The Cognitive Controller:
A Hybrid, Deliberative/Reactive Control Architecture for Autonomous Robots

Faisal Qureshi[1], Demetri Terzopoulos[1,2], and Ross Gillett[3]

[1] Department of Computer Science, University of Toronto,
Toronto, ON M5S-3G4, Canada
`{faisal,dt}@cs.toronto.edu`
[2] Courant Institute, New York University,
New York, NY 10003-6806, USA
[3] System Design Department, MD Robotics Limited,
Brampton, ON L6S-4J3, Canada
`rgillett@mdrobotics.ca`

Abstract. The Cognitive Controller (CoCo) is a new, three-tiered control architecture for autonomous agents that combines reactive and deliberative components. A behaviour-based reactive module ensures that the agent can gracefully handle the various real-time challenges of its environment, while a logic-based deliberative module endows the agent with the ability to "think ahead", performing more complex high-level tasks that require planning. A plan execution and monitoring module establishes an advisor-client relationship between the deliberative and reactive modules. We demonstrate CoCo in the context of space robotics—specifically the control of a vision-guided robotic manipulator that can autonomously capture a free-flying satellite in the presence of unforeseen mission anomalies.

1 Introduction

Humans are sophisticated autonomous agents that are able to function in complex environments through a combination of reactive behaviour and deliberative reasoning. Motivated by this observation, we propose a hybrid robotic control architecture, called the Cognitive Controller or CoCo, which combines a behaviour-based reactive component and a logic-based deliberative component. CoCo is useful in advanced robotic systems that require or can benefit from highly autonomous operation in unknown, time-varying surroundings, such as in space robotics and planetary exploration systems, where large distances and communication infrastructure limitations render human teleoperation exceedingly difficult. In our implementation, CoCo operates an autonomous, vision-guided robotic agent designed to service satellites in orbit. In realistic laboratory test scenarios, we subject CoCo to anomalous operational events, forcing its deliberative component to modify existing plans in order to achieve the mission goals. CoCo demonstrates the capacity to compensate in important ways for the absence of a human operator.

In our space robotics application, we are specifically interested in the task of safely capturing a satellite, transporting it to a service bay, performing the desired service, and releasing it back into orbit. From the perspective of the software responsible for controlling the sensory apparatus and robotic manipulator, the first step is the most interesting and challenging. Once the satellite is secured, we can assume a static workspace and handle the remaining steps using more primitive scripted controllers [15].

Our autonomous robotic agent competently captures a satellite while handling anomalous situations such as sensor failures, hardware failures, and aberrant satellite behaviour. It gathers information about its environment through an imperfect vision system that is sensitive to lighting conditions and motion. CoCo determines whether the vision system is performing reliably, which is a non-trivial task that involves explaining current environmental events. If the explanation is unexpected, then either the vision system is failing or the environment is being erratic, and the agent must take corrective actions.

CoCo draws upon prior work in AI planning, plan-execution, mobile robotics, ethology, and artificial life. We review relevant prior work in the next section. In our technical development, Section 3 details the CoCo architecture and Section 4 describes its implementation. Section 5 presents results from CoCo's application. Section 6 concludes the paper.

2 Related Work

Early attempts at designing autonomous robotic agents employed a sense-model-plan-act (SMPA) architecture with limited success [9,12,13]. The 1980s saw the emergence of a radically different, ethological approach to robotic agent design, spearheaded by Brooks' subsumption architecture [4] and the mantra "the world is its own best model". Most notable among modern ethological robots is Sony Corporation's lovable robotic dog, AIBO [2], which illustrates both the strengths (operation in dynamic/unpredictable environments) and the weaknesses (inability to reason about goals) of the strict ethological approach. Hybrid architectures, containing both deliberative and reactive components, first appeared in the late 1980s. A key issue is how to interface the two layers. AuRA (Autonomous Robot Architecture) binds a set of reactive behaviours to a simple hierarchical planner that chooses the appropriate behaviours in a given situation [3]. In SSS (Servo Subsumption Symbolic), a symbolic planner controls a reactive module [6]. In ATLANTIS, the deliberative module advises the reactive behaviours [1,9].

