To appear in *Robotics and Autonomous Systems*, **Elsevier, 1999**

Interactive Assembly by a Two-Arm Robot Agent

Jianwei Zhang, Yorck von Collani and Alois Knoll

Faculty of Technology, University of Bielefeld, 33501 Bielefeld, Germany
Tel: ++49–521–106–2951/2952
Fax: ++49–521–106–6440
E-mail: zhang | ycollani | knoll @ techfak.uni-bielefeld.de

Abstract

The development of a robotic system interacting with a human instructor requires not only highly-skilled sensorimotor coordination and action planning but also the capability of understanding and communicating with a human being in a natural way. A typical application of such a system is interactive assembly. A human communicator sharing a view of the assembly scenario with the robot instructs the latter by speaking to it in the same way that he would communicate with a human partner. His instructions can be under-specified, incomplete and/or context-dependent.

After introducing the general purpose of our project, we present the hardware and software components of a robot agent necessary for interactive assembly tasks. The architecture of the robot agent with two stationary robot arms is discussed. We then describe the functionalities of the cognition, scheduling and execution levels. The development tool used for modularly realising these functionalities is presented. The implementation of a learning methodology for a general sensor/actor system is briefly introduced.

Key words: human-robot interface, cognition architecture, sensor-based control, skill learning, multiple sensor/actor systems

1 Introduction

In the Technical Computer Science research group of the University of Bielefeld, a two-arm robotic system is being developed. The general goal of this system is to model and realise human sensorimotor skills for performing manipulation and assembly tasks. This requires a comprehensive set of actuators

and sensors, which perform functions similar to those of the human arms, hands and perception channels (i.e. vision, touch, acoustics). To organise the interaction of the complex sensor and control subsystems, sensor data cannot be acquired and processed independently of the movements of the actuators and the human partner. It is mandatory that these perceptions and actions be performed simultaneously and in view of the task the actuators will perform. This is the reason, therefore, that the three major components of an intelligent robot system, i.e. perception, planning and control, are considered synergetically instead of separately. In this way complex, cooperative behaviours involving sensors and actuators can be realised.

The development of our robotic system is closely linked to the on-going interdisciplinary research program of the project SFB [1] 360 "Situated Artificial Communicators" at the University of Bielefeld. The SFB 360 is aimed at the discovery of linguistic and cognitive characteristics of human intelligence for communication purposes. The results of the project are to be transferred to several application domains, one of which is the emulation of human cognitive principles for information processing systems, [6]. The primary example for demonstrating the usefulness of these newly developed techniques is the robot system mentioned above, whose numerous sensor and actuator modules can be used as a test-bed for investigating the interaction between human "natural" communicators and machine systems in the real-world. Furthermore it will be used for validating the complete concept by integrating different linguistic and cognitive components. As a basic scenario, the assembly procedure of a toy aircraft (constructed with "Baufix" parts, see Fig. 1) was selected. A number of separate parts must be recognised, manipulated and built together to construct the model aircraft. Within the framework of the SFB, in each of these steps, a human communicator instructs the robot, which implies that the interaction between them plays an important role in the whole process.

One challenge of this research program for robotics is to automate the process of multi-sensor supported assembly by gradually enabling the robot and sensor system to carry out the individual steps in a more and more autonomous fashion. A fully automatic assembly, however, presupposes a precise task description; unfortunately, not much work has been done

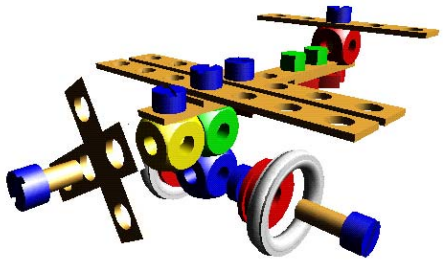


Fig. 1. The assembly of a toy aircraft.

[1] Collaborative research unit funded by the Deutsche Forschungsgemeinschaft (DFG).

in this potentially very fruitful
area of robotics research. While simulated robot agents are becoming a popular research theme, e.g. [7], few work on communicative agents realised with real robots has been reported. A recent project was performed with the KAMRO system [2], in which a natural language interface was used as the "front-end" of an autonomous robot.

