AN EXPERIENCE-BASED INTERFACE FOR ABSTRACTING THE MOTION CONTROL OF KINEMATICALLY COMPLEX ROBOTS

Alexander Dettmann[1], **Sebastian Bartsch**[2], **and Frank Kirchner**[1,2]

[1]*Faculty of Mathematics and Computer Science, University of Bremen, 28359 Bremen, Germany, firstname.lastname@uni-bremen.de*

[2]*German Research Center for Artificial Intelligence - Robotics Innovation Center (DFKI RIC), 28359 Bremen, Germany, firstname.lastname@dfki.de*

ABSTRACT

In order to provide higher mobility and to assist humans in building up infrastructure in future extraterrestrial space missions, kinematically complex robots are needed. One key challenge which needs to be addressed is to handle their complex motion control and to make use of their high potential. Utilizing the possibility to achieve various actions even in different ways by tuning manually numerous parameters of the motion control can be very demanding and even unmanageable when also taking communication delay into account.

Thus, the proposed experience-based interface is encapsulating the motion control of complex robots by autonomously mapping application-specific action parameters to robot-specific motion control parameters depending on the current context. Therefore, the robot is using experiences collected from previously executed behaviors. Apart from acquiring experiences during operation of the real robot, they can also be collected in simulation. The possibility to test in low gravity environments makes the latter a valuable tool for increasing the robot's knowledge base for space missions.

The experiments in this paper show that reconfiguring the motion control can be beneficial and that in simulation optimized behaviors can easily be integrated in the experience-based control interface to improve the performance of a robot. In addition, the transferability from simulation to the real system is shown.

Key words: kinematically complex robots, experienced-based control, behavior adaptation.

1. INTRODUCTION

In current and past space missions for planetary exploration, wheeled systems with a passive suspension system have been preferred due to their simplicity and robustness. Even though they provide a reasonable degree of mobility, they always face the risk of getting stuck in a trench, which could result in a complete mission failure. In future missions, a higher degree of mobility will be required to reach places of scientific or ecological interest, e.g., steep craters where water ice can be found [6] or well-illuminated hills for stable power supply. For this reason and to support complex manipulation tasks, e.g., for building up infrastructure, a flexible and active locomotor system will be required.

Kinematically complex robots with an active suspension system like walking robots or hybrid rovers meet the requirements in terms of mobility and flexibility, but demand a more sophisticated motion control. In common control architectures, e.g., the functional reference model [7], a mission layer tries to fulfill mission goals, therefore deriving tasks which lead to a consecutive execution of actions. For instance, the mission goal of analyzing a region of scientific interest results in a set of tasks for reaching and investigating this location. The deliberative navigation and planning layer then generates movement commands, which have to be executed, usually by an reactive action layer. When controlling kinematically complex robots, the latter is realized by a rather complex motion control which in the end generates motor commands.

However, the motion control has more tunable parameters than the task layer could take into consideration. Thus, an operator has to tune most of these manually to produce the desired behavior, which increases the operator load. Taking also the problem of communication delay into account, makes the control of kinematically complex robots almost unmanageable. In addition, the behavior performance is heavily depending on the current context and numerous possibilities exist to achieve the desired action but with different performance, e.g., stability or energy efficiency. Consequently, an appropriate component is required which autonomously maps scenario-specific actions to robot-specific control parameters.

In this paper, a motion control interface is proposed which uses previous experiences to generate a proper configuration for the motion control depending on the current environmental context and the desired action pro-

