TOPOLOGICAL TASK SPACE MODELLING FOR AUTONOMOUS SPACE ROBOT ACTION PLANNING

J. MATTHIESEN

Technical University of Darmstadt, Control Systems Theory and Robotics Department, headed by Prof. Dr. rer.-nat. H. Tolle, Landgraf-Georg-Str. 4, D 64283 Darmstadt, GERMANY

Abstract: Future space robot generations will replace astronauts in deep space missions and routine operations. They will use tools, perform assembly, disassembly and handling tasks for maintenance and repair purposes, among others. A key feature of autonomous, task-level commandable maintenance robots is a valid and complete representation of an application's task space for planning and optimizing a rough action sequence facing a specific sensorially classified situation. This paper shows such a representation for a robot based automated material science experiment setup und proposes a method of analysis by which a valid and complete task space model can be obtained. Results of practical experiments with a terrestrial laboratory mock-up using the novel representation scheme are presented as well.

Key Words: Space robotics, autonomous robot control, task level programming, model based task planning.

1. INTRODUCTION

Man's investment into space technology during the past decades pays off nowadays as widely increased capabilities for communications, global navigation, earth observation and a large amount of new and fascinating insights into the nature of our solar system. As applications become more widespread the amount of work to be performed on satellites or space-stations will rise beyond the technical and financial means available today (*SPACE* magazine, 1991). Automating loading, berthing, docking and handling tasks using advanced robot technology is already under way (Hirzinger 93), though currently man is still in the loop. In the long run lightweight space robots will be endowed with enough mechanical degrees of freedom and robust sensor technology to cope with many routine tasks in an autonomous fashion. Prerequisies for this to become true are adequate structuring of hard- and software, dedicated automation of mechanical dexterity for certain complicated tasks and a systematical means of designing and modelling a space robot's working environment such that local decisions can be made by an Expert-System like high-level control module. In this paper we will focus on the systematical modelling for autonomous decision taking. We will, however, put this into some perspective, reviewing related work done for assembly automation and embedding the concept presented here into a hard- and software control framework developed at THD's control systems theory and robotics department during the past three years. Practical implementation and experiments underpin the necessity for a systematical design of high-level robot control algorithms.

2. EXAMPLE AND PROBLEM STATEMENT

Research on automating handling tasks with robots performed on ground in a laboratory setup will always face some scepticism concerning the transferability of its results to a real space environment. Therefore, some care has been invested into the selection of working scenarios exhibiting a clear relevance for space automation tasks. In order to simplify the comparison to previous approaches a classical scenario has been chosen for exemplifying our current work:

In the Spacelab D1-mission, a material science experiment rack called Critical Point Facility (CPF) was

flown which was re-built by Kegel (Kegel 90) in a simplified fashion for our laboratory (Fig. 1). The CPF-mock-up called THEMSE (Testbed for Hierarchical Expert Manipulation of Space Experiments) consists of a rack with several integrated units. Among them is a mechanically operated drawer containing four metal probes (Fig. 2). These probes have to be processed in a furnace and a freezer device. The furnace's interior is hidden behind a flap which may be opened or closed by operating a switch in order not to touch the "hot" surface. The freezer's cabinet is closed by a lid with a bajonet lock.

design (Fig. 3). The robot system's sensor suite consists of a wrist force/torque sensor manufactured by LORD Corp., a laser triangulation distance sensor, grip width, grip force and an optical slippage detector grid within one of the gripper jaws. 3D-stereo-vision is scheduled to be integrated in the course of 1994.

In fully autonomous mode the robot system will be given a number of tasks in symbolical teletype form, e.g. "process probes 1 and 2 in the furnace for 15 min., cool them down in the freezer for 5 min. and store them in the drawer again."

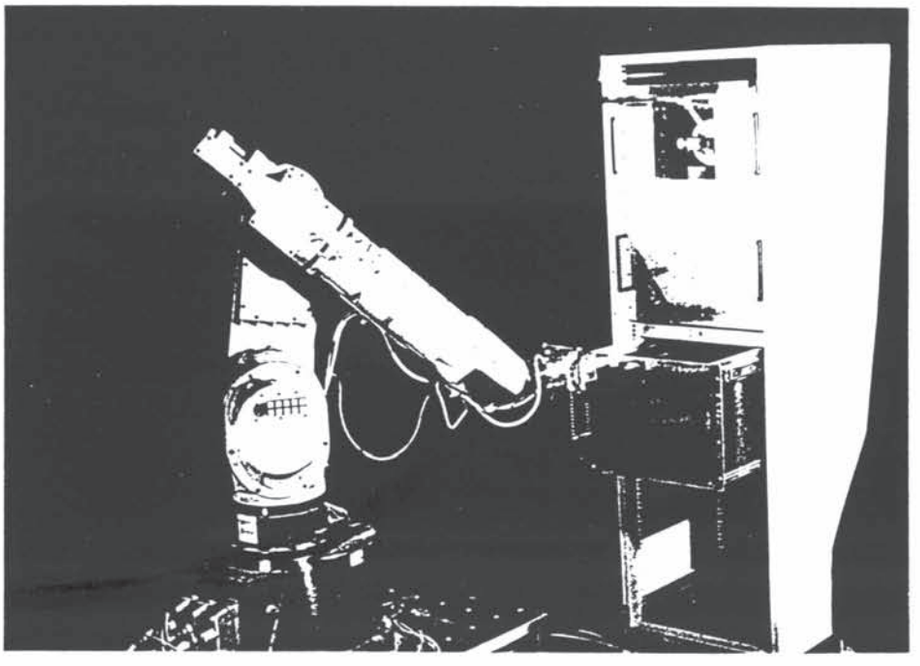

Fig. 1. CPF-experiment mock-up

The set of tasks to be performed by the robot being of standard industrial type (manutec r3, a 6 DOF Puma-type robot) comprises:

- operating the drawer, the freezer's lid and the furnace's flap.
- transporting the probes between their various storage and processing places.
- waiting for process events to trigger continued probe handling.
- detection of abnormal conditions, error diagnosis and treatment to an extent the robot system is capable of.

