# EUROPEAN ROBOTIC ARM: THE PROBLEM OF PREVENTING COLLISIONS

F. Fusco, R. Gallerini

Tecnospazio S.p.A., Milano, Italy

ABSTRACT

The European Robotic Arm (ERA), to be used for assembly and servicing activities on the Russian Segment of the International Space Station (ISS), is an 11 meter long arm, moving both automatically and under cosmonaut control. A major safety issue is to avoid possibility of collisions. Although the most obvious approach is the use of sensors, no sensor-based solution is practical. A different solution is adopted, based on a software simulating the ERA motion, using a geometrical and kinematic model of the arm and its payload, within a geometrical model of the ISS Russian Segment. The simulation is aimed to predict collisions in order to stop the arm before it can actually cause damages to the station and to itself. This software has been delivered by Tecnospazio to Fokker Space as part of the ERA onboard software, and it is currently subject to the ERA system test campaign.

Keywords: Robotic Arm, Collision Detection, Software Simulation, 3D Modeling.

## 1. INTRODUCTION

The European Robotic Arm, a project funded by the European Space Agency, is an anthropomorphic, relocatable, nearly symmetric arm. Its purpose is to be used for assembly and servicing activities on the Russian Segment of the International Space Station. It is a seven-joint arm operated with one joint fixed, leading to six degrees of freedom. It can be controlled both automatically, through pre-programmed procedures (Mission Plans) or manually, under direct control of the cosmonauts, from inside or outside the ISS. Different modes of operation are implemented, to cover the foreseen needs: free motion in joints or Cartesian kinematics, proximity motion assisted by video camera image processing, and compliant motion assisted by torque/force sensors.

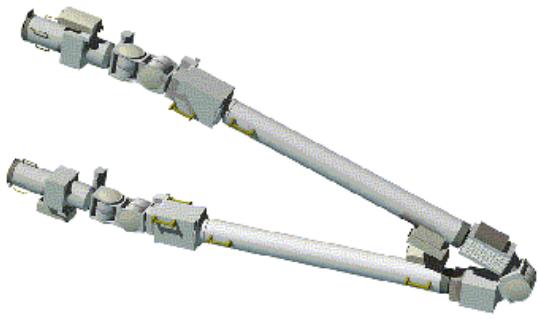

Figure 1. ERA layout

Given the wide working area of the arm, which is about 11 m long, with a maximum payload capacity of 3000 kg, the potential danger of collisions with the ISS or with the arm itself has been carefully considered. This paper describes the work done to provide ERA with the capability of preventing collisions. The requirements were specified by Fokker Space, while the development (requirement analysis, design, implementation and test up to validation) was performed by Tecnospazio; the Collision Detection was delivered to Fokker Space in May 1999, where it was integrated in the ERA Control Computer, which is currently in the system testing phase.

## 2. WHY A SOFTWARE FOR COLLISION DETECTION

In the normal situation, ERA operates under Mission Plans, moving along paths defined during the mission preparation, and checked on ground to be collision free. When these paths are executed in flight, an online Path Deviation Check makes sure that the actual path of the arm does not differ, within a given tolerance, from the safe pre-planned path. If any deviation is detected, an emergency procedure causes the arm to be stopped immediately.

On the other hand, during manual control, when the operator moves the arm along unforeseen paths not verified on ground, a Collision Detection system is necessary in addition to the visual control of the cosmonaut.

To increase safety, it has been decided to have the Collision Detection active also during Mission Plans execution, in addition to the Path Deviation Check.

In preliminary studies of the problem, several approaches were analyzed. The most obvious collision detectors are sensor-based, like vision or distance/proximity sensors. These solutions can be grouped in two categories: local and global sensors. Each of them, if applied to ERA, have serious drawbacks.

