Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2024) 000–000

www.elsevier.com/locate/procedia

10th CIRP Conference on Assembly Technology and Systems

APLAN: open assembly planning framework in FreeCAD

Martijn Cramer[a,b,], Karel Kellens[a,b], Eric Demeester[a,b]

[a] *KU Leuven, Diepenbeek Campus, Dept. Mechanical Engineering, Research unit ACRO, B-3000 Leuven, Belgium*
[b] *Flanders Make @ KU Leuven, B-3001 Heverlee, Belgium*

* Corresponding author. Tel.: +32-11-751-780. *E-mail address:* martijn.cramer@kuleuven.be

Abstract

In the latest decade, the consumer's desire for personalisation has clearly taken off, challenging manufacturers to diversify their product portfolio while maintaining the same (or near) mass-produced prices. As a result, production is nowadays geared towards low volumes with high variety for which adaptive assembly systems are required. Hence, being able to automatically map out the assembly of complex products yields significant strategic advantages over competing manufacturers. Although a wide range of assembly sequencing algorithms have been developed in the past, researchers and companies still experience intellectual and financial thresholds to applying these software packages themselves. Either the algorithms have to be largely re-engineered from scientific literature, or are offered as part of costly commercial CAD software in the form of plugins and add-ons. However, in some cases, the developed source code is made publicly available via online collaboration platforms though in a variety of programming languages and often requiring the cumbersome installation of additional software libraries.

Out of these needs, the proposed APLAN module arose: a software framework allowing developers to share their (dis)assembly planning algorithms in a centralised, uniform manner through the open-source 3D parametric modelling software FreeCAD. As impetus, several in-house developed algorithms for determining topological and geometrical (dis)assembly constraints are included as well as an academic (dis)assembly planner. Previously, such algorithms had to be executed via the terminal, after which the extensive liaison, blocking, and AND/OR graphs were outputted as text or static images. Through APLAN, users are now able to inspect and edit these graphs interactively and dynamically. To conclude, this manuscript finally reports on the benchmarking of the aforementioned connection and obstruction detectors, and their academic and industrial application for human-robot collaboration.

Keywords: mass customisation; assembly sequence generation; liaison graph; blocking graph; AND/OR graph

1. Introduction

In order to stay ahead of the intense competition, companies are nowadays compelled to speed up the pace and scale at which new products are being launched [1]. On top of this, product personalisation is experiencing a surge, forcing manufacturers to expand their offerings and comply with individual customer needs [2]. Due to this ever-evolving market, today's industry is characterised by high-mix, low-volume production demanding frequent changeovers and reconfigurable assembly lines.

Furthermore, products are outdating faster compared to the past. Together with the growing awareness and legislation for the sustainable use of natural resources, the repair, remanufacturing and recycling of end-of-life goods figure high on the agenda. In order to carefully recover precious materials and reusable parts, thoughtful disassembly is key [3].

Unsurprisingly, in both academia and the industry, flexible and configurable (dis)assembly technologies are rapidly gaining attention. Instead of ad hoc or experience-based solutions, automated software tools are being developed to plan for the most optimal (dis)assembly sequence given a variety of objectives. These computer-aided planners all unite in that they exploit the designer's knowledge embedded in a product's 3D models; how this product and assembly information (PAI) is extracted from these models is what sets them apart.

1. External or file-based methods [4] aim to derive the information necessary for assembly planning from neutral CAD file formats, e.g. STEP, IGES and QIF. Due to their standardisation, virtually every contemporary CAD software supports these interoperable data formats. File-based approaches, therefore, do not depend on a single proprietary CAD application. However, during conversion the data present in the system's native file format might not be entirely transferred. This is defined by the international standard of the neutral format and the software

vendor's compliance [5]. Besides, in order to parse machine-readable files like STEP, one often must rely upon dedicated libraries such as STEPcode, OntoSTEP [6, 7] and JSDAI [8], or CAD kernels including OCCT [9, 10] and ACIS.

