Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2017) 000–000

www.elsevier.com/locate/procedia

11th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 19-21 July 2017, Gulf of Naples, Italy

Application of generic CAD models for supporting feature based assembly process planning

Csaba Kardos[a,b]*, József Váncza[a,b]

[a]*Fraunhofer Project Center for Production Management and Informatics*

Institute for Computer Science and Control, Hungarian Academy of Sciences

[b]*Department of Manufacturing Science and Technology, Budapest University of Technology and Economics*

* Corresponding author. Tel.: +36-1-279-6189; *E-mail address:* csaba.kardos@sztaki.mta.hu

Abstract

The paper discusses a novel geometric reasoning method that supports the definition of assembly sequence planning models departing from the CAD models of the parts involved. Specifically, by means of the presented algorithms that use a so-called collision point cloud approach one can determine the precise disassembly directions of parts having complex polygon mesh models. This information can be applied when defining assembly planning models both for suggesting precedence constraints as well as parameters for assembly operations. The presented heuristic algorithm was able to overcome certain shortcomings of earlier methods working with polygon mesh representations, and proved to be successful both in handling abstract and real-life industrial use cases. Working examples from both categories are presented in the paper.

Peer-review under responsibility of the scientific committee of the 11th CIRP Conference on Intelligent Computation in Manufacturing Engineering.

Keywords: Assembly, Planning, Geometric reasoning

1. Introduction

As assembly technology and systems are of key importance to achieve high variety production in a cost efficient way, the need for automated assembly planning methods are discussed broadly and from different aspects in the literature [1]. A common classification of assembly planning problems defines the following sub-problems: Assembly Sequence Planning (ASP), Assembly Line Balancing (ALB) and Assembly Path Planning (APP).

This paper focuses on the problem of ASP, namely on finding an executable—and, if possible, optimal—order of assembly operations along with the assignment of appropriate resources such as tools, fixtures and workforce. Since there are many kinds of factors that may have an impact on the feasibility of an assembly plan—like detailed geometrical dimensions and relations of objects involved, assembly technology, tolerances, elasticity of parts, etc.—most of the ASP approaches are in common that they decompose the problem into (1) a macro-level planning problem where all the combinatorial complexities of sequencing and resource assignment are concentrated, and (2) into a micro-level problem which is responsible for tackling all kinds of domain constraints in their entire diversity. Feasibility of assembly plans on the micro level can be warranted by considering appropriate constraints already on the macro level. A major category of such constraints are precedence constraints between tasks which realize distinct, well-defined elementary features—such as placing, inserting, welding—of the overall planning problem. These precedence constraints are coming from the micro-level analysis of geometrical, physical and technological feasibility [2] and therefore there is a close connection between ASP and APP [3].

Even though macro-level planning relies on the results coming from the micro level, just because of the different modeling techniques used they are usually handled separately. Nevertheless, in order to completely utilize the potential of combinatorial ASP models, they need to be supported with

Peer-review under responsibility of the scientific committee of the 11th CIRP Conference on Intelligent Computation in Manufacturing Engineering.

extensive micro-level knowledge through their whole operation. A generic way to provide this support is to extract the required micro-level input data directly from the CAD models of the products, thus bringing closer the ASP problem to product design. This paper in particular presents a novel approach to support this process with extracting parametric precedence relations from the polygon mesh models of objects involved in assembly operations.

The rest of the paper is organized as follows: in section 2 a short overview of ASP representations is given and previous works focused on supporting the definition of ASP including CAD product models are introduced. Section 3 presents the problem of extracting this information from polygon mesh models and discusses the limitations of this representation, while section 4 proposes a heuristic algorithm for solving the problem. Section 5 details the implementation of the algorithms, demonstrates its capabilities in abstract and real-life use cases and discusses the limitations of the approach.

2. Review of previous works

As there are several approaches to solve ASP and its related problems, variety in modeling also shows in the applied representations: there is no standardized representation. Without providing the detailed overview of the various existing representations a brief listing can be given: liaison graph, AND/OR graph, blocking graph, precedence and interference matrices, object and ontology based hierarchies, feature based approaches [1].

