Machine Vision and Applications
DOI 10.1007/s00138-007-0085-z

ORIGINAL PAPER

Intelligent perception and control for space robotics

Autonomous Satellite Rendezvous and Docking

Faisal Qureshi · Demetri Terzopoulos

Received: 24 July 2006 / Accepted: 23 February 2007

Abstract We present a space robotic system capable of capturing a free-flying satellite for the purposes of on-orbit satellite servicing. Currently such operations are carried out either manually or through discrete-event scripted controllers. The manual approach is costly and exposes astronauts to danger, while the scripted approach is tedious and brittle. Consequently, there is substantial interest in performing these operations autonomously, and the work presented here is a step in this direction. To our knowledge, ours is the only satellite-capturing system that relies on vision and cognition to deal with an uncooperative satellite. Our innovative system combines visual perception (object identification, recognition, and tracking) with high-level reasoning in a hybrid deliberative/reactive computational framework. The reasoning module, which encodes a model of the environment, performs deliberation to control the perception pipeline—it guides the vision system, validates its performance, and suggests corrections when vision is performing poorly. Furthermore, it advises the behavioral controller to carry out its tasks. Reasoning and related elements, among them intention, context, and memory, are responsible for the robustness and reliability of the overall system. We demonstrate our prototype system controlling a robotic arm that autonomously captures a free-flying satellite in a realistic laboratory setting that faithfully mimics on-orbit conditions.

F. Qureshi (✉)
Department of Computer Science, University of Toronto,
10 Kings College Road, Room 3304,
Toronto, ON M5S-3G4, Canada
e-mail: faisal@cs.toronto.edu

D. Terzopoulos
Computer Science Department, University of California,
Los Angeles, 4732 Boelter Hall, Los Angeles,
CA 90095-1596, USA
e-mail: dt@cs.ucla.edu

1 Introduction

Since the earliest days of the computer vision field, researchers have struggled with the challenge of effectively combining low-level vision with artificial intelligence (AI). Some of the earliest work involved the combination of image analysis and symbolic AI to construct robots capable of autonomous, task-driven operation [31,36]. These early attempts met with limited success, in part because the vision problem is hard [43]. The focus of vision research then shifted from vertically-integrated vision systems to low-level vision modules. Currently available low- and intermediate-level vision algorithms are sufficiently competent to support subsequent levels of processing. Consequently, there now is renewed interest in high-level vision, which is necessary if we are to realize autonomous robots capable of performing useful talks in dynamic, unpredictable environments.

In this paper, we report on research in the domain of space robotics. In particular, we design a visually guided robotic system capable of autonomously performing the challenging task of capturing a non-cooperative, free-flying satellite for the purposes of on-orbit satellite servicing. Our innovative system features object recognition and tracking combined with high-level symbolic reasoning within a hybrid deliberative/reactive computational framework, called the Cognitive Controller (CoCo).

The work reported herein was done in collaboration with MD Robotics, Ltd.(currently MDA Space Missions), a Canadian company that has supported human space flight since the early 1980s through advanced robotic systems, such as the Space Shuttle's Canadarm and the Mobile Servicing System for the International Space Station. The company, which undertakes extensive R&D projects in-house and through collaborations with universities and research institutions, regards autonomy as a necessary capability for future space

robotics missions. The reported work was done as part of the ROSA (Remote Operation with Supervised Autonomy) project [17], which arose from this long-term vision. ROSA's goal is to advance the state of the art (operator commands and discrete-event scripted control) by making possible a remote system that can perform decisions in real time within a dynamic environment using high-level artificial intelligence techniques combined with robotic behavioral control and machine vision.

1.1 On-orbit satellite servicing

On-orbit satellite servicing is the task of maintaining and repairing a satellite in orbit. It extends the operational life of the satellite, mitigates technical risks, reduces on-orbit losses, and helps manage orbital debris. Hence, it is of interest to multiple stakeholders, including satellite operators, manufacturers, and insurance companies [13,28]. Although replacing a satellite is more cost-effective in some cases, on-orbit servicing is critical for more expensive satellite systems, such as space-based laser and global positioning system constellations, or for one-of-a-kind systems like the Hubble telescope, which costs $2.5 billion. As early as the 1980s, the National Aeronautics and Space Administration realized the importance of on-orbit servicing for protecting their assets in space [28].

Currently, on-orbit satellite servicing operations are carried out manually; i.e., by an astronaut. However, manned missions are usually very costly and there are human safety concerns.[1] Furthermore, it is currently impracticable to carry out manned on-orbit servicing missions for satellites in geosynchronous equatorial orbit (GEO), as the space shuttle can not reach them. Unmanned, tele-operated, ground-controlled missions are infeasible due to communications delays, intermittence, and limited bandwidth between the ground and the servicer. A viable alternative is to develop the capability of autonomous on-orbit satellite servicing.

1.2 Autonomous satellite rendezvous and docking

A critical first phase of any on-orbit satellite servicing mission, be it for the purpose of refueling, reorbiting, repairing, etc., involves rendezvousing and docking with the satellite. From the perspective of the software responsible for controlling the sensory apparatus and robotic manipulator, the rendezvousing 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 [17]. Most national and international space agencies realize the important role of autonomous rendezvous and docking (AR&D) operations in future space missions and now have technology programs to develop this capability [19,45].

Autonomy entails that the on-board controller be capable of estimating and tracking the pose (position and orientation) of the target satellite and guiding the robotic manipulator as it (1) approaches the satellite, (2) maneuvers itself to get into docking position, and (3) docks with the satellite. The controller should also be able to handle anomalous situations, which might arise during an AR&D operation, without jeopardizing its own safety or that of the satellite. Another requirement that is desirable for space operations is that of *sliding autonomy*, where a human operator can take over the manual operation of the robotic system at any level of the task hierarchy [10,40]. Sliding autonomy enhances the reliability of a complex operation and it expands the range and complexity of the tasks that a robotic system can undertake.

1.3 Contributions

In this paper, we develop a visually-guided AR&D system and validate it in a realistic laboratory environment that emulates on-orbit lighting conditions and target satellite drift. To our knowledge, ours is the only AR&D system that uses vision as its primary sensory modality and can deal with an uncooperative target satellite. Other AR&D systems either deal with target satellites that communicate with the servicer craft about their heading and pose, or use other sensing aids, such as radar and geostationary position satellite systems [33].

Our system features CoCo, a new hybrid robot control framework that combines a behavior-based reactive component and a logic-based deliberative component. CoCo draws upon prior work in AI planning, plan-execution, mobile robotics, ethology, and artificial life. Its motivation comes from the fact that humans, who are sophisticated autonomous agents, are able to function in complex environments through a combination of reactive behavior and deliberative reasoning. We demonstrate that CoCo is useful in advanced robotic systems that require or can benefit from highly autonomous operation in unknown, non-static surroundings, especially in space robotics where large distances and communication infrastructure limitations render human teleoperation exceedingly difficult. In a series of realistic laboratory test scenarios, we subject our CoCo AR&D system to anomalous operational events, forcing its deliberative component to modify existing plans in order to achieve mission goals. The AR&D controller demonstrates the capacity to function in important ways in the absence of a human operator.

Our AR&D prototype meets the operational requirements by controlling the visual process and reasoning about the events that occur in orbit. The system functions as follows:

[1] The Hubble telescope captured the imagination of the public during its highly publicized repair missions, which were carried out by astronauts. By some estimates, these repairs cost taxpayers as much as $12 billion.

Fig. 1 Images acquired during satellite capture. The left and center images were captured using the shuttle bay cameras. The right image was captured by the end-effector camera. The *center image* shows the arm in hovering position prior to the final capture phase. The shuttle crew use these images during satellite rendezvous and capture to locate the satellite at a distance of approximately 100 m, to approach it, and to capture it with the Canadarm—the shuttle's manipulator

First, captured images are processed to estimate the current position and orientation of the satellite (Fig. 1). Second, behavior-based perception and memory units use contextual information to construct a symbolic description of the scene. Third, the cognitive module uses knowledge about scene dynamics encoded using the *situation calculus* to construct a scene interpretation. Finally, the cognitive module formulates a plan to achieve the current goal. The scene description constructed in the third step provides a mechanism to verify the findings of the vision system. Its ability to plan enables the system to handle unforeseen situations.

The performance of the system results from the cooperation of its components, including low-level visual routines, short and long-term memory processing, symbolic reasoning, and the servo controllers of the robotic arm used to capture the satellite. Competent, reliable low-level visual routines are essential for meaningful higher-level processing. Consequently, the AR&D system depends upon the reliable operation of the low-level object recognition, tracking, and pose-estimation routines. The AR&D system is able to handle transient errors in the low-level visual routines, such as momentary loss of tracking, by using short-term memory facilities. However, it cannot accomplish the task when the low-level vision algorithms altogether fail to track the satellite, in which case the high-level routines abort the mission. Stable servoing routines that account for the manipulator's dynamics are vital for a successful AR&D mission. Hence, the AR&D prototype developed here assumes that the robotic arm can servo competently under the guidance of the higher level modules. Although we have not proved the correctness of the reasoning module, it appears in practice to meet the task requirements—autonomous and safe satellite rendezvous and docking.

1.4 Overview

The remainder of this paper is organized as follows: In the next section we present relevant prior work. We then present the CoCo framework in Sect. 3. Section 4 explains the visual servo behaviors for the task of satellite capturing. Section 5 describes the satellite recognition and tracking module. We explain the reactive module in Sect. 6. Sections 7 and 8 describe the deliberative and the plan execution and monitoring modules, respectively. Section 9 describes the physical setup and presents results. Finally, Sect. 10 presents our conclusions.

2 Related work

Early attempts at designing autonomous robotic agents employed a sense-model-plan-act (SMPA) architecture with limited success [31,38,39]. The 1980s saw the emergence of a radically different, ethological approach to robotic agent design, spearheaded by Brooks' subsumption architecture [9] and the mantra "the world is its own best model". Most notable among modern ethological robots is Sony Corporation's robotic dog, AIBO [7], 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 components. Autonomous robot architecture (AuRA) binds a set of reactive behaviors to a simple hierarchical planner that chooses the appropriate behaviors in a given situation [5]. In Servo subsumption symbolic (SSS), a symbolic planner controls a reactive module [12]. In ATLANTIS, the deliberative module advises the reactive behaviors [2,16].

The state of the art in space robotics is the mars exploration rover, Spirit, that visited Mars in 2004 [30]. Spirit is primarily a tele-operated robot that is capable of taking pictures, driving, and operating instruments in response to commands transmitted from the ground, but it lacks any cognitive or reasoning abilities. The most successful autonomous robot to date that has cognitive abilities is "Minerva," which takes visitors on tours through the Smithsonian's National Museum of American History; however, vision is not Minerva's

primary sensory modality [11]. Minerva relies on other sensors, including laser range finders and sonars. Such sensors are undesirable for space operations, which have severe weight/energy limitations.