Local sensors are devices mounted on the robot surface, to measure the distance (or return a proximity threshold flag) of the closest obstacle within a certain cone. Examples are optical devices (light emitter and photodiode), magnetic devices (detecting the proximity of metallic bodies) or mechanical contact-sensitive devices. The local approach is effective for compact robots operating in simple environments, but given the size of ERA and the presence of obstacles in all space directions, such a solution would require a very high number of sensors and a complicated cabling of the arm. Moreover, when ERA carries a payload, the payload itself needs to be checked for collisions, so it should mount further sensors, and they have to be electrically interfaced with ERA. It is clear that this approach leads to unacceptably complex setup of the arm and its payloads.

Global sensors are independent devices, placed on static structures, like panoramic cameras mounted on the ISS. Used in combination, to cover the entire robot workspace, they could, in principle, detect collisions in the same way human observers do, i.e. through image processing and pattern recognition. Apart the obvious technical difficulty to implement such a sophisticated system, in ERA it would face

two additional problems: the difficult lighting conditions of the orbital environment and the relocatability of the arm. The ability to relocate at different positions on the ISS implies that a system of panoramic cameras aimed to collision monitoring would have to ensure coverage of the whole Russian Segment, which is, again, impractical.
Discarded the sensor-based solutions, a completely different approach was investigated, and finally adopted: it consists of including in the control software a simulation of the robot motion within a 3D geometrical model of the environment. When ERA starts moving, its kinematics is mirrored in a virtual ISS model, and a geometrical check is activated, so that, using suitable speed-dependent margins, it can detect danger of collision in the real world and react by warning the user or immediately stopping the arm.
Several types of collisions have to be taken into account: ERA, including its payload, with itself (called link-link and payload-link) and with the ISS (link-obstacle and payload-obstacle).
Since 3D simulations are resource-consuming, the main challenge for the Collision Detection software is to meet the constraints given by the limited resources (memory and processing power) available on the ERA Control Computer.
Particular attention has been devoted to the issue, defining a compact, optimized environment model and a fast algorithm. They are described in the next sections.

## 3. GEOMETRICAL MODELS

In order to create our representation of the environment, two models are defined: the *ERA model*, representing the arm, and the *world model*, representing the ISS Russian Segment.
The ERA model consists of a set of four connected *links*. A link is defined as the *spherical extension* of a segment, i.e. as the volume containing all the points having a distance from the segment less or equal than a properly chosen value R (figure 2).

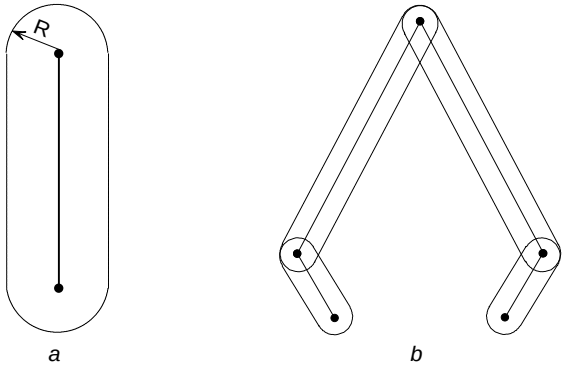


Figure 2. link model (a) and ERA model (b)

Besides the geometrical definition, the ERA model is also a kinematic model, that is, when the arm moves, the angles between links are set according to the joint resolvers readout.
The model dimensions are tailored to contain the entire shape of ERA. It can be observed from figure 1 that not all ERA links have full cylindrical symmetry, due to handles and other small protrusions. To minimize R, the link model axis is aligned to the real link axis only in the two short limbs of era (connected to the hand and to the shoulder). In the long limbs (connected to the elbow) the link model axis is slightly disaligned, to have an higher accuracy in modeling the real link geometry (figures 3 and 10).
The advantage of such a simple ERA model is that the calculation of distances between the arm and the environment deals essentially with segments (taking into account the radius R).