2 System Overview

Our robot system has been developed for meeting the demands of flexible "fixture-less" assembly. Its hardware configuration enables a high-speed of the (partly massive) data flows inside the system and the possibility for adding further actuator and sensor components. The structure of the software was so designed as to ensure the compatibility of different program modules to make the whole system work without collisions and deadlocks. The robot control architecture combines the functional modules of perception, planning, control and the human interface.

2.1 Hardware Configuration

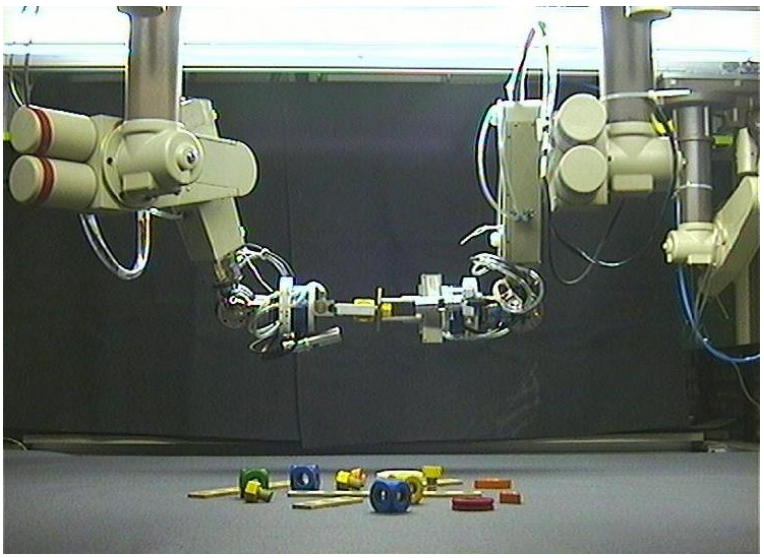


Fig. 2. Two-arm robot system in an assembly scenario.

The physical set-up of our system consists of the following components (Fig. 2):

Main actuators: Two 6 d.o.f. PUMA-260 manipulators are installed overhead in a stationary assembly cell. A third robot arm is currently being installed around the assembly area for extending the work space of the robot

system and aiding in very complex assembly operations. On each wrist of the manipulator, a pneumatic jaw-gripper with integrated force/torque sensor and "self-viewing" hand-eye system is mounted.

Computer system: A multi-computer system consists of robot controllers, UNIX-workstations, Linux and Windows-NT PCs for actuator control, sensor data acquisition, and data processing.

Sensors:

- Three 6 d.o.f. commercial JR3 force-torque sensors are installed on the robots' wrists. They are used for detecting contact, force control as well as two-arm coordination.
- Two miniature colour cameras (Panasonic WV-KS152), each of which is mounted next to the robot gripper. Their function is to perceive local information for fine-manipulation, like grasping, searching a hole, etc.
- Multiple cameras, some of them articulated, are installed around the assembly table. Their tasks are to build 2D/3D world models, to supervise gross motion of the robot as well as to trace the hand and viewing direction of the human instructor.
- A microphone is connected with a voice recognition system, *IBM ViaVoice*, to recognise human speech instructions.

2.2 Software Organisation

On the lowest level, the main actuators of the our system are controlled by Multi-RCCL/RCI (*Robot Control C Library/Real-time Control Interface*), see [3]. With this library, multiple robots can be synchronised and can run in interpolation cycles as short as 10 ms. The high-speed communication between the sensor systems and the robot task-level control is realised using parallel buses.

The motions of the two robot manipulators are controlled by the main control program, which runs on one UNIX-workstation. Image processing, speech recognition and simulation programs communicate with the main control program through sockets. The generated motion steps of the two manipulators are sent to the "trajectory generator", which computes the exact joint values for each control cycle. Through a bus adaptor, joint data are further transferred to the joint controllers of the two PUMA-robots.

The control of the robot is divided in two parts. The first part is the real-time position and joint control of the robot, which is distributed over both the joint controller and the Unix-Workstation running the main control program. The other part is the programming environment, which includes the high-level software of basic skills of the robots, the communication with the robots and the human partner. We are currently implementing a framework

called OPERA (*Open Environment for Robot Applications*) to create software for the robot agent, which will be explained in more detail in section 4. To program an assembly sequence, multiple modules are needed. These modules are loaded dynamically from the environment and include different complex parameterised movements, which are defined as basic primitives for the assembly scenario.