Applying these models nevertheless can be challenging and time consuming even in simple cases, which limits their overall usability [4]. Hence, numerous researches are aimed to overcome these limitations and propose a solution to support ASP model building with geometrical reasoning which is firmly based on data extraction.

The starting point of these approaches is usually the CAD model of the assembly and a possible classification of the applied methods can be given based on the different types of the CAD models and the application tools put in use to handle them. Assembly CAD models can be available (1) through the interfaces of a commercial CAD software, or (2) they can also be accessed through a descriptive open exchange format such as STEP, or (3) presented in a completely generic representation such as polygon meshes.

In [6] the proposed system utilizes the programmable interfaces (API) of a commercial CAD software to analyze the analytical surfaces to search for contacts defining the assembly relations between parts. In [7] the API of a CAD software is applied to determine feasible assembly sequences for layered assemblies (with only one assembly direction). The approach presented in [8] accesses the mating relations of an assembly CAD model through a commercial software in order to derive the assembly operations. In contrast of these approaches, in [9] it is stated that there is not necessarily coherence between mating relations defined during the product design and those required for assembly planning, and an approach is suggested which relies only on part contact information.

In [10] an early approach is presented to identify assembly/disassembly directions using surface normals, but only for the three Cartesian directions. In the research presented in [11] the assembly directions are retrieved by using normal vectors of of contact surfaces, while in [12] the Object-oriented Bounding Boxes (OBB) of parts and the so-called separation axis theorem is applied in order to identify directions of assembly/disassembly. Also there are several other methods that apply motion planning in order to find assembly/disassembly directions of parts [3, 13]. This approach is advantageous as six-dimensional paths can be obtained, opposed to the assumptions of mostly translational or helical movements of other approaches. However, for assemblies where the parts are tightly constrained spatially and only fine movements are executable, applying path planning can be difficult [3].

3. Problem statement

Concluding from the previously introduced researches on geometrical reasoning supported data extraction for ASP, the following shortcomings of current approaches can be pointed out:

- Using APIs of commercial CAD software bounds the solution to a given software, which usually has its native (and closed) file format, thus a conversion is often required.
- Succeeding in the above mentioned conversion or using an open exchange file format as STEP can solve this problem, however in industrial use cases very strict rules can apply to the accessibility of original CAD models, if they are shared at all.
- Applying generic polygon meshes is widely accepted with much less restrictions, they can be obtained even from 3D scanning. However, they often have lower quality: the meshes can intersect each other, there is no guarantee to have proper surface normal information available. All these limit the usual geometrical reasoning approaches.

In order to overcome the above shortcomings the goal of the paper is to define a geometrical reasoning approach that supports extraction of assembly parameters for a feature based ASP model using generic polygon mesh models, while fulfilling the following requirements: It should be able to

- work with low quality, intersecting polygon meshes;
- operate on parts with tight spatial constraints; and
- identify arbitrary translational assembly axis.

Feature based assembly models are introduced in [14,15]; the features in scope for the research are insertion and placement of components (where insertion can also be applied as a simplified model for screw joints). The assumptions related to this task are the following:

1. Parts are unique (rigid, tolerance-free), free flying, 3D geometries which allow the assembly-by-disassembly approach [3]
2. Assembly operations described by the features are two-handed and implement translational movements in any direction.
3. The translational movements are completely defined by the vertices in contact and their near neighborhood.
4. The type of assembly features are predefined.

4. Polygon meshes in micro-level process planning

Following the requirements and assumptions of the problem statement the basic idea of assembly-by-disassembly was put in use: starting from the assembled state of two parts a feasible disassembly path (in this case a direction) produces a feasible assembly path. Having the feature types predefined, the analysis considers two parts at a time. An algorithm for finding translational disassembly path of two polygon meshes is described in [5], which uses contact information between two parts to build so called Local Translational Freedom Cones (LTC): "a Local Translational Freedom Cone is the subset of directions, from the entire sphere of directions, over which a given part can move locally without colliding with another given part [5]". Similar approach is introduced in [12] where the Constraint Directions (CD) represent the directional blocking relationship between two parts. Both methods rely on moving one component in directions sampled from a unit sphere and on collision detection to determine if there is a block in that direction. However, these approaches fail in cases when the mesh models are overlapping (i.e. are colliding) in their assembled state as there will be no collision free translational movement to disassemble them. The approach presented here is a heuristics, which tries to overcome this limitation often occurring in the case of real-life mesh models.

