N90-29903

A COLLISION AVOIDANCE SYSTEM FOR A SPACEPLANE MANIPULATOR ARM

A. SCIOMACHEN, P.G. MAGNANI

Tecnospazio S.p.A.
Via Mercantesse,3 - 20021 Baranzate di Bollate
MILANO (ITALY)

Abstract

The aim of this paper is to report on part of the activity performed by Tecnospazio in the area of collision avoidance related to the Hermes spaceplane. In particular, a collision avoidance system which has been defined, developed and implemented in this project is presented.

1. Introduction

The collision avoidance capability is a necessary feature for good safety performances of both automatic and teleoperated space robotic systems [1,2].

In **Figure 1** the Hermes spaceplane currently under development at ESA is represented [3,4]. The spaceplane, similarly to the US-STS, carries a manipulator arm presently designed with six degrees of freedom, two at the shoulder, one at the elbow and three at the EE, respectively. The arm is about 10 m. in lenght, presents a non negligible flexibility and can operate both automatically, through programmed sequences, and under direct human teleoperation.

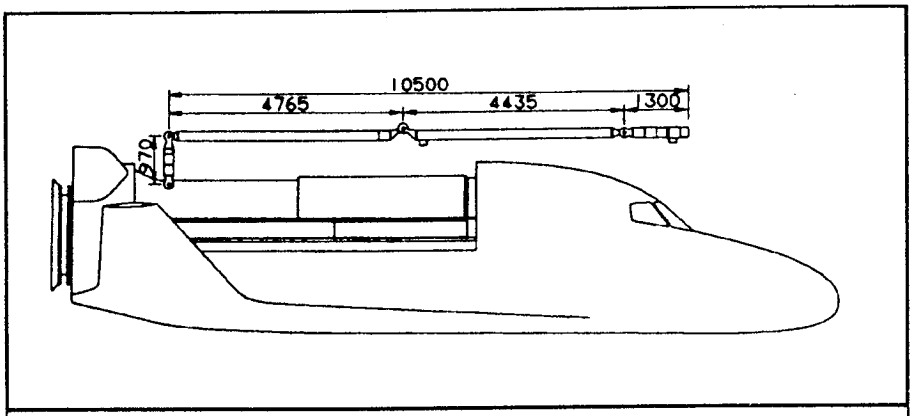

Fig. 1 - HERMES Spaceplane

It must anyway be noted that during the teleoperation the astronaut has not full visibility of the arm/obstacle relative position, while during automatic mode a protective "software mask" is advisable in order to prevent any system failure. It is therefore essential to have an anticollision capability flexible enough to operate off-line (during task planning), on-line (during task execution) and to allow the operator easy environment modifications.

The proposed collision avoidance software approach is presently under development/preliminary implementation in the industrial world. Nevertheless,the techniques and the concepts that are considered are already widely used in specific applications, e.g. CAD/CAM, multiarm robots. Therefore the approach used, even if advanced in its conceptions, has been kept as practical as possible, avoiding the use of techniques not yet established.

2. Mission Requirements for a Collision Avoidance System

In this scenario, a collision avoidance system is a software tool whose task is to detect a status of collision which may occur between the Hermes spaceplane, the objects hardcoupled with Hermes, the manipulator arm and a payload attached to the manipulator arm.

A status of collision must be verified through the execution of the related algorithms both under real conditions and under simulation. In the first case HERA is moving and the system is in operating mode; otherwise, no HERA movement is involved.

The on-line system is active both in automatic mode and in operator controlled mode/single joint mode. Automatic mode means execution and verification of preprogrammed trajectories under automatic control. In this case the system asks for the operator support any time a collision is detected. In the other mode, operator controlled movements are executed and verified; the operator controls directly the arm through a suitable device, for instance a joystick. The single joint mode is activated in case of failures of the control system in order to retract HERA to its stowage position moving one joint at a time and in operations in which a fully stretched arm is required.

The HERA programming mode takes place off-line and is executed on ground, by an operator. If a collision is detected, the operator must modify the current trajectory using the information provided by the system.

The inputs to the system are a set of joint coordinates representing different configurations of the arm during its movement along a planned trajectory, and the stopping distance value, that is the distance required to stop the arm. Starting from these information, the system verifies whether a collision is possible or not. The following four types of collision have to be verified:

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

Due to the geometric characteristics of the objects likely to be involved in a collision, each of these types of check is performed sequentially by a separated algorithm.

