In Proceedings of the 8th ESA Workshop on Advanced Space Technologies for Robotics and Automation
'ASTRA 2004' ESTEC, Noordwijk, The Netherlands, November 2 - 4, 2004

NEW GENERATION ROBOTIC CONTROLLER SOFTWARE FOR SPACE M ANIPULATORS

N. Cikic [2], E. De Marchi [2], M. Filippini [2], F. Fusco [3], T. Grasso [2], A.. Olivieri [1], A. Rusconi [3]

[1] ASI
Contrada Terlecchia, 75100, Matera, ITALY
Email: angelo.olivieri@asi.it

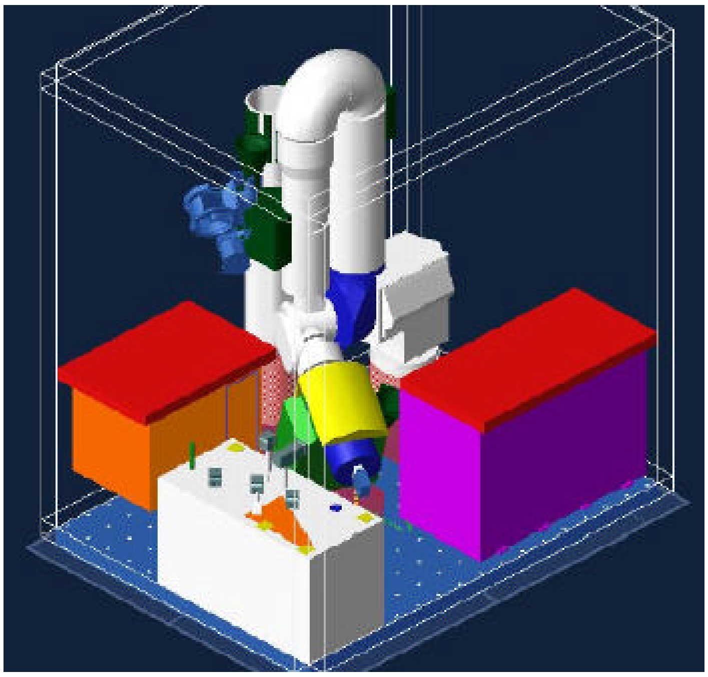
[2] Tecnomare SpA
S. Marco 3584, 30124 Venezia, ITALY
Email: tiberio.grasso@tecnomare.it

[3] Galileo Avionica
via Montefeltro 8, 20156 Milano, ITALY
Email: andrea.rusconi@galileoavionica.it

Introduction

EUROPA (External Use of Robotics for Payload Automation) is a robotic experiment to be flown on the International Space Station (ISS) and is intended to perform a realistic end-to-end robotic technology demonstration in exposed environment. It is mainly composed by:

- a taskboard, where significant robotic experiments are housed
- a robotic arm, based on the existing ASI 7 dof SPIDER arm, used to perform the robotic experiments
- a robot controller, to control arm motion and to perform safety checks
- data handling and power unit (DHPU) , taking care of interfacing with ISS and providing power to all the equipments
- hold down mechanism, to latch arm during launch and re -entry phases



The assigned volume for all the equipment is a workcell whose dimensions are about 1m x 1m x 1m (see figure above). This means that, considering arm dimensions, the arm shall be provided with specific capabilities for an effective motion without volume protrusions while performing operations in both free and contact modes.
The EUROPA robot controller provides the necessary computing power to include sophisticated control algorithms and reach a high degree of autonomy for the on-board operations. It is composed by a VME crate where the following boards are housed:

- Main CPU board, based on MPC750 processor where control software runs
- Four Driver boards, providing power to the arm motors and motor shaft resolvers acquisition
- two microcontroller boards, based on 8086 processor, used for both output shaft resolvers acquisition and independent safety checks, as from Computer Based Control System (CBCS) approach ([6])
- one I/O board, used for digital I/O and Force/Torque Sensor acquisition. Digital I/O are mainly connected with the hold down presence sensors
- one Power supply board

The Controller is at an advanced implementation stage and has passed the Safety Review 0/1 with NASA.
The proposed paper will focus on the innovative features of the SW Robotic Controller. The part related to Robot Control is structured according a NASREM -like architecture, and implements the following main items:

- Execution of Robotic Programs written in Europa Programming Language (EPL)
- Resolution of arm redundancy based on arm configuration angle, seen as the 7th cartesian degree of freedom
- Position/Force control at the end effector, allowing precise interaction with the environment
- Possibility to move the arm in teleoperation mode, in joint or cartesian space
- Possibility to customize software behaviour, by means of some configuration and database files. They allow operator to modify control gains, check thresholds, command parameter limits, timing and so on.