### 2.1 Relationship to Previous Hybrid Architectures

Like ATLANTIS, CoCo consists of both deliberative and reactive modules, featuring a reactive module that performs competently on its own and a deliberative module that guides the reactive module. CoCo was originally inspired by experience implementing self-animating graphical characters for use in the entertainment industry. In particular, our approach was motivated by the "virtual merman" of Funge *et al.* [8],

which extends a purely behavioural control substrate [14] with a logic-based deliberative layer employing the situation calculus and interval arithmetic in order to reason about discrete and continuous quantities and plan in highly dynamic environments.

CoCo differs in the following ways: First, its deliberative module can support multiple specialized planners where deliberative, goal-achieving behaviour is the result of cooperation between more than one planner. The ability to support multiple planners makes CoCo truly taskable. Second, CoCo features a powerful and non-intrusive scheme for combining deliberation and reactivity, which heeds advice from the deliberative module only when it is safe to do so. Here, the deliberative module advises the reactive module through a set of motivational variables. Third, the reactive module presents the deliberative module with a tractable, appropriately-abstracted interpretation of the real world. The reactive module constructs and maintains the abstracted world state in real-time using contextual and temporal information.

3 Cognitive Controller Architecture

CoCo is a three-tiered architecture that consists of deliberative, reactive, and plan execution and monitoring modules. The deliberative module implements a high-level symbolic planning system. The reactive module implements a low-level behaviour-based controller with supporting perception and memory subsystems (Fig 1). At the intermediate level, the plan execution and monitoring module enforces an advisor-client relationship between the deliberative and reactive modules.
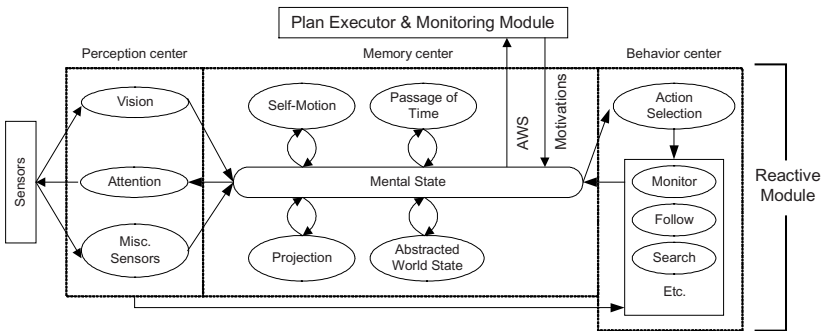
Fig. 1. The three functional components of the reactive module, each consisting of asynchronous processes.

3.1 The Reactive Module

CoCo's reactive module is a behaviour-based controller that is responsible for the immediate safety of the agent. As such, it functions competently on its own and runs at the highest priority. At each instant, the reactive module examines sensory information supplied by the perception system, as well as the motivational variables whose

values are set by the deliberative module, and it selects an appropriate action. Its selection thus reflects both the current state of the world and the advice from the deliberative module. The second responsibility of the reactive module is to abstract a continuum of low-level details about the world and present a tractable discrete representation of reality within which the deliberative module can effectively formulate plans.

CoCo's reactive module comprises perception, memory, and behaviour components. This functional decomposition simplifies the design of the reactive module in order to implement basic behaviours, such as tracking, following, station-keeping, and capturing. These functional units are implemented as a set of asynchronous processes.

Perception Center. The perception center manages the vision system, which consists of long, medium, and short range vision modules. The long range module performs a search that returns an ongoing estimate of the satellite's pose once it has been detected. The estimate of the satellite's pose from the long range module initializes the medium range module, which is active from five meters to around two meters and uses model-based stereo-vision algorithms to track the satellite. The short range module takes over when the distance to the satellite is less than two meters. It tracks the satellite using visual features on the satellite's docking interface. The perception center decides which vision modules to activate and how to combine the information from these modules depending on their characteristics, such as processing times, operational ranges, and noise. An alpha-beta tracker filters out the noise from the vision readings. The perception center incorporates an attention mechanism that gathers information relevant to the current task, such as the status of the satellite chaser robot, the docking interface status, and the satellite's attitude.

