Challenges of CAD conversion to 3D development environments with respect to kinematic dependencies

Philipp Braun Martin Sliwinski Johannes Hinckeldeyn Jochen Kreutzfeldt

Institute for Technical Logistics, Hamburg University of Technology, Germany, {philipp.braun, martin.sliwinski, johannes.hinckeldeyn, jochen.kreutzfeldt}@tuhh.de

Abstract

Due to the increasing utilization of 3D development environments for industrial use cases by emerging topics like virtual reality, new challenges in the conversion of computer-aided-design (CAD) models to 3D models have become visible. Particularly, the conversion of the kinematic constraints turns out to be complex and requires extensive manual efforts. This paper discusses these challenges with a focus on the problems in the conversion of kinematic constraints. Further, a new approach is proposed and validated regarding the automated conversion of the kinematic constraints of an exemplary CAD model. Thereby, this work contributes to an accelerated and simplified development process of industrial applications within 3D development environments.

Keywords: computer aided design (CAD), 3D, kinematic dependencies, CAD conversion, constraints

1 Introduction

CAD systems are indispensable in industrial product development processes and represent the state of the art in the area of product design software. However, product development tasks often go far beyond pure product design and require additional software solutions for simulation, visualization or other tasks. While CAD systems are the preferred tools for product development, 3D development environments[1] (3DDE) are used for visualization in high-quality renderings, videos or virtual reality applications (Davila Delgado et al., 2020). Because of the limited compatibility between CAD systems and 3DDE, file conversions are usually required and lead to a loss of information (e.g. parameters, materials, kinematic constraints) (Raposo et al., 2006). Manual restoration of lost information can be time consuming and must be repeated with every adjustment of the design. Even existing CAD conversion tools are not able to fully convert kinematic constraints (Stelzer et al., 2014). Other scientific publications identified this problem as well but could not present a general applicable solution (Whyte et al., 2000). For that reason, this paper will present existing challenges regarding the conversion of kinematic constraints followed by an approach for an automatic conversion concept.

This paper is structured as follows: In section 2, the state of the art highlights current challenges regarding the conversion of CAD models to 3DDE compatible files. Further, problems of current approaches to overcome the existing challenges in the file conversion will be discussed. In section 3, an approach for a concept which addresses the identified issues is presented. A validation and discussion of the presented approach is given in section 4. This work closes with a conclusion in section 5.

2 State of the art

The following section focuses on the state of art and identified challenges in the conversion workflows from CAD models to files supported by 3DDE. First, currently available file formats will be discussed. Second, challenges regarding the conversion of kinematic constraints are presented. Finally, current approaches to overcome the conversion issues will be presented.

2.1 File formats for 3D-data exchange

Challenges in the conversion of 3D models from CAD systems to 3DDE arise, because most software systems use their own native file formats. In addition, the most software systems offer only limited capabilities to export or import data to other software. In order to overcome these issues, neutral exchange formats have been developed, which can generally be read by any common CAD- or 3D-system. However, the converted content often does not go beyond pure geometry (Dimitrov and Valchkova, 2011). The possibility to store additional information like materials, parameters, kinematic constraints etc. can vary from format to format. Furthermore, only a limited number of CAD systems offer the possibility to export these data.

[1] Popular examples are 3DsMax, Cinema4D, Blender or Unity.

One major difference between CAD and 3DDE models is the representation of surfaces. CAD-Models are described by non-uniform-rational-B-spline (NURBS) surfaces which are an exact mathematical representation of the geometry. The most commonly used CAD-exchange formats are STEP (.stp) and IGES (.igs) which are both neutral formats. In contrast, 3DDE use a discretized, polygon-based grid to generate surfaces as current 3DDE cannot render NURBS surfaces in real-time (Kumar and Manocha 1996; Sikorska 2008). When moving from CAD to a 3D application, polygonization (tessellation) of the CAD model is generally required. As a result, the used file formats differ fundamentally. In general, the tessellation process only translates surfaces belonging to the 3D model. Construction geometry like points, axes or planes are not considered and consequently lost during conversion. The most important neutral 3D-exchange formats are Wavefront OBJ (.obj), COLLADA (.dae), STL (.stl) and VRML (.wrl). Further, the proprietary formats Filmbox (.fbx) and 3DS (.3ds) are provided by Autodesk but are supported by many different 3DDE. An overview of the mentioned file formats is presented in Table 1.