Gripping and manipulating is performed by means of a parallel jaw electrically controlled gripper of custom

The principal question addressed in this paper reads: In order to be able to autonomously plan and carry out a sequence of sensor-guided robot and gripper motions that reliably implement a probe-handling sequence like the one sketched above - which is a suitable high-level representation of the application's task space and how can it be arrived at? Follow-up topics that cannot be fully elaborated in this paper are planning, optimization and execution of task sequences.

3. RELATED WORK

Earlier attempts at automating the probes' processing took their cue from the widespread task-decomposi-

tion paradigm (Kegel, 90), (Lumia, 90), (Putz, 91). In short, this paradigm calls for regarding command symbols as decomposable functions or "methods" of "objects" within an application, e.g. moving a probe symbolized by the verb "put" would be a method of object "probe" its target parameters being "furnace", "freezer", "drawer" and an additional method modifier being "into", "before" or "on". Thus, control of an application can be effected on a relatively high level, e.g. "put probe_1 into furnace" adhering to a simple predefined formal grammar. The method "put" of probe_1 can be decomposed either on a global scale (Kegel, 90) into general subtasks like

- prepare target-object
- prepare source-object
- grip source-object
- transport source-object
- release source-object
- post-process source-object
- post-process target-object

which are instantitated consequently by an object specific list of motion procedures provided by a robot systems programmer. These motion procedures may be interspersed with procedures testing sensor conditions enabling program-flow modifications in a go-to manner. Since the domain of application of these motion procedures is known at the time of programming, additional parameters needed for calling can be extracted directly from a frame-based object data representation.

A more strictly object oriented approach to task decomposition developed at Sandia Labs is reported in (Miller, 92). By adhering to the well-known assets of object oriented techniques a robot independent programming environment was realized which enables implementing a robot based application on a fairly high level by abstracting from real robots and sensors via C++'s virtual function features. Here, the command flow is implemented by traditional message passing between objects distributed over a heterogeneous UNIX/SUN and VxWorks/VME-bus environment. Task decomposition happens in the mind of the programmer who assigns responsibilities to the software objects, e.g. robot.grip(waste_cask) in C++'s syntax.

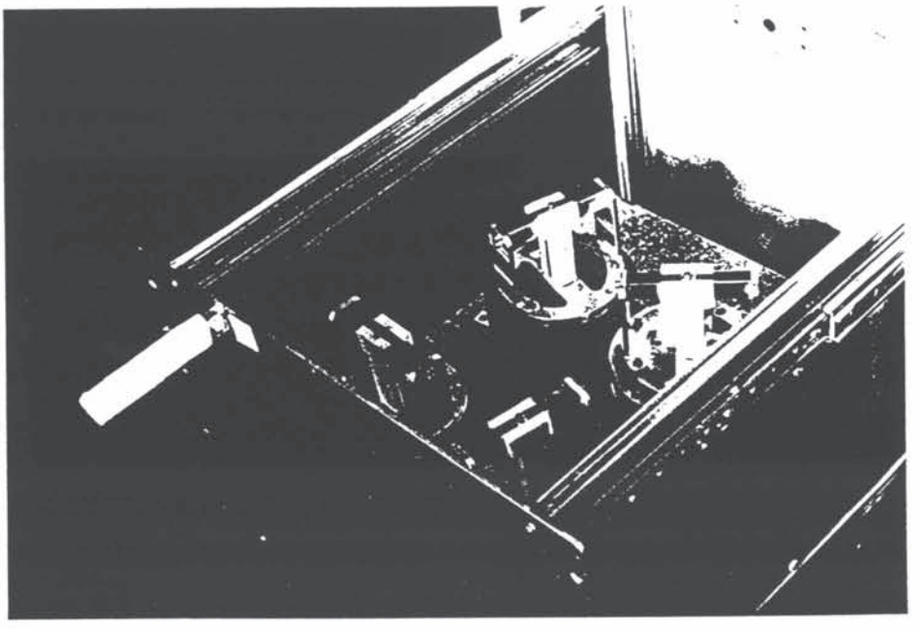

Fig. 2. CPF-probes inside drawer

A drawback of these approaches from the viewpoint of system autonomy consists in modelling and implementing only one optimal or a few nominal high-level sequences stopping task abstraction at a point where combinatorial explosion will forbid explicit coding of all alternatives. This does not constitute a serious

problem if the application exhibits a linear character (i.e. the control decision tree has only few ramifications). Examples are fixed sequence strategies like the one presented for "put probe into furnace". Increasing system autonomy beyond this stage without generalizing and adding to the control paradigm results in exponentially growing coding and testing costs. Related to this, control program error probabilities increase. Both effects are intolerable for space missions.

