KASBA: a knowledge-based system for flexible assembly execution
F. Sprumont & J.-P. Müller
Institut d’Informatique et Intelligence Artificielle, Université de Neuchâtel, Chantemerle 20, CH-2000 Neuchâtel, Switzerland

ABSTRACT

The planning and the execution of an assembly task need to solve constraints depending both on the product and on the tools, expressing uncertainty and unpredictability.

The system KASBA (for Knowledge and Sensor Based Assembly) is a flexible executor for assembly tasks. It is composed of a reasoning module and a robot system integrating a robot manipulator and exteroceptive sensors of vision and forces. The characteristics of KASBA are that the reasoning module uses directly the assembly expertise of the method engineer to solve the unpredictability and that the robot system is able to solve the uncertainty in real time during the execution.

The expertise of the method engineer, as in manufacture for manual assembly, is contained in a workplan that describes the operations to perform to assemble the product. The input of KASBA is such a workplan. By interpreting the workplan, it constructs several models expressing different aspects of the product for the assembly execution. KASBA works on three levels: the product-dependent geometric constraints, the tool-dependent constraints and the uncertainty constraints. Each level uses its own models. The information contained in these models about the product and the assembly operations make KASBA able to achieve the task and to react to unpredictable situations.

The exteroceptive sensors feed back on the tools and the robot performance. These loops correct the uncertainty constraints in real time during action execution. So KASBA does not need an explicit representation of the uncertainty. This simplifies the models and the treatment.
INTRODUCTION

An assembly task is performed by ordered actions which both satisfy the resource-independent constraints depending on the product (relative positions and orientations of the parts, tolerances) and the resource-dependent constraints depending on the tools (forces application, grasping, collision avoidance), to obtain a functional product.

The problem of deriving the actions from the product to actually assemble it has been tackled by numerous systems. These systems use a geometrical model of the parts and a relational model of the product defined as the physical connections between the parts. A physical connection expresses a contact between two parts and it is symmetric. Most systems use the resource-independent constraints to produce an intermediate assembly model defined as a graph of assembly connections. By assembly connection, we mean a relation between parts that describes which part has to be assembled to which one. An assembly connection is therefore oriented. Then, the planner constructs the ordered actions by applying operators starting from the assembly operations which achieve the corresponding assembly connections. The operators are often represented by production-rules with preconditions which are the execution conditions of the actions, and postconditions which are the consequences of the actions. The resulting plan is then used to command an assembly cell generally composed of robots.

Pleiades (Homem de Mello and al. [7]) is an environment that enables the comparison of different products through their assembly feasibility and complexity by generating all the possible ordered assembly connections of these products. Pleiades starts from CAD-models for the geometrical description of the parts. From the geometrical models, it constructs, interactively with the user, the relational model of the product where the physical connections are described in terms of contact surfaces, fastening and degrees of freedom. The strategy used to generate all possible ordered assembly connections is to disassemble the product and to control this decomposition by applying a set of feasibility predicates (about geometry, fastening possibility, tools) and precedence relations, evaluated using the different models. The results are represented by AND/OR graphs.

CIARC (Cooperative Intelligent Assembly Robotics Community) (Ramos and Oliveira [6]) is a multi-agent system composed of five agents. The first one is the MODELS agent that gathers all data about parts: position, orientation, vision patterns, supporting faces, etc. ... The second agent is VISION which is able to define, from two camera images of the workplace, the position, the orientation and the stable states of all elements in the view. The WD agent (World Descriptor) converts the results of the VISION agent to symbolic part relationships ("A is on B", "C is inserted in D", "B is on
the floor”) and geometric constraints about grasping (“G is obstructed by A, B and C”) producing the description of the initial state of the world. The TLP agent (Task Level Planner) uses the results of the WD agent and the description of the goal state to build an action sequence that performs the task and satisfies the grasping constraints. It uses some heuristics to prune the search graph and to control the combinatorial explosion. The TE agent (Task Executor) translates the action plan in the robot control language and manages the space in the working area: collision avoidance, decision about the best way to handle objects and task execution monitoring. This agent reacts to failures: re-grasping and grasping of unknown objects.

SPAR (Simultaneous Planner for Assembly Robots) (Hutchinson and Kak [3]) is an extension of the TWEAK planner (Chapman [2]). It achieves operational, geometrical and uncertainty-reduction goals. The operational goals express the assembly connections to create and the other goals describe the geometrical and tolerance constraints. The plan is incrementally built by action addition using a nonlinear, constraint-posting method. An addition to the plan consists in instantiating action templates with the relevant data, monitoring the coherence of the already existing plan. The geometrical goals are satisfied by placing constraints on the execution of the actions. The uncertainty-reduction goals determine thresholds for introducing sensing actions to the plan to reduce the uncertainty in the world description.