Table 1: Overview of the most important CAD- and 3D-exchange file formats.

File format	Suffix	Type	Context
STEP	.stp	neutral	CAD
IGES	.igs	neutral	CAD
Filmbox	.fbx	proprietary	3D
3DS	.3ds	proprietary	3D
Wavefront OBJ	.obj	neutral[2]	3D
COLLADA	.dae	neutral	3D
STL	.stl	neutral	3D
VRML	.wrl	neutral	3D

2.2 Kinematic constraints in CAD and 3DDE

Kinematics are commonly used in the product development to test and illustrate motions of future products within the design process. The correct definition of kinematic constraints is further required for dynamic simulation (Engelson, 2000) or functional tests in a virtual environment. The creation of such kinematics in CAD is usually done by assigning geometric relationships and thus deliberately restricting degrees of freedom (DOF). Different geometries and relations can be used here. The most frequently used geometries are primitives like points, axes and planes, which can be defined as congruent, parallel, angular, or with a distance from each other. The totality of all geometrical constraints of a CAD model component results in its respective DOF of movement, which defines the kinematics of the assembly.

In contrast, 3DDE offers special features, commonly called kinematic joints, to simulate special kinematic behaviors. Due to the mesh-based surface representation in 3DDE, these joints are normally defined by assigning coordinates and vectors and manually restricting the respective degrees of freedom. Thus, the definition of kinematic constraints is strongly differing from the geometry-based constraint definition in CAD, which makes the conversion of constraints from CAD to 3DDE challenging.

Summarizing, the two main issues of file conversion from CAD to 3DDE result from the existence of a variety of different file formats and the loss of data during file conversion which exceed pure geometry. Especially kinematic constraints are lost, as none of the prevailing 3D-supported file formats can fully support the preservation of kinematic constraints from CAD models. There are different approaches to overcome these issues which are presented in the following subsection.

2.3 Existing approaches for the automated conversion of CAD models

To overcome the issues introduced in subsection 2.2, several approaches have been developed with a focus on the conversion of CAD files to 3DDE. These approaches can be divided into three categories based on the types of data which are transferred from the CAD software to 3DDE:

1. Manual conversion and recreation in 3DDE
2. Automatic conversion of geometries and user-defined meta-data
3. Automatic conversion of geometries and kinematic constraints

In the manual approach, 3D computer graphics software is used to convert CAD models to 3D supported file formats including the geometries to represent a certain model. After the conversion to a 3D-supported file format, the files can be imported into a 3DDE. Since all information regarding the kinematic dependencies are lost, a high manual effort is required to remodel these information to enable realistic kinematic simulations of interactions and motions (Wolfartsberger, 2019). This approach is currently the common workflow in the development of applications which include representations of CAD files in 3DDE.

[2] The ASCII variant of the Wavefront OBJ format is neutral while the binary variant is proprietary.

Within this workflow, the remodeling needs to be performed with every update of the CAD model and the effort reoccurs evenly. As a consequence, the second and third category focus on the development of automated solutions for the conversion of CAD files.

Approaches in category 2 focus on the conversion of CAD parts and attached meta-data from several subsystems like product data management systems. Compared to previously described approaches, these ones generate two datasets within the conversion process (Gebert et al., 2017). The first dataset includes the geometries of the CAD model in a common file format. The second dataset stores information which are not related to geometry. This includes information like the hierarchy structures of assemblies in CAD models and pre-defined animations of kinematic motion sequences. A further possibility in these approaches is the integration of process knowledge like production information from additional subsystems (Górski, 2017). Compared to the first category, the second category of approaches can support the automatic integration of pre-defined animations, though the integration of kinematic constraints is still not supported. Thereby, motion sequences can be visualized but a realistic interaction with a certain 3D model still needs to be modeled manually in a 3DDE.

