Procedia CIRP 97 (2020) 39–44

Contents lists available at ScienceDirect

Procedia CIRP

journal homepage: www.elsevier.com/locate/procir

Semi-automatic extraction of digital work instructions from CAD models

Dorothy Gors[a,*], Jeroen Put[b], Bram Vanherle[b], Maarten Witters[a], Kris Luyten[b]

[a] Flanders Make, Gaston Geenslaan 8, B-3001 Leuen, Belgium
[b] Hassel University - tUL - Flanders Make, Expertise Centre for Digital Media, Belgium

ARTICLE INFO

Keywords:
Digital work instructions
CAD models
Assembly sequence

ABSTRACT

Currently process engineers are using documents or authoring tools to bring the assembly instructions to the work floor. This is a time-consuming task, as instructions need to be created for each assembly operation. Furthermore, the engineer needs to be familiar with the assembly sequence. To assist the engineer, a tool is developed that i) uses a heuristic based on visibility, part similarity and proximity to semi-automatically determine the assembly sequence from a CAD model and ii) according to the computed sequence generates digital work instructions including visualizations and animations extracted from the CAD model. In essence, the assembly sequence generation works reversely: it determines the order in which components can be removed from the assembly, by evaluating whether the visibility of a component is obstructed by the remaining assembly. The reversed order is then returned as assembly sequence. During this process the engineer can modify the proposed sequence, add annotations and alter the visualizations of the proposed instructions, i.e., images or 3D-animations. We illustrate that the developed tool effectively supports process engineers and speeds up the creation of digital work instructions by some industrial validation cases, e.g., the assembly of a weaving machine.

1. Introduction

In industry 4.0 there is a strong trend towards automatization, better software support and data exchange in manufacturing processes. Digital work instructions and cost-effective assembly sequences can enable to cope with challenges in low-volume, high-variety production processes. Providing intuitive instructions in a digital format to the operators on the work floor is also a promising method to decrease training time with new variants. However, there is a lack of authoring tools to support process engineers (i.e. person responsible for creating work instructions in a factory) in selecting an adequate assembly sequent and to create the assembly instructions for operators in a fast way.

There exists a wide selection of authoring tools available on the market. Companies like Dassault Systèmes, SAP, VISCOPIC Steps, VKS, ... have all developed their own software to create illustrations that can be added to the assembly's technical documentation. However, these tools are time consuming to use and offer little support in selecting the assembly sequence: to add a new step to the scene, the user needs to indicate manually a component. Sequentially, the user adds arrows and annotations to the scene to highlight how the component needs to be inserted in the assembly. This process is repeated for each step of the assembly process, whereby the assembly in the scene is gradually being build. When the assembly sequence is finalized, it is exported as a set of images.

This approach places a strain on the process engineer, who has to be acquainted of feasible assembly sequences and needs to manually create the visual content of each step of the sequence. A (semi-)automatic determination of assembly sequence from a CAD model and (semi-)automatic creation of the corresponding digital instruction of each step can significantly speed-up this process.

1.1. Previous work

Deriving assembly instructions automatically from CAD models is something that is highly sought after in the manufacturing industry, judging from the numerous previous attempts at solving the problem. For a comprehensive survey of these attempts, we refer the reader to the review paper of Raju Bahubalendruni and Biswal (2016). We will, however, lift out a subset of previous literature to position our work.

* Corresponding author.
E-mail address: dorothy.gors@flandersmake.be (D. Gors).

https://doi.org/10.1016/j.procir.2020.05.202
2212-8271/© 2020 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/)

A significant portion of literature focuses on the presentation of the instructions to the end user (Feiner, 1985; Mackinlay, 1986; Rist et al., 1994; Seligmann and Feiner, 1991; Strothotte, 2012). The two most common forms of diagrams are structural diagrams and action diagrams. Although the methods discussed in this paper are not limited to either form, the generated instructions that are output by our tool mostly come in the form of action diagrams, which spatially separate the parts to be attached from the parts that are already attached and use guidelines (e.g. dotted lines) to indicate where the new parts attach to the earlier part.

Most of the earlier literature is centered on generating an optimal assembly sequence planning in robotics (Bourjault, 1984; Homem De Mello and Sanderson, 1991; Romney et al., 1995; Wilson, 1992; Wolter, 1991; Melckenbeeck et al., 2020). These techniques are largely based on calculating geometric features from the input CAD models and performing exhaustive searches for the optimal solutions in the assembly graphs. This involves high computational load, jeopardizing the deployment of these approaches for the large industrial products.

