An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions

Alexander Perzylo¹, Nikhil Somani¹, Markus Rickert¹ and Alois Knoll²

Abstract—In this paper, we introduce an approach for leveraging CAD description to a semantic level, in order to link additional knowledge to CAD models and to exploit resulting synergy effects. This has been achieved by designing a description language, based on the Web Ontology Language (OWL), that is used to define boundary representations (BREP) of objects. This involves representing geometric entities in a semantic meaningful way, e.g., a circle is defined by a coordinate frame and a radius instead of a set of polygons. Furthermore, the scope of this semantic description language also covers geometric constraints between multiple objects. Constraints can be specified not only on the object level, but down to single edges or faces of an object. This semantic representation is used to improve a variety of applications, ranging from shape-based object recognition to constraint-based robot task descriptions. Results from a quantitative evaluation are presented to assess the practicability of this approach.

I. INTRODUCTION

All aspects of industrial products are usually handled in a process called product lifecycle management (PLM). As part of this approach, the geometric product specifications are typically generated with Computer Aided Design (CAD) software. On the other side, a lot of applications are designed to use polygon-based geometry models, which only represent an approximation of the constructed model. This approach has many shortcomings, e.g., it only allows for a fixed level of detail and lacks the means to describe semantically meaningful geometries. For instance, a set of polygons which form a circular shape cannot easily be recognized as such, despite this information being already available at design time. If a circle and its parameters are described by the CAD model instead, the exact mathematical representation of the geometry is known and a triangulation can be carried out at any time using a level of detail that is most suitable to the current application’s requirements. This separation of represented data and use-case dependent calculations is as important as the separation of data and the software that was used to generate it. These paradigms allow flexible sharing and re-use of information.

Apart from polygon based CAD formats, there are other formats that represent mathematical models of the contained geometries, e.g., STEP or IGES. STEP has been developed by the International Organization for Standardization as ISO 10303-21, whereas IGES was defined by the United States National Bureau of Standards as NBSIR 80-1978. Both formats have evolved to become widely accepted standards for CAD data exchange and storage. However, active development of IGES has been discontinued almost two decades ago. Still these formats lack the flexibility to be easily linked with other sources of information. Many pieces of CAD software feature the specification of materials for created geometries, but this information can only be exported to their proprietary data formats and gets lost when the CAD model is exported to STEP or IGES.

In this paper we present a BREP-based (see Section III) semantic representation, i.e., a BREP ontology, of points, curves, surfaces and volumes built using the Web Ontology Language (OWL). A complete specification of BREP’s geometric and topological representation can be found in ISO 10303-42. Apart from the already mentioned benefits of having an open and application independent exchange format for CAD data, there are other major advantages. By having a semantic description language that is based on a logical formalism, it is possible to link to, and to automatically combine knowledge from different sources. For instance, a robot’s CAD model can be extended by linking it with its kinematic model. Automatic reasoning can be used, e.g., to determine the right tools or parameters for applying
certain production steps on a work piece by considering the workpiece’s shape, material, or mass properties. As the second contribution of this paper, we introduce an extension to the BREP ontology (in Section IV), which adds geometric interrelation constraints between points, curves, and surfaces. When composing a robot task description, e.g., an assembly plan, the BREP-ontology-based representation can be used to parameterize the task with a set of geometric constraints between sub-elements of the involved assembly parts. Figure 1 shows a graphical user interface, which allows the definition of such constraints in an intuitive way. This is one of the example applications explained in Section V. The third contribution is a software component to automatically convert standard CAD models, e.g., STEP or IGES, to the proposed ontological representation. In order to use this approach in a real system, certain performance criteria are important. A quantitative evaluation of these aspects are explained in Section VI.

II. RELATED WORK

There have been different approaches for enriching CAD models with semantics. OntoCAD [1] is an ontological annotation approach that is based on labeling geometry elements of CAD models using concepts from a CAD ontology. [2] created an ontology for describing CAD models in the Drawing Exchange Format (DXF). In [3], the authors presented an ontology-based approach of semantically describing CAD data and features. They use a rule system to automatically classify CAD features in order to provide a compatible mapping between different CAD systems. [4] showed how not only CAD models but the complete Product Lifecycle Management (PLM) approach can be semantically modeled. The RoboEarth project [5] followed a holistic approach to knowledge representation and sharing for robots. The RoboEarth language [6], which was built using the Web Ontology Language (OWL), was used to describe many aspects of robot tasks, e.g., task structures, semantic environment maps and models of interaction objects. RoboEarth object models are, however, only meta-descriptions for linked industry-standard CAD models. [7], [8], [9] presented an approach for describing coordinate frames and geometric constraints between the frames for applications in the robotics domain.