2. Internal or interface methods [4] try to capture as much of the valuable designer knowledge as possible by directly interacting with the CAD system's application programming interface (API). In this way, thorough access is provided to a wealth of PAI without intermediate conversion steps. However, due to a lack of consensus among CAD software developers, today's interfaces are neither generic nor accessible via the same programming language or library. Hence, internal tools mostly end up being tailored to a particular—often costly and proprietary—CAD package such as Autodesk Inventor [11], Dassault's CATIA [12, 13] and SolidWorks [10, 14], or PTC CREO [15, 16], of which some license advanced API access separately.

Furthermore, it has been noticed that these tools, whether file or interface-based, are rarely made openly available compelling interested users to largely recreate the developed PAI extractors and assembly sequencing algorithms from scientific literature. In some cases, source code is shared via online collaboration platforms, e.g. [13], though scattered across separate code repositories, in different programming languages, and often requiring the installation of extra software packages.

The assembly planning framework outlined in this research aims to address these hurdles. Indeed, by creating a module or add-on to existing CAD software, users can benefit from comprehensive accessibility to the PAI housed within that CAD system, similarly to interface methods but without being restricted by the API. By opting for FreeCAD [17]—a well-established open-source 3D parametric modeller carried by an extensive developers community—similar independence as external methods is provided without resorting to commercial software.

Two recent initiatives sharing similarities with the presented APLAN module are ASPIP [18] and CAASP [19]. Both also aim at providing the industry with a graphical software application for CAD-based assembly sequence generation. However, ASPIP only supports topological constraints. Moreover, these two apps do not appear to be publicly available. The authors expect that due to its open-sourceness, APLAN will contribute to future researchers sharing their assembly planners in a centralised, ready-to-use way without users needing to install additional libraries or delve into the source code themselves.

The remainder of this paper is organised as follows. Section 2 discusses the building blocks that make up the presented framework. To curb the combinatorial complexity related to assembly planning, it is essential to set constraints. Therefore, Section 3 elaborates on the solvers available through APLAN to automatically determine the assembly's topological and geometrical constraints together with two in-house developed refinement methods for accelerating this search. Furthermore, this section reports on the experimental validation of their performance on six academic and industrial products. Section 4 subsequently analyses the obtained results, and reflects on past applications for which the APLAN workbench proved to be instrumental. Finally, Section 5 summarises the presented work and points to future research directions.

2. Framework architecture

Similar to commercial design software, FreeCAD is centred around the concept of workbenches: a collection of tools developed to serve one specific task often gathered under the same tab, ribbon or menu of the graphical user interface (GUI). FreeCAD's 21st release ships with 20 built-in workbenches and over 100 additional ones, thanks to its solid community.

The set of commands available via the proposed APLAN workbench[1] focuses on assembly planning. Since a substantial portion of FreeCAD's user base is already familiar with the workflow of the FEM (finite element method) workbench, it has been decided to structure APLAN the same way, using the same concepts (e.g., containers, solvers, constraints) as this would flatten any learning curve and ease its adoption.

2.1. Analysis container

First and foremost, the user is required to create an analysis container. As its name suggests, this container functions as a data structure for collecting every object related to the assembly planning process. In addition, it makes sure that the results outputted by the constraint detectors and planners for a particular assembly product are stored in the same designated working directory. However, there is no limitation on the number of assembly containers one can create for a single assembly. The user could, for instance, decide to experiment with different constraint solvers, planner settings, or to filter some parts, and keep these settings and outputs separated into multiple uniquely labelled container objects.

2.2. Part filter

Prior to starting off, the parts considered for planning should be determined. Thus far, the APLAN workbench solely considers products whose geometries remain unaltered. Operations requiring the manipulation of non-rigid materials, or components to temporarily or permanently deform during (dis)assembly are not modelled in the CAD environment. In those cases, the planning of possible (dis)assembly sequences is still feasible provided all self-obstructing features are removed, flexible materials are approximated as rigid, or these deformable components are excluded from the analysis altogether. For the latter option, a part filter can be created, which allows the user to (un)select the components suitable for analysis.

