The Cognitive Controller: A Hybrid, Deliberative/Reactive Robot Control Architecture

Content Areas: autonomous agents, cognitive robotics, applications, AI architectures

Abstract

The Cognitive Controller (CoCo) is a three-tiered control architecture for autonomous agents that combines reactivity and deliberation. A behavior-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 complex high level tasks that require planning. A plan execution and monitoring module, which interfaces the deliberative and reactive modules, establishes an advisor-client relationship between them. We demonstrate CoCo in the context of space robotics; specifically the control of a robotic manipulator designed to perform satellite servicing tasks. The manipulator autonomously captures a free-flying satellite using only vision based sensors while handling anomalous situations, such as vision failures, aberrant satellite behavior, and hardware failures.

1 Introduction

Humans are sophisticated autonomous agents that are able to function in their complex environment by combining reactive and deliberative capabilities. Motivated by this observation, we propose a robotic control architecture that we call *The Cognitive Controller*, or CoCo, which is aimed at advanced applications in space robotics. CoCo combines a behavior-based reactive module and a logic-based deliberative module in order to control an autonomous robotic agent designed to service satellites in orbit.

Safety is a top priority for any space robotic agent. This is a difficult challenge for an autonomous agent that inhabits a complex, in-orbit environment which defies accurate and detailed modeling within tight time constraints, limited perception skills, and inaccurate motor skills. The solution to this fundamental problem is more than a matter of brute force compute power. Among other things, better control architectures are needed, and CoCo is a step in that direction.

In our application, we are specifically interested in the task of safely capturing a satellite, bringing it to the service bay, performing the desired service, and re-inserting it into the correct orbit. From the perspective of the control software and the physical components of the autonomous agent, which include the robotic manipulator and sensory apparatus, the first step is the most interesting and challenging, as we can assume a static workspace once the satellite is secured and a scripted controller can handle the remaining steps. We demonstrate an agent that can safely and autonomously capture a free-flying satellite by using vision based sensors.

Our autonomous agent competently captures a satellite while handling anomalous situations such as sensor failures, hardware failures, and aberrant satellite behavior. It gathers information about its environment through a vision system that is, at best, fragile. It determines whether the vision system is performing reliably, which is a non-trivial task that involves explaining what is happening in the environment. If this explanation is contrary to expectations, then either the vision system is misbehaving or the environment has been erratic. In either case, the agent must take corrective actions.

CoCo draws upon prior work in the fields of planning, plan-execution, mobile robotics, ethology, and artificial life. We review relevant prior work in the next section. Section 3 develops the CoCo architecture, Section 4 describes its implementation, and Section 5 presents results from its application. Section 6 concludes the paper.

2 Prior Work

Early attempts at designing autonomous robotic agents employed a sense-model-plan-act (SMPA) architecture with limited success [Fikes and Nilsson, 1971; Sacerdoti, 1974; 1975]. The 1980s saw the emergence of a radically different, ethological approach to robotic agent design, spearheaded by Brooks' subsumption architecture [Brooks, 1985] and the mantra "the world is its own best model" [Brooks, 1991]. An ethological robot is controlled by a collection of processes (behaviors) that handle their own sensing and react in a manner aimed at satisfying goals. The overall agent behavior emerges from the interaction of the processes with one another and with the external world, as demonstrated in insect-like and more conventional mobile robots capable of performing simple tasks. Most notable among ethological robots is Sony Corporation's lovable robotic dog, AIBO [Arkin *et al.*,], which illustrates both the strengths and the weaknesses of the strict ethological approach: Behavior-based agents can function in dynamic, unpredictable environments for prolonged periods without human intervention. Unfortunately, however, a behavior-based

agent cannot reason about the goals that are implicit in its design.