In this paper, we introduce a semantic description language to describe not only meta-information for a given CAD model [6], but also the content of the model. Given the complete ontological representation of CAD models, we can define geometric constraints not only on the object level [6] or only on a frame level [8], but also on all semantically meaningful intermediate levels, e.g., points, edges, and faces of CAD models.

III. BOUNDARY REPRESENTATION (BREP)

A Boundary Representation (BREP) of CAD data describes the geometric properties of points, curves, surfaces and volumes using mathematical models as its basis. CAD models are created by defining boundary limits to given base geometries. The BREP specification distinguishes geometric and topological entities, as illustrated in Figure 2. Geometric entities hold the numerical data, while the topological entities group them and arrange them in a hierarchical fashion.

A. Topological Entities

The BREP standard specifies eight kinds of topological entities: Vertex, Edge, Face, Wire, Shell, Solid, CompSolid, and Compound. Only Vertices, Edges, and Faces have direct links to geometric entities. A Vertex is represented by a point. An Edge is represented by a curve and bounded by up to two Vertices. A Wire is a set of adjacent Edges. When the Edges of a Wire form a loop, the Wire is considered to be closed. A Face is represented by a surface and bounded by a closed Wire. A Shell is a set of adjacent Faces. When the Faces of a Shell form a closed volume, the Shell can be used to define a Solid. Solids that share common Faces can be grouped further into CompSolids. Compounds are top-level containers and may contain any other topological entity.

B. Geometric Entities

The topological entities may link to three types of geometric entities, which are Points, Curves, and Surfaces. They represent 0-, 1-, and 2-dimensional geometries respectively. Curves and Surfaces are defined through parameterizable mathematical models. Supported curve types can be categorized as unbounded curves, e.g., lines, parabolas, or hyperbolas, and bounded curves, e.g., Bezier curves, B-spline curves, circles, or ellipses. Offset curves represent a translated version of a given base curve along a certain vector, whereas trimmed curves bound a given base curve by limiting the minimum and maximum parameters of their mathematical model. In case the exact model is unknown, a curve might also be approximated by a polygon on triangulated data. The geometric representation of an Edge might be specified by a 3D curve, or a 2D curve in the parameter space of each surface that the Edge belongs to.
Fig. 3: Taxonomy of topological BREP entities

Fig. 4: Upper taxonomy of geometric entities specified in 3D space

Surfaces rely on unbounded mathematical models, e.g., planes, cones, or cylindrical surfaces, and bounded models, e.g., Bezier surfaces, B-spline surfaces, spheres, or toruses. Surfaces can also be defined as linearly extruded curves. An offset surface translates a base surface along a given vector, and a revoluted surface is created by rotating a given base curve around a given direction vector. Again, if the exact mathematical model of a surface is unknown, an approximation based on triangulation might be specified.

IV. CAD ONTOLOGY

The proposed semantic description language for representing CAD models has been implemented using the Web Ontology Language (OWL). OWL is an open web standard developed by the World Wide Web Consortium (W3C) and is based on a formal specification, i.e., a description logic, which facilitates logical inference. OWL distinguishes between classes, individuals (instances of classes), and properties defined for classes or individuals. Classes and properties are arranged in a hierarchical manner and may be derived from multiple super classes and properties respectively. In this section, the semantic BREP representation and the taxonomy of geometric constraints will be explained. OWL listings will be given in Manchester OWL syntax.

A. Boundary Representation of objects

Following the structure of the BREP representation, as explained in Section III, a taxonomy of topological (see Figure 3) and geometric (see Figures 4) entities has been implemented. The topological entities are connected through properties which resemble the relations depicted in Figure 2. The remainder of this section explains details of the representation following illustrative excerpts of an exemplary CAD model. Figure 5 shows the ontological representation of a CAD model loaded in the ontology editor Protege. The selected individual Solid9 is highlighted by a custom rendering plug-in, which we have implemented to provide model inspection directly in Protege.

The solid Solid9 is bounded by shell Shell9.