Another functionality offered by this part filter object consists of grouping individual components into uniquely labelled compounds. A compound represents a collection of topological shapes that can be viewed and manipulated as though it is a single solid component (e.g. the bolts of a flange mounting). Another use could be the creation of non-divisible subassemblies when it is undesired to map out a product's entire assembly sequence down to its last bolt. In this case, one could exclude all assembly operations associated with a particular subassembly by grouping its constituent parts in a compound object.

[1] Interested readers are referred to the code repository for details and videos: https://github.com/martcram/FreeCAD-APLAN.

2.3. Constraints

Since the (dis)assembly of an n-sized product can be regarded as an n-permutation, a factorial relationship ($n!$) exists between the size of a product and the number of sequences in which it can be theoretically (dis)assembled. However, due to practical limitations, only a tiny proportion of this set is also physically possible. Hence, before creating assembly plans in APLAN, it is paramount to specify the topological and geometrical constraints of the product under analysis.

Topological constraints are graphically represented as a connection diagram (or liaison graph): an undirected graph whose vertices correspond to the individual components, and the edges represent the components standing in direct physical contact with each other after assembly. Geometrical constraints reflect how the geometries of these components impact each other during (dis)assembly and are formalised as several obstruction diagrams (or blocking graphs), each associated with a specific (dis)assembly motion. The vertices of these undirected graphs correspond to the product's components, while their edges point from the target component to the components that block this target when being disassembled along the direction considered.

Via the APLAN workbench, users have two options for creating connection and obstruction diagrams: (1) manually using the interactive graph visualiser, or (2) automatically using so-called connection and obstruction detectors (see Section 3). While in previous studies, assembly planners mainly conveyed their results as textual representations in the terminal or via static images, the constraint diagrams generated by APLAN are displayed as dynamic force-directed graphs allowing to inspect, create and modify them in an interactive manner. Since JavaScript does not integrate with FreeCAD by default, a browser-like environment was set-up in the proposed module.

2.4. AND/OR graph

Due to their sheer number, a product's feasible assembly sequences must be compiled in a suitable format for human process planners to inspect and learn from them. The AND/OR graph, a directed hypergraph popular for studying decomposable production systems [20], provides such a condensed and human-interpretable representation. Its vertices stand for all subassemblies that can practically be constructed considering topological and geometrical constraints, whereas the hyperarcs connecting them reflect all feasible (dis)assembly operations.

To maintain a combinatorially manageable search for potential assembly sequences, the reverse cut-set algorithm described in [21, p. 188] was implemented in APLAN, which is capable of automatically determining a product's AND/OR graph based on its connection and obstruction diagrams.

2.5. Assumptions

In order to demonstrate the potential of the presented assembly planning module as well as its functionalities, some frequently made assumptions have been adopted. *Assembly-by-disassembly (AbD)*: assembly sequences are formed by reversing the different orders in which the components of a product can be disassembled. *Translational Cartesian movements*: obstruction detectors strictly consider linear (dis)assembly motions along the six Cartesian directions for collisions. *1-movability*: every component can be (dis)assembled from the rest of the product using a single translational motion without creating intermediate assembly states. *Monotony*: no other components should be relocated prior to removing a component. Non-monotone products are often found in locking mechanisms. *2-fold or binary operations*: restricted by our morphology, humans can only efficiently manipulate two things simultaneously, hence focussing on (dis)assembly operations involving two components or subassemblies. *Rigidity*: all assembly components are assumed to be solid and rigid. Additionally, (dis)assembly operations do not require temporary or permanent deformations. *Stability*: components are not subjected to internal (elasticity and friction) or external forces (gravity). When touching, components make up a stable composition.

