Autonomous Capture of a Tumbling Satellite

Ioannis Rekleitis, Eric Martin, Guy Rouleau, Régent L'Archevêque, Kourosh Parsa, and Eric Dupuis
Canadian Space Agency
Space Technologies 6767 route de l'Aéroport
Longueuil, QC J3Y 8Y9, Canada
e-mail: Ioannis.Rekleitis@space.gc.ca
e-mail: EricMartin@space.gc.ca
e-mail: Guy.Rouleau@space.gc.ca
e-mail: Regent.LArcheveque@space.gc.ca
e-mail: Kourosh.Parsa@space.gc.ca
e-mail: Erick.Dupuis@space.gc.ca

Received 5 June 2006; accepted 12 February 2007

In this paper, we describe a framework for the autonomous capture and servicing of satellites. The work is based on laboratory experiments that illustrate the autonomy and remote-operation aspects. The satellite-capture problem is representative of most on-orbit robotic manipulation tasks where the environment is known and structured, but it is dynamic since the satellite to be captured is in free flight. Bandwidth limitations and communication dropouts dominate the quality of the communication link. The satellite-servicing scenario is implemented on a robotic test-bed in laboratory settings. The communication aspects were validated in transatlantic tests.

1. INTRODUCTION

Over the past few decades, robots have played an increasingly important role in the success of space missions. The Shuttle Remote Manipulator System, also known as Canadarm, has made the on-orbit maintenance of assets such as the Hubble Space Telescope possible. On the International Space Station (ISS), Canadarm2 has been a crucial element in all construction activities. Its sibling, named Dextre, will be essential to the maintenance of the ISS. JAXA also demonstrated the use of a robotic arm in the ETS-VII space servicing demonstration mission in 1998–1999 (Kasai, Oda & Suzuki, 1999).

In light of the missions currently being planned by space agencies around the world, the coming years will only show an increase in the number and the criticality of robots in space missions. Examples include the Orbital Express mission of the U.S. Defence Advanced Research Project Agency (DARPA)

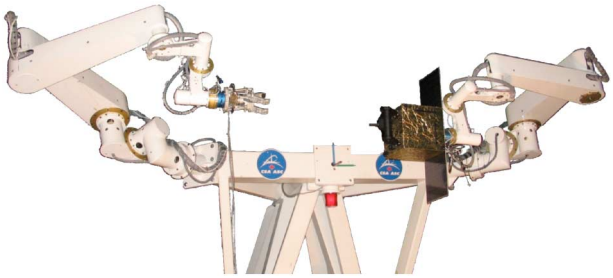


Figure 1. A dual manipulator system that simulates the tracking and capture scenario; the manipulator on the left is equipped with a hand, and the manipulator on the right is mounted by a mock-up satellite.

(Whelan, Adler, Wilson, & Roesler, 2000), and the ConeXpress Orbital Life Extension Vehicle (CX-OLEV™) of Orbital Recovery (Wingo et al., 2004).

One important area for the application of space robotics is autonomous on-orbit servicing (OOS) of failed or failing spacecrafts. A common characteristic to most OOS missions is the necessity to approach and capture the spacecraft to be serviced. Because the communication link between the ground operator and the servicer will be subject to latency, bandwidth limitations, and communication drop-outs, some amount of on-board autonomy will be required to perform the rendezvous and capture in a safe and efficient manner.

In addition, the commercial viability of such operations will require the usage of an efficient process for the planning, verification, and execution of operations. In this paper, we describe the laboratory experiments that verify the feasibility of our approach to perform autonomous missions by demonstrating this aspect of the process. In particular, we report on the use of a manipulator system named CART, which has two 7-degree-of-freedom arms to demonstrate an autonomous capture of a tumbling satellite. As shown in Figure 1, a mock-up satellite, the *target*, is mounted on one arm while the second arm equipped with a robotic hand, the *chaser*, approaches and captures the target.

In the next section, we present related work. Section 3 discusses an outline for a typical OOS mission, which provides the motivation for the research reported in this paper. Section 4 provides an overview of the autonomous aspects of the work. Trajectory planning for the two satellites is outlined in Section 5. Section 6 contains the experimental results, and the last section presents our conclusions.

2. RELATED WORK

For many years, robots such as Canadarm and Canadarm2 have been used in space to service expensive space assets (Stieber, Sachdev & Lymer, 2000). Canada has also developed another robot called Dextre for the ISS; Dextre is to be launched in 2007 and will be used to perform maintenance tasks. Other countries are also developing robots for the ISS: The European Space Agency (ESA) has developed the European Robotic Arm (ERA) (Didot, Oort, Kouwen & Verzijden, 2001), and the Japanese Space Agency has developed the JEMRMS (Sato & Doi, 2000).

In order to speed up the acceptance of OOS and to decrease operational costs, a few technology demonstration missions have already been or will soon be conducted. Each mission demonstrates some of the typical operations described in Section 3. As early as 1989, JPL demonstrated in a lab the capture of a rotating satellite (Wilcox, Tso, Litwin, Hayati & Bon, 1989) using a camera system (Gennery, 1992). Japan first conducted the ETS-VII mission in 1998–1999 (Kasai et al., 1999). ETS-VII involved the capture of a target satellite using a chaser satellite equipped with a robotic arm. Both satellites were launched together to minimize risks associated with the rendezvous portion of the mission. The robotic capture was performed while the two satellites were still tied using the latching mechanism, again for reducing the risks (Yoshida, 2003; Yoshida, 2004). The mission goal was successfully accomplished. In the framework of this mission, future work is also discussed for a non-cooperative satellite (Yoshida et al., 2004).

DARPA is currently funding the development of the Orbital Express mission to be launched in 2006[1] (Potter, 2002). This mission intends to prove the feasibility of OOS and refueling. The Orbital Express's servicer spacecraft ASTRO is equipped with a robotic arm to perform satellite capture and ORU exchange operations. Recently, the US Air Force Research Lab demonstrated key elements of extended-proximity operations with the XSS-11 mission (Grossman & Costa, 2003; Lewis, 2004). A mission by NASA with

[1]Orbital Express may be already launched at the time of publication.

similar objectives, DART, flew in 2005 (Rumford, 2003). The objective was to perform an autonomous rendezvous; unfortunately, the mission failed.

The first commercial mission could be realized by Orbital Recovery Limited, who are developing the technologies to permit life extension of spacecraft using their CX-OLEV™. This spacecraft, as explained by (Wingo et al., 2004), is designed to mate with any three-axis stabilized spacecraft and would have sufficient supplies to keep a 3000-kg parent spacecraft in a geo-stationary orbit for up to an additional 10 years of life. The first mission has been planned for 2008.

The TEChnology SAtellites for demonstration and verification of Space systems (TECSAS) is a mission proposed by DLR (Sommer, 2004; Martin, Dupuis, Piedboeuf & Doyon, 2005). The objectives of the mission are to prove the availability and advanced maturity of OOS technologies, and the mastering of the capabilities necessary to perform unmanned on-orbit assembly and servicing tasks. For TECSAS, a servicer satellite carrying a robotic subsystem and a client satellite would be launched together. The mission intends to demonstrate the various phases required for an OOS mission: far rendezvous, close approach, inspection fly around, formation flight, capture, stabilization and calibration of the coupled system, flight maneuvers with the coupled system, and manipulation on the target satellite. This mission is currently being redefined.

There have also been several spacecrafts designed for transporting logistics to the ISS such as Russia's Progress, Europe's ATV (Boge & Schreutelkamp, 2002), and Japan's HTV (Kawasaki, Imada, Yamanaka & Tanaka, 2000). Many key technologies required for OOS have already been or will be demonstrated with these missions.

Because of the unfortunate Columbia accident, NASA had considered using a robotic mission to rescue the ailing Hubble Space Telescope. A rescue mission using robotic arms derived from the ISS's Dextre was tentatively selected to replace the batteries, gyroscopes, and possibly a scientific instrument of the Hubble. A de-orbiting module was also to be carried by the chaser spacecraft to the orbit. This module was to be attached to the Hubble for the purpose of de-orbiting the Hubble at the end of its life (King, 2004). However, at the time of writing this paper, this mission has been cancelled.

Other missions are being considered to demonstrate spacecraft servicing technologies. One of them is SUMO, which is sponsored by DARPA and executed by the Naval Center for Space Technology at the Naval Research Laboratory. (Bosse et al., 2004) state that the purpose of the program is to demonstrate the integration of machine vision, robotics, mechanisms, and autonomous control algorithms to accomplish autonomous rendezvous and also the grapple of a variety of interfaces traceable to future spacecraft servicing operations. However, at the time of writing this paper, this demonstration mission, initially planned for 2008, is still unapproved, although laboratory work is being done to develop the technologies. Another mission is CESSORS, which is currently being planned by Shenzhen Space Technology Center of China. According to (Liang, Li, Xue & Qiang, 2006), included in the mission will be the detection, fly-around, and autonomous rendezvous and capture of a floating target, as well as the teleoperation of the robotic manipulator mounted on the chaser satellite; the authors, however, do not specify any time frame for the mission.

To determine the technology readiness level of space servicing technologies, CSA closely studied the missions mentioned above. These missions have either occurred, are being conducted, or are in the planning phase. The operations involved in these missions fit the typical descriptions given in Section 3. Each operation may be performed in one of three different modes: manual, semi-autonomous, and autonomous. In the manual mode, an operator is responsible for conducting the mission by sending elementary commands or by using hand controllers. In the semi-autonomous mode, an operator is still responsible for performing the operation, but part of the operation is automated using scripts that contain the elementary commands or by using higher-level commands decomposed automatically into elementary commands. Finally, in the autonomous mode, the operation is performed fully autonomously with a minimal number of interventions from the operator. The operator sends only high-level commands, e.g., "Capture." An operator may supervise the mission and be ready to send an abort command if needed.

Table I lists, to the best of the authors' knowledge, all relevant space-servicing missions. The operations performed in each mission are identified, differentiating if they were performed manually, semi-autonomously, or autonomously. A "C" is used to indicate an operation performed on a cooperative

Table I. Comparative study of OOS missions.

