Fig. 2 shows the structure of our proposed model, which will be described in the following. The input of the system is a STEP file. In the first step, a Python module extracts the required product data from this file. Before data can be stored in an ontology, the corresponding STEP schema must first be mapped into an ontology. Extracted data can then be subsequently stored in the ontology. After this mapping, another Python module recognizes and semantically describes the component's geometry in terms of features. Input for this module are STEP instances that were stored in the KG. Recognized features are then saved in the KG.

4.1. Extraction of STEP-Entities

The feature recognition process consists of two steps: (1) pre-treatment and (2) recognition. In the pre-treatment step, entities extracted from a STEP file are analysed to derive required high-level information, such as connection types. In the recognition step, features are recognized using existing information and logical rules. First, the STEP schema is mapped into an ontology. Then, features and their corresponding relations to each other and to other geometric elements are defined and stored in the KG.

In this section, the workflow for extracting STEP instances and storing these instances in the KG is presented. The workflow consists of three steps: (1) mapping of STEP schema, (2) extracting instances, and (3) storing instances (see also Fig. 2).

To store the instances of a STEP file in the ontology, a mapping of the STEP schema in an ontology is required. In this mapping, objects of the STEP schema are represented as *classes* or *properties*. After that, STEP instances extracted from a STEP file are stored

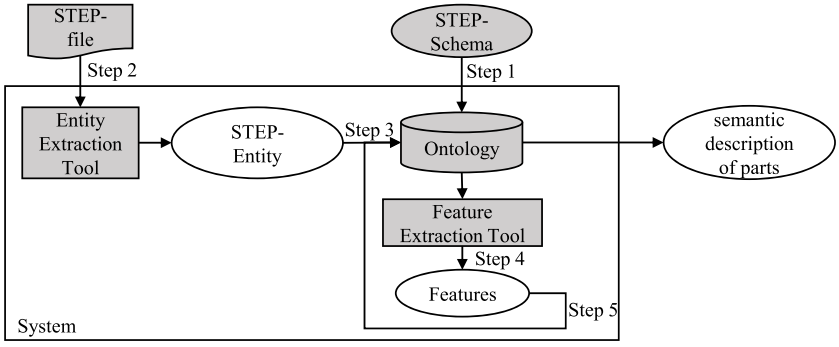


Fig. 2. Illustration of the proposed model with input and output data.

under the corresponding classes.

As described before, the STEP schema has a similar structure as OWL. Therefore, most objects of the STEP schema can be directly represented in ontologies. Fig. 3 shows the representation of STEP entities in the ontology. STEP entities are represented as classes in the ontology, and attributes of entities are represented as properties. Similar methods have been used in some previous research. The basic idea of mapping the STEP schema is inspired by Ketan and Yaqoub [20], Perzylo et al. [54] and Wang and Yu [8].

Although most objects of the STEP schema are similar in structure to OWL objects, there are exceptions. The biggest difference between STEP schema and OWL are data types (*datatype* in OWL and *TYPE* in STEP schema). The OWL datatypes are essentially classic datatypes such as *int*, *float*, *Boolean*, *string*, etc. However, data types of the STEP schema are more complex. In addition to the classic data types, the STEP schema has specially defined data types such as *measure_value*, *mass_value*, and *geometric_tolerance_target*. All data types defined in STEP schema that represent a number or a string can be considered as subtypes of respective classical data types, since these data types inherit properties of respective classic data types. In this work, these special data types are mapped as their corresponding simple data types in the ontology. An accompanying drawback is the loss of semantic meaning of these datatypes. For example, if the datatype *mass_value* is mapped as a *float*, it is no longer possible to recognize that the content of this datatype is a weight measurement. This problem was solved by using additional properties. The semantic meaning of datatype *mass_value* can be represented by object properties like *hasMassValue* combined with datatype *float*.

Except for complex data types, there is a data type *list* in the STEP schema. Although OWL has no such data type, it is possible to represent this data type via a workaround. In ontologies, a property can connect any number of objects, which is why a list can be represented by using a property multiple times. While the content of the list can be mapped, the order of list elements is not yet mapped. In STEP schema elements of a list have a certain order. This information is particularly important for the representation of B-spline lines. In the representation of B-spline lines, nodes of lines are stored as a series of coordinates in a list. Changing the order of these nodes can lead to changing the shapes of a B-spline line.

In the paper of Ketan and Yaqoub [20] a solution for this problem was presented. For each element of a list, an individual is created in the KG. First, all individuals are connected to the list. Then individuals are connected according to the order by the object properties *hasNext* and *isNextTo*. With the help of these properties, one can successfully map the order in an ontology. In this work, this method is used to represent nodes of B-spline lines (see Fig. 4).

This section of our paper focuses on how STEP instances can be modelled as OWL individuals in a KG. As described previously, the STEP schema is mapped into an ontology in order to further process individuals and their dependencies. This ontology contains classes as well as their relations and is called *TBox* (Terminological Box). Individuals of these classes are described in the KG. The ontology containing individuals is called *ABox* (Assertional Box). By separating the *ABox* from the *TBox*, the structure of our model can be made more flexible and clearer, and the concept and application scenarios can be considered in a more modular way. Fig. 5 shows the workflow of this process step. Modelled individuals can be linked to the respective objects by properties, just like instances of STEP files. For example, an instance of an entity *line* is connected to an instance of an entity *cartesian_point*. Furthermore, the instance of *cartesian_point* is connected to a list of numbers. This can be used to describe the geometry and position of a line.

4.2. Feature recognition

After extracting STEP instances, geometric data of the STEP file is stored in a KG. This data was used to perform the feature recognition in this work. To determine the design of the recognition process, a suitable feature recognition method has to be selected first. In this work, an ontology-based LRBR method was used in order to compensate for the drawbacks of this method. In the following, the compensation methods used in this work are presented.

4.2.1. Hierarchical structure

One major drawback of the LRBR method are complex logical rules for feature recognition. This drawback is in particular

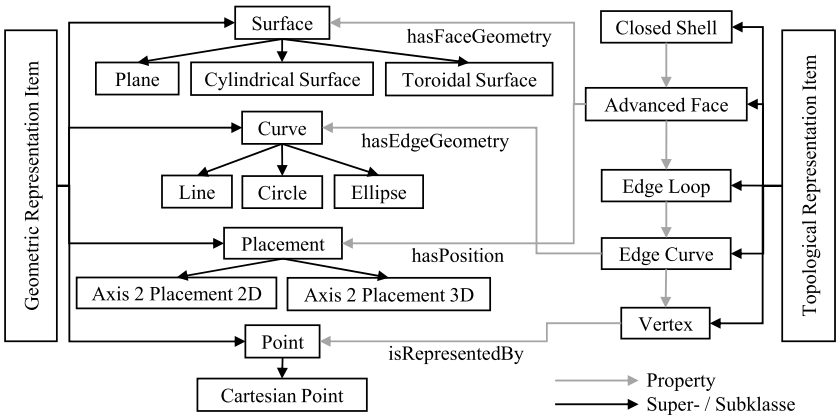


Fig. 3. STEP Schema (entities and attributes) which is represented in the ontology (according to Refs. [8,20,54]).