3 Control Architecture

In order to achieve the main objective described in section 1, the system adopts the interactive hierarchical architecture according to Fig. 3. A *Human Communicator* (HC) is closely involved in the whole assembly process.

3.1 Cognition Level

Our robot agent integrates research results from speech recognition, linguistic analysis, intention detection, etc. of different projects within the SFB. The robot system must understand not only the simple verbal instructions, but also to detect the context-related ambiguity with profound linguistic analysis.

The system and the HC interact through natural speech (and in the near future with hand-gestures). First, an instruction is spoken to the robot system and recognised with the *ViaVoice* speech engine. In our current system, *ViaVoice* recognises only sentences, which the pre-defined grammar allows for. In practice, hundreds of grammar rules can be used. If the recognition succeeds, the results are forwarded to the speech analysis module. In Fig. 3, speech recognition and analysis are both included in the "Speech Reception" box.

By their very nature human instructions are situated, ambiguous, and frequently incomplete. In most cases, however, the semantic analysis of such utterances will result in sensible operations. An example is the command: *Grasp the left screw*. The system has to identify the operation (*grasp*), the object for this operation (*screw*), and the further specification of the objects (*left*).

With the help of a hand-gesture the operator can further disambiguate the object. The system may then use the geometric knowledge of the world to identify the right object. Other situated examples are: *insert in the hole above*, *screw the bar on the downside in the same way as on the upside*, *do it again*, etc.

The output of the analysis is then verified to check if the intended operation

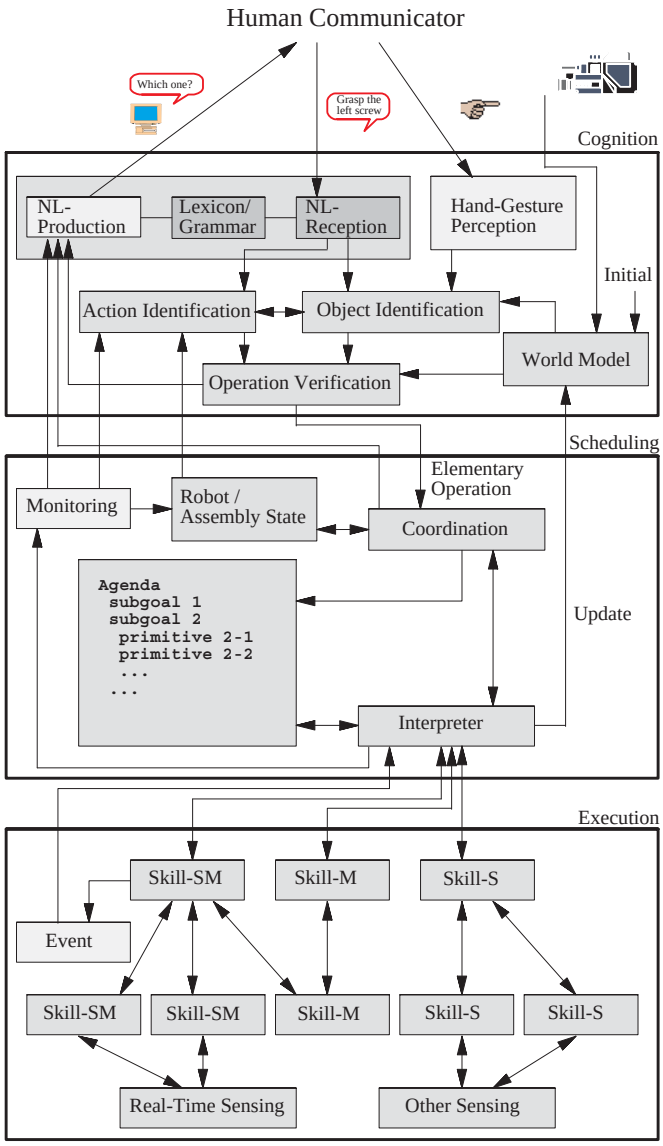


Fig. 3. Control architecture of the communicative robot agent.

can be carried out. If in doubt, the robot agent asks for further specifications or the right to pick an object by itself. Once the proper operation is determined, it is is given to the *coordination* module on the next level. The final result of this level consists of an Elementary Operation (EO) and the objects to be manipulated with the manipulation-relevant information such as type, position/orientation, colour, pose (standing, lying, ...).