Behaviour Center. The behaviour center manages the reactive module's behavioural repertoire. This by no means trivial task involves arbitration among behaviours. At each instant, the action selection mechanism chooses an appropriate high level behaviour by taking into account the current state of the world and the motivations provided by the deliberative module. We have implemented six such behaviours for the satellite servicing application—*search*, *monitor*, *approach*, *align*, *contact*, and *avoid*. The chosen action then activates lower level supporting behaviours, as necessary. The action selection mechanism chooses a behaviour that is relevant to the goals of the agent while ensuring its safety. The reactive module will heed the advice of the deliberative module only when it is safe to do so.

Memory Center. The memory center manages the short-term memory of the agent. It holds the relevant sensory information, motivations, state of the behaviour controller, and the abstracted world state. The robot observes its environment egocentrically. External objects change their position with respect to the robot as it moves. Behaviour self-motion constantly updates the internal world representation to reflect the current position, heading, and speed of the robot, otherwise the confidence in the accuracy of the world representation should decrease with time in the absence of new readings from the perception center (Fig. 2). The memory center filters out unnecessary details from the detailed sensory information consumed by the reactive module and it

generates the abstracted world state (Fig. 3; Table 1) which expresses the world symbolically for use by the deliberative module.
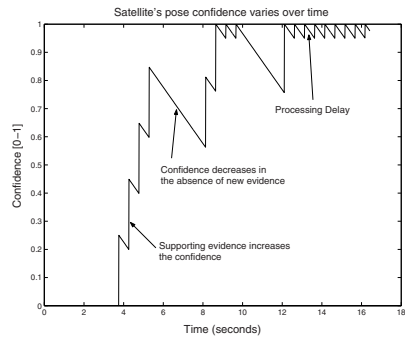
Fig. 2. Confidence in the satellite's pose decreases in the absence of sensory evidence from the vision system. How the confidence in a particular feature decreases depends on the feature (e.g., the confidence in the position of a dynamic object decreases more rapidly than that of a static object) and the penalty associated with acting on the wrong information.
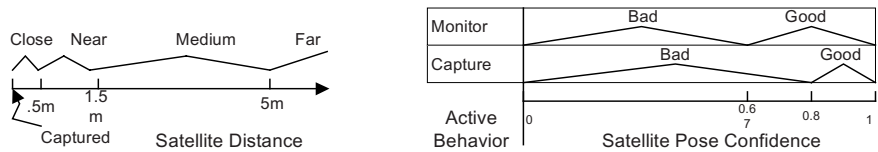
Fig. 3. The abstracted world state represents the world symbolically. For example, the satellite is either *Captured*, *Close*, *Near*, *Medium*, or *Far*. The conversion from numerical quantities in the memory center to the symbols in the abstracted world state takes into account the current state of the agent.

Table 1. The abstracted world state for the task of satellite servicing. The choice of fluents for describing the abstracted world state depends on the active task.

fStatus	fSatPosConf	fSatPos	fSatSpeed	fError
fLatch	fSatCenter	fSatAlign	fSensor	fSatContact

3.2 The Deliberative Module

The deliberative module endows our agent with planning ability, enabling it to perform high level tasks that are too difficult to perform without planning. The deliberative module maintains a set of planners, each with its own knowledge base, planning strategy, and task list. Each planner sees the world at an abstract level, which makes reasoning tractable, as opposed to planning among a myriad of low-level details. The reactive module determines the lowest level of abstraction for a planner, explicitly through the abstracted world state, and implicitly through its implemented behaviours (these behaviours—a.k.a., grounded actions—form the basis of the plans generated by

the deliberative module). For any application, it is essential to choose the right level of abstraction in advance (Table 1).

Upon receiving a top-level command from the operator in the ground station, the deliberative module selects an appropriate planner (by using the task lists associated with the planners), updates the planner's world model using the abstracted world state, and activates the planner. Only one planner is active at a time in order to avoid un-wanted interactions between actions proposed by different planners. The planner computes a plan as a sequence of zero (when the planner cannot generate a plan) or more actions, passes this plan to the deliberative module, which forwards it to the plan execution and monitoring module. Each action of an executable plan contains execution instructions, such as which behaviour to activate, and specifies its pre- and post-conditions.