		Robotic Missions								Non-robotic Missions					
Operations	Mode	Canadarm (1981–)	ETS-7 (1998–1999)	Canadarm2 (2001–)	Orbital Express (2006)	Dextre (2007)	ERA (2009)	JEMRMS (2008)	TECSAS (2010)	Progress (1978–)	XSS-11 (2005–)	DART (2005)	ATV (2006)	CX-OLEV (2008)	HTV (2009)
Far-range rendezvous	M	-	C	-	-	-	-	-	C	C	-	-	C	NC	C
	S	-	C	-	-?	-	-	-	C	C	-	-	C	-	C
	A	-	C	-	C?	-	-	-	C	C	NC	C	-	-	C
Close-range rendezvous	M	-	C	-	-	-	-	-	C	C	-	-	C	NC	C
	S	-	C	-	-?	-	-	-	C	C	-	-	C	-	C
	A	-	C	-	C?	-	-	-	C	C	NC	F	-	-	C
Docking	M	-	C	-	-	-	-	-	-	C	-	-	C	NC	-
	S	-	C	-	C	-	-	-	-	C	-	-	C	-	-
	A	-	-	-	-	-	-	-	-	C	-	-	-	-	-
Undocking	M	-	C	-	-	-	-	-	-	C	-	-	C	NC	-
	S	-	C	-	C	-	-	-	-	C	-	-	C	-	-
	A	-	C	-	-	-	-	-	-	C	-	-	-	-	-
Robotic capture	M	C	-	C	C	-	C	C	NC	-	-	-	-	-	-
	S	-	C†	-	C	-	-	-	NC	-	-	-	-	-	-
	A	-	-	-	-	-	-	-	NC	-	-	-	-	-	-
Robotic release	M	C	-	C	C	-	C	C	NC	-	-	-	-	-	-
	S	-	C†	-	C	-	-	-	NC	-	-	-	-	-	-
	A	-	-	-	-	-	-	-	NC	-	-	-	-	-	-
Berthing	M	C	C	C	C	-	C	C	C	-	-	-	-	-	-
	S	-	C	-	C	-	-	-	C	-	-	-	-	-	-
	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-
De-berthing	M	C	C	C	C	-	C	C	C	-	-	-	-	-	-
	S	-	C	-	C	-	-	-	C	-	-	-	-	-	-
	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ORU operations	M	-	C	-	C	C	-	C	C	-	-	-	-	-	-
	S	-	C	-	C	-	-	-	C	-	-	-	-	-	-
	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Orbit transfer of system	M	-	-	-	C	-	-	-	NC	-	-	-	-	-	-
	S	-	-	-	C?	-	-	-	NC	-	-	-	-	-	-
	A	-	-	-	-?	-	-	-	NC	-	-	-	-	-	-
Refueling/ Fluid transfer	M	-	C	-	C	-	-	-	-	-	-	-	-	-	-
	S	-	C	-	C	-	-	-	-	-	-	-	-	-	-
	A	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Fly around	M	-	-	-	C	-	-	-	C	-	-	-	-	NC	-
	S	-	-	-	C	-	-	-	C	-	-	-	-	-	-
	A	-	-	-	-	-	-	-	C	-	NC	F	-	-	-
Capture	M	-	-	-	-	-	-	-	NC‡	-	-	-	-	-	-
	S	-	-	-	-	-	-	-	NC‡	-	-	-	-	-	-
	A	-	-	-	-	-	-	-	NC‡	-	-	-	-	-	-

† These operations were performed with the target satellite moving by partially releasing the docking mechanism; see (Inaba and Oda, 2000).

‡ This operation will be performed with or on a slowly tumbling satellite.

Table II. Legend for Table I.

M	Manual Mode
S	Semi-Autonomous Mode
A	Autonomous Mode
-	Not demonstrated
C	Demonstration for a cooperative satellite
NC	Demonstration for a non-cooperative satellite
	Already demonstrated in the past
	Will be demonstrated
F	Mission failed

satellite,[2] while "NC" is used for an operation involving a non-cooperative spacecraft. A dash indicates that the operation was not performed during that particular mission. A question mark indicates information that cannot be confirmed because part of the information is classified. Finally, we use a shading scheme to differentiate operations already demonstrated (gray) from those planned to be demonstrated in the future (light gray). Dark gray is used to indicate an operation that was planned to be demonstrated in a particular mission, but that was not successful. This notation is summarized in Table II for easy reference.[3]

Over the last several years, several control architectures have been proposed to address different problems associated with space robotics. In particular, several architectures have been developed to address the issues associated with ground control. Since the usual mode of operation for space robotics in the past has been tele-operation (Hirzinger, Brunner, Dietrich & Heindl, 1993) with direct operator control or supervision, most of the approaches have not focused on the implementation of autonomy. In the late 1990's, the Canadian Space Agency (CSA) and their industrial partner, MD Robotics, developed the Intelligent, Interactive Robotic Operations (IIRO) framework (Dupuis, Gillett, Boulanger, Edwards & Lipsett, 1999), which allowed the tele-operation of remote equipment in operational settings. The Remote Operations with Supervised Autonomy (ROSA) architecture was a follow-on to IIRO and addressed the issue of scripted control and basic autonomy requirements (Dupuis and Gillet, 2002; Dupuis, Gillett, L'Archevêque & Richmond, 2001). ROSA has been used as the basis for the development of the ground control station for the robotic elements on the Orbital Express satellite-servicing mission. Similar architectures were developed in Europe at the same time. The Modular Architecture for Robot Control (MARCO) developed by DLR addresses similar issues in the context of tele-operation, ground control, and tele-presence (Brunner, Landzettel, Schreiber, Steinmetz & Hirzinger, 1999). The MARCO architecture and its relatives have been used on several missions including ROTEX and ETS-7. Two other architectures were also developed under the leadership of the European Space Agency: FAMOUS (Fontaine, Steinicke & Visentin, 1996) and DREAMS also concentrated on the issues associated with the tele-operation of robots in space. In both cases, special attention was dedicated to the issues surrounding planning, verification, and execution of command sequences. Similarly, in the US, the JPL has developed a set of tools for rover ground control from planning to post-flight analysis. This tool, called RSVP (Rover Sequencing and Visualization Program) has been used with the Mars exploration rovers (Maxwell, Cooper, Hartman, Wright & Yen, 2004).

Despite the wealth of research in autonomous robotics and in control architectures for space robots, relatively little has been done to address specifically the needs of autonomous space robots. NASA/JPL have been developing/proposing three different architectures for applications with a higher degree of autonomy in the last few years: FIDO, CAMPOUT, and CLARAty. FIDO (Hoffman, Baumgartner, Huntsberger & Schenker, 1999) is a three-layer software architecture for real-time control of single rover systems equipped with scientific payloads. CAMPOUT (Huntsberger et al., 2003) is a control architecture for the real-time operation of multiple rovers. CLARAty (Nesnas, Wright, Bajracharya, Simmons & Estlin, 2003; Volpe et al., 2001) is a proprietary architecture that is becoming a requirement for many missions. The main goal of CLARAty is to provide a systematic framework for treating all the different vehicles/robots/instruments involved in Mars missions. CLARAty is object oriented and according to the publications provides a high degree of reusability. In a similar effort, the Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) has developed a software architecture for autonomy (Alami, Chatila, Fleury, Ghallab & Ingrand, 1998). This architecture is

[2]In this paper, a cooperative satellite is defined as a satellite designed to be serviced and not tumbling. On the other hand, a non-cooperative satellite is understood as one either not designed to be serviced or that it is tumbling.

[3]It is also important to note that the Hubble *robotic* repair/de-orbit Mission was not included in Table I since, as of the time of writing this paper, it is considered cancelled. The same applies to the SUMO mission since it has yet to be approved for a space demonstration.

composed of two main levels: a decision level whose role is to make plans and supervise their execution and a functional execution level whose role is to carry out low-level actions.

In parallel with these efforts, the Canadian Space Agency has been developing the Autonomous Robotics and Ground Operations (ARGO) software suite (Dupuis, L'Archevêque, Allard, Rekleitis & Martin, 2006). ARGO provides a framework for the integration of the space operations process from planning to post-flight analysis. The objective of ARGO is to reduce operational costs and increase efficiency by providing operator aids and permitting the implementation of a level of autonomy appropriate to the application.

3. PHASES OF A TYPICAL OOS MISSION

A typical on-orbit servicing mission is conducted in a series of operations. Some of these operations such as rendezvous and docking are vehicular, namely, they are performed by the chaser satellite, while others such as capture and berthing are robotic and are to be performed by the robotic arm of the chaser satellite. Although the current goal of our work is only the autonomous capture of a tumbling satellite, we will also describe some other critical phases of a typical on-orbit servicing mission in order to provide the reader with an insight into the intricacies involved in OOS missions.

3.1. Long-Range Rendezvous

During this phase, the chaser satellite has completed its orbit phasing and has entered the same orbit as the target satellite or a slightly lower orbit, called drift orbit. At this point the chaser is within a distance of 5 km to 300 m of the target, as seen in Figure 2(a).

The chaser satellite is mainly guided and navigated using absolute navigation aids, e.g., sun and earth sensors, star trackers, and GPS, with some help from relative navigation equipment such as radar sensors or lidar. The attitude match between the two satellites in this phase is not a very important factor. Actions in this phase include acquiring and updating the orbit knowledge of the target spacecraft; synchronizing the mission time-line; and achieving the necessary position, velocity, and angular velocity of the chaser satellite with respect to the target spacecraft for the subsequent close-range rendezvous and docking[4] or capture.

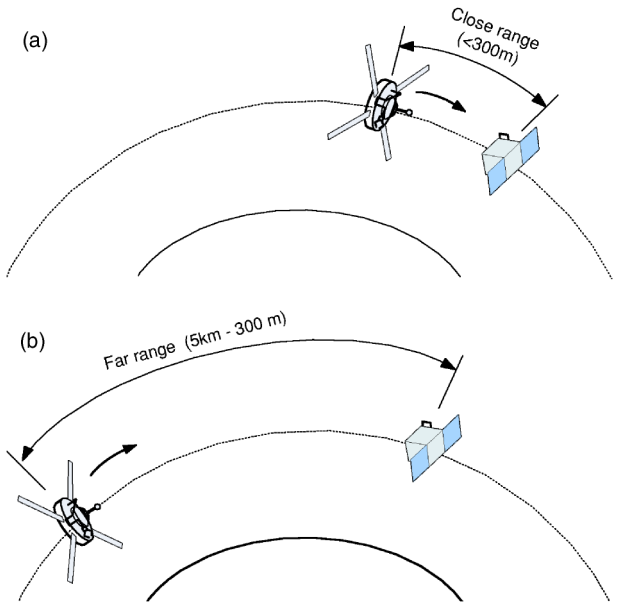


Figure 2. (a) Long-range rendezvous; (b) short-range rendezvous.

3.2. Short-Range Rendezvous

When the chaser satellite gets within 300 m of the target satellite, the next phase, i.e., the short range rendezvous, starts [see Figure 2(b)]. This phase is active for distances from 300 m to several meters. The operation has to be controlled by relative-navigation techniques via sensing the relative position, attitude, and velocities of the target satellite directly. The chaser satellite has to reduce not only the distance but also the relative attitude as well as the relative translational and angular velocities of the two satellites. Depending on the design of the docking or the robotic capture interface, the required accuracies in position, orientation, and translational and angular velocities are in the order of 0.01 m, 1 deg, 0.01 m/s, and 1 deg/s, respectively.

The operation can be autonomously controlled

[4]As explained by (Fehse, 2003), docking is the process whereby the guidance, navigation, and control system (GNC) of one spacecraft controls the state of that spacecraft to bring it into entry to the docking interface of a second spacecraft.

using onboard vision sensors and the control system of the chaser. The ground control system is only responsible for monitoring the operation and providing emergency safety measures. The transition to and from the short-range rendezvous mode under nominal conditions is driven by the relative distance and velocity between the two satellites as provided by the on-board sensors. It is possible to exit the short-range rendezvous mode upon encountering anomalous conditions such as losing visual contact with the target or incorrect approach rates, which will require a transition to error-recovery modes.