An EO is defined in our system as the an operation which does not need any further action planning. Typical EOs are: *grasp*, *place*, *insert in*, *put on*, *screw*, *regrasp*, *alignment* (for an illustration of them see Fig. 6 below).

3.2 Scheduling Level

On the scheduling level, an assembly task of the toy aircraft, or an aggregate of which, is decomposed into a sequence of primitive robot operations. The final decision about the motion sequence depends on the instructions of the human user as well as the generated plan. The *coordination* module should not only be able to understand the human instructions, but also to learn from the human guidance and improve its planning abilities gradually.

The *coordination* module on the scheduling level receives an EO from the cognition level. By referencing the *robot/assembly state*, the *coordination* module chooses the corresponding basic primitive sequence for the operation. This sequence is a script of basic primitives for implementing the given EO. The task here includes planning of the necessary trajectories, choosing the right robot or robots and basic exception handling. The screw operation, for example, is based on the following (simplified) script:

(1) Find contact between screw and nut.
(2) Find the thread and insert the screw.
(3) Find the notch point.
(4) Screw in.

Other sample primitives are: *Find the hole in a ledge*; *Adjust hand*; *Close hand*; *Get location*; *Move to location*.

Sequences are executed by the *interpreter*, which activates different skills on the next level, *execution*. The *interpreter* also receives an event report that is generated by the *execution* level. If the event is a failure detection, the *interpreter* has to handle this exception and to inform the *monitoring* module. The *monitoring* module updates the robot/assembly state. If it is found that the robot agent can re-do the operation, the *coordination* module will try again. Otherwise, the *monitoring* module asks the human communicator how to handle the exception and waits for an instruction. After the execution of each operation, the *world model* is updated.

If the operation is a simple intervention instruction such as "stop!", it is directly forwarded to the *interpreter* and activates the corresponding motion command.

3.3 Execution Level

The *interpreter* on the scheduling level uses the assembly skills provided by the execution level to perform a sequence. In our approach a skill is a powerful

command (a sequence of robot actions). The complexity of the skills can range from opening the hand to collision-free control of the two arms to the meeting point. Advanced skills are composed of one or more basic skills. Generally, we classify three different kinds of skills:

Motoric skills: Motoric skills are single robot movements, which are provided by most commercial robots. Some examples are: *Open and Close gripper; Drive joint to; Drive arm to; Rotate gripper; Move arm in approach direction; Move camera.*

Sensor skills: A sensor skill takes one or more sensors and generates useful information for the scheduling level or other sensorimotor skills. These skills are divided into two groups: skills with real-time sensing (force control and visual servoing) and skills that use sensors in a non real-time environment. Typical sensor skills are: *Get joint; Get position in world; Get force in approach direction; Get torques; Check if a specific position is reachable; Take a camera picture; Detect object; Detect moving robot; Track an object.*

Sensorimotor skills: A sensorimotor skill is an encapsulation of sensing (processing of sensor feedback) and action (trajectories). The main types of force-sensor based skills can be: *Force-guarded motion; Force-supervised contact finding; Force-controlled rotation; Force-balanced two-arm carrying; Maintaining force along a motion on a surface.*

Vision-based motion skills are: *Vision-guided gross movement to a goal position; Visual servoing of the gripper to optimal grasping position; Appearance-based fine positioning;* etc.

Events: The skills on the highest level can also signal an event to the coordination level. These events can be for example: *A force exceeds a defined threshold; A camera detects no object; Singularity; Collision;* etc.

4 OPERA – The Development Framework

OPERA is a framework and programming environment for integrating various software modules. This environment facilitates the programming of reusable software modules and allows the composition of sequences based on these modules. It is an engineering tool as well as the realisation of powerful sensor-based task-oriented operations.

