

Task-Level Assembly Modeling in Virtual Environments

Bernhard Jung

University of Bielefeld, Germany
Faculty of Technology, Knowledge-Based Systems Group
Laboratory for Artificial Intelligence and Virtual Reality

Abstract. This contribution introduces a new framework for assembly modeling in Virtual Reality. Aiming at an easy instructability of the virtual environment, the framework provides a task-level interface which directly maps logical assembly commands to corresponding changes in the geometry scene. For example, the visual assembly of two parts is achieved given only a single command 'connect(a,b)'. This is in contrast to the assembly modeling style of conventional CAD systems which forces the designer to break down each conceptual assembly task into a series of lower-level subtasks. The proposed framework consists of two parts: (1) A knowledge-based model of connection-sensitive part features ("ports") and the connections between them; and, (2), a set of algorithms that define the task-level interface for assembly, disassembly, and adjustment operations. All algorithms are computationally efficient and easily meet the real-time requirements of virtual environments. At the user interface, both direct manipulation and directive interfaces, e.g. based on natural language instructions are supported. A family of implemented VR-systems, including CAVE and Internet-based applications, demonstrates the feasibility of the approach.

1 Introduction

One of the commercially most relevant application areas of Virtual Reality (VR) are CAD-tasks that involve the design and evaluation of so-called virtual prototypes. A main goal is to enable the testing of different aspects of part design and behavior without the need to manufacture a physical prototype first [1, 9]. Whereas in most of today's industrial applications, VR is still used as mere visualization tool for prototypes modeled in external CAD systems, recent years have seen an intensified research interest in *immersive assembly modelers* which could ultimately enable the design of virtual prototypes from within the virtual environment. Our main interest are immersive assembly modelers that not only provide powerful modeling functionalities but also – and in particular – can be easily instructed, e.g. by means of intuitive language- and gesture-based instructions. Figure 1 shows examples of the kinds of virtual environments considered.

While conventional CAD systems provide powerful assembly modeling functionalities attempting to transfer the CAD-based assembly modeling style to VR



Fig. 1. Virtual environments involving the assembly of CAD-based parts.

seems rather misguided. Two of the CAD systems' inadequacies concern their insufficient real-time and collision handling capabilities. For the present discussion, a further aspect is central: In CAD systems, the conceptually atomic process of adding a part to an assembly is modeled by the successive definition of geometric relations (constraints) between part surfaces. For example, to insert a screw into a nut usually three constraints such as **concentric**, **coincident**, and **parallel** need to be defined. Thus, when adding a part to an assembly, the cognitive load is on the user who has to mentally break down the task into a series of lower-level constraint definitions. Moreover, the constraints can be defined between arbitrary surfaces of the parts (where typical industrial models are easily composed of hundreds of surfaces). This surface selection process increases the mental load of the user even more. In desktop-based CAD systems, an assembly relation browser helps the user to understand the structural constraints of the assembly. In VR, in contrast, the internal structure of the assemblies is usually hidden from the user. This, in turn, requires the availability of alternative, higher-level assembly modeling methods.

The remainder of this contribution, after reviewing related work, introduces such an alternative approach to assembly modeling based on the notion of *task-level control* [22]. Building on a knowledge-based model of part connectivity, these methods achieve e.g. the visual assembly of two parts in the virtual environment given only a single command *connect(a,b)*. Besides assembly operations, similar methods for disassembly and adjustments of assemblies are supported. The main goal of the proposed framework is, however, not so much a full reconstruction of the powerful CAD-based modeling functionalities in VR (although it is in fact more powerful than many of the existing immersive assembly modelers); instead, it's main purpose is to support the easy instructability of VR-based virtual prototyping environments.

2 Related Work

VR systems labeled in the literature as “virtual assembly” can be roughly divided into two categories (of which only the latter provides modeling functionalities).