According to assumption 3 of the problem statement, the key issue is to determine the assembly directions by exploring the contacts between two parts. In order to discover the contacts one part (*moved part*) is translated in directions of a unit sphere defined by the following:

$$w_1 = \cos(\alpha) \cdot \sin(\beta) \cdot r \quad (1)$$

$$w_2 = \sin(\alpha) \cdot \sin(\beta) \cdot r \quad (2)$$

$$w_3 = \cos(\beta) \cdot r \quad (3)$$

$$v = w_1 e_1 + w_2 e_2 + w_3 e_3 \quad (4)$$

Where $\alpha \in [0, 2\pi], \beta \in [0, \pi], r = 1$ and e_1, e_2, e_3 are orthogonal vectors. Discretizing the intervals for α and β is defined by an *angular_step* (which is by default 10°) results in 614 different directions forming a unit sphere. After applying the sampled translations, by capturing the contact points (i.e. vertices of colliding triangles on both parts) two collision point clouds (CPL_1 for the *non-moved part* and CPL_2 for the moved part) are registered. A CPL represents the intersections of the two parts in terms of blocking relations (see Fig. 2 for illustration).

In the simplest case of placement, where the two parts are mated by a coplanar assembly constraint, the CPL forms a plane and the expected LTCs build up a half-sphere. A normal vector of this plane therefore is a valid disassembly direction. The normal vector of a point cloud forming a plane can be determined by using three dimensional Principal Component Analysis (PCA), as the point cloud will have its first two principal axes lying on the plane and since the principal axes are orthogonal, the third axis is a normal vector of the planar point cloud. Fig. 2 shows an example of the resulting CPL of two cuboid geometries, which are sharing a common side.

Experiments showed that using PCA to other CPL primitives (e.g. in cylindrical or cuboid shapes) returns good initial guesses for feasible disassembly directions. Using the axes of PCA as the basis (e_1, e_2, e_3) a set of vectors (D) are calculated using (4). These vectors are utilized in determining the LTC.

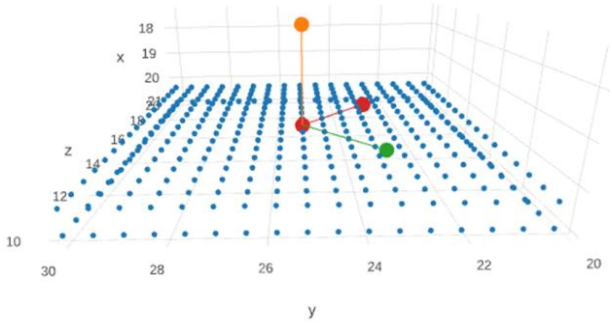

Figure 1: The Collision Point Cloud (CPL) of two cuboid geometries. The red, green and orange vectors are the results of PCA.

Calculating the geometrical mean of CPL_2 defines a center point (C) from which LTCs are constructed. Another set of vectors (V_1) are defined such that:

$$v_{1i} = P_i - C$$

for all $p_i \in CPL_1$.

The LTCs are determined by comparing the vectors of V_1 to directions of D by using the cosine distance:

$$d_{\cos(u,v)} = \frac{u \cdot v}{\|u\|_2 \cdot \|v\|_2}$$

which returns a distance matrix (M), such that:

$$M_{i,j} = d_{\cos(v_{1i}, d_j)}$$

for all $v_{1i} \in V_1$ and $d_j \in D$. Selecting the minimal value of each column of M gives the angle of the LTC around the axis corresponding to the direction represented by each column. In case of an insertion feature the direction with the largest value gives the insertion direction, while in the case of placement applying a lower threshold for these values provides the set of directions which can be possible placing directions. It is worth mentioning that though it would appear that the application of PCA is very intuitive, it is only used for providing the basis of sampling and with a higher sampling resolution, the significance of the chosen bases decreases.)