1.2. Proposed work principle and validation

In this paper we propose a procedure for semi-automatic generation of assembly instructions based from an underlying CAD model. The contribution is twofold:

- Firstly, a feasible assembly sequence is automatically derived from the geometric data using a visibility evaluation heuristic. Proposed algorithm allows to compute a suggestion for the assembly sequence in limited time, (about 80 min for 300 part assembly, with 20% manual work). Subsequently, this sequence can be altered by the process engineer, taking into account his expert knowledge.
- Secondly, the instructions can be exported as text, images, 3D content and animation, for use in an external viewing system to display them to operators.

Our approach also takes into account subassemblies, which are very common in assembly operations, allowing to perform a single action on a collection of parts instead of a part-by-part approach.

2. Assembly sequence extraction

2.1. Assembly sequence generation through reverse disassembly

Our approach assumes assembly instructions are equivalent to the disassembly steps: which are in turn assembly but in reverse. Disassembly has the advantage that a disassembly operation cannot block the disassembly of the remaining components, however one assembly operation can make another assembly operation unreachable. Therefore, it is easier to determine in which order the components can be removed from the assembly. Under the generalization that a part is removable from an assembly, if other components of the assembly do not obstruct the part's movement. The second assumption is that a components is removable if the visibility of a component is not obstructed by the remaining assembly, when looking from outside towards the assembly.

2.1.1. Visibility evaluation heuristic

To investigate a part's removability we evaluate its visibility in its six principal directions. The idea is to produce orthogonal renders of the part and see if any other part of the assembly blocks it in that direction. We check for blocking by first rendering the tested part in black and then, without clearing the color buffer, drawing all other geometry in grey. If the black pixel count has changed after drawing the other geometry over it, other components in the assembly were blocking the tested part. Fig. 1 illustrates the visibility evaluation to determine if the screw (part 1)