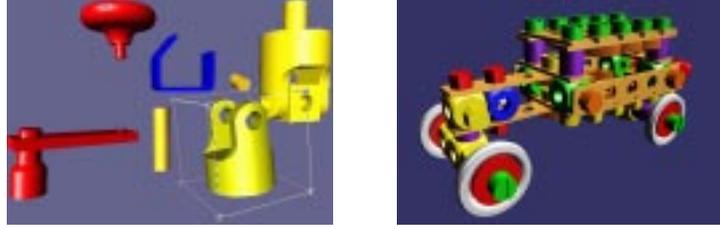


Fig. 2. Left: Direct manipulation based desktop user interface for task-level assembly. Right: A car model built from 124 parts.

A first kind of virtual assembly systems enables a VR-based assembly, typically using one or two data gloves, of prototypes defined in external modelers. The purpose of such systems is e.g. to check the general (dis-)assemblability of the design, part accessibility, tool usage, generation of sequences and trajectories of assembly operations, and so on. Examples include the VADE system [10], Zachmann's work with Virtual Design II [6], and applications of DBView developed at DaimlerChrysler [4]. Empirical studies confirm the usefulness of VR-based systems for assembly planning and verification as compared to traditional blueprints or nonimmersive desktop systems [2].

The second kind of virtual assembly systems, immersive assembly modelers, additionally supports the combination of CAD-based parts to novel assemblies. Typically, such systems implement some sort of snapping mechanism which automatically completes an assembly operation when two parts are moved close enough in the virtual environment. Some systems are restricted to assemblies composed of a few types of primitives, e.g. boxes or cylinders [5] or Lego bricks [13]; more recently, the SeamlessDesign system used a larger set of more complex primitives [14]. At Volkswagen, an immersive VR system for interactive cable layout has been developed [20]. Other systems explicitly model part connectivity by means of simple reference (snap) points on the object's surfaces [8]. The approach of [7] uses an automatic alignment based on the objects' bounding boxes for snapping two parts together. ISAAC implements snap-to-grid and snap-to-orientation methods for precision placement of parts [17]. The MAESTRO system uses "sensitive polygons" to constrain the objects' movement during the coarse positioning phase [19]. Another class of immersive assembly modelers uses constraint-based approaches very similar to those of modern CAD systems, e.g. ICBAM [21] or the constrained virtual environment developed in Salford [16]; here, snapping can be implemented by a combination of automatic detection and assertion of geometric constraints. Snapping addresses some of the issues raised in the introduction concerning the transfer of CAD-style assembly modeling to VR; however, as the constraint detection mechanism constantly has to check for spatial relations between arbitrary part surfaces, such systems appear to be restricted to assemblies consisting of a small number of industrial-complexity parts. It is further unclear, how conflicts between asserted constraints and object

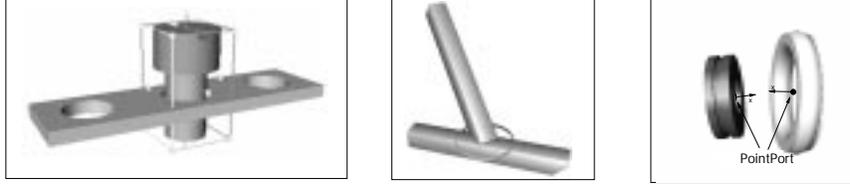


Fig. 3. Extrusion, Plane, and Point port connections.

collisions could be resolved. In all of the above systems, the interactive assembly simulation is triggered by direct manipulation, e.g. using one or two datagloves.

Our approach to immersive assembly modeling aims to overcome some of the limitations of existing systems by combining the following properties: (1) it supports not only assembly and disassembly but also adjustments in the relative positioning of parts; (2) it scales well for assemblies consisting of a large number (e.g. in the order of hundreds) of parts; and, (3) it supports not only direct manipulation, but also natural language instructions. The last aspect, and even more so the combination of these features makes our approach unique.

3 A Knowledge-Based Model of Ports and Connections

The proposed framework for task-level assembly modeling utilizes a knowledge-based model of connection-sensitive part subvolumes or *ports*, e.g. the shaft of a screw or the inside thread of a nut, and constraints imposed by different kinds of *connections*, e.g. screwed, inserted, or welded. Taxonomies with predefined port and connections concepts capture the most common mating properties of CAD-based parts which facilitates the reuse of once developed models.