Hybrid architectures, containing both deliberative and reactive components, first appeared in the late 1980s. The common theme among these architectures is that the reactive module deals with the continuous, detailed, and uncertain aspects of the world, and it mediates between the world and the deliberative layer. A key issue is how to interface the two layers. AuRA (Autonomous Robot Architecture) binds a set of reactive behaviors to a simple hierarchical planner that chooses the appropriate behaviors in a given situation [Arkin, 1990]. In SSS (Servo Subsumption Symbolic), a symbolic planner controls a reactive module [Connell, 1992]. In ATLANTIS, the deliberative module advises the reactive behaviors [Agre and Chapman, 1987; Gat, 1992]. RHINO is a notable example of a hybrid architecture by virtue of its ability to guide visitors through museum tours in Bonn, Germany [Burgard *et al.*, 1998]. It uses the *situation calculus* based language GOLOG [Lespérance *et al.*, 1997], to plan its actions. A sequencer expands each action of the plan into low-level behaviors in a reactive module, which uses probabilistic schemes for sensor fusion, map building, and navigation [Hdhnel *et al.*,].

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. Like the developers of RHINO, we believe that logic is an appropriate medium for describing the high level planning activity of an agent. Although CoCo was designed for controlling physical robots, it was originally inspired by experience implementing self-animating graphical characters for use in the entertainment industry. In particular, we were inspired by the "virtual merman" [Funge *et al.*, 1999], a sophisticated simulated physical agent, which achieves intelligent control through a hybrid architecture. This architecture extends a purely behavioral control substrate [Tu and Terzopoulos, 1994] with a logic-based deliberative layer that employs the situation calculus and interval arithmetic in order to reason and plan in highly dynamic environments.

CoCo differs from the previous agent architectures in the following ways: First, its deliberative module can support multiple specialized planners, each responsible for some of the tasks that the agent can perform. Deliberative, goal-achieving behavior is the result of cooperation between these planners. A plan execution and monitoring module arbitrates the planners. We can extend the repertoire of the agent simply by adding another task-specific planner to the deliberative module. This is easier than attempting to develop a monolithic planner that handles all tasks. Second, CoCo features a powerful and non-intrusive scheme for combining deliberation and reactivity. It follows advice from the deliberative module only when it is safe to do so, by treating the advice as *motivation* when deciding what to do next. Third, the reactive module, which comprises perception, memory, and behavior components, presents the deliberative module a tractable, appropriately abstracted interpretation of the real world.

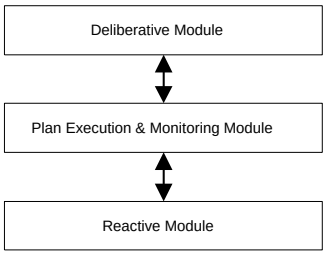


Figure 1: CoCo is a three-tiered hybrid control architecture.

Class	Input	Output	Functional Unit
1	External	External	Behavior Center (Reflex actions)
2	External	Internal	Perception Center (Sensing)
3	Internal	External	Behavior Center (Motor commands)
4	Internal	Internal	Memory Center (Mental state maintenance), Behavior Center (High level behaviors), and Perception Center(Sensor Fusion)

Table 1: Four classes of asynchronous processes (behaviors) constitute the reactive module.

3 Cognitive Controller Architecture

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

3.1 The Reactive Module

CoCo's reactive module is a behavior-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 its perception system and motivational variables set by the deliberative module, and it selects an appropriate action. Its selection thus reflects both the current state of the world and 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 behavior components. This functional decomposition simplifies the design of the reactive module in order to implement basic behaviors, such as tracking, following, station-keeping, and capturing. These functional units are implemented as a set of asynchronous processes, grouped into four classes (Table 1).

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 estimate of the satellite's pose. The satellite's pose estimate from

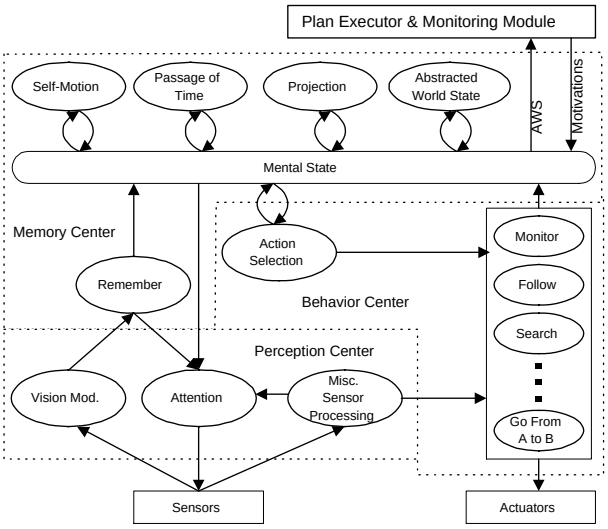


Figure 2: The three functional components of the reactive module, each consisting of asynchronous processes.