The part of the software not related to Robot Control is mainly devoted to communication with ISS through DHPU, and to safety checks, according to the CBCS approach.

Arm redundancy resolution

EUROPA arm has seven degrees of freedom and is kinematically redundant, since six joints are sufficient for generic end-effector position and orientation. In recent years many techniques to solve the problem have been proposed; for the most part, they focus on local optimization to achieve EE (end effector) trajectory while optimizing additional criteria like obstacle avoidance, singularity avoidance, motor torque minimization, joint limits avoidance. See, for example, [1], [2], [3].
The use, for example, of the pseudoinverse (or other local minimization) control results, in general, in the loss of cyclicity of manipulator configuration as the end effector cycles through a periodic motion. This means that as the EE is constrained to move along a well specified path, the arm may experience unpredictable self-motion wanderings, and this is unacceptable in EUROPA application, where all the tasks that the manipulator has to accomplish will be planned and tested in the ground reference model and then executed in orbit, expecting about the same behaviour tested on ground.
The approach used to solve the problem is presented in the following figure 1, that shows the EUROPA arm self motion; note that the elbow point E make exactly a circle around the line connecting the shoulder point S and the wrist point W that are fixed. Analyzing the arm self motion, a "natural" way to solve redundancy is to add, as additional task the arm has to accomplish, the control of the elbow position E along the circle traveled during self motion.
This is obtained introducing a new kinematic equation to directly control the arm angle (see [4]), defined as the angle from a reference plane containing a reference unit vector (with origin in S) and the shoulder-wrist line SW to the shoulder-elbow-wrist plane SEW in the right hand sense about the vector w = W-S. Selecting as reference vector the joint 1 axis, the arm angle ψ is defined by the following equation

$$\tan y = \left[\hat{w}^T \cdot \left(\hat{V} \times p\right), \hat{V}^T \cdot p\right] \qquad (1)$$

where $\hat{V}$ is the reference unit vector, $\hat{w} = \frac{w}{\|w\|}$, $\hat{p} = \frac{p}{\|p\|}$, $p = \left(I_3 - \hat{w} \cdot \hat{w}^T\right) \cdot e$ and e = E-S.

The arm angle jacobian can be also analytically computed, and is given by:

$$J^y = \frac{(\hat{w} \times \hat{p})^T}{\|p\|}\mathbf{E} + \left\{\frac{\hat{V}^T w}{\|l\|}\left(\hat{w} \times \hat{l}\right)^T - \frac{\hat{w}^T e}{\|w\|\|p\|}(\hat{w} \times \hat{p})^T\right\}\mathbf{W}, \qquad (2)$$

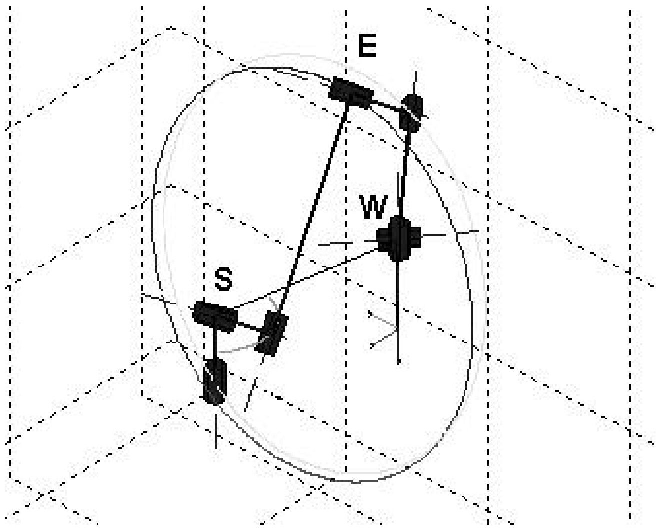


Fig 1: EUROPA arm self-motion.

where $\hat{l} = \frac{l}{\|l\|}$, $l = (w \times \hat{V}) \times w$, and **E** and **W** are the Jacobians that relate joint rates to elbow E and wrist W linear velocities.