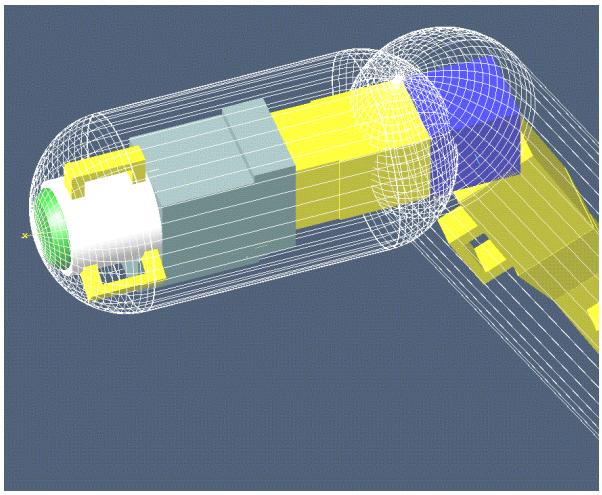

Figure 3. link model of the ERA hand

The world model follows a different philosophy. It shall be capable of representing complex shapes of ISS components, that don't fit in any general simplification. In this case, the approach is modular and hierarchical. Since the ISS is not yet in place we define, as starting assumption, the "real" ISS Russian Segment by its geometrical specification. Then, a first split into big, recognizable components is made. These components are called *obstacles*. For each obstacle, a hierarchy of nested sub-components is defined, in a way that matches the hierarchical organization of the check algorithm (figure 4). The idea is that while ERA comes closer and closer to an obstacle, the obstacle is seen as smaller and more refined models, that at the lowest level coincide with the real ISS shape. Only when ERA is very near to the environment, the algorithm needs to go all the way down, ending with the real ISS shape; in all other cases, a few high level checks are enough to guarantee operational safety.
Obstacles are composed of *elements*. An element is characterized by a *bounding box* oriented as the main reference frame *x*, *y*, *z* (the *world frame*) which is uniquely defined in the Russian Segment. To get the maximum benefit from the bounding box usage, the modeler should define as elements those parts of obstacles that best fit in parallelepipeds aligned with the world frame (figure 5).
As explained in the next section, all objects shall be enclosed in additional margins. In the following, when talking of bounding boxes, bounding spheres, etc., the margins shall be considered included, as shown in the figures.
Elements are composed of convex *polyhedra*. Note that the convexity is not required for elements. A polyhedron is characterized by a set of *faces* connected by *edges* and *vertices*, which are, finally, just points of the space. A face is also characterized by an orientation, i.e. a unit vector located on the face itself and pointing outward from the polyhedron, that will be used by the algorithm to tell the outside from the inside of the polyhedron