The *port* taxonomy is organized around geometric and mechanical properties which differentiate the port concepts at the higher and, resp., lower levels of the taxonomy. At the top-level, the taxonomy is divided into the following concepts:

- *Extrusion ports* model connection properties of object subvolumes with extrusion geometries. For example, extrusion ports model connection properties of objects involved in peg-in-hole type assembly operations. In general, connections between extrusion ports afford one translational and one rotational degree of freedom. Extrusion ports are differentiated into *male ports*, such as the shaft of a screw and *female ports*, e.g. the hole of a nut. Matings between two extrusion ports generally involve one male port and one female port. For two extrusion ports to be connected, they must be collinear, parallel (in same or opposed direction), and their capacities must overlap.
- *Plane ports* model connection properties of planar object surfaces. Connections between two plane ports afford two translational and one rotational degree of freedom. For two plane ports to be connected, they must be coplanar, anti-parallel, and their capacities must overlap.

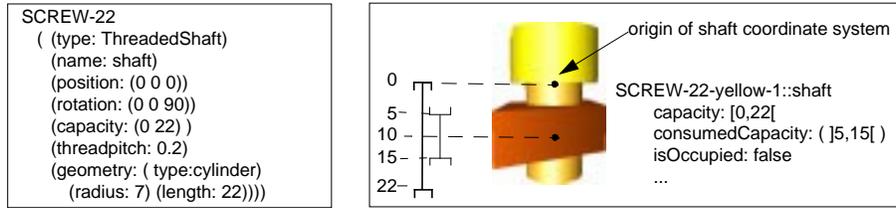


Fig. 4. Left: Port definition for a screw with a 22mm shaft. Right: In concrete assembly tasks, among other dynamic attributes, a list of consumed capacities is maintained.

- *Point ports* model point-like object connections that induce no translational and one rotational degree of freedom when objects are connected. Connected point ports are coincident and parallel. Point ports can be understood as borderline cases of both extrusion and plane ports but introduce a separate concept that has conceptual and computational advantages.

The port taxonomy currently consists of 20+ concepts; it is complemented by a taxonomy of connection concepts that define further constraints concerning the relative movement of connected objects. E.g. screw-in-hole insertions can be defined with either independent or dependent rotational and translational degrees of freedom (in the latter case the screw needs to be rotated in order to be inserted into the hole). Or, a welding-type of connection can be defined as one that allows no transformational degree of freedom at all. The connection taxonomy mirrors a taxonomy developed by Roth in the field of mechanical engineering [18].

Figure 4, left, shows how the CAD model of a screw is enriched with port information. The port definition makes reference to the port taxonomy – the screw’s shaft thus inherits e.g. the property to take part in screwing-type operations – and defines additional part-specific, dimensional information. During an on-going assembly task, among other dynamic attributes, a list of consumed capacities is maintained for each port. The consumed capacity of a port is used e.g. by the assembly algorithms to prevent matings with third parts at those positions. A detailed description of the port and connection representations can be found in [11].

4 Algorithms for Task-Level Assembly Modeling

Aiming at a task-level instructability of virtual prototyping environments, a comprehensive set of algorithms has been developed that transforms high-level specifications of various assembly-related tasks to corresponding changes of the virtual environment’s lower-level graphics representation. In particular, these algorithms accomplish the assembly and disassembly of parts and aggregates as well as adjustments of already connected parts along their remaining degrees of freedom. Building on the port and connection representations described above,

connect(a,b)

1. Heuristically refine the input data, i.e. select ports, hotspots, assembly direction (if not already present in input descriptions) based on current snap policy
2. Perform preliminary mate of the two parts, account for already consumed capacities of ports being mated
3. Handle collisions possibly resulting from step 2
4. Infer the new connection relationships and assert them in the connection graph

Fig. 5. The connect algorithm for task-level assembly.