The use of the arm angle as extra kinematic equation induces a control algorithm that have the property of cyclicity; furthermore it allows a precise elbow point control (the widely moving point during self motion), that has turned out to be a fundamental feature to avoid workcell protrusions during arm operations.

EUROPA robotic controller SW

Robotic controller software has been developed adopting a hierarchical architecture inspired to NASREM. Four hierarchic levels have been implemented: Task, EMove, Primitive and Servo; their main tasks are described in the following.

- *Task Level Controller*
  This is the highest level controller, and its main function is to implement a virtual machine able to execute byte code produced by EPL language compiler and linker. When a call to a function that requires arm or gripper motion is processed, the elementary moves necessaries to execute the motion are generated and passed (one by one) to the underlying EMove level controller. Synchronization with EMove level is achieved through feedback that are received each time an elementary move has been executed.
  This level is not synchronized: the next byte code will be executed, if present, once the current is finished.

- *EMove Level Controller*
  The main function of EMove level controller is to process elementary moves coming from Task level. When an elementary move is received, a set of motion primitives is generated for both arm and gripper subsystem and sent (all together) to the underlying Primitive level controller for execution. Primitives are mainly composed of five order polynomials describing the desired behaviour of the arm and gripper. Usually three primitives are generated for each subsystem: one for the acceleration phase, one for constant speed phase and one for deceleration phase. Primitives are generated in such a way to assure a coordinated motion with continuity of trajectories, and of their first and second derivatives. Synchronization with Primitive level is obtained through feedbacks: the execution of an elementary move is finished when all generated primitives have been executed from Primitive level controller.

Like Task level, also EMove level is not synchronized: the next elementary move will be executed, if present, when the current is finished.

- *Primitive Level Controller*
  Primitive level controller is mainly composed of two modules: one dedicated to execution of primitives coming from EMove level, and one dedicate to execution of teleoperation inputs. They are mutually exclusive: when a robotic program is running (default mode), no teleoperation inputs are accepted, and vice versa, when teleoperation mode is enabled, no robotic programs can be executed.

  Teleoperation mode:
  Arm can be teleoperated either in joint mode or cartesian mode. In joint mode joint speeds are accepted as teleoperation input; in cartesian mode the operator specify the tool linear and angular speeds, the arm angle speed, as well as the tool frame and the reference frame. Gripper can be teleoperated only specifying its speed.
  Depending on teleoperation type, suitable servo setpoint are generated to assure the desired arm behaviour and sent to Servo level for execution.

  Robotic Mode:
  In default mode, Primitive level executes primitives coming from EMove level sampling trajectory specification polynomials at a fixed rate. Samples are used to compose servo setpoint that are sent to Servo level for execution. When the execution of a primitive finishes, the execution of the next one, if any, starts; if no more primitives are in queue, t he last servo setpoint is repeated.

  In both teleoperation and default mode Primitive level controller handles controlled stop requests stopping the arm and the gripper in a controlled manner; this is done generating suitable primitives that applies a smooth deceleration profile while remaining on the same path the arm is treading.
  Synchronization with servo level is assured by feedback received each time a servo has been executed.
  As Servo level sends feedback at a precise rate by the use of a real time clock, Primitive level controller results synchronized.