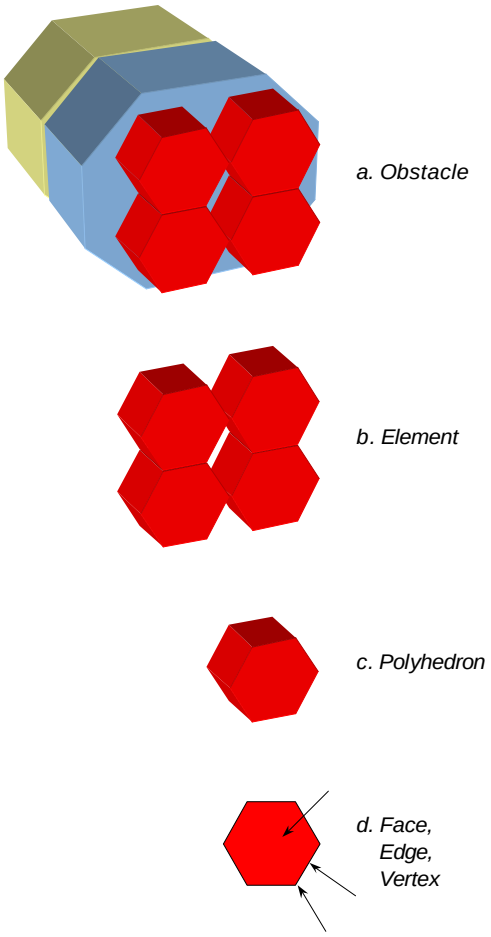


Figure 4. World model hierarchy

It is easily understandable that the polyhedron level of description requires intensive calculations. To speed up part of them, an intermediate model has been introduced: the *bounding sphere*, i.e. a sphere circumscribing the polyhedron (plus margin). The bounding sphere can be a more or less accurate approximation, depending on how much the given polyhedron shape is close to a sphere.
The last model is the *spherical extension* of the polyhedron, defined as the volume containing all the points having a distance from the polyhedron less or equal to the margin (figure 6). The spherical extension is nothing more than the real shape enlarged by the margin; it is the most accurate and CPU demanding representation.

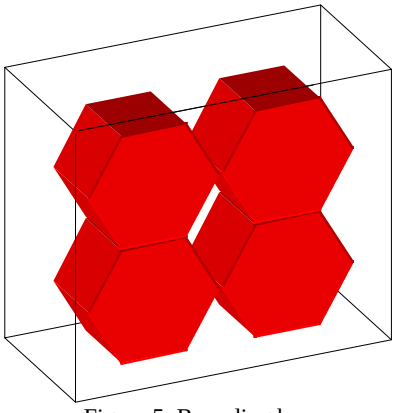

Figure 5. Bounding box

The generation of the ERA model and the world model is performed on ground with the support of an additional software, also developed by Tecnospazio, which converts CAD models into an optimized format. This optimized format is more compact then the CAD format and tailored to the Collision Detection algorithm; this is the format in which the models are uploaded into the ERA Control Computer.
The world model is easily kept up to date with the reality: when future modifications will occur in the shape of the ISS Russian Segment, they will be reflected into the software by loading a new version of the world model, a routine operation to be performed in flight.

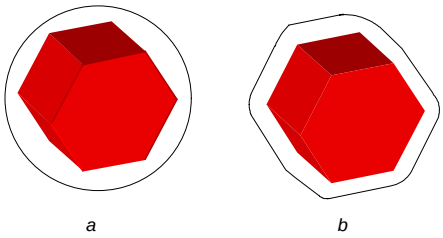


Figure 6. Bounding sphere (a) and spherical extension (b)

A special case is represented by the modifications of the ISS caused by ERA itself after grappling and releasing payloads (figure 7).

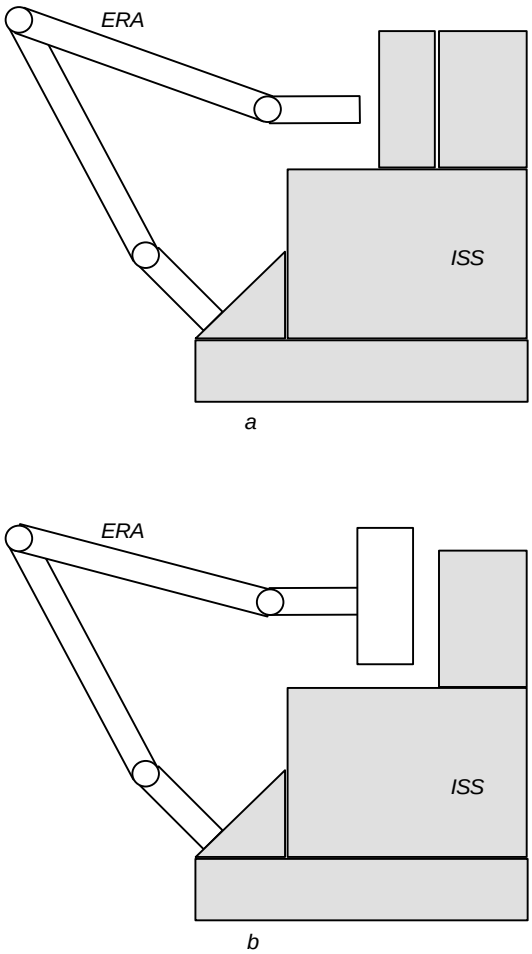


Figure 7. Change in the world model after payload grappling

A payload fixed to the external structure of the ISS is indeed part of the ISS, modeled as obstacle as long as ERA does not grapple it. After grappling, two things change in the geometry:

- ERA contains a new appendix (the attached payload), that shall be checked for collisions against the environment, just as a part of the arm;
- a “hole” has been produced in the world model (the space formerly occupied by the payload) that shall be discarded from the volumes to check.