5. Implementation and test results

5.1. Implementation of prototype

In order to implement the heuristics described in Section 4 different tools were put together using the Python programming language. A crucial part of the implementation is the collision detection, which provides the CPLs. The Flexible Collision Library (FCL) was used as it offers powerful and efficient capabilities to handle polygon meshes in collision detection by applying the Bounding Volume Hierarchies (BVH) representation [16]. Another important requirement was the efficient computation of large distance matrices, which was provided by the numpy computation module. The prototype system was implemented by using the interactive computing environment of IPython/Juypter notebooks [17], which supports JavaScript based visualization through the Plotly

module [18]. The pseudo-code of the algorithms are presented below:

```
function GET_FEATURE_AXIS(body1, body2)
1       bvh1 ← CREATE_BVH_MODEL(body1)
2       bvh2 ← CREATE_BVH_MODEL(body2)
3       sample_directions ← GET_DIRECTIONS(
          i
          j,
          k,
          angular_step,
          radial_distance=1
        )
4       contact_triangles1, contact_triangles2 ← CHECK_COLLISION(bvh1, bvh2, sample directions)
5       contact_points1 ← body1[contact_triangles1]
6       contact_points2 ← body2[contact_triangles2]
7       REMOVE_DUPLICATES(contact_points1)
8       REMOVE_DUPLICATES(contact_points2)
9       bases ← PCA(contact_points1, n_dim=3)
10      center ← MEAN(contact_points2)
11      axis ← AXIS_SELECTION(bases, center, contact_points1, angular_step, radial_distance=1)
12      return axis

function AXIS_SELECTION(bases, center, contact_points, angular_step, radial_distance)
1       directions ← GET_DIRECTIONS(
          bases[0],
          bases[1],
          bases[2],
          angular_step,
          radial_distance=1
        )
2       for i in LENGTH(contact points):
3         for j in LENGTH(directions):
4                 p ← contact_points[i]
5                 d ← directions[j]
6                 vector ← p - center
7                 distance_matrix[i,j] ← CDIST(vector, d)
8       axis ← directions[MIN_INDEX(distance_matrix)]
9       return axis
```

5.2. Abstract and industrial test cases

To test and evaluate the algorithms four abstract test cases were constructed of cuboid geometries, each representing different CPL shapes. The geometries of the test cases are shown in Fig. 2, while Fig. 3 shows the results of the algorithm. It can be concluded that in the test cases the CPLs are indeed representing the surfaces creating the blocking relations between the parts. The free path of movements, which are displayed by spheres positioned to the center of the CPL of the moved part, are also valid compared to the original geometries. In order to discuss the results of the test cases a series of tests were executed, which aimed to analyze the algorithms' sensitivity to the number of faces in the geometries. The tests were executed on a virtual machine, with 8 virtual CPU cores and 16 GB of memory.

Increasing the number of faces on one hand obviously demands more computational capacity, but on the other hand the less faces in contact decreases the density of the CPLs and therefore the reliability of the algorithm. Table 1 and Fig. 5. show the results of the tests, where the number of faces representing the geometries were increased in 5 steps. In Table 1 the threshold for the axis selection was set arbitrarily to 0.03 and the results show that in each case the more faces resulted more contacts and therefore less feasible directions, which in these cases mean better precision. Notably, in case "a" with the finest model resolution the algorithm failed to return the results with the given memory limit. Since this case is an insertion feature the geometries are constrained more than those are in the other cases. This produces more contact faces and almost all of them had to be identified for every direction during the collision tests, which means an increased computational load.