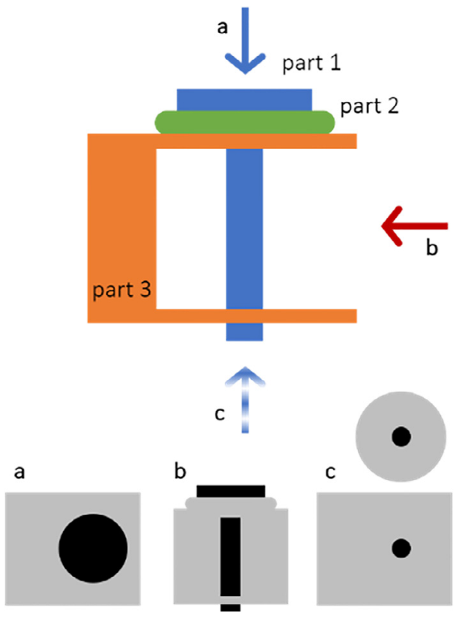


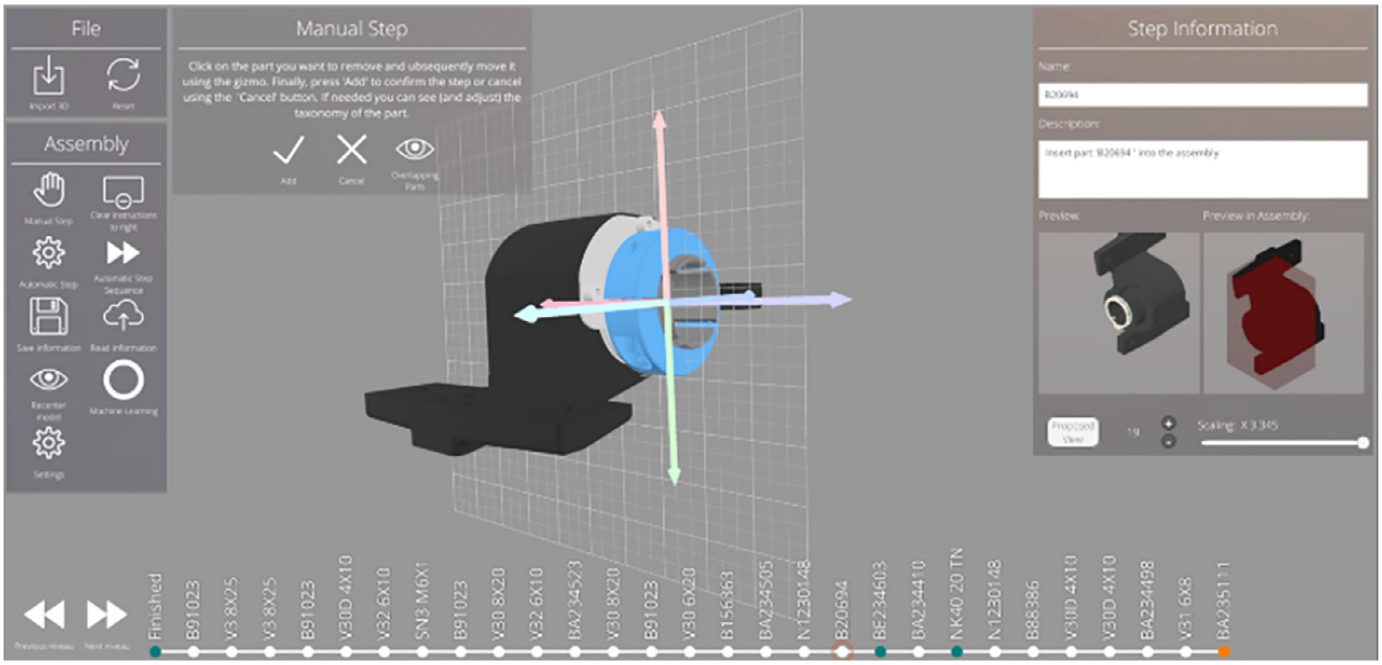
Fig. 1. Blue screw (part 1) is completely free to move upwards: (A) Part 2 and part 3 does not obstruct the visibility of part 1 from a viewpoint above the assembly and (C) part 1 does not obstruct the visibility of part 2 and part 3 from a viewpoint below the assembly. Blue screw is not free to move to the right: (B) Part 2 and part 3 obstruct the visibility of part 1 from a viewpoint right from the assembly.

can be removed from the assembly (part2 and part 3) in the upwards or in the rightwards direction. For the first test we look towards the screw in the direction indicated by the solid blue arrow (Fig. 1a) from a viewpoint outside the assembly bounding-box. The black pixel count isn't changed by overlaying the grey pixels comprising part 2 and part 3. However, when we look towards the screw from the right, as indicated by the solid red arrow (Fig. 1b), we see clearly that the grey pixels of part 2 and part 3 obstruct the visibility of the black pixels. When this visibility evaluation returns positive an additional test is performed: Now the viewpoint is changed, in order to look to the assembly from the opposite direction. In this case, according the dashed blue arrow (Fig. 1,c). Also the roles of tested part and remaining assembly are reversed. This to investigate if part 2 and part 3 can be removed from the screw in a downwards direction. In this example the black pixel count is not changed, so we can conclude that the screw can be removed in an upwards direction, but not a rightwards direction. Note, that those two tests are not flawless and false positives can still occur when there are internal blockages that are not visible from the outside, like e.g. retention hooks.

2.1.2. Improved heuristic by combining visibility with part similarity and proximity

The disassembly sequence is derived by evaluating the visibility of each of the remaining parts until a part is detected where the movement along one of its six principal directions is not obstructed by the remaining assembly. This part is then removed along the detected moving direction and a step is added to the disassembly sequence (See Algorithm 1). Doing so may alter the removability of the remaining parts, since previous obstructions are gradually removed. The disassembly sequence expands by removing parts one-by-one until no parts are left.

Using the visibility evaluation heuristic is not sufficient to obtain a useful assembly sequence. First, it is more convenient for an operator to build the assembly in an inside-out matter, this avoids situations where parts need to be inserted in difficult to reach places. Second, an operator will mount similar parts in close spatial



Click on the part you want to remove and ubsequently move it using the gizmo. Finally, press 'Add' to confirm the step or cancel using the 'Cancel' button. If needed you can see (and adjust) the taxonomy of the part.

Name: B20694

Description: Insert part 'B20694 ' into the assembly

Scaling: X 3.345

Fig. 2. UI of the tool. Left: menu with import/export options and choice between automatic or manual instruction generation for the new step. Below: timeline of the assembly sequence, allowing to navigate between steps and different subassembly levels. Right: Preview of the generated instruction and fields to add extra info. Middle: Manual step is selected, user can select a part and one of its 6 principal directions to insert a new step.