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. [15], which augments a purely behavioral control substrate [42] 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 such that deliberative, goal-achieving behavior results from the 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.

A survey of work about constructing high-level descriptions from video is found in [20]. Knowledge modeling for the purposes of scene interpretation can either be handcrafted [3] or automatic [14] (i.e., supported by machine learning). The second approach is not immediately feasible in our application since it requires a large training set, which is difficult to gather in our domain, in order to ensure that the system learns all the relevant knowledge, and it is not always clear what the system has learnt. The scene descriptions constructed in [4] are richer than those constructed by our system; however, they do not use scene descriptions to control the visual process and formulate plans to achieve goals.

3 CoCo control framework

CoCo is a three-tiered control framework that consists of deliberative, reactive, and plan execution and monitoring modules (Fig. 2). The deliberative module encodes a knowledge-based domain model and implements a high-level symbolic reasoning system. The reactive module implements a low-level behavior-based controller with supporting perception and memory subsystems. The reactive module is

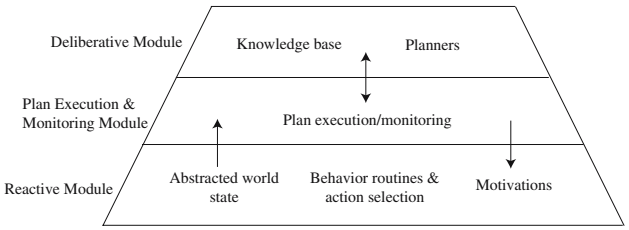


Fig. 2 CoCo three-tiered architecture

responsible for the immediate safety of the agent. As such, it functions competently on its own and runs at the highest priority. At the intermediate level, the plan execution and monitoring module establishes an adviser–client relationship between the deliberative and reactive modules (Fig. 3).

In typical hybrid control frameworks, the reactive module serves as a mechanism to safely execute commands produced through high-level reasoning [12,25] (a notable exception is [6]). A reactive module is capable of much more as is shown by Tu and Terzopoulos [44], Blumberg [8], and Arkin [7], among others. Agre and Chapman [2] observe that most of our daily activities do not require any planning whatsoever; rather, deliberation occurs when a novel, previously unseen situation is encountered. This further highlights the importance of a reactive module in any autonomous robot. CoCo features an ethologically inspired behavior based reactive system fashioned after those developed for autonomous characters in virtual environments [41,44].

In CoCo, the deliberative module advises the reactive module on a particular coarse of action through motivational variables. In contrast to other control architectures where the deliberative module replaces the action selection mechanism built into the reactive module [16], our approach provides a straightforward mechanism for providing high-level advice to reactive behavior without interfering with the action selection mechanism built into the reactive module.

Figure 2 illustrates the AR&D system realized within the CoCo framework. The satellite recognition and tracking routines compute the position and orientation of the satellite and supply the perceptual input to the reactive module, where the servo behaviors that control for the kinematic and dynamic actions of the robotic manipulator provide the relevant motor skills.

4 Motor skills: visual servo behaviors

Satellite rendezvous and docking operations, like all space missions, place stringent requirements on the safety of both the astronauts and the equipment. Therefore, these missions adhere to strict operational guidelines and fully scripted and rehearsed activities. The Mobile Servicing Systems Guide for International Space Station Robotic Systems [21] defines

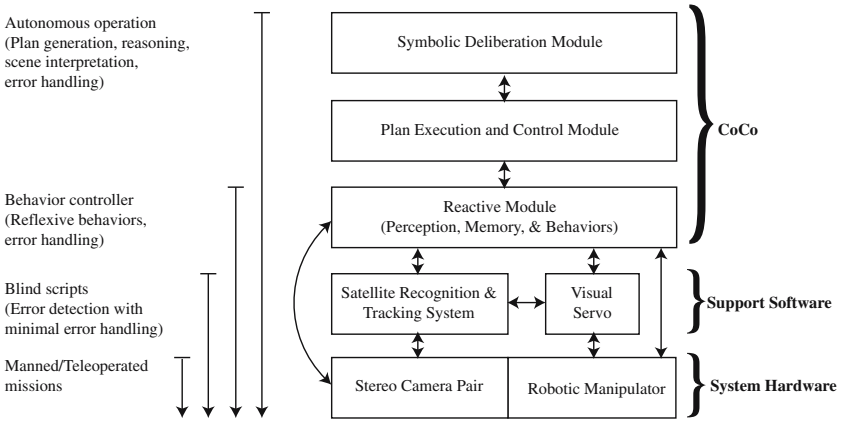


Fig. 3 CoCo system architecture. The satellite rendezvous and docking system comprises an ethologically-inspired behavior module guided by a deliberative module with high-level reasoning abilities. We list the mission capabilities corresponding to different levels of control on the *left*. The degree of autonomy increases as we add more levels of control. At the lowest level, for example, an operator uses the live video feed from the stereo camera pair to tele-operate the robotic manipulator (a.k.a. the chaser robot or the servicer) to dock with the target satellite. At the next level of control, visual servo routines that depend upon the target satellite pose estimation module enable the robotic manipulator to automatically capture the target satellite. Here, the lack of error handling capability renders the operation brittle at best. Besides, the system requires a detailed mission script. The addition of the reactive module results in a more robust autonomous operation, as the reflexive behaviors allow the system to respond to the various contingencies that might arise in its workspace. Still, however, the system can not formulate plans through deliberation to achieve its goals. Consequently, the system requires a mission script. The top-most level of control boasts the highest degree of autonomy. Here, the symbolic deliberation module enables the system to generate the mission script on the fly through reasoning

approach trajectories and envelops as well as mission stages for robotic manipulators during contact operations. During a manned satellite capture operation, an astronaut controls the robotic arm and moves the end-effector through a sequence of way points, which are defined relative to the target satellite. These way points are defined so as to reduce the time that the end-effector spends in close proximity to the target satellite.

AR&D operations will most likely follow the operational guidelines developed for manned operations, especially the concepts of way points, incremental alignment, and stay-out zones. Jasiobedzki and Liu [27] divide a satellite capture operation into six phases (Fig. 4): (1) visual search, (2) monitor, (3) approach, (4) stationkeep, (5) align, and (6) capture, which comply with the robotic manipulator approach guidelines prescribed in [21]. During the *visual search* phase, the cameras are pointed in the direction of the target satellite, and images from the cameras are processed to compute an initial estimate of the position and orientation of the satellite. The *monitor* phase fixates the cameras on the detected satellite while maintaining distance between the satellite and the end-effector. The *approach* phase reduces the distance between the end-effector and the target satellite while keeping the cameras focused on the target. During *stationkeeping*, the distance between the end-effector and the target is preserved and the cameras are kept locked onto the target. The *align* phase controls all six degrees of freedom, aligning the end-effector with the docking interface of the target satellite. Finally, in the *capture* phase, the end-effector moves in to dock with the satellite.

Jasiobedzki and Liu [27] also developed visual servo behaviors corresponding to the six phases identified above. Pose-based servo algorithms that minimize the error between the desired pose and the current pose of the end-effector implement the visual servo behaviors. Poses are defined with respect to the end-effector or the target satellite. During teleoperated missions, the desired poses are set by the operator; whereas, during autonomous operation the desired poses are selected by the active servo behavior, such as monitor, approach, etc. Likewise, the higher-level controller can also set the desired poses, especially when the vision system is failing, to move the end-effector along a particular trajectory. The vision system, which estimates the transformation between the current pose and the desired pose, provides the error signal for the pose based servo routines. The visual servo behaviors provide the motor skills that are essential for successful satellite rendezvous and docking missions.

5 Visual sensors: satellite recognition and tracking

The satellite recognition and tracking module (Fig. 5) processes images from a calibrated passive video camera-pair mounted on the end-effector of the robotic manipulator and estimates the relative position and orientation of the target

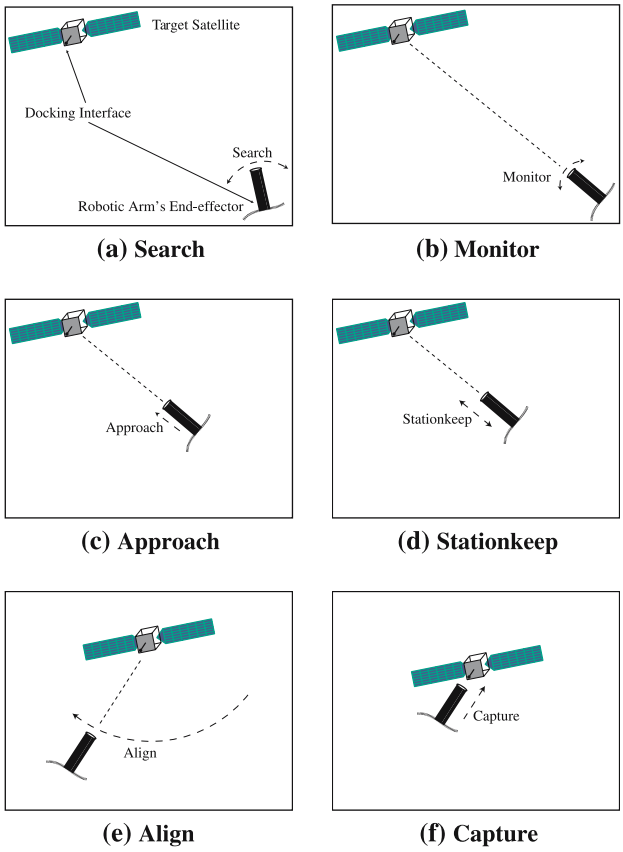


Fig. 4 Six phases during a satellite rendezvous and docking operation [27]

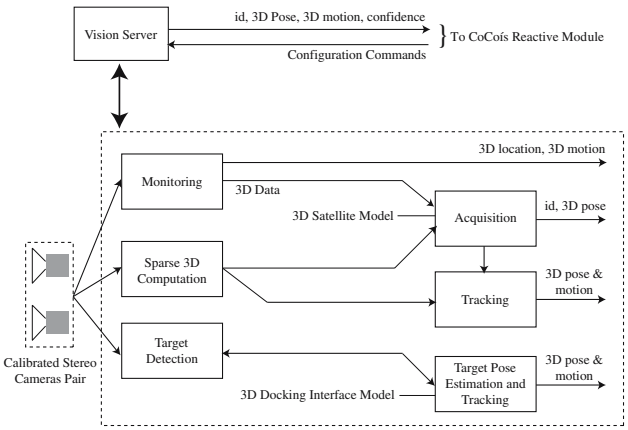


Fig. 5 Satellite recognition and tracking system

satellite [24]. It supports medium and short range satellite proximity operations; i.e., approximately from 6 to 0.2 m. The minimum distance corresponds to the separation between the camera and the satellite in contact position.