The reverse happens when a payload is released by ERA and accommodated on the ISS.

The software automatically reflects these modifications into the Collision Detection database. In fact, part of the grapple/release procedure consists of updating the world model so that an obstacle becomes a payload or vice-versa.

The last model still to be described is the payload when grappled by ERA. Unlike the obstacle case, a general simplification is done here. Based on the shapes foreseen for ERA payloads, it has been decided that a realistic model can be defined using the spherical extension of a segment (as done for the link model), intersected by two spheres (figure 8).

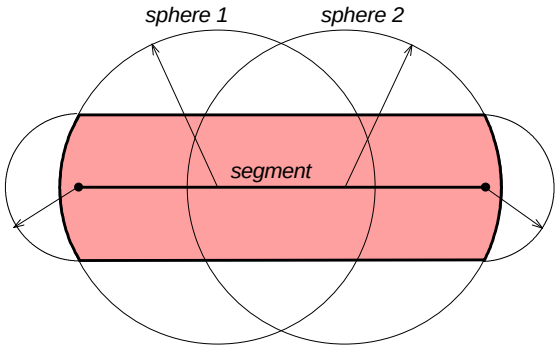


Figure 8. Payload model

The purpose of the spheres is to minimize the error made at the segment extremes. The optimal radius and centers are automatically computed by the ground software. This model provides acceptable approximation of the foreseen payload shapes and presents the benefit of requiring not too heavy geometrical calculations.

## 4. HOW THE CHECK WORKS

The ERA onboard software is a real-time multitasking Ada program, where periodic operations are implemented as cyclic tasks with suitable frequency and priority. Among them, the family of the Health Checks consists of tasks aimed to detect deviations from the safe functioning of the most critical subsystems. The general way these checks work is that in every cycle they return a check status that can be *nominal*, *caution* or *danger*. The check status is then read by a supervisor task, that performs the proper recovery action.

The Collision Detection check fits in this framework as a low priority, 1 Hz frequency Health Check. It is activated only when ERA is moving, and can be disabled by the user, like most Health Checks. Nominal means no collision, caution means that the collision is signaled to the user display, requiring to manually stop the arm. Danger means that the computer shall take control and immediately engage the robot brakes whatever the user does.

The Collision Detection check reads from the robotic motion subsystem the current arm position and the speed of each joint. Then, it calculates two numbers: the caution and the danger margin. These margins are enlargements to be added to the shapes used by the algorithm (bounding boxes, bounding spheres, spherical extensions), to take into account a number of contributions; the most important are the following:

- speed of the arm: the sampling of input data introduces a loss of resolution in the order of the maximum displacement of ERA during one period;
- stopping distance: from the instant when the software detects a collision to the instant when the brakes are actually engaged, an amount of time is spent, due to software reaction time and brakes activation inertia;
- arm elasticity: when fully braked, the arm will oscillate in the same direction of the motion, for an extension that depends on the speed and on the physical characteristics of limbs and joints;
- cosmonaut reaction time (only for the caution margin): a customizable database parameter representing the human reaction delay from the warning signal to the controlled stop command, usually assumed in the order of one second.