One tool for unravelling the tangles of a complex robot application is systematical decoupling, i.e. the components of a global application state should be controllable one by one or with as little interaction as possible. Therefore, representation plays an all-important role in achieving system performance.

Assembly and disassembly are part of the complex handling tasks a maintenance robot has to carry out. There have been a number of researchers who developed high-level state representations for the assembly of mechanical parts.

Petri nets have been a useful tool among them in representing partial order graphs for robot assembly control (Hörmann 89). These graphs result from a geometrical and technological analysis of the product to be assembled (e.g. the Cranfield assembly benchmark). Lefebvre and Saridis also report in (Lefebvre 92) on a Petri-net based approach for coordination-level control of a multi-robot system and stress the suitability of the graphical net rendition as an intuitive user interface. They did not yet address the problem of a systematical net design, however, given an arbitrary application.

Starting from a geometrical and contact model of the product, Sanderson and de Mello (Sanderson, 87) construct an AND/OR-graph of all feasible subassemblies for which correctness and completeness has been shown in (de Mello, 91). Feasibility is tested regarding contact degrees of freedom and geometrical penetration. Assembly operations that can be carried out in parallel are clearly marked. Thus, decoupling is an inherent feature of this approach. An assembly state is given by a pattern of markers on nodes of the AND/OR-graph denoting currently active subassemblies.

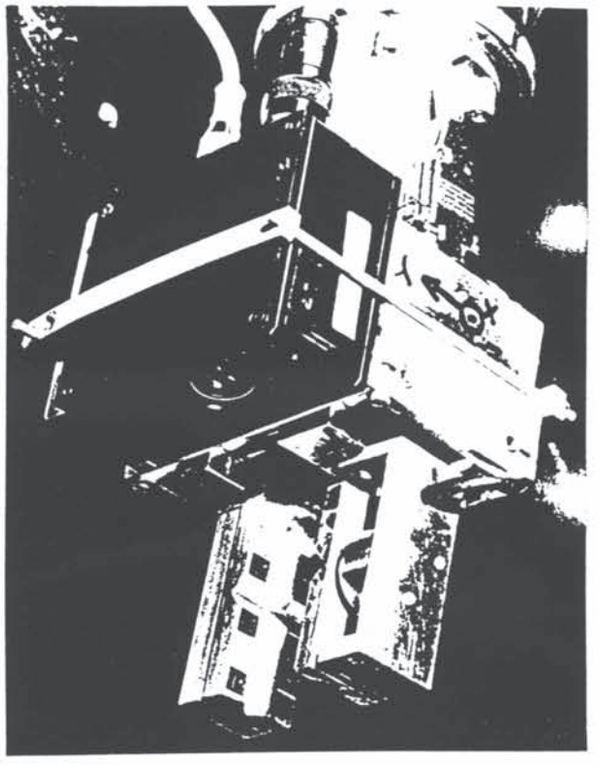

Fig. 3. MARS-gripper

De Fazio and Whitney (de Fazio 87) deduct temporal restrictions (precedence relationships) for assembly operations corresponding to contacts between subassemblies in a precursory analysis phase. In this, they may consider the product's geometry and technological restrictions imposed e.g. by fluid containers or sensitive units. Taking the precedence relationships, an assembly state graph with directed links representing assembly operations is generated. A node of this graph corresponds to exactly one assembly state and vice versa. In this representation, alternative assembly operations result in forking of the decision tree. The number of states grows exponentially with the complexity of the application. Parallel operation is not mentioned. On the other hand, v. Dungern in his dissertation (v. Dungern 92) maintains that the de Fazio/Whitney representation is better suited to the way of reasoning of a human assembly expert and does neither presuppose a formal product model nor computer implementations. He extends their approach by explicitly enumerating product aspects leading toward precedence relationships taking into account necessary fixtures. He also distinguishes between intentional contacts and by-products thus eliminating problems with co-occurrence of contact initiation and

does not extract an optimal assembly sequence from the graph like de Fazio and Whitney but retains the whole graph for an online planning phase. V. Dungern acknowledges that this representation is not necessarily complete owing to a possible over-restrictiveness of the (manually generated) precedence relationships.

Fundamental work concerning online situation-based planning has been presented by Fox and Kempf (Fox, 86; Fox, 87). They argue along the line that in the offline analysis phase only part of the information relevant for selection of alternative operation sequences is available. Any offline decision based on hypotheses about online sensor data etc. will most likely be inappropriate to the real situation. Fox and Kempf propose to use a *least commitment strategy*: taking decision at the latest possible moment or only in order not to violate side conditions. Arbitrary decisions are thus prevented. On the other hand, the term *opportunistic decision taking* was coined for online selecting only among locally available options. Selection is an optimization problem and can be based on local or global criteria which may be evaluated statically offline in the modelling phase or dynamically online while the robot is running. Offline global statical evaluation results in fixed local priorities for the alternatives which are efficient for online processing. Some degree of online dynamical evaluation of path costs can prevent the system from taking suboptimal paths. The depth of the forecast should sensibly depend on the expected accuracy of the results which decreases with the number of steps.