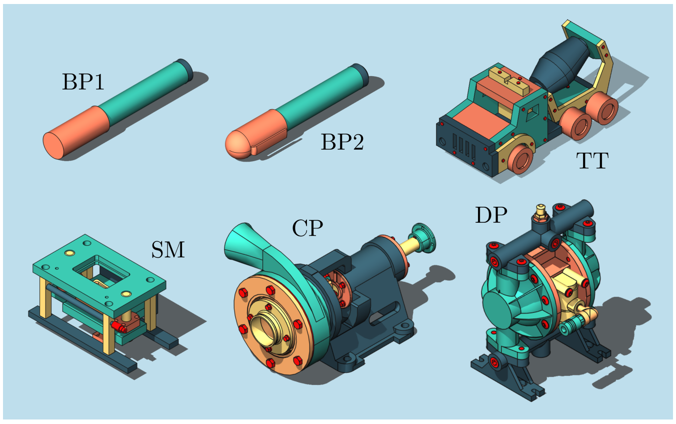

Fig. 1. CAD models of the six assemblies studied in this work: [BP1] simple Bourjault's ballpoint pen (n = 6), [BP2] complex Bourjault's ballpoint pen (n = 6), [TT] toy truck (n = 21), [SM] shear mould (n = 17), [CP] centrifugal pump (n = 22), [DP] diaphragm pump (n = 27) with n: the number of components.

3. Constraint detectors

This section covers the range of solvers available through APLAN to automatically identify topological and geometrical constraints as well as two refinement methods to speed up this process. In order to demonstrate the performance of these automatic techniques and the impact of the developed refiners, six differently-sized assemblies, including both academic and industrial products, are subjected to APLAN's constraint detectors. Figure 1 pictures the CAD models of these products, which can be accessed on GitHub[2].

3.1. Connection detector

This constraint detector identifies which components of a given product physically touch after assembly, without considering the type of connection. Its output consists of the set of

[2] https://github.com/martcram/FreeCAD-APLAN-benchmark.git

topological constraints represented as a connection diagram and stored in the JSON file format. In APLAN, one can create a connection detector supporting the following five solver methods that use the Open CASCADE Technology (OCCT) geometric modelling kernel.

- *BRepExtrema_ShapeProximity*: determines the overlapping faces between two given shapes. The underlying algorithm could be set to solely detect intersecting faces or faces located within a specific tolerance value d_{ov}. Due to its use of pre-triangulated shapes, accuracy is traded for performance.
- *BRepExtrema_DistShapeShape*: computes the minimum distance between two shapes. If the resulting distance is below a certain threshold value d_{min}, the components linked to these shapes are considered touching each other.
- *GeoData & BRepClass3d_SolidClassifier*: samples points on the surface of the smallest of two given components, considering a certain sampling rate. Each point is then sequentially checked on being inside or on the other component's shape.
- *BRepMesh & BRepClass3d_SolidClassifier*: creates a triangle mesh for the smallest of two given components, considering a certain maximum linear deflection. Each vertex is then sequentially checked on the same conditions (using the same function) as for the previous solver.
- *BRepAlgoAPI_Section*: performs a section operation between two shapes. If the resulting shape consists of any vertices, the components linked to both input shapes are considered touching each other.

The constraint detection itself entails determining for every bipartite combination that results from the set of n assembly components whether they stand in physical contact using the formerly described methods. However, it is important to use these functions deliberately and sparingly as they operate on the entire geometry of each component, making them expensive operations. Checking $n \cdot (n-1)$ combinations would only be useful for strongly connected products, which are rare if not non-existent in practice. Hence, the pairs of potentially touching components should first be refined by checking whether the bounding boxes of these components intersect after initially enlarging them along the six Cartesian axes by half of a pre-specified swell distance d_{sw}. The performance of the *BRepExtrema_DistShapeShape* solver and the impact of this refinement method is validated quantitatively in Table 1.

3.2. Obstruction detector

To automatically determine the geometrical constraints associated with a product's (dis)assembly, an obstruction detector can be created. The purpose of this constraint searcher is to examine, for each of the product's components ($\mathbb{C}$) and predefined disassembly directions, which other components prevent the considered target component (T) from being removed during its disassembly. More specifically, starting from the product's assembled state, every component is alternately displaced in discrete steps until entirely separated while the others stay put. During this virtual and stepwise disassembly, the selected solver identifies collisions between T and the set of remaining components ($\mathbb{C}^-$), followed by constructing the obstruction diagram for that particular disassembly motion ($\vec{d}$). This procedure is repeated for all pre-specified motion directions after which the results are exported to JSON.