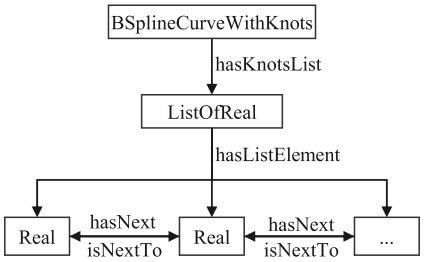


Fig. 4. Representation of knots of a B-Spline-Line in ontology according to [20].

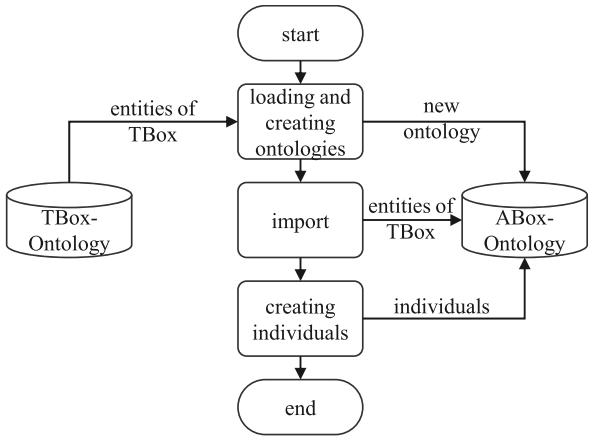


Fig. 5. Process of loading, creating and saving extracted information in the ontology.

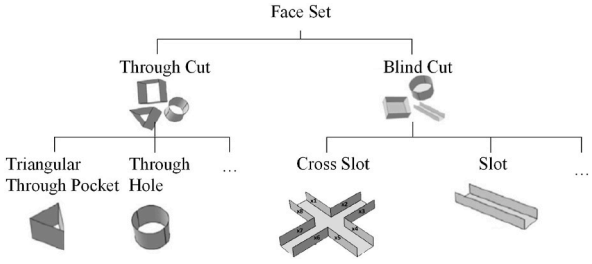


Fig. 6. Taxonomy of features according to wang and yu [8].

noticeable in the detection of complex interactive features. Since interactive features consist of simple features, it is reasonable that much of the information derived from the recognition of simple features can also be used for the recognition of interactive features. Therefore, complex logical rules can be simplified by reusing previously derived information.

Wang and Yu present an ontology-based AFR model in their work [8]. In their model, features are hierarchically defined as superclasses and subclasses in an ontology (see Fig. 6). Due to the properties of ontologies and taxonomies, subclasses can inherit definitions of superclasses, allowing the reuse of previously inferred information. This hierarchical feature library significantly reduces the required logical rules in feature recognition. For example, *Through Cut* is a group of faces that are connected as a loop. *Through Hole* is a subclass of *Through Cut* and inherits the definitions of this class. Therefore, the constraint for relationships between the faces is not required for the subclass, only a specific constraint on shapes of faces.

In this work, the hierarchical structure of the feature library from Wand and Yu [8] was adapted, to reduce the complexity of definitions of complex features. At the same time, the extensibility of feature models can be improved by reducing the work required to extend new features. However, there is still potential for improvement in reusing information. The hierarchical structure of feature library enables the description of features by vertical relations between subclasses and superclasses, however, interactive features cannot be described ideally in this way yet. For example, *Cross Slot* in Fig. 6 is a combination of four Slots. The relation between *Cross Slot* and *Slot* is not "*Cross Slot* is a *Slot*", but "*Cross Slot* consists of four *Slots*". Therefore, this relation cannot be described by subclasses and superclasses. A horizontal relation between two features defined by properties is needed. The challenge here lies in the fact that *Cross Slot* is on the same hierarchical level as *Slot*. The description of relation of two objects standing on the same hierarchy level is not

logical and could complicate the hierarchy. In order to describe this horizontal relation more logically, simple features and complex features (which consist of simple features) should be on two different levels. Therefore, in this paper, a novel layered structure of the feature library is presented to solve this problem. This layered structure also serves as a solution for separating the representation module from the recognition module and thus a solution for integrating modules.

The main requirement for BFs is a high degree of coverage. In this work, a high degree of coverage is defined as the usage of every face of a 3D model for the definition of at least one BF. Hence, the complete part is represented by BFs. If a face is not assigned to a BF, information of this face is lost for the definition of MFs. A direct solution to this would be to list all possible geometric elements. This is nearly impossible and would be also very time-consuming. The solution to this is to define features topologically using a hierarchical classification. For example, cylindrical, rectangular, or conical holes are geometrically different, but these features can be described topologically as "a set of the concave connected faces that together form a closed curve". The three features can be classified as belonging to the class hole. It must be guaranteed that the top hierarchy of BFs can cover all situations, so that all possible situations can be classified as subclasses of that level. With the help of inheritance of information and properties, it is possible to extend features downwards easily and quickly.

In this paper, the top hierarchy of the BF taxonomy is classified according to topological properties into the following types: *pocket*, *hole*, *step*, *slot*, *boss* and *contour*. For each feature additional variants are existing. The goal of this classification is to cover all possible situations at the top level of the taxonomy. Through further constraints, these features can be defined more precisely in an hierarchy, as shown in Fig. 7. In Table 1 the BFs illustrated in Fig. 7 are described.

The definitions of BFs are mainly based on connection types between individual faces. Connection types are extracted using the developed Python script and stored in the ontology. In this software tool, the base faces of a 3D model are used as starting point. Therefore, during the recognition process, base faces are first determined and extracted. After determining the base faces, the type of connection to adjacent faces is determined. Base face, adjacent faces and connection types of them are used to recognize BFs.

MFs have the purpose of providing semantic information relevant to manufacturing processes. There is no hierarchical relation between MFs and BFs, but relations described by properties. MFs are connected to BFs by properties such as *isDefining* and *isDefinedBy* (see Fig. 10). Within MFs, there are sub- and superclasses. Thus, it is possible to define new MFs by a finer detailing of the top level of the taxonomy. This process is similar to the extension of BFs. The introduction of MFs was done by the authors of this work in a previous publication [4] using current manufacturing engineering standards. Fig. 8 shows the MFs defined in Ref. [4]. The exact definition of the individual features won't be described in this paper, as it focuses on the extraction of BFs.