Algorithm 1
Process of selecting the next assembly part to be removed.

```
function GetRemoval(assemblyState)
  parts ← assemblyState.remainingParts
  SortByDistance(parts, previousRemoval, assemblyCenter)
  for all parts ∈ parts do
    for all direction ∈ part.principalDirections do
      if EvaluateVisibility(part, direction) then
        return part, direction
  return null
```

proximity (e.g. set of screws) in sequential steps. Thereby, keeping orientation changes, and operator relocation to a minimum, thus reducing production time and costs.

To integrate those two basic rules, the order in which the remaining parts are evaluated at each step is continuously adjusted based on part similarity and proximity. These assembly rules are translated into the following three disassembly criteria: (1) Parts closest to the previous removed part, (2) with similar bounding-box dimensions are given priority, as well as (3) parts furthest from the assembly's center. This lead to minimizing the following distance measurement:

$$d = 0.5 \cdot d_1 + 0.25 \cdot d_2 + 0.25 \cdot d_3 \quad \text{if tested part is no plate,}$$
$$d = 0.5 \cdot d_1 \quad \text{otherwise.}$$
$$d_1 = \left\| P_j - P_i \right\| / d_A, \quad d_3 = \left(0.5 d_A - \left\| P_j - C_A \right\|\right) / 0.5 d_A \quad \text{and}$$
$$d_2 = \frac{1}{3}\left(min\left(\frac{|S_{j,x} - S_{i,x}|}{S_{i,x}}, 1 \right) + min\left(\frac{|S_{j,y} - S_{i,y}|}{S_{i,y}}, 1 \right) + min\left(\frac{|S_{j,z} - S_{i,z}|}{S_{i,z}}, 1 \right) \right)$$

with P_j, P_i and C_A the position of the center of the bounding-boxes of the tested part, the previous removed part and remaining assembly bounding-box, respectively and d_A, $S_{j,x}$ and $S_{i,x}$ the diagonal length of the remaining assembly bounding-box, the length (in x-direction) of the bounding-box of the tested part and previous removed part, respectively. A part is considered a plate if its length along one of its principal axes is ten times smaller than along another principal axis. Considering plate-like parts prevents unstable situations where plates are assembled without any tightening screw. An additional benefit of defining the part similarity measurement (d_2) like this, is that disassembling smaller parts are given priority over larger parts in the disassembly order. This is desired, since during the assembly, it is more intuitive to start the assembly with larger and end with smaller parts.

2.2. Limitations and user-friendly UI for manual interactions

In some situations the algorithm fails to detect removable parts. Under specific conditions, it is possible that the rendering of the CAD model causes a reduction of a part's visibility, even if the removal of the parts remains feasible:

- CAD often comprises their elastic deformable parts, e.g. springs, in an expanded state. This causes physical penetration of the elastic parts and adjacent parts. Other critical parts are cables, wires and circlips.
- CAD can contain imperfections, e.g. bolts not in line and center to the screw-whole, which also causes physical penetration.
- The assembly can have overhanging structures, causing the reachability of some parts to be reduced, but not made impossible. However, the part visibility is reduced by the overhanging structure when using viewpoints outside the assembly bounding-box.

Those shortcoming are inherent to the CAD design. It is inevitable that our automatic algorithm will stumble on those issues, causing the process to end prematurely, since no removable parts can be automatically detected. To overcome this problem we introduce manual interaction.

When the algorithm is not able to automatically recognize feasible actions or tries to automatically add an unfeasible action, the user can perform a manual correction. Such a manual interaction consists of choosing an axis for the part to remove, or choosing another part to remove altogether. Afterwards, the automatic algorithm can continue. Removing one (or a few) of those CAD inherent troublous parts is often enough to clear the blockage, after which the automatic algorithm can again initialize.

An important aspect of the development of the tool is the integration of an intuitive UI, this to reduce the burden on the process engineer as much as possible. An assisting feature of the proposed authoring tool is its timeline. That is, all the disassembly steps are linearly ordered on a timeline at the bottom of the screen (Fig. 2). If the user is unhappy with the manner in which a certain step is handled, he or she can simply go back to that point and perform a manual correction. The steps that come after the correction are either retained or erased and regenerated back with the automatic algorithm or with further manual interactions, according the user's insights.

Moreover, the user can inserts additional steps in between the generated sequences and alternate sequence order without influencing the steps already defined on the consecutive part of the timeline.