Observing **Figures 1 and 2**, representing respectively the Hermes and the HERA referring configuration, some considerations about the workspace can be derived. In particular, no collision is possible between the first link and any obstacle, between contiguous links and between the payload and the last three links. No restrictions exist as far as a possible collision between the payload and any obstacle is concerned: the payload, in fact, can collide with an obstacle in any position reachable by the arm.

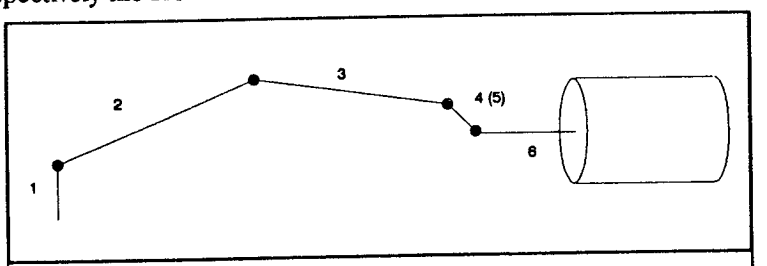

Fig. 2 - HERA Referring Configuration

The collision status (yes/not), together with other suitable information for a path planning and a validation support, are provided by the system in order to allow a timely and efficient intervention. The outputs are sent both to an user-interface module as video messages and audio warnings and to the manipulator controller. In particular, when a collision is foreseen, the following information are given:

- a collision warning
- the objects involved in the detected collision, i.e. arm, obstacle, payload
- the colliding link, the face of the obstacle and the intersection point, in case of collision of the arm with an obstacle
- the two involved links and the collision point, in case of collision of the arm with itself
- the face of the obstacle intersecting the payload, in case of collision of the payload with an obstacle
- the colliding link, in case of collision of the arm with the payload.

A single operator interface must manage all the warnings and the information needed. The above information are not presented directly by the collision avoidance system but are returned to the calling program that must send them to the operator interface of the overall system. It has to be observed that, apart the collision detection, all the operations required in the recovery phase, e.g. switching between operating mode, are not in charge of the collision avoidance software. The collision avoidance routines send information to the system that performes the necessary actions.

It is required that the collision avoidance system be effective keeping at the minimum the false warnings. A reasonable upper bound of the resolution of the system is the stopping distance: this means that the sum of the margins used to take into account discretizations, uncertainties, tolerances and approximations in the geometrical models cannot be greater than the maximum stopping distance.

The resource availability for the collision avoidance system is very limited: the overall system must run, at most, at a rate of 18 kflops. This value includes the time required to check the conformity between the outer world and the model contained in the database. Moreover, the memory available on board is about 40 kbytes.

3. The Proposed Collision Avoidance Method

The proposed collision avoidance software system [5,6] computes the intersection between the solids representing the arm, the payload and the objects. Only the approximated geometrical models of any kind of objects, increased by the stopping distance value, are considered. So two objects are defined to be in collision if and only if their relative distance is less than the stopping distance.

The computation of the stopping distance, which is a function of the mass and velocity of the arm and of the payload, and of the brake feature, is performed by an external module.

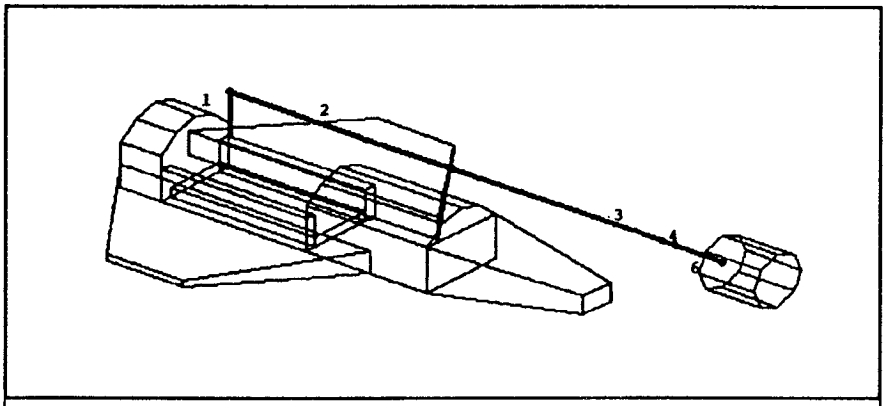

Fig. 3 - The HERMES Model

Information about the arm position and the outer world are given through geometric models and the related data are contained in specific databases. The obstacle database contains the geometrical description of all the fixed objects, i.e. Hermes parts and other obstacles, if any. The basic element is a convex polyhedron: non convex polyhedra can be modelled as the union of a finite number of convex polyhedra. For each object contained in the database the equations of the faces and the coordinates of the edges, defined with respect to the global reference frame, are stored.