Using detected BFs, complex MFs can be defined by combining BFs and be added to the feature library without changing other parts of the model. Using the inherited information, the complexity of logical rules can be reduced. In general, the definition of a MF consists of four conditions; (1) composition of BFs, (2) geometric shapes of BFs, (3) geometric orientation of BFs in three dimensions, and (4) relative position of BFs. First, the required BFs are taken from the ontology. Then, the constraints are checked. The conditions include shapes, directions, and positions of BFs. If all conditions are met, an individual is created for a MF in the ontology. This individual is connected to the relevant BFs by the property *isDefinedBy*.

The process of detecting MFs is presented by using the example of an *Outer Rotary Feature* (see Fig. 11a). The conditions for the exemplary *Outer Rotary Feature* are:

Composition: *Outer Rotary Feature* is to be defined from two interconnected cylindrical bosses;

Shape: The bases of those two *Boss* features are defined by a flat circular face;

Orientation: The normal vectors of those two bases are oriented in opposite directions;

Position: The side faces and bases of those two *Boss* features shall be coaxial.

The topological and geometrical description of a component is achieved by Basic- and MFs. For a comprehensive semantic description of a component, however, properties such as dimensions, fits or tolerances still have to be extracted and described. For this

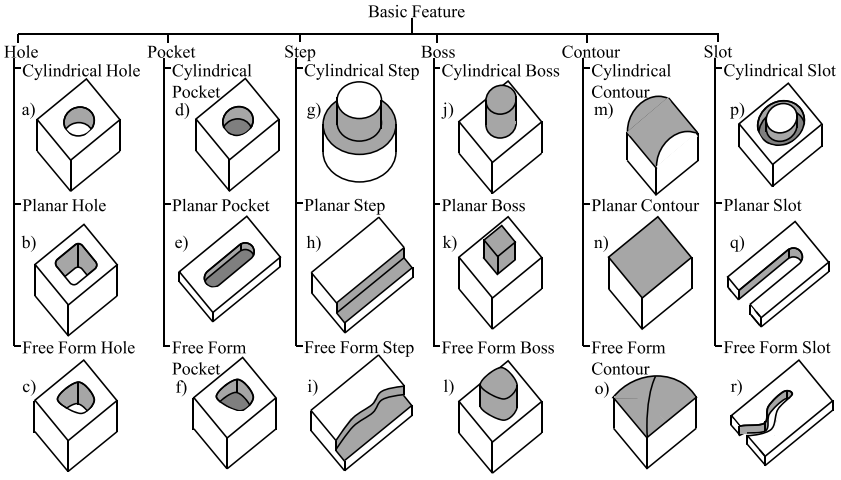


Fig. 7. Taxonomy of basic features.

Table 1
Definitions of BFs.

Basic Feature	Rule
Hole	A *Hole* is defined by a series of concave connected faces. These faces together form a closed face (see Fig. 7). A *Hole* has no base, but by extending the definition of a base face, the feature *Hole* can be determined. As shown in Fig. 11 b, *Hole* can be identified by a series of inner edges of an outer base face. In a STEP file, interconnected inner edges are assigned under an *edge_loop*. If edges of an *edge_loop* are concave, the faces connected to the base by these edges together form a *Hole*. The classification of the *Hole* class can be described as follows. *Cylindrical Hole*: A *Cylindrical Hole* is a *Hole* that has only one cylindrical side face (see Fig. 7a); *Planar Hole*: A *Planar Hole* is a *Hole* that has only flat side faces. (see Fig. 7b); *Free Form Hole*: A *Free Form Hole* is a *Hole* that has at least one B-spline face (see Fig. 7c).
Pocket	A *Pocket* is defined by a series of concave connected faces. These faces together form a closed face and have a concave connected base face (see Fig. 7). All outer edges of the base face are connected to the base face in a concave pattern. The classification of the *Pocket* class can be described as follows. *Cylindrical Pocket*: A *Cylindrical Pocket* is a *Pocket* that has only one cylindrical side face (see Fig. 7d); *Planar Pocket*: A *Planar Pocket* is a *Pocket* that has only the flat side faces (see Fig. 7e); *Free Form Pocket*: A *Free Form Pocket* is a *Pocket* that has at least one B-spline face (see Fig. 7f).
Step	A *Step* is defined by a side face with a concave connected base face. At least one of the faces is not part of another feature (see Fig. 7). According to the definition of *Step*, every *Pocket* and *Slot* can be considered as a combination of *Steps*. However, *Pockets* and *Slots* have a higher semantic meaning. In order to avoid the double recognition, base faces already recognized as *Pocket* or *Slot* are not checked by the conditions of *Step*. If a base face belongs to neither *Pocket* nor *Slot*, the outer edges of the face are checked separately. The face connected to the base face by a concave edge and the base face together form a *Step*. The classification of the *Step* class can be described as follows. *Cylindrical Step*: A *Cylindrical Step* is a *Step* whose side face is cylindrical (see Fig. 7g); *Planar Step*: A *Planar Step* is a *Step* whose side face is flat (see Fig. 7h); *Free Form Step*: A *Free Form Step* is a *Step* whose side face is a B-spline face (see Fig. 7i).
Boss	A *Boss* can be seen as a series of *Steps*. Since a *Boss* has more semantic information than a *Step*, *Boss* is also defined as a BF. A *Boss* is a set of side faces that are concavely connected to the base face by the inner edges of the base face (see Fig. 7). In a STEP file, interconnected inner edges are assigned under an *edge_loop*. If edges of an *edge_loop* are convex, the faces connected to the base by these edges together form a *Boss*. The classification of the *Boss* class can be described as follows. *Cylindrical Boss*: A *Cylindrical Boss* is a *Boss* that has only one cylindrical side face (see Fig. 7j); *Planar Boss*: A *Planar Boss* is a *Boss* that has only the flat side faces (see Fig. 7k); *Free Form Boss*: A *Free Form Boss* is a *Boss* that has at least one B-spline face (see Fig. 7l).
Contour	A *Contour* is defined by a face that has no outer concave connection (see Fig. 7). All outer edges of the base face are connected to the base face in a convex pattern. The classification of the *Contour* class can be described as follows. *Cylindrical Contour*: A *Cylindrical Contour* is a *Contour* whose face is cylindrical (see Fig. 7m); *Planar Contour*: A *Planar Contour* is a *Contour* whose face is flat (see Fig. 7n); *Free Form Contour*: A *Free Form Contour* is a *Contour* whose face is a B-spline face (see Fig. 7o).
Slot	A *Slot* can be described as two or more adjacent *Steps*. First, all edges of the base face are analysed. Two edges parallel to each other are taken as one edge group. After taking these edge groups, all groups are checked by the conditions of the Slot types below: *Cylindrical Slot*: A *Cylindrical Slot* has two cylindrical side faces. The radii of the two side faces are not equal. The axes of the two side faces are equal (see Fig. 7q); *Planar Slot*: A *Planar Slot* has two parallel flat faces. These two faces are concavely connected to a base face (see Fig. 7p); *Free Form Slot*: A *Free Form Slot* is a slot whose side faces are B-spline faces (see Fig. 7r).