The vision algorithms implemented rely mostly on the presence of natural image features and satellite models. During the medium range operation, the vision system cameras view either the complete satellite or a significant portion of it (left image in Fig. 6), and the system relies on natural features observed in stereo images to estimate the motion and pose of the satellite. The medium range operation consists of the following three configurations:

- *Model-free motion estimation* In the first phase, the vision system combines stereo and structure-from-motion to indirectly estimate the satellite motion in the camera reference frame by solving for the camera motion, which is the opposite of the satellite motion [37].
- *Motion-based pose acquisition* The second phase performs binary template matching to estimate the pose of the satellite without using prior information [18]. It matches a model of the observed satellite with the 3D data produced by the last phase and computes a six degree of freedom (DOF) rigid transformation that represents the relative pose of the satellite. The six DOFs are solved in two steps. The first step, which is motivated by the observation that most satellites have an elongated structure, determines the major axis of the satellite. The second step solves for the remaining four DOFs—the rotation around the major axis and the three translations—through exhaustive 3D template matching over the four DOFs.
- *Model-based pose tracking* The last phase tracks the satellite with high precision and update rate by iteratively matching the 3D data with the model using a version of the iterative closest point algorithm [23]. This scheme does not match high-level features in the scene with the model at every iteration. This reduces its sensitivity to partial shadows, occlusion, and local loss of data caused by reflections and image saturation. Under normal operative conditions, model based tracking returns an estimate of the satellite's pose at 2 Hz with an accuracy on the order of a few centimeters and a few degrees.

The short range operation consists of one configuration, namely *visual target based pose acquisition and tracking*. At close range, the target satellite is only partially visible and it cannot be viewed simultaneously from both cameras (the center and right images in Fig. 6); hence, the vision system processes monocular images. The constraints on the approach trajectory ensure that the docking interface on the target satellite is visible from close range. Markers on the docking interface are used to determine the pose and attitude of the satellite efficiently and reliably at close range [24]. Here, visual features are detected by processing an image window centered around their predicted locations. These features are then matched against a model to estimate the pose of the satellite. The pose estimation algorithm requires at least four points to compute the pose. When more than four points are visible, sampling techniques choose the group of points that gives the best pose information. For the short range vision module, the accuracy is on the order of a fraction of a degree and 1 mm right before docking.

Fig. 6 Images from a sequence recorded during an experiment (*left image* at 5 m; *right* at 0.2 m)

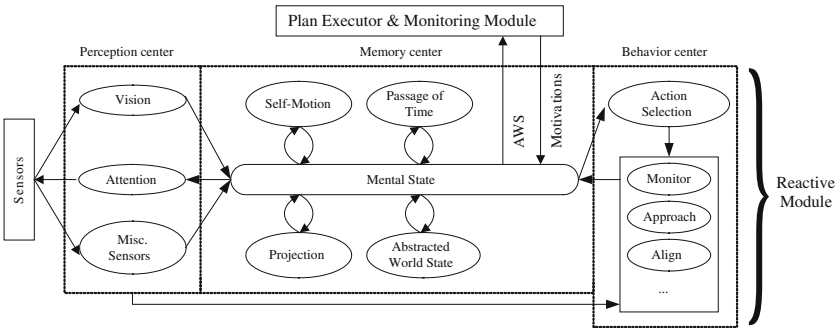


Fig. 7 Functional decomposition of the reactive module, which is realized as a set of asynchronous processes

The vision system returns a 4×4 matrix that specifies the relative pose of the satellite, a value between 0 and 1 quantifying the confidence in that estimate, and various flags that describe the state of the vision system.

The vision system can be configured on the fly depending upon the requirements of a specific mission. It provides commands to activate/initialize/deactivate a particular configuration. At present this module can run in four different configurations, which may run in parallel. Each configuration is suitable for a particular phase of the satellite servicing operation and employs a particular set of algorithms. Active configurations share the sensing and computing resources, which reduces the mass and power requirements of the vision system, but can adversely affect its overall performance.

6 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 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 three functional units: perception, memory, and behavior (Fig. 7). This functional decomposition is intuitive and facilitates the design process. The reactive module is implemented as a collection of asynchronous processes (Table 1), which accounts for its real-time operation.

6.1 Perception center

From an implementational point of view, one can imagine two extremes: one in which a single process is responsible for computing every feature of interest, and the other in which every feature is assigned its own sensing process. In the first scenario, the overall speed of sensing is determined by the feature that takes the longest time to compute, whereas a higher process management overhead is associated with the second scenario to ensure that the sensed values are coherent. For a particular application, it is up to the designer to decide how best to implement the perception system. We chose the second approach where various routines process different perceptual inputs asynchronously in order to compute higher order features, which are then immediately available for subsequent processing. Each data item is assigned a timestamp and a confidence value between 0 and 1, and it is managed by the memory center, which is responsible for preventing other processes from using outdated or incorrect information.[2]

The perception center manages the vision system which was described in Sect. 5. It 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. In addition, the

[2] Working memory is sometimes referred to as the *short term memory* or STM.

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

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) Perception center (sensor fusion)

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 control status. The perception center processes the raw perceptual readings that appear at its inputs, constructs appropriate perceptual features, and stores them in the working memory (memory center) for later use by the behavior center during action selection and behavior execution. A perceptual reading is either from an actual physical sensor (e.g., the docking interface sensor) or the result of a multi-stage operation (e.g., the target satellite's position and orientation). Each perceptual reading is processed independently. Consequently, different perceptual features become available to the reactive module as soon as they are computed.

The perception center includes daemon processes for every perceptual input (Figs. 8, 9). The daemon processes, which awaken whenever new information arrives at their input ports, assign a confidence value to the readings, timestamp them, and push them up the perception pipeline for subsequent processing. The confidence value for a reading is in the range [0, 1], where 0 reflects a total lack of confidence and 1 reflects absolute certainty. It is computed either by the associated daemon or by the process responsible for producing the perceptual reading in the first place. For instance, the vision routines determine the confidence for the estimated position and orientation of the satellite and the daemon responsible for the docking interface sensor assigns a value of 1 to each new reading it receives from the sensor.

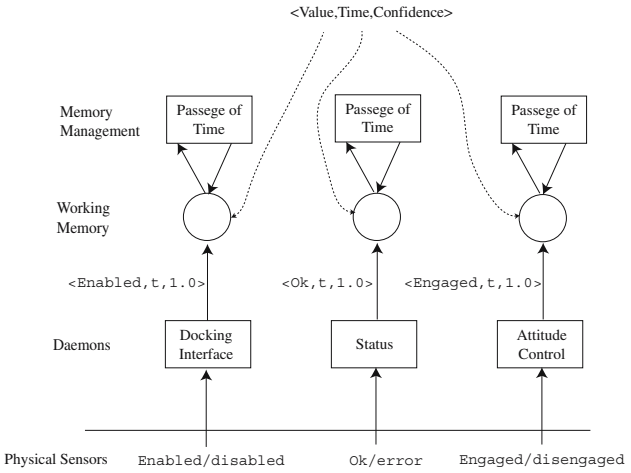


Fig. 8 Daemon processes for monitoring and reading satellite attitude control, docking interface, and robotic arm status sensors. The daemons that collect information from the sensors are associated with the perception center (*bottom row*), whereas those that operate upon the working memory belong to the memory center (*top two rows*)

6.1.1 Communicating with the vision module

Figure 9 shows the interface to the vision sub-system. Long range vision operates anywhere between 20 and 5 m, and the maximum separation between the mock-up satellite and robotic arm is roughly 6 m. To estimate the position and orientation of the satellite, the perception center uses contextual information, such as the current task, the predicted distance from the target satellite, the operational ranges of the various configurations, and the confidence values returned by the active configurations. The perception center is responsible for the transitions between the different vision configurations, and it also performs a sanity check on the operation of the vision sub-system. A decision about whether or not to accept a new pose reading from an active vision module is made by thresholding the confidence value of the reading. The minimum acceptable confidence value for a medium range estimate is 0.3 and it is 0.6 for a short range estimate. These threshold values reflect the expected performance characteristics of the vision system and are selected to impose more stringent performance requirements on the vision system when the robotic arm is in close proximity to the target satellite.

An $\alpha\beta$ tracker validates and smoothes the pose readings from the vision configurations (See [34] for details). The validation is done by comparing the new pose against the predicted pose using an adaptive gating mechanism. When new readings from the vision system consistently fail the validation step, either the vision system is failing or the satellite is behaving erratically and corrective steps are needed. The $\alpha\beta$ tracker thus corroborates the estimates of the visual routines. In addition, it provides a straightforward mechanism for compensating for visual processing delays by predicting the current position and orientation of the target satellite.

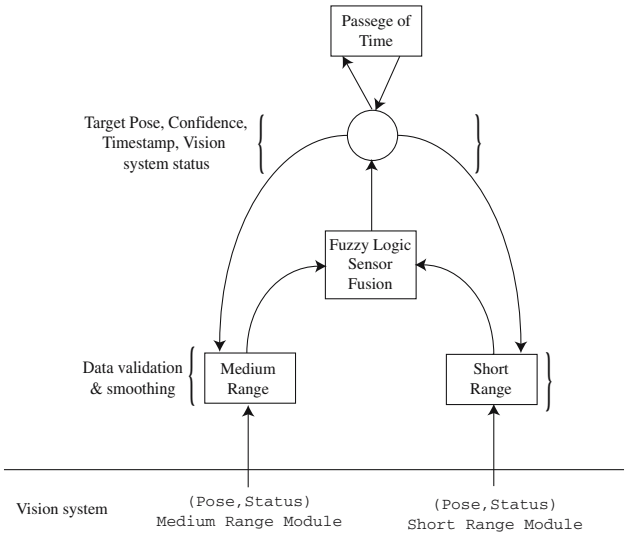


Fig. 9 The perception center is in charge of the vision system that implements satellite identification, recognition, and tracking routines. The deamon processes associated with the visual processing awakens when new vision readings become available and copy the new readings into the working memory. The vision readings are validated and smoothed using an $\alpha\beta$ tracker. A fuzzy logic based sensor fusion scheme combines the readings when multiple vision configurations are active. A passage-of-time behavior associated with the satellite pose information implements a forgetting mechanism, which prevents the reactive system from using out-dated information

6.1.2 Visual processing handover

In the final stages of a successful satellite capture operation, the distance between the robotic arm and the target satellite can vary anywhere from around 6–0 m. The perception center is responsible for transitioning the visual tracking task from the medium to the short range module as the robotic arm approaches the target satellite and vice versa as it pulls back. The perception center uses the estimated distance of the target satellite and the confidence values returned by the active vision configurations to decide which vision module to activate/deactivate.

The strategy for controlling the transition between medium and short range vision modules is based on the following intuitions:

- Since the vision modules are designed to perform reliably only in their operational ranges, a vision module whose estimate falls outside of its operational range should not be trusted.
- When the estimates returned by the active vision module nears its operational limits, activate the more reliable vision module. The operational range of a vision module and the estimated distance of the target satellite determines the suitability of the vision module. For example, when the medium range vision module is active and the target distance estimate is less than 2 m, the short range vision module is activated. The short range vision module uses the current pose of the satellite as estimated by the medium range module to initialize satellite tracking.
- A vision module that is currently tracking the target satellite should not be deactivated unless another vision module has successfully initiated target tracking.
- Avoid unnecessary hand-overs.