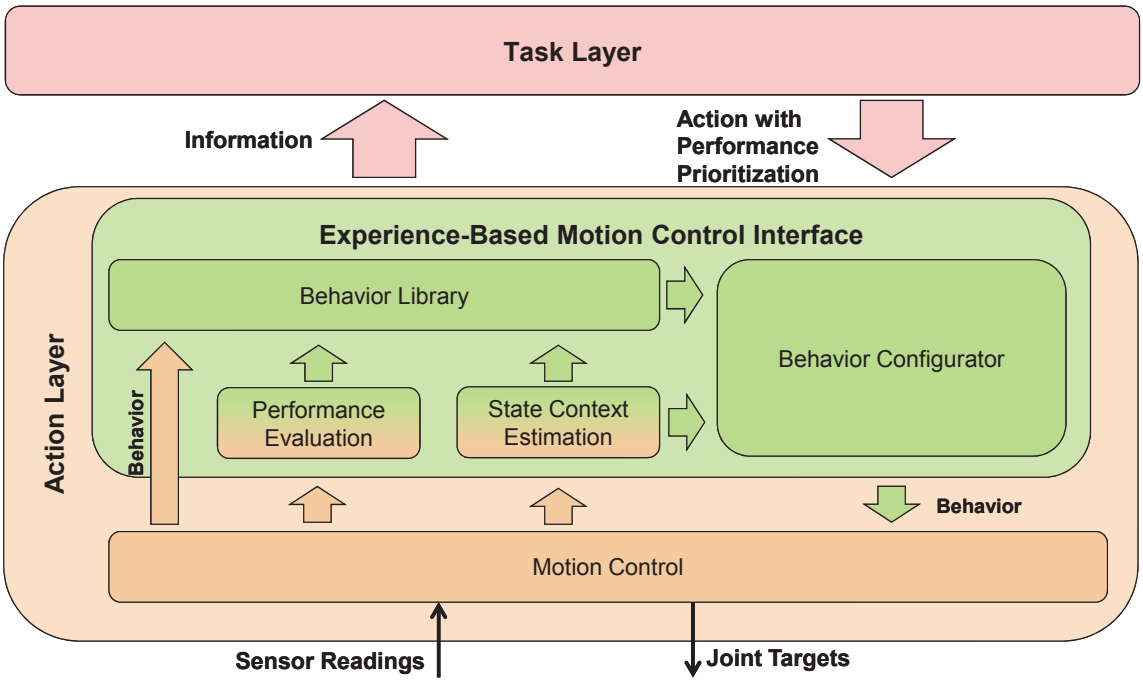


Figure 1. Experience-based interface (green) which extends the motion control in the action layer (orange) to allow scenario-specific commands and to generate additional information for higher layers. Experiences, i.e. performance information of behaviors in various state contexts, are stored in a behavior library *and utilized by a behavior configurator to map scenario-specific actions to robot-specific control parameters of the motion control. The estimation of performance and state context features is based on the robot's processed sensor values and may already be part of the motion control itself.*

vided by the task layer. The interface and its components are described in Section 2. Section 3 describes the needed knowledge base and its utilization. The experiments in Section 4 analyze how a reconfiguration of the motion control can be used to adapt the locomotion behavior of a walking robot. In addition the direct utilization of in simulation optimized behaviors on a real robot is shown. Section 5 briefly concludes the paper and provides an outlook.

2. EXPERIENCE-BASED MOTION CONTROL INTERFACE

Controlling kinematically complex robots is a challenging task because numerous parameters have to be tuned simultaneously to get the desired robot behavior. Whenever the desired action or current environmental condition change, new parameters have to be selected simultaneously leading to high operator load. In order to decouple the operator load from the complexity of the robot's motion control and to maintain the possibility to change the overall robot behavior depending on the current scenario, it is desirable to control a robot by tuning application-specific instead of robot-specific parameters. Consequently, the input for a generic motion control interface is the desired action with an application-specific performance prioritization which then has to be mapped to robot-specific motion control parameters with consideration of the current context (Fig. 1). For instance, a certain path can be followed by focusing on stability, energy efficiency, precision, or a mix of the former.

The idea in the proposed approach is to build up a knowledge base which holds all relevant robot-dependent information about known behaviors and their performance in various state contexts. This so-called *behavior library* (BL), can then be utilized by a behavior configurator to find the required mapping, thus generating different behaviors. Since the robot-dependent information is implicitly stored in the BL, a generic robot-independent mapper can be implemented as an interface to the action-generating task layer encapsulating the robot-specific motion control. In addition, the experienced performance information provided by the BL can be utilized by the task layer to improve planning and navigation or to detect anomalies.

3. EXPERIENCE MANAGEMENT

In the proposed experience-based control approach, the mapping of desired actions to behaviors is based on the current context and the experiences stored in the BL. This section provides an overview of the structure of this knowledge base and its utilization to select appropriate behaviors.

3.1. Behavior Library

The BL consists of behaviors which are defined through a motion control algorithm and parameter sets [5], evaluated contexts, and *behavior evaluations*. Each of the latter holds a reference to one behavior and multiple *context evaluations* which represent the experience stored in the BL (Fig. 2).