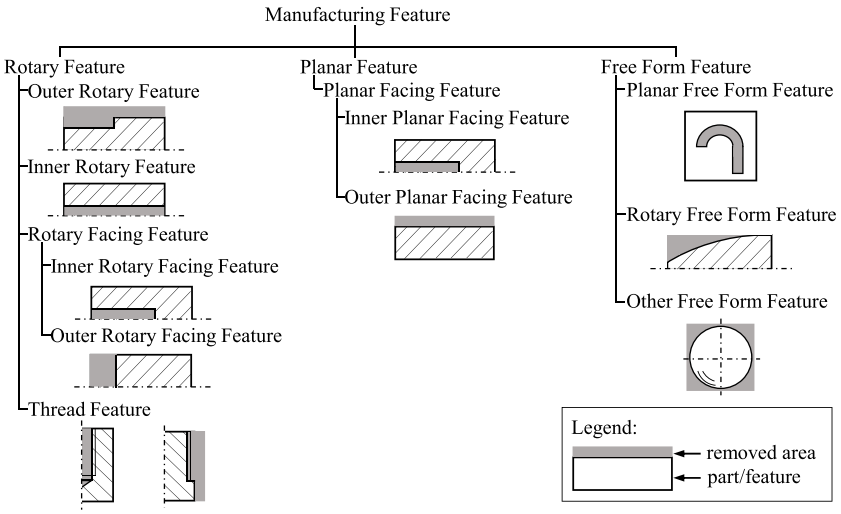


Fig. 8. Taxonomy of Manufacturing Features according to Koehler et al. [4].

purpose, the taxonomy of Manufacturing Restriction (see Fig. 9) was presented in a previous work [4]. The taxonomy is modelled on current standards in manufacturing and design engineering. This part of the KG contains production-relevant information, such as Geometric Dimensioning and Tolerancing (GD&T), minimum manufacturable feature dimensions and surface properties. In this work, the taxonomy is only briefly discussed for the sake of completeness and is not described further, since the extraction of the Manufacturing Restrictions is not considered.

4.2.2. Layered structure

In our proposed layered structure, the feature library is divided into two levels: (1) geometry-related BFs and complex, manufacturing-related MFs that can be used directly to determine manufacturing processes. BFs are described by the combination of faces and serve as building units for MFs. A MF consists of one or more BFs and may contain data relevant to manufacturing. Ideally, each face of a 3D model should be assigned to at least one BF. MFs are defined and recognized by the combinations or modifications of BFs with appropriate rules.

With the help of this structure, the relations between BFs and interactive features can be described clearly and logically. Furthermore, the representation of a 3D model is separated from the recognition of features. A 3D model is represented by BFs. Definitions of BFs are independent of definitions of MFs. Thus, a component can be completely described by abstract MFs. These in turn are defined geometrically by the BFs in a second layer. BFs are defined by STEP entities mapped in the KG. Fig. 10 shows the described relationships in a graphical representation of the ontology. This structure allows relations between simple and interactive features to be described clearly and logically. Furthermore, the representation of a 3D model is separated from the recognition of features. The 3D model is represented by BFs. Definitions of BFs are independent of definitions of MFs.

5. Results and discussion

The basic function of the presented model is a semantic description of the geometry of 3D models by extracting features and storing extracted information in a knowledge base. In addition to the basic function, the model should have a good extensibility, so that the extension with additional features can be realized quickly. In this chapter, the results of the analysis of the model are presented. Subsequently, the model is discussed based on the presented objective.

To demonstrate the capability of the developed model, two test components are used and analysed. Both 3D models were exported in STEP file (AP242) format. One test component was created in this work, (see Fig. 12), another test component was taken from Wang and Yu's [8] work (see Fig. 13). This second sample part is used in this paper to get a comparison to the state-of-the-art and to check the general applicability of the model. Fig. 12a shows benchmark part 1 and the features to be detected, Fig. 12b shows the actual detected features. The detected features are explained in the following. *Slot_y_1* is actually a combination of *Step* and *Boss*. However, a *Slot* can provide all information of a *Step*, which is why *Step* is not stored as an individual at this point. *Slot_y_6* to *Slot_y_9* are created by the combination of *Pocket_y_1* and *Boss_y_1*. The distance between the side faces of the pocket and the side faces of the boss is smaller than the length of these side faces, therefore these side faces form four slots. *Step_y_5* to *Step_y_7* are recognized together as a slot in x-direction. According to the definition, the lengths of the two side faces of the slot should be greater than the distance between the side faces. In the y-direction this condition is not fulfilled, therefore *Step_y_5* to *Step_y_7* are recognized in the y-direction only as Steps. At the position of *OuterRotaryFeature_1* there are two connected *Boss* Features. As introduced, these *Boss* features are recognized together as one MF *Outer Rotary Feature*. *Slot_y_10* is at a certain angle to the y-direction, since the angle between the working direction of this feature and the y-axis is less than 45°, *Slot_y_10* is also recognized as a feature in y-direction. The exact direction of *Slot_y_10* can be detected by its footprint.

While most features are detected as desired, there are some features that are not recognized. Those features are highlighted in Fig. 12a. On benchmark part 1 there is a *Planar Slot* which has a cylindrical base (compare the left side of the sample part in Fig. 12a). This *Slot* is not recognized. In the centre of benchmark part 1, an angled *Cylindrical Boss* should be detected. However, this *Boss* is not detected. Next to *Step_y_2* there is also a *Free Form Step*, which is also not recognized. Furthermore, *Step_y_8* to *Step_y_13* should be recognized together as one *Slot*. However, these are only recognized separately as *Steps*.

Furthermore, the capabilities of the model are tested by analysing a state-of-the-art component. For this purpose, a component from the publication by Wang and Yu [8] is used. The features present and recognized on benchmark part 2 are shown in Fig. 13. Fig. 13a show the features to be detected on benchmark part 2 and Fig. 13b the features which were actually detected.

As shown in Fig. 13b, the model can recognize all features of the second benchmark part. Furthermore, all faces of the 3D model are represented by features. The model can recognize not only *Straight Slots* but also *Cylindrical Slots*. *Cylindrical Slots* (*Slot_y_1* and *Slot_y_2*) are stored under the *Cylindrical Slot* class. In the current state of our model, the *Pocket* class is divided into *Cylindrical Pocket*, *Planar*

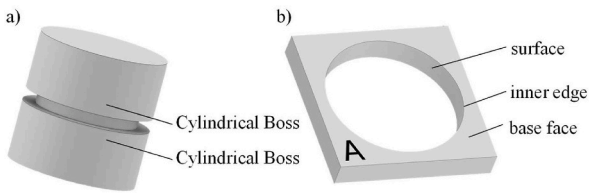


Fig. 9. Taxonomy of Manufacturing Restrictions according to Koehler et al. [4].