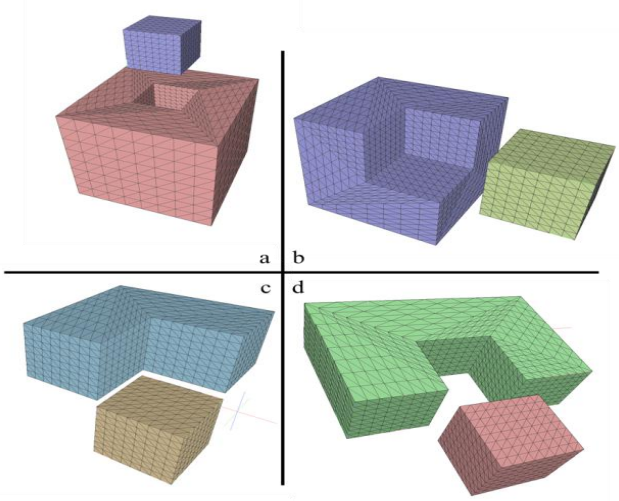

Figure 2: Four abstract test cases for the proposed algorithm, each representing different CPLs.

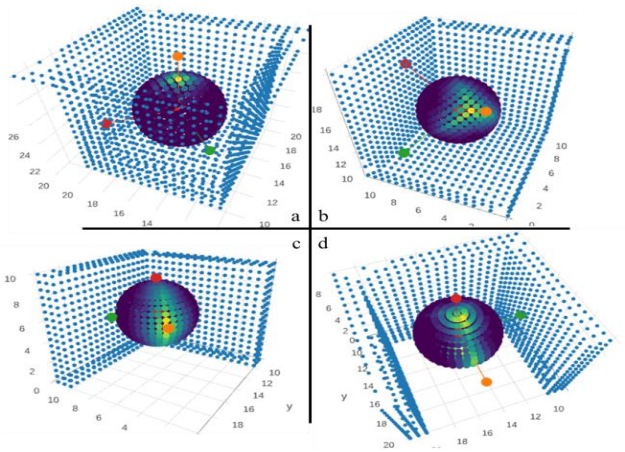

Figure 3: The resulting CPL for each of the abstract test cases and the evaluation of the sampled directions, represented by the coloured spheres. Brighter colours display better directions for disassembly.

After the promising results of the abstract test cases, further tests were executed on subassemblies from real-life industrial demonstrator cases. The assembly presented in Fig. 4 is composed of two parts, one is functioning as a shell of the other, each part represented by more than 80k faces. The parts have several wrongly defined surface normal and in their initial assembled state there are more than 40k colliding faces, thus making it hard to extract information with traditional methods. The results in this real-life use case turned out to be reliable and further subassemblies of the same demonstrator were analysed successfully and the results were put in use in an ASP model.

Res. of model	Assembly ID	a	b	c	d
1	#Faces	160	144	128	160
	#Contacts	168	168	120	144
	#Feasible axes	391	378	476	406
	Exec. time (s)	0.7	0.7	0.5	0.7
2	#Faces	640	576	512	640
	#Contacts	576	480	336	432
	#Feasible axes	137	332	382	253
	Exec. time (s)	2.0	1.5	1.1	1.5
3	#Faces	2560	2304	2048	2560
	#Contacts	2112	1536	1056	1440
	#Feasible axes	37	272	296	123
	Exec. time (s)	4.9	3.5	2.4	3.7
4	#Faces	10240	9216	8192	10240
	#Contacts	8448	6144	4224	5760
	#Feasible axes	37	268	270	122
	Exec. time (s)	13.5	8.9	6.7	10.1

Table 1: Analysing the algorithm on different mesh model resolutions of the four test cases. The larger number of contacts results more accurate calculations and therefore less feasible directions.

5.3. *Discussion of limitations and future works*

The abstract and industrial use-cases showed that the proposed algorithm is capable to extract feasible disassembly axes from polygon meshes. However, the heuristic nature of the algorithm demands certain limitations to be pointed out. Some of these are stemming from initial assumptions, such as that only translational disassembly directions can be extracted. The assembly features types need to be predefined, as currently the algorithm is only able to return a relative evaluation of the possible disassembly axes. Otherwise, result may be returned even for a completely blocked part as well. There are cases when using the resulting CPL is not suitable because the geometry of the joining parts are violating assumption 3. Other limitations are imposed on the algorithm by its input parameters: as shown in Table 1, there is a lower and upper bound on the number of faces in contact (the former for ensuring the CPL to be dense enough and the latter for reducing computational load). Not discussed in details, the *angular_step* and the *radial_distance* parameters applied during the sampling can also influence the CPL and therefore the results of the algorithm.