The *context evaluations* hold performance information of the referenced behavior for evaluated contexts. Each includes a reference to a visited context which is represented by a list of application-specific context features. In the field of exploration in unstructured terrain, features like slope in longitudinal and lateral direction, maximum step hazard, as well as roughness of the terrain are meaningful characteristics which have to be estimated for the area beneath the robot and the region in direction of movement during the next evaluation period. The latter is the time needed to execute an action, which corresponds here to a step cycle. Through continuous self-evaluation, the robot is measuring its behavior performance. Here two categories of performance features exist. First, action-specific features are needed to characterize the execution of the current action, e.g., longitudinal and lateral velocity, as well as turn rate in the field of locomotion. Second, meta performance features are needed to characterize behaviors which result in the same action but differ in stability, energy efficiency, undesired body vibration, etc. Mean value, mean squared value (needed to calculate the standard deviation of the feature over the number of evaluations), mean of the standard deviation of one evaluation period, and a unit for each performance feature is stored in every *context evaluation*.

Simulated data, e.g., produced by behavior learning algorithms, or real experiences from everyday usage can be stored in the BL. The type of experience needs to be indicated for every *context evaluation* by a setup identifier, because these data must not be merged. Simulated experiences might be needed in certain situations, but after having real experiences collected for the same context, they lose importance due to the simulation reality gap. The number of evaluations indicates the reliability of a *context evaluation*. It is also required when new experiences arrive to update the mean values of existing *context evaluations*.

Whenever a behavior was continuously executed during a whole evaluation period, context and performance features can be averaged, matched to the current behavior, and finally stored in the BL (Fig. 3). To have a limited number of state contexts, each context feature is discretized before storing. Therefore, each feature has an application-specific range and is divided into equally distributed steps where incoming values are mapped to the closest one.

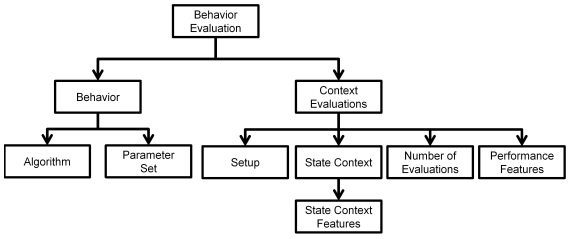


Figure 2. Content of behavior evaluations

3.2. Behavior Configurator

The behavior configurator is the instance which generates a mapping between current context and incoming action to robot-dependent motion control parameters. The BL holds the necessary information to build this mapping. In the proposed approach, a case-based reasoner [4] is used to select the behavior which fulfills the current desired action with the best performance. Therefore, each behavior is rated by computing the following steps.

First, the similarity of each *context evaluation* of every *behavior evaluation* to the current context is determined. Therefore, each feature is normalized according to a robot specific feature range. The mean squared error is used as similarity metric, as it is more sensitive to large differences of one single feature than to small differences of several features compared to the weighted mean absolute error.

Second, the *context evaluation* with the highest state similarity is selected, because its performance information will provide the closest behavior performance approximation. The expected behavior performance is estimated by computing the weighted mean squared error between desired action features and corresponding performance features of the *context evaluation*. Meta performance features are compared with their constant optimal values. The weight for each feature is used to influence its prioritization. This gives the operator or higher layers the opportunity to influence the robots behavior, e.g., the operator may define whether the commanded speed or turn rate has to be followed precisely, stability, energy efficiency, or a mix of the former is of importance in the current scenario.

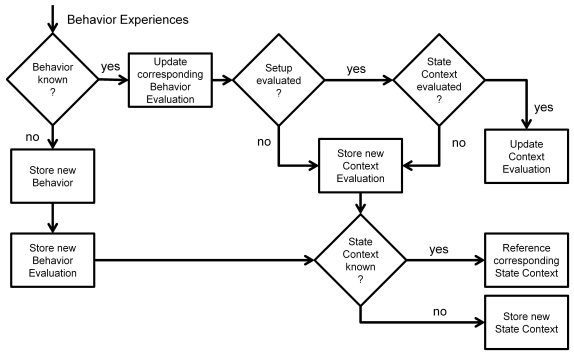


Figure 3. Procedure to store new experiences in the behavior library

Third, the overall behavior score, which is used to select the most suitable behavior, is calculated by weighting the expected behavior performance with the context similarity, because the accuracy of the behavior performance expectation is depending on it.