4.1 Architecture of OPERA

OPERA features are:

- Complete hiding of the robot command language and support of human

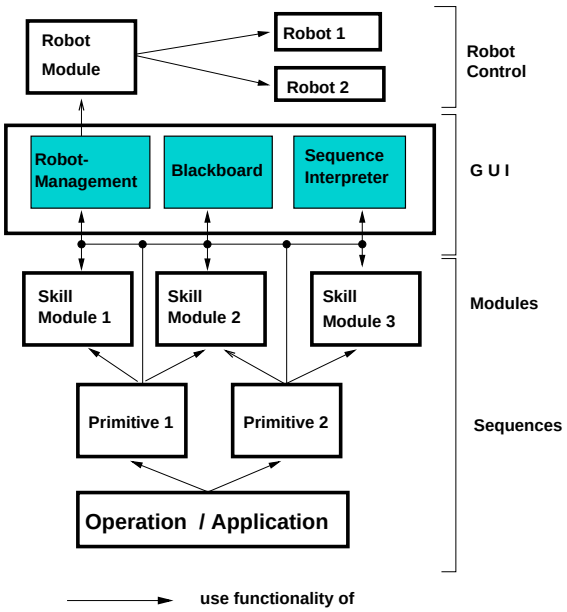


Fig. 4. Architecture of OPERA.

comprehensible commands.

- Task-oriented programming with sensor-based operations.
- A combination of programming, testing and runtime environment with a graphical user interface for easy programming (without compile cycles), testing and sensor monitoring.
- A clear and comprehensible interface for programming with as few restrictions as possible to make changes/enhancements of skills or their parameters.
- An object-oriented design for simple reuse of code.
- A single interface for all robot models.
- Mechanisms to reorder the control flow without direct human interaction.
- A large set of predefined skills.

4.2 Architecture Description

The architecture of OPERA (Fig. 4) is divided into three main parts: modules, robot control, and programming and working environment.

Modules: The functionality of OPERA is based on modules. Each module represents a single operation which adds to OPERA's functionality. A module is loaded on demand and the whole environment has access to this module. All other parts of the system, i.e. other modules, can use the new functionality. The functionality of a module need not be a skill of a robot, it can also be an operation of another part of the assembly system. The combination of a module and a parameter-set for this module is called a

command. The concept is to implement sensor-based operations as modules, and task-oriented operations as sequences of commands. At presence, OPERA supports the following types of modules:

Interactive modules provide the methods described above. All robot skills are implemented with this type.

Run modules are called when they are loaded. These types of modules are used to enhance the GUI or to monitor sensor values.

Exception modules handle a predefined exception and are called when this exception is "thrown" by another module. Exceptions can be interventions from the HC, errors from the robot or the notification that there is a special situation during an assembly operation.

Robot modules provide the interface to the robots.

Robot control: The interface to the robots is realised through an abstract robot class. The methods of this class are low-level robots commands like *move to* as well as higher level commands like *compliant motion*. Through this class, we completely hide the real interface to the robots and can control various robots. Additionally, there is an abstraction from reality possible in that the robot class need not represent a real robot. The class can also be a virtual robot, which in reality is an agent system with the functionality realised by two or more real robots. This robot class contains methods from simple absolute or tool oriented (e.g. "move in normal direction") to force controlled (*screwing* [8]) movements.

Programming and working environment: This part is the user interface of OPERA and contains the following components:

Blackboard: Global information is stored on a blackboard. Each module can load from or store data on the blackboard under a specific name. Information stored on this blackboard can be displayed interactively. The blackboard is used for information which must be shared between several modules (e.g. world model, robot position, robot state).

Sequence Interpreter: Modules can be composed of a sequence of commands, which are executed by the *sequence interpreter*. These sequences can be stored, loaded and run. To compose a sequence, the user chooses the needed modules step by step and edits the parameter set. Due to the pre-initialisation of the parameter classes, the user need not know the possible command settings. Loops and conditions are provided. It is also a powerful tool for testing.

User interface: The environment provides the input/output facilities for the whole system (see Fig. 5).