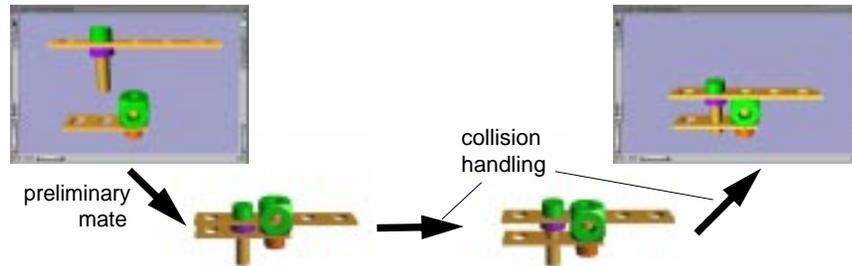


Fig. 6. Phases of the connect algorithm.

the algorithms operate on and manipulate several part, port, and connection representations at once. The algorithms further incorporate polygon-precision collision handling methods and thus ensure physical realism to a large extent. Due to space limitations, the present discussion is focussed on the algorithm for task-level assembly. A more detailed pseudo code description of all algorithms can be found in [11].

The connect algorithm achieving a task-level assembly is shown in Figure 5. A first and important issue concerns the data structures of its input arguments. Just like the other task-level algorithms, the connect algorithm expects only information as input that can be derived from natural language (NL) instructions. NL instructions, however, can specify the task at very different levels of granularity. E.g. in the assembly task shown in Figure 6, the objects to be connected could have been described as individual parts or as aggregates (*insert the screw into the bar* vs. *insert the upper aggregate into the lower one*); or, a specific port may have been specified (*...into the middle hole of the bar*); further, an assembly direction (*...from above*) or the amount of the insertion (*insert the screw fully* ...) may have been provided. Correspondingly, the connect algorithm is called with parameters of a flexible data-structure. At a minimum, the set of involved parts must be specified. Optionally, further fields describing e.g. a specific port or a hotspot (snap point) on that port can be supplied. In step 1 of the algorithm,

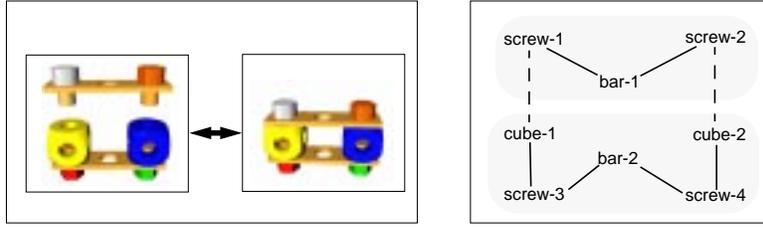


Fig. 7. Each assembly operation may establish multiple connections; in the connection graph, two components are merged. Likewise, each disassembly operation may require the resolution of multiple connections; in the connection graph, the component representing the original aggregate is split in two components.

the unfilled fields of the two input parameters are heuristically filled, based on a globally defined *snap-policy*. In Figure 6, e.g., a snap-policy is assumed that tries to mate the closest ports, in an assembly direction that matches best the moved objects' current orientation, and with a preference to insert extrusion ports as far as possible into another.

Steps 2 and 3 of the connect algorithm are illustrated in Figure 6. In step 2, the parts are preliminarily mated based on the locally available information about consumed port capacities. While the mating conditions are locally satisfied for the two ports, a collision occurs between two other parts in the assembly. Therefore, step 3 performs a collision avoidance procedure. The screw is pulled out (in inverse assembly direction) of the bar until a collision-free state is reached. Collision avoidance operates at polygonal precision by utilizing detailed geometry information about the CAD-based parts. The final step 4 of the connect algorithm tests the resulting assembly for new connection relations and asserts them in the *connection graph*. Figure 7 shows an example, where multiple connections result from one assembly operation.

Figure 7 can also be read as an example of an disassembly operation. Just as a single assembly task may result in the establishment of multiple new connections, a single disassembly task may involve the breakup of several connection relationships. Put slightly differently: While assembly operations merge two components of the connection graph into one component, disassembly operations split one component of the connection graph into two. Accordingly, the disconnect algorithm, shown in Figure 8, first splits the connection graph into two components, and then fully separates the two components in the graphics scene. Collision avoidance ensures that the disassembled parts do not penetrate any other objects in the virtual environments.