The third category focuses on the conversion of kinematic constraints from CAD models to 3D supported file formats to enable full realistic simulations and animations of kinematic motion sequences in real-time based on user-defined input values. For this reason, the approach of Lorenz et al. (2016) also starts with the automatic conversion of CAD models and pre-defined animations. To enable interactive kinematic motion sequences, the functionalities of mechanisms in certain CAD models has to be reproduced in the 3DDE. Therefore, the calculation of kinematic constraints is performed in a CAD system which is directly connected to the 3DDE via the programming interface. Using this connection, all manipulations of positions or orientations of parts in the 3DDE are streamed to the CAD system. Based on this transmitted information, the positions and orientations of all connected parts within the CAD model are recalculated. These recalculated data are streamed back to the 3DDE which enables a full interactive representation of CAD models in 3DDE (Andaluz et al., 2016). However, by connecting a 3DDE to a CAD system the real-time streaming between these tools is indispensable. Using this approach, a stand-alone solution for the conversion of kinematic constraints in a 3DDE is difficult to implement. Further, a reliable streaming highly dependents on the stability and performance of the connection.

In summary, none of the mentioned categories of approaches supports the full implementation of all constraints from CAD models in 3DDE as a stand-alone application. To implement a stand-alone application in a 3DDE, an automated remodeling of all kinematic constraints and DOF of the parts in certain CAD models is required. This remodeling must be implemented exclusively with tools provided by the 3DDE.

3 Approach for an automated conversion of kinematic constraints

This section presents an approach of a conversion concept from CAD systems to 3DDE with respect to kinematic constraints. The aim is to show a general and conceptual way of translating kinematic constraints from CAD to 3DDE not to present a fully functional software tool. The first subsection discusses prerequisites like the used file format and the basic conversion principle. Subsequently, the solution for the assembly hierarchy conversion and the conversion of constraints is presented. The origin of this work is based on an industrial use case in which a developed product had to be tested in a virtual environment. The design was created in Autodesk Inventor, while the visualization in 3D takes place in the 3DDE Unity. The following implementation of the conceptual approach applies basically to the conversion from Inventor to Unity, but can be adapted to further CAD systems.

3.1 Prerequisites for the conversion concept

The approach presented in this work uses the programming interfaces provided by Autodesk Inventor and Unity. The programming language used for Autodesk Inventor is VBA while Unity uses C#. Since the most CAD systems and 3DDE offer programming interfaces, the proposed solution can be adapted to other software combinations.

As shown in chapter 2.2, 3DDE like Unity only support polygon-based file formats for the 3D-data-exchange. For this work, we chose the .obj format which was selected primarily because of its easy processing and manipulation capabilities. In contrast to other file formats like .fbx and .dae, the .obj format can be natively exported from Autodesk Inventor. The disadvantage is the limited capability to store additional information. However, this work is not intended to present a final solution for the constraint problem, but to show a general way to transform kinematic constraints from CAD to 3DDE.

Due to the loss of hierarchy information, a manipulation of the .obj file is necessary to restore the local coordinate systems of the part, which is described more detailed in the following section. In addition to the .obj file, a text-based extension file is generated, which is used for storing missing information. This text file is automatically created within the CAD conversion and contains necessary information for every part of the assembly to restore the original structure. An illustration of the basic conversion concept is presented in figure 1. Two main steps can be differentiated here: the