R. Almgren and T. Helgesson (Almgren and Helgesson [1]) propose an integrated system that deals with the complete problem. The product model is composed of instances of a part class describing each part individually (CAD-models) and a mating class describing the physical connections between the parts (surfaces, type of the contact). It generates all valid assembly connections by disassembling the product. All of these assembly connections are grouped in an assembly tree. An assembly expert system chooses the best sequence by applying “if-then-production-rules” expressing the human expertise. The sequence resulting of this manipulation is given to the task planner that uses the classical method to generate the plan. This plan is then expanded to a problem description in a robot and sensor generic language dealing with path and grasp planning.

There is a strong contrast between these approaches where the use of robots is assumed and the manual assembly planning in an manufactural setting. In these approaches, no competence is given to the robots for actually dealing with uncertainty and unpredictability. Consequently, the planner has to manipulate complex representations of the objects and their tolerances. In manufacture, the method engineer has only to specify the assembly operation sequence, the worker being able to visually locate, grasp and assemble the parts.

Our project is aimed at exploiting this distribution of competence, of
which the concept is exposed in Malcolm and Smithers [4]. We are using a robot able to perform actions controlled by exteroceptive sensors. For example, grasping is controlled by a camera and insertion is controlled by both camera and force-sensors. Therefore, we do not need to have explicit representation of uncertainty in higher level, even if we have to take its consequences into account (unpredictability). This is in contrast with Pleiades which uses sensors only to acquire the initial state of the world or SPAR which uses sensors to correct the internal model during execution. On the other side, we want to use the competence of the method engineer rather than to derive the possible assembly operation sequences from geometric models of the product. Our system has to represent the possible assembly operation sequences as described by the method engineer in order to be able:

1. to derive the robot action plan that achieves the task
2. to optimize the actual execution of assembly operations by taking into account resource-dependent constraints
3. to reorganize the actual execution of assembly operations to deal with unpredictability

So, the inputs of our system, KASBA (for Knowledge And Sensor Based Assembly), are a workplan and a sketch of the workplace expressing the method engineer expertise. By interpreting these two documents, the system constructs different models and completes the information about the product (parts, connections) and the tools (action sequences). KASBA works on three levels of representation: the assembly connections, the assembly operations realizing the assembly connections and the actions realizing the assembly operations. For each level, it uses a particular model.

The figure 1 shows the architecture of KASBA.

The purpose of this paper is to present these different models and how these representations are used by the system.

THE METHOD ENGINEER KNOWLEDGE

The method engineer produces two kinds of documents: the workplan which is an assembly operation sequence to be performed to obtain the product and a description of the spatial organization of the assembly workplace.

The workplan is the expression of the method engineer expertise and satisfies resource-independent constraints. Its structure is a list of short sentences as illustrated in the figure 2 for an air-valve. Its interpretation
Figure 1: The architecture of KASBA with the two input documents, the gray box of the models, the dotted box of the reasoning module composed of the three levels and the output system.
assembly of the air valve no 92345-A

workplan:
1. take one valve 34678-c (3) and put it in the fixture
2. take the cone of guidance and put it on the valve
   take one joint BC67298 (4) and put it on the cone
   insert the joint over the cone in the valve
3. take the valve corps 92345 (1) and put it in the fixture
   (bigest aperture up)
4. take the valve, insert it in the valve corps
5. take one valve guide 666279-Al (2) and push it in the valve corps
   (be careful that the valve is well inserted in the valve guide at the end of the operation)
6. return the valve corps and put it in the fixture
7. take the automatic screwdriver and aspirate one screw
   nut M4-2157 (5) screw the screw nut on the valve
8. take the assembled air valve and put it in the box

Figure 2: The workplan for an air valve (the part-numbers are fictive). The numbers in parenthesis make reference to the parts of the air valve of the figure 3

allows to have a global view on the problem to solve. Effectively, the workplan describes successive assembly operations from which we can deduce the assembly connection graph and partially how each assembly connection can be achieved, in particular the tool utilization. These two levels are described in the next two sections.

The assembly connection tree
The assembly connections are the most abstract elements of our system. An assembly connection is a relation oriented from the part to assemble called the component towards the part, called the receptor, used as the reference for the assembly operation execution. Because only one reference is used to realize an assembly connection, a part can be the component of one and only one connection. But a part can be the receptor of several connections.

All connections are grouped in one tree: the assembly connection tree. The nodes are the parts of the product and the arcs the assembly connections between the parts. This tree is a representation of the product to assemble. The assembly connection tree of the air valve extracted from the workplan of figure 2 is illustrated in figure 3. This figure shows that an assembly connection does not necessarily correspond to a physical contact between the two parts which it connects: the valve3 has no physical contact with the corps1, but the two parts are related by an assembly connection.
Figure 3: The assembly connection tree of an air valve with an instantiation of the assembly connection template

because the corps1 is the reference to assemble the valve3.