Also, the process to insert a manual step is made simple, requiring little input of the user. Through the UI, the user can easily inspect the 3D content of the remaining assembly and select a part that he or she wants to remove. Subsequently, the tool displays the part's 6 principal axes, on which the user can click on the axis of his choice (Fig. 2).

Another limitation is that the integrity of the CAD model is already affected by (and even before) importing it in our tool. Conversion between different CAD design software's and the import of large models with PiXYZ[1] (i.e. CAD import software) may induce approximation errors and result in loss of the geometric detail of the model. Consequently, negatively influencing the visibility renders. This further increases the need for manual interactions.

2.3. Working with subassemblies

The described procedure is linear, meaning that in each disassembly step only one component is removed from the assembly. Or reciprocally, that the operator is instructed to fasten one part at the time. This, however, does not represent the real practice at the work floor. Commonly, subassemblies are preassembled in other work cells and are afterwards mounted as a whole on the total assembly. Furthermore, the repetition of redundant instruction (e.g. sequentially screwing of a set of screws) will annoy the operator more than being informative.

Decomposing subassembly operations as a linear set of single-part operations has also an adverse effect on the working of our tool:

- Disassembly of a subassembly can be blocked by other components in the total assembly. This decreases the automatically detection of removable parts.
- Disassembly steps of a subassembly can be automatically distributed over the disassembly sequence, instead of being grouped together.
- The same issue for steps concerning similar adjacent parts.

An attribute of our procedure is, however, that it considers the subassembly structure defined within the CAD model. This structure is used to create a tree-hierarchy. Each node in the tree represents a subassembly that is constructed by assembling the parts found in its children. These children can other subassemblies or individual parts that do not need to be further assembled. This leads to a non-linear timeline, as now every non-leaf node in the tree has an assembly sequence.

When generating the disassembly sequence, we start at the top level of the hierarchy. Picking which part to extract is done in the same manner as described in Section 2.2. To achieve this, we consider subassemblies comprising of multiple parts as one part, to be removed entirely. This process is repeated while moving deeper into the hierarchy, until a subassembly is reached whose parts are all individual parts. This leads to a set of assembly sequences for the operators.

The subassembly handling is a powerful contribution to the tool for the creation of practical work sequences, however it is highly depended on the meaningfulness of the subassembly structure defined within the CAD model. A designer need to consider following rules:

- All parts (and only the parts) of a subassembly needs to be grouped together in the structure.
- To group repetitive actions on similar parts, those parts needs to be grouped, as if they are a subassembly.
- The subassembly needs to be removable from one single direction. E.g. all components needs to translate in the same direction to move away from the assembly without causing any collision with the assembly.

To avoid to be completely reliant on the CAD model structure, we added the possibility to circumvent predefined structure when manually inserting steps. The user can then decompose improperly defined subassemblies or combine parts into subassemblies, allowing him or her to remove a subset of parts in a single step.

3. Digital work instructions

Each step on the timeline is associated with information for that assembly step. Each step has a name, description and preview image (left preview image on the right panel of Fig. 2) that shows how the final work instruction will look like. Those fields are automatically filled with the name extracted from CAD and a standard text "Insert part in the assembly". The user can manually enter additional information in the description-field (e.g. which tools to use).

Clear visibility of the manipulated part during the assembly step at hand is crucial to achieve high quality work instructions. From one of the 25 selected viewpoints (all corner points and center of the ribs and faces of a bounding-box surrounding the selected part) our algorithm selects the one that maximizes this visibility. This visibility optimization is based on the same orthogonal rendering technique used to determine a part's removability. Under the assumption that on good illustrations the visible area of the manipulated part is maximal and the insertion trajectory clearly displayed, the optimization problem is defined as follow: determine at which viewpoint both the pixel count of the part (both inserted, as translated over its length to be fully free of the assembly) and the length of the orthogonal projection of this trajectory are maximized.(While keeping the distance between part and viewpoint for all 25 test equal.) Additionally, viewpoints wherefore the trajectory moves away from the viewpoints are discarded.

A nice side-effect of the sequence extraction is that if parts that can be removed in the same direction those actions can often be displayed from the same viewpoint as well. Thus our method enables us to very efficiently generate a sequence of assembly instructions with viewpoints that are naturally close to each other.

At each step on the assembly sequence timeline, the tool shows the preview image of the selected viewpoint. If the user is not satisfied with this proposal, he or she can alter both viewpoint and projection focus with sliders on the right panel of the tool's interface.