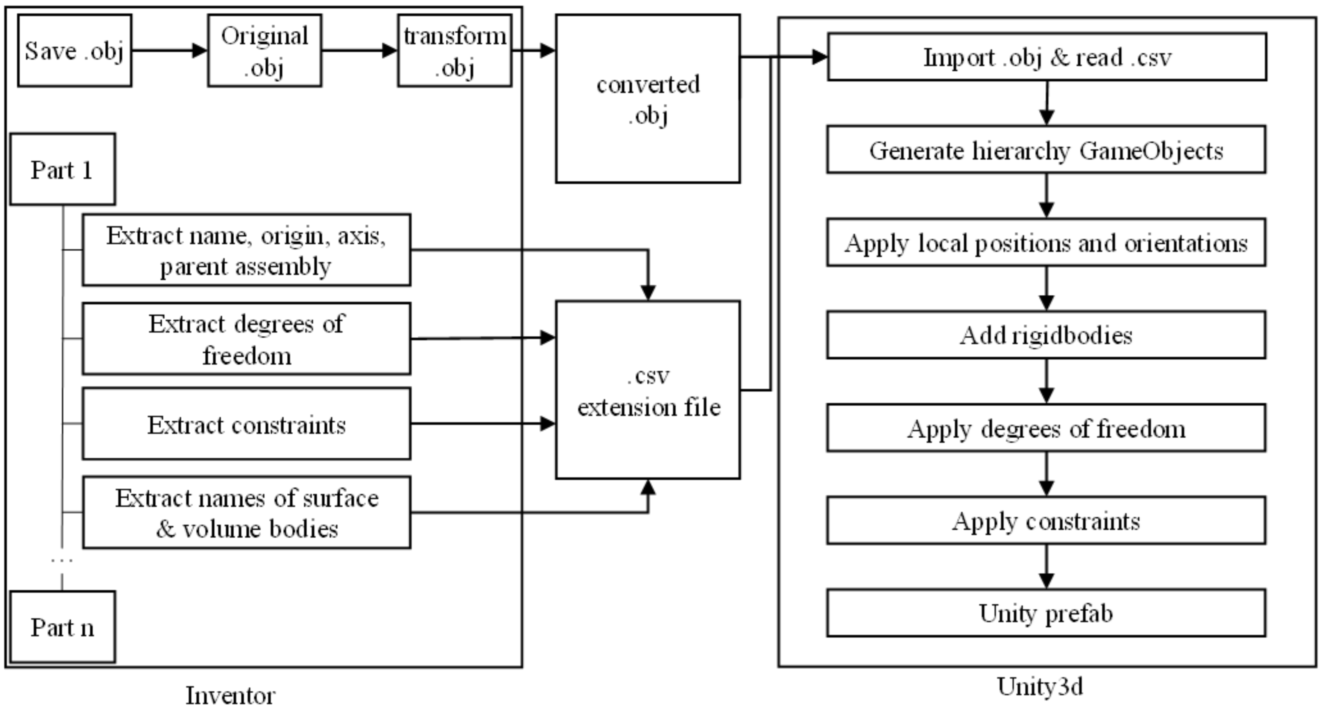

Figure 1: Software architecture of the proposed conversion approach.

conversion of the assembly hierarchy including all local coordinate systems and the conversion of the entire movement kinematics of the assembly.

3.2 Conversion of assembly hierarchy

A notable disadvantage of the .obj file format is the missing capability of preserving the assembly hierarchy structures. Thereby, the conversion leads to the consolidation of all volume and surface bodies on different hierarchy levels of an assembly resulting into one unstructured collection of bodies (see figure 2). Furthermore, the .obj format does not support the use of different coordinate systems, which means that all bodies are positioned with respect to a common origin point during the conversion.

Further, the volume and surface bodies[3] of the entire assembly are converted into a mesh-based geometry and written to the .obj file one-by-one during the conversion. If multi-body parts were used in the original design, this is no longer recognizable after the conversion and the individual bodies can no longer be assigned to the associated parts.

To correctly restore the hierarchy structure of the CAD model after the conversion, knowledge of the local coordinate systems of each part in relation to the global coordinate system is essential. The assignment of parts and subassemblies to the various hierarchical levels must also be known. For this reason, each part or sub-assembly of the main-assembly used in Autodesk Inventor is investigated in a loop and the name, type, superior assembly as well as associated volume and surface bodies are written to the extension file. The name corresponds to the component part-name that was assigned in Inventor. The type indicates whether it is a part or an assembly and the higher-level assembly is required to restore the hierarchy levels.

In this approach, the lost parts and assemblies of the CAD model are represented by newly instantiated Unity GameObjects[4]. After instantiation, the volume and surface bodies from the CAD model are attached to the instantiated GameObjects representing their superordinate parts. After attaching volume and surface

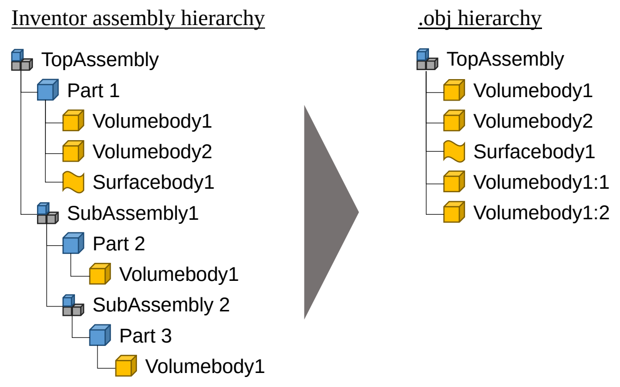

Figure 2: Assembly hierarchy structure in CAD models and .obj files after conversion.