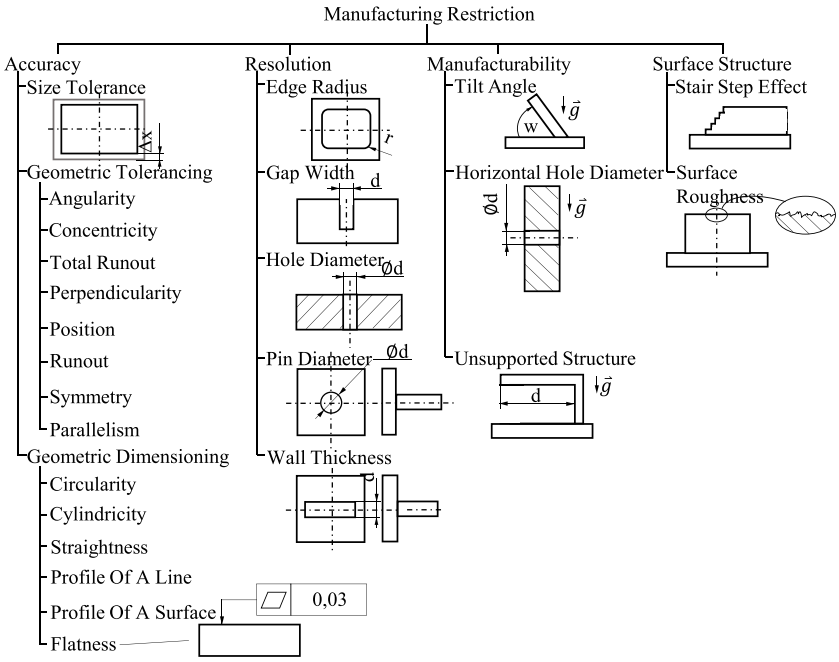


Fig. 10. Structure of relations between the main classes in the proposed ontology.

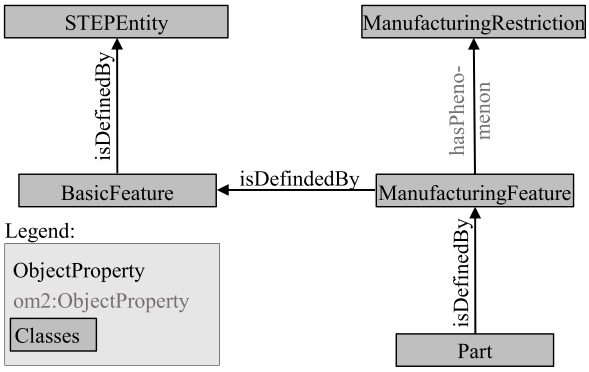


Fig. 11. Illustration of Features a) MF *Outer Rotary Feature*; b) BF *Hole*.

Pocket and *Free Form Pocket*. *Pocket_y_1* is currently assigned to *Planar Pocket*.

In Wang and Yu's [8] benchmark part, a total of two combined features are defined. These are *T-Slot* and *Holed-Blind-Pocket* (see Fig. 13). To determine these combined features with our model, *Slot_y_3*, *Slot_y_4* and *Pocket_y_2* must be recognized together as *T-Slot*. Due to the layered and hierarchical structure of the feature library, it is possible to combine simple features into combined features. If a combined feature is useful for further analysis, a complex MF can be created by combining BFs.

The analysis of the first benchmark part showed that the model can recognize five different types of BFs (*Pocket*, *Hole*, *Slot*, *Step* and *Boss*). Furthermore, it is possible to classify those five types of BFs more finely into 14 variants. It could be shown that the feature recognition is not disturbed by the connection of features to each other (for example *Pocket_y_1* and *Boss_y_1* as well as *Slot_y_2* to *Slot_y_5*). Furthermore, it can be stated that the recognized features are correctly stored in the KG as instances of the respective classes. Through the associated properties, geometric and topological information of features can be recognized and assigned. In summary, the proposed model fulfils the main function of an AFR model (feature extraction). However, it must be noted that some features are not yet recognized as desired. In the following, the extensibility of the model, as well as the ability to handle complex features will be discussed. The definition of the subclasses of the BFs (such as *Cylindrical Pocket* and *Planar Pocket*) is based on a hierarchical structure of the feature library. The recognition of extended BFs (subclasses) represents a subclassification of the superclasses of the BFs. The previously derived information can thus be reused (inherited). The results of the analysis of benchmark part 1 have shown that the extended BFs can be correctly recognized by this method.

With the help of the layered structure of the feature library, the mapping of a 3D model in an ontology and the recognition of

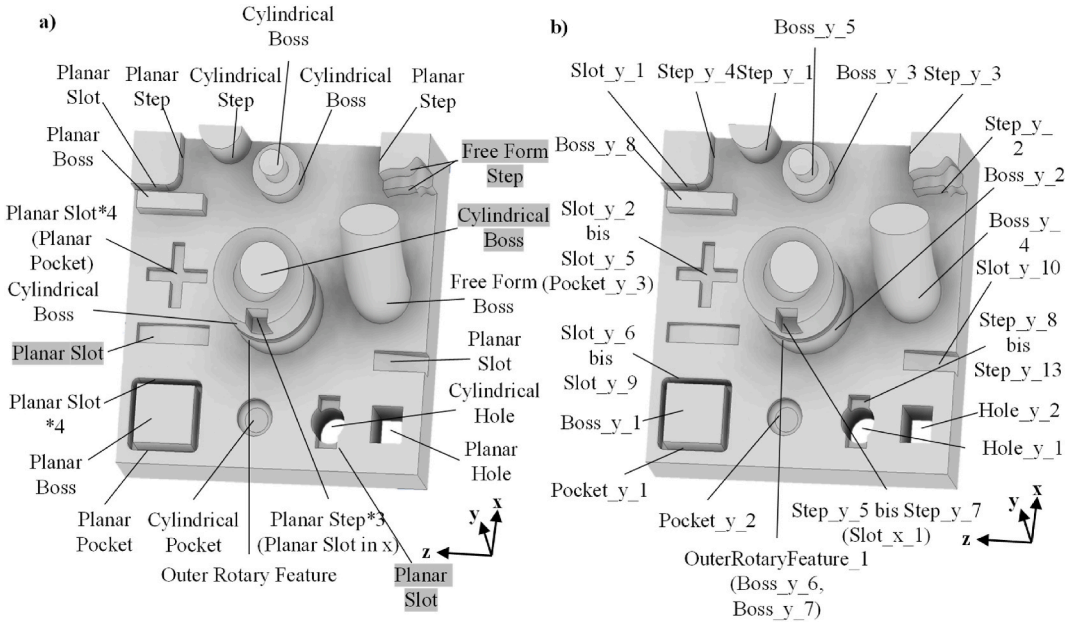


Fig. 12. a) Illustration of features, which should be recognized for benchmark part 1; b) Illustration of features, which were recognized by the proposed method for benchmark part 1.