Summing up, the Sanderson/de Mello-approach seems most suitable for representing large scale applications with a high number of alternative ways of manipulation due to its decoupling methodology and computer support features for semi-automatically generating high-level state-space representations. Petri-nets also are a well established tool for modelling precedence relationships, parallel processing as well as resources' limits and in the last respect they outperform AND/OR graphs. V. Dungern's work inspires a modelling scheme that takes into account not just the product, but the entire environment. Finally, Fox and Kempf underline the necessity to distinguish between decision criteria that can be evaluated during the preparation phase and information which is available only at the time of execution. Separation of the decision process into an offline and an online part is a prerequisite for efficient run-time system performance.

4. TASK-SPACE MODELLING

Our approach owes much to the work of Sanderson/-de Mello: Its foundation is a model of geometry and objects contacts. So far, our representation is restricted to rigid solid bodies (cf. Fig. 4). Due to deficiencies detected when trying to model several different lab applications we made the following modifications and extensions: Our modelling scheme is encompassing as it includes the manipulator, any necessary fixtures and ancillary devices as well as the set of objects traditionally looked at. Apart from that we look at multiple instances of objects and locations where objects can be linked.

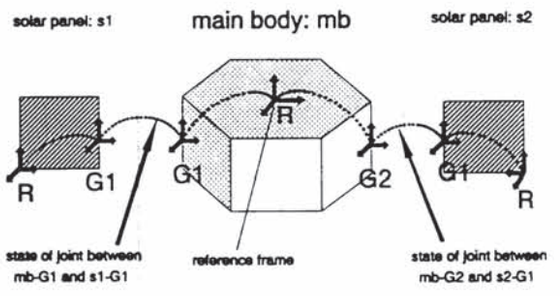

Fig. 4. simplified satellite model

Our geometrical model of solid objects comprises

- volume as a set of convex polyhedrons. The volume model is the base for collision avoiding path planning and object visibility analysis.
- surface as a set of flat polygon patches forming a calculation base for distance measurements.
- 3D-vertices as a reference for stereo or shape-from-motion vision serving identification and localization purposes.

The contact model describes matching contact frames of objects by so-called *anchor points*. They are assigned *key* and *lock* attributes. Key anchor points only match with lock anchor points. Contact classes are modelled according to geometrical and technological considerations, e.g. the class "grips" with a grip source frame (key) located between the jaws of the gripper and any number of grip target frames (locks)

located at suitable positions of objects that have to be manipulated. Contact frames are positioned with respect to an object reference frame by homogeneous transforms, just like the geometrical features mentioned above. Contact characteristics relevant for a semi-automated generation of all feasible high-level states of manipulator plus environment are modelled as contact attributes:

- Permanent or transient contact, e.g. robot joint or robot grip.
- Fixed link or number and kind of mechanical degrees of freedom for automatic checking of motion capabilities.
- Passive or computer controllable joints, e.g. between the drawer and the THEMSE-frame or, again, the robot joints. This information is used for sorting out unfeasible contact transitions in the model generation phase.

At this moment, no attempt is made to deduct small range handling tactics automatically from contact attributes like local degrees of freedom, because according to our practical lab experience these handling tactics often carry the characteristics of manual "dexterities" which are not well enough understood today to be modelled in frame-like structures.

Every individual anchor point is formally represented by a quadruple consisting of object class name, object instance name, anchor point class name and anchor point instance name. Contact instance descriptions comprise two fully specified anchor point quadruples located on different objects. The notion of contact classes is explained here for later reference: *First order contact classes* merely specify the matching anchor point classes, e.g. the set of grips existing within the CPF-experiment scenario. The rest of the places in the contact octuple can be imagined to contain joker symbols. *Second order contact classes* specify the type of anchor points (anchor point classes) and one or both objects (object classes) involved in a contact. The spatial relation between two objects is defined by any second order contact class except for the case of objects featuring multiple instances of that anchor point class which is mentioned in the contact class description, e.g. the THEMSE drawer which has several places for probes. For these cases *third order contact classes* additionally specify the anchor point instance name(s) of one or both of the objects coupled. All of these classes can be viewed as templates for object pairs that may appear in the course of manipulation. Their relevance emerges when trying to subsume many fully specified contact states under one more general state because the kinds of manipulation intended to be used in those states do not depend on all of the state components which would be theoretically available.

Objects are ordered in an inheritance hierarchy of object classes subsuming identical geometrical and contact features. Every object class of lowest abstraction level may have an arbitrary number of instances. This is where combinatorial explosion creeps in in the first place. In the second, every object class may have several anchor point instances of one class, e.g. the THEMSE-drawer has four storage places for probes. The maximum number of different fully specified contacts between two objects of any class equals the product of the number of possible anchor point pairs and the number of different object instance pairs.

Starting from this object model, a formal graph representation is to be found that contains all feasible high-level system states (system=manipulator+environment) as nodes and actions of the manipulator or other controllable devices corresponding to a state transition as arcs. This representation must withstand the following requirements:

- The number of states must not explode when applications grow to a realistic scale.
- All operational sequences feasible and sensible from the offline point of view must be included or reproducible in order to achieve a maximum of autonomy.
- The definition of goals by a human user should be simple. This corresponds to the demand that model states should intuitively correspond to situations in the real application and transitions should correspond to operations on the everyday abstraction level like grip, transport, open, etc.
- Planning and finding alternative execution strategies when trying to reach goals should be as efficient as possible.
- Objects should stay individuals: Assembly fuses objects, disassembly rips them apart again. Since we want to be able to specify the process history of individual objects, e.g. the THEMSE-probes in furnace and freezer, their instance identity must be preserved.