Considering these limitations in its current form the algorithm is rather a support tool and is not to be used without supervision. However, the results were already put in use in real-life industrial use cases, reducing the effort required to provide data for ASP models. The limitations also show the direction of further developments: specifying the basic assumptions in a more formal way would ensure that the algorithm is only fed with suitable cases; determining the feature types and infeasibility from the CPL would reduce the algorithms' exposition to predefined input. Furthermore a comprehensive sensitivity analysis on the parameters *angular_step* and *radial_distance* are also required.

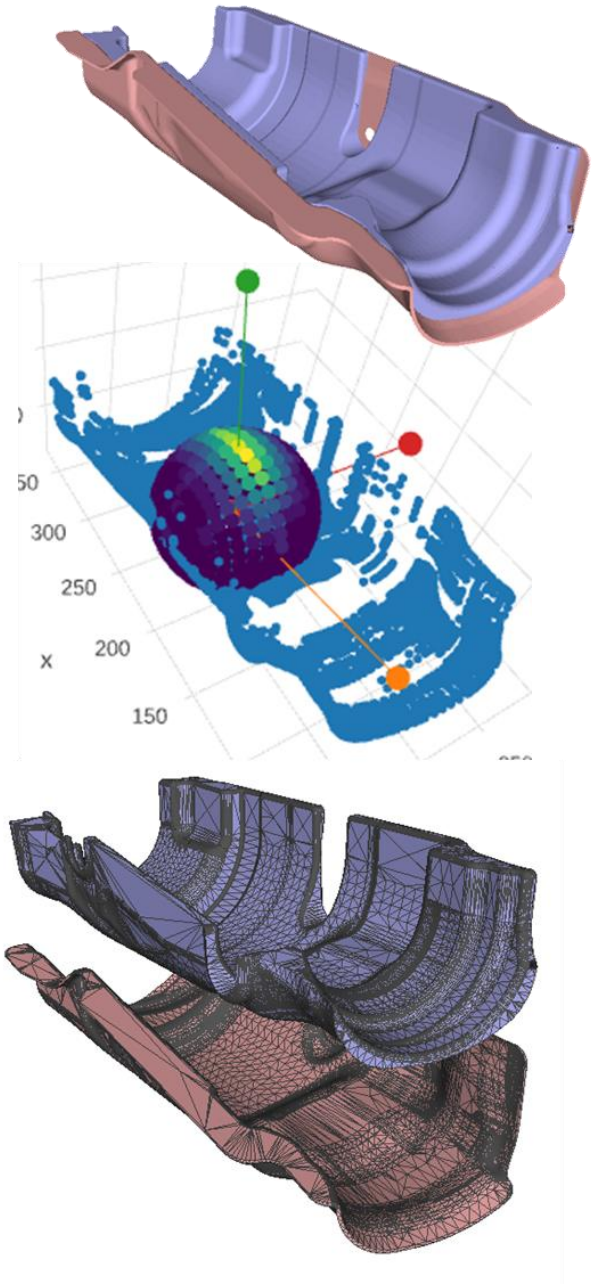

Figure 4: Demonstration of the algorithm in a real-life industrial use with two components. The components were disassembled along the direction with the highest angle LTC. The CPL is composed of more than 10^5 points.

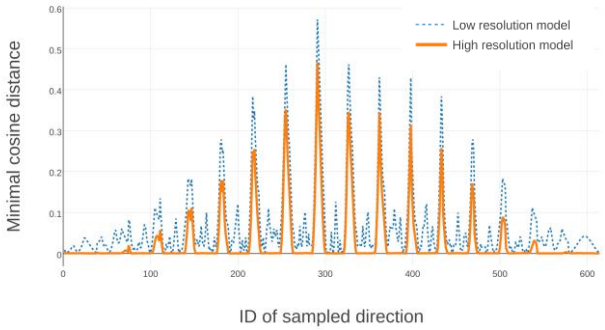

Figure 5: The diagram compares the minimal value of cosine distance in each direction between models with high and low resoultion (test case b, resolution 1 and 4). It can be seen that the higher model resolution results in less suitable directions.