In space applications, the selection of the putatively best behavior is a preferable approach because it is based on former experiences. In the case of an desired action in an unknown context, indicated by low similarity values, there is always the possibility to gain experiences from experiments with a replicate on earth or with simulation. The latter of course might yield inaccurate expectations due to the simulation reality gap. But especially the possibility to simulate low gravity can lead to more accurate results compared to hardware tests with a replicate. The transferability from simulated data on the real system must be assured, which is exemplary shown in the next section.

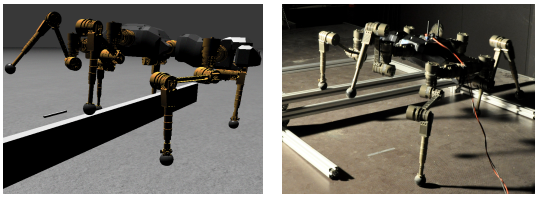
4. EXPERIMENTS

In this section, experiments to evaluate the autonomous selection of appropriate behaviors by using the experience-based interface are described. In the first experiment, the behavior configuration benefit is analyzed, and in the second, the integration of optimized behaviors is tested in simulation. The third experiment investigates the transferability to the real system.

In all experiments, a locomotion scenario was used, where the six-legged walking robot SpaceClimber [2] as an example for kinematically complex robots, had to traverse an obstacle course consisting of a 2.5 m flat plane, followed by an obstacle of 0.2 m height, and another flat plane of 1.15 m towards the finish line (Fig. 4). The needed energy per distance (EPD) was used to evaluate the robot's performance (Eq. 1), where E is the energy needed to pass the course, d the overall distance of 3.65 m, P the average power consumption, and T the time needed to pass the course.

$$EPD = \frac{E}{d} = \frac{P \cdot T}{d} \tag{1}$$

SpaceClimber has 24 degrees of freedom which are controlled by an reactive behavior-based motion control, which has to be parameterized to produce different walking behaviors. The motion control consists of a Cartesian-based central pattern generator to generate trajectories around posture-dependent foot positions. The sum of both are transformed by inverse kinematics into joint space and finally applied to the motors. In addition, the robot movement is stabilized by an elevation and depression reflex (EDR) which crouches the leg if an obstacle is encountered during touchdown phase or stretches the leg when expected ground contact is missing during stance phase. Depending on the parameterizations of the motion control, SpaceClimber produces different locomotion behaviors with distinctive properties.



(a) Simulation (b) Real system

Figure 4. SpaceClimber traversing the obstacle course

4.1. Behavior Adaptation

To be able to traverse the obstacle course, SpaceClimber needs to walk with high body and step height and with activated and sensitive EDR. The parameters (Table 1, col. 4) produce this desired obstacle behavior. When utilizing the autonomous selection of the behavior configurator, a second behavior (plane behavior, Table 1, col. 2) can be used in the plain parts of the obstacle course to improve stability and energy efficiency. The parameters were derived by expert knowledge and were already extensively tested in previous experiments [1].

A simulation[1] was used to evaluate both locomotion behaviors in both contexts (plane and obstacle context, distinguished by step hazard of zero and 20 cm respectively). The experiences were stored in the BL.

As reference, the obstacle behavior was used to traverse the obstacle course. Then the behavior configurator was used to switch the locomotion behavior when encountering the obstacle. Fig. 5 shows the resulting average power consumption (five runs per setup). When starting on the flat plane, the behavior configurator chose the plane behavior because it was more energy-efficient in the past. It is visible that the robot is saving energy until reaching the obstacle compared to the obstacle pattern. When detecting the obstacle (at t=35 s), the increased step hazard caused the behavior configurator to choose the obstacle behavior since it had a higher performance score for this context. This led to an increased power consumption, but the obstacle was passed. Since time was saved in the plane part of the obstacle course, the robot reached the finish line earlier (at t=93 s) compared to the setup where only the obstacle behavior was used. The resulting EPD with reconfiguration was 26% less than without reconfiguration (Fig.6).