We attach to the assembly connections the resource-independent geometric constraints. All we need is the position and orientation of the component relatively to its receptor and the method used for the fastening. Then an assembly connection is described by the instantiation of the following template:

connection-id : the identifier of the connection

component : the part assembled to the receptor

receptor : the part used as reference

component-frame : frame attached to the component at the relevant point for the connection

receptor-frame : frame attached to the receptor at the relevant point for the connection

component-orientation : orientation of the component frame relative to the receptor frame after the connection creation

fastening : fastening method used to fix the assembly connection. We distinguish three methods: fastening without fastener, fastening with...
discrete fastener (screw, rivet) and fastening with conditioned fastener (glue, soldering).

KASBA interprets the workplan to determinate and instantiate all assembly connections of the product. Missing data are completed interactively by the user. Some information are lost during the passage from the workplan to the assembly connection tree. The reason is that the workplan is over-constrained being a totally ordered assembly operations sequence, while the assembly connection tree is under-constrained being a representation of partially ordered sequences. But we acquire more reactivity, the system being able, at this level, to select another sequence when failure is occurring. For example, when a tool is broken rendering impossible the execution of some operations, the system can decide to execute a part of the assembly connection tree waiting the tool reparation.

To represent the possible sequences for creation of the assembly connections, we introduce the precedence graph. This graph is directly derived from the assembly connection tree. The nodes of the precedence graph are the assembly connections and the arcs are the temporal links for the connection creation. These arcs can be oriented expressing a precedence relation between the two assembly connections. To obtain this graph, KASBA applies accessibility criteria based on the structure of the assembly connection tree, the fastening methods and the frames of the receptors. The user updates it when ambiguity is appearing. A possible sequence for the creation of the assembly connections is a total order compatible with the partial order expressed in a hamiltonian path through the precedence graph.

The model of operations

The creation of an assembly connection is the result of an assembly operation sequence. Each assembly connection consists in a sequence of five assembly operations. The type of these operations is distinguished according to the object it deals with and its effects on this object:

1. location: this assembly operation deals with the receptor of the assembly connection. It determines the position and the orientation of this part relatively to an absolute frame of the workplace. Its goal is to fix the receptor in stable state at a given position on the workplace. Generally, location needs a fixture.

2. distribution: this assembly operation deals with the component of the assembly connection. Its goals are to separate from bulk and to orientate the component to enable easy grasping. Distribution uses specialized tools.
3. **prepositioning**: this assembly operation deals with the component. It groups two actions: the grasping of the component at the distribution position and its transfer to the receptor with the orientation defined in the assembly connection.

4. **fastening**: this assembly operation is the most complex. Its goal is to block the assembly connection by application of a force to the component. Because we can use different methods to apply the force, the elementary actions which constitute the fastenings are directly described with the specialized tools used to perform them.

5. **check**: this assembly operation is a check over the result of the assembly connection realization. According to the result of the check, we can decide to continue the assembly task.

Temporal links, marking precedence or parallelism, exist between these assembly operations for their execution: we cannot perform a fastening operation before a prepositioning operation, we cannot perform a prepositioning operation before the component is distributed, we can locate the receptor while the component is being distributed if resources allow it, etc. In addition, the result of some assembly operation may be already achieved during previous connection realization (e.g. location of the common receptor of several assembly connections). This relationship defines possible assembly operation sequences.

At this level, KASBA expands the nodes of the precedence graph (the assembly connections) to assembly operation sequences. It interprets the workplan to instantiate the assembly operations and interactively complete the data when information is missing: operation identifier, connection created by the operation, tools allocated for the operation execution and parameters. This process allows taking the available knowledge into account while automatically filtering implicit information.

The result of this expansion is represented by the time graph. This graph mentions each part of the product and has a time axis. The time scale is non-linear but represents increasing dates along the time axis. The interval between the events can vary. The assembly operations are represented by specific symbols. The time graph represent the chronology of the assembly operations and the possibility to execute in parallel some assembly operations. Figure 4 shows the time graph of the air valve.
THE COMPETENCE OF THE ROBOT

The competence of the robot to deal with uncertainty is expressed in the structure of the actions. An action is an elementary operation of the system, executed by tools and the robot, and resulting in the collaboration and coordination of the tools and the exteroceptive sensors. The cooperation is expressed in the regulation loop formed by the sensor on the tool (robot) performance, the tool modifying the world and the sensor measuring the world changes to control the tool performance. The loop structure enables to correct in real time the performance of the tools through the information fed back of the sensor. So the uncertainty can be corrected. The coordination of tools and sensors is possible by the translation of data measured by the sensor to data usable by the tool action procedure. The actions use the information of the model of the parts.