Before elaborating on a semi-automatic generation scheme for all feasible system states some underlying concepts have to be explained:

A number of coupled objects is referred to as a *cluster*. There may be several copies of one cluster depending on the available number of objects in an application. We declare these copies to be instances of one *cluster class* or *cluster template*.
Up to here, any mechanical contact counts, so there has to be a criterion for decoupling clusters since on earth objects do not float around and hence they would be all connected via the ground which contradicts our effort to decouple states. We adhere to the following simple rule: If in a kinematic chain all joints are changed in every conceivable manner and any one object does not move at all, e.g. because it is fixed to the ground like the THEMSE-rack or the robot's base, this object is declared a *terminal* and counted out from the cluster. This exclusion is essential, otherwise such objects might simultaneously belong to several clusters which is undesirable from the point of view of conserving object individuality. The cluster, however, still includes the contact definition with any terminal object in order to distinguish between the various ways of being linked to its surroundings. In an application, there are clusters consisting merely of permanent contacts, e.g. the robot. We call them *permanent clusters*. Since their joints will never be taken apart in the course of action there is no need to represent corresponding system states. Therefore, clusters linked permanently are treated in planning mostly like single objects. We will henceforth refer to both of them collectively as permanent clusters because single objects can be viewed as a special case. On the other hand, clusters containing transient contacts, i.e. contacts which can be taken apart again by the manipulator without demolition, represent a topological state of the objects or subclusters subsumed and are referred to as *transient clusters*.
A special case is introduced by the notion of irreversible contact transitions, e.g. contact initiation by snap fittings or welding or contact demolition by disruptive disassembly. These transitions actually subdivide the state graph into various sections which may be entered or left only via one-ways by the autonomous robot system. Within each of these regions a specific set of permanent clusters is valid. When performing some non reversible assembly on two permanent clusters these are actually fused into a new permanent cluster. The opposite applies to one-way disassembly: one permanent cluster is split up into at least two parts.

The formal state space model we use resembles a Coloured Petri net to a large extent (Baumgarten, 90). It consists of

- Tokens:
  They represent permanent cluster individuals as a list of their contacts and available free anchor points. A token consists of a type (=colour) and object instantiation information. The individuality of tokens is kept by including object instance names in the contact and anchor point definition.

- Places:
  They describe transient cluster contact states. For planning purposes a place is modelled as a container with a limited capacity for tokens of different colours, e.g. the place modelling the robot gripping a probe in the drawer has a capacity of 1 for the drawer, 4 for probes and 1 for the robot. The capacity results primarily from the number of anchor points available but can be limited if objects intersect. This is taken care of in the model generation phase. So, for the planning phase individuality is ignored.
  During execution of a plan consisting of a series of transitions, the actual contact state is registered in a place by associating the variables in a cluster template with object and anchor point instance values from the tokens for reference by the transition operator programs.

- Transitions:
  They connect places and transfer tokens between them. Transitions represent real-life operators or control programs that connect or disconnect subclusters. The description of a transition for planning purposes comprises only the token transfer capacity of its arcs. A transition has $m \geq 1$ input and $n \geq 1$ output arcs each associated with a list of formal token transfer specifiers denoting which subclusters are moved where (type and count) when the transition fires. Firing preconditions are: There must be enough tokens of the necessary colours lying on the input place(s) to saturate the input arc(s) and the capacity of the output places

must not be exceeded by the firing of the transition. A place corresponding to a fully specified contact state must be completely emptied by a transition, e.g. if the robot takes a probe out of the drawer, the tokens for the rest of the probes and the drawer are transferred to the place in which no grip contact with the robot exists. This need not be so for underspecified places but we cannot go into more detail here. The phenomenon of creating or disrupting permanent clusters is represented by fusing or splitting tokens as a side effect of the corresponding transition operator.

This modelling scheme can represent the finest granularity of contact states by working with fully specified contact definitions or third order contact classes if necessary. In these cases the specific place token-capacities are equal to one. Often however, the associated states need not be represented explicitly: If a control strategy that can handle the set of contacts associated with a second order contact class exists, e.g. a gripping strategy that will pick up any probe from the THEMSE drawer no matter at which place it is located inside, the use of second or lower order contact classes in defining a cluster can serve to reduce the number of contact states drastically. Thus, in our example a reduction of the total number of states from approximately 10.000 to the relatively small number of 22 was reached, mainly due to the fact that permutations of probes inside the drawer were modelled not to have impact on manipulation strategies.

In the process of generating all feasible system contact states we proceed as follows:

- Starting from an initial finite set of objects linked by permanent and transient contacts, permanent clusters are automatically identified.
- Free anchor points of different clusters are systematically matched according to their key and lock classes in order to find possible follow-up clusters. Simulating to a certain extent the real procedure, only anchor points located in a kinematic chain consisting of controllable contacts (e.g. the gripper frame of the manipulator) are considered. Places and transitions already defined or equivalent due to the inclusion of contact classes in the place definition are skipped.
- Filters are applied to eliminate unfeasible or undesirable contact states and transitions. Currently these are still carried out mentally by the system designer, but simulation techniques for some of the subproblems, esp. collision avoiding path planning exist. Such filters are:
  - Collision test: There must be a collision-free path from one contact state to the next at the time of execution. Since we do not know the run-time circumstances (number and location of parts and obstacles) at the time of model generation, only a primary feasibility check based on the geometric description of those parts whose contact is changed, is performed. Online a path planner may still fail to find a collision-free path due to the real arrangement of objects but this has to be dealt with at the time of execution by replanning.
  - Stability test: The resulting contact state must not collapse due to external forces because recovering from such a turmoil is currently out of the question.
  - System engineer's desire: The system designer may specify via templates a number of contacts or contact classes that are undesirable within the application. Technological conditions which are not caught by the geometry/contact model may thus be implemented.
- The designer can specify which of the contacts contained in the new place definition are relevant for the (robot) control programs that will be defined for transfers between clusters or modification of internal degrees of freedom. Thus, e.g. by telling the algorithm that for gripping the drawer the presence of probes is irrelevant, the multitude of fully specified clusters that would result from the probes' permutation within the drawer can be prevented from being explicitly generated.
- The contact state is merged with the graph of states already generated. If the place already exists but no transition is defined, a transition link is inserted from the source to the target places. Otherwise, the graph is extended by a new place.

This procedure is applied in a loop visiting all feasible places of the application. The generation process terminates when all possibilities have been exhausted.

An example modelling run for the THEMSE application will clarify our algorithm. For the sake of brevity, we skip the preliminary cluster analysis and

start with a relevant selection of object classes and a reduced set of permanent clusters:

Object classes:
MARS_gripper, drawer, freezer, probe

Permanent clusters:
robot, drawer, furnace, probe_1-4

In the example, the permanent contacts are left out, though they are part of the representation and conditions must be specified for the online planning phase referring to joint states of these permanent contacts, e.g. the drawer must be open in order for the robot to be able to grip a probe.

The lines of the following table defining legal token types that represent permanent clusters should be read as follows: "The robot token has one free anchor point of class grip_source". Anchor points are given in their full quadruple representation. Object and anchor point instances are represented as variables Oi and Aj, jokers are represented by stars (*). Indices are normally counted only within a token type or a place definition. Different indices are used here for improved reader's insight.

token type free anchor point names

T1	MARS_gripper:O1.
	grip_source:grip
T2	THEMSE_drawer:O2.
	grip_target:grip,
	THEMSE_drawer:O2.
	probe_depot:depot_1
	...
	THEMSE_drawer:O2.
	probe_depot:depot_4
T3	THEMSE_furnace:O3
	probe_depot:depot
T4	THEMSE_probe:O4
	grip_target:grip, THEMSE_probe:O4
	probe_base:base

A token is instantiated by substituting the object variables consistently with object names. Every token instance as an individual carries its own name. The following tokens appear within the example:

T1: O1=gripper, name=robot
T2: O2=drawer, name=drawer

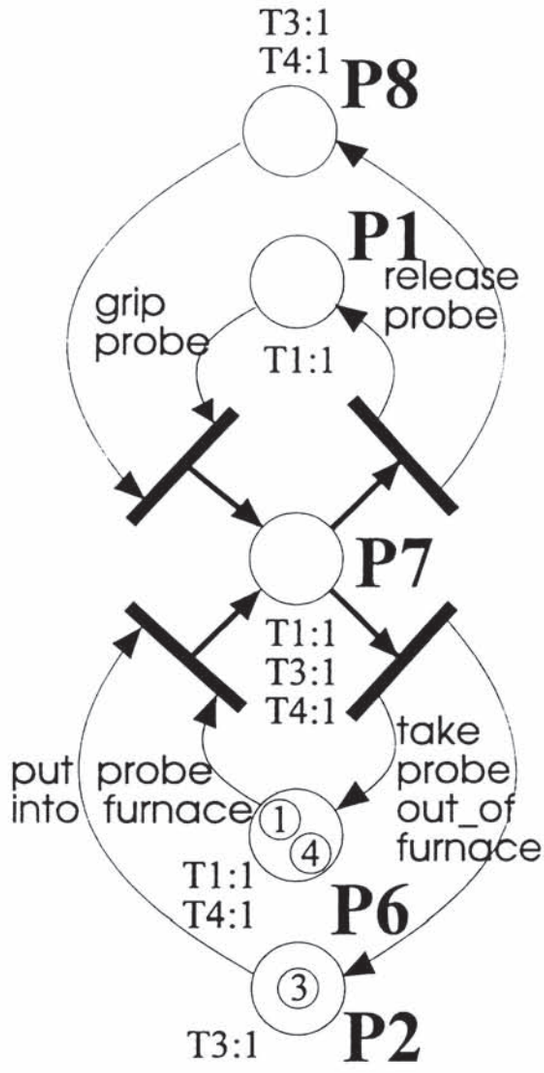

Fig. 5. part of THEMSE state space graph.

T3: O3=furnace, name=furnace
T4: O4=probe_1, name=probe_1
T4: O4=probe_2, name=probe_2
T4: O4=probe_3, name=probe_3
T4: O4=probe_4, name=probe_4

Matching anchor points classes are:
grip_source - grip_target
probe_base - probe_depot

Combining these anchor points among the permanent clusters results in the following topological states or transient clusters represented by places. Remember, that a cluster is defined as a set of specific contacts between objects of a given type. A second order contact class definition for drawer and probes is part of the example. Null entries denoting free anchor points

are represented by the keyword 'void'. Permanent contacts are left out again.