Shell9 consists of a set of 41 faces that form a closed volume, which corresponds to the highlighted part of the CAD model in Figure 5.

Face97 is one of the faces that are part of Shell9. It is locatedAt a certain position with a certain orientation. This pose is given by TransformationMatrix739. The face is representedBy the geometrical entity CylindricalSurface34 and boundedBy Wire99. The orientation of the face is set as reversed. Setting the orientation to be forward or reversed on the topological level allows for sharing the same geometric entities among multiple topological entities, e.g., when the same surface is part of two solids.

The geometric entity CylindricalSurface34 represents a parameterizable cylindrical surface. The three direction vectors Vector879, Vector880, and Vector881 define the cylindrical surface’s coordinate system at Position469. The cylindrical surface also has a radius.
Having the unbounded \texttt{CylindricalSurface34} as the base geometry for the topological entity \texttt{Face97}, a boundary limit on the surface is required and is given through \texttt{Wire99}. The wire consists of the four edges \texttt{Edge234}, \texttt{Edge201}, \texttt{Edge147}, and \texttt{Edge145}. The firstElement of the wire is explicitly specified. Further information on the connectivity of edges is defined in the edge entities.

Individuum: \texttt{cad:Wire99}

\begin{itemize}
  \item Types: \texttt{cad:Wire}
  \item Facts:
    \begin{itemize}
      \item \texttt{cad:firstElement cad:Edge234 ,}
      \item \texttt{cad:contains cad:Edge234 ,}
      \item \texttt{cad:contains cad:Edge201 ,}
      \item \texttt{cad:contains cad:Edge147 ,}
      \item \texttt{cad:contains cad:Edge145 ,}
    \end{itemize}
\end{itemize}

Looking at \texttt{Edge234} reveals that it is \texttt{representedBy Circle97} and \texttt{boundedBy Vertex98} and \texttt{Vertex99}. Is has exactly one adjacentEdge connected to it. \texttt{Edge234}'s pose is given by \texttt{TransformationMatrix538}. By asserting all contained edges for wires and all adjacent edges for edges, it is possible to share edges between multiple wires. The specification of \texttt{Circle97} is done in a manner analogous to the cylindrical surface.

Individuum: \texttt{cad:Edge234}

\begin{itemize}
  \item Types: \texttt{cad:Edge}
  \item Facts:
    \begin{itemize}
      \item \texttt{cad:representedBy cad:Circle97 ,}
      \item \texttt{cad:boundedBy cad:Vertex98 ,}
      \item \texttt{cad:boundedBy cad:Vertex99 ,}
      \item \texttt{cad:adjacentEdge cad:Edge147 ,}
      \item \texttt{cad:locatedAt TransformationMatrix538}
    \end{itemize}
\end{itemize}

\texttt{Vertex98} is \texttt{representedBy Point52} and has a pose defined by \texttt{TransformationMatrix538}. Setting the orientation to be forward or reversed on the vertex level allows sharing the same point among multiple vertices, e.g., when two edges share a common end point.

Individuum: \texttt{cad:Vertex98}

\begin{itemize}
  \item Types: \texttt{cad:Vertex}
  \item Facts:
    \begin{itemize}
      \item \texttt{cad:locatedAt TransformationMatrix52 ,}
      \item \texttt{cad:representedBy Point52 ,}
      \item \texttt{cad:isReversed false}
    \end{itemize}
\end{itemize}

\texttt{Point52} has asserted \texttt{x}, \texttt{y}, and \texttt{z} coordinates. Vectors are defined in the same way.

Individuum: \texttt{cad:Point52}

\begin{itemize}
  \item Types: \texttt{cad:Point}
  \item Facts:
    \begin{itemize}
      \item \texttt{cad:x "67.5"^^xsd:double ,}
      \item \texttt{cad:y "-6.162975822039715E-32"^^xsd:double}
      \item \texttt{cad:z "47.0"^^xsd:double ,}
    \end{itemize}
\end{itemize}

Apart from the explained geometric entities \texttt{CylindricalSurface}, \texttt{Circle}, and \texttt{Point}, there are many more supported types. In Section \texttt{III} most of them have been mentioned. Listing all of their properties is out of the scope of this paper, but they can be investigated in the related ontology file, which is available online\footnote{https://github.com/OntoBREP/ontobrep}.