At this level, KASBA expands the assembly operations to actions by, first, allocating resources for the operation execution and then using the description of the tools allocated. It monitors the execution and sends the result to the upper level.

The model of the tools

The term “tool” used here, designates all the instruments necessary to manipulate and act on the parts, with the object of facilitating and helping the robot work. We classify the tools according to three criteria:

1. operation : the assembly operation for which the tool is allocated. The same tool can be used for several assembly operations.
2. mobile/fixed: designates the possibility to move or not the tool in the workplace. This feature determines a robot action sequence for the tool utilization: in the case of a fixed tool (i.e., a fixture for a location) the robot must bring the parts to the tool and use the tool as reference, but in the case of a mobile tool the procedure is inverted.

3. active/passive: designates if the tool has at least one motorized degree of freedom. This feature determines the existence of tool specific actions if it is active.

The functionality of the tool defines its different possible actions. Each action defines a procedure which can command the system controller. This procedure has two inputs: the order value given by the assembly operation, and the results of the data translation by the sensor that close the regulation loop on the procedure. The content of the procedure depends on the assembly operation for the parameters and the structure, and the possible tool (robot) commands for the form. The output is the commands to system controller.

The procedure execution being sensor-controlled, the description of the tool integrates the kind of sensor used. The sensors used for this control depends on the parameters with which the procedure is dealing.

All this information is grouped in a tool-defining structure. The template of this structure is:

```
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tool-id</td>
<td>the identifier of the tool</td>
</tr>
<tr>
<td>fixed/mobile</td>
<td>designates the type of the tool</td>
</tr>
<tr>
<td>active/passive</td>
<td>designates if active, the motorized degrees of freedom</td>
</tr>
<tr>
<td>goal</td>
<td>designates the operation for which the tool is designed</td>
</tr>
<tr>
<td>action sequence</td>
<td>sequence of robot and tool (if active) actions</td>
</tr>
<tr>
<td>sensors</td>
<td>list of sensor associated to robot and tool actions</td>
</tr>
<tr>
<td>procedures</td>
<td>tool specific command sent to system controller (case of active tools)</td>
</tr>
</tbody>
</table>
```

The figure 5 gives an example of tool description: an automatic screwdriver enabling grasping of the screw nut by aspiration.
The model of the parts

The description of the part is based on the idea of context. A context is the circumstances in which the parts are to be considered. In our case, the contexts are defined by the two relations in which the parts are implicated, during action execution:

1. the tool-to-part relation
2. the sensor-to-part relation

So a context is the utilization of a tool or a sensor and determines the kind and the quantity of information useful about the part: the color of an object can be useful for vision recognition, but is useless for grasping... It is like a filter for the whole information that we can extract from the object.

This approach is presented by Moureau[5] for action execution. We use it in other contexts.

So, each assembly action determines two *data blocks* attached at the part described: one in relation with the tool and one in relation with the sensor. A data block is the minimal information needed about the part in the context. Each part is described by a set of these data blocks.
The template for the data block is the following:

object-id : the identifier of the part which the information is about

context : the name of the tool or sensor which defines the information.
   For example: screwdriver1, force-sensor.

values : list of pairs (slot value). The slots depend on the action.

Each part has a particular data block, called *absolute framework* of the object: its content is a geometrical frame that allows to know the position and the orientation of the object in the world and to express geometrical data about the object.

The advantages of this representation are:

1. we deal with the minimal information, then we simplify the treatment
2. we can elaborate methods to manipulate partially unknown objects and acquiring information about them, knowing what information we need.

CONCLUSION

The architecture and the models of KASBA based on the competence distribution makes it simpler and more reactive than the existing systems. The reasons are:

1. we use directly the method engineer expertise expressed in the workplan
2. we do not deal with uncertainty at the high level of planning
3. we use the context idea to keep a trace of the information source

This approach opens new research ways for the behaviours of assembly robots and for the insertion of planners into the industrial world.

The low level of KASBA was implemented by Fabio Manzini and Marie-Lorraine Camacho of the Institut de Microtechnique of the Ecole Polytechnique Fédérale de Lausanne. Different actions, exploiting the coordination between tool and sensor, are running on an industrial ASEA IRB 2000 robot. This robot is able to perform some assembly task taking into account the uncertainty: grasping, insertion, ... The robot is equipped with a three cameras vision sensor and a 6-axes force sensor.
The specification of the models and a part of their manipulation was already implemented. The further development of KASBA will complete this implementation of the method and models, and the test on the robot.

This project was supported by the Swiss National Fund for Scientific Research in the cadre of National Project 23: “Artificial Intelligence and Robotics”.

REFERENCES