[3] A single CAD-part can have multiple volume and surface bodies. After the conversion to .obj and reimport the bodies appear as independent parts.

[4] GameObjects are the basic class of entities within a Unity virtual environment.

bodies to their respective GameObjects, the hierarchy structure of the CAD model has to be restored.

Due to the loss of the local coordinate systems during the conversion, the origin point of the bodies is no longer coincident with the original coordinate system origin. All vertices describing the body’s mesh inside the .obj file are given relative to the global coordinate system. To correct this, a coordinate transformation of all vertices is required. This transformation takes place directly in the original .obj file where all bodies are moved to the global origin. To transform the coordinates, the individual transformation matrix of each part is needed. The values of this matrix can be evaluated in Inventor and are written to the extension file. After the transformation all vertices of a body are given in relation to the local coordinate system of the associated part. When importing this file to Unity, all local coordinate systems of the bodies are coincident with the global coordinate system. To relocate the individual bodies, the extension file contains the coordinates of the origin points and the direction vectors of the X-, Y- and Z-axis for each component in relation to the global coordinate system.

Using this information, the parts and assemblies can be relocated with correct coordinate systems in Unity. This is performed by a Unity extension script which reads the information regarding the local position and orientation of all parts from the extension file. In the next step, local positions and orientations are assigned to all GameObjects representing the parts and assemblies of the CAD model. Additionally, the assigned position and orientation values must be translated from the right-handed coordinate system of the CAD system to the left-handed coordinate system of Unity.

3.3 Conversion of kinematic dependencies

In the following subsection, the algorithm for converting kinematic dependencies, which is the core of this work, will be described. The basic concept of the algorithm is valid across different software tools and file formats. The implementation can be considered as a proof of concept.

In common CAD systems, the connection between multiple rigid bodies within an assembly design is realized by assigning kinematic constraints to pairs of rigid bodies. These constraints are based on primitive geometries like points, lines or planes, which can be defined as coincident, angular and distant (Haller et al., 2012). Most of the 3DDE offer only a limited functionality compared to CAD systems for assigning kinematic constraints. Instead, constraints are simulated using a physics engine in combination with special attributes assigned to the individual objects. In this work, two essential attributes of Unity (hinge joint and configurable joint) are used to remodel kinematic constraints. The hinge joint connects two bodies to form a joint, restricting all translational and rotational DOF excluding the rotation around one joint axis. The joint axis is defined by an origin point and a direction vector. The configurable joint is a flexible component, which enables the creation of kinematic joints, but also the restriction and unlocking of certain DOF. This is performed by assigning an individual coordinate system to the specific configurable joint. Subsequently, all six DOF can be restricted or unlocked individually.

The algorithm for automatically converting the kinematic dependencies consists of four steps. The first and second steps must be performed in a CAD system and the last steps in a 3DDE:

1. Evaluation of DOF (CAD system)
2. Identification of rotational kinematic constraints (CAD system)
3. Reinstantiation of DOF (3DDE)
4. Setup of rotational kinematic constraints (3DDE)

In the first step, the DOF of all parts and assemblies within the CAD model are evaluated. Thus, the number of translational and rotational DOF as well as the respective coordinate directions needs to be determined. Any unrestricted DOF in combination with the corresponding direction vector is then written to the .csv file.

Subsequently, rotational constraints between different parts or assemblies within the CAD model are identified in step 2. As already described in the previous section, the connection between two parts is established by geometric relationships between two parts or assemblies within the CAD model. At the present early development stage, it was not possible to cover all possible combinations. Only constraints based on the congruence of two axes as a rotational axis are considered. Corresponding relationships are written into the extension file, stating the connected part or assembly and the axis direction.

The third step uses the input of the first step within the extension script for Unity to remodel the DOF of all parts and assemblies within the CAD model. Therefore, configurable joints are attached, corresponding to the information from the extension file, to restrict or unlock the DOF of the GameObjects representing the parts and assemblies of the CAD model.

Finally, components of the type hinge joint are applied to remodel the rotational constraints between different parts or assemblies of the CAD model within Unity. Therefore, the origin of the rotational axis is assigned to the hinge joint in the local coordinate system of the corresponding GameObject. Afterwards, the rotational axis is defined by the direction vector.