We describe the hand-over strategy between different vision modules in [34]. Figure 10 shows the operational status of the vision module during a typical satellite capture mission. Initially, the medium range vision module is tracking the target; however, as the robotic arm approaches the satellite and the distance to the satellite decreases below 2 m, the short range module is activated. Once the short range vision module successfully locks onto the satellite and commences visual tracking, the medium range vision module is deactivated to conserve energy.

6.1.3 Target pose estimation using multiple visual processing streams

To improve the quality of target pose estimates and to ensure smooth transition between different vision modules, we have implemented a fuzzy logic based sensor fusion scheme that combines pose estimates from active vision modules [34]. The sensor fusion scheme takes into account target pose estimates along with their associated confidences and the operational ranges of the vision modules to compute a weighted sum of the pose estimates from the active modules. Currently, it works only with short and medium range vision modules.[3]

The position $\mathbf{p}$ of the satellite is given by

$$\mathbf{p} = w\mathbf{p}_s + (1 - w)\mathbf{p}_m, \tag{1}$$

where $0 \leq w \leq 1$ is the weight assigned to the short-range module's estimate and which is determined by the fuzzy logic based controller, and $\mathbf{p}_s$ and $\mathbf{p}_m$ are the position estimates for the short and medium range modules, respectively. Similarly, we combine the orientation estimates from the short and medium range vision modules by expressing the orientation as quaternions and interpolating between them using w:

$$\mathbf{q} = (\mathbf{q}_s \mathbf{q}_m^{-1})^w \mathbf{q}_m, \tag{2}$$

where $\mathbf{q}_s$ and $\mathbf{q}_m$ are the rotation estimates from the short and medium range vision modules, respectively [34]. When w is 0 the computed pose of the target is the medium range estimate, whereas when w is 1, it is the estimate returned

[3] The long range vision module is used only initially to locate and identify the target. Once the target is identified, the medium range module takes over. At present, the long and medium range vision modules do not operate concurrently.

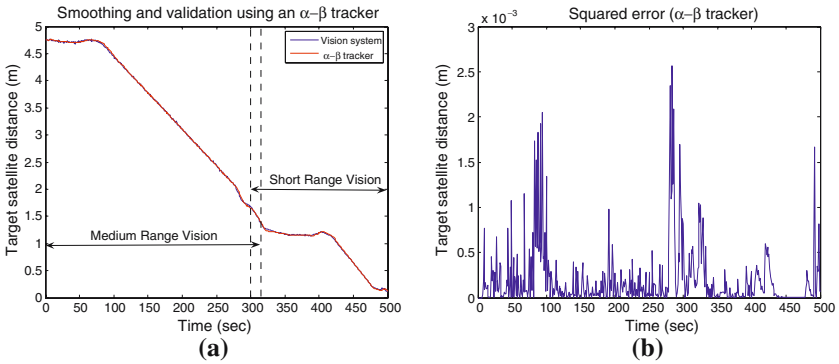


Fig. 10 **a** Medium range vision hands over target tracking to the short range vision module as the chaser moves closer to the target. **b** The prediction error of the $\alpha\beta$ tracker. The fuzzy logic based sensor fusion scheme fuses the information from active vision modules to form a single coherent target pose estimate

by the short range module. The details of the sensor fusion module are provided in [34].

6.2 Behavior center

The behavior center manages the reactive module's behavioral repertoire. This by no means a trivial task involves arbitration among behaviors. The reactive module supports multiple concurrent processes, and arbitrates between them so that the emergent behavior is the desired one. We have, however, benefited from dividing the reactive module into three components (perception, behavior, and memory), minimizing behavior-interaction across the components, thus simplifying the management of 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. The chosen action then activates lower level supporting behaviors, as necessary. The current state of the world takes precedence over the motivations, i.e., the reactive module will follow the advice from the deliberative module only when the conditions are favorable. 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.

6.2.1 Motivational variables

The behavior controller maintains a set of internal mental state variables, which encode the motivations of the robotic arm: (1) search, (2) monitor, (3) approach, (4) align, (5) contact, (6) depart, (7) park, (8) switch, (9) latch, (10) sensor, and (11) attitude control. The mental state variables take on values between 0 and 1, and at each instant the action selection mechanism selects the behavior associated with the motivational variable having the highest value. Priority among the different motivations resolves behavior selection conflicts when multiple motivations have the same magnitudes. Once the goal associated with a motivational variable is fulfilled, the motivational variable begins to decrease asymptotically to zero.[4] A similar approach to action selection is used by Tu [44] in her artificial fish and by Shao [41] for his autonomous pedestrians.

We model a *level-of-interest* to prevent one behavior from excluding other behaviors while it infinitely pursues an unattainable goal [8]. A maximum cutoff time is specified for each motivational variable and if, for whatever reason, the associated goal is not fulfilled within the prescribed cutoff time, the value of the motivational variable starts to decay to 0 (Fig. 11). We also employ a Minsky/Ludlow [29] model of mutual inhibition to avoid behavior dither; a situation where the action selection keeps alternating between two goals without ever satisfying either of them. Mutual inhibition is implemented by specifying a minimum duration for which a behavior must remain active and by initially increasing the value of the associated motivation variable (Fig. 12).

The values of the motivational variables are calculated as follows:

$$m_t = \begin{cases} \max\left(0, m_{t-1} - d_a \Delta t \left(1 - d_b m_{t-1}^2\right)\right) \\ \quad \text{when } t > t_c \text{ or the associated behavior} \\ \quad \text{achieves its goal,} \\ \min\left(1, m_{t-1} + g_a \Delta t \left(e^{m_{t-1}^2} - g_b\right)\right) \\ \quad \text{when the associated behavior is first initiated,} \end{cases}$$

where $0 \leq g_a, g_b, d_a, d_b \leq 1$ are the coefficients that control the rate of change in the motivational variables, which are set empirically to 0.5, 0.99, 0.05, and 0.99, respectively.[5] Δt is the time step. The values of a motivational variable at time t and $t - \Delta t$ are m_t and m_{t-1}, respectively. The associated cutoff time is t_c. The cutoff time for a particular motivation

[4] This is consistent with the "drive reduction theory" proposed by Hull [22], whose central theme is that drive (motivation) is essential in order for a response to occur; furthermore, a response is chosen so as to reduce (or satisfy) the most pressing drive.

[5] In a more general setting the values of the coefficients can be chosen on a per-motivation basis.

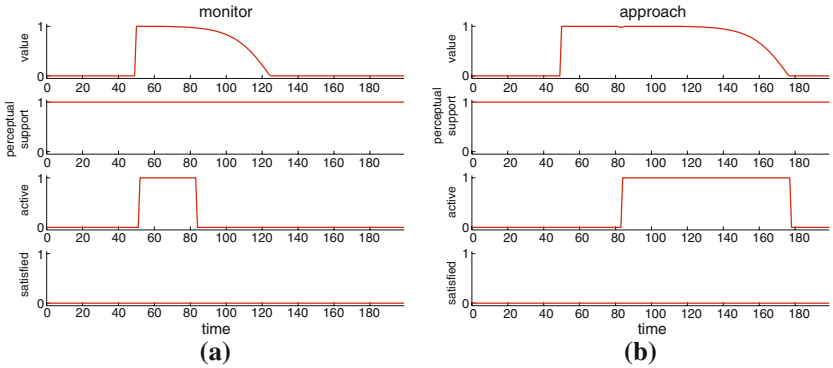


Fig. 11 Priority among motivations and level-of-interest modeling. The deliberative module sets the value of motivational variables *monitor* **a** and *capture* **b** to 1. The action selection mechanism selects the *monitor* behavior, which has a higher priority than the *approach* behavior. The *monitor* behavior fails to achieve its objectives within the prescribed time, the motivational variable *monitor* begins to decay to 0. When the value of the *monitor* variable is less than that of the *approach* variable, the *approach* behavior is activated. In this particular scenario, the *approach* behavior did not meet its objectives within the prescribed time and the *approach* variable decreases to zero. In either case, the decay in the motivational variables is due to the level-of-interest modeling

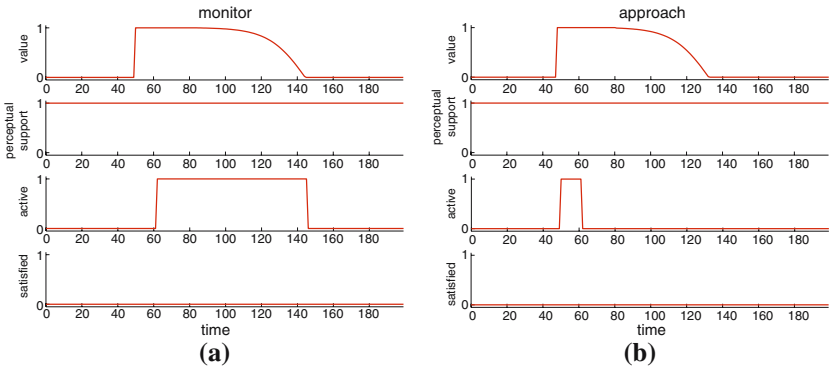


Fig. 12 Mutual inhibition. The *monitor* behavior has a higher priority than the *approach* behavior; however, when the *approach* behavior is active, it inhibits the *monitor* behavior for some prescribed time and prevents the *monitor* behavior from becoming active. After the inhibition period, the *monitor* behavior becomes active, deactivating the *approach* behavior

depends upon two factors: the motivation in question and whether or not other motivational variables are greater than 0:

$$t_c = \begin{cases} t_1 & \text{when other motivational variables, } > 0 \\ t_2 & \text{otherwise,} \end{cases}$$

where $0 > t_1 > t_2 > \infty$.

The higher-level deliberative module suggests an action to the reactive module by setting the relevant motivational variable(s) to 1 or 0. Any parameters associated with the suggested action are passed directly to the behavior linked to the motivational variable. It is up to the reactive module to decide whether or when to execute the suggested action by activating the associated behavior. Furthermore, the reactive module is not responsible for communicating its decision or status to the deliberative module. The plan execution and monitoring module determines whether or not the suggested action was ever executed or that it failed or succeeded through the abstracted world state (Fig. 13).

A consequence of the design proposed here is that the behavior-based reactive module is oblivious to the existence of the deliberative and plan execution and monitoring modules. The sole agenda of the reactive module is to minimize the internal motivational variables by activating appropriate behaviors. The system operates at a diminished capacity when higher level modules are disabled. Built into the reactive module is a provision for overriding the action selection mechanism during a teleoperated mission; i.e., when the system is being controlled by an astronaut.