3.3. Station Keeping

Before attempting any contact operations, the chaser has to ensure that it is in a safe trajectory with respect to the target satellite. This is achieved during the formation-flight phase. During this phase, the chaser satellite is in very close proximity of the target spacecraft such that the target satellite is within the reach of the chaser robotic arm. The relative position and orientation of the chaser satellite with respect to the target satellite must be strictly controlled in order to avoid collisions.

Because the satellites are close to each other during this phase, the communication delays and blackouts could result in damage to either or both spacecrafts through a collision or contamination by a plume from the propulsion system. It is therefore imperative to close a control loop on board to maintain a safe distance and to deal with any anomalous conditions such as drift of the satellites or blinding of the vision sensor.

3.4. Capture

The capture operation is the phase of the mission with the highest risk because it involves contact between the two spacecrafts, and because it requires a timely cooperation of the control systems on both satellites. In this phase, as shown in Figure 3, the chaser's robotic arm approaches the free-floating target satellite and grasps it. More specifically, the operation includes the following steps:

(1) Power on and unfold the robotic arm.
(2) The arm maneuvers toward the target satellite and aligns its end-effector with the grapple interface.
(3) Upon completing alignment of the robot end-effector and the grapple interface, the robotic arm goes into either force-control mode, or limping mode, namely, the joints become passive.
(4) The chaser may also deactivate its Attitude and Orbit Control System (AOCS), so that it becomes free-floating. The AOCS of the target should also be deactivated before capture.
(5) The robot end-effector then grasps the target spacecraft through the grapple interface.
(6) Upon the completion of firm grappling, the robotic arm returns to its position-control mode to prevent collisions between the chaser and the captured satellite. The chaser should then reactivate its AOCS for the stabilization of the coupled system.

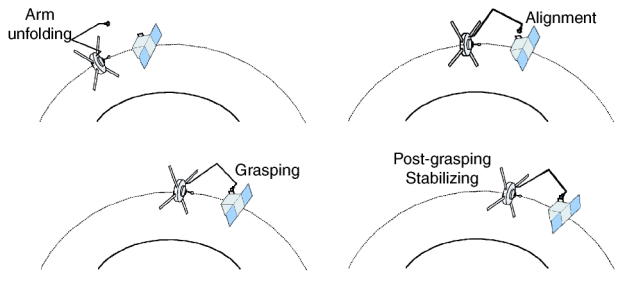


Figure 3. Capture operation.

Before capture, the target satellite must be in safe-hold mode (SHM), freely floating in its orbit. Visual servoing of the arm is used for alignment. After being caught, the motion of the target spacecraft should be completely controlled by the robotic arm. Anomalies that can be encountered during this phase include the possibility of the target spacecraft drifting out of the capture envelope of the manipulator; the blinding of the vision sensor; the reduction of the distance between the two spacecrafts below an acceptable limit; and a failed capture, which may induce tumbling of the target satellite. This phase is at the core of our experiments.

3.5. Securing the Target Satellite

After the target satellite is captured, different procedures can be followed to ensure that the two spacecrafts fly together without putting any undue stress on the link between the two of them. When servicing is completed a reversal of the securing procedure is performed to release the target spacecraft.

If a robotic arm was used to conduct the capture maneuver, then securing the target satellite can be done by berthing it to the chaser spacecraft. The robotic arm maneuvers the target satellite toward the berthing interface and makes the required alignment. Then, the arm pushes the target satellite against the chaser satellite until the two parts of the interface of the two satellites are latched. A locking mechanism is then activated to rigidly fix the two satellite bodies to each other, creating one compound body.

During berthing, the chaser satellite may leave its Attitude and Orbit Control System (AOCS) active so as to compensate for some disturbances due to the contact in the interface or due to the motion of the manipulator. If the chaser satellite is in free-floating condition during berthing, it should reactivate its AOCS immediately afterwards to stabilize the compound system.

The reverse of the berthing operation is called de-berthing, whereby the two satellites are detached and moved away from each other by the robotic arm. The chaser satellite may or may not be under active AOCS control during de-berthing, but the target satellite must be in safe-hold mode with its AOCS deactivated. De-berthing is considered successfully completed only after the two satellites are at a safe distance, but still connected by the robotic arm.

In the absence of a robotic arm, securing the target satellite can also be done by docking. During docking, the chaser satellite makes a final closing and makes a physical contact with the free-floating target satellite using its momentum (or relative velocity) through a docking interface. As a result, the two satellites are physically rigidly attached together. The target satellite must be in the safe-hold mode (SHM) such that the body of the satellite is free-floating in the orbit. The reverse operation, called undocking, is considered as successfully completed when the spacecrafts have been separated and are at a safe distance.

3.6. Service Operations

There are two major operations that are considered for on-orbit servicing of satellites: inspections and orbital replacement unit (ORU) operations.

ORU replacement operations are typically carried using a robotic arm after the target satellite is captured and securely berthed to the chaser satellite. Potential operations include replacing a battery or a fuel tank, installing a refuelling interface, and opening and closing a hinged door, among others.

Inspection can also be done robotically after the target satellite is secured or during the station keeping phase by flying the chaser around the target satellite. Such maneuvers are quite risky because of the possibility of collision between the two spacecraft or damage through impingement of the chaser's thrust plume.

3.7. Release

This is the reverse of the capture operation. The operation includes the following steps: releasing the grapple interface from the robot end-effector, and moving the arm away from the target satellite to avoid collision with the released satellite. Once the release is completed, the chaser is ready to perform an orbital maneuver to move away from the target satellite.

4. AUTONOMY

In our experiments, we demonstrated the fully autonomous capture of a free-flyer object using the robotic system equipped with two 7-degree-of-freedom arms shown in Figure 1. One arm emulated the free-flyer dynamics, while the other acted as the chaser robot equipped with a SARAH hand, developed by Laval University (Laliberté, Birglen & Gosselin, 2002). The Laser Camera System (LCS) from Neptec was used to determine the pose of the target; this information is employed by the trajectory planner of the chaser robot. Moreover, two surveillance cameras were used to provide video output to the remote operator. A hierarchical finite state machine engine was used to coordinate the autonomous capture task and to provide feedback to the operator. The only role of the remote operator was to initiate the capture by submitting the high-level command and to monitor the chaser robot while performing the task. The sequencing of the events, i.e., approach, fly together, and capture, was fully autonomous. In case of an emergency, the operator could send a halt or abort command to the autonomy engine.

4.1. Encoding Autonomy

The central tool for the autonomous operation is the Canadian Space Agency's *Cortex* Toolbox, which is

used to implement command scripts and sets of reactive behaviors. Cortex has been developed to ease the development of such behavior, which rapidly becomes labor intensive even for simple systems when using low-level programming languages. Cortex is based on the Finite State Machine (FSM) formalism, which provides a high-level way of creating, modifying, debugging, and monitoring reactive autonomy engines. Two advantages of this representation are its intuitiveness and the ease with which it can be graphically constructed and monitored by human operators.

In general, FSM's are used to represent a system using a *finite* number of configurations, called *states*, defined by the system parameters or its current actions. The system can *transition* from one state to another based on its current state, conditions, and outside events. Conditions on transitions are often referred to as *guards* and are implemented as statements that can be evaluated as being either true or false. Outside events, called *triggers*, make the FSM evaluate its transition's guards and enable a transition to occur.

The concept of hierarchical FSM's allows a high-level FSM to invoke a lower-level FSM. This provides the capability to implement hierarchical decomposition of a high-level task into a sequence of lower-level tasks. If the FSM is implemented in a modular fashion, it allows for the implementation of libraries that provide the operator with the possibility of reusing an FSM in other applications.

The use of an intuitive graphical representation of FSM's, shown in Figures 4(a) and 4(b), allows the developer and the operator alike to concentrate on the problem to be solved instead of concentrating on the programming skills to implement the solution. Moreover, the use of hierarchical FSM's can address a wide spectrum of autonomy levels, from sense-and-react behaviors relying on sensor inputs to high-level mission scripts.

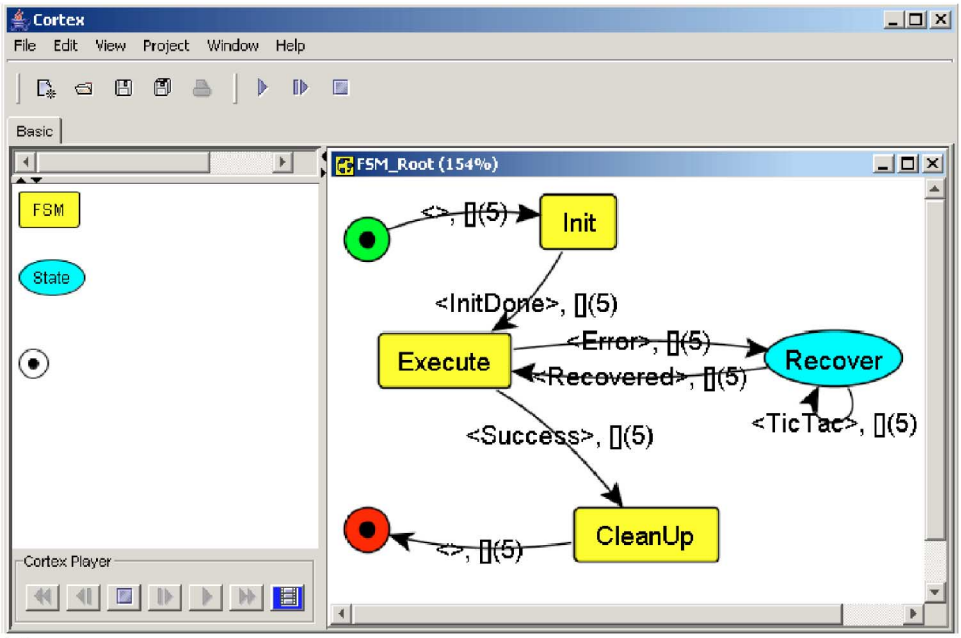
FSM's are assembled graphically using states, sub-FSM's, junctions,[5] and transitions [Figure 4(a)]. The user can add states (blue ellipses) or sub-FSM's (yellow rectangles) from the panel on the left-hand side to the current FSM shown in the right-hand side panel by drag-and-drop. Transitions between states are added by dragging the mouse between two states/FSM's and are represented by arrows. The operator can provide JAVA code snippets for state actions and the transition's guard expressions. In addition, he can define the inputs, local variables, and output parameters of each sub-FSM. The user can also assign a priority to each transition to enforce the order in which their guards are tested during execution. It is worth noting that Cortex provides the user with the capabilities of a periodic trigger that would be activated in user-set intervals. Components from other FSM's can also be graphically "cut-and-pasted" to the current FSM to allow for the reuse of existing FSM's.