4.3 Application of OPERA

The various skills on the *execution* level are realised through modules. Through the facility of inter-module calls the hierarchy of the skills can be mapped di-

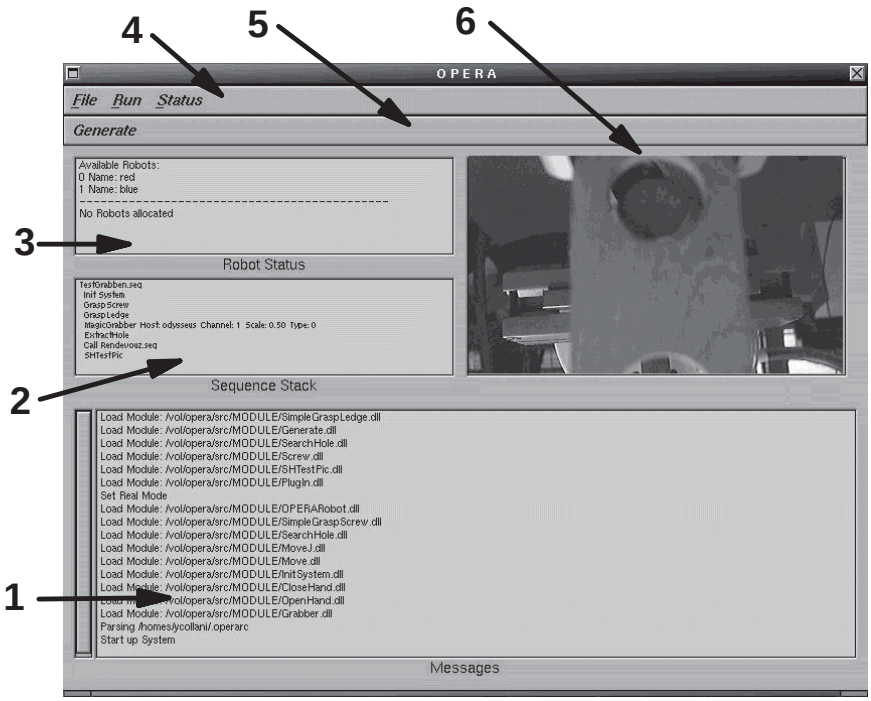
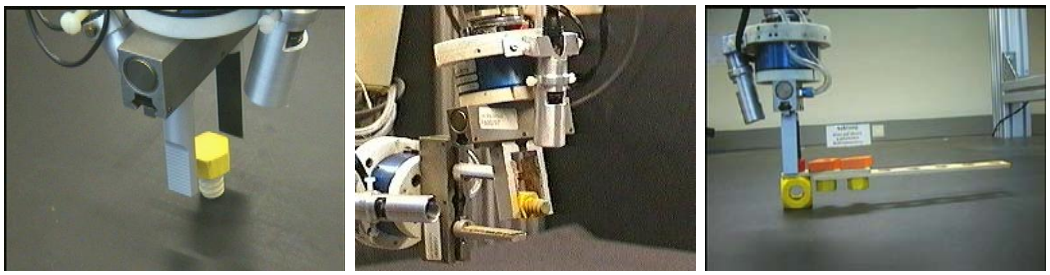


Fig. 5. GUI of OPERA.1) Window for messages, 2) Running script, 3) Robot state, 4) System menu, 5) Custom menu, 6) Sensor display.

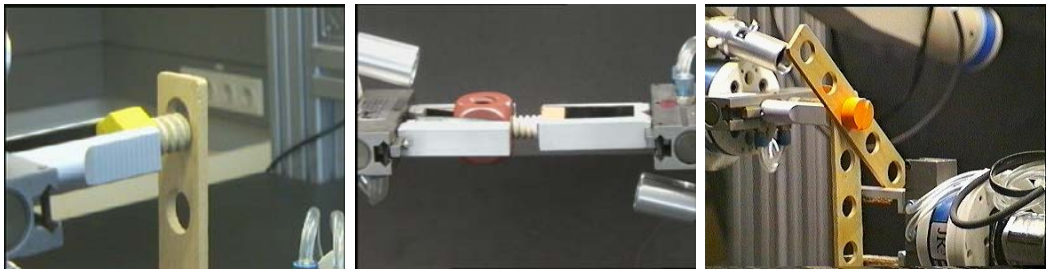
rectly into programs. The *scheduling* level is a combination of the blackboard and the sequence interpreter. The robot and assembly states are stored on the blackboard for global access. After choosing the right sequence of *basic primitives* the interpreter executes the corresponding script, which is a sequence of commands. If an event occurs on the *execution* level, the interpreter is interrupted and the responsible event-module is called, which handles the exception. An event from the HC, such as "stop", is forwarded directly to the *interpreter* and the robot control to stop the current motion as quickly as possible. Additionally an event is *thrown* to handle this situation with a special sequence or module. Altogether, the system has a set of sensor-based robot commands and can implement many assembly tasks with just a few advanced operations.