We illustrate the correspondence between certain contact states and our lab system with some of the photos. The complete list of places is given in the appendix. Here we illustrate the topology by a sketch of a part of the state graph (Fig. 5). Ti:j denotes the place's capacity j for token type i, encircled numbers symbolize tokens of a certain type. The token transfer notation of the transitions is left out for simplicity.

List of places:
place P1
The robot is free
place P2
The furnace is free
place P3 (Fig. 2)
The drawer is loaded with probes
place P4 (Fig. 1)
The robot holds the drawer
place P5
The robot holds a probe in the drawer
place P6
The robot holds a probe in free space
place P7 (Fig. 6)
The robot sticks a probe into the furnace
place P8
A probe is in the furnace

5. MODELLING ONLINE RESTRICTIONS

Conditions which have to be evaluated at run time before activating an operator are attached to the transitions. These can be of the following types:

- joint state of the application
- sensor conditions
- time conditions

Within one contact state single controllable joints or groups of joints are associated each with a high-level control program (operator) with is responsible for the manipulation of these joints. Thus a certain locality of the control algorithms is guaranteed which facilitates program generation. A method for programming application-oriented operators and the architecural issues of the run-time system have been published elsewhere (Matthiesen 92), (Adolphs 92). A global list contains those contact states in which a given joint can be manipulated for reference by the online planner.

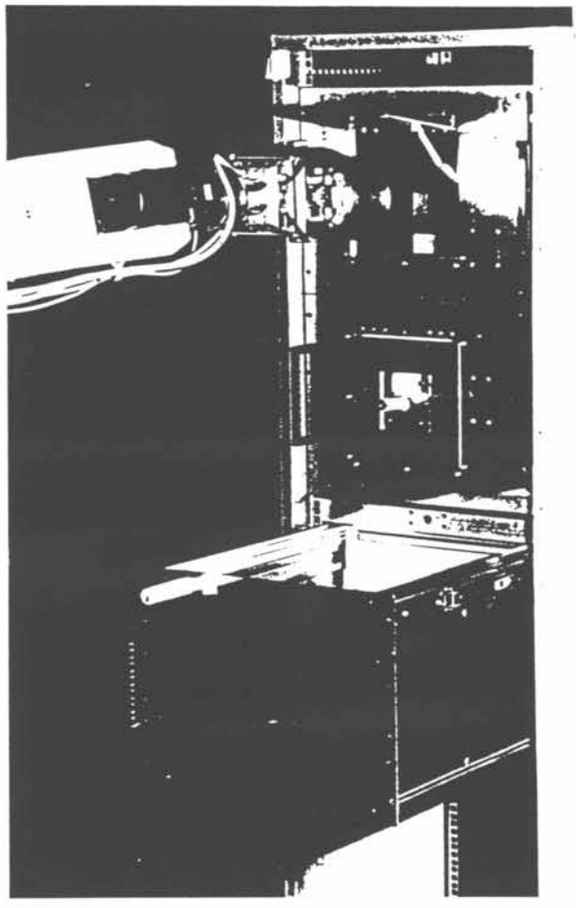

Fig. 6. robot puts probe into furnace

6. CONCLUSION

Building upon experiences gained with traditional task decomposition and work done in the field of assembly automation a powerful improved modelling scheme for highly autonomous robotic handling, maintenance and repair systems has been presented and illustrated by a practical example. We claim to have addressed several aspects of task space modelling that have been underestimated or ignored up to now:

- The entirety of mechanisms and objects used in an application has to be included in an overall work space model.
- All geometrically feasible task sequences need to be represented for the sake of completeness of the representation. Due to the possibly large number of these sequences only an implicit coding seems to be realistic. This can be done by means of a contact state graph which reduces the effort for application programming: Instead of explicit coding of possibly a myriad of task level action sequences only transfer control programmes for

switching between a limited number of contact states need to be written and tested. The task level action sequence can be generated online by adapted graph search mechanisms. This will pave the way for enhanced robot system autonomy.

- The layout of an algorithm for supporting the semi-automatic generation of the new topological contact state models was presented and hints were given how to avoid generating too many states by decoupling and locally ignoring irrelevant parts of the representation..

Work continues in the direction of further improving the modelling and model generation scheme, planning, optimizing and reliably executing complex action sequences.

7. ACKNOWLEDGEMENT

This work was partly performed in a project sponsored by DARA and headed by ISRA Systemtechnik GmbH. The author is grateful for this support.

8. APPENDIX

place P1
The robot is free
MARS_gripper:O1.grip_source:grip-
*:void.grip-target:void

place P2
The furnace is free
THEMSE_furnace:O1.probe_depot:depot-
*:void.probe_base:void

place P3 (Fig. 2)
The drawer is loaded with probes
THEMSE_drawer:O1.probe_depot:A2-
probe:O2.probe_base:base
THEMSE_drawer:O1.probe_depot:A3-
probe:O3.probe_base:base
THEMSE_drawer:O1.probe_depot:A4-
probe:O4.probe_base:base
THEMSE_drawer:O1.probe_depot:A5-
probe:O5.probe_base:base
THEMSE_drawer:O1.grip_target:grip-
*:void.grip_source:void
probe:O2.grip_target:grip-
*:void.grip_source:void
probe:O3.grip_target:grip-
*:void.grip_source:void
probe:O4.grip_target:grip-
*:void.grip_source:void
probe:O5.grip_target:grip-
*:void.grip_source:void