Once the user is satisfied with the sequence of assembly instructions, he or she can export the instructions in four formats:

- Text: Instruction sequence, including information like the name, description and additional geometrical parameters (e.g. insert direction, insert distance and final position of the part).
- Image: Illustration sequence (Fig. 4).

[1] PiXYZ STUDIO is an CAD data preparation & optimization software. It helps companies and 3D consumers re-use CAD data for any visualization scenario.

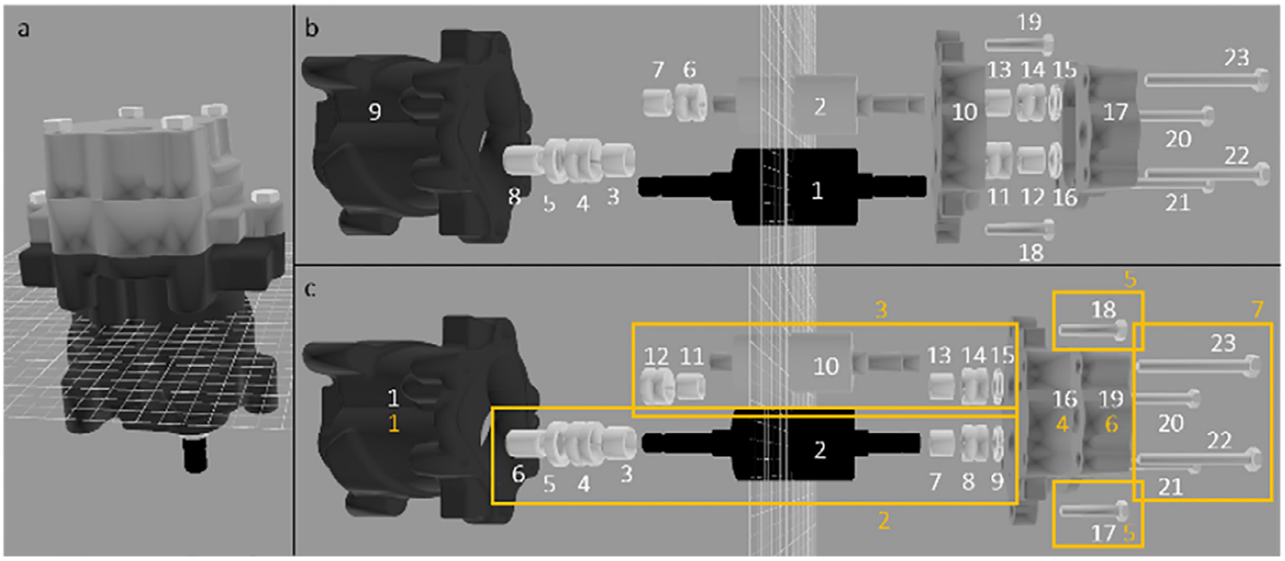


Fig. 3. Assembly of the Atlas Copco compressor. (A) The compressor. (B) Automatic derived sequence. (C) Manual inserted sequence (white) and example of properly defined subassemblies (orange).

- 3D models: Sequence of CADs of the added part and current state of the assembly.
- Animation: Sequence of movies animating the movement of the added part towards the assembly at each state.

4. Validation

The above explained procedure has been implemented in C#, within the Unity[2] platform. Unity allows us to create an intuitive graphical user interface and, together with the PiXYZ plugin, to interact with 3D models. In this paper, we illustrate the tool's performances and constrains on two industrial assemblies: The first, an Atlas Copco compressor air-end (Fig. 3a), is a simple straightforward assembly, while the second, a part of the weaving machine of Picanol (Fig. 5, top), is a more extensive and complex assembly.

The compressor comprises 23 parts, that can all be inserted with part manipulations along the same axis, as illustrated in Fig. 3. For this validation we let the tool create a fully automatically assembly sequence. Within 8 s, the sequence depicted in Fig. 3(b) was generated. As reference, we created completely manually a feasible assembly sequence. After 1.5 min of selecting parts and directions, we defined the sequence depicted in Fig. 3(c). The main difference between the automatically derived and manually inserted sequences locates with the two rotors, their bearings and the main housing. Following the automatic instruction the operator needs to hold the rotors and bearings in an unstable configurations, before the main housing is inserted over this combination. Furthermore, it is more practical to first slide the inner bearings onto the rotors, before sliding the outer bearing on top of them, in contrast to what the automatic instruction suggest. The distance measurement defined for the sequence order is not well defined to considering stability aspects. Even manually, it is not possible to define the optimal sequence, this due to the wrongly defined or, in this case, absence of subassembly structure. However, the tool allows us to manually combine multiple parts and manipulate them as a subassembly, this would result in a sequence depicted in orange in Fig. 3.