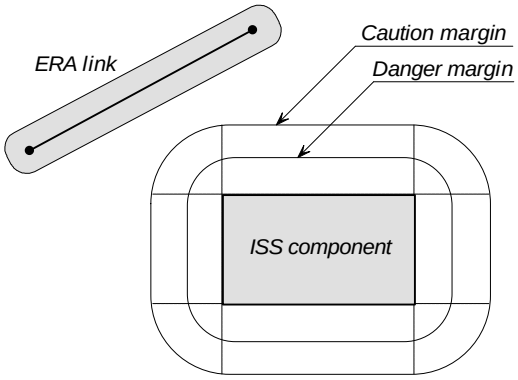


Figure 9. Caution and danger margin

Once the caution and danger margins are known, the algorithm is run against all obstacles, first using the danger (smaller) margin: if it finds a collision, the check result is danger, otherwise the algorithm is run using the caution (larger) margin. If it finds a collision, the check result is caution. Otherwise, it is nominal.

The relation between the arm speed and the margins leads to the following typical scenario: after a danger collision is detected and the arm stops, the control is returned to the user, who shall then retract the arm to a safe position in order to restart the move. But the arm is intersecting, by definition, a danger volume, therefore if the arm speed is the same as before the collision, obviously the check will trigger again and the arm will not move at all. The only way to get out from the danger volume is by reducing the margin, which means that the user is forced to reduce the speed until the danger volume does not intersect the arm anymore; only in that condition the retract maneuver can be performed. ERA provides a set of predefined velocity scaling factors, so that it is possible to re-use properly rescaled standard moves, with no needing of dedicated "slow motion" commands.

An exception in the described check functioning is the proximity motion, i.e. the control mode used by ERA to approach a payload to be grappled (and, conversely, to retract from released payloads). In this mode, the arm is driven very close to the payload, using data from the camera mounted on the ERA hand. Clearly, the Collision Detection cannot be used while this operation is in progress, because it will trigger collision between the arm and the payload, making impossible the approach. Therefore, during proximity motion, the Collision Detection is simplified: it executes only the link-link check, and then the link-obstacle check only for links connected to the elbow (i.e. far from the payload).

## 5. ALGORITHM DESCRIPTION

Up to now we have discussed how the Collision Detection check operates in the context of the ERA onboard software and what are the check inputs and outputs. Now the check algorithm itself will be described.
After the caution and danger margin calculation, the next step is to check for all possible collisions.
Four collision types are defined, each of them leading to an independent part of the algorithm:

- link-link,
- link-obstacle,
- payload-link,
- payload-obstacle.

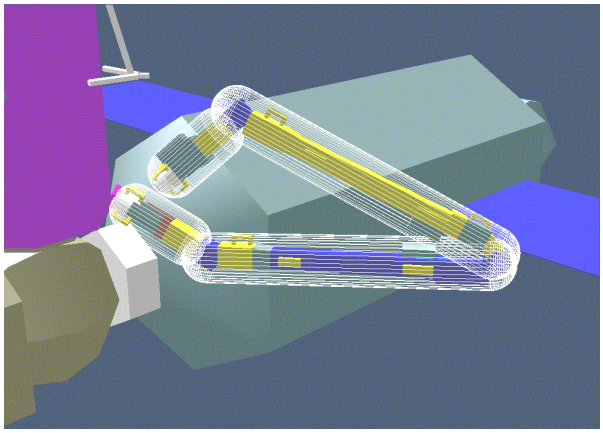

Figure 10. Link-link collision

The link-link algorithm (figure 10) checks for collisions between non consecutive links (consecutive links are obviously always in contact). It compares the current margin (danger or caution) with the distance between spherical extensions of the links.

The link-obstacle algorithm (figure 11) checks for collisions between all ERA links and all obstacles in the world model. For each obstacle, the check is performed hierarchically on all sub-components. First, the elements: for each of them, each link position is established against bounding box of the element. If no intersection is found, then the current element is discarded (the contained objects certainly don't collide) and the check moves on to the next element. Otherwise, if the link intersects the bounding box, then there are two possibilities: the link is really colliding with an object inside the element, or it is colliding with an empty part of the bounding box, therefore, further refinement is required. In this case, the algorithm shall go down to the polyhedra level. It first checks the bounding sphere of each polyhedron in the element.