6.3 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. At each instant, it copies whatever new sensory information is available at the perception center, and it provides a convenient way of handling perception delays. At any moment, the memory center has a time-delayed version of the sensory information, and it projects this information to the current instant. Thus, the behavior center need not wait for new sensory information; it can simply use the information stored in the memory center, which is responsible for ensuring that this information is valid.

The memory center uses two behavior routines (per feature), *self-motion* and *passage-of-time*, to ensure the currency

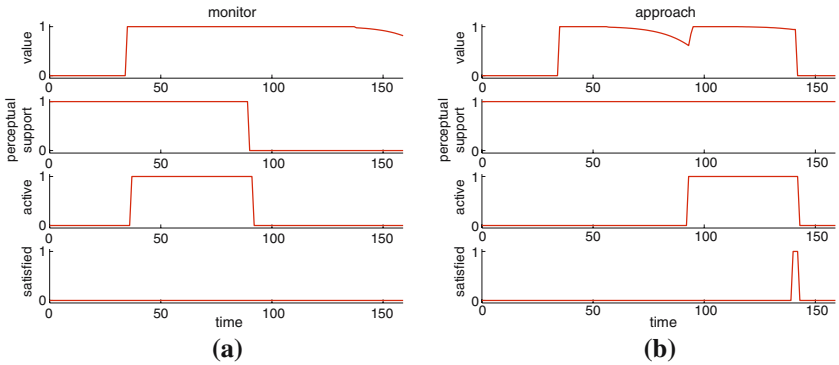


Fig. 13 **a** Perceptual support: *monitor* behavior is deactivated when perceptual support for the *monitor* behavior vanishes. **b** The *approach* behavior initially increases the *approach* variable to encourage persistence, and the *approach* variable decreases as the *approach* behavior is doing its job

and coherence of the information. The robot sees its environment egocentrically. External objects change their position with respect to the agent as it moves. The *self-motion* behavior routine constantly updates the internal world representation to reflect the current position, heading, and speed of the robot.

Each perceptual feature is represented as a tuple ⟨*Value*, *Timestamp*, *Confidence*⟩ in the working memory. *Value* represents the present value of the feature, *Timestamp* stores the time at which the feature was generated, and *Confidence* ∈ [0, 1] is the current confidence value of the feature. In the absence of new readings from the perception center, the confidence in the world state should decrease with time (Fig. 14). 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.[6]

The ability to forget outdated sensory information is critical to the overall operation of the reactive module, providing a straightforward mechanism to prevent it or the deliberative module from operating upon inaccurate, or worse, incorrect information, and can be used to detect sensor failures. The confidence value for a perceptual feature tends to zero in the absence of fresh information from the relevant sensor. The lack of new information from a sensor can be construed as a malfunctioning sensor, particularly for sensors, such as the docking interface status sensor, that periodically send new information to the perception center.

6.3.1 Abstracted world state (AWS)

The reactive module requires detailed sensory information, whereas the deliberative module employs abstract information about the world. The memory center filters out unnecessary details from the sensory information and generates the abstracted world state which expresses the world symbolically (Fig. 15). The abstracted world state is a discrete,

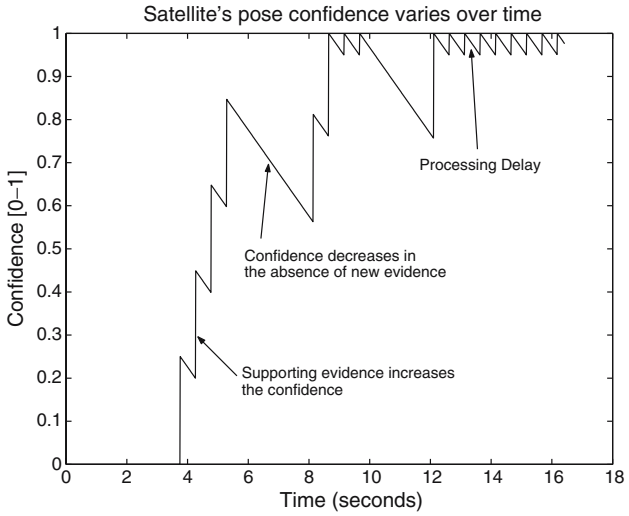


Fig. 14 The confidence in the satellite's pose decreases in the absence of supporting evidence from the vision system

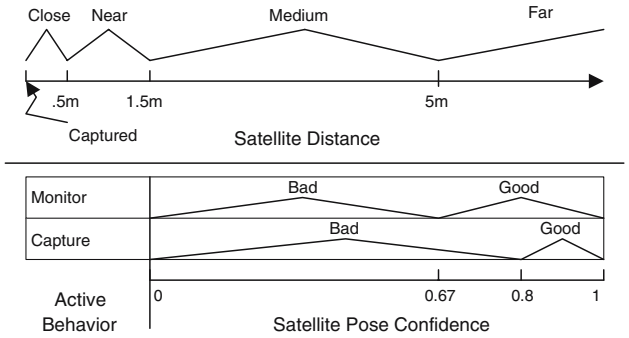


Fig. 15 The abstracted world state represents the world symbolically. For example, the satellite is either *Captured*, *Close*, *Near*, *Medium*, or *Far*. In the memory center, the conversion from numerical quantities to symbols takes into account the current state of the agent

multi-valued representation of an underlying continuous reality.

Discretization involves dividing a continuous variable into ranges of values and assigning the same discrete value to all values of the continuous variable that fall within a certain range. Discretization, however, is not without its problems. When the value of the continuous variable hovers about a

[6] Decreasing confidence values over time is motivated by the decay theory for short term (working) memory proposed by Peterson and Peterson in 1959 [32].

discretization boundary, the discretized value can switch back and forth between adjacent discrete values, which can pose a challenge for a process that relies on the stability of a discrete variable. We address this problem by imitating hysteresis during the discretization operation. We illustrate our strategy in Fig. 16a, where variable y resists a change from α to β and vice-versa; thereby, avoiding alternating between the two values when the value of x fluctuates about 0.5. A related approach is taken when converting binary (or discrete multi-valued) sensory information to binary (or multi-valued) fluents. Consider, for example, mapping a binary variable $x \in [0, 1]$ to $y \in [\alpha, \beta]$,

$$y = \begin{cases} \alpha \text{ if } x = 0 \\ \beta \text{ if } x = 1 \end{cases}.$$

The value of y does not faithfully follow the value of x. Rather, the value of y is only switched from α to β when the value of x is consistently 1. Similarly, the value of y is switched from β to α when the value of x stays at 0.

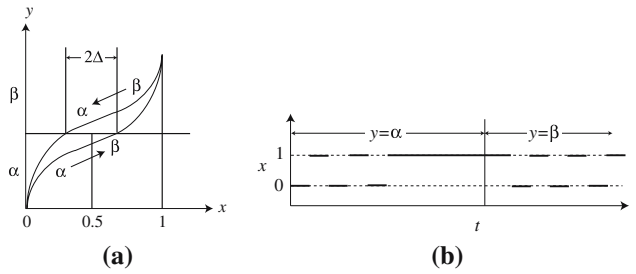


Fig. 16 Emulating hysteresis during discretization. **a** y is a discretization of x that takes values between 0 and 1. If y is α and the value of $x > 0.5 + \Delta$, y becomes β. Otherwise, if y is β and the value of $x < 0.5 - \Delta$, then y becomes α. **b** $x \in [0, 1]$ is mapped to $y \in [\alpha, \beta]$. The state $y = \alpha$ indicates that $x = 0$ and $y = \beta$ indicates that $x = 1$. The variable y resists changing its value from α to β and vice-versa, which allows y to exhibit more stable behavior by ignoring spurious changes in x

Using the above scheme, we convert continuous sensory information, such as the estimated distance from the satellite and the estimated speed of the satellite, as well as binary values, such as the status of the latch, to appropriate fluents that comprise the abstracted world state. The list of fluents is provided in Table 2.

7 The deliberative module

The deliberative module endows our agent with the ability to plan its actions, so that it can accomplish high-level tasks that are too difficult to carry out without "thinking ahead." To this end, the deliberative module maintains a set of planners, each with its own knowledge base and planning strategy. Generally, the world behaves much more predictably at higher levels of abstraction. Hence, each planner understands the world at an abstract level, which makes reasoning tractable, as opposed to ill-conceived attempts to formulate plans in the presence of myriad low-level details. The lowest level of abstraction for a particular planner is determined by the reactive module explicitly through the abstracted world state and implicitly through the behaviors that it implements. The latter constitute 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 (Table 2).

Symbolic logic provides the right level of abstraction for developing high level planners that elegantly express abstract ideas. We advocate using a high-level agent language, such as GOLOG [26], to develop planners for the deliberative

Table 2 The abstracted world state for the satellite servicing task

Fluents/arity	Values	Description
fStatus/1	On/off	Status of the servicer
fSatPosConf/1	Yes/no	Confidence in the estimated pose of the satellite
fSatPos/1	Near/medium/far/contact	Distance from the satellite
fSatSpeed/1	Yes/no	Whether the satellite's relative speed is within the acceptable limits
fLatch/1	Unarmed/armed	Status of the latch (docking interface)
fSatCenter/1	Yes/no	Whether the satellite is in the center of the field of view
fSatAlign/1	Yes/no	Whether servicer is aligned with docking interface of the satellite
fSensor/2	Short/medium, on/off	Current configuration of the vision system
fError/1	Sensor/shadow/any/no	Error status
fSatContact/1	False/true	Whether satellite is already docked
fSatAttCtrl/1	On/off	Whether or not the satellite's attitude control is active
fSun/1	Front/behind	Location of the Sun relative to the servicer
fRange/1	Near/far	Distance from the satellite

The choice of fluents describing the abstracted world state depends upon the target application

Table 3 Primitive actions available to the planner that creates plans to accomplish the goal of safely capturing the target satellite

Actions/#args	Arguments' values	Description
aTurnon/1	On/off	Turns on the servicer
aLatch/1	Arm/disarm	Enables/disables the latching mechanism
aErrorHandle/1		Informs the operator of an error condition
aSensor/2	Medium/near, On/off	Configures the vision system
aSearch/1	Medium/near	Initiates medium/short visual search sequence
aMonitor/0		Initiates monitor phase
aAlign/0		Initiates align phase
aContact/0		Moves in to make contact
aGo/3	Park/medium/near, Park/medium/near, Vis/mem	Moves to a particular location using either current information from the vision system (if vision system is working satisfactorily) or relying upon the mental state
aSatAttCtrl/1	Off/on	Asks ground station to turn off the satellite attitude control
aCorrectSatSpeed/0		Informs the operator that the satellite is behaving erratically

module. Consequently, the deliberative module comprises high-level, non-deterministic GOLOG programs whose execution produce the plans for accomplishing the task at hand. GOLOG is a logic programming language for dynamic domains with built-in primitives (fluents) to maintain an explicit representation of the modeled world, on the basis of user supplied domain knowledge. The domain knowledge consists of 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. A GOLOG program can reason about the state of the world and consider various possible courses of action before committing to a particular choice, in effect performing deliberation. The GOLOG language has been shown to be well suited to applications in high-level control of robotic systems, industrial processes, software agents, etc. An advantage of GOLOG over traditional programming languages like C is that programs can be written at a much higher level of abstraction. GOLOG is based on a formal theory of action specified in an extended version of the situation calculus [35], so GOLOG programs can be verified using theorem proving techniques. A prototype GOLOG interpreter for SWI-Prolog [1] is presented in [34]. We treat GOLOG programs as planners; hence, in the remainder of this paper we will use the term planner and GOLOG program interchangeably.