\section*{B. Geometric constraints}

Given the rich semantic description of CAD models, as explained in Section \texttt{IV-A}, it is possible to refer to arbitrary parts of a CAD model and to link it with additional information. In this paper we want to focus on how to add geometric constraints. Figure \texttt{6} shows the upper taxonomy of the geometric constraints that have been designed. Those constraints are meant to be specified between points, curves, and surfaces of objects. We explain the constraints formulation using a couple of examples, which provide a fair understanding of our approach. A complete formal description can be found in the aforementioned ontology file. In our representation a geometric constraint refers to two
geometric entities: a base entity with a defined pose and a constrained entity whose pose depends on the fixed entity and the constraint itself. The null-space of a geometric constraint describes a set of relative transformations between the involved geometries that satisfy the constraint.

1) **CylinderCylinderConcentricConstraint**: This constraint can be defined between two cylindrical surfaces. It constrains the symmetry axis of the constrainedGeometry to align with the symmetry axis of the baseGeometry. The resulting null-space of this constraint is specified by a Line, which is identical to the symmetry axis of the baseGeometry.

   Class: CylinderCylinderConcentricConstraint  
   SubClassOf: ConcentricityConstraint,  
               baseGeometry exactly 1 CylindricalSurface,  
               constrainedGeometry exactly 1 CylindricalSurface,  
               hasNullSpace exactly 1 Line

2) **PlanePlaneCoincidenceConstraint**: This constraint can be defined between two planar surfaces. It constrains the distance of the constrainedGeometry from the baseGeometry and also the rotation of the constrainedGeometry along the two directions orthogonal to the normal vector of the baseGeometry. The resulting null-space of this constraint is specified by a Plane, which is identical to the baseGeometry.

   Class: PlanePlaneCoincidenceConstraint  
   SubClassOf: CoincidenceConstraint,  
               baseGeometry exactly 1 Plane,  
               constrainedGeometry exactly 1 Plane,  
               hasNullSpace exactly 1 Plane

Practical applications of the semantic BREP representation and the geometric constraints are given in Section V.

V. APPLICATIONS

A. **Object Recognition**

Object recognition and pose estimation using CAD models is a classical research area in the field of computer vision. There are several approaches that rely on object detection models composed of primitive shapes. We extended our previous work [10] to directly obtain primitive shapes from object models that are described using the BREP ontology presented in this paper. This improves the performance by eliminating errors due to incorrect detections of shape primitives from point cloud models.

In [11], we presented an approach for detecting symmetrical objects using geometric constraints. This was extended to use semantic BREP models of primitive shapes and geometric constraints between them (see Figure 7). The resulting under-specified object poses, controlled-spaces, and null-spaces can now be represented as geometric entities themselves.

B. **Task specification based on geometric constraints**

Intuitive interfaces for robot programming [12] are an important target application of this work. Semantically rich descriptions of CAD models allow users to refer not only to manipulation objects, but to the geometric entities that they are comprised of. Robot tasks can then be defined using interrelation constraints between the relevant geometric entities, e.g., to specify an assembly pose [13]. Figure 8 shows a dialog of a graphical user interface used to visualize object models and to define a set of geometric constraints between them. Figure 8 presents a constraint-based definition of an assembly task involving these workpieces, which is parameterized by constraints between the geometrical entities.
that compose the workpieces. Using the object poses obtained from the vision system (see Section V-A), the geometric constraints are solved to generate target poses for task execution.

C. Constraint-based robot control

In this application, the robot task specifications that are created using geometric constraints (see Section V-B) are combined with semantic descriptions of the robotic workcell. A semantic workcell description contains the workcell layout, sensors, tools, robots and their kinematic structure, etc. The task’s geometric constraints are solved to generate corresponding constraints on the robot’s pose. The resulting robot-dependent null-space of the solved geometric constraints can be linked to the task instance. The constrained robot poses are then executed on the low-level robot controller, which may exploit the information about null-spaces to optimize the robot’s motion without violating the given constraints. Details of this application can be found in [14].

VI. QUANTITATIVE EVALUATION

In order to mitigate the required efforts for describing a CAD model using our semantic description language, we implemented a software component for converting STEP and IGES files to the proposed ontological representation. As most widely used CAD software support these interchange formats, our converter enables us to continue relying on these mature and highly specialised software tools for designing the CAD models. The converter has been implemented in Java and uses the OWL API for creating OWL axioms and a Java Native Interface (JNI) to the community edition of Open CASCADE for parsing STEP and IGES files.