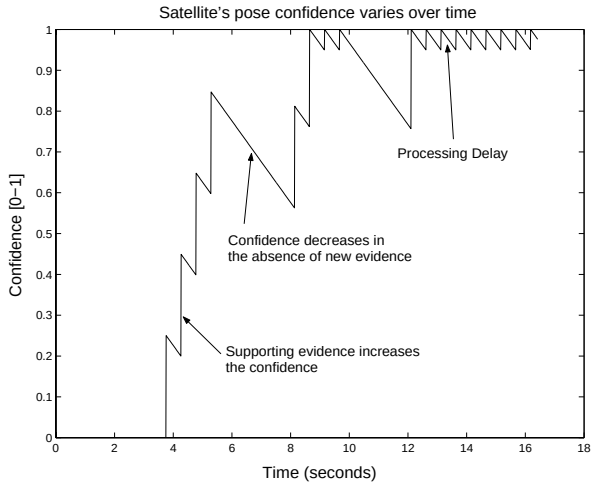


Figure 3: Confidence in the satellite's pose decreases in the absence of supporting evidence from the vision system.

fStatus	fSatPosConf	fSatPos	fSatSpeed
fLatch	fSatCenter	fSatAlign	fSensor
fError	fSatContact		

Table 2: The abstracted world state for the task of satellite servicing. The choice of fluents for describing the abstracted world state depends upon the active task.

the long range module initializes the medium range module, which is active from five meters to around two meters, that 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 markers on the the satellite's docking interface. The perception center decides which vision modules to activate and how to combine the information from these modules depending upon 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, such as chaser robot status, docking interface status, and satellite's attitude control status, which is relevant to the current task.

Behavior Center

The behavior center manages the reactive module's behavioral repertoire. This by no means a trivial task involves arbitration among behaviors. At each instant, the action selection mechanism chooses an appropriate high level behavior by taking into account the current state of the world and the motivations. We have implemented six such behaviors for the satellite servicing application: *search*, *monitor*, *approach*, *align*, *contact*, and *avoid*. The chosen action then activates lower level supporting behaviors, as necessary. The current state of the world takes precedence over the motivations, which essentially means that the reactive module will follow the advice from the deliberative module only when the conditions are favourable. When no motivation is available from the deliberative module, the action selection mechanism simply chooses a behavior that is the most relevant, usually one that ensures the safety of the agent.

Memory Center

The memory center manages the short-term memory of the agent. It holds the relevant sensory information, motivations, state of the behavior controller, and the abstracted world state. The robot sees its environment egocentrically. External objects change their position with respect to the agent as it moves. Behavior *self-motion* constantly updates the internal world representation to reflect the current position, heading, and speed of the robot. In the absence of new readings from the perception center, the confidence in the world state should decrease with time (Fig. 3). 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. The reactive module requires detailed sensory information; whereas, the deliberative module deals with abstract features about the world. The memory center filters out unnecessary details from the sensory information and generates the abstracted world state (Fig. 4) which expresses the world symbolically.