The simple FSM presented in Figure 4 illustrates a generic behavior. After some initializations, the behavior is being executed in the main sub-FSM; in the event of an error condition "Error," the flow of execution moves to a recovery state. When the error is fixed, Cortex returns back to the "Execute" sub-FSM; at the event of successful completion, Cortex moves to the "CleanUp" sub-FSM to tidy-up the termination of the behavior and then terminates. Please note that by clearly naming the states, FSM's, and transitions, a new user can easily follow the flow of execution and understand the logic of an existing Cortex engine without knowledge of the low-level details.

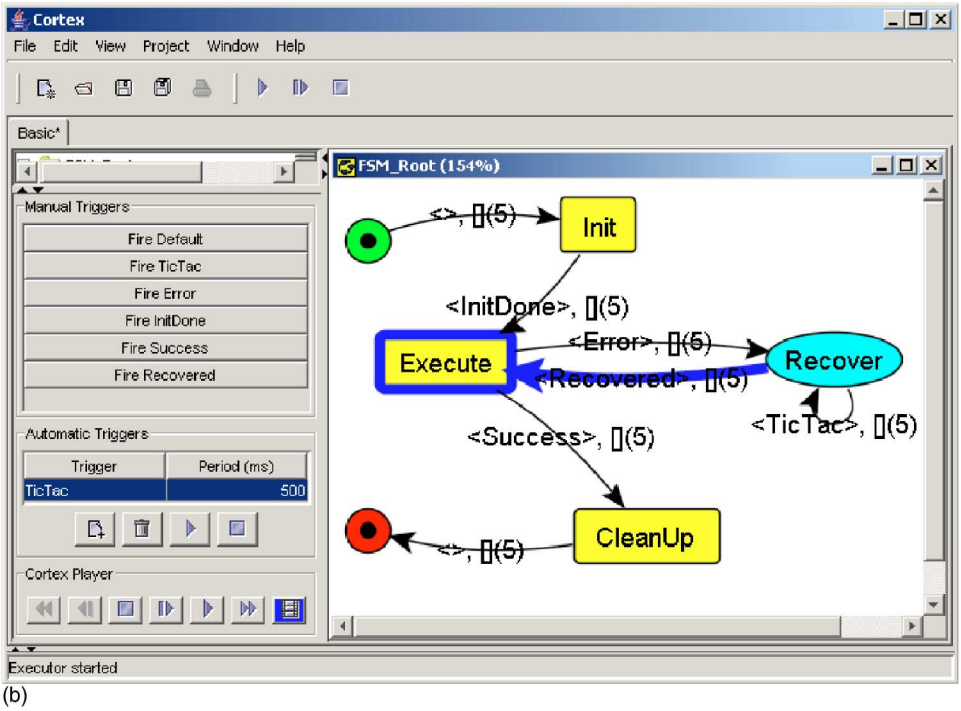
Figure 5 illustrates the implementation of the operations sequence described in Section 3 using the Finite State Machine formalism. Each discrete phase of the operation sequence is represented by a state or sub-FSM as mentioned before. Transitions between phases are represented by arrows linking the states of sub-FSM's. As can be seen in Figure 5, the Cortex engine would revert to an earlier state in the event of a failure at the current state. For example, if during the "CloseRangeApproach" the target satellite drifts out of range, then Cortex would transition back to the "ShortRangeRendezvous" state and attempt a second approach. For simplicity, we have omitted all operator-triggered transitions from Figure 5. Such triggers would be present in order to allow for the ground operator to take over the process in case of an unforeseen emergency.

In our experimental setup, Cortex was used throughout the process as the chaser manipulator and the target satellite were only a few meters apart. Transitions between phases of the operation are triggered by sensory events. The Cortex engine considers anomalies such as the possibility of the target spacecraft to drift out of the capture envelope of the manipulator (through translation or rotation); blinding of the vision sensor or loss of sight; reduction of

[5]Junctions are constructs that are used to combine transitions.



(a)



(b)

Figure 4. The Cortex GUI: (a) FSM editing environment. The user can drag and drop states and sub-FSM's from the left panel to the current FSM and then add transitions. Ellipses (blue) represent states, rectangles (yellow) represent sub-FSM's, and arrows represent transitions from one state/FSM to another. (b) FSM monitoring environment. The user can activate manual or periodic triggers in the left panel and monitor the flow of the autonomy engine at the panel in the right.

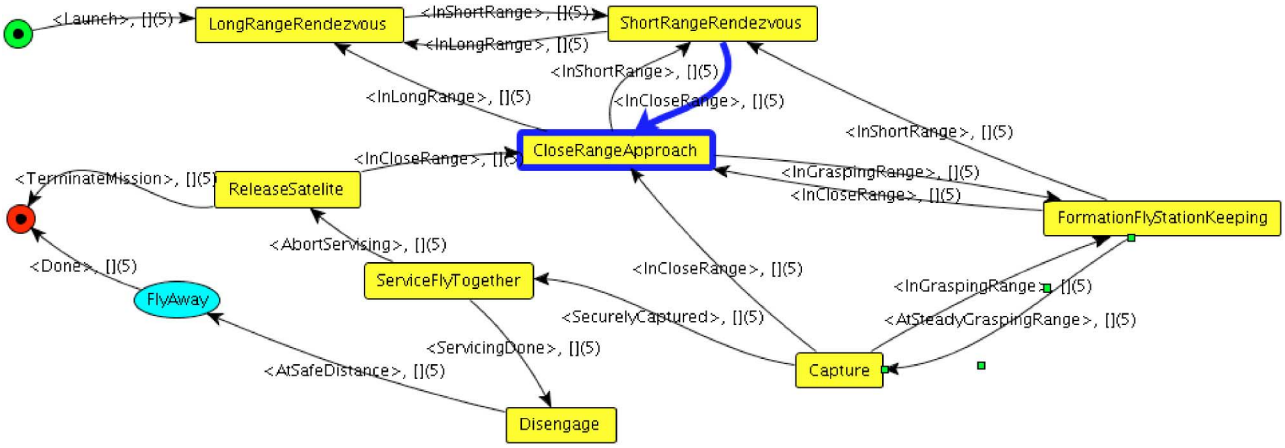


Figure 5. A high-level plan for an OOS mission, implemented using the Cortex framework.

the distance between the manipulator and the target satellite below an acceptable, safe limit; or failed capture.

Figure 6 shows a Cortex implementation of the later phases of a typical OOS scenario after an autonomous far rendezvous has been performed. The operator has overriding control of pausing the process using a soft-stop and then restarting it, or completely terminating the process by using a hard-stop. The actual capture sequence is itself a state machine of three stages: approach, align, and capture. We have used the same sub-FSM "SoftStop" to handle possible error conditions such as vision failures or satellite misalignment, and operator interruptions.

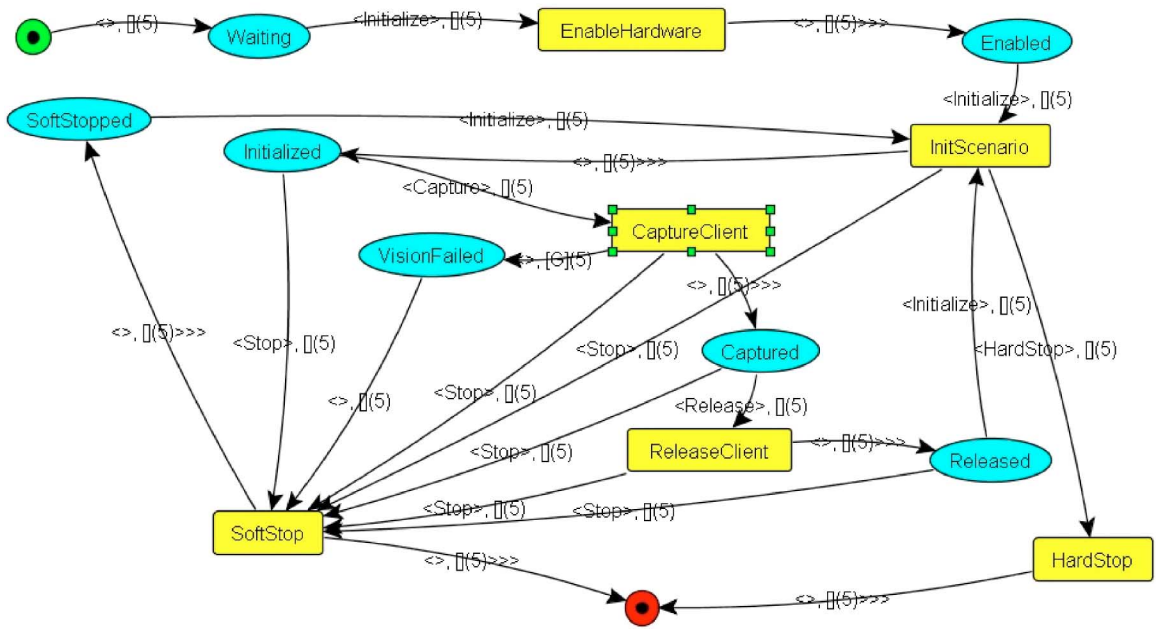


Figure 6. The top-level Cortex diagram of the autonomous-capture scenario.

In all cases, the chaser arm moves back to a safe position and resets itself with respect to the target satellite, then the chaser waits for the LCS to reestablish contact and to start tracking the target satellite; this procedure is independent of the cause of the interruption. In contrast, the "HardStop" case indicates a more drastic failure; thus, there will be no attempt to capture the satellite.

The Cortex toolbox has been applied successfully to encode autonomy behaviors for planetary exploration tasks here at CSA. It is also part of a larger suite of applications named ARGO that includes remote operation-monitoring software and Ground Control Station capabilities; see (Dupuis et al., 2006) for more information. Next, we briefly discuss the Ground Control Station features used in our experiments.

4.2. Operations and Monitoring

One very important aspect of space robotics is the capability to remotely monitor a device and to operate it. In our architecture, the Ground Control Station (GCS) is used first to develop as well as monitor the command scripts and autonomous behaviors using the Cortex Toolbox. In this context, the GCS is connected to a simulator of the remote system, which includes the dynamics of the two satellites, the robot arm, and the vision system. The simulator is then used to validate the command scripts and the autonomous behaviors associated with each phase of the mission. Depending on the predictability of the parameters triggering transitions in the Cortex engine, some phases can be simulated in a very deterministic manner. For example, the manipulation of ORU's is performed in a very controlled and static environment, as the target satellite itself is subject to very little uncertainty. On the other hand, other phases of the mission, such as the capture of the target satellite, will be subject to unpredictable factors such as initial conditions on the satellite attitude. The formal validation of the portions of the command script and autonomous behaviors associated with these events is still an open question; in practice, however, validation is achieved by using a broader range of parameters to systematically verify the plausible outcomes of the mission.

Once the verification is completed, the GCS is reconfigured to connect to the real system. The validated Cortex behaviors and scripts are then uploaded to the system for execution. The synergy of the Cortex autonomy engine and the GCS allows for operator intervention at different phases of an operation without hindering the autonomous nature of the operation.

In our experiments, we used the GCS, shown in Figure 7, to remotely monitor and command the two-arm system from various remote cities, including one from a different continent (trans-Atlantic) over the Internet. Due to the autonomous nature of the experiment, the operator gave only high-level commands. More importantly, the operator was able to monitor the safe execution of the experiments and, for demonstration purposes, to halt and restart the experiment as if he had detected a fault.