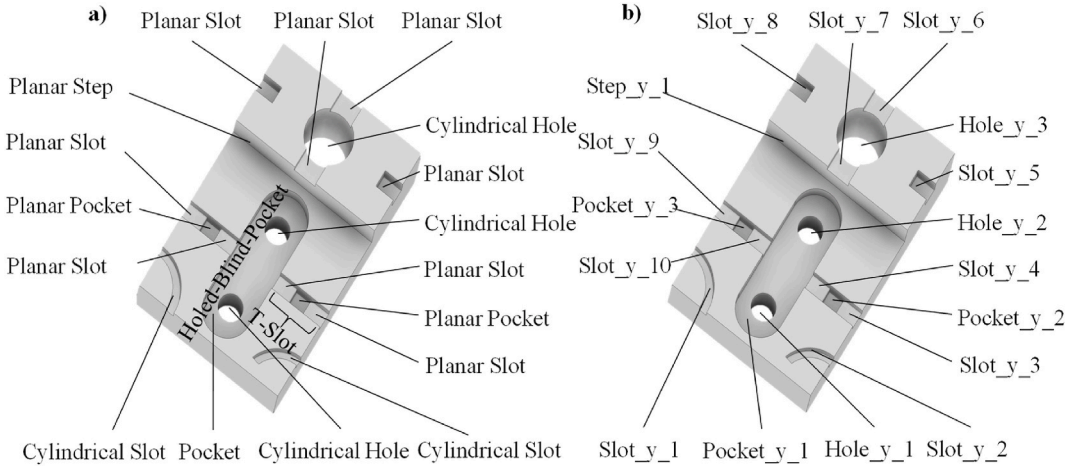


Fig. 13. a) Illustration of features, which should be recognized for benchmark part 2 (designed according to Ref. [8]); b) Illustration of features, which were recognized by the proposed method for benchmark part 2 (designed according to Ref. [8]).

features are separated from each other. The definition of BFs is done by analysing all existing faces. Thus, all faces of a 3D model can be assigned to at least one BFs. It could be shown, that the representation method presented in this paper works theoretically and practically. Due to the layered structure, the MFs can be defined as the combinations of BFs. In this work, *Outer Rotary Feature* is defined as a MF combined from BFs. The definition of *Outer Rotary Feature* is considerably simplified because many geometric and topological definitions for *Outer Rotary Feature* are already determined when defining a *Boss* Feature. The results have shown that our model can correctly recognize MF as a combination of two BFs.

6. Conclusion

As a link between design and manufacturing, an automated process planning system has the task of automatically analysing a 3D model and then determining suitable manufacturing steps and machines. A 3D model is described by geometric and topological information. However, this information cannot be used directly for planning. Therefore, a semantic description of the 3D model, which is useful for the manufacturing process, is required. This description can be done by defining so-called Manufacturing Features (MF). An

AFR (Automatic Features Recognition) model derives these features from geometrical and topological information of the 3D component.

The aim of the presented work is the development of a model for a semantic description of 3D CAD data in knowledge bases. This goal was achieved by developing an AFR model that extracts features of 3D models and then stores and classifies them in a Knowledge Graph. One of the main requirements for the proposed method is an easy extensibility, so that the features library can be enlarged in the future.

The task of the developed model can be divided into product data extraction and feature recognition. In the first step, geometric and topological data of a 3D model (STEP format) generated by a CAD system are extracted by the developed model and stored in a knowledge base. The extracted product data is then analysed to detect features relevant to manufacturing processes.

In this work, the STEP standard (AP242) according to ISO 10303–242 [55] is used as the format of the CAD files. This information is defined in the form of instances in the STEP schema and stored in the STEP file. Due to the similar structure of an ontology and the STEP schema, a Knowledge Graph in OWL format is used as the knowledge base in this work. Instances of a STEP file are extracted by a software algorithm developed in this work. Then, extracted information can be stored in the developed ontology by our algorithm. This predefined ontology contains the classes of the STEP schema.

The feature recognition process is also performed by the developed algorithm. The recognition is based on the Logic-Rule-Based-Recognition method. In addition, an ontology-based hierarchical, and layered structure of the feature library is used to enable extensibility of the model and to facilitate the recognition of interactive features.

With the help of the hierarchical structure, relations between features can be described as superclasses and subclasses. Thus, the previously derived information is reusable through inheritance and the definition of new subclasses is simplified. Furthermore, features are divided into two different layers (Basic Features (BF) and MFs). On the first layer, a set of simple primitive features is defined as BFs, these features have the task to represent the 3D model. On the second layer, abstract MFs are defined by the combinations of BFs. These abstract MFs can be used to semantically describe manufacturing processes and manufacturing machines. By analysing the MFs, it is therefore, possible to make a statement about the manufacturability of a component with certain processes. In the hierarchical structure, only relations between superclasses and subclasses are reusable. The layered structure then additionally allows the reuse of information of features described by properties of the ontology. In addition, the classes of the different branches can be linked together, making more information reusable. Consequently, adding new features, especially complex features, can be greatly simplified.

The functionality of the presented approach was proven by the analysis of benchmark parts. However, it was shown that some features are not yet recognized as desired. For example, the rule base for MFs needs to be extended to detect all desired features. Thus, the result of feature recognition strongly depends on the underlying logical rules. If no rules are defined for a particular feature, it is either not recognized, or assigned to a different feature category. In addition, the rule definitions are based on the connection types (concave or convex) between faces, lines and edges of 3D models. If connections are detected incorrectly or not at all, features are also extracted incorrectly.

It was shown that a significant contribution to the area of automatic feature recognition could be contributed by the developed graph-based feature extraction approach. Especially the hierarchical and layer-based approach allows to continuously extend the underlying knowledge base. Thus, it is possible to describe the geometric properties of components semantically and, in particular, in a computer-interpretable format, which marks a novelty for the state-of-the-art. This enables the automated evaluations of these data with the help of computer algorithms and to derive new statements from them, e.g. about appropriate manufacturing machines and processes.