- *Servo Level Controller*
  The main servo level controller function is to generate, at a fixed rate, the current setpoints for the motor driver boards controlling the eight motors of the arm and gripper. Inputs for the control loop are obtained through linear interpolation of servo setpoints coming from Primitive level controller.
  Gripper can be driven in two modes: position mode and current mode. In current mode the current to be provided to the gripper motor is directly specified by the setpoint; in position mode a PID controller is used to compute current from position setpoint.
  Arm can be also driven in two modes: joint mode and cartesian mode.
  In joint mode, arm motor currents are computed through seven PID controllers each one controlling one joint; in cartesian mode a cartesian hybrid position/force control loop, depicted in Fig. 2 is used. This control scheme has been derived from [5].
  As it can be seen in Fig. 2 the inputs of the control loop are the desired force F_d, the desired position P_d and the desired arm angle aa_d. The task specification matrices Ω and $\tilde{\Omega}$, are built up starting from user specification of force and position controlled direction in a suitable task frame. Position is, of course, not controlled along force controlled axis, as well as force is not controlled along position controlled axis. $L_0(q)$ is the inertia matrix in the operational space and allows, with Coriolis, centrifugal and gravity term μ, the dynamic decoupling, in such a way that the tool becomes equivalent to a single unit mass. The transpose jacobian $J_0^T(q)$, built up from base jacobian, and arm angle jacobian, is used to convert generalized forces from operational space to joint space. Note that through L_0 large variations of inertia to be controlled at the end effector, e.g. due to grabbing of large payloads, can be compensated, improving the dynamic performances of the system.
  Finally, no explicit inverse kinematics is foreseen in the loop.

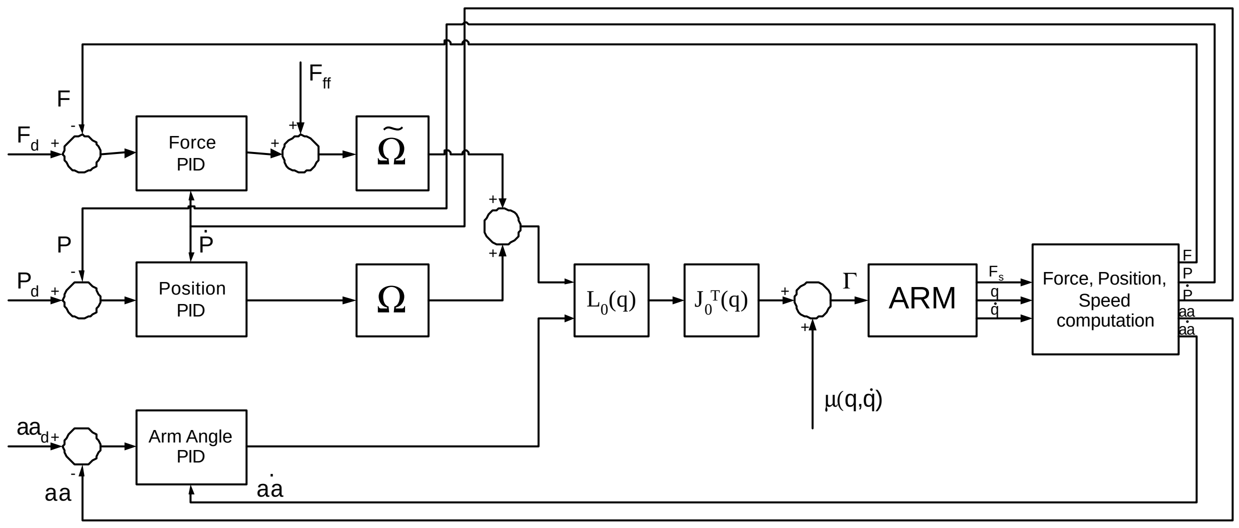


Fig. 2: Arm cartesian hybrid position/force control scheme.

Servo level controller also handles emergency stop requests stopping immediately the arm and the gripper disabling their motor drivers and engaging brakes.
Servo level is strictly synchronized by a real time clock running at a frequency of 1000 Hz.

EUROPA programming language (EPL)

Robotic programs are, in EUROPA project, the main way to move the arm and the gripper (the other one is the teleoperation mode). They are written using EPL language, a versatile novel robotic language that allows operator to perform arm motion, controller status management and flexible data processing. EPL is a programming language, directly derived from PDL2, already used in previous projects, with specific features for programming EUROPA robotic applications.
A program is divided in three sections:

- prototype section, where the program prototype is specified defining its name, input arguments and returned value
- declaration section, where variables, constants and functions used in the program shall be declared
- statement section, where the body of the program is written.