Table 1: Computation times in seconds (mean ± SEM, k = 5) and *number of contact checks* for finding the topological constraints of six assembly products with and without refinement. Refiner: *SwellBoundBox* (d_{sw} : 0.01 mm); Solver: *BRepExtrema_DistShapeShape* (d_{min} : 1×10^{-5} mm)

Refine	**Model ID**					
	BP1 (n=6)	BP2 (n=6)	SM (n=17)	TT (n=21)	CP (n=22)	DP (n=27)
No	0.091 ± 0.001	0.147 ± 0.003	2.411 ± 0.004	6.201 ± 0.029	24.38 ± 0.02	28.67 ± 0.03
	30	*30*	*272*	*420*	*462*	*702*
Yes	0.070 ± 0.001	0.124 ± 0.003	0.639 ± 0.006	3.223 ± 0.015	8.215 ± 0.022	8.789 ± 0.015
	10	*10*	*24*	*48*	*59*	*60*

At every position of T, and for each combination of $\{T\} \times \mathbb{C}^-$, the fast-but-less-accurate *BRepExtrema_ShapeProximity* function first examines if the faces of the target shape overlap with those of the other component before calling the selected solver. The solver method subsequently confirms whether or not these geometries actually collide. It has been observed that this two-step screening procedure outperforms in terms of computation time as opposed to directly putting the slower, more accurate solver functions into operation. The first three solver methods available through APLAN for establishing a product's geometrical constraints employ the same OCCT functions described in Section 3.1 (number 2 – number 4) but are succeeded by the *BRepFeat_IsInside* operation to ascertain that the provided shapes actually intersect instead of touch. The two remaining solvers rely on the following functions:

- *BOPAlgo_Common*: performs a boolean intersection operation on two given shapes, keeping their common section. If the volume of the resulting geometry exceeds a pre-specified threshold value, both shapes are identified as colliding.
- *BOPAlgo_Fuse*: performs a boolean union operation on two given shapes, joining them together. If the volume of the resulting geometry exceeds the sum of both shapes' separate volumes minus a pre-specified threshold value, both shapes are identified as colliding.

As follows from the foregoing Cartesian product, the solver must perform $n-1$ expensive collisional checks from every element of $\mathbb{C}^-$, at every position of T. To speed up the obstruction analyses, reducing the amount of collision checks is vital. Therefore, the objective of the developed refinement stage is to determine, at each discrete position of T, which subsection of $\mathbb{C}^-$ might potentially obstruct its disassembly along $\vec{d}$. To this end, the presented refiner divides T's disassembly motion into distinct zones, i.e. obstruction intervals, each individually associated with a subset of $\mathbb{C}^-$ that may intersect with T (see Fig. 2). Consequently, when the solver moves T through one of these intervals, only the components in these subsets will be verified for possible collisions resulting in significant time gains.

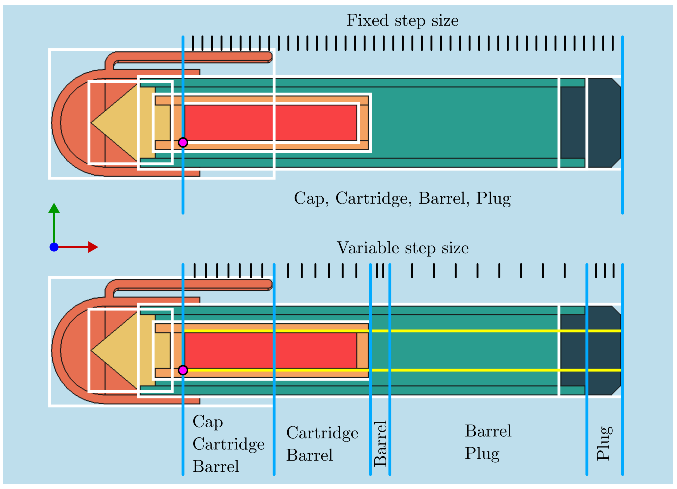

Fig. 2. Expected obstacles for each obstruction interval when disassembling the ink (T) of Bourjault's ballpoint pen along the positive X axis, without (top) and with (bottom) refinement. Legend: (white) bounding boxes, (yellow) T's elongated bounding box, (blue) interval boundaries, (pink) T's origin, (black) positions taken by T's origin when travelling through each obstruction interval.