An assembly, more representative for industrial applications, is the Picanol's weaving machine (Fig. 5). This assembly comprises

[2] Unity is a cross-platform game engine developed by Unity Technologies.

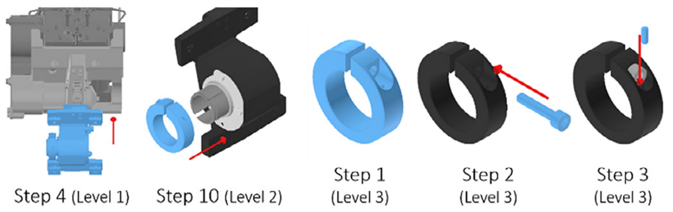


Fig. 4. Visualization of the work instructions for a Picanol weaving machine. Illustrates the sequential actions on different levels of the assembly. Same use-case as in Fig. 5.

384 components, that are grouped in 67 subassemblies going 5 levels deep. For this second experiment, we let again the tool generate automatically thee assembly sequence with automatically proposed instruction illustrations. However, for this complex product, manually interactions are needed, as the next disassembly step is not always automatically detected. We evaluate the semi-automatically derived assembly sequence on the feasibility of the assembly steps and the clarity of the instruction illustrations.

After completing the disassembly procedure, we obtain a semi-automatically derived sequence comprising 451 steps, from which 98 where manually inserted. Situations where manual interference where needed, have already been listed in Section 2.2. Fig. 5 shows some intermediate explosion graphs (assisting the process engineer when working with the tool) and Fig. 4 some of the corresponding work instructions (that the operator will use on the work floor).

To evaluate the feasibility of the assembly sequence the following conditions are used:

- When using straight insertion movements, no parts shall collide with the already assembled assembly.
- It is not allowed to have non-attached (e.g. floating parts) parts during the assembly process.
- After placing a plate on the assembly, the tightening of the screws should be the subsequent step.

With those criteria we detected 41 invalid steps (out of the 451 steps). The majority of those, where violating the first condition, due to improperly CAD design or subassembly definition. Note that those conditions does not consider the assembly's stability or the cost-effectiveness of the task.

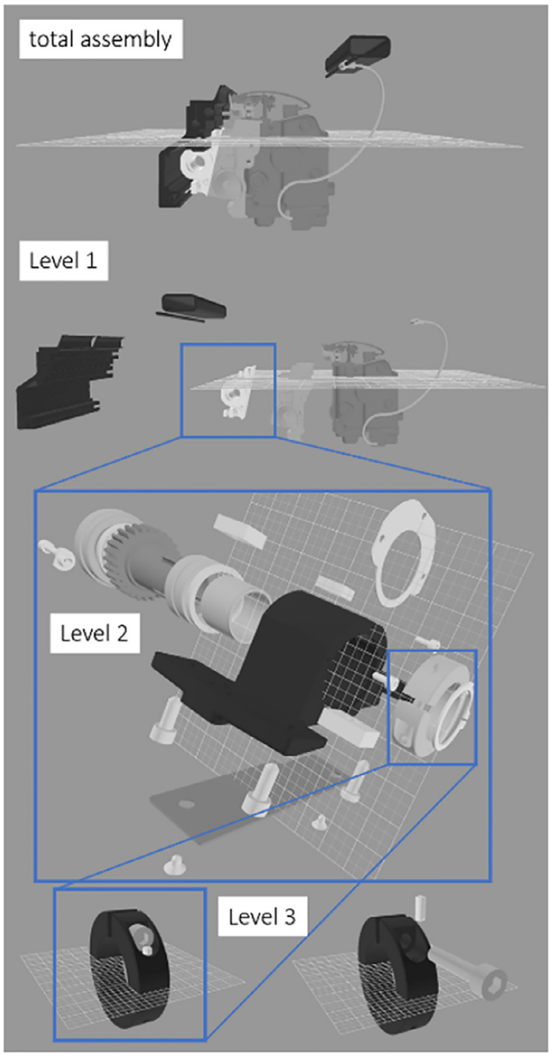


Fig. 5. Assembly of a Picanol weaving machine. Collage of different screenshot of the tool. Shows the sequential actions on different levels of the assembly. Corresponding to Fig. 4.