Basic data types are REAL to represent real values, INTEGER for integer values, BOOLEAN for boolean data and CHAR for characters. Furthermore ARRAY (up to two dimensions) of the basic data types can be defined.
All common logical (AND, OR, NOT, XOR), arithmetic (+, -, *, /, %), relational (<, <=, =, >, >=, <>), assignment (:=) and precedence (()) operators are available.
Program flow can be controlled using the IF THEN ELSE ENDIF statement to choose between two possible flows of program depending on a boolean expression, or the WHILE DO ENDWHILE statement to execute a sequence of statements as long as a boolean expression is true.
Previously declared functions can be called (also recursively) passing them parameters either by value or by reference.

EPL language is provided with a set of built-in functions that make it suitable for writing robotic application. They are divided in several groups:

- Mathematical, that allow the operator to execute operations such as ABS, SIN, COS, SQRT, LN, ROUND, POW, etc.
- String Manipulation, that allow the operator to deal with string; examples are: ENCODE_REAL, DECODE_INTEGER, STRCAT, STRCPY, STRCMP , etc.
- Motion Control, that enable the operator to move the arm and the gripper in several ways calling function like MOVE_RELATIVE, MOVE_JOINTS, CHANGE_GRIPPER_WIDTH, etc. For example, to move the arm in contact,

the operator can use the MOVE_IN_CONTACT built-in function specifying the tool frame (wrt a reference frame), the cartesian speed, the desired forces, the directions to control in force and in position, the relevant tool, reference frames for the contact operations, and a termination condition for the force action (time, a linear/angular displacement, a force/torque threshold or an external event).

- Data Acquisition , to allow the operator to acquire (up to 1000 Hz frequency) and store a selectable set of robotic data (like joints position, speeds, cartesian data, etc.), or data coming from external devices such as distance sensors or vision systems.
- Controller status, to retrieve the current status of the arm and use it in the program; examples are GET_ARM_JOINT_POSITION, GET_ARM_JOINT_SPEED, GET_GRIPPER_CURRENT, GET_FORCE, etc.
- Database Management built-in functions, to retrieve and update data from internal databases; examples are GET_STANDBY_CONFIG, SET_SAFETY_CONTACT_STATE, etc.
- I/O Management: to write/read to/from digital I/O channels: GET_D_IO, SET_D_IO
- Other function like printing functions, program control functions (DELAY, BREAK).

EPL programs are compiled and linked to produce a byte-code that can be sent to task level controller virtual machine for interpretation. This can be done using the EPLc and EPLlink programs: a compiler and a linker expressly developed for the EPL language.

Tests

The EUROPA arm controller software architecture and coding have been developed using Rational Rose RealTime CASE tool and WindRiver Tornado II running on WIN32 host platform. The target board is an AITECH(TM) S210 board provided with a PPC750 @ 233MHz processor, 64Mb RAM, 64Mb File flash, 8 Mb User flash, 2 Mb boot flash, 2 RS422 serial lines, 1 ethernet controller. Target operating system is VxWorks(TM) 5.4. Some preliminary tests to verify target board computational load have been carried out with the aid of a dynamic simulator of the arm. This simulator runs on a win32 system, and is connected with the target via ethernet TCP/IP protocol. Table 1 shows the results of a test performed while a robotic program executing arm cartesian motion is running.
Due to heavy computational load of dynamic simulator of the arm the test has been carried out in simulated time (1 simulated s $\cong$ 2 s); actual load column represent the real target CPU load detected through vxWorks spy functions, while the third and forth columns show the expected CPU load when the servo loop is triggered respectively with a 500 Hz and 1000 Hz real time clock.

Table 1: CPU expected loads

Modules	**Actual Load**	**Expected Load @500Hz**	**Expected Load @1000Hz**
Robotic control modules	8.8%	19.4%	38.8%
Safety modules	2.3%	2.9%	4.0%
Communications modules	0.6%	0.6%	0.6%
Other modules	1.1%	1.1%	1.1%
Tot	12.9%	24.0%	44.5%

As it can be seen, the expected medium CPU load is lower than 50% with servo loop running at 1000 Hz. Of course great part of the CPU power is used from robotic control modules to close cartesian servo loop.