3.2 The Deliberative Module

The deliberative module endows our agent with planning ability, such that it can 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 and planning strategy. Each planner sees the world at an abstract level, which makes reasoning tractable compared to any attempt to formulate plans in the presence of myriads of low-level details; i.e, the world behaves much more predictably at higher levels of abstraction. The lowest level of abstraction for a planner is determined by the reactive module, explicitly

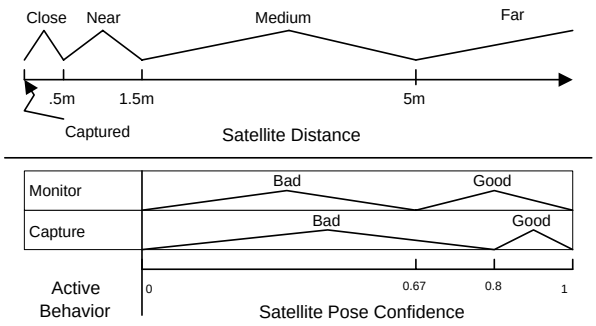


Figure 4: The abstracted world state represents the world symbolically. For example, the satellite is either Captured, Close, Near, Medium, or Far. The conversion from numerical quantities in the memory center to the symbols in the abstracted world state takes into account the current state of the agent. For example, translation from numberical value of satellite pose confidence to the symbolic value Good or Bad depends upon the active behavior—for behavior *Monitor*, satellite position confidence is Good when it is greater than 0.67; whereas for behavior *Capture* satellite position confidence is Good only when it is greater than 0.8.

through the abstracted world state, and implicitly through the high level behaviors that it implements (these behavior form the basis—grounded actions—of the plans generated by the deliberative module). For any application, it is essential to choose the right level of abstraction.

On receiving a request from the plan execution and monitoring module, the deliberative module selects an appropriate planner, updates the planner's world model using the abstracted world state, and activates the planner. The planner computes a plan, which is a sequence of zero (when the planner cannot come up with a plan) or more actions, to the deliberative module which then forwards it to the plan execution and monitoring module. Each action of an executable plan contains execution instructions, such as which behavior to activate, and specifies its pre and post conditions.

A Planner for the Satellite Capturing Task

Symbolic logic provides the right level of abstraction for developing high level planners that elegantly express abstract ideas. We use GOLOG [Lespérance *et al.*, 1997], which is an extension of the *situation calculus*, to develop a planner for the task of satellite capturing. GOLOG uses logical statements to maintain an internal world state (fluents) and 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 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 fluents' values from the abstracted world state, and executes the GOLOG program. The execution generates a plan, such as the one in Table 3,

Current State	The Plan	Goal State
fStatus(off)	aTurnon(on)	fStatus(on)
fLatch(unarmed)	aSensor(medium,on)	fLatch(armed)
fSensor(medium,off)	aSearch(medium)	fSensor(medium,off)
fSensor(short,on)	aMonitor	fSensor(short,on)
fSatPos(medium)	aGo(medium,near,vis)	fSatPos(zero)
fSatPosConf(no)	aSensor(short,on)	fSatPosConf(yes)
fSatCenter(no)	aSensor(medium,off)	fSatCenter(yes)
fAlign(no)	aAlign	fAlign(yes)
fSatSpeed(yes)	aLatch(arm)	fSatSpeed(yes)
fSatAttCtrl(on)	aSatAttCtrl(off)	fSatAttCtrl(off)
fSatContact(no)	aContact	fSatContact(yes)
fError(no,X)	aGo(zero,park,no)	fError(no,X)

Table 3: A linear plan generated by the GOLOG planner. The fluents in column 1 describe the current (initial) state of the world. The plan in column 2 transforms this state to the goal state that is represented by the fluents in column 3.

aTurnon	aMonitor	aLatch	aContact
aSensor	aGo	aSatAttCtrl	
aSearch	aAlign	aErrorHandle	

Table 4: Grounded actions for the GOLOG planner—these action are directly executable by the reactive module.

whose purpose is to transform the current state of the world to the goal state—a state where the chaser robot has securely captured the satellite.

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, 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 post-conditions are deactivated, next un-executed actions whose preconditions are satisfied are activated (Fig. 5).

We divide actions into three categories—grounded actions (these actions can be directly executed by the reactive module, such as the action *aMonitor*), 2) conditional actions (these actions affect the choice of the next action to be executed, such as a *sensing action*), and 3) non-grounded actions (these 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 will prevent race conditions, deadlocks, and any undesirable side affects of concurrent execution.