Finally, besides assembly and disassembly operations, the present framework for task-level assembly modeling further supports adjustment operations that modify the relative placement of some parts in an assembly w.r.t. other, stationary parts. Both rotational and translational adjustments are supported, corresponding to the degrees of freedom of port connections; see Figure 9 for an

`disconnect(a, b, dirhint)`

1. Split the connection graph of the aggregate containing *a* and *b* into exactly two components *a'* and *b'* by retracting one or more connection relations. All retracted connection relations must support the same disassembly direction *dir*, which must approximately match the optional *dirhint*, if provided.
2. Graphically separate the parts *a'* from *b'* in direction *dir*, such that their distance exceeds a threshold and the removed parts *a'* do not collide with any other parts in the environment.

Fig. 8. Disconnect algorithm for task-level disassembly.

example of a translational adjustment. The adjustment algorithms are similar to the disassembly algorithm in that they first (although temporarily) split the connection graph of the modified aggregate into two components of the transformed and stationary parts. The following graphical displacement, however, is limited such that in the resulting state the mating conditions of the modified connections still hold and no parts are separated from the assembly.

5 Conclusions

We have presented a novel framework for task-level control of virtual environments for assembly modeling. As representational foundation, a knowledge-based model of ports and connections was devised that captures stereotypical mating properties of CAD-based parts. Compared to conventional lower-level CAD descriptions, port and connection representations provide a more abstract modeling means closer to the human conceptualization. Building on the port and connection representations, a set of algorithms was developed that accomplishes a task-level control of various assembly related operations such as assembly, disassembly, and adjustments. Desirable properties of the algorithms include their real-time capabilities and built-in collision avoidance. Further, the algorithms



Fig. 9. Translational adjustment by direct manipulation: The selected part is moved, constrained by the mating conditions of its connection to another part.

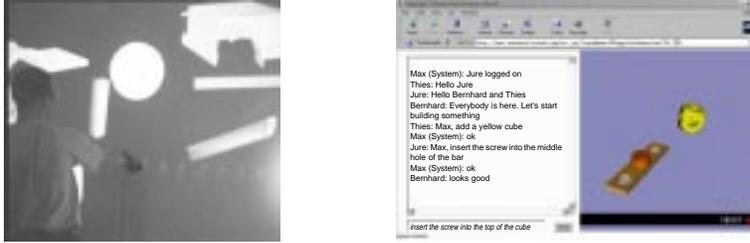


Fig. 10. Left: Immersive assembly modeling in a large-screen environment using speech and gesture. Right: A web-browser interface based on typed natural language.

can accommodate a variety of information present in verbal (or gestural) instructions of the user. They are, however, also able to perform the task when only minimal information is provided in the instruction.

The framework for task-level assembly modeling has been integrated into several complete virtual assembly systems. In the large screen environment shown on the left of Figure 10, natural language and gesture instructions enable an interaction with both close and distant parts [15]. In web-browser based environments, as the one shown on the right of Figure 10, the updates rates of the visualization are particularly slow; here, the natural language interface proves advantageous as it allows for instantaneous updates of the virtual environment as opposed to direct manipulations that require continuous updates [12]. Of course, as illustrated in Figures 2 and 9, the task-level interface to virtual assembly also supports user interfaces based on direct manipulation. Current work aims at extending the task-level interface to scaling operations that preserve the shape of certain subparts [3].

Acknowledgment. This work is partly supported by the Deutsche Forschungsgemeinschaft (DFG).

References

1. J. A. Adam. Virtual reality is for real. *IEEE Spectrum*, 30(10):22–29, 1993.
2. A. Banerjee, P. Banerjee, N. Ye, and F. Dech. Assembly planning effectiveness using virtual reality. *Presence*, 8(2):204–217, 1999.
3. P. Biermann, B. Jung, M. E. Latoschik, and I. Wachsmuth. Virtuelle Werkstatt: A plattform for multimodal assembly in VR. In *Proceedings Virtual Reality International Conference (VRIC 2002)*, pages 53–62, 2002.
4. M. Buck and E. Schömer. Interactive rigid body manipulation with obstacle contacts. In *Proceedings 6th International Conference in Central Europe on Computer Graphics and Visualization*, pages 49–56, 1998.
5. J. Butterworth, A. Davidson, S. Hensch, and T. M. Olano. 3DM: A three dimensional modeler using a head-mounted display. In *Proceedures of the Symposium on Interactive 3D Graphics*, pages 134–138. ACM Press, 1992.

6. A. G. de Sa and G. Zachmann. Virtual reality as a tool for verification of assembly and maintenance processes. *Computer Graphics*, 23(3):389–403, 1999.
7. P. Drews and M. Weyrich. Interactive functional evaluation in virtual prototyping illustrated by an example of a construction machine design. In *IECON-98 – Proc. 24th Annual Conference of the IEEE Industrial Electronics Society*, volume 4, pages 2146–2151. IEEE, 1998.
8. J. Gausemeier, M. Grafe, and R. Wortmann. Interactive planning of manufacturing systems with construction sets. In *IECON-98 – Proc. 24th Annual Conference of the IEEE Industrial Electronics Society*, volume 4, pages 2146–2151. IEEE, 1998.
9. R. Gupta, D. Whitney, and D. Zeltzer. Prototyping and design for assembly analysis using multimodal virtual environments. *Computer-Aided Design*, 29(8):585–597, 1997.
10. S. Jayaram, U. Jayaram, Y. Wang, K. Lyons, and P. Hart. VADE: A virtual assembly design environment. *IEEE Computer Graphics and Applications*, 19(6), 1999.
11. B. Jung. Intelligent virtual prototyping environments: Foundations and applications of task-level CAD in virtual reality. Habilitation Thesis, Faculty of Technology, University of Bielefeld, 2002.
12. B. Jung, T. Pfeiffer, and J. Zakotnik. Natural language based virtual prototyping on the web. In *Proceedings Structured Design of Virtual Environments and 3D-Components*. Aachen: Shaker, 2002.
13. K. Kiyokawa, H. Takemura, Y. Katayama, H. Iwasa, and N. Yokoya. Vlego: A simple two-handed modeling environment based on toy blocks. In *Proceedings ACM Symposium on Virtual Reality Software and Technology (VRST '96)*, 1996.
14. K. Kiyokawa, H. Takemura, and N. Yokoya. SeamlessDesign: A face-to-face collaborative virtual/augmented environment for rapid prototyping of geometrically constrained 3-d objects. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Vol. 2*, pages 447–453, 1999.
15. M. Latoschik, M. Fröhlich, B. Jung, and I. Wachsmuth. Utilize speech and gestures to realize natural interaction in a virtual environment. In *IECON'98 - Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, volume 4, pages 2028–2033, 1998.
16. L. Marcelino, N. Murray, and T. Fernando. A constraint manager to support virtual maintainability. In *1st Ibero-American Symposium in Computer Graphics*, 2002.
17. M. A. Milne. ISAAC: A meta-CAD system for virtual environments. *Computer Aided Design*, 29(8):547–553, 1997.
18. K. Roth. *Konstruieren mit Konstruktionskatalogen*, volume I. Springer-Verlag, Berlin, 2. edition, 1994.
19. R. Steffan and T. Kuhlen. Maestro - a tool for interactive assembly simulation in virtual environments. In B. Fröhlich, J. Deisinger, and H.-J. Bullinger, editors, *Immersive Projection Technology and Virtual Environments*, pages 141 – 152. Springer Verlag, 2001.
20. M. Symietz. *Echtzeitbasierte Generierung und Verlegung von Leitungsobjekten in einem digitalen Fahrzeugmodell mit einem Virtual-Reality-System*. PhD thesis, University of Bielefeld, Faculty of Technology, 2000.
21. M. R. Thompson, J. H. Maxfield, and P. M Dew. Interactive virtual prototyping. In *Proceedings of Eurographics UK '98*, pages 107–120, 1998.
22. D. Zeltzer and S. Gaffron. Task-level interaction with virtual environments and virtual actors. *International Journal of Human-Computer Interaction*, 8(1):73–94, 1996.