5. TRAJECTORY PLANNING

This section presents the algorithms developed to generate the motion for both the target and the chaser satellites. In our scenario, we assume that the target satellite moves at a speed slower than the chaser arm's velocity limits. With this assumption, a strategy involving a progressive reduction of the distance between the target and the chaser is acceptable. This implementation is presented in Section 5.2 after presenting the generation of the target satellite motion in Section 5.1. Finally, in Section 5.3, the redundancy resolution scheme implemented to realize a desired motion of the end-effector in an optimal manner is outlined.

5.1. Target Satellite

Many satellites use momentum wheels to stabilize and to control their attitude. When the attitude-control system fails, due to friction, the angular momentum stored in the wheels is transferred to the satellite body over time. This momentum transfer causes tumbling in the satellite. At the same time, the satellite is under the action of small external, nonconservative moments, which make the satellite tumbling motion occur mostly about its major principal axis (Kaplan, 1976), i.e., the axis about which the satellite moment of inertia is maximum.

Therefore, to mimic the tumbling motion of a satellite, we assume that the satellite is initially rotating about an axis very close to its major principal axis. Then, to create the trajectory, we solve the Euler equations assuming no external moments. It should be noted that, even though there are dissipative mo-

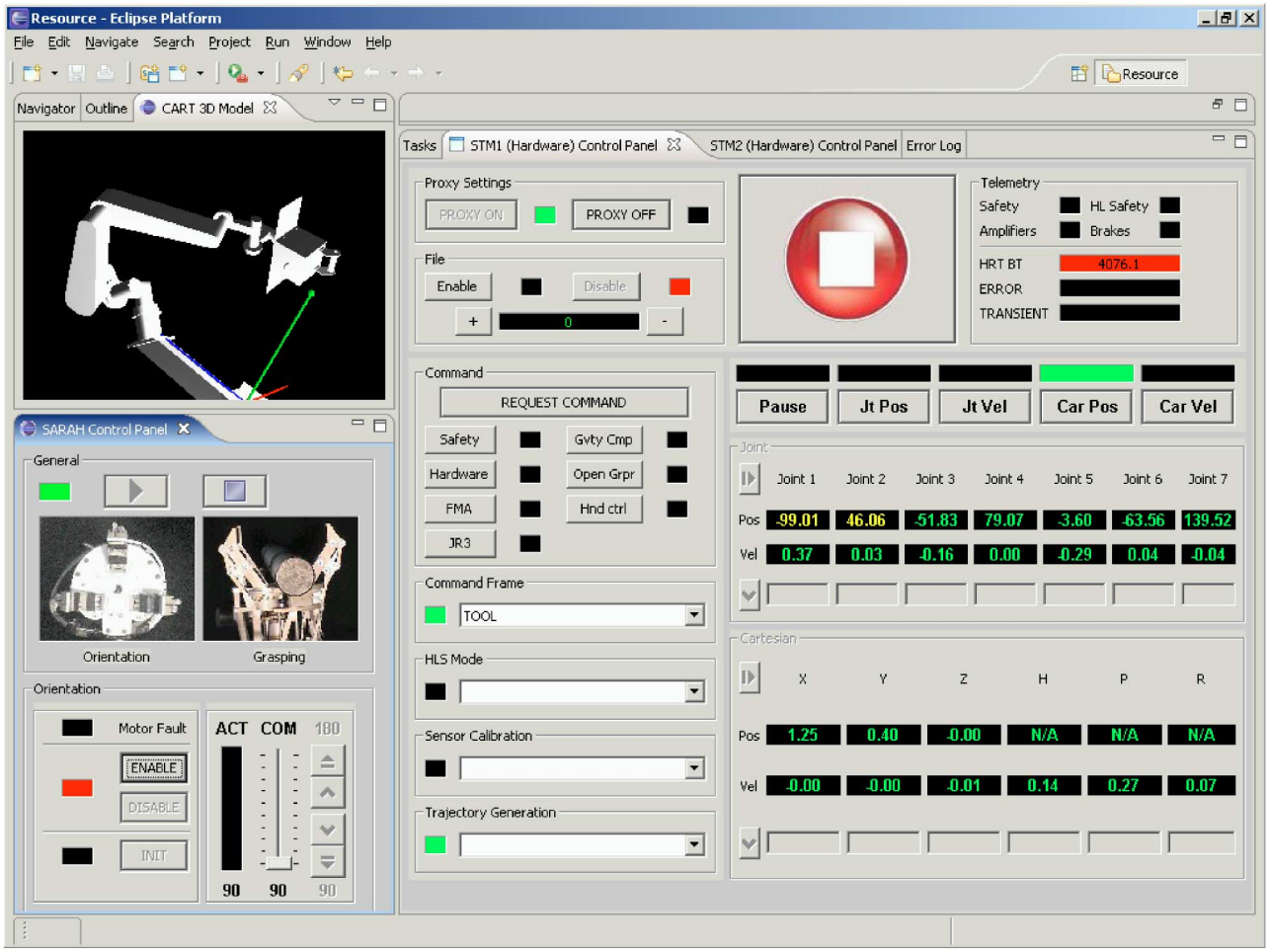


Figure 7. A snapshot of the Ground Control Station.

ments acting on the satellite, they would not have any significant effect over the short period of capture and the angular momentum of the satellite would be conserved. The ensued motion of the target satellite can be characterized as a major rotation about a spin axis with small precession and nutation (Goldstein, 1980). This is similar to the motion described in (Nagamatsu, Kubota & Nakatani, 1996).

In order to make the problem of tracking the moving satellite handle more interesting and to be able to simulate capture for larger satellites, we imposed an elliptic translational motion on the center of the satellite mock-up. Our mock-up is a two-thirds model of a Canadian micro-satellite called QuickSat.

By varying the initial conditions of the Euler equations, we have generated different sets of possible trajectories for the target satellite. These scenarios start from simple slow motions, but gradually grow to faster motions with relatively larger nutation angles and precession rates. A sample trajectory is shown in Figure 8.

Since physical restrictions prevent the robot joints from indefinite rotations, we could only generate a continuous momentum-conserving motion for 120 s. Therefore, to continue the motion, the motion of the satellite mock-up is slowed down toward the end and reversed. By doing so, we can move the satellite as long as we want, without running into the joint limits or the robot singularities.

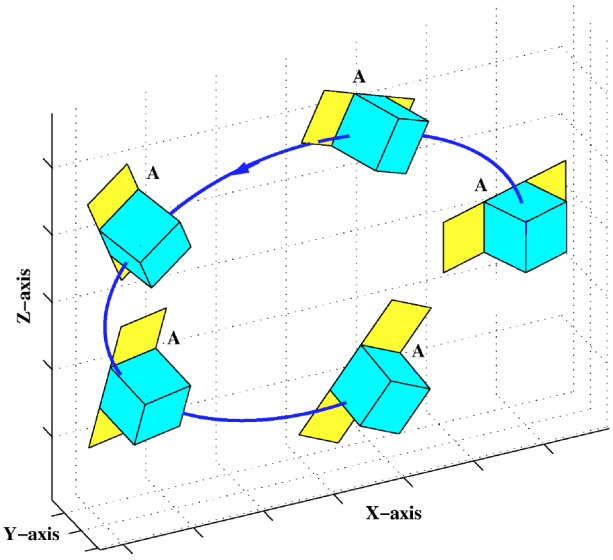


Figure 8. The Cartesian trajectory of the target satellite.

5.2. Chaser Manipulator

This section describes how the trajectory of the chaser manipulator is generated. The Laser Camera System (LCS), shown in Figure 9, and the software CAPE for Collision Avoidance and Pose Estimation, both from Neptec (Ruel, English, Anctil & Church, 2005), are used to generate the pose (position and orientation) of the target satellite with respect to the LCS frame.

The LCS sensor is particularly suited for space application as it is immune to harsh and/or changing lighting conditions (Ruel et al., 2005). The LCS is capable of handling solar inference, and it has been successfully flown on the Space Shuttle Discovery (STS-105) in 2001 and is part of all space shuttle missions since its return to flight, as it is used to inspect its underneath tiles. The range data from the LCS sensor are processed using Neptec's proprietary software CAPE and the pose of the satellite is obtained. The pose estimation method is model based, using a CAD model of the target satellite and a modified version of the Iterative Closest Point algorithm. For more information, see (Ruel et al., 2005).

Figure 9. Laser Camera System from Neptec.

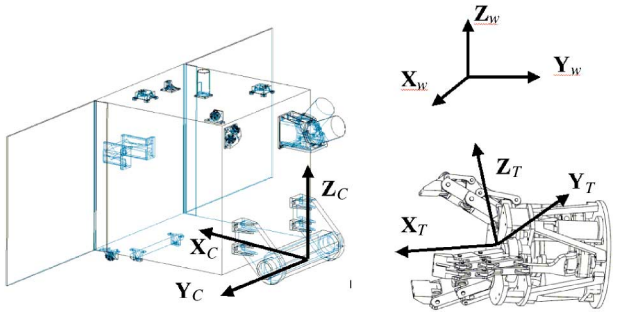


Figure 10. CAD models of Quicksat and SARAH showing reference frames.

The pose is calculated at about 2 Hz with a delay of 0.5 s on the pose of the object at a given instant. The location of the LCS sensor with respect to the manipulator inertial frame was calculated using the kinematic model of the robot arm holding the target satellite. Consequently, the actual pose of the target satellite in the inertial frame could readily be calculated.

An extended Kalman filter is used to filter the LCS raw measurements and to provide a smoothed pose of the target satellite every millisecond. The Kalman filter also included a predictor to estimate the pose of the target satellite 0.5 s forward in time. This predictor is used to compensate for the 0.5 s delay in obtaining the pose using the LCS sensor.

Based on the filtered poses of the capture frame $\mathcal{F}_C$ attached to the capture handle, and the tool frame $\mathcal{F}_T$ attached to the grasping device (see Figure 10), a Cartesian velocity command is generated in the tool frame $\mathcal{F}_T$ of the manipulator to progressively reduce the distance between them. Generating the command in the tool frame provided the oppor-

tunity to independently activate and assign each degree of freedom to a different task. In the experiments, the activation of the tasks is based on the distance between the grasping device and the target satellite. Each of these tasks is described in the following subsections and generates a part of the desired end-effector twist as

$$\mathbf{t}_{desired} = \begin{bmatrix} \dot{\mathbf{r}}_{desired} \\ \boldsymbol{\omega}_{desired} \end{bmatrix} = \begin{bmatrix} v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^T. \quad (1)$$

At the completion of these tasks, with all the corresponding controllers active, the chaser manipulator is positioned very close to the target and the algorithm switches to realize the capture.