When inspecting the automatically proposed 2D visualization of each step, we found 68 instructions that did not show a clear view of the required assembly actions. In those cases, either the insertion motion or the final destination of the part was not clear visible, due to visual coverage by the assembly. This indicates that the visibility evaluation to determine the optimal viewpoint can be improved.

To conclude; The practicality of the automatic derived sequence and the need for manual interactions are mainly (negatively) influenced by CAD design and structure. In addition, 85% of the proposed illustrations are useful for the operator. Therefore, we have demonstrated that the tool has a high added value for the process engineer. Especially, since the manual corrections, when the automatic procedure should fail, are easily inserted.

5. Conclusion and future work

This paper presented a tool to support process engineers to generate digital work instructions for assembly tasks. It includes a heuristic algorithm based on visibility of parts to synthetizes semi-automatically the assembly sequence from a CAD-model and generates corresponding visualizations of the assembly action to include them in the instructions.

The intuitive UI allows the process engineer to modify the generated instructions according their own insights and needs.

The tool has been validated for several industrial products. Apart from some minor issues the results of the digital instructions generation were very satisfying and recognized as very helpful to reduce the burden to create the assembly instructions.

Directions for future work include exploration of different techniques for generating the assembly sequence, such as constraint programming facilitating to trade-off different objective e.g. cost, collision detection or precedence constraints. Also, handling flexible parts such as seals, springs, cables, etc. remains a challenge.

Acknowledgment

This research is supported by VLAIO (Flanders Innovation & Entrepreneurship) Flanders Make, the strategic center for the manufacturing industry in Flanders, within the framework of the OperatorKnowledge ICON-project (HBC.2017.0395). We like to thank Jens Brulmans, Niels Billen, Robbe Cools and Georges Verpoorten for their contributions to the implementation, and our partners Arkite, Augnition, Barco, CNHi, Atlas Copco, ASCO and Picanol for providing test data and context for this work.

References

Bourjault, A., 1984. Contribution à Une Approche Méthodologique de L'assemblage Automatisé: Élaboration Automatique des Séquences Opératoires. Université de Franche-Comté.
Feiner, S., 1985. APEX: an experiment in the automated creation of pictorial explanations. IEEE Comput. Graph. Appl. 5 (11), 29–37 1985.
Homem De Mello, L.S., Sanderson, A.C, 1991. A correct and complete algorithm for the generation of mechanical assembly sequences. IEEE Trans. Robot. Autom. 7 (2), 228–240 1991.
Mackinlay, J., 1986. Automating the design of graphical presentations of relational information. ACM Trans. Graph. 5 (2), 110–141 1986.
Melckenbeeck, I., Burggraeve, S., Doninck, B.V., et al., 2020. Optimal assembly sequence based on design for assembly (DFA) rules. In: Proceedings of the 30th CIRP Design 2020.
Raju Bahubalendruni, M.V.A., Biswal, B.B., 2016. A review on assembly sequence generation and its automation. Proc. Inst. Mech. Eng. Part C: J. Mech. Eng. Sci. 230 (5), 824–838 2016.
Rist, T., Krüger, A., Schneider, G., Zimmermann, D., 1994. AWI: a workbench for semi-automated illustration design. In: Proceedings of the Workshop on Advanced Visual Interaces. ACM, pp. 59–68.
Romney, B., Godard, C., Goldwasser, M., Ramkumar, G., 1995. An efficient system for geometric assembly sequence generation and evaluation. In: Proceedings of the 15th ASME International Computers in Engineering Conference (CIE), Boston, Massachusetts. Comput. Eng. 699–712 1995.
Seligmann, D.D., Feiner, S., 1991. Automated generation of intent-based 3D illustrations. In: Proceedings of ACM SIGGRAPH Computer Graphics, 25. ACM, pp. 123–132.
Strothotte, T., 2012. Computational Visualization: Graphics, Abstraction and Interactivity. Springer Science & Business Media.
Wilson, R.H., 1992. On Geometric Assembly Planning. Stanford Univ CA Dept of Computer Science Technical Report.
Wolter, J.D., 1991. On the automatic generation of assembly plans. In: Computer-aided Mechanical Assembly Planning. Springer, pp. 263–288.