The symbolic reasoning module comprises two *specialist* planners. Planner A is responsible for generating plans to achieve the goal of capturing the target satellite. Planner B attempts to explain the changes in abstracted world state. It effectively produces high level explanations of what might have happened in the scene (workspace). The primitive actions available to the two planners are listed in Tables 3 and 4, respectively. The planners experience the world through the fluents (AWS) listed in Table 2.

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 motivational variables to use, and specifies its preconditions and postconditions.

7.1 Scene interpretation

The cognitive vision system monitors the progress of the current task by examining the AWS, which is maintained in real-time by the perception and memory module. Upon encountering an undesirable situation, the reasoning module tries to explain it by constructing an interpretation. If the reasoning module successfully finds a suitable interpretation, it suggests appropriate corrective steps; otherwise, it suggests

Table 4 Primitive actions available to the planner that constructs abstract, high-level interpretations of the scene by explaining how the AWS is evolving

Actions/#args	Arguments' values	Description
aBadCamera/0		Camera failure
aSelfShadow/0		Self-shadowing phenomenon
aGlare/0		Solar glare phenomenon
Sun/1	Front/behind	The relative position of the Sun
aRange/1	Near/medium	Distance from the satellite

the default procedure for handling anomalous situations. The default error handling procedure for our application, like all space missions, is to safely abort the mission, i.e., to bring the robotic manipulator to its rest position while avoiding collisions with the target satellite. The procedure for finding explanations is as follows:

1: Construct plans that account for the current error conditions by using the knowledge encoded within the error model.
2: Sort these plans in ascending order according to their length. (We disregard the default plan, which usually has a length of 1.)
3: **for all** Plans **do**
4: Simulate plan execution; this consists of querying the perception and memory unit or asking the operator.
5: **if** The execution is successful **then**
6: The current plan is the most likely explanation.
7: Break
8: **end if**
9: **end for**
10: **if** No explanation is found **then**
11: The default plan is the most likely explanation.
12: **end if**
13: Generate a solution based on the current explanation; this requires another round of reasoning.
14: **if** The solution corrects the problem **then**
15: Continue doing the current task.
16: **else**
17: Abort the current task and request user assistance.
18: **end if**

A fundamental limitation of the proposed scene interpretation strategy is that it requires a detailed error model—i.e., a knowledge base of what might go wrong and how—and for a general scene it might be infeasible to acquire this knowledge. Space missions, however, can benefit from the approach, since they are usually studied in detail for months and sometimes, years by a team of engineers and scientists who run through all the foreseeable scenarios. Indeed, on-orbit missions are carefully planned and highly scripted activities. Furthermore, they generally take place in *uncluttered* environments, so the number of possible events can be managed. Therefore, our framework appears to be useful for vision-based robotic systems for AR&D.

7.2 Cooperation between active planners

The planners cooperate to achieve the goal—safely capturing the satellite. The two planners interact through a plan execution and monitoring unit to avoid undesirable interactions. Upon receiving a new "satellite capture task" from the ground station, the plan execution and monitoring module activates Planner A, which generates a plan that transforms the current state of the world to the goal state—a state where the satellite is secured. Planner B, on the other hand, is only activated when the plan execution and monitoring module detects a problem, such as a sensor failure. Planner B generates all plans that will transform the last known "good" world state to the current "bad" world state. Next, it determines the most likely cause for the current fault by considering each plan in turn. After identifying the cause, Planner B suggests corrections. In the current prototype. Possible corrections consist of "abort mission," "retry immediately," and "retry after a random interval of time" (the task is aborted if the total time exceeds the maximum allowed time). Finally, after the successful handling of the situation, Planner A resumes (Tables 5, 6).

8 Plan execution and monitoring module

The Plan Execution and Monitoring (PEM) 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 preconditions and postconditions 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, then unexecuted actions whose preconditions are satisfied are activated (Fig. 17). Together the preconditions and postconditions constitute the plan execution control knowledge.

Table 5 A linear plan generated by the GOLOG program to capture the target

Starting world state:
fStatus(off) ∧ fLatch(unarmed) ∧ fSensor(all,off) ∧ fSatPos(medium) ∧ fSatPosConf(no) ∧ fSatCenter(no) ∧ fAlign(no) ∧ fSatAttCtrl(on) ∧ fSatContact(no) ∧ fSatSpeed(yes) ∧ fError(no)

Execution result:
aTurnon(on) → aSensor(medium,on) → aSearch(medium) → aMonitor → aGo(medium,near,vis) → aSensor(short,on) → aSensor(medium,off) → aAlign → aLatch(arm) → aSatAttCtrl(off) → aContact

The GOLOG program is provided in [34]

Table 6 Planner B uses the error model to determine possible explanations of an error condition

Starting world state: fRange(unknown) ∧ fSun(unknown) ∧ fSatPosConf(yes)
Proposed explanation 1: aBadCamera
Proposed explanation 2: aSun(front) → aGlare
Proposed explanation 3: aRange(near)→aSun(behind)→aSelfShadow

The plan execution and monitoring module executes these plans in sequence to pick the most likely cause of the error. A solution is suggested once the cause of the error is identified

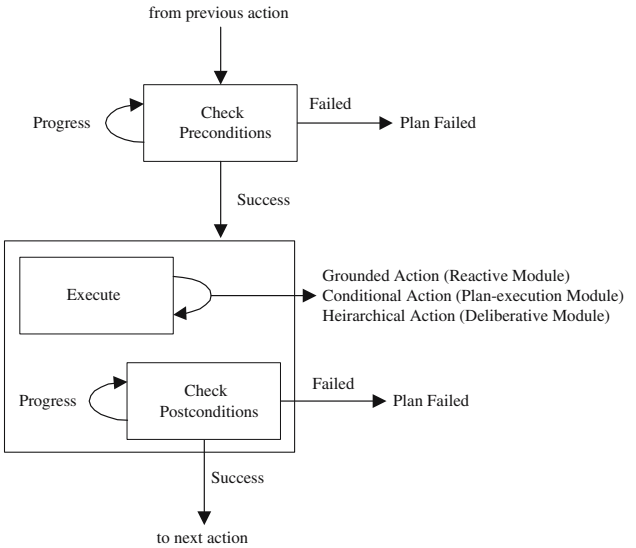


Fig. 17 The plan execution and monitoring module sequentially executes each action. It checks the current action's preconditions until they succeed or fail. If they succeed, it enters the current action's execution/postcondition-check loop, wherein it activates the current action's execution code until the postconditions either succeed or fail. Upon success, it proceeds to the next action

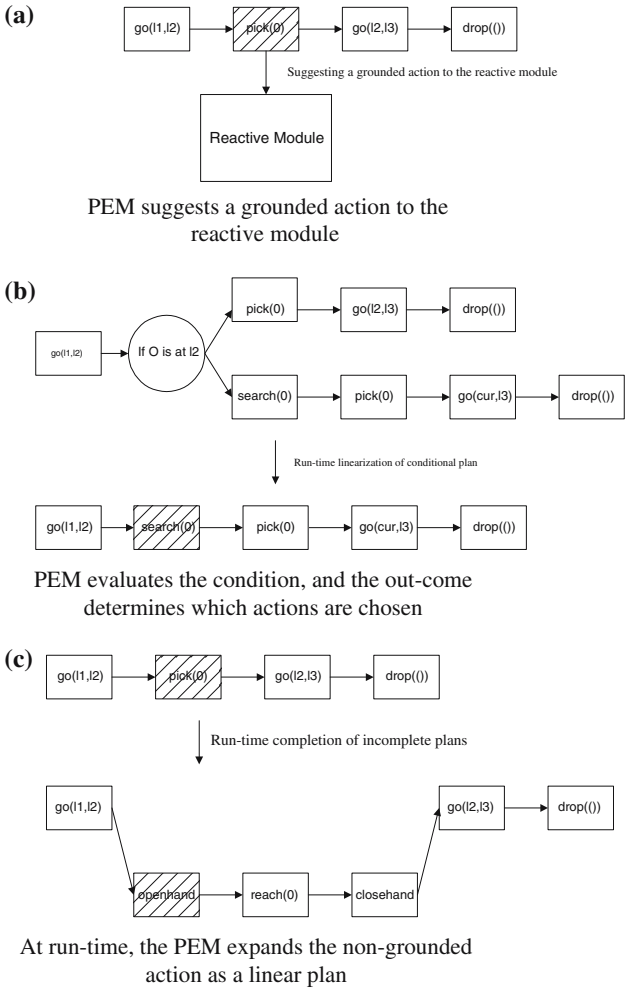


Fig. 18 PEM module executing linear, conditional, and hierarchical plans

The PEM module can handle linear, conditional, and hierarchical plans; thereby, facilitating the *sliding autonomy* capability of the overall controller (Fig. 18). Plans constructed by the deliberative module have a linear structure. Every action of the plan is directly executable on the reactive module, and each action must succeed for the plan to achieve its objectives. Scripts uploaded by human operators usually have a conditional/hierarchical structure. For conditional plans, it is not sufficient to execute each action in turn, rather the outcome of an action determines which of the remaining actions to execute next. On the other hand, in hierarchical plans some actions acts as macros that represent other plans that have to be computed at runtime. The plan execution and monitoring module handles linear, conditional, and hierarchical plans through the following *plan linearization* process (Fig. 18):

1: **if** Current action is "grounded" (i.e., directly executable on the reactive module) **then**
2: Send to the reactive module.
3: **else if** Current action is "conditional" (e.g., a sensing action, etc.) **then**
4: Evaluate condition and pick the next action based on the outcome.
5: **else if** Current action is "non-grounded" (i.e., it requires further elaboration) **then**
6: Perform elaboration and replace the hierarchical action with the outcome (plan stitching).
7: **else**
8: Unknown action type. Plan execution failure.
9: **end if**

The PEM module can execute multiple actions concurrently; 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.

8.1 Plan execution control knowledge

The PEM relies upon execution control knowledge to properly execute a plan. Execution control knowledge is defined over the abstracted world state, and it consists of conditions that must hold before, during, or after an action (or a plan). Some of these conditions span the entire plan while others are action-dependent. Together these conditions answer the following questions that are vital for the correct execution of a plan:

- Plan validity (a plan might become irrelevant due to some occurrence in the world).
- Action execution start time.
  - Now.
  - Later.
  - Never; the plan has failed.
- Action execution stop time.
  - Now; the action has either successfully completed or failed.
  - Later; the action is progressing satisfactorily.