4 Validation and Discussion

The validation of the proposed approach for the automated conversion of kinematic dependencies from CAD models in Autodesk Inventor to the 3DDE Unity is performed using an exemplary CAD model of a mechanical assembly (Figure 3). Within the assembly, a rotational motion from a hand lever (A) is translated via a shaft to an eccentric panel (B). The bar (C) translates rotational motion to the slider (D) which has a single and linear direction of motion. The conversion of the CAD model was performed by the internal .obj converter of Autodesk Inventor and the developed extension script. To implement the DOF each GameObject representing a part or an assembly has two attached configurable joints. One is accountable for defining translational DOF and one for the rotational DOF. The rotation axis of A is modeled by a hinge joint as well as the centers of rotation on the top and bottom of C. The linear motion of D is modeled by unlocking the corresponding DOF within the configurable joint of the GameObject. The developed extension scripts are attached as a GitLab repository of the Hamburg University of Technology (Braun et al., 2020).

The presented simple and exemplary assembly was converted using the proposed approach including the rotational kinematic dependencies. This initially provides a proof of concept for the conversion concept with respect to the presented assembly. Currently, the proposed concept only considers constraints based on the congruence between two axes because these are primarily used for the generation of rotational DOF. Constraints based on further geometric relationships cannot be considered, yet. Translational DOF are not affected, since these are evaluated using the DOF analysis integrated in Invertor and do not require an analysis of the existing constraints.

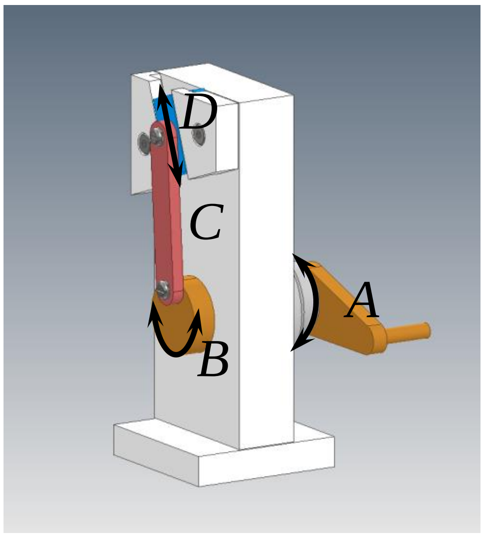

Figure 3: Exemplary CAD assembly for the validation of the proposed approach.

Future research activities should focus on algorithmic evaluation of the remaining combinations of constraints. Therefore, a more complex analysis of DOF is required (Lee-St.John, 2012). Especially, a possibility to find parts that are connected to each other as well as the type of their connection (translational / rotational) based on the existing DOF and the existing constraints of an assembly is of special interest. Additionally, subsequent research activities should focus on the used file formats. A detailed analysis is necessary to verify whether file formats like .fbx or .dae offer extended possibilities to transfer kinematic constraints of CAD models.

Despite the existing limitations, the presented approach can support the conversion of kinematic constraints in the application development process within 3DDE. This can lead to a faster and easier development of 3D applications for design reviews. Besides the easy visualization of kinematic dependencies, the automatic conversion supports the representation of results from simulations (Wolfartsberger et al., 2017).

5 Conclusion

This paper summarizes challenges in the conversion of CAD models to files supported by 3DDE. The challenges originate in the concurrent usage of CAD systems for mechanical designs and 3D development engines for high quality visualizations and application developments. Current file formats only support the conversion of geometries, but the conversion of kinematic constraints from CAD models to 3D files is not feasible. Therefore, different approaches from time-consuming manual restoration of kinematic constraints in 3DDE to the automated synchronization of CAD systems with 3DDE has been developed.

The conceptual approach for the conversion of kinematic dependencies of CAD models contributes to the standalone representation of CAD models in 3DDE. The current implementation enables the automated conversion of selected rotational kinematic constraints from Autodesk Inventor to Unity. Within the elaborated workflow, a file is generated to represent the geometries of the CAD model. An additional extension file contains missing information regarding the hierarchy structure, local coordinate systems, DOF and kinematic constraints. Using this information, the initial kinematic dependencies from the CAD model can be restored in Unity. In future research activities, it is planned to apply the proposed approach for a wider range of kinematic constraints.