The Hermes model, depicted in **Figure 3**, is composed of 8 convex polyhedra, 54 faces and 76 vertices. In particular, it is composed of:

- tail : a prism with 7 base sides;
- cargo bay : 3 prisms with 4 base sides each;
- left wing : a generic convex polyhedron;
- right wing : a generic convex polyhedron;
- cockpit : a prism with 7 base sides;
- forward fuselage : a pyramid with 4 base sides.

As the shape of the links is about cylindrical, the geometry of a link is defined giving the length of the axis and the radius. The payload is modelled as a cylinder.

Starting from the data contained in the world model database, the collision avoidance tasks are assigned both to a geometric method and to a forbidden values method. A possible combined implementation of the two methods can be used; in this approach, the forbidden values method checks the collision between the first three links and any obstacle, while the geometric method is used for all the other types of collision.

As it has been already said, in the collision detection phase four main modules are used in the collision avoidance system, namely:

- link-obstacle module
- link-link module
- payload-obstacle module
- payload-link module.

Before the execution of the geometric method can start, an initialization procedure is required in order to compute the coordinates of the geometric baricenter of the obstacles contained in the database and the radia of the spheres circumscribing the obstacles. As it is herebelow described, these data are required in the preliminary collision tests.

A basic step of any collision avoidance module is the direct kinematic computation. From joint coordinates the cartesian positions of the two extremes of each link are obtained using the Denavitt-Hartemberg matrices [7].

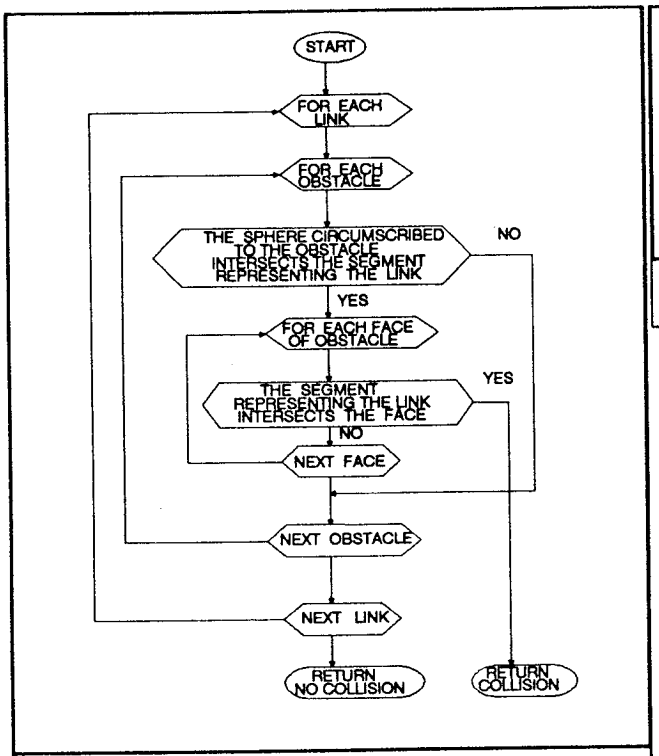

Fig. 5 - Link Obstacle Module Flow-Chart

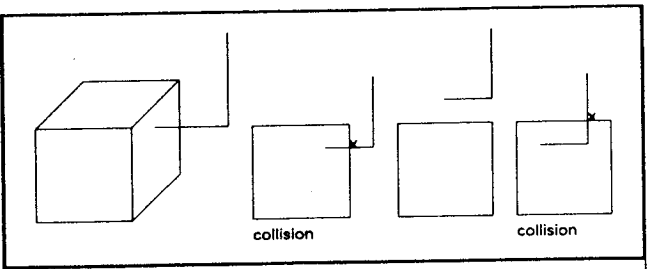

Fig. 4 - Link Obstacle Collision Situation

Link-Obstacle Module

The links involved in the link-obstacle collision module are 2, 3, 4 and 6 (see Figs. 1 and 2). Link 1, by definition, cannot collide and link 5 has length = 0 and hence it is not considered.

This module finds the intersections of the segment representing a link with the obstacles, that are enlarged by a value equal to the maximum radius of the link. This is the so called 'obstacle growing' technique. The possibility of a collision between the link under consideration and any obstacle is initially verified intersecting the sphere circumscribing the obstacle with the segment representing the link. If no intersection is found the distance between