In future work, the feature library of MFs and the fundamental rule base will be extended. Furthermore, the presented model is currently extended by the extraction of Model-Based Definitions (MBD) such as Product Manufacturing Information (PMI). The use of STEP AP242 enables the export of MBD, which allows them to be stored in the Knowledge Graph and used for the automated derivation of new statements. In addition, MBDs such as form and position tolerances are already modelled in the Knowledge Graph (see Fig. 9 and Koehler et al. [4]), which also facilitates extraction.

Author contribution statement

Tobias Köhler: Conceived and designed the experiments; Performed the experiments; Analysed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper. Buchao Song: Conceived and designed the experiments; Performed the experiments; Analysed and interpreted the data; Contributed reagents, materials, analysis tools or data.Jean Pierre Bergmann: Conceived and designed the experiments. Diana Peters: Conceived and designed the experiments; Contributed reagents, materials, analysis tools or data.

Data availability statement

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We thank Qingmai Wang and Yinghou Yu (Royal Melbourne Institute of Technology, Melbourne) for useful discussions on possibilities of ontologies in automatic feature recognition frameworks.

References

[1] H. Kagermann, W. Wahlster, J. Helbig, Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0, Abschlussbericht des Arbeitskreises Indistrie 4.0 (2013).
[2] A. Roth, Einführung und Umsetzung von Industrie 4.0, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
[3] M. Al-wswasi, A. Ivanov, H. Makatsoris, A survey on smart automated computer-aided process planning (ACAPP) techniques, Int. J. Adv. Manuf. Technol. 97 (2018) 809–832, https://doi.org/10.1007/s00170-018-1966-1.
[4] T. Köhler, L. Kleinhenz, P.M. Schäfer, J.P. Bergmann, D. Peters, Development of a Methodology for the Digital Representation of Manufacturing Technology Capabilities. https://doi.org/10.15488/11259..
[5] C. Weber, What is a feature and what is its use: results of FEMEX working group I, in: D. Roller (Ed.), 29th International Symposium on Automotive Technology and Automation: Florence, Italy, 3rd - 6th June 1996; the Largest European Automotive Forum with International Participation, Automotive Automation Ltd, Croydon, 1996.
[6] J. Han, M. Pratt, W.C. Regli, Manufacturing feature recognition from solid models: a status report, IEEE Trans. Robot. Autom. 16 (2000) 782–796, https://doi.org/10.1109/70.897789.
[7] B. Babic, N. Nesic, Z. Miljkovic, A review of automated feature recognition with rule-based pattern recognition, Comput. Ind. 59 (2008) 321–337, https://doi.org/10.1016/j.compind.2007.09.001.
[8] Q. Wang, X. Yu, Ontology based automatic feature recognition framework, Comput. Ind. 65 (2014) 1041–1052, https://doi.org/10.1016/j.compind.2014.04.004.
[9] S. Haasis, Integrierte CAD-Anwendungen: Rationalisierungspotentiale und zukünftige Einsatzgebiete, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995.
[10] G. Spur, F.-L. Krause, Das virtuelle Produkt: Management der CAD-Technik, Hanser, München, 1997.
[11] H.-P. Wiendahl, D. Gerst, L. Keunecke, Variantenbeherrschung in der Montage, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
[12] e.V. Verein Deutscher Ingenieure, Informationsverarbeitung in der Produktentwicklung: Feature-Technologie, vol. 40, Beuth Verlag GmbH, Berlin ICS 03, 2003, 100.
[13] M.W. Humpa, CAD-Methodik zur Produktivitätssteigerung in der Prozesskette Konstruktion-Fertigung, Dissertation, 2016.
[14] O. Tietze, Gabler edition wissenschaft, in: Strategische Positionierung in der Automobilbranche: Der Einsatz von virtueller Produktentwicklung und Wertschöpfungsnetzwerken, Deutscher Universitätsverlag, Wiesbaden, 2003.
[15] S. Vajna, C. Weber, K. Zeman, P. Hehenberger, D. Gerhard, S. Wartzack, CAx für Ingenieure: Eine praxisbezogene Einführung, third., vollständig neu bearbeitete Auflage, Springer Vieweg, Berlin, Germany, 2018.
[16] X. Zhang, A. Nassehi, S.T. Newman, Feature recognition from CNC part programs for milling operations, Int. J. Adv. Manuf. Technol. 70 (2014) 397–412, https://doi.org/10.1007/s00170-013-5275-4.
[17] M. Al-wswasi, A. Ivanov, A novel and smart interactive feature recognition system for rotational parts using a STEP file, Int. J. Adv. Manuf. Technol. 104 (2019) 261–284, https://doi.org/10.1007/s00170-019-03849-1.
[18] V.B. Sunil, S.S. Pande, Automatic recognition of machining features using artificial neural networks, Int. J. Adv. Manuf. Technol. 41 (2009) 932–947, https://doi.org/10.1007/s00170-008-1536-z.
[19] A.K. Verma, S. Rajotia, A review of machining feature recognition methodologies, Int. J. Comput. Integrated Manuf. 23 (2010) 353–368, https://doi.org/10.1080/09511921003642121.
[20] H.S. Ketan, Z.H. Yaqoub, Build automatic feature recognition system based on sweeping primitive rule, in: Seventh Jordanian International Mechanical Engineering Conference, 2010, pp. 15–30.
[21] S. Meeran, M.J. Pratt, Automated feature recognition from 2D drawings, Comput. Aided Des. 25 (1993) 7–17, https://doi.org/10.1016/0010-4485(93)90061-R.
[22] D. Yip-Hoi, D. Dutta, Z. Huang, A customizable machining feature extraction methodology for turned components, J. Manuf. Syst. 22 (2003) 82–98, https://doi.org/10.1016/S0278-6125(03)90007-0.
[23] M. Flasiński, J. Jurek, Fundamental methodological issues of syntactic pattern recognition, Pattern Anal. Appl. 17 (2014) 465–480, https://doi.org/10.1007/s10044-013-0322-1.
[24] K.S. Fu, Introduction to syntactic pattern recognition, in: K.S. Fu, W.D. Keidel, H. Wolter (Eds.), Syntactic Pattern Recognition, Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 1977, pp. 1–30.
[25] D.-B. Perng, Z. Chen, R.-K. Li, Automatic 3D machining feature extraction from 3D CSG solid input, Comput. Aided Des. 22 (1990) 285–295, https://doi.org/10.1016/0010-4485(90)90093-R.
[26] A. Arivazhagan, N.K. Mehta, P.K. Jain, Development of a feature recognition module for tapered and curved base features, Int. J. Adv. Manuf. Technol. 39 (2008) 319–332, https://doi.org/10.1007/s00170-007-1212-8.
[27] J. Zhu, M. Kato, T. Tanaka, H. Yoshioka, Y. Saito, Graph based automatic process planning system for multi-tasking machine, J. Adv. Mechanical Design, Sys. Manufact. 9 (2015).
[28] K. Rahmani, B. Arezoo, A hybrid hint-based and graph-based framework for recognition of interacting milling features, Comput. Ind. 58 (2007) 304–312, https://doi.org/10.1016/j.compind.2006.07.001.
[29] S. Joshi, T.C. Chang, Graph-based heuristics for recognition of machined features from a 3D solid model, Comput. Aided Des. 20 (1988) 58–66, https://doi.org/10.1016/0010-4485(88)90050-4.
[30] M. Marefat, R.L. Kashyap, Geometric reasoning for recognition of three-dimensional object features, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 949–965, https://doi.org/10.1109/34.58868.
[31] A.-J. Fougères, E. Ostrosi, Intelligent agents for feature modelling in computer aided design, J. Computational Design and Eng. 5 (2018) 19–40, https://doi.org/10.1016/j.jcde.2017.11.001.
[32] W.D. Li, S.K. Ong, A.Y.C. Nee, A hybrid method for recognizing interacting machining features, Int. J. Prod. Res. 41 (2003) 1887–1908, https://doi.org/10.1080/0020754031000123868.
[33] L. Liu, Z. Huang, W. Liu, W. Wu, Extracting the turning volume and features for a mill/turn part with multiple extreme faces, Int. J. Adv. Manuf. Technol. 94 (2018) 257–280, https://doi.org/10.1007/s00170-017-0862-4.
[34] V. Rameshbabu, M.S. Shunmugam, Hybrid feature recognition method for setup planning from STEP AP-203, Robot. Comput. Integrated Manuf. 25 (2009) 393–408, https://doi.org/10.1016/j.rcim.2007.09.014.
[35] P. Holland, P. Standring, H. Long, D. Mynors, Feature extraction from STEP (ISO 10303) CAD drawing files for metalforming process selection in an integrated design system, J. Mater. Process. Technol. 125–126 (2002) 446–455, https://doi.org/10.1016/S0924-0136(02)00364-3.
[36] S. Sivakumar, V. Dhanalakshmi, A feature-based system for CAD/CAM integration through STEP file for cylindrical parts, Indian J. Eng. Mater. Sci. (2013) 21–26.
[37] M.R. Henderson, D.C. Anderson, Computer recognition and extraction of form features: a CAD/CAM link, Comput. Ind. 5 (1984) 329–339, https://doi.org/10.1016/0166-3615(84)90056-3.
[38] M. Kang, J. Han, J.G. Moon, An approach for interlinking design and process planning, J. Mater. Process. Technol. 139 (2003) 589–595, https://doi.org/10.1016/S0924-0136(03)00516-8.