We investigated the quantitative aspects of importing an existing STEP or IGES model and creating corresponding ontological entities. This includes the time required to parse the given STEP or IGES file, to initialize the OWL API’s manager and factory classes, and to translate the in-memory data structure of the source model to OWL individuals, classes, and object property and data property assertions. The resulting in-memory OWL representation was serialized again to a file using different formats, i.e., the Manchester OWL syntax and the RDF/XML syntax. Figure 9 shows the models used in this evaluation. In Table I, file sizes of the different CAD formats, including the two OWL-based serializations, are compared. As OWL representations feature unique text identifiers for all geometric and topological entities, they are bigger in size. Due to the redundancy in identifiers that share common prefixes, the discrepancy in size, e.g., between STEP and the RDF/XML syntax shrinks from a factor of 7 to 1.7 when both files are compressed. Table II lists the types and counts of topological entities for the models used in this evaluation and also the types and counts of OWL axioms that were created to describe the same objects.

![Fig. 9: Models used for the quantitative evaluation](image)

<table>
<thead>
<tr>
<th>Model</th>
<th>Converting STEP</th>
<th>Loading OWL in Sesame</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time in ms</td>
<td>time in ms</td>
</tr>
<tr>
<td>CUBE</td>
<td>365</td>
<td>25</td>
</tr>
<tr>
<td>FRAME</td>
<td>805</td>
<td>343</td>
</tr>
<tr>
<td>ROTOR</td>
<td>1018</td>
<td>704</td>
</tr>
</tbody>
</table>

There are various ways to store and query the created OWL-based CAD models. One common option is to use the Sesame triple store. In a second experiment, the OWL representations of models have been loaded into Sesame in order to benchmark loading time. Required times for importing STEP models and loading the OWL representation in Sesame are listed in Table III. All experiments have been carried out on an Intel Xeon quad-core CPU running at 2.80 GHz with 12 GB of RAM and a mechanical harddisk. Listed times have been calculated as an average score based on ten trials each, with only minor deviations among single results, i.e. less then 10%.

VII. CONCLUSION

In this paper, we present a semantic description language for CAD models, which is based on a boundary representation of points, curves, surfaces, and volumes. This language also offers features to specify geometric constraints between arbitrary parts of CAD models. They may not only refer to the coordinate frame of an object, but also directly to its vertices, edges, or faces. The benefits of this approach have been showcased through a set of applications, that range from object recognition to intuitive parameterization of task descriptions and constraint-based robot control. A quantitative evaluation has been carried out, in order to assess the complexity of representing complete CAD models in a semantic way. The results of the evaluation show that the approach is not only feasible but practically useful.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 287787 in the project SMErobotics.

[http://rdf4j.org/](http://rdf4j.org/)
TABLE I: Comparison of file sizes for given CAD models represented in different formats. OWL-based formats are more verbose than the industry standards STEP and IGES, but compressing the models drastically decreases the discrepancy to an acceptable level.

<table>
<thead>
<tr>
<th>Model</th>
<th>BREP</th>
<th>STEP</th>
<th>IGES</th>
<th>OWL Manchester</th>
<th>OWL RDF/XML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>plain</td>
<td>zipped</td>
<td>plain</td>
<td>zipped</td>
<td>plain</td>
</tr>
<tr>
<td>CUBE</td>
<td>4.0</td>
<td>0.9</td>
<td>15.9</td>
<td>2.9</td>
<td>21.2</td>
</tr>
<tr>
<td>FRAME</td>
<td>143.5</td>
<td>15.9</td>
<td>353.7</td>
<td>38.9</td>
<td>444.7</td>
</tr>
<tr>
<td>Rotor</td>
<td>170.8</td>
<td>20.0</td>
<td>650.8</td>
<td>63.3</td>
<td>896.0</td>
</tr>
</tbody>
</table>

TABLE II: Number of topological BREP entities for the given CAD models and how they translate into the proposed OWL representation.

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of topological BREP entities</th>
<th>Number of OWL axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ve</td>
<td>Ed</td>
</tr>
<tr>
<td>CUBE</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>FRAME</td>
<td>152</td>
<td>228</td>
</tr>
<tr>
<td>Rotor</td>
<td>270</td>
<td>405</td>
</tr>
</tbody>
</table>

REFERENCES