5.2.1. Tracking

The first task is tracking the target. If a camera is mounted on the end-effector of the chaser manipulator, the goal of this task is to position the visual features in the center of the camera field of view. The pan-tilt motion of the manipulator uses two rotational degrees of freedom and is generated using

$$\omega_y = -k_{R_y} \tan^{-1}(r_z/r_x), \quad (2)$$

$$\omega_z = -k_{R_z} \tan^{-1}(r_y/r_x), \quad (3)$$

where k_{R_y} and k_{R_z} are control gains for their respective axes, and r_x, r_y, and r_z are the components of $[\mathbf{r}_{T/C}]_T$, namely, the position of the capture frame $\mathcal{F}_C$ with respect to the tool frame $\mathcal{F}_T$. It can be calculated as

$$[\mathbf{r}_{T/C}]_T = \mathbf{R}_T^T([\mathbf{r}_C]_w - [\mathbf{r}_T]_w), \quad (4)$$

where the position of the capture frame $\mathbf{r}_C$ is the output of the Kalman filter discussed above, and $\mathbf{r}_T$ and $\mathbf{R}_T^T$ are calculated from the kinematics model of the chaser manipulator. These quantities are all expressed in the base frame $\mathcal{F}_w$.

5.2.2. Initial Approach

When the task of tracking is initiated, the manipulator end-effector orients itself toward the target. Based on that assumption, the desired distance between the grasping device and the target is controlled by the translational degrees of freedom that move the end-effector forward and backward. The command is computed from

$$\dot{\mathbf{r}} = \mathbf{K}\mathbf{R}_T^T\mathbf{R}_C([\mathbf{r}_T]_C - [\mathbf{r}_{\text{des}}]_C), \quad (5)$$

where $\mathbf{R}_T$ and $\mathbf{R}_C$ are respectively the rotation matrices representing the orientation of the tool frame $\mathcal{F}_T$ and the capture frame $\mathcal{F}_C$ with respect to the base frame $\mathcal{F}_w$; $\mathbf{K}$ is the control gain matrix defined as $\mathbf{K} = \text{diag}(k_x, k_y, k_z)$; $[\mathbf{r}_T]_C$ is the position of the tool frame expressed in the capture frame; and, finally, $[\mathbf{r}_{\text{des}}]_C$ is the desired tool-frame position relative to the capture frame.

5.2.3. Translation Alignment

In order to avoid undesirably large end-effector velocities, the grasping device is aligned perpendicular to the target only when they are close to each other. This task uses the two remaining translational degrees of freedom. Equation (5) shows how the translational velocity command is generated.

In Eq. (5), the gains k_x, k_y, and k_z are not used simultaneously. At first, the approach gain (k_x) is activated; then, when the distance becomes small, the alignment gains (k_y and k_z) are activated. When the three tasks are active, the grasping device is positioned directly in front of the target. Depending on the grasping device used for the capture, a final roll alignment may be required.

5.2.4. Roll Alignment

In our experiment, the SARAH hand is used to grasp a cylindrical handle. The last task is to align the hand so that the handle fits between the fingers of the hand. To that end, the remaining rotational degree of freedom is used:

$$\omega_x = -k_{R_x}\theta_{x_{T/C}} \quad (6)$$

where k_{R_x} is the control gain and $\theta_{x_{T/C}}$ is the orientation of the capture frame $\mathcal{F}_C$ about the x axis of the tool frame $\mathcal{F}_T$.

5.2.5. Capture

With all tasks activated, one can achieve the capture by adjusting the desired relative pose, $[\mathbf{r}_{\text{des}}]_C$, to have the handle in the middle of the hand and then closing the SARAH hand fingers.

5.3. Redundancy Resolution

Both arms of the CART testbed used in our experiments are kinematically redundant serial manipulators with 7 degrees of freedom. If the redundancy is not resolved in an optimal manner, the manipulators will not use their workspace efficiently, i.e., the joints will often reach their limits. In the present section, we present the algorithms implemented for the control of the arms to ensure that they always generate the desired end-effector motion in an optimal manner. This is of particular importance for the creation of a realistic motion emulating a tumbling satellite.

For this work, the kinematic redundancy-resolution scheme that is used is based on the Gradient Projection Method (GPM) described in (Nakamura, 1991). This method relies on the evaluation of a cost function representing a performance-criterion function of the joint positions. The gradient of this cost function projected onto the null space of the main-task Jacobian is used to produce the motion necessary to minimize the specified cost function.

The velocity command is computed using the following equation:

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\dot{\mathbf{x}} + (\mathbf{1} - \mathbf{J}^{\#}\mathbf{J})\left(-k_{SM}\frac{\partial p(\mathbf{q})}{\partial \mathbf{q}}\right), \tag{7}$$

where $\dot{\mathbf{x}}$ is the m-dimensional end-effector task velocity vector, $\dot{\mathbf{q}}$ is the n-dimensional joint velocity vector $\dot{\mathbf{q}}$, $\mathbf{J}$ is the Jacobian matrix, k_{SM} is a control gain that must be tuned, $\mathbf{J}^{\#}$ is the Moore-Penrose inverse of the Jacobian matrix, and $\mathbf{1}$ is the $n \times n$ identity matrix. The first term in Eq. (7) is the minimum-norm solution, and it only affects the end-effector task, while the second term is the gradient projection that controls the self-motion of the manipulator without affecting the end-effector task $\dot{\mathbf{x}}$.

The cost function $p(\mathbf{q})$ defined as

$$p(\mathbf{q}) = m(\mathbf{q})w(\mathbf{q}) \tag{8}$$

is the product of two independent cost functions used to avoid singularities and joint limits, as defined next in Eqs. (9) and (10).

The cost function used to avoid joint limits and singularities is based on the work of (Nelson and Khosla, 1993) and is computed as

$$w(\mathbf{q}) = 1 - \exp\left(-k_{jt}\prod_{i=1}^{n}\frac{(q_i - q_{i\min})(q_{i\max} - q_i)}{(q_{i\max} - q_{i\min})^2}\right), \tag{9}$$

where $q_{i\min}$ and $q_{i\max}$ are the limits for joint i, and k_{jt} is a parameter controlling the shape of the cost function. With this equation, the value of $w(\mathbf{q})$ is close to one when far from joint limits, and it equals to zero at the joint limits.

To measure the distance from a singularity, the most commonly used measure is the manipulability index as defined by (Yoshikawa, 1985) which can be computed as the product of the singular values, σ_i, of the Jacobian $\mathbf{J}$:

$$m(\mathbf{q}) = \prod_{i=1}^{n}\sigma_i. \tag{10}$$

Using this redundancy resolution algorithm, the performance of both CART manipulators was greatly improved. This improvement will be illustrated in Section 6 by presenting some experimental results.

6. EXPERIMENTAL RESULTS

A variety of experiments were performed in our laboratory. Figure 11 demonstrates a sequence of images from one of the autonomous capture scenarios performed. In Figures 11(a)–11(c), the chaser manipulator approaches the target maintaining a constant orientation; the thumb of the hand is at the bottom. When the target is close enough, the chaser manipulator adjusts the hand orientation to match that of the handle on the target. This can be seen in the next three pictures, Figures 11(d)–11(f), where the SARAH hand is aligned with the handle and follows it. Finally, Figure 12(a) shows the capture of the mock-up satellite by the chaser. Figures 12(b) and 12(c) display the views of the satellite from the LCS point-of-view (pov) as well as the attempted match between the LCS data and the CAD model. As can be seen in Figure 12(c), the LCS scan pattern (a rosette) provides very sparse data, which is enough to provide a reli-

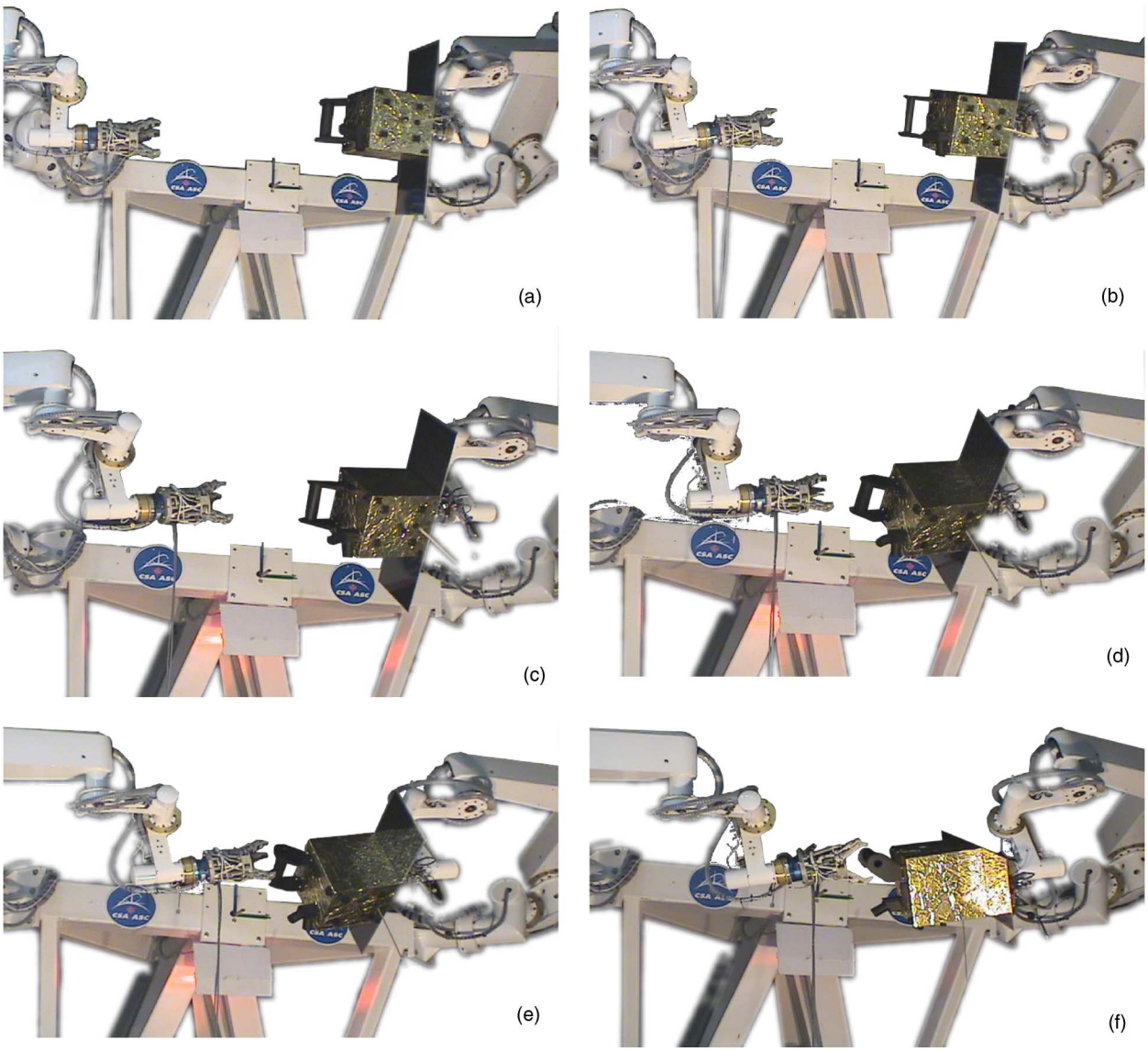


Figure 11. (a)–(e) The chaser satellite approaching the target satellite and tracking its motion. (f) The hand in place to perform capture.

able tracking, even when the view of the target satellite is occluded by the robotic arm performing the capture.