Table 2. Grounded actions for the GOLOG planner - these actions are directly executable by the reactive module

aTurnon	aSearch	aGo	aLatch	aErrorHandle
aSensor	aMonitor	aAlign	aSatAttCtrl	aContact

Table 3. The deliberative module transforms the current task into a world state, called the de-sired world state. The Golog planner then constructs a plan to transform the current world state into the desired world state. Upon execution, this plan will fulfill the task.

| |
|---|
| fStatus(off) & fLatch(unarmed) & fSensor(medium,off) & fSensor(short,off) & fSatPos(medium) & fSatPosConf(no) & fSatCenter(no) & fAlign(no) & fSatSpeed(yes) & fSatAttCtrl(on) & fSatContact(no) & fError(no,X) |
| **Initial (current) state** |
| fStatus(on) & fLatch(armed) & fSensor(medium,off) & fSensor(short,on) & fSatPos(zero) & fSatPosConf(yes) & fSatCenter(yes) & fAlign(yes) & fSatSpeed(yes)& fSatAttCtrl(off) & fSatContact(yes) & fError(no,X) |
| **Goal (desired)** |
| aTurnon(on) -> aSensor(medium,on) -> aSearch(medium) -> aMonitor -> aGo(medium,near,vis) -> aSensor(short,on) -> aSensor(medium,off) -> aAlign -> aLatch(arm) -> aSatAttCtrl(off) -> aContact -> aGo(zero,park,no) |
| **The plan that transforms the initial state into the goals state** |

A Planner for the Satellite Capturing Task.   Symbolic logic provides an appropriate level of abstraction for developing high level planners that elegantly express abstract ideas. We use GOLOG [11], an extension of the situation calculus, to develop a planner for the satellite capturing task. GOLOG uses logical statements to maintain an internal world state (fluents) and to describe what actions an agent can perform (primitive action predicates), when these actions are valid (precondition predicates), and how these actions affect the world (successor state predicates). GOLOG provides high level constructs, such as if-then-else and non-deterministic choice, to specify complex procedures that can model an agent and its environment. The logical foundations of GOLOG enable us to prove plan correctness properties, which is desirable.

The deliberative module updates the values of fluents from the abstracted world state and executes the GOLOG program. The execution generates a plan, such as the one in Table 3, whose purpose is to transform the current state of the world to the goal state—a state in which the chaser robot has securely captured the satellite. Depending on the script, this could be followed by more explicitly-scripted operations to service the satellite or it could continue to be directed through servicing operations by CoCo.

3.3 Plan Execution and Monitoring Module

The plan execution and monitoring module interfaces the deliberative and reactive modules. It initiates the planning activity in the deliberative module when the user has requested the agent to perform some task, when the current plan execution has failed or is otherwise unable to meet the desired post-conditions, when the reactive module is stuck, or when it encounters a non-grounded action that requires further elaboration. The execution is controlled through pre- and post-conditions specified by the plan's actions. Together, these conditions encode plan execution control knowledge. At each instant, active actions that have either met or failed their postconditions are deactivated. Next, un-executed actions whose preconditions are satisfied are activated (Fig. 5).