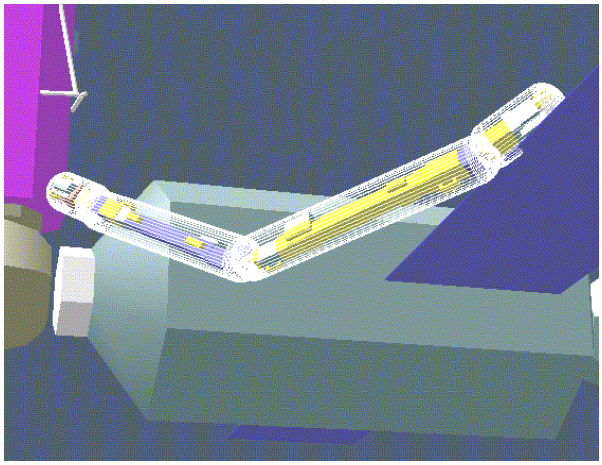

Figure 11. Link-obstacle collision

Applying the same philosophy described above, if there is no collision, it goes to the next polyhedron, elsewhere it refines the check by switching to the spherical extension of the Current polyhedron. This check is in turn split as follows: check for link intersection with faces, then, to follow the spherical rounding, a correction on the edges and then on the vertices of the polyhedron is applied.
All these checks are executed in the described order, that is, they are sorted by increasing complexity. This choice allows considerable CPU time saving, because for most obstacles it is unnecessary to extend the check depth to the detailed shapes leading to intensive calculations. The algorithm operates at full depth only for those objects that are complex, close to the arm, and that don't collide with it (if they do, the task immediately returns positive result and turns to idle until the next cycle begins). In all other cases, the tests show that the CPU load is reasonably light.

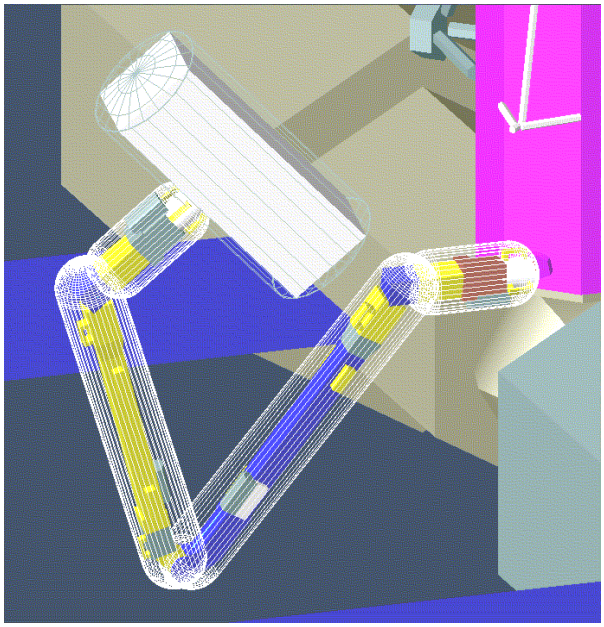

Figure 12. Payload-link collision

In other words, the CPU load due to the Collision Detection for a given trajectory is proportional to the “model

complexity density" along that trajectory, not to the overall world model complexity. Indeed, if however complex objects are far from the arm, they are seen as elements, and the computation time to discard them from the candidates to collision is small.

The payload-link algorithm (figure 12) checks for collisions between the grappled payload, if any, and all ERA links, excluding the tip (which is connected to the payload itself). This algorithm is simpler than the previous one because of the simpler payload model format (intersection between the spherical extension of a segment and two spheres). The possible collision of a link model with the payload model is reduced to logical operations applied to the separate intersections between the link and each sub-component of the payload (link-spherical extension, link-sphere1 and link-sphere2).