[39] J. Oussama, E. Abdelilah, R. Ahmed, Manufacturing computer aided prosess planing for rotational parts: Part 1: automatic feature recognition from STEP AP203 Ed2, Int. Journal of Engineering Research and Application (2014) 14–25.
[40] L. Zehtaban, O. Elazhary, D. Roller, A framework for similarity recognition of CAD models, J. Computational Design and Eng. 3 (2016) 274–285, https://doi.org/10.1016/j.jcde.2016.04.002.
[41] M. Al-wswasi, A. Ivanov, A features subtraction system for rotational parts based on manufacturing and metal removing concepts, Int. J. Adv. Manuf. Technol. 107 (2020) 1835–1857, https://doi.org/10.1007/s00170-020-05063-w.
[42] P. Arunkumar, A.S. Deshpande, A.C.S. Kumar, A system for extracting product features from CAD models: a step approach, Contemporary Engineering Science (2008) 139–146.
[43] B. Venu, V.R. Komma, D. Srivastava, STEP-Based feature recognition system for B-spline surface features, Int. J. Autom. Comput. 15 (2018) 500–512, https://doi.org/10.1007/s11633-018-1116-0.
[44] J.H. Vandenbrande, A. Requicha, Spatial reasoning for the automatic recognition of machinable features in solid models, IEEE Trans. Pattern Anal. Mach. Intell. 15 (1993) 1269–1285, https://doi.org/10.1109/34.250845.
[45] J. Han, A.A.G. Requicha, Integration of feature based design and feature recognition, Comput. Aided Des. 29 (1997) 393–403, https://doi.org/10.1016/S0010-4485(96)00079-6.
[46] W.C. Regli, Geometric Algorithms for Recognition of Features from Solid Models, Dissertation, Maryland, USA, 1995.
[47] M.R. Henderson, G. Srinath, R. Stage, K. Walker, W. Regli, Boundary representation-based feature identification, in: Advances in Feature Based Manufacturing, Elsevier, 1994, pp. 15–38.
[48] S. Prabhakar, M.R. Henderson, Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models, Comput. Aided Des. 24 (1992) 381–393, https://doi.org/10.1016/0010-4485(92)90064-H.
[49] M.G. Marchetta, R.Q. Forradellas, An artificial intelligence planning approach to manufacturing feature recognition, Comput. Aided Des. 42 (2010) 248–256, https://doi.org/10.1016/j.cad.2009.11.007.
[50] Y. Shi, Y. Zhang, S. Baek, W. de Backer, R. Harik, Manufacturability analysis for additive manufacturing using a novel feature recognition technique, Computer-Aided Design and Applications 15 (2018) 941–952, https://doi.org/10.1080/16864360.2018.1462574.
[51] H. Ma, X. Zhou, W. Liu, Q. Niu, C. Kong, A customizable process planning approach for rotational parts based on multi-level machining features and ontology, Int. J. Adv. Manuf. Technol. 108 (2020) 647–669, https://doi.org/10.1007/s00170-020-05437-0.
[52] A. Malyshev, S. Slyadnev, V. Turlapov, CAD model simplification using graph-based feature recognition, Proceedings of GraphiCon 2017 (2017).
[53] S.S. Madurai, L. Lin, Rule-based automatic part feature extraction and recognition from CAD data, Comput. Ind. Eng. 22 (1992) 49–62, https://doi.org/10.1016/0360-8352(92)90032-F.
[54] A. Perzylo, N. Somani, M. Rickert, A. Knoll, An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),, IEEE, Hamburg, Germany, 2015, pp. 4197–4203, 02.10.2015.
[55] International Organization for Standardization, Industrial automation systems and integration - product data representation and exchange: Part 242: application protocol, Managed model-based 3D engineering 25 40 (2020), 040.
[56] w3c.org, OWL 2 Web Ontology Language: Document Overview (2012). Second Edition, https://www.w3.org/TR/owl-overview/. (Accessed 29 October 2020).
[57] M. Horridge, A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools: Edition 1, 3, 2011. http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf.
[58] J.-B. Lamy, Owlready: ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies, Artif. Intell. Med. 80 (2017) 11–28, https://doi.org/10.1016/j.artmed.2017.07.002.