We divide actions into three categories: (i) grounded actions (directly executed by the reactive module; see Table 2), (ii) conditional actions (affect the choice of the next action to be executed), and (iii) non-grounded actions (require further elaboration, such as the user command *dock*). The plan executor and monitoring module can handle all three categories of actions, so it can execute linear, conditional, and hierarchical plans. It can also execute multiple actions, and hence multiple plans, simultaneously; however, it assumes that the plan execution control knowledge for these plans within the deliberative module will have prevented race conditions, deadlocks, and any undesirable side affects of concurrent execution.
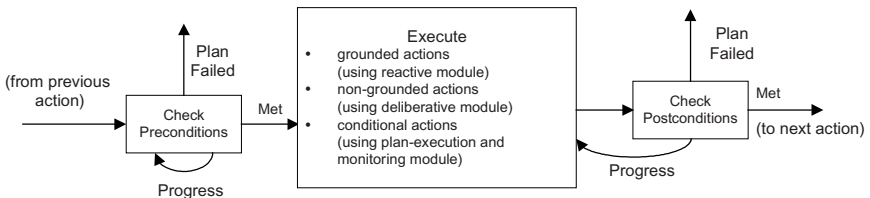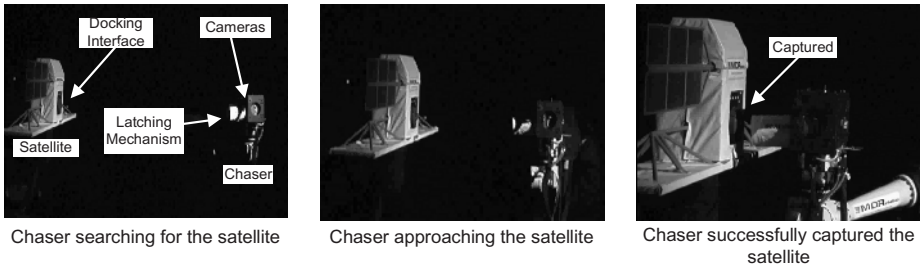

Fig. 5. The plan executor and monitoring module sequentially executes each action. It checks the current action preconditions until they succeed or fail, checking preconditions and postconditions before each transition.

4 Implementation

To facilitate the design, development, and debugging of CoCo, we implemented an Autonomous Agent Design and Simulation Testbed (AaDST)—a software testbed for developing autonomous agents in virtual environments. In our application, AaDST

contains a physics-based model of the chaser robot, a kinematically controlled satellite that can exhibit realistic satellite motion by following prescribed trajectories, and a virtual sun whose position affects lighting conditions. The virtual chaser has synthetic visual sensors that model the characteristics of the actual vision system, including processing delays, noise characteristics, and lighting effects. The virtual chaser also provides motor commands that are similar to those of the physical robot.

The physical setup consisted of MDRobotics Ltd. proprietary "Reuseable Space Vehicle Payload Handling Simulator", comprising two Fanuc robotic manipulators and their associated control software (Fig. 6). One robot with the stereo camera pair mounted on its end effector acts as the chaser. The other robot carries a grapple fixture-equipped satellite mockup and generates realistic satellite motion. The robot lab is specially designed with black walls, ceiling and floor to mimic the lighting conditions of the space environment—very little ambient light, strong sunlight and harsh shadows.

Chaser searching for the satellite   Chaser approaching the satellite   Chaser successfully captured the satellite

Fig. 6. The chaser robot captures the satellite using vision in simulated orbital lighting conditions.

First, we implemented the controller for the virtual chaser—it takes input from the synthetic vision system and issues motor commands to the virtual chaser. Next, we modified it to control the physical robot by adding the necessary communication modules that enable it to obtain sensory data from the actual vision system and issue motor commands to the physical robot over the local intranet. As we had hoped, this transition from controlling a virtual chaser in a simulated environment to controlling the physical robot required only minimal changes to CoCo. We merely had to modify the low-level behaviours in the reactive module, because the dynamic responses of the physical robot were not identical to those of the virtual chaser.

5 Results

Typically, the ground station would upload a mission plan to the on-orbit sequencer that would then pass to CoCo the relevant commands (Fig. 7). However, as a benefit of the level of autonomy that CoCo affords this system, only a single high-level command, *dock(time-out_in_seconds, max_attempts)*, needs to be uploaded from the ground station in order to perform the free-flyer satellite capture operation in which we are interested. We tested CoCo in the simulated environment and also on the physical robots, and it met its requirements; i.e., safely capturing the satellite while