References

V. H. Andaluz, F. A. Chicaiza, C. Gallardo, W. X. Quevedo, J. Varela, J. S. Sánchez, and O. Arteaga. Unity3D-MatLab Simulator in Real Time for Robotics Applications. In Lucio Tommaso de Paolis and Antonio Mongelli, editors, *Augmented Reality, Virtual Reality, and Computer*

Graphics, pages 246–263. Springer International Publishing, 2016. ISBN 978-3-319-40620-6. doi: 10.1007/978-3-319-40621-3_19.

P. Braun, M. Sliwinski, J. Hinckeldeyn, and J. Kreutzfeldt. *CAD to 3D Converter*. Hamburg University of Technology. Hamburg, 2020. Available online at https://collaborating.tuhh.de/w-6/forschung/depot-projekt/cad-to-3d-converter, checked on 07 July, 2020.

J. M. Davila Delgado, L. Oyedele, P. Demian, and T. Beach. A research agenda for augmented and virtual reality in architecture, engineering and construction. *Advanced Engineering Informatics*, 45 (3):101–122, 2020. doi: 10.1016/j.aei.2020.101122.

L. Dimitrov and F. Valchkova. Problems with 3D data exchange between CAD systems using neutal formats. *Proceedings in Manufacturing Systems*, 6 (3):127–130, 2011.

V. Engelson. *Kinematic information formats. Standards applicable for mechanical model translation from CAD to Modelica*. MathCore AB. Linköping, Sweden, 2000.

M. Gebert, W. Steger, R. Stelzer, and K. Bertelmann. Meta-model for VR-based design reviews. In *Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 4: Design Methods and Tools, International Conference on Engineering Design, 21-25 August, 2017, Vancouver, Canada*, pages 337–346, 2017.

F. Górski. Building Virtual Reality Applications for Engineering with Knowledge-Based Approach. *Management and Production Engineering Review*, 8 (4):64–73, 2017. doi: 10.1515/mper-2017-0037.

K. Haller, A. Lee-St.John, M. Sitharam, I. Streinu, and N. White. Body-and-cad geometric constraint systems. *Computational Geometry*, 45 (8):385–405, 2012. doi: 10.1016/j.comgeo.2010.06.003.

A. Lee-St.John. Kinematic Joint Recognition in CAD Constraint Systems. In *Proceedings of 24th Canadian Conference on Computational Geometry, Canadian Conference on Computational Geometry, 8-10 August, 2012, Charlottetown, Canada*, pages 173–178, 2012.

M. Lorenz, M. Spranger, T. Riedel, F. Pürzel, V. Wittstock, and P. Klimant. CAD to VR – A Methodology for the Automated Conversion of Kinematic CAD Models to Virtual Reality. *Procedia CIRP*, 41 (3):358–363, 2016. doi: 10.1016/j.procir.2015.12.115.

A. Raposo, E. T. L. Corseuil, G. N. Wagner, I. H. F. dos Santos, and M. Gattass. Towards the use of cad models in VR applications. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications, VRCIA06: ACM International Conference on Virtual Reality Continuum and Its Applications 2006, 14-17 June, 2006, Hong Kong, China*, pages 67–74, 2006. doi: 10.1145/1128923.1128935.

R. Stelzer, E. Steindecker, S. Arndt, and W. Steger. Expanding VRPN to Tasks in Virtual Engineering. In *Volume 1B: 34th Computers and Information in Engineering Conference, ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, 17-20 August, 2014, Buffalo, USA*, pages 1–8, 2014. doi: 10.1115/DETC2014-34277.

J. Whyte, N. Bouchlaghem, A. Thorpe, and R. McCaffer. From CAD to virtual reality: modelling approaches, data exchange and interactive 3D building design tools. *Automation in Construction*, 10 (1):43–55, 2000. doi: 10.1016/S0926-5805(99)00012-6.

J. Wolfartsberger. Analyzing the potential of Virtual Reality for engineering design review. *Automation in Construction*, 104:27–37, 2019. doi: 10.1016/j.autcon.2019.03.018.

J. Wolfartsberger, J. Zenisek, C. Sievi, and M. Silmbroth. A virtual reality supported 3D environment for engineering design review. In *Proceedings of the 2017 23rd International Conference on Virtual Systems and Multimedia (VSMM), International Conference on Virtual Systems and Multimedia, 31 October - 4 November 2017, Dublin, Ireland*, pages 1–8, 2017. doi: 10.1109/VSMM.2017.8346288.