6. Conclusions and future works

The paper discussed the importance of generic geometric reasoning algorithms for ASP models, supporting especially the model building phase, where extracted micro-level, typically geometric information can play a crucial role in solving the macro-level counterpart of the overall assembly planning problem. Specifically, by means of the suggested geometric reasoning algorithms one can determine the precise disassembly directions of parts having complex polygon mesh models. This information can be applied when defining assembly planning models both for suggesting precedence constraints as well as parameters for assembly operations. The presented heuristic algorithm was able to overcome certain shortcomings of earlier methods working with polygon mesh representations, and proved to be successful both in handling abstract and real-life industrial use cases [19].

Acknowledgments

This research has been supported by the GINOP-2.3.2-15-2016-00002 grant on an "Industry 4.0 research and innovation center of excellence" and by the EU H2020 Grant SYMBIO-TIC No. 637107.

References

[1] Hu SJ, Ko J, Weyand L, ElMaraghy HA, Lien TK, Koren Y, Bley H, Chryssolouris G, Nasr N, Shpitalni M (2011) Assembly system design and operations for product variety. CIRP Annals - Manufacturing Technology 60(2): 715–733.

[2] Jiménez P (2013) Survey on assembly sequencing: a combinatorial and geometrical perspective. Journal of Intelligent Manufacturing 24(2): 235–250.

[3] Ghandi S, Masehian E. (2015) Review and taxonomies of assembly and disassembly path planning problems and approaches. Computer-Aided Design 67–68: 58–86.

[4] Bahubalendruni MVAR, Biswal BB (2016) A review on assembly sequence generation and its automation. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 230(5):824–838.

[5] Romney B, Godard C, Goldwasser M, Ramkumar G (1995) An efficient system for geometric assembly sequence generation and evaluation. Computers in Engineering: 699–712.

[6] Lupinetti K, Giannini F, Monti M, Pernot JP (2016) Automatic Extraction of Assembly Component Relationships for Assembly Model Retrieval. Procedia CIRP 50: 472–477.

[7] Michniewicz J, Reinhart G, Boschert S (2016) CAD-Based Automated Assembly Planning for Variable Products in Modular Production Systems. Procedia CIRP 44: 44–49.

[8] Bahubalendruni MVAR, Biswal BB, Upadhyay V (2014) Assembly Sequence Generation and Automation. International Conference on Design, Manufacturing and Mechatronics 2014: 185-192.

[9] Viganò R, Osorio Gómez G (2013). Automatic assembly sequence exploration without precedence definition. International Journal on Interactive Design and Manufacturing (IJIDeM) 7: 79–89.

[10] Lee S, Shin YG (1990) A cooperative planning system for flexible assembly. Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing: 306–313.

[11] Wan W, Harada K, Nagata K (2016). Assembly Sequence Planning for Motion Planning. arXiv Preprint arXiv:1609.03108.

[12] Su Q, Lai SJ (2010) 3D geometric constraint analysis and its application on the spatial assembly sequence planning. International Journal of Production Research 48: 1395–1414.

[13] Morato C, Kaipa KN, Gupta SK (2013) Improving assembly precedence constraint generation by utilizing motion planning and part interaction clusters. Computer-Aided Design 45(11): 1349–1364.

[14] Kardos C, Kovács A, Váncza J (2016) Towards Feature-based Human-robot Assembly Process Planning. Procedia CIRP 57:516–521.

[15] van Holland W, Bronsvoort WF (2000) Assembly features in modeling and planning. Robotics and Computer-Integrated Manufacturing 16(4): 277–294.

[16] Pan J, Chitta S, Manocha D (2012) FCL: A general purpose library for collision and proximity queries. In 2012 IEEE International Conference on Robotics and Automation (ICRA): 3859–3866.

[17] Perez F, Granger BE (2007) IPython: A System for Interactive Scientific Computing. Computing in Science & Engineering 9(3): 21–29.

[18] Plotly Inc, (2015) Collaborative data science. https://plot.ly

[19] Kardos C, Kovács A, Váncza J (2017) Decomposition approach to optimal feature-based assembly process planning. CIRP Annals: (in press)