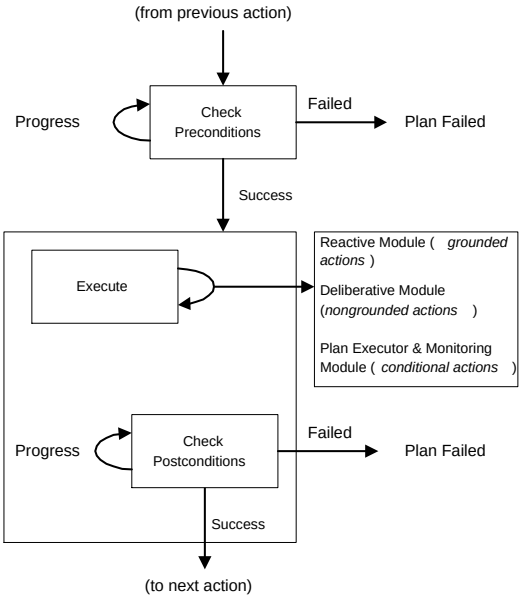


Figure 5: The plan executor and monitoring module sequentially executes each action. It checks the current action's preconditions until they succeed or fail; if they succeed, it goes into the current action's execution/postcondition-check loop, wherein it activates the current action's execution code until the postconditions either succeed or fail. Upon success, it proceeds to the next action.

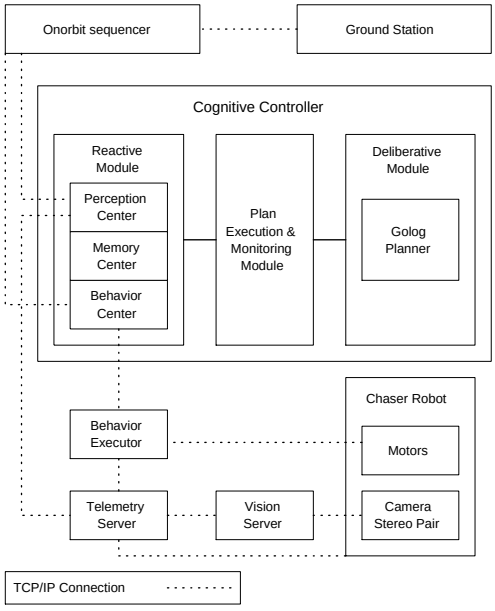


Figure 6: CoCo receives commands from the ground station through the on-orbit sequencer and controls the chaser robot.

4 Implementation

To design, develop, and debug CoCo, we implemented an *Autonomous Agent Design and Simulation Testbed* (AaDST)—a 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 provided by the physical robot.

The physical setup consisted of MDRobotics, Inc., proprietary "Remote Payload Handling System", comprising two Fanuc robotic manipulators and the associated control software. One robot with the grapple fixture and camera stereo pair mounted on its end effector acts as the chaser. The other robot carries a dummy model of the satellite and exhibits realistic satellite motion. The robot lab is specially designed to mimic the lighting conditions of the space environment—very little ambient light, harsh shadows, and solar glare.

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 a physical robot required only minimal changes to CoCo. We had merely to modify the low level behaviors in the reactive module, because the dynamics of the physical robot were not identical to those of the virtual chaser.

5 Results

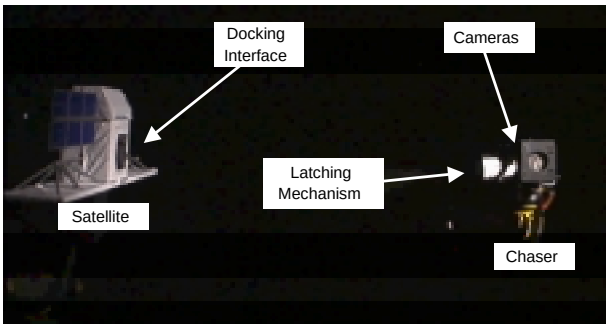
Typically, the ground station would upload the mission plan to the on-orbit sequencer that would then pass to CoCo the relevant commands. Currently, CoCo only provides a single high level command `dock(time-out_in_seconds, max_attempts)`. CoCo attempts to complete the requested task and returns the result to the ground station via the on-orbit sequencer. 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 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 5), such as vision system failure and hardware failure, and the chaser gracefully handled all of them. It successfully captured the satellite whenever it was able to recover from these failures—it re-acquired the satellite by re-inializing the vision system, or it corrected the hardware problem by requesting human assistance. In situations where it could not resolve the error, it safely parked the manipulator and informed the ground station of its failure.