5 Learning Assembly Skills

In section 3.1 we summarised some typical EOs. The robustness of these operations mainly depends on the quality of the different skills.



(a) Grasp a screw (b) Regrasp (c) Place an aggregate



(d) Put a part in (e) Screw (f) Alignment

Fig. 6. Examples of Elementary Operations.

5.1 Learning by Practising – an Approach for Acquiring Skills

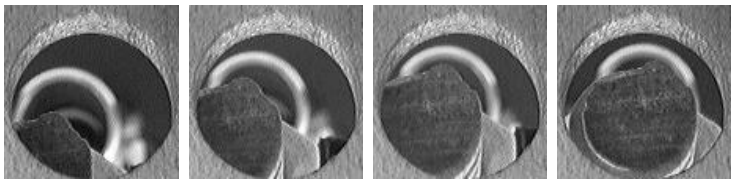
Up to now, several sensor-based skills have been acquired with an automatic learning method. We view the problem of skill learning as finding an optimal mapping function from sensor pattern to robot motion. Since in most cases such a direct mapping function is non-linear, we adopt an adaptive B-spline model for the learning process [10]. For vision-guided fine-motion, the appearance-based approach by using dimension reduction with PCA (principal component analysis) was proposed in [9].

5.2 Grasping

To grasp an object at an arbitrary position and orientation, the main sensor data come from a vision system. The important vision-based skills are multicamera-guided gross motion [5] and optimal grasping using a hand-eye system [4].

5.3 Finding a Hole

A frequent operation for connecting parts is screwing. This simple operation may fail if the exact position of the hole is unknown. Fig. 6(d) shows a typical situation. To find the hole, we developed an approach using visual learning [8]. Fig. 7 shows the visually guided position correction with the learned controller.



(a) (b) (c) (d)

Fig. 7. Correction sequence guided by a hand-camera.

5.4 Screwing

Screwing with two arms is a frequently used operation in our assembly scenario. In order to realise the skills for screwing under diverse uncertainties, we proposed an on-line reinforcement learning method in [12]. After repeatedly practising a specified skill in the real world, a controller can find the optimal compliance parameter by itself.

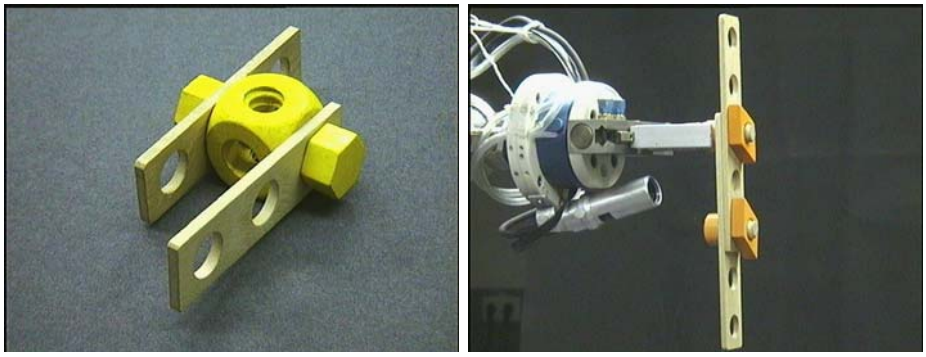
5.5 Assembly of the "Elevator Control" and the "Fuselage"

The assembly of the first aggregate, the "elevator control" (Fig. 8(a)) and the "fuselage" (Fig. 8(b)) of the Baufix toy aircraft, has been successfully performed under a subset of natural language instructions, [1,11].