[1] MARS (Machina Arte Robotum Simulans) is a simulation and visualization tool for developing control algorithms and designing robots. It consists of a core framework containing all main simulation components, a GUI, OpenGL-based visualization, and a physics core that is currently based on ODE (`http://www.ode.org`). It is available at `https://rock-simulation.github.io/mars/`

Table 1. Parameters of used walking behaviors (all other parameters are kept at their default value and are omitted for clarity)

behavior	plane expert knowledge	plane optimized	obstacle expert knowledge	obstacle optimized
body shift x in mm	0	100	0	90
body height in mm	250	275	450	440
speed x in mm/s	50	50	25	25
length factor [0...1]	0.5	0.5	1.0	0.6
lift time in ms	200	240	400	840
shift time in ms	1400	1560	1000	1120
touchdown time in ms	200	1600	1000	1360
phase shift [0...1]	0.0	0.0	1.0	0.6
swing amplitude in mm	100	100	300	280
EDR sensitivity [0...10]	2	2	5	4
EDR touchdown delay in ms	120	120	80	40
EDR max crouch in mm	-100	-100	-250	-150
EDR max stretch in mm	100	100	0	110

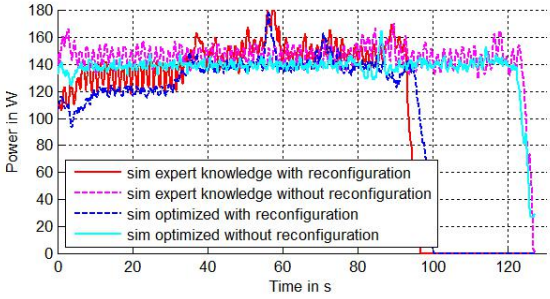


Figure 5. Average power consumption while traversing the obstacle course in simulation

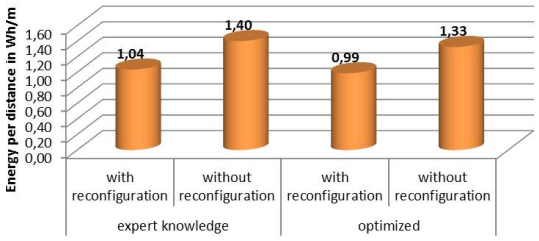


Figure 6. Resulting energy per distance for accomplishing the obstacle course in simulation

4.2. Integration of Optimized Behaviors

The proposed experience-based motion control interface provides the possibility to apply optimization results on the fly to influence the behavior of the robot. Therefore, two further locomotion behaviors were learned in simulation. CMA-ES [3] was used to optimize the parameters from the expert knowledge derived behaviors, except speed, which was kept constant to maintain comparability. SpaceClimber had to traverse a plane and an obstacle covered ground, respectively. The EPD metric was evaluated after a specified time limit and used to rate the individuals. The optimized parameters (Table 1, col. 3 and 5) indicate that a longer touchdown phase reduces ground impact and therefore increases energy efficiency. It is also noticeable that the body is shifted slightly to the front to improve the stability. The resulting parameter behaviors as well as all information about their performance in these specific contexts were stored in the BL.

Because the proposed behavior configurator can directly utilize new experiences from the BL, SpaceClimber was able to choose from four possible locomotion behaviors without restarting its motion control. The optimized behaviors were automatically chosen during the obstacle course due to their better EPD value in both encountering contexts (plane ground and obstacle). The resulting EPD value for reaching the finish line with reconfiguration was 4% less than using the expert knowledge behaviors. This is because SpaceClimber walked smoother with less power peaks (Fig. 5).

4.3. Transferability to the Real Robot

Finally, the transferability to the real system was analysed by repeating the tests with the real robot (Fig. 7). The results show that the time needed to complete the obstacle course between each run had more fluctuations (indicated by the stepwise decrease at the end of each averaged power curve). In addition, the nominal power consumption and resulting EPD was smaller compared to the simulation results, indicating the simulation reality gap. However, changing the parameters of the locomotion control by switching from plane to obstacle behavior reduced the EPD metric by 18% (Fig. 8). In addition, the optimized behaviors outperformed the ones derived by expert knowledge by 15%. Due to less interaction force between foot and ground during touchdown phase, slippage was reduced which resulted in completing the required distance in less time. This is underlined by the fact that the optimized obstacle pattern without reconfiguration finished earlier than the expert knowledge