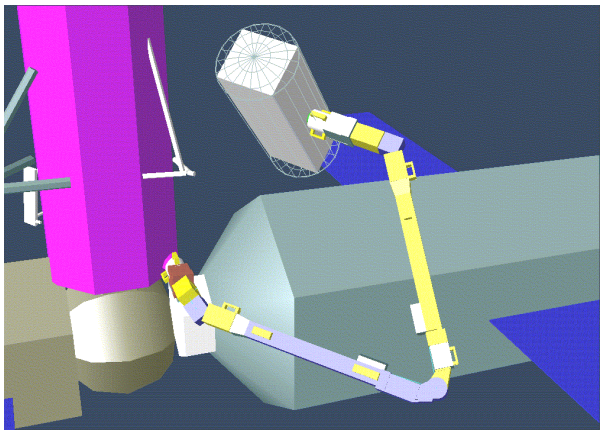

Figure 13. Payload-obstacle collision

The payload-obstacle algorithm (figure 13) checks for collisions between the grappled payload, if any, and all obstacles in the world model. Since the payload model is composed of a segment spherical extension intersected with two spheres, the algorithm takes the advantage of re-using the link-obstacle algorithm previously described, to first intersect the spherical extension with the obstacle. The spherical extension is bigger than the payload model, therefore if no collision is found here, that's enough to discard the current payload from the candidates. Elsewhere, if a collision is found, it is necessary to refine the check by intersecting the two spheres composing the payload model with the obstacle. If at least one of the spheres collides, then the payload collides.

## 6. PERFORMANCE

The performance figures collected in the simulated test environment authorize to say that the algorithm fits in the allocated budgets of CPU and memory. A summary is reported in figure 14. It can be noticed that, in this phase, it has been found convenient to use elements containing only one polyhedron each.
The total memory needed for this model is 110 Kbytes. Additional spare memory has been allocated up to 164 Kbytes, leaving room for future model changes and improvements.
As far as CPU is concerned, the Collision Detection execution time measured for many random trajectories proved that it is indeed quite fast, and the ERA Control Computer can run it in parallel to its nominal control tasks.
The algorithm is quite accurate: the worst case approximation has been found to be 50 mm, i.e. less than 0.5% of the arm length. Of course, additional approximations are introduced by CAD modeling of real ISS, which is matter of trade off between memory occupation and accuracy.

| | |
|---|---|
| World model composition | Obstacles: 21 |
| | Elements: 91 |
| | Polyhedra: 91 |
| | Faces: 767 |
| | Edges: 1698 |
| | Vertices: 1113 |
| Used memory [kb] | 110 |
| Total memory [Kb] | 164 |
| Worst case execution time [ms] | 26 |
| Worst case numeric error [mm] | 50 |

Figure 14. Performance summary

## 7. CONCLUSIONS

The Acceptance Tests successfully performed at Tecnospazio show that the developed software matches the specified functional and performance requirements. In particular, the practical feasibility of the Collision Detection concept has been demonstrated with the execution of a simulated ERA Reference Mission containing proximity, grappling and releasing operations, with the Collision Detection check always enabled.

## 8. REFERENCES


R. Gallerini, A. Sciomachen, "Collision Detection using linear programming", 2nd European In-Orbit Operations Technology Symposium, Toulouse, 1989

J.F.T. Bos, M.J.A. Oort, "Failure Detection, Isolation and Recovery system concept for the European Robotic Arm", ESREL ’97 European Safety and Reliability Conference, Lisbon, 1997

P.G. Beerthuizen, C. Maegaard, A. Rusconi 1998, "ERA Safety Strategy", DASIA ’98 Conference on Data Systems in Aerospace, Athens, 1998

R.A. Bosman, J.F.T. Bos 1998, "Control of the joint runaway hazard for the European Robotic Arm", ESREL ’98 European Safety and Reliability, Conference, Trondheim, 1998

P.G. Beerthuizen, W. Kruidhof 1999, "System and Software Safety Analysis for the ERA Control Computer", SAFECOMP ’99 European Workshop on Industrial Control Systems: Technical Committee 7 – Safety, Reliability and Security, Toulouse, 1999