First, the global bounding box ($\mathbb{C}_{\boxtimes}$) is determined that encloses all $\mathbb{C}$ of the assembly. Next, the target component's bounding box ($T_{\boxtimes}$) is extended to the edge of $\mathbb{C}_{\boxtimes}$ according to $\vec{d}$. Subsequently, the intersections are calculated between the bounding boxes of $\mathbb{C}^{-}$ and the elongated bounding box ($T^{*}_{\boxtimes}$) resulting from the previous process step. The outer dimensions of these beam-shaped intersections, projected onto $\vec{d}$ and relative to the origin (O), represent the components' intersection intervals. O is defined to equal the initial position of T when in its assembled state, from which all other positions and dimensions are referenced. The lower bound of each intersection interval is subsequently reduced by the size of $T_{\boxtimes}$, according to $\vec{d}$, and where necessary, truncated at the lower bound of $T_{\boxtimes}$ projected onto $\vec{d}$; the upper bound of each interval remains unaltered. Finally, the boundaries of the previously introduced obstruction intervals can be obtained by sorting the limits of these intersection intervals in ascending order and grouping them into pairs. The set of expected obstacles ($\mathbb{E}_i$) in each obstruction interval i is then obtained by establishing which intersection intervals of the components in $\mathbb{C}^{-}$ overlap with these obstruction intervals. If the solver has identified every potential collision in a particular obstruction interval j, given by $\mathbb{E}_j$, the travel of T through that interval is aborted and continued in the next.

Besides reducing the number of collision checks at each position of T, additional time gains are obtained by decreasing the number of positions traversed inside an obstruction interval. To this end, the fixed step size x over which the solver incrementally moves T is varied proportionally as a function of the length of the current obstruction interval considering ratio r and minimum step size x_{min}. A final saving is achieved by distributing the individual obstruction analyses for each $\vec{d}$ across multiple processes, therewith performing these operations in parallel. The performance of the *BRepExtrema_DistShapeShape* solver and the impact of the aforementioned enhancements are determined quantitatively in Table 2.

Table 2: Computation times in seconds (mean ± SEM, k = 3) and *number of collision checks* for finding the geometrical constraints of six assembly products along the three positive Cartesian directions, with and without refinement and a varying step size, each executed on a separate process of the PC used. Refiner: *BoundBox* (x : 1.0 mm, x_{min} : 1.0 mm, r : 0.10); Solver: *BRepExtrema_DistShapeShape* (d_{ov} : 1×10^{-5} mm, d_{min} : 1×10^{-5} mm)

Refine	**Model ID**					
	BP1 (n=6)	BP2 (n=6)	SM (n=17)	TT (n=21)	CP (n=22)	DP (n=27)
No	2.229 ± 0.003	5.877 ± 0.020	186.1 ± 0.8	325.6 ± 0.3	> 3600	1063 ± 4
	6,500	*3,275*	*71,952*	*84,440*	*471,597*	*265,902*
Yes	0.455 ± 0.001	1.072 ± 0.006	6.568 ± 0.041	57.47 ± 0.09	407.4 ± 1.2	392.6 ± 2.2
	1,217	*1,217*	*3,116*	*6,313*	*23,769*	*14,496*

4. Discussion

4.1. Benchmarking

Tables 1 and 2 report on the recorded computation times required by APLAN's connection and obstruction detectors for respectively determining the topological and geometrical constraints of the six differently-sized products pictured in Figure 1. In addition, the impact of the two self-developed refinement procedures is analysed. These calculation were performed on a PC with an Intel® Core™ i9-9920X CPU @ 3.50 GHz × 24 and 62.5 GB RAM memory. For the study of both detectors, the *BRepExtrema_DistShapeShape* solver was chosen due to its relatively high accuracy and acceptable speed compared to the other solvers described in Sections 3.1 and 3.2.