place P4 (Fig. 1)
The robot holds the drawer
THEMSE_drawer:O1.probe_depot:A1-
probe:O2.probe_base:base
THEMSE_drawer:O1.probe_depot:A2-
probe:O3.probe_base:base
THEMSE_drawer:O1.probe_depot:A3-
probe:O4.probe_base:base
THEMSE_drawer:O1.probe_depot:A4-
probe:O5.probe_base:base
THEMSE_drawer:O1.grip_target:grip-
MARS_gripper:O6.grip_source:grip
probe:O2.grip_target:grip-
*:void.grip_source:void
probe:O3.grip_target:grip-
*:void.grip_source:void
probe:O4.grip_target:grip-
*:void.grip_source:void
probe:O5.grip_target:grip-
*:void.grip_source:void

place P5
The robot holds a probe in the drawer
THEMSE_drawer:O1.probe_depot:A1-
probe:O2.probe_base.base
THEMSE_drawer:O1.probe_depot:A2-
probe:O3.probe_base.base
THEMSE_drawer:O1.probe_depot:A3-
probe:O4.probe_base.base
THEMSE_drawer:O1.probe_depot:A4-
probe:O5.probe_base.base
MARS_gripper:O6.grip_source:A5-
probe:O2.grip_target.grip
THEMSE_drawer:O1.grip_target:grip-
*:void.grip_source:void
probe:O3.grip_target:grip-
*:void.grip_source:void
probe:O4.grip_target:grip-
*:void.grip_source:void
probe:O5.grip_target:grip-
*:void.grip_source:void

place P6
The robot holds a probe in free space
MARS_gripper:O1.grip_source:A1-
probe:O2.grip_target.A2
probe:O2.probe_base:base-
*:void.probe_depot:void

place P7 (Fig. 6)
The robot sticks a probe into the furnace
MARS_gripper:O1.grip_source:grip-
probe:O2.grip_target.grip
THEMSE_furnace:O3.probe_depot.depot-
probe:O2.probe_base.base

place P8
A probe is in the furnace
THEMSE_furnace:O1.probe_depot.A1-
probe:O2.probe_base.A2
probe:O2.grip_target:grip-
*:void.grip_source:void

9. REFERENCES

Adolphs, P., Matthiesen. J. (1992). Task-Level Programming with Collision Avoidance for Autonomous Space Robots, Proc. *12th IFAC Symposium on Automatic Control in Aerospace*, Ottobrunn, Germany, 1992.

Baumgarten, B. (1990). *Petri-Netze*, Grundlagen und Anwendungen, BI-Wissenschaftsverlag, 1990.

de Fazio, T.L., Whitney, D.E. (1987). Simplified Generation of all Mechanical Assembly Sequences. *IEEE Journal of R&A*, Vol. RA-3, No. 6, 1987.

de Mello, L.S.H., Sanderson, A.C. (1991). A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences, *IEEE Trans. on R&A* Vol. 7 No. 2, April 1991.

Fox, B.R. (1986). The Implementation of Opportunistic Scheduling. *Intelligent Autonomous Systems Conference*, Amsterdam, Elsevier Publishers, 1986.

Fox, B.R., Kempf, K.G. (1987). Reasoning about Opportunistic Schedules, *1987 IEEE Conf. on R&A*, Raleigh, N.C.

Hirzinger, G. (1993). Das Raumfahrt-Robotik-Experiment ROTEX - Konzepte, Erfahrungen und Perspektiven, *VDI-Berichte 1994, Reihe Steuerung und Regelung von Robotern GMA-Fachtagung, Langen*, 1993.

Hörmann, A. (1989). A Petri net based control architecture for a multi-robot system. *IEEE Int. Symp. on Intelligent Control*, September 1989, Albany, USA.

Kegel, G. (1990). Erhöhung der Autonomie von Robotersystemen durch multisensorielle Informationen und Nutzung einer Wissensbasis. *Dissertation, TH Darmstadt*, Fachgebiet Regelsystemtheorie und Robotik.

Lefebvre D.R., G.N. Saridis (1992). Integrating Robotic Functions and Operator Supervision Using Petri Nets, *Proc. IROS 92*, pp. 20-26, Raleigh, NC, 1992.

Lumia, R., Fiala, J., Wavering, A. (1990). The NASREM robot control system and testbed, *Int. J. of R&A*, Vol. 5, No. 1, 1990.

Matthiesen J., H. Tolle, S. Wienand and E. Ersü (1992). Sensorinformationsverarbeitung und Planung in einem autonomen Steuerungssystem für Wartungs- und Reparaturaufgaben in der Raumfahrt. *Fachtagung Automatisierung*, Dresden.

Miller, D.J., Lennox, R.C., (1992). RIPE: A robot independent programming environment, The International Society for Optical Engineering, *Proc. on Intelligent Robots and Computer Vision - Algorithms and Techniques (1992)* Vol. 1607 p. 518-29.

Putz, P., Mau, K.-D. (1991). Baseline A&R control development methodology definition report prepared for ESA/ESTEC. Doc. No. CT2/CDR/DO/BL, Dornier GmbH, 1991.

Sanderson, A. C., Peshkin, M.A., de Mello, L.S.H (1987). Task Planning for Robotic Manipulation in Space Applications. *Workshop on Space Telerobotics*, JPL, Pasadena, Ca, 1987.

SPACE Magazine (1991). Shepard Press, May/June 1991, pp. 10-12.