6 Future Work

Future work on the development of the communicative situated robot agent can be summarised as follows:

Scaling the robot skills to 3D aggregates: The complex manipulation tasks cannot succeed until multiple sensors are applied simultaneously. To enhance



(a) The "elevator control" (b) The "fuselage"

Fig. 8. Two aggregates made by our interactive assembly system.

the robustness, a manipulation skill should be applicable to similar but varied geometry, colour or illumination conditions. Neuro-fuzzy approaches are promising for skill adaptation and skill transfer.

Learning an assembly sequence: In the current implementation, no planning module has been integrated yet. In the future, the robot agent should not always need to be told the explicit assembly sequence. We will investigate how the robot can learn from the assembly sequences it has carried out and plan subgoals by itself. The robot should be able to differentiate between meaningful long-term memory and state-recording short-term memory. The functional modules in our future architecture will be more closely coupled than the case at the moment.

Planning: The robot can show more assembly intelligence if it can not only generate action sequences based on memory but also plan robot-independent assembly steps by using structural models, disassembly knowledge or by reading illustrated instructions. Furthermore, scheduling of multiple robots for an EO needs planning skills for resource management. Planning collision-free motion for uncalibrated multiple arms to a meeting point will need the integration of geometry-based path planning and vision-based on-line control capability.

Comprehensive human-robot communication: Real understanding of natural continuous speech of human involves various aspects of psycholinguistics, dynamic naming, dynamic knowledge representation, etc. Other types of human perception such as gesture recognition, intention detection by observing motion sequence will help the robot disambiguate both in the speech and visual recognition domain. The results of the parallel research work in the framework of the SFB 360 will be further integrated into our architecture. More comprehensive situated dialogues will be performed between our robot agent and the human communicator.

References

[1] A. Knoll, B. Hildebrandt, and J. Zhang. Instructing cooperating assembly robots through situated dialogues in natural language. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.

[2] Th. Laengle, T.C. Lueth, E. Stopp, and G. Herzog. Natural language access to intelligent assembly robots: Explaining automatic error recovery. *Artificial Intelligence: Methodology, Systems, Applications*, 1996.

[3] J. Lloyd and V. Hayward. Multi-RCCL Users' Guide. Technical report, McGill Research Center for Intelligent Machines, McGill University, 1989.

[4] P. Meinicke and J. Zhang. Calibration of a "self-viewing" eye-in-hand configuration. In *Proceedings of the IMACS Multiconference on Computational Engineering in Systems Applications*, 1996.

[5] C. Scheering and B. Kersting. Using distributed sensing and sensor fusion for uncalibrated visual manipulator guidance. In *Proceedings of the 4th International Symposium on Distributed Autonomous Robotic Systems*, 1998.

[6] SFB360. Sonderforschungsbereich Situierte Künstliche Kommunikatoren - Finanzierungsantrag. Technical report, Universität Bielefeld, 1993.

[7] K. R. Thorissen. *Communicative Humanoids - A Computational Model of Psychosocial Dialogue Skills.* PhD thesis, MIT Media Lab., 1997.

[8] Y. Von Collani, J. Zhang, and A. Knoll. A neuo-fuzzy solution for fine-motion control based on vision and force sensors. Technical report, Department of Technology, University of Bielefeld, 1997.

[9] J. Zhang and A. Knoll. Constructing fuzzy controllers for multivariate problems using statistical indices. In *Proceedings of the IEEE International Conference on Fuzzy Systems, Alaska*, 1998.

[10] J. Zhang and A. Knoll. Constructing fuzzy controllers with B-spline models - principles and applications. *International Journal of Intelligent Systems*, 13(2/3):257–285, Feb./Mar. 1998.

[11] J. Zhang, A. Knoll, B. Jung, I. Wachsmuth, and G. Rickheit. Experiments of robotic assembly instructed by situated natural language. In *Video Proceedings of IEEE International Conference on Robotics and Automation, Albuquerque*, 1997.

[12] J. Zhang, Y. v. Collani, and A. Knoll. On-line learning of sensor-based control for acquiring assembly skills. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.