handling numerous anomalous situations. We performed 800 test runs in the simulated environment and over 25 test runs on the physical robots. CoCo never jeopardized the safety of the satellite or the chaser. For each run, we randomly created error conditions (see Table 4), such as vision system failure and hardware failure. CoCo's chaser robot gracefully handled all of them, successfully capturing the satellite whenever it was able to recover from these failures. In situations where it could not resolve the error, it safely parked the manipulator and informed the ground station of its failure.
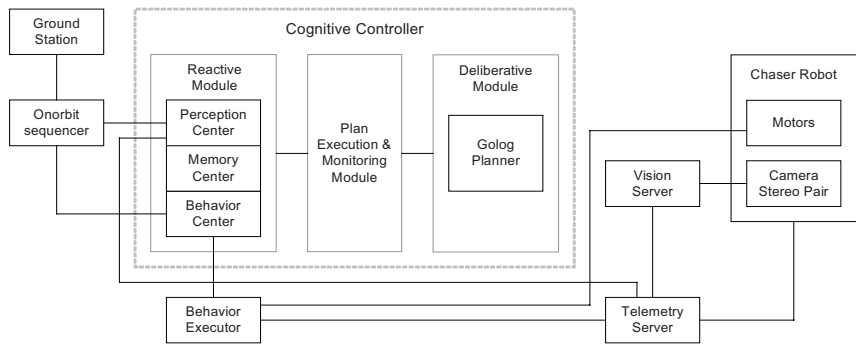
Fig. 7. CoCo receives commands from the ground station through the on-orbit sequencer and controls the chaser robot.

Table 4. CoCo handled these error conditions that were randomly generated during various test runs.

Vision System Errors	Hardware Errors
Camera failure	Grapple fixture errors
Self shadowing	Joint errors (critical)
Solar glare	Satellite's attitude control error
Failed transitions between vision module	

6 Conclusion

Toward the design of intelligent, hybrid controllers for autonomous agents, CoCo advocates general principles that address the critical challenge of combining reactivity and deliberation for autonomous robots inhabiting complex, dynamic environments. CoCo features a behaviour-based reactive system and a logic-based deliberative system, and it provides an elegant scheme for combining the two through a plan execution and monitoring system. Safety is ensured by a fully competent reactive module that can override when necessary the suggestions of the deliberative module. The CoCo architecture is also taskable, because its deliberative layer allows multiple planners specialized to the various tasks the agent must perform. Our domain of application, space robotics, requires an autonomous robot to deal with a myriad of operational events and anomalies that require real-time response. Within this domain,

we have successfully implemented a controller that meets these challenges. Another benefit of our approach is that the uploaded command set from the operator is greatly simplified due to the ability of the on-orbit system to decide among the various operational details autonomously. The proposed architecture will apparently be useful in developing intelligent hybrid controllers for autonomous agents in other domains.

References

1. P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of American Association of Artificial Intelligence*, San Mateo, CA, 1987.
2. R. C. Arkin, M. Fujita, and R. Takagi T., Hasegawa. Ethological modeling and architecture for an entertainment robot. *ICRA*, 2001.
3. R. C. Arkin. Integrating behavioural, perceptual and world knowledge in reactive navigation. *Journal of Robotics and Autonomous Systems* (**1-2**), June 1990, 6, 1990.
4. R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, **RA-2**, 1986.
5. W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of AAAI/IAAI*, 1998.
6. J. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992.
7. R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, **5**(2), 1971.
8. J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Proceedings of the Conference on Computer Graphics (Siggraph99)*, 1999. ACM Press.
9. E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mo- bile robots. In *Proceedings of the National Conference on Artificial Intelligence*, 1992.
10. D. Hdhnel, W. Burgard, and G. Lakemeyer. GOLEX—Bridging the gap between logic (GOLOG) and a real robot. In *Proc. of the 22nd German Conf. on AI (KI-98)*, 1998.
11. Y. Lesp´erance, R. Reiter, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, **31**, 1997.
12. E.D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, **5**(2), 1974.
13. E.D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, 1975.
14. D. Terzopoulos, X. Tu, and R.Grzeszczuk. Artificial fishes: Physics, locomotion, perception, behaviour. *Artificial Life*, **1**(4), 1994.
15. R. Gillett, M. Greenspan, L. Hartman, E. Dupuis, D. Terzopoulos. Remote Operation with Supervised Autonomy (ROSA). In *Proceedings of the 6th International Conference on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2001),* 2001.