It has been observed that the connection detector's performance depends on the product size n (cfr. SM and DP) and the complexity of its constituent components (cfr. BP1 and BP2), measured by the number of vertices, edges and faces they comprise of. Without any further optimisations, the set of n components can be paired into $n \cdot (n-1)$ combinations requiring an equal amount of computationally expensive contact checks, which lead to calculation times between 91 ms (BP1) and 28.7 s (DP) for the products considered. To reduce the number of solver operations, a refinement procedure was implemented that first screens every component pair based on whether their enlarged bounding boxes intersect. This reduced the amount of contact checks between 66.7% and 91.5% translating to a time saving of 15.7% (BP2) up to 73.5% (SM).

As defined by the assumptions in Section 2, only Cartesian directions of disassembly are considered for obstruction detection in APLAN. Moreover, since the obstruction graph along a particular direction equals the transpose of that graph for the opposite direction, calculations were only performed for the three positive coordinate axes, and distributed across separate processes. In absence of any refinement stage, the selected solver must at most perform, for each component sequentially, $n-1$ collision checks at every position of this target during its disassembly along one of the pre-specified directions. For a fixed step size of 1 mm, this means a maximum of 3,275 (BP2) and 471,597 (CP) computationally demanding solver operations to compose three constraint diagrams. In terms of computation times, the obstruction graphs of the six products were found in 2.2 s (BP1) up to 17.7 min (DP). To be of practical relevance for the industry, a refiner was presented that reduced the number of collision checks by 62.8% up to 95.7%, resulting in significant time gains between 63.1% (DP) and 96.47% (SM).

4.2. Applications

The authors formerly employed the introduced FreeCAD workbench in the area of human-robot collaborative assembly [22]. From APLAN's output, a probabilistic decision model was automatically constructed based on which a robotic assistant could select and perform joint assembly operations, considering its human partner's intention. The assembly planning framework has also been used in an industrial research project targeting the development of an inclusive assembly cell for sheltered workers [23]. APLAN assisted in exploring optimal assembly sequences as well as designing multi-purpose jigs and fixtures.

5. Conclusion

While many assembly sequencing algorithms already exist, academia and industry still face intellectual and financial barriers when applying these software tools themselves. This paper, therefore, presents the APLAN workbench: a software framework that enables developers to share their (dis)assembly planning algorithms as part of the open-source 3D parametric modeller FreeCAD in a ready-made and centralised way.

As a starting point, APLAN offers two different detectors for automatically identifying an assembly's topological and geometrical constraints. As demonstrated by performance tests on six academic and industrial products, the two newly developed refinement procedures drastically decreased the computation times, therewith facilitating this module's uptake by industry. In order to ease the inspection or modification of the resulting constraint graphs, APLAN also features an interactive graph visualiser. Given the former constraints, users are finally provided with an academic AND/OR graph generator to determine all feasible sequences in which a product can be (dis)assembled.

Future work will focus on benchmarking the remaining solvers of APLAN's connection and obstruction detectors and including extra assembly sequence generators. As understanding the type of connection between components could improve assembly orders and computing times, adding such classifiers will be considered for upcoming extensions. Furthermore, since AND/OR graphs may quickly become cluttered for large or complex products, the interactive graph visualiser will be extended to support these hypergraphs as well.

Acknowledgements

Martijn Cramer would like to thank the FWO (Research Foundation - Flanders) for facilitating this research under grant agreements 1SA6919N and 1SA6921N.

References

[1] A. Buffoni, A. de Angelis, V. Grüntges, A. Krieg, How to make sure your next product or service launch drives growth, https://www.mckinsey.com, accessed: 2023-09-26 (2017).

[2] N. Arora, et al., The value of getting personalization right—or wrong—is multiplying, https://www.mckinsey.com, accessed: 2023-09-26 (2021).