Moreover, as mentioned in Section 4, the experiments were initiated and monitored remotely over the Internet from a variety of locations; some experiments were performed from Munich in Germany and from Denver, CO, in the USA. The "Capture" command was sent from these remote locations; the actual, low-level, commands to the robot were generated locally in Montreal, Canada, by the Cortex Autonomy Engine described in Section 4 using the scenario of Figure 6. The remote operator had access to a Cortex and a GCS window, and also to visual feeds from two cameras. The bandwidth requirements, though not particularly high, were proven to be an obstacle in one instance, especially due to the presence of tight security and firewalls. In all the ex-

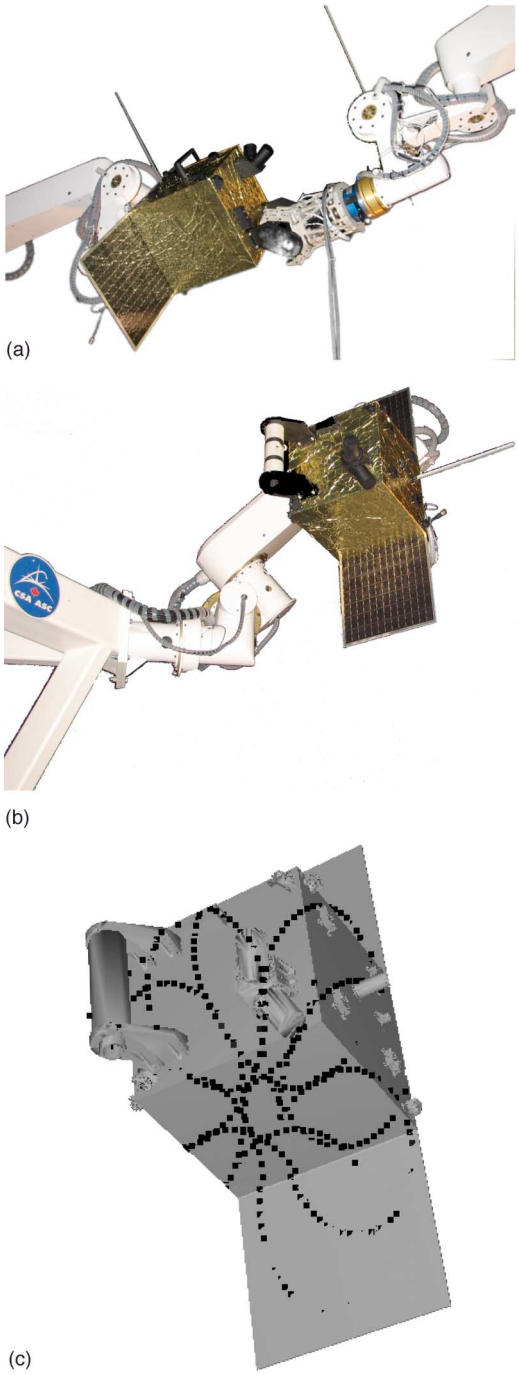


Figure 12. (a) The SARAH hand just before grasping the target satellite; (b) the target satellite (LCS-pov) at a stable position; and (c) the match between the LCS data and CAD-model for satellite shown in (b).

periments, the chaser manipulator firstly approached the target, and then followed the target satellite maintaining a safe constant distance. In the end, the command was issued for the arm to perform the final approach and to grasp the handle mounted on the mock-up satellite using the SARAH hand.

The use of an active vision system, such as the LCS, eliminates the errors due to illumination variations. In particular, the LCS was robust with respect to different lighting conditions. Different trajectories were tested for the target satellite. The combination of the target satellite, the version of the LCS, and the software used imposed some limitations on the speed of the target satellite for robust tracking. In particular, there were some configurations where the tracking failed for higher speeds.

In order to make the experiment more challenging, we sometimes placed an object between the LCS and the target satellite. The LCS reported to the Cortex Autonomy Engine the loss of the target and Cortex stopped the chaser arm and then guided it away to a safe pose. This "artificial" error condition was included both during the tracking phase and during the final capture phase.

Some of the tracking experimental data are plotted in Figure 13, where the trajectory of the x, y, and z coordinates of a point 15 cm in front of the center of the capture handle is shown. The dotted (black) lines show the desired tracking position calculated from the raw data provided by the LCS. The solid (blue) lines are the outputs of the Kalman filter, namely, the desired tracking position. Lastly, the dash-dotted (red) lines show the actual coordinates of the end-effector of the chaser, which was commanded to track the desired tracking position.

The effect of the Gradient Projection Method is illustrated in Figure 14. This figure shows the joint angles, q_1 and q_7, respectively, being the base and the wrist-roll joint angles, computed when the tumbling satellite trajectory described in Section 5.1 is commanded to the end-effector. The solid line represents the joint space trajectory of the robot when the GPM is activated, and the dashed line is the trajectory with GPM deactivated. The resulting joint motion is clearly different depending on whether the GPM solution is added to the minimum-norm solution or not. In the case where the GPM is not used, the minimum-norm solution results in joint angles approaching their limits. In particular, joint angles q_2 and q_6 reach their limit values in approximately 40 s, thus triggering the manipulator's stop.

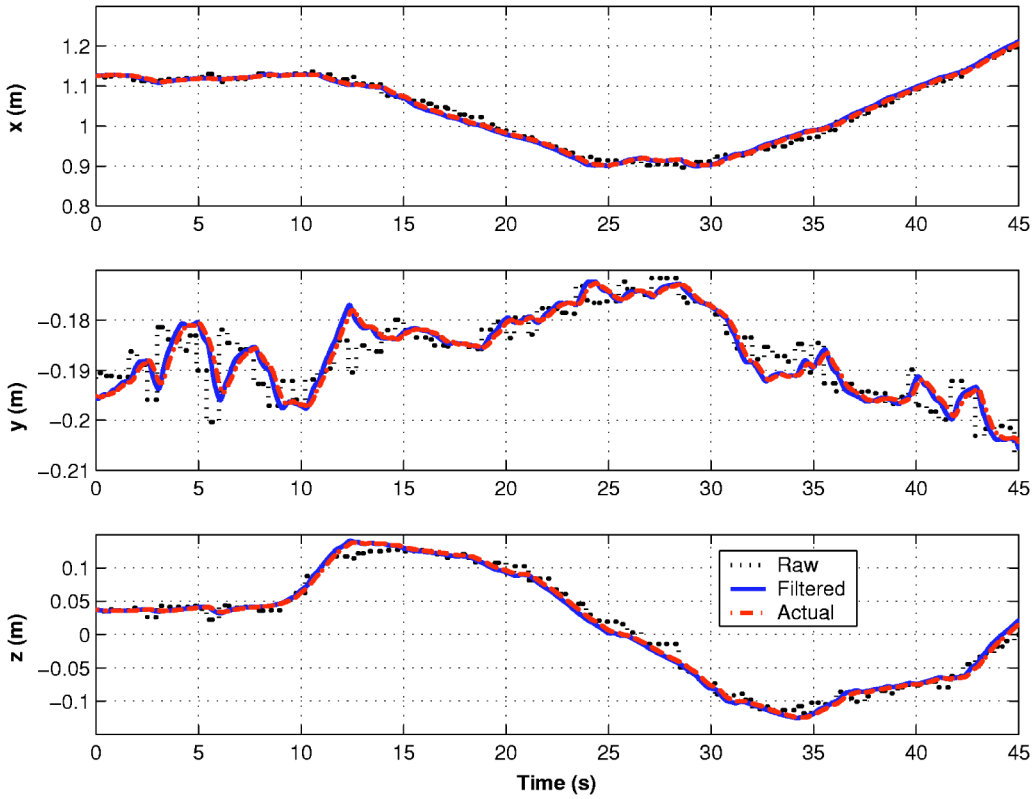


Figure 13. Experimental results; tracking of the capture handle.

7. CONCLUSIONS

In this paper, we describe a framework for the autonomous capture and servicing of satellites. A typical on-orbit servicing scenario was tested in the laboratory to validate some portions of autonomy and vision-based servoing aspects. The task selected for testing was the autonomous capture of a tumbling satellite, which is one of the most challenging tasks to be performed in any OOS mission.

The experiments were run on the CSA's CART test-bed, which has two 7-DOF manipulators. The target tracking was performed using the Laser Camera System developed by Neptec, Inc. The autonomy was implemented using the CSA's Cortex autonomy engine. The results of the experiments showed successful capture of the satellite mockup with minimal operator intervention. Trans-Atlantic tests were conducted over the Internet with limited bandwidth and occasional communication drop-outs.

Future extensions of this work include the development of a secondary vision system to increase the operational robustness, together with ongoing improvements in the visual tracking software. The development of different trajectories of the target satellite, including random motions, would help us study the capture scenario of an erratically moving satellite. Testing the capture scenario for different amounts of noise inserted in the tracking data is also part of our future work.

The successful use of the autonomy engine Cortex in different space robotic scenarios such as planetary exploration and OOS has motivated the development of a second version of Cortex. The new version, currently under testing, will be fully integrated with the Eclipse Java development environment, following the Object-Oriented paradigm. One of the new features is the concept of FSM inheritance, which provides the ability to inherit from an FSM that implements a behavior and just encode some varia-

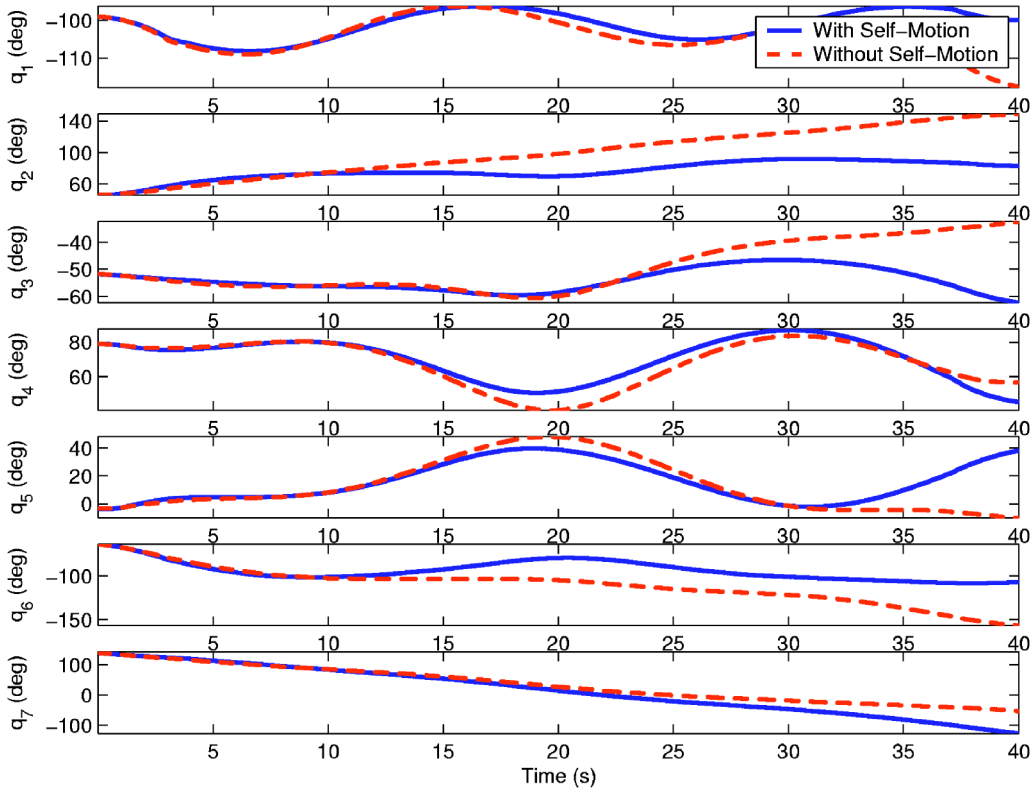


Figure 14. Joint angles during the tumbling satellite trajectory, with and without the GPM.

tion. Furthermore, new features such as parallel FSM's operating concurrently and greater modularity will be also implemented.