Configuration databases

EUROPA arm controller is fully configurable through a set of ASCII databases that can be updated from the operator. They are mainly divided in three categories: robotic controller databases, safety databases, environment databases.

- Robotic Controller Databases: they contains all the parameters related to robotic system controller; examples of contained parameters are: Denhavit-Hartenberg kinematic parameters of the arm, arm joint and gripper offsets, arm joint and gripper limits, resolver lookup tables, transformation matrices (world-base, ee-gripper, etc.), PIDs gains, dynamic parameters of the arm and motors (masses, inertias, etc.), servo and primitive sample periods, etc.
- Safety Databases: they contains all the parameters related to safety module; example of such parameters are: speed thresholds, current thresholds, resolvers cross check thresholds, distance thresholds between arm and environment, encumbrance of the arm links (in term of cylinders) , encumbrance of the environment (in terms of cylinders, planes,

quadrilaterals and spheres), collision pairs (pairs of object which distance is computed to check if a collision is impending), etc.

- Environment databases: they contains the description of the environment the arm has to interact with, in terms of objects position, actions that can be executed on each object, and motion parameters that shall be used to execute each allowed action. Such motion parameters can be retrieved from robotic programs using suitable built-in functions.

Safety module

In the EUROPA mission, the Computer Based Control System (CBCS, as from [6]) has been adopted to control critical hazards such as envelope protrusion, collision with internal objects, hold down wrong ascertainment. In order to control these hazards,. The following types of checks have been implemented in three independent CPUs, allowing for independent paths of intervention:

- Workspace: check that arm parts do not protrude the assigned envelope
- Impact energy: check that the speed and current of each joint do not overcross given thresholds
- Sensor reliability: check that the resolver readings for each joint (eg motor, coarse output shaft and fine output shaft resolvers) are coherent
- CPU cross monitoring: check that the other boards are correctly running
- Threshold cross check: check that the safety thresholds, stored in 3 different locations, are not corrupted by SEU - SBHU

For every communication with the safety module (i.e. customizations of check thresholds, reading of hold down status), a suitable Safe Protocol has been developed. This protocol (between EUROPA Control Station –the only initiator- and destination computer) allows safe and reliable communications against possible software or timing errors that might cause inadvertent safety module modification, leading to hazards. Safe Protocol is based on an encoding function and a safety key, owned only by EUROPA Control Station and the destination computer. Each time a message is sent by EUROPA Control Station, suitable data encoding is done and the destination is able to understand if the initiator is Ground itself or an intermediate computer (due to errors).

Conclusion

The innovative feature of the presented SW controller for space robotic arms derives from its being specifically developed for telerobotic applications in space. As such, it addresses all the specific issues deriving from this scenario:

- programmability from remote, with different levels of interaction with the human operator, including teleoperation
- full re-configurability, including updating of kinematic parameters after arm recalibration
- provision for a general set of manipulation motion/force primitives, with explicit force control
- management of arm redundancy
- management of safety according to NASA requirements.

The controller is going to be integrated with the Ground Reference Model of the Europa robotic arm.

References

[1] A. Liégeois, “Automatic supervisory control of the configuration and behaviour of multibody mechanisms”, in IEEE Trans. Syst., Man., Cyber., vol. SMC-7, no. 12, pp. 868-871, 1977

[2] J. Baillieul, “Kinematic programming alternatives for redundant manipulators”, in Proc. IEEE Int. Conf. Robotics Automat., St. Louis, Mar. 1985, pp. 722-728

[3] Y. Nakamura and H. Hanafusa, “Task priority based redundancy control of robot manipulators”, in Proc. 2nd Int. Symp. on Robotics Research, Kyoto, Japan, Aug. 1984

[4] H. Seraji, “Configuration Control of Redundant Manipulators: Theory and Implementation”, in IEEE Trans. on Robotics and Automat., vol. 5, no. 4, August 1989, pp. 472-490

[5] O. Khatib, “A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation”, in IEEE Journal of Robotics and Automation, Vol. 3, no. 1, February 1987, pp. 43-53

[6] NASA JSC letter MA2-97-083, “Fail Safe Approach for Computer-Based Control System”, September 19, 1998