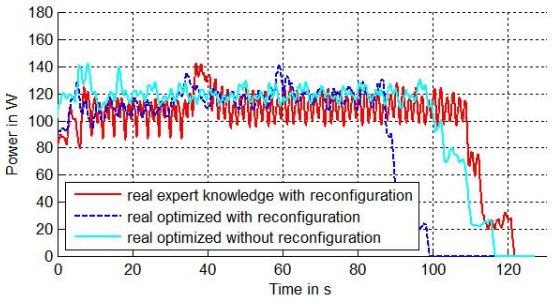


Figure 7. Average power consumption of the real system while traversing the obstacle course

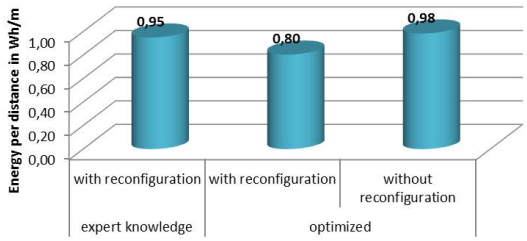


Figure 8. Resulting energy per distance for accomplishing the obstacle course with the real system

patterns with reconfiguration. The experiments with the real SpaceClimber show that the difference between optimized and expert knowledge patterns was larger than in simulation. But it was also noticeable when using the optimized obstacle pattern, that SpaceClimber almost tipped over indicating a too large longitudinal body shift. This example shows that optimized behaviors in simulation are not optimal on the real system. Nevertheless, they provide a good basis for further online adaptations on the system.

5. CONCLUSION AND OUTLOOK

The proposed experience-based motion control interface autonomously maps scenario-specific actions to robot-specific parameters while incorporating the current context. A case-based reasoner is used to select the putatively best behavior based on experiences stored in a BL. Higher layers can command a desired action and prioritize scenario-specific performance features to influence the robot behavior instead of tuning robot-specific parameters. Consequently, the motion control is encapsulated, generating an abstract interface which facilitates the usage of different kinds of kinematically complex robots.

By continuously collecting new experiences during operation, the robot implicity learns because the BL is growing, thus gaining confidence and incorporating system wearout over lifetime. In addition, especially useful in extraterrestrial exploration, when encountering a situation where no adequate behavior is known, a simulation or a replica on earth can be used to find a suitable behavior which then can be stored together with collected experiences in the BL to increase the robots competence online.

The experimental results show that adapting the control layer to a changing context can be beneficial. Taking a locomotion scenario as an example, the best known behaviors concerning energy efficiency were selected by the robot while traversing an obstacle course. In addition, the operator or higher layers did not have to manage all available locomotion parameters. Another advantage of the proposed approach is shown by the direct usage of optimization results in the robot control. The better expected performance of the optimized behaviors, based on the experiences stored in the BL, led to their selection, and finally proved to increase the robot's performance. The transferability from simulation to reality was shown, which motivates the approach to support space missions with simulation results.

Since in simulation optimized behaviors are not optimal on the real system, further work has to be done to adapt their parameters online which finally leads to a life-long learning approach. In addition, the generation and utilization of larger BLs has to be analyzed in more complex scenarios.

ACKNOWLEDGMENTS

The presented work was carried out in the project LIMES, a collaboration between the DFKI Robotics Innovation Center and the University of Bremen, funded by the German Space Agency (DLR, Grant numbers: 50RA1218, 50RA1219) with federal funds of the Federal Ministry of Economics and Technology (BMWi) in accordance with the parliamentary resolution of the German Parliament.

REFERENCES

[1] Bartsch, S. 2013, PhD thesis, University of Bremen

[2] Bartsch, S., Birnschein, T., Römmermann, M., et al. 2012, Journal of Field Robotics, 29, 506

[3] Hansen, N. & Ostermeier, A. 2001, Evolutionary Computation, 159

[4] Kolodner, J. L. 1992, Artificial Intelligence Review, 6, 3

[5] Langosz, M., Quack, L., Dettmann, A., Bartsch, S., & Kirchner, F. 2013, Int. Conf. on Climbing and Walking Robots, 495

[6] NASA.gov – LCROSS. 2009, LCROSS Impact Data Indicates Water on Moon, `http://www.nasa.gov/mission_pages/LCROSS/main/prelim_water_results.html`

[7] Putz, P. & Elfving, A. 1991, Control Techniques 2, Tech. Rep. CT2/CDR/DO/BL, Dornier GmbH, issue: 1.1, ESTEC Contract 9292/90/NL/JG (SC)