For our application, the plan execution control knowledge is readily available in the form of precondition action axioms and successor state axioms.

9 Results

We have developed and tested the CoCo AR&D system in a simulated virtual environment, as well as in a physical lab environment at MDA Space Missions, Ltd., that faithfully reproduces on-orbit movements and the illumination conditions of the space environment—strong light source, very little ambient light, and harsh shadows. The physical setup consisted of the MD Robotics, Ltd., proprietary "Reuseable Space Vehicle Payload Handling Simulator," comprising two Fanuc robotic manipulators and the associated control software. One robot with the camera stereo pair mounted on its end effector acts as the servicer. The other robot carries a grapple-fixture-equipped satellite mock-up and synthesizes realistic satellite motion.

The capture procedure is initiated by a single high-level command from the ground station. Upon receiving the command, the system initializes the long-range vision module to commence a visual search procedure. Once the satellite is found, and its identity confirmed, the system guides the robotic arm to move closer to the satellite. The performance of the long-range vision module deteriorates as the separation between the robotic arm and the satellite decreases due to the fact that the cameras are mounted on top of the end-effector. In response, the cognitive vision system turns on the medium range vision module and it turns off the long-range vision module to conserve power once the medium range system is fully initialized and reliably tracking the satellite. Next, the robotic is arm tries to match the satellite's linear and angular velocities, a procedure known as *station keeping*. Then, short-range vision processing is initiated, and a message is sent to the ground station to turn off the satellite's attitude control system. The robotic arm should not capture a satellite whose attitude control system is functioning, as that might destroy the satellite, the robotic arm, or both. When the attitude control system is inactive, the satellite begins to drift; however, the robotic arm follows it by relying upon the short-range vision system. Upon receiving a confirmation from the ground station that the satellite's attitude control system is off, the robotic arm moves in to make contact.

We performed 800 test runs in the simulated environment and over 25 test runs on the physical robots. For each run, we randomly created error conditions (see Table 7), such as a vision system failure and/or hardware failures. The cognitive vision controller gracefully handled all of them and met its requirements; i.e., safely capturing the satellite using vision-based sensing (Fig. 6 shows example sensed images) while handling anomalous situations. The controller never jeopardized its own safety nor that of the target satellite. In most cases, it was able to guide the vision system to re-acquire the satellite by identifying the cause and initiating a suitable search pattern. In situations where it could not resolve the error, it safely parked the manipulator and informed the ground station of its failure.

Figure 19 shows a satellite capture sequence in the lab, where the servicer was able to capture the satellite without incident.

During the capture sequence shown in Fig. 20, we simulated a vision system failure. The servicer gracefully handled the error by relying upon its cognitive abilities and successfully captured the satellite. When there is an error, such as a vision system failure, the reactive system responds immediately and tries to increase its separation from the satellite. In the absence any new perceptual information, the system relies upon its time-aware and context-sensitive mental state. Meanwhile, the deliberation module is using its knowledge base to explain the error and suggest a recovery.

Figure 21 shows a simulated satellite rendezvous sequence. Upon receiving a *dock* command from the ground station, the servicer initiates a visual search behavior, which points the cameras towards the incoming satellite (Fig. 21a,b). Once the satellite's identity is confirmed, the servicer begins to approach it (Fig. 21c). Initially, the servicer only has information about the position of the satellite; however, as it approaches the satellite, it activates the medium vision module. The medium range vision processing estimates the orientation of the satellite. The servicer aligns itself with the grapple fixture of the satellite, following an arc around

Table 7 CoCo handled these error conditions that were randomly generated during various test runs

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	

Fig. 19 The servicer robot captures the satellite using vision in harsh lighting conditions like those in orbit

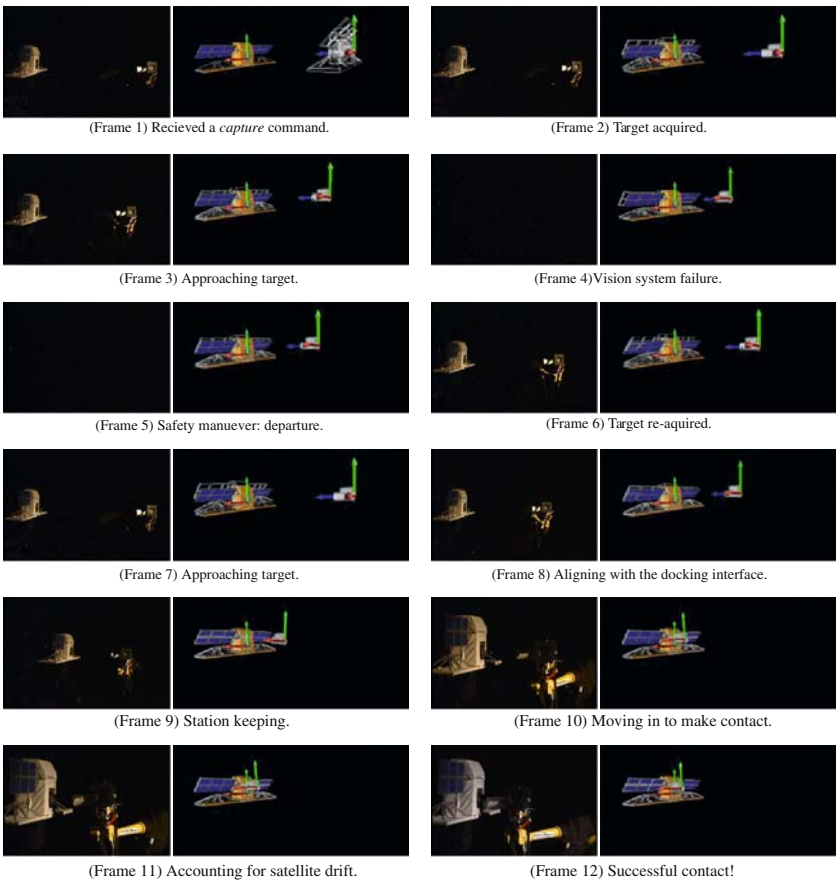


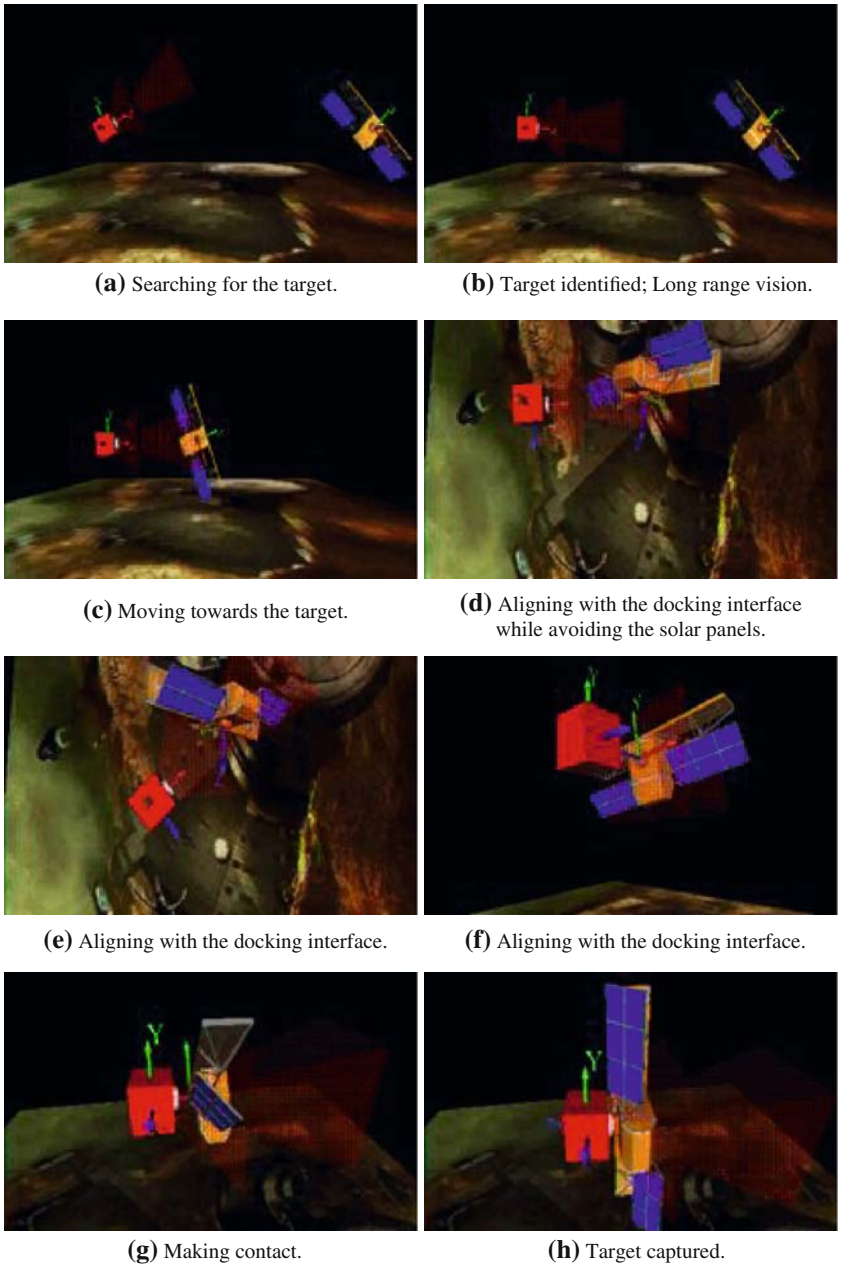
Fig. 20 Despite a simulated vision system failure, the servicer robot captures the satellite using vision by making use of its cognitive abilities. On the *right of each frame*, we show the view from the simulation environment that runs the controller code. The simulation environment communicates with the physical robots over the network. Here, the wireframe represents the position of the satellite as estimated by the robotic arm. The 3D model of the satellite represents the actual position of the satellite according to the sensors on the Fanuc robot arm. The *gray cylinder* represents the position of the chaser robot end-effector according to the telemetery information. Note that the estimated position is maintained in the absence new perceptual information (Frames 3 and 4). A vision failure was induced by shutting off the ambient light

the satellite to avoid the delicate solar panels (Fig. 21d–f). The servicer initiates stationkeeping, where it matches the position and orientation of the satellite (Fig. 21g). At this stage, the servicer's view is limited to the grapple fixture mounted on the satellite, as the cameras are mounted on the docking mechanism of the servicer. Therefore, the servicer activates the short range vision module. Finally, it moves towards the satellite to make contact and capture it (Fig. 21h).

10 Conclusion

Like the earliest machine vision systems [31,36], future applications of vision will require more than just image analysis. They will also need a high-level AI component to guide the vision system in a deliberate, task-appropriate manner, to diagnose sensing problems, and to take corrective actions. The AI system must rely on a low-level reactive component responsible for sensorimotor control.

Fig. 21 Satellite rendezvous simulation



(a) Searching for the target.

(b) Target identified; Long range vision.

(c) Moving towards the target.

(d) Aligning with the docking interface while avoiding the solar panels.

(e) Aligning with the docking interface.

(f) Aligning with the docking interface.

(g) Making contact.

(h) Target captured.

We have demonstrated such a system in the domain of space robotics, specifically in the context of on-orbit satellite autonomous rendezvous and docking. Our practical vision-based robotic system interfaces object recognition and tracking with classical, logic-based AI through a behavior-based perception and memory unit. Using its reactive sensorimotor and deliberative reasoning abilities, it successfully performs the complex task of autonomously capturing a free-orbiting satellite in harsh illumination conditions. In most of our simulation tests as well as in a commercial robotic lab environment that emulates the relevant real-world conditions, when prompted by a single high-level command our system successfully captured the target satellite while dealing with anomalous situations.

Acknowledgments The authors acknowledge the valuable technical contributions of P. Jasiobedzki, R. Gillett, H.K. Ng, S. Greene, J. Richmond, M. Greenspan, M. Liu, and A. Chan. This work was funded by MDA Space Missions, Ltd. (formerly MD Robotics, Ltd.) and Precarn Associates. We thank L. Gregoris for motivation, encouragement, and support.

References

1. Swi-prolog. http://www.swi-prolog.org/
2. Agre, P., Chapman, D.: Pengi: an implementation of a theory of activity. In: Proceedings of American Association of Artificial Intelligence, San Mateo, CA. Morgan Kaufmann (1987)
3. Arens, M., Nagel, H.H.: Behavioral knowledge representation for the understanding and creation of video sequences. In: Gunther, A., Kruse, R., Neumann, B. (eds), Proceedings of the 26th German Conference on Artificial Intelligence (KI-2003), pp. 149–163, Hamburg, Germany (15–18 September 2003)
4. Arens, M., Ottlik, A., Nagel, H.-H.: Natural language texts for a cognitive vision system. In: van Harmelen, F. (ed), Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-2002), pp. 455–459, Amsterdam, The Netherlands. IOS Press (21–26 July 2002)
5. Arkin, R.C.: AuRA: a hybrid reactive/hierarchical robot architecture. In: Proceedings of the IEEE International Conference on Robotics and Automation, Workshop on Architecture for Intelligent Control Systems, 1992
6. Arkin, R.C., Riseman, E.M., Hanson, A.R.: ArRA: An architecture for vision-based robot navigation. In: Proceedings of the DARPA Image Understanding Workshop, Los Angeles, CA (1987)
7. Arkin, R.C., Fujita, M., Takagi, T., Hasegawa, R.: Ethological modeling and architecture for an entertainment robot. In: Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA 2001), pp. 453–458, Seoul, South Korea. IEEE, (2001)
8. Blumberg, B.M.: Action selection in hamsterdam: lessons from ethology. In: Proceedings of the third International Conference on the Simulation of Adaptive Behaviour, Brighton. The MIT Press, Cambridge (1994)
9. Brooks, R.A.: A layered intelligent control system for a mobile robot. In: Faugeras, O.D., Giralt, G. (eds.) Robotics Research, The Third International Symposium. MIT press, (1986)
10. Brookshire, J., Singh, S., Simmons, R.: Preliminary results in sliding autonomy for assembly by coordinated teams. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), pp. 706–711 (2004)
11. Burgard, W., Cremers, A.B., Fox, D., Hahnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S.: Experiences with an interactive museum tour-guide robot. Artif. Intell. **114**(1–2), 3–55 (1999)
12. Connell, J.: SSS: A hybrid architecture applied to robot navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation (1992)
13. Davinic, N., Arkus, A., Chappie, S., Greenberg, J.: Cost-benefit analysis of on-orbit satellite servicing. J. Reducing Space Mission Cost **1**(1), 27–52 (1988)
14. Fernyhough, J., Cohn, A.G., Hogg, D.C.: Constructing qualitative event models autmatically from video input. Image Vision Comput. **18**, 81–103 (2000)
15. Funge, J., Tu, X., Terzopoulos, D.: Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In: Rockwood, A. (ed.), Proceedings of the Conference on Computer Graphics (Siggraph99), NY. ACM Press, NewYork (8–13 August 1999)
16. Gat, E.: Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In: Proceedings of Tenth National Conference on Artificial Intelligence, Menlo Park, CA, USA (1992)
17. Gillett, R., Greenspan, M., Hartman, L., Dupuis, E., Terzopoulos, D.: Remote operations with supervised autonomy (rosa). In: Proceedings 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space (ISAIRAS '01), Montreal (2001)
18. Greenspan, M., Jasiobedzki, P.: Pose determination of a free-flying satellite. In: Motion Tracking and Object Recognition (MTOR02), Las Vegas, NV (24–27 June 2002)
19. Gurtuna, O.: Emerging space markets: engines of growth for future space activities (2003). http://www.futuraspace.com/EmergingSpaceMarkets_fact_sheet.htm.
20. Howarth, R.J., Buxton, H.: Conceptual descriptions from monitoring and watching image sequences. Image Vis. Comput. **18**, 105–135 (2000)
21. Mobile Servicing System (MSS) to User (Generic) Interface Control Document (1997) http://www.spaceref.com
22. Hull, C.: Principles of behavior. Appleton–Century–Crofts, New York (1943)
23. Jasiobedzki, P., Greenspan, M., Roth, G.: Pose determination and tracking for autonomous satellite capture. In: Proceedings of the sixth International Symposium on Artificial Intelligence and Robotics & Automation in Space (i-SAIRAS 01), Montreal, Canada (2001)
24. Jasiobedzki, P., Greenspan, M., Roth, G., Ng, H.K., Witcomb, N.: Video-based system for satellite proximity operations. In: seventh ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA 2002), ESTEC, Noordwijk, The Netherlands (19–21 November 2002)
25. Jenkin, M., Bains, N., Bruce, J., Campbell, T., Down, B., Jasiobedzki, P., Jepson, A., Majarais, B., Milios, E., Nickerson, B., Service, J., Terzopoulos, D., Tsotsos, J., Wilkes, D.: Ark: autonomous mobile robot for an industrial environment. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '94), pp. 1301–1308, Munich, Germany (1994)
26. Lespérance, Y., Reiter, R., Lin, F., Scherl, R.: GOLOG: a logic programming language for dynamic domains. J. Logic Progr. **31** (1–3), 59–83 (1997)
27. Liu, M., Jasiobedzki, P.: Behaviour based visual servo controller for satellite capture. In: Proceeding 7th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA 2002), (19–21 November 2002)
28. Middleton, R., Waltz, D., Schrock, S. (eds.) Satellite Servicing Technology Development Missions, Cocoa Beach, FL, USA (26–28 April 1984)
29. Minsky, M.: The Society of Mind. Simon and Schuster, Inc., New York (1985)
30. Jet Propulsion Labs, NASA: Mars exploration rover mission home (2004). http://www.marsrovers.nasa.gov
31. Nilsson, N.J.: Shakey the robot. Technical Report 323, Artificial Intelligence Center. SRI International, USA (1984)
32. Peterson, L.R., Peterson, M.J.: Short-term retention of individiual verbal items. J. Exp. Psychol. **58**(3), 193–198 (1959)
33. Polites, M.: An assessment of the technology of automated rendezvous and capture in space. Technical Report NASA/TP-1998-208528, Marshall Space Flight Center, Alabama, USA (1998)
34. Qureshi, F.: Intelligent Perception in Virtual Sensor Networks and Space Robotics. PhD thesis, University of Toronto (2006, in preparation)
35. Reiter, R.: Knowledge in Action—Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press, Cambridge (2001)
36. Roberts, L.: Machine perception of 3-D solids. In: Trippit, J.T., Berkowitz, D.A., Chapp, L.C., Koester, C.J., Vanderburgh, A. (eds.) Optical and Electro-Optical Information Processing. pp. 159–197. MIT Press, New York (1965)
37. Roth, G., Whitehead, A.: Using projective vision to find camera positions in an image sequence. In:Vision Interface (VI 2000), pp. 87–94, Montreal, Canada (14–17 May, 2000)

38. Sacerdoti, E.D.: Planning in a hierarchy of abstraction spaces. Artif. Intell. **5**(2) (1974)
39. Sacerdoti, E.D.: The nonlinear nature of plans. In: Proceedings of the Fourth International Joint Conference on Artificial Intelligence (1975)
40. Sellner, B., Hiatt, L., Simmons, R., Singh, S.: Attaining situational awareness for sliding autonomy. In: Proceedings first Annual Conference on Human–Robot Interaction (HRI2006), Salt Lake City, Utah, USA (2006)
41. Shao, W., Terzopoulos, D.: Autonomous pedestrians. In: Proceedings ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 19–28, Los Angeles, CA (2005)
42. Terzopoulos, D., Tu, X., Grzeszczuk, R.: Artificial fishes with autonomous locomotion, perception, behavior, and learning in a simulated physical world. In: Brooks, R.A., Maes, P. (eds.) Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems Artificial Life IV, Cambridge, MA, USA MIT Press, Cambridge (1994)
43. Tsotsos, J.K.: A 'complexity level' analysis of immediate vision. International Journal of Computer Vision **2**(1), 303–320 (1988)
44. Tu, X., Terzopoulos, D.: Artificial fishes: physics, locomotion, perception, behavior. In: Proceedings of SIGGRAPH '94, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, ACM Press (1994) ISBN 0-89791-667-0
45. Wertz, J., Bell, R.: Autonomous rendezvous and docking technologies—status and prospects. In: SPIE's 17th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls, Orlando, FL (21–25 April 2003)

Author biographies

Faisal Qureshi obtained a PhD in Computer Science from the University of Toronto in 2007. He also holds an M.Sc. in Computer Science from the University of Toronto, and an M.Sc. in Electronics from Quaid-e-Azam University, Pakistan. His research interests include sensor networks, computer vision, and computer graphics. He has also published papers in space robotics. He has interned at ATR Labs (Kyoto, Japan), AT&T Research Labs (Red Bank, NJ, USA), and MDA Space Missions (Brampton, ON, Canada). He is a member of the IEEE and the ACM.

Demetri Terzopoulos is the Chancellor's Professor of Computer Science at the University of California, Los Angeles. He graduated from McGill University and obtained a PhD degree in MIT in 1984. He is a fellow of the IEEE, a fellow of the Royal Society of Canada, and a member of the European Academy of Sciences. His many awards include an academy award for technical achievement from the Academy of Motion Picture Arts and Sciences for his pioneering work on physics-based computer animation. He is one of the most highly cited computer scientists and engineers in the world, with approximately 300 published research papers and several volumes, primarily in computer graphics, computer vision, medical imaging, computer-aided design, and artificial intelligence/life.