During our experiments, we successfully demonstrated the development of key features necessary for an OOS mission. Features such as visual tracking robust to lighting variations, long-distance monitoring and managing of missions, and autonomous capture and release of tumbling satellites will be at the core of any on-orbit robotic mission of the future.

REFERENCES

Alami, R., Chatila, R., Fleury, S., Ghallab, M., & Ingrand, F. (1998). An architecture for autonomy. Inernational Journal of Robotics Research 17(4), 315–337.

Boge, T., & Schreutelkamp, E. (2002). A new commanding and control environment for rendezvous and docking simulations at the EPOS facility. In Proc. 7th Int. Workshop on Simulation for European Space Programmes (SESP), Noordwijk.

Bosse, A.B., Barnds, W.J., Brown, M.A., Creamer, N.G., Feerst, A., Henshaw, C.G., Hope, A.S., Kelm, B.E., Klein, P.A., Pipitone, F., Plourde, B.E., & Whalen, B.P. (2004). SUMO: Spacecraft for the universal modification of orbits. In Proc. SPIE, Spacecraft Platforms and Infrastructure, Vol. 5419, pp. 36–46.

Brunner, B., Landzettel, K., Schreiber, G., Steinmetz, B., & Hirzinger, G. (1999). A universal task-level ground control and programming system for space robot applications—the marco concept and its applications to the ets-vii project. In Proc. Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS 99), pp. 217–224, ESTEC, Noordwijk, The Netherlands.

Didot, F., Oort, M., Kouwen, J., & Verzijden, P. (2001). The ERA system: Control architecture and performance results. In Proc. 6th Int. Symposium on Artificial Intelligence Robotics and Automation in Space (*i*-SAIRAS), Montreal, Canada.

Dupuis, E., & Gillet, R. (2002). Remote operation with supervised autonomy. In 7th ESA Workshop on Avanced Space Technologies in Robotics and Automation (ASTRA 2002), Noordwijk, The Netherlands.

Dupuis, E., Gillett, G.R., Boulanger, P., Edwards, E., & Lip-

#1

sett, M.G. (1999). Interactive, intelligent remote operations: Application to space robotics. In Proceedings of the SPIE Telemanipulator and Telepresence Technologies VI, Boston, MA.

Dupuis, E., Gillett, R., L'Archevêque, R., & Richmond, J. (2001). Ground control of international space station robots. Machine Intelligence and Robotic Control 3(3), 91–98.

Dupuis, E., L'Archevêque, R., Allard, P., Rekleitis, I., & Martin, E. (2006). Intelligent space robotics, chapter A Framework for Autonomous Space Robotics Operations (pp. 217–234) San Antonio, Texas: TSI Press.

Fehse, W. (2003). Automated rendezvous and docking of spacecraft. Cambridge Aerospace Series (No. 16). Cambridge, UK: Cambridge University Press.

Journal of Field Robotics DOI 10.1002/rob

Fontaine, B., Steinicke, L., & Visentin, G. (1996). A reusable and flexible control station for preparing and executing robotics missions in space. In International Symposium on Space Mission Operation & Ground Data Systems (SpaceOps 96), Munich, Germany.

Gennery, D. B. (1992). Visual tracking of known three-dimensional objects. Int. J. Comput. Vision 7(3), 243–270.

Goldstein, H. (1980). Classical mechanics, 2nd ed. Reading, MA: Addison-Wesley.

Grossman, E., & Costa, K. (2003) Small, experimental satellite may offer more than meets the eye. *Inside The Pentagon*.

Hirzinger, G., Brunner, B., Dietrich, J., & Heindl, J. (1993). Sensor-based space robotic-rotex and its telerobotic features. IEEE Transactions on Robotics and Automation 9(5), 649–663.

Hoffman, B.H., Baumgartner, E.T., Huntsberger, T.A., & Schenker, P.S. (1999). Improved state estimation in challenging terrain. Autonomous Robots 6(2), 113–130.

Huntsberger, T., Pirjanian, P., Trebi-Ollennu, A., Nayar, H.D., Aghazarian, H., Ganino, A., Garrett, M., Joshi, S., & Schenker, P. (2003). Campout: a control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration. IEEE Transactions on Systems, Man and Cybernetics, Part A 33(5), 550–559.

Inaba, N., & Oda, M. (2000). Autonomous satellite capture by a space robot: world first on-orbit experiment on Japanese robot satellite ets-vii. In Proceedings. ICRA '00. IEEE International Conference on Robotics and Automation, Vol. 2, pp. 1169–1174, San Francisco, CA.

Kaplan, M.H. (1976). Modern spacecraft dynamics and control. New York: Wiley.

Kasai, T., Oda, M., & Suzuki, T. (1999). Results of the ETS-7 mission, rendezvous docking and space robotics experiment. In Proc. 5th Int. Symposium on Artificial Intelligence, Robotics and Automation In Space (*i*-SAIRAS), Noordwijk, The Netherlands.

Kawasaki, O., Imada, T., Yamanaka, K., & Tanaka, T. (2000). On-orbit demonstration and operations plan of the H-II transfer vehicle (HTV). In The 51th Int. Astronautical Congress, Rio de Janeiro, Brazil.

King, D. (2004). Saving Hubble. In The 55th Int. Astronautical Congress, Vancouver, Canada.

Laliberté, T., Birglen, L., & Gosselin, C. (2002). Underactuation in robotic grasping hands. Japanese Journal of Machine Intelligence and Robotic Control, Special Issue on Underactuated Robots 4(3), 77–87.

Lewis, J. (2004). Space weapons in the 2005 US defense budget request. In Workshop on Outer Space and Security, Geneva, Switzerland.

Liang, B., Li, C., Xue, L. & Qiang, W. (2006). A Chinese small intelligent space robotic system for on-orbit servicing. In Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, pp. 4602–4607, Beijing, China.

Martin, E., Dupuis, E., Piedboeuf, J.-C., & Doyon, M. (2005). The TECSAS mission from a Canadian perspective. In Proc. 8th International Symposium on Artificial Intelligence and Robotics and Automation in Space (*i*-SAIRAS), Munich, Germany.

Maxwell, S., Cooper, B., Hartman, F., Wright, J., & Yen, J. (2004). The design and architecture of the rover sequencing and visualization program (rsvp). In Proceedings of the 8th International Conference on Space Operations, Montreal, Canada.

Nagamatsu, H., Kubota, T., & Nakatani, I. (1996). Capture strategy for retrieval of a tumbling satellite by a space robotic manipulator. In Proc. IEEE Int. Conf. Robotics and Automation (ICRA), pp. 70–75, Minneapolis, MN.

Nakamura, Y. (1991). Advanced robotics: redundancy and optimization. Reading, MA: Addison-Wesley.

Nelson, B., & Khosla, P. (1993). Increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance. In Proc. IEEE Int. Conf. Robotics and Automation, Vol. 3, pp. 418–423.

Nesnas, I., Wright, A., Bajracharya, M., Simmons, R., & Estlin, T. (2003). Claraty and challenges of developing interoperable robotic software. In International Conference on Intellligent Robots and Systems (IROS), pp. 2428–2435, Las Vegas, Nevada.

Potter, S. (2002). Orbital express: Leadilng the way to a new space architecture. In Proc. Space Core Tech Conference, Colorado Spring, CO.

Ruel, S., English, C., Anctil, M., & Church, P. (2005). ^{3D}LASSO: Real-time pose estimation from 3D data for autonomous satellite servicing. In Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space –*i*SAIRAS, Munich, Germany.

Rumford, T.E. (2003). Demonstration of autonomous rendezvous technology (DART) project summary. Proc. SPIE, 5088, 10–19.

Sato, N., & Doi, S. (2000). JEM remote manipulator system (JEMRMS) human-in-the-loop test. In Proc. 22nd Int. Symp. Space Technology and Science, Morioka, Japan.

Sommer, B. (2004) Automation and robotics in the german space program Unmanned on-orbit servicing (OOS) & the TECSAS mission. In Proc. the 55th Int. Astronautical Congress, Vancouver, Canada.

Stieber, M., Sachdev, S., & Lymer, J. (2000). Robotics architecture of the mobile servicing system for the interna-

tional space station. In Proc. 31st Int. Symp. on Robotics (ISR), Montreal, Quebec.

Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R. & Das, H. (2001). The claraty architecture for robotic autonomy. In Proceedings of the 2001 IEEE Aerospace Conference, Big Sky, MT.

Whelan, D., Adler, E., Wilson, S., & Roesler, G. (2000). DARPA Orbital Express program: effecting a revolution in space-based systems. In Proc. SPIE, Small Payloads in Space, Vol. 4136, pp. 48–56.

Wilcox, B., Tso, K., Litwin, T., Hayati, S., & Bon, B. (1989). Autonomous sensor based dual-arm satellite grappling. In NASA-Conference on Space Telerobotics, Pasadena, CA.

Wingo, D.R. (2004). Orbital Recovery's responsive commercial space tug for life extension missions. In Proc. the AIAA 2nd Responsive Space Conference, Los Angeles, CA.

Yoshida, K. (2003). Engineering test satellite vii flight experiments for space robot dynamics and control: Theories on laboratory test beds ten years ago, now in orbit. The International Journal of Robotics Research 22(5), 321–335.

Yoshida, K. (2004). Contact dynamics and control strategy based on impedance matching for robotic capture of a non-cooperative satellite. In Proc. 15th CISM-IFT0MM Symp. on Robot Design, Dynamics and Control - Romansy, St-Hubert, Canada.

Yoshida, K., Nakanishi, H., Ueno, H., Inaba, N., Nishimaki, T., & Oda, M. (2004). Dynamics, control, and impedance matehing for robotic capture of a non-cooperative satellite. RSJ Advanced Robotics 18(2), 175–198.

Yoshikawa, T. (1985). Manipulability of robotic mechanisms. Int. J. of Robotic Research 4(5), 3–9.