[3] S. K. Ong, M. M. L. Chang, A. Y. C. Nee, Product disassembly sequence planning: state-of-the-art, challenges, opportunities and future directions, Int. J. Prod. Res. 59 (11) (2021) 3493–3508.

[4] B. Hasan, J. Wikander, A review on utilizing ontological approaches in integrating assembly design and assembly process planning (APP), Int. J. Multicult. Educ. 4 (11) (2017) 5–16.

[5] A. Petruccioli, F. Pini, F. Leali, Model-based approach for optimal allocation of GD&T, in: Proc. Int. Conf. Des. Tools Methods Ind. Eng., 2022, pp. 277–284.

[6] H. Gong, L. Shi, D. Liu, J. Qian, Z. Zhang, Construction and implementation of extraction rules for assembly hierarchy information of a product based on OntoSTEP, Procedia CIRP 97 (2021) 514–519.

[7] J. Qian, Z. Zhang, C. Shao, H. Gong, D. Liu, Assembly sequence planning method based on knowledge and ontostep, Procedia CIRP 97 (2021) 502–507.

[8] C. Pan, S. S. Smith, G. C. Smith, Automatic assembly sequence planning from STEP CAD files, Int. J. Comput. Integr. Manuf. 19 (8) (2006) 775–783.

[9] D. Agrawal, S. Kumara, Automated assembly sequence planning and subassembly detection, in: Proc. ISERC, 2014, pp. 781–788.

[10] A. Neb, Review on approaches to generate assembly sequences by extraction of assembly features from 3D models, Procedia CIRP 81 (2019) 856–861.

[11] P. Neto, N. Mendes, R. Araújo, J. Norberto Pires, A. Paulo Moreira, High-level robot programming based on CAD: dealing with unpredictable environments, Ind. Rob. 39 (3) (2012) 294–303.

[12] M. V. A. R. Bahubalendruni, B. B. Biswal, B. B. V. L. Deepak, Computer aided assembly attributes retrieval methods for automated assembly sequence generation, Int. J. Min. Miner. Eng. 11 (4) (2017) 759–767.

[13] S. Münker, R. H. Schmitt, CAD-based AND/OR graph generation algorithms in (dis)assembly sequence planning of complex products, Procedia CIRP 106 (2022) 144–149.

[14] C. M. Costa, et al., Automatic generation of disassembly sequences and exploded views from SolidWorks symbolic geometric relationships, in: IEEE ICARSC, 2018, pp. 211–218.

[15] L.-M. Ou, X. Xu, Relationship matrix based automatic assembly sequence generation from a CAD model, Comput. Aided Des. 45 (7) (2013) 1053–1067.

[16] S. Tao, M. Hu, A contact relation analysis approach to assembly sequence planning for assembly models, Comput. Aided Des. Appl. 14 (6) (2017) 720–733.

[17] J. Riegel, W. Mayer, Y. van Havre, FreeCAD (version 0.22.0), [Software] Available from `https://www.freecadweb.org` (2023).

[18] R. Viganò, G. Osorio-Gómez, ASPIP - automatic and assisted definition of assembly sequences based on component liaisons and subassemblies approach, J. Adv. Manuf. Technol. 5 (3) (2021) 15–28.

[19] M. Yao, et al., Computer-aided assembly sequence planning for high-mix low-volume products in the electronic appliances industry, in: Proc. CPSL, 2022, pp. 91–100.

[20] L. S. Homem de Mello, A. C. Sanderson, AND/OR graph representation of assembly plans, IEEE Trans. Robot. Autom. 6 (2) (1990) 188–199.

[21] A. J. D. Lambert, S. M. Gupta, Disassembly modeling for assembly, maintenance, reuse, and recycling, The St. Lucie Press series on resource management, CRC Press, 2005.

[22] M. Cramer, K. Kellens, E. Demeester, Probabilistic decision model for adaptive task planning in human-robot collaborative assembly based on designer and operator intents, IEEE Robot. Autom. Lett. 6 (4) (2021) 7325–7332.

[23] Flanders Make, Innovative assembly cell boosts self-confidence of workers, https://www.flandersmake.be, accessed: 2023-09-26 (2023).