Vision System Errors	Hardware Errors
Camera Failure	Grapple Fixture Error
Self Shadowing	Joints Error (critical)
Solar Glare	Satellite's Attitude Control Error
Failed transition between vision modules	

Table 5: CoCo handled these error conditions that were randomly generated during various test runs.

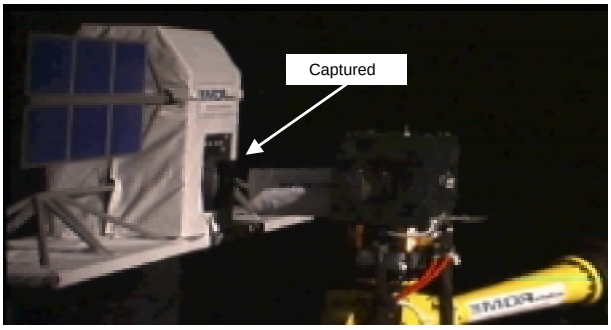
6 Conclusion

How best to combine reactivity and deliberation is a critical challenge in designing intelligent, hybrid robot controllers. CoCo provides general guidelines for developing hybrid controllers for autonomous agents inhabiting complex, dynamic environments. CoCo features a behavior based reactive system and a logic based deliberative system, and provides an elegant scheme for combining the two through a plan execution and monitoring system. The CoCo architecture is also taskable, because its deliberative layer allows multiple planners specialized to the various tasks the agent must perform. Our space robotics domain of application requires an autonomous robot to deal with, e.g., unreliable sensors, inaccurate motor controllers, real-time operation, hard safety constraints, and tasks that require planning. Within this domain, we have successfully implemented a controller that meets these challenges. It appears that the proposed architecture will be useful for developing intelligent hybrid controllers for autonomous agents in other domains.



Chaser searching for the satellite


Chaser approaching the satellite



Chaser successfully captured the satellite

Figure 7: The chaser robot captures the satellite using vision in harsh lighting conditions like those in orbit.

References

[Agre and Chapman, 1987] 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. Morgan Kaufmann.

[Arkin *et al.*,] R. C. Arkin, M. Fujita, and R. Takagi T., Hasegawa. Ethological modeling and architecture for an entertainment robot.

[Arkin, 1990] R. C. Arkin. Integrating behavioural, perceptual and world knowledge in reactive navigation. *Journal of robotics and autonomous systems(1-2), June 1990*, 6, 1990.

[Brooks, 1985] R. A. Brooks. A robust layered control system for a mobile robot. Technical Report AIM-864, Artificial Intelligence Lab, Massachusetts Institute of Technology, September 1985.

[Brooks, 1991] R. A. Brooks. Intelligence without representation. *Artificial Intelligence, January 1991(1-3)*, 47, 1991.

[Burgard *et al.*, 1998] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *AAAI/IAAI*, 1998.

[Connell, 1992] J. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1992.

[Fikes and Nilsson, 1971] 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.

[Funge *et al.*, 1999] J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In Alyn Rockwood, editor, *Proceedings of the Conference on Computer Graphics (Siggraph99)*, N.Y., August8–13 1999. ACM Press.

[Gat, 1992] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of National Conference on Artificial Intelligence*, 1992.

[Hdhnel *et al.*,] D. Hdhnel, W. Burgard, and G. Lakemeyer. GOLEX—Bridging the gap between logic (GOLOG) and a real robot.

[Lespérance *et al.*, 1997] Y. Lespérance, R. Reiter, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31, 1997.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2), 1974.

[Sacerdoti, 1975] E. D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, 1975.

[Tu and Terzopoulos, 1994] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH '94*, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, ACM Press, jul 1994. ISBN 0-89791-667-0.