Automatic Decomposition of Planned Assembly Sequences Into Skill Primitives

Heiko Mosemann and Friedrich M. Wahl

***Abstract*—This paper presents a new method to decompose complex sequences of assembly operations into skill primitives. This can be realized by analyzing hyperarcs of the underlying AND/OR-graphs representing automatically generated assembly plans. Features like local depart spaces, symbolic spatial relations, and the necessary tools classify the type of assembly operation (peg in hole, placements, alignments, etc.). Skill primitives are robot movements or commands for grippers and tools. The unified modeling language (UML) is used to model the robot tasks and skill primitives. A robot control system uses the skill primitives as input to select the desired control scheme (position, force, or hybrid). In addition to this, we use an algorithm to identify assembly process states considering static friction under uniform gravity to execute skill primitives. This enables a robot to select and modify its motion strategies adequately according to the state of the assembly operation.**

***Index Terms*—Assembly planning, assembly sequence generation, skill primitives, task planning.**

I. Introduction

ONE OF THE goals of robotics research is the development of the third robot generation at the millennium. A robot of the third generation must execute tasks automatically and autonomously. Also, the robot has to be adaptive to its environment. This requires the integration of multisensor information and software tools to control the robot. Robots' main potential is flexibility; they can be used to execute different tasks like, e.g., painting, grinding, and assembling. The desired flexibility only can be obtained by using powerful robot programming and task planning systems. These systems support users to specify and program complex tasks. Task planning deals with specification of robot tasks rather than programming robots explicitly; the user specifies actions on objects only [1]–[3]. A task planning system analyzes task specifications and translates them into robot movements and actions [4]–[9]. This has to be interpreted by the task planning system, translating the task into a sequence of robot movements and actions. The system must take into account the following constraints:

- where are objects located in the workcell?
- are there collision free paths for robot/object movements?
- what kind of tool has to be used to manipulate objects?
- how are spatial relations between objects defined?
- how are objects fixed?

This paper deals with the decomposition of automatically generated assembly sequences into skill primitives considering the above constraints. Skill primitives are elementary sensor-based robot movements ("MoveTo," etc.), system commands ("OpenGripper," etc.) and sensor functions ("LocateObject," etc.). A task specification is for example *Put Piston into fixture and insert piston-pin into hole*; it consists of two sub tasks (putting down the piston and inserting the pin). In [8], [10] we presented our High Level Assembly Planning system $^{\text{High}}$*LAP* which generates and evaluates assembly sequences. In this system the user specifies the goal state of the assembly only. Basing on this specification, the planning system calculates the necessary robot tasks to assemble the desired assembly. In this paper we discuss how to decompose these tasks into skill primitives. Our approach is based on the analysis of hyperarcs of AND/OR-graphs representing the automatically generated assembly plans.

Features like the local depart spaces, symbolic spatial relations, and tools classify the type of assembly sequence (peg in hole, placements, alignments, etc.). The unified modeling language (UML) is used to model the tasks and skill primitives. A robot control system uses the generated skill primitives sequences as input to select the desired control schemes (position, force, or hybrid). Fig. 1 outlines the ideas and organization of this paper.

II. Related Work

Early assembly reasoning systems were strongly based on user interaction. They queried the user for information (e.g., geometric reasoning, precedence relations) and generated assembly sequences on the basis of the given answers [11], [12]). One basic idea in most of the work on assembly planning is the *assembly by disassembly strategy*: The assembly sequences are generated by starting with the completed mechanical product and decomposing it step by step by disassembly operations. Much research work has been done to reduce the necessary user interaction. But even today it is very difficult for an assembly planning system to automatically generate assembly sequences in general. Kavraki *et al.* [13] proved, that the problem of automatically generating assembly sequences is NP-complete even in the two-dimensional (2-D) case. As a

Manuscript received July 26, 2000; revised May 2, 2001. This paper was recommended for publication by Associate Editor Y. M. Wang and Editor S. Hutchinson upon evaluation of the reviewers' comments. This work was supported by the Deutsche Forschungsgemeinschaft. The material in this paper was partially presented at the IEEE International Conference on Robotics and Automation, Videoproceedings, Detroit, MI, May 1999; the IEEE International Conference on Robotics and Automation, Belgium, May 1998, pp. 233–238; and the IEEE International Conference on Robotics and Automation, San Francisco, CA, April 2000.

The authors are with the Institute for Robotics and Process Control, Technical University of Braunschweig, 38114 Braunschweig, Germany (e-mail: HMosemann@aol.com; F.Wahl@tu-bs.de).

Publisher Item Identifier S 1042-296X(01)09464-2.

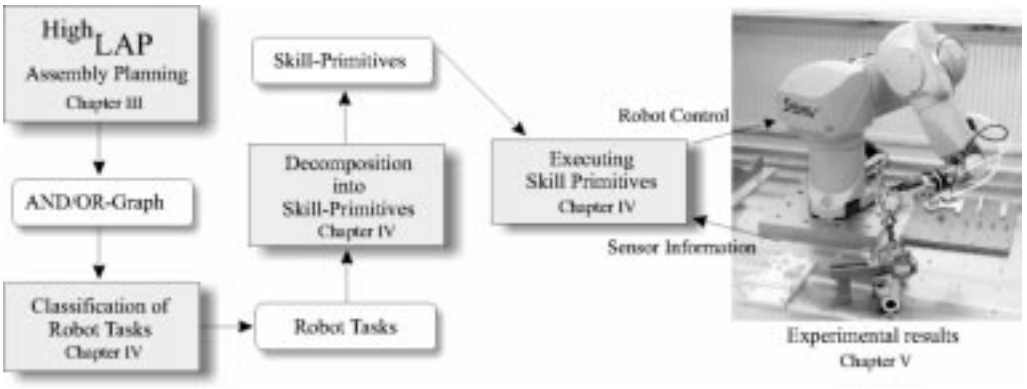

Fig. 1. From assembly planning to execution of skill primitives.

matter of fact, past and present work in this research area make simplifications in order to cope with complexity. In [14], a method to generate *disassembly trees* has been developed showing which subassemblies must be removed before other subassemblies. The system removes one subassembly at a time via single-step translations. In [5] and [15], the computation of the directions of possible part removals is suggested by means of contact information between the subassemblies. Homem de Mello and Sanderson introduce in [16] the AND/OR-graph representation of assembly sequences. Wolter [15] assumes, that geometric reasoning results in a list of directions in which each part might move, along with other parts that interfere with these motions. The feasibility of the motions is determined by sweeping the subassemblies in the proposed direction. Romney *et al.* [9] extended this approach by condensing the local analysis into a single compact data structure called the *nondirectional blocking graph.* Kaufman *et al.* introduce an assembly planning system in [4] which generates an optimized assembly sequence plan from a CAD model of the assembly and translates the plan into robot code for one specific class of assemblies. Raghavan *et al.* [7] introduce an interactive system for generating and evaluating assembly sequences. Interaction bases on augmented reality. In [17], an assembly planning system is introduced which focuses on reusing assembly sequences in different assembly tasks. They discuss how to use precalculated sequences for assembly tasks.

Popplestone *et al.* [3] presented *RAPT* which uses symbolic spatial relations. Laugier *et al.* [1] deal with special problems of task-planning like grasp-planning, coarse and fine motion-planning. In [18] and [19] *skills* have been proposed. Skills are elementary robot actions to reach predefined goal states; sequences of them are used to perform robot tasks. The authors define three different skills: *Move-to-Touch-Skill* is the movement from free space into a vertex–face contact, *Rotate-to-Level-Skill* is the transition from vertex–face to face–face contact, and *Rotate-to-Insert-Skill* is the ability of the manipulator to insert a 2-D peg into a hole.

Hirai [20], [21] use the theory of polyhedral convex cones to recognize contact states based on a geometric model for manipulative operations. He proposes a method to estimate the contact states by using force information acquired in the mating process and develops an algorithm for generating the state classifiers based on geometric models of the objects. One drawback of Hirai's work is, that he ignores friction in his model. Bicchi *et al.* [22] discuss the problem of resolving the location of a contact, the forces at the interface and the torques about the contact normals. In [23], a systematic and general approach to identify topological transitions in contact situations during compliant motion is presented; the identification is based on energy considerations.

Knoll *et al.* [24] pursue the development of a complete system that integrates all relevant aspects of learning, planning and data fusion necessary for autonomously assembling structurally complex aggregates from arbitrarily placed objects.

III. Assembly Planning

$^{\mathrm{High}}$*LAP* developed in our laboratory uses a relational assembly model to reason about the feasibility of assembly tasks. The assembly representation allows the planning system to automatically make deductions from the information contained in the representation in a straightforward and efficient manner. We described the basic features and algorithms of $^{\mathrm{High}}$*LAP* in detail in [8], [25]–[27]. For a better understanding of this paper, we summarize the most important features of $^{\mathrm{High}}$*LAP*:

A. Relational Assembly Model

A relational model $\mathcal{RM}$ of a n-component assembly is a triple, $\mathcal{RM} = \langle C, R, H \rangle$ with

- $C = \{c_1, c_2, \ldots, c_n\}$, $n \in \mathbb{N}$, is a set of triples representing the assembly components.
  $c_i = \langle B_i, \mathcal{F}_i, \mathrm{CSG}_i \rangle$, $i \in [1, \ldots, n]$
  * $B_i \in \mathbb{R}^{4\times4}$ is a homogeneous transformation representing the base frame of the ith component
  * $\mathcal{F}_i = \{\langle F_{i_1}, f\_\mathrm{type} \rangle, \ldots, \langle F_{i_k}, f\_\mathrm{type} \rangle\}$ is a list of tuples describing the feature frames of the ith component. $F_{i_j} \in \mathbb{R}^{4\times4}$, $j \in [1, \ldots, k]$, $k \in \mathbb{N}$, is a homogeneous transformation $f\_\mathrm{type} \in \{\mathrm{face}, \mathrm{shaft}, \mathrm{hole}, \mathrm{edge}\}$ corresponds to a feature type specification.
  * CSG_i is the constructive solid geometry (CSG) representation of the ith component.
- $R = \{r_1, r_2, \ldots, r_{\max}\}$ is a set of 4-tuples representing relations between different components of the assembly, where $\max = \binom{n}{2}$.

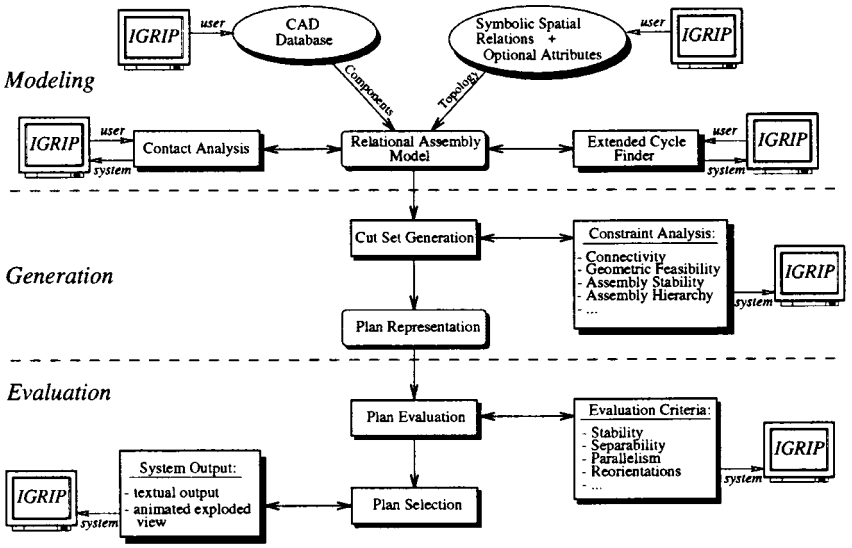

Fig. 2. Architecture of the assembly planning system.

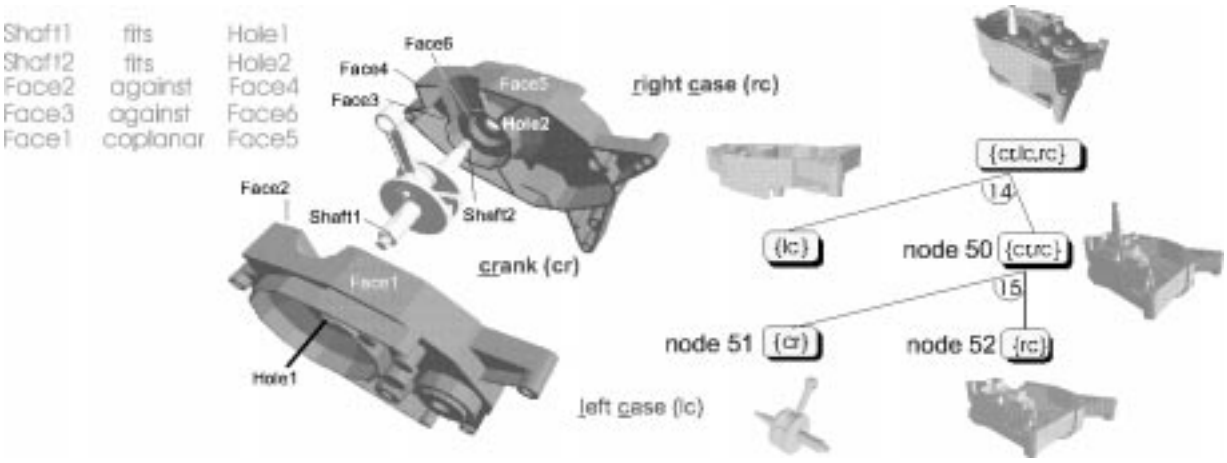

Fig. 3. A subassembly with corresponding symbolic spatial relations and resulting assembly sequence.

$r_l = \langle c_p, c_q, {}^pT_q, \mathcal{C}_{pq} \rangle, l \in [1, \ldots, \max]$

* $c_p, c_q \in C, (p \neq q)$
* ${}^pT_q \in \mathbb{R}^{4\times4}$ is a homogeneous transformation representing the spatial relationship between components c_p and c_q.
* $\mathcal{C}_{pq}$ is a list of tuples describing a set of symbolic spatial relations and the set of all physical contacts between c_p and c_q:

$$\mathcal{C}_{pq} = \begin{cases} \text{NULL}, & \text{no relations} \\ \langle \mathcal{SSR}_{pq}, \mathcal{PC}_{pq} \rangle, & \text{else} \end{cases}$$

  - $\mathcal{SSR}_{pq} = \{\langle F_{p_1}, F_{q_1}, r\_\text{type}\rangle, \ldots, \langle F_{p_k}, F_{q_k}, r\_\text{type}\rangle\}$, $r\_\text{type} \in \{\text{against, fits, aligned, coplanar}\}$, describes a list of symbolic spatial relations between compatible features F_{p_i} and F_{q_i}, $i \in [1, \ldots, k], k \in \mathbb{N}$.
  - $\mathcal{PC}_{pq} = \{\langle c\_\text{region}_{pq_1}, \vec{n}_{pq_1}, c\_\text{type}, a\_\text{type}\rangle, \ldots, \langle c\_\text{region}_{pq_k}, \vec{n}_{pq_k}, c\_\text{type}, a\_\text{type}\rangle\}$ describes all physical contacts between c_p and c_q. $c\_\text{region}_{pq_i}$ lists the vertices of the ith contact region of the physical contact between c_p and c_q, $i \in [1, \ldots, k], k \in \mathbb{N}$, $\vec{n}_{pq_i} \in \mathbb{R}^3$ denotes the normal vector of the ith contact region pointing from c_p toward c_q, $c\_\text{type} \in \{\text{face\_face, face\_edge, face\_vertex, edge\_edge, edge\_vertex, vertex\_vertex}\}$ describes the contact type $a\_\text{type} \in \{\text{glue, pressure\_fit, screw, no\_attach}\}$ corresponds to an attachment acting on $c\_\text{region}_{pq_i}$.
- H is an optional specification of an arbitrary assembly hierarchy.

The assembly components c_i are represented as CSG using the solid modeling system of the robotics simulation system *IGRIP* [28]. As $^{\text{High}}$*LAP* has a modular structure and an open architecture (Fig. 2) we also can deal with other representations like *Boundary Representations* or *Octrees* by easily exchanging the CAD database. Fig. 3 shows a subassembly of a motorcycle-engine and the corresponding symbolic spatial relations to define the topology; the corresponding assembly sequence is depicted on the right side.

B. Generation of Assembly Sequences

The generation of valid assembly sequences bases on the well-known assembly by disassembly philosophy combined with a cut-set method [16]. Our cut-set method uses a simple undirected graph $G = (N, E)$ with $N = C$ representing the set of nodes and E representing the set of edges. An edge $e = (c_i, c_j)$ exists between a pair of components if $\mathcal{PC}_{ij} \neq \emptyset$. $^{\text{High}}$*LAP* distinguishes between *local* and *global* assembly constraints induced by the assembly. Local constraints take into account the corresponding assembly operation; global constraints are applied during the whole assembly process.

Local Constraints:

Connectivity

The cut (C', C'') in G generates two connected subgraphs.

Geometric Feasibility

During the assembly operation of the subassemblies C' and C'' no interpenetration of any components c', c'' ($c' \in C'$, $c'' \in C''$) occurs.

Assembly Stability

Accidental motions of components during the assembly operation caused by gravity, accidental assembly motions, accidental assembly forces etc. are avoided. That is, at least one common stable orientation for the subassemblies C' and C'' must exist or additional fixtures have to be used increasing manufacturing costs.

Global Constraints:

Assembly Hierarchy

Requisite, that subassemblies specified in the optional hierarchy occur in the assembly plan as subtrees.

C. Assembly Stability

Most of the automatic assembly planners developed up to date use heuristical or user defined criteria to determine assembly stability for plan generation and plan evaluation. In order to bring automatic assembly planning closer to reality, $^{\text{High}}$*LAP* performs a powerful geometrical and physical reasoning resulting in mechanical *stable assembly sequences*. $^{\text{High}}$*LAP* is the first assembly planning system taking into account the range of all stable orientations of intermediate subassemblies as well as of the final assembly considering friction for the assembly plan evaluation.

Our stability analysis follows similar lines as the work of Mattikalli *et al.* [29]. In contrast to [29], the algorithm used by $^{\text{High}}$*LAP* guarantees termination, especially in case of degeneracy of the corresponding linear program. In addition, our system computes the set of forces leading to a stable orientation. Knowing these forces, $^{\text{High}}$*LAP* is able to select the most desirable stable orientation depending on these forces. Details are described in [25]–[27].

D. Evaluation of Assembly Sequences

Each cut corresponding to a *feasible assembly operation* is stored as a hyperarc of an AND/OR-graph comprising a compact representation of all feasible assembly sequences. Due to the large number of feasible assembly sequences usually represented by the AND/OR-graph it is desirable to select either a few best sequences or the best assembly sequence. This, however, has been hampered up to now by the difficulty of selecting proper performance criteria (see, e.g., [30]–[32]) and relating them directly to assembly cost. $^{\text{High}}$*LAP* offers a solution to this problem; it takes into account evaluation criteria (stability, separability, parallelism, directionality, and reorientations) and assigns weights to the nodes and to the hyperarcs of the AND/OR-graph to select an assembly plan minimizing the costs from the initial nodes up to the goal node [8].

IV. Decomposition of Assembly Sequences

A. Classifying Assembly Sequences

Assembly operations represented by an AND/OR-graph have to be classified. The classification allows us to decompose the robot task into skill primitives. The following 5-tuple defines a robot task $\mathcal{RT}$:

$\mathcal{RT} = \langle \text{Type}, C, \text{SSR}, \text{LDS}, \text{Tool} \rangle$:

- *Type* is the type of the task.
- $C = \{c_a, c_p\}$ represents the active and passive subassemblies of the task.
  $c_i = \langle B_i, \mathcal{F}_i, \text{CSG}_i \rangle$, $i \in [a, p]$
  * $B_i \in \mathbb{R}^{4\times4}$ is a homogenous transformation representing the ith object's coordinate system in the world.
  * $\mathcal{F}_i = \{\langle F_{i_1}, f\_\text{type} \rangle, \ldots, \langle F_{i_k}, f\_\text{type} \rangle\}$ represents the features coordinate system of the ith object.
    $F_{i_j} \in \mathbb{R}^{4\times4}$, $j \in [1, \ldots, k]$, $k \in \mathbb{N}$, is a transformation $f\_\text{type} \in \{\text{face}, \text{shaft}, \text{hole}, \text{edge}\}$ is a feature.
  * CSG_i is the CSG-representation of the ith object.
- $\text{SSR} = \{\mathcal{SSR}_{ap}\}$ is the set of symbolic spatial relations between the active and passive subassemblies $\mathcal{SSR}_{ap} = \{\langle F_{a_1}, F_{p_1}, r\_\text{type} \rangle, \ldots, \langle F_{a_k}, F_{p_k}, r\_\text{type} \rangle\}$, $r\_\text{type} \in \{\text{against}, \text{fits}, \text{aligned}, \text{coplanar}\}$, describes a list of symbolic spatial relations of the corresponding features F_{a_i} and F_{p_i}, $i \in [1, \ldots, k]$, $k \in \mathbb{N}$.
- $\text{LDS} \in \{\text{space}, \text{halfspace}, \text{quadrant}, \text{polygon}, \text{plane}, \text{halfplane}, \text{sector}, \text{line}, \text{halfline}, \text{point}, \text{empty}\}$ is the local depart space (LDS) of the active and passive subassembly [33].
- $\text{Tool} \in \{\text{jaw\_gripper}_i, \text{multi\_finger\_gripper}_j, \text{screw\_driver}_k\}$, $i, j, k \in \mathbb{N}$ is the tool used for the task.

The left-hand side of Fig. 4 shows the UML-class diagram of the robot task. Table I shows robot tasks considered in this paper and their corresponding features (see the above definition of $\mathcal{RT}$). The algorithm in Fig. 5 classifies hyper-arcs of an AND/OR-graph calculated by $^{\text{High}}$*LAP* and determines the elements of the 5-tuple $\mathcal{RT} = \langle \text{Type}, C, \text{SSR}, \text{LDS}, \text{Tool} \rangle$. $^{\text{High}}$*LAP* stores the generated and evaluated assembly sequences into a text file. Fig. 6 shows the textual output for the first assembly sequence of the AND/OR-graph depicted in Fig. 3 (see labeled nodes 50, 51, and 52). The algorithm in Fig. 5 traverses the textual output in inorder and calls the procedure classify_RT for each parent node of the AND/OR-graph. Features of the robot task like local depart space, symbolic spatial relations, and tools are the criteria to classify the underlying robot task (see Table I). For example, the algorithm in Fig. 5 is called for parent node number 50 of Fig. 6. As the left node 51 is active the elements

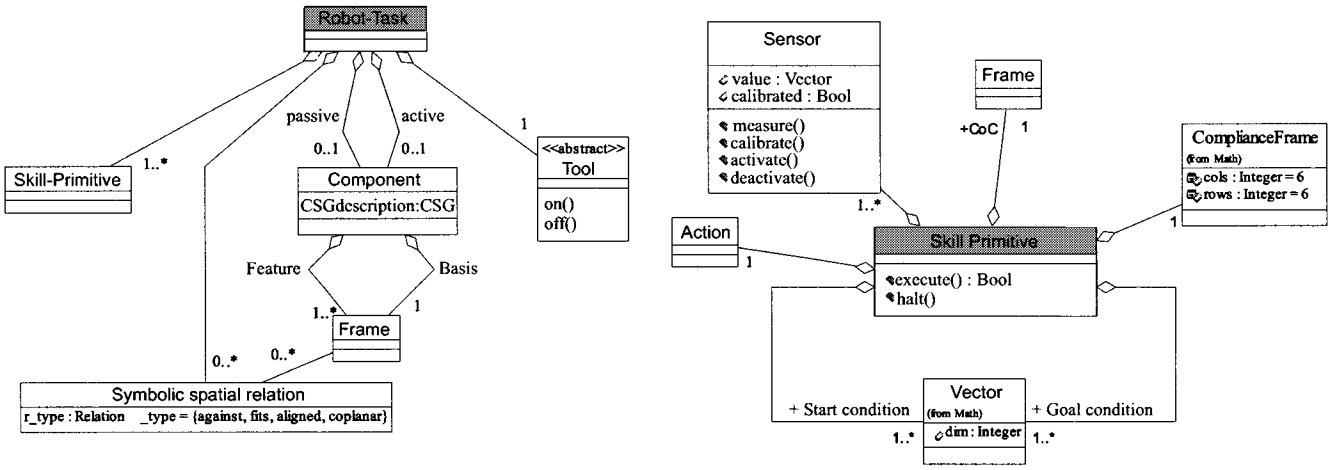

Fig. 4. UML-class diagrams for robot task and skill primitive.

```
Traverse AND/OR-Graph-File in Inorder and call
procedure classify_RT for each parent node V.

procedure classify_RT(V) {
    if (Left node of V is passive) then {
        C.c_p = Elements of left node of V;
        C.c_a = Elements of right node of V; }
    else {
        C.c_p = Elements of right node of V;
        C.c_a = Elements of left node of V; }
    SSR = SSR of V;
    LDS = LDS of V;
    Tool = input user;
    if FITS ∈ SSR then{
        if only one FITS relation in SSR then {
            if Tool == screw-driver then
                Type = Screwing;
            else Type = Peg-In-Hole; }
        else Typ = Peg-In-Hole n-times; }
    else {
        if HALFSPACE ∈ LDS then
            Typ = Put-Down;
        else Typ = Alignment; }
}
```

Fig. 5. Algorithm to automatically classify robot tasks.

of this node are assigned to $C.c_a$ and the elements of node 51 are $C.c_p$. Now, C of $\mathcal{RT}$ is defined. *SSR* of $\mathcal{RT}$ are the symbolic spatial relations of the parent node 50. Also *LDS* of $\mathcal{RT}$ is the local depart space of the parent node. The *Tool* of the $\mathcal{RT}$ must be defined by the user. The *Type* of the robot task (see Table I) is classified by the *SSR* and *LDS* of the $\mathcal{RT}$.

B. Skill Primitives

Skill primitives are elementary sensor-based robot movements Move(Frame) and HybridMove(Frame), system commands OperateTool(On) (this command closes for example a gripper), OperateTool(Off), ChangeTool(Tool), and sensor function LocateObject(object) (this function yields the frame of the corresponding object) and necessary information to execute the primitives like the used sensors, start- and goal-conditions, center of compliance, and compliance frame. The reasons/advantages of decomposing complex robot tasks into skill primitives are

- applicability of *devide and conquer*;
- reuseability of skill primitives;
- ease of error localization;
- skill primitives are the interface to robot control;
- carefully *a priori* designed skill primitives are stored as a ready to use library;
- *skill primitive-morphing*: new tasks can be composed from predefined skill primitives.

TABLE I
ROBOT TASKS CONSIDERED IN THIS PAPER

Task	Features
Peg-In-Hole	SSR = {FITS} LDS = {line, halfline}
Peg-In-Hole n-times	SSR = {FITS$_n$}, $n \in \mathbb{N}$ LDS = {halfline}
Screwing	SSR = {FITS} LDS = {line, halfline} Tool = {screw}
Put-Down	SSR = {AGAINST$_i$} $i \in \mathbb{N}$ LDS = {halfspace}
Alignment	SSR = {AGAINST$_i$, COPLANAR$_j$}, $i, j \in \mathbb{N}$ LDS = {quadrant, polygon}

Sensors are very important for execution of skill primitives. Hager [34] defines a sensor as follows:

Definition IV.1 (Sensor): A sensor S measures observable properties $\vec{p}$ of an object and refers them to measured values $\vec{z}$, i.e.,

$$\vec{z} = S(\vec{p}). \tag{1}$$

The formal definition of a skill primitive $\mathcal{SP}$ is a 6-tuple $\mathcal{SP} = \langle \text{Action}, , \mathcal{S}, \text{start}, \text{goal}, \text{CoC}, \text{CF} \rangle$

- Action $\in$ {Move(Frame), HybridMove(Frame), OperateTool([On, Off])} is the action performed by the robot. The action changes the state of the assembly process and is observed by sensors. Move(Frame) is a movement of the manipulator to Frame $\in \mathbb{R}^{4\times4}$. HybridMove(Frame) is a complex sensor-guided motion of the manipulator [35]. OperateTool([On, Off]) determines the function of the mounted tool.

```
Parent node: 50
Elements: 2, case_right, crank
Node-Costs: 1.20000
Arc-Costs: 0.70000
Stability: PARTIAL
Depart: LINE
SSR: shaft2 of crank fits hole2 of case_right
     face3 of crank against face6 of case_right

Left node: 51
Elements: 1, crank
Node-Costs: 0.00000
active
Stability: TOTAL
Depart: SPACE

Right node: 52
Elements: 1, case_right
Node-Costs: 0.00000
passive
Stability: TOTAL
Depart: SPACE
```

Fig. 6. Part of the textual output of the assembly planning system which describes the first assembly sequence of the AND/OR-graph depicted in Fig. 3.

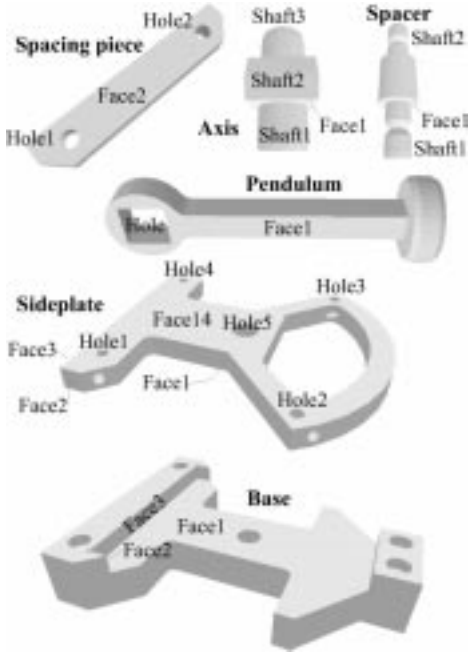

Fig. 7. Objects and features of the Cranfield benchmark.

- $\mathcal{S} = \{S_i\}$, is the set of sensors used to observe the execution of skill primitives.
- start $= \{S_i(\vec{p})\}$ is the start condition of the skill primitive determined by sensor values.
- goal $= \{S_i(\vec{p})\}$ is the termination condition of the skill primitive determined by sensor values.
- $\mathrm{CoC} \in \mathbb{R}^{4\times4}$ is the Center of Compliance (CoC) [36].
- $\mathrm{CF} \in \mathbb{R}^{6\times6}$ is the Compliance Frame (CF) [36].

An object located in space has six degrees of freedom; if its movability is constrained, some of its degrees of freedom are lost. This can be described with a *Compliance Frame* which is an orthogonal coordinate system whose axes may specify task freedoms for compliant motion [36].

The right-hand side of Fig. 4 shows the UML-class diagram of skill primitives.

C. Skill Primitives as Input for Robot Control

A hybrid controller is needed to execute skill primitives [37]. The robot should be controlled in a Cartesian coordinate system (*CoC*) [38], [39]. All necessary input information for the controller can be specified in terms of the skill primitives. That is, skill primitives are an ideal input for hybrid controllers and highly simplify robot task programming.

D. Executing Skill Primitives

The execution of automatically generated assembly plans by robots will become one of the key technologies of modern and flexible manufacturing. During the execution of assembly sequences the robot comes into contact with its environment. Since there are positional and geometrical uncertainties from object representations, robot motions, and sensors, compliance typically is used to prevent excessive contact forces. The contact forces and the resulting torques provide information about the contact geometry which may be used to guide the assembly operation if no hybrid robot controller is available. The identification of assembly process states enables a robot to select and modify its motion strategies adequately according to its current state. To realize the identification of *assembly process states*, it is necessary to identify contact formations [1], [40]–[42]. We propose to do this by using the theory of polyhedral convex cones (PCCs) including the consideration of static friction [35]. A measured force-torque vector belongs to a distinct contact formation if this vector is element of the polyhedral convex cone describing this contact formation. This can be checked by using the PCC-tools described in [20]. In [43] we presented an algorithm to calculate contact graphs representing contact formations of subassemblies. This information is used to generate polyhedral convex cones for each contact formation considering friction. These polyhedral convex cones describe static equilibrium for each contact formation and are used to identify the contact state by measuring forces and torques during a part-mating. We use this algorithm to execute the skill primitives.

V. Results

A. Assembly Planning

We published detailed results of our assembly planning system in [8] and [27]. Furthermore, we presented a video about $^{\mathrm{High}}$*LAP* in [44]. Among others, we used the Cranfield benchmark [45] consisting of 10 parts (see Fig. 7) to illustrate our results. Fig. 8 shows the automatically generated AND/OR-graph of the Cranfield benchmark.

B. Classifying Assembly Sequences

The AND/OR graph of the Cranfield benchmark is used to classify the robot task necessary to execute the assembly. The output of the classification following algorithm in Fig. 5 is ($\mathrm{subassembly}_n$ denotes the subassembly assembled with task

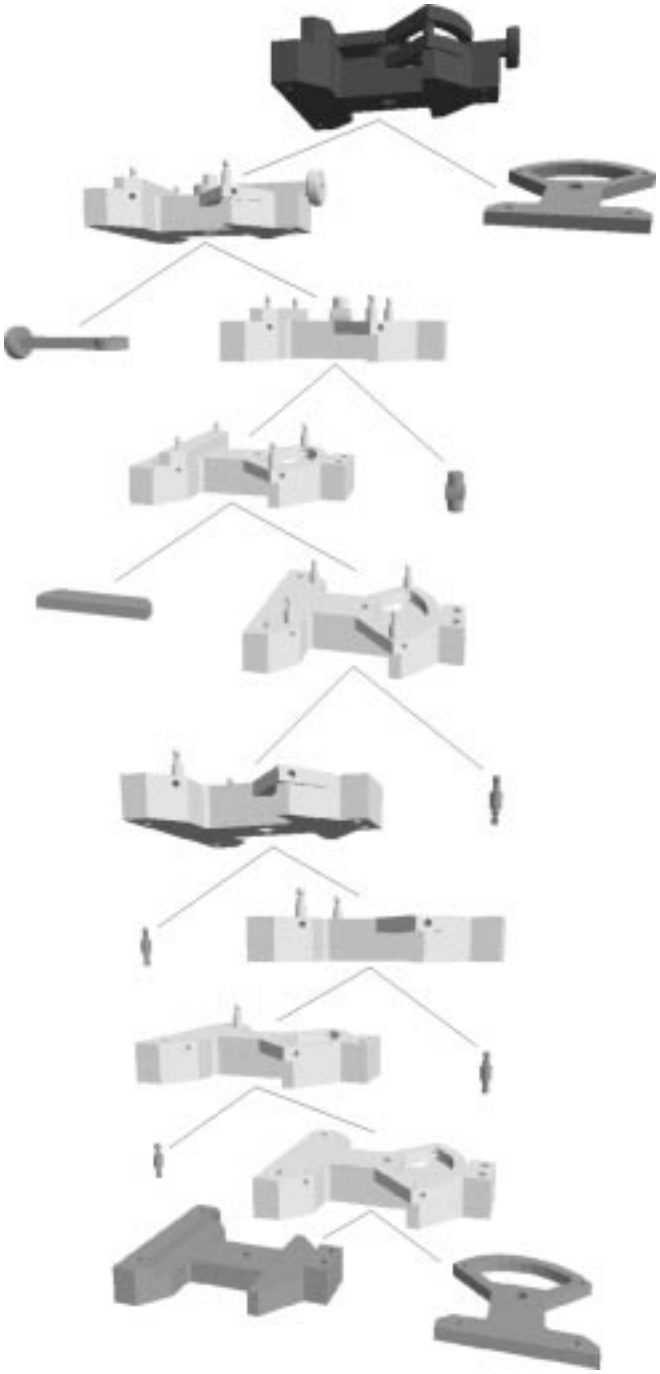

Fig. 8. AND/OR-graph of the Cranfield benchmark.

$\mathcal{RT}_n$):

- $\mathcal{RT}_1$ = ⟨Put-Down, {Sideplate_1, Base}, {Sideplate_1.Face_1 AGAINST Base.Face_1, Sideplate_1.Face_2 AGAINST Base.Face_2, Sideplate_1.Face_3 AGAINST Base.Face_3}, halfspace, jaw_gripper_1⟩
- $\mathcal{RT}_2$ = ⟨Peg-In-Hole, {Spacer_1, Sideplate_1}, {Spacer_1.Shaft_1 FITS Sideplate_1.Hole_1, Spacer_1.Face_1 AGAINST Sideplate_1.$\text{Face}_1$4}, halfline, jaw_gripper_1⟩
- $\mathcal{RT}_3$ = ⟨Peg-In-Hole, {Spacer_2, Sideplate_1}, {Spacer_2.Shaft_1 FITS Sideplate_1.Hole_2, Spacer_1.Face_1 AGAINST Sideplate_1.$\text{Face}_1$4}, halfline, jaw_gripper_1⟩
- $\mathcal{RT}_4$ = ⟨Peg-In-Hole, {Spacer_3, Sideplate_1}, {Spacer_3.Shaft_1 FITS Sideplate_1.Hole_3, Spacer_1.Face_1 AGAINST Sideplate_1.$\text{Face}_1$4}, halfline, jaw_gripper_1⟩
- $\mathcal{RT}_5$ = ⟨Peg-In-Hole, {Spacer_4, Sideplate_1}, {Spacer_4.Shaft_1 FITS Sideplate_1.Hole_4, Spacer_1.Face_1 AGAINST Sideplate_1.$\text{Face}_1$4}, halfline, jaw_gripper_1⟩
- $\mathcal{RT}_6$ = ⟨Peg-In-Hole, {Axis, Sideplate_1}, {Axis.Shaft_1 FITS Sideplate_1.Hole_5, Axis_1.Face_1 AGAINST Sideplate_1.$\text{Face}_1$4}, halfline, jaw_gripper_1⟩
- $\mathcal{RT}_7$ = ⟨Peg-In-Hole2-times, {SpacingPiece, subassembly_6}, {subassembly_6.Spacer_1.Shaft_2 FITS SpacingPiece.Hole_1, subassembly_6.Spacer_2.Shaft_2FITS SpacingPiece.Hole_2, subassembly_6.Sideplate_1.$\text{Face}_1$4AGAINST SpacingPiece.Face_1}, halfline, jaw_gripper_1⟩
- $\mathcal{RT}_8$ = ⟨Peg-In-Hole, {Pendulum, subassembly_7}, {subassembly_7.Axis.Shaft_3 *FITS* Pendulum.Hole, subassembly_7.Sideplate_1.$\text{Face}_1$4 AGAINST Pendulum.Face_1}, halfline, jaw_gripper_1⟩
- $\mathcal{RT}_9$ = ⟨Peg-In-Hole5-times, {subassembly_7, Sideplate_2}, {subassembly_7.Axis.Shaft_2 *FITS* Sideplate_2.Hole_5, subassembly_7.Spacer_1.Shaft_2 FITS Sideplate_2.Hole_1, subassembly_7.Spacer_2.Shaft_2 FITS Sideplate_2.Hole_2, subassembly_7.Spacer_3.Shaft_2 FITS Sideplate_2.Hole_3, subassembly_7.Spacer_4.Shaft_2 FITS Sideplate_2.Hole_4, Sideplate_2.Face_1 AGAINST SpacingPiece.Face_2}, halfline, jaw_gripper_1⟩

Corresponding to the number of hyperarcs in the AND/OR-graph there are nine robot task necessary to build the Cranfield benchmark. After the classification of the robot tasks the system is able to decompose them into skill primitives.

C. Skill Primitives

The decomposition of the above robot tasks into skill primitives enables the system to execute them in the real robot workcell. Here, we only present the decomposition of a peg-in-hole robot task due to limited space. The considered robot is equipped with a parallel-jaw gripper. Furthermore, we use an internal position sensor Ip, an internal gripper sensor Ig, an external force/torque sensor Ef, and an external position sensor Ep. Table II shows the corresponding skill primitives necessary for the peg-in-hole robot task.

D. Executing Skill Primitives

After decomposition of the nine robot tasks into skill primitives we execute them in the real robot workcell. As the used robot has only position control we used the algorithms

TABLE II
SKILL PRIMITIVES FOR THE PEG-IN-HOLE ROBOT TASK

Transfer init to approach$_{grasp}$					
Action	S	Start Condition	Goal Condition	CoC	CF
Move (Approach$_{grasp}$)	Ip Ig St Sa Sf	S_{Ip}(gripper) = init ∧ S_{Ig}(gripper) = off ∧ S_{St}(move) = TimeOk ∧ S_{Sa}(emergency) = Off ∧ S_{Sf}(destroy) = Off	S_{Ip}(gripper) = approach$_{grasp}$ ∨ S_{St}(move) = TimeOut ∨ S_{Sa}(emergency) = On ∨ S_{Sf}(destroy) = On	TCP	
Transfer approach$_{grasp}$ to grasp					
Action	S	Start Condition	Goal Condition	CoC	CF
Move (Grasp$_{monitor}$)	Ip Ep Ig	S_{Ip}(gripper) = approach$_{grasp}$ ∧ S_{Ig}(gripper) = off	 S_{Ep}(peg) = interrupt	TCP	
Close gripper					
Action	S	Start Condition	Goal Condition	CoC	CF
OperateTool (on)	Ep Ig	S_{Ep}(peg) = interrupt ∧ S_{Ig}(gripper) = off	S_{Ig}(gripper) = on	TCP	tz, rx, ry
Transfer grasp to depart$_{grasp}$					
Action	S	Start Condition	Goal Condition	CoC	CF
Move (Depart$_{grasp}$)	Ep Ig Ip	S_{Ep}(peg) = interrupt ∧ S_{Ig}(gripper) = on	S_{Ep}(peg) = no_peg ∨ S_{Ip}(gripper) = approach$_{grasp}$	TCP	
Transfer depart$_{grasp}$ to approach$_{hole}$					
Action	S	Start Condition	Goal Condition	CoC	CF
Move (Approach$_{assembly}$)	Ip Ep Ig	S_{Ip}(gripper) = depart$_{grasp}$ ∧ S_{Ep}(peg) = interrupt ∧ S_{Ig}(gripper) = on	S_{Ip}(gripper) = approach$_{hole}$ ∨ S_{Ep}(peg) = no_peg	TCP	
Transfer approach$_{hole}$ to hole (approaching)					
Action	S	Start Condition	Goal Condition	CoC	CF
Move (hole$_{monitor}$)	Ip Ep Ig Ef	S_{Ip}(gripper) = approach$_{hole}$_tilt ∧ S_{Ep}(peg) = interrupt ∧ S_{Ig}(gripper) = on	S_{Ep}(peg) = no_peg S_{Ep}(peg) = no_peg ∨ S_{Ef}(gripper) ≥ ϵ	TCP	tz, ry, ry
Transfer hole to depart$_{hole}$ (departing)					
Action	S	Start Condition	Goal Condition	CoC	CF
Move (Depart$_{hole}$_tilt)	Ep Ig Ef Ip	S_{Ep}(peg) = interrupt ∧ S_{Ig}(gripper) = on S_{Ef}(gripper) ≥ ϵ S_{Ip}(TCP) ≥ hole_top$_z$	S_{Ep}(peg) = no_peg ∨ S_{Ip}(TCP) ≥ depart_tilt	TCP	
Transfer depart$_{hole}$ to approach$_{hole}$_next (next_holeposition)					
Action	S	Start Condition	Goal Condition	CoC	CF
Move (Hole$_{next}$)	Ep Ig Ip	S_{Ep}(peg) = interrupt ∧ S_{Ig}(gripper) = on S_{Ip}(TCP) ≥ approach$_{xyz}$_tilt	S_{Ep}(peg) = no_peg ∨ S_{Ip}(TCP) ≥ searcharea$_{xy}$ ∨ S_{Ip}(gripper) = approach$_{hole}$_tilt_next	TCP	
Inserting peg					
Action	S	Start Condition	Goal Condition	CoC	CF
HybridMove (Assembly)	Ep Ig Ef Ip	S_{Ep}(peg) = interrupt S_{Ig}(gripper) = on ∧ S_{Ef}(TCP) ≥ ϵ ∧ S_{Ip}(peg_tip) ≤ hole_top$_z$	S_{Ep}(peg) = no_peg ∨ S_{Ip}(peg_tip) = in_hole	Tip	tx, ty, rx, ry
Oppen gripper					
Action	S	Start Condition	Goal Condition	CoC	CF
OperateTool (off)	Ep Ig Ip	S_{Ep}(peg) = interrupt ∧ S_{Ig}(gripper) = on ∧ S_{Ip}(peg_tip) = in_hole	 S_{Ig}(gripper) = off	TCP	
Transfer to depart$_{assembly}$					
Action	S	Start Condition	Goal Condition	CoC	CF
Move (Depart$_{assembly}$)	Ep Ig Ip	S_{Ep}(peg) = interrupt ∧ S_{Ig}(gripper) = off ∧ S_{Ip}(peg_tip) = in_hole	 S_{Ip}(TCP) = depart$_{assembly}$	TCP	

presented in [35] and [43] (see Section IV-D) for the execution. The URL http://www.cs.tu-bs.de/rob/download/cranfield_video.mpg shows a video of the execution of the Cranfield benchmark.

VI. Conclusion and Future Work

In this paper, we outlined advanced assembly planning with ^{High}LAP and presented a new approach to decompose an assembly sequence into skill primitives. Assembly sequences are too complex to be executed directly by a robot. Therefore, skill primitives which can be executed by robots are necessary. We showed how to analyze hyperarcs of AND/OR-graphs for the classification of robot tasks based on features like the local depart space, symbolic spatial relations, and the necessary tools. Our presented methods are based on the output of our assembly planning system ^{High}LAP. The classification is fully automatic with no user interaction. This means, that robot tasks for complex assemblies are generated automatically by our algorithms using the user defined assembly topology only (CAD data, symbolic spatial relations, workcell description). Furthermore, we discussed how to decompose complex robot tasks into skill primitives. Here, we specified robot tasks and skill primitives with UML. The sequence and type of skill primitives that solve a given robot task are determined off-line. Skill primitives can be reused for different robot tasks as we store *a priori* designed skill primitives as a ready to use library. Skill primitives are a suitable interface to robot control. This is another important advantage of using skill primitives for executing complex robot tasks. In order to illustrate the efficiency of our algorithms we presented a real experiment.

In the future we will use skill primitives as input for a hybrid robot controller [39]. Furthermore, we will implement software tools for specifying skill primitives by users. These tools will guide the user to avoid possible errors during the definition of skill primitives because the robustness of the presented algorithms depends on the correctness of the defined skill primitives. Furthermore, we have to develop a strategy adaption if, for example, the goal condition of a primitive is not reached. Another important point is the model based code generation for robot programming. Here, we are planning to generate robot programs by using skill primitives.

Acknowledgment

The authors would like to thank the reviewers and editors for their constructive comments and their review effort. They would also like to thank R. Gutsche and F. Roehrdanz for their contribution on assembly planning.

References

[1] C. Laugier, "Planning fine motion strategies by reasoning in the contact space," in *IEEE Int. Conf. Robot. Automat.*, 1989, pp. 653–659.

[2] T. Lozano-Pérez and P. H. Winston, "LAMA: A language for automatic mechanical assembly," in *5th Int. Joint Conf. Artif. Intell.*, 1987, pp. 710–716.

[3] R. J. Popplestone, A. P. Ambler, and I. M. Bellos, "An interpreter for a language for describing assemblies," *Artif. Intell.*, vol. 14, pp. 79–107, 1980.

[4] S. G. Kaufman, R. H. Wilson, R. E. Jones, and T. L. Calton, "The archimedes 2 mechanical assembly planning system," in *IEEE Int. Conf. Robot. Automat.*, 1996, pp. 3361–3368.

[5] A. C. Lin and Y. G. Shin, "3d maps: Three-dimensional mechanical assembly planning system," *J. Manufact. Syst.*, vol. 12, no. 6, pp. 437–456, 1993.

[6] R. Mantripragada and D. E. Whitney, "Modeling and controlling variation propagation in mechanical assemblies using state transition models," *IEEE Trans. Robot. Automat.*, vol. 15, pp. 124–140, 1999.

[7] V. Raghavan, J. Molineros, and R. Sharma, "Interactive evaluation of assembly sequences using augmented reality," *IEEE Trans. Robot. Automat.*, vol. 15, pp. 435–449, 1999.

[8] F. Röhrdanz, H. Mosemann, and F. M. Wahl, "Generating und evaluating stable assembly sequences," *J. Advanced Robot.*, vol. 11, no. 2, pp. 97–126, 1997.

[9] B. Romney, C. Godard, M. Goldwasser, and G. Ramkumar, "An efficient system for geometric assembly sequence generation and evaluation," in *ASME Int. Comput. Eng. Conf.*, 1995, pp. 699–712.

[10] H. Mosemann, F. Röhrdanz, and F. M. Wahl, "A sophisticated assembly planning system for flexible robot-based manufacturing," in *Int. Conf. Manufact. Automat.*, Hong Kong, Apr. 1997, pp. 329–334.

[11] A. Bourjault, *Methodology of Assembly Automation: A New Approach*: Springer-Verlag, 1987, pp. 37–45.

[12] T. L. De Fazio and D. E. Whitney, "Simplified generation of all mechanical assembly sequences," *IEEE J. Robot. Automat.*, pp. 640–658, Dec. 1987.

[13] L. Kavraki and M. Kolountzakis, "Partitioning a planar assembly into two connected parts is np-complete," *Inform. Process. Lett.*, vol. 55, pp. 159–165, 1995.

[14] T. C. Woo and D. Dutta, "Automatic disassembly and total ordering in three dimensions," *Trans. ASME*, vol. 113, no. 2, pp. 207–213, 1991.

[15] J. Wolter, "On the automatic generation of assembly plans," in *IEEE International Conference on Robotics and Automation*, 1989, pp. 62–68.

[16] L. S. Homem de Mello and S. Lee, Eds., *Computer-Aided Mechanical Assembly Planning*. Boston, MA: Kluwer Academic, 1991.

[17] A. Swaminathan, S. A. Shaikh, and K. S. Barber, "Design of an experience-based assembly planning sequence planner for mechanical assemblies," *Robotica*, vol. 16, pp. 265–283, 1998.

[18] T. Hasegawa, T. Suehiro, and K. Takase, "A model-based manipulation system with skill-based execution," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 535–544, Oct. 1992.

[19] A. Nakamura, T. Ogasawara, T. Suehiro, and H. Tsukune, "Skill-based backprojection for fine motion planning," in *IEEE Int. Conf. Intell. Robots Syst.*, 1996, pp. 526–533.

[20] S. Hirai, "Identification of contact states based on a geometric model for manipulative operations," *Advanced Robot.*, vol. 8, no. 2, pp. 139–155, 1994.

[21] S. Hirai and H. Asada, "Kinematics and statics of manipulation using the theory of polyhedral convex cones," *Int. J. Robot. Res.*, vol. 12, no. 5, pp. 434–447, Oct. 1993.

[22] A. Bicchi, J. K. Salisbury, and D. L. Brock, "Contact sensing from force measurement," *Int. J. Robot. Res.*, vol. 12, no. 3, pp. 249–262, June 1993.

[23] S. Dutre, H. Bruyninckx, and J. De Schutter, "Contact identification and monitoring based on energy," in *IEEE Int. Conf. Robot. Automat.*, Apr. 1996, pp. 1333–1338.

[24] A. Knoll, B. Hildebrandt, and J. Zhang, "Instructing cooperating assembly robots through situated dialogues in natural language," in *IEEE Int. Conf. Robot. Automat.*, Albuquerque, NM, 1997, pp. 888–894.

[25] H. Mosemann, F. Röhrdanz, and F. M. Wahl, "Stability of assemblies as a criterion for cost evaluation in robot assembly," in *8th Int. Symp. Robot. Res.*, Hayama, Japan, Oct. 1997, pp. 66–71.

[26] ——, "Stability analysis of assemblies considering friction," *IEEE Trans. Robot. Automat.*, vol. 13, pp. 805–813, Dec. 1997.

[27] ——, "Assembly stability as a constraint for assembly sequence planning," in *IEEE Int. Conf. Robot. Automat.*, Leuven, Belgium, May 1998, pp. 233–238.

[28] Deneb Robotics, "GSL User Manual, IGRIP Version 4.1," Deneb Robotics, Auburn Hills, MI, 1998.

[29] R. Mattikalli, D. Baraff, P. Khosla, and B. Repetto, "Finding all stable orientations of assemblies with friction," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 290–301, 1996.

[30] R. E. Jones, R. H. Wilson, and T. L. Calton, "On constraints in assembly planning," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 849–863, Dec. 1998.

[31] S. Lee and C. Yi, "Assemblability evaluation based on tolerance propagation," in *IEEE Int. Conf. Robot. Automat.*, May 1995, pp. 1593–1598.

[32] C. K. Shin, D. S. Hong, and H. S. Cho, "Disassemblability analysis for generating robotic assembly sequences," in *IEEE Int. Conf. Robot. Automat.*, May 1995, pp. 1284–1289.

[33] L. Homem de Mello, "Task sequence planning for robotic assembly," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 1989.

[34] G. D. Hager, *Task-Directed Sensor Fusion and Planning*. New York: Kluwer, 1999.

[35] H. Mosemann, A. Raue, and F. M. Wahl, "Identification of assembly process states using polyhedral convex cones," in *IEEE Int. Conf. Robot. Automat.*, Detroit, MI, 1999, pp. 2756–2761.

[36] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. 11, pp. 418–432, June 1981.
[37] H. Mosemann and F. Wahl, "Dekomposition und Ausführung von Roboteraufgaben auf der Basis von automatisch generierten Montageplänen," in *Robotik 2000*, Berlin, Germany, 2000.
[38] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space approach," *IEEE J. Robot. Automat.*, vol. 3, pp. 43–53, 1987.
[39] M. H. Raibert and J. J. Craig, *Hybrid Position/Force Control of Manipulators*. Cambridge, MA: MIT Press, 1984, ch. 5 Compliant Motion, pp. 419–438.
[40] R. S. Desai and R. A. Volz, "Identification and verification of termination conditions in fine motion in presence of sensor errors and geometric uncertainties," in *IEEE Int. Conf. Robot. Automat.*, 1989, pp. 800–807.
[41] J. Xiao and L. Zhang, "A general strategy to determine geometrically valid contact formations from possible contact primitives," in *IEEE Int. Conf. Robot. Automat.*, Nagoya, Japan, May 1995, pp. 2728–2733.
[42] X. Ji and J. Xiao, "Automatic generation of high-level contact state space," in *IEEE Int. Conf. Robot. Automat.*, Detroit, MI, May 1999, pp. 238–244.
[43] H. Mosemann, T. Bierwirth, F. Wahl, and S. Stoeter, "Generating polyhedral convex cones from contact graphs for the identification of assembly process states," in *IEEE Int. Conf. Robot. Automat.*, San Francisco, CA, 2000.
[44] H. Mosemann *et al.*, "The high-level assembly planning system Highlap," in *IEEE Int. Conf. Robot. Automat., Video Proceedings*, Detroit, MI, 1999, pp. 2756–2761.
[45] K. Collins, A. J. Palmer, and K. Rathmill, "The development of a european benchmark for the comparison of assembly robot programming systems," in *1st Robot. Euro. Conf.*, Brussels, June 1984.

Heiko Mosemann was born in Braunschweig, Germany, in 1968. He received the diploma in computer science and the Ph.D. degree, both from the Technical University of Braunschweig, in 1995 and 2000, respectively.

From 1995 to 2000 he was a member of the research staff of the Institute for Robotics and Process Control. His teaching activity is concerned with robot programming languages, geometric modeling, and task planning. His main fields of interest are assembly planning, task planning, sensor-based fine-motion planning, compliant motion, and stability analysis of assemblies. Since 2001, he has been working on security robots with RoboWatch Technologies, Berlin, Germany.

Friedrich M. Wahl received the Diploma in electrical engineering, the Ph.D. degree, and the "venia legendi" in digital signal and image processing in 1974, 1980, and 1984, respectively, from the Technical University of Munich, Germany.

From 1974 to 1981, he conducted research with the Institute of Communication Engineering, Technical University of Munich, in the fields of pattern recognition and signal and image processing. From 1981 to 1986, he was with IBM Research Labs, San Jose, CA, and Zürich in the areas of document analysis, industrial image analysis, and machine vision. Since 1986, he has been a Professor of computing science at the Technical University of Braunschweig, Germany, where he set up the new Institute for Robotics and Process Control. His main interests are in robotics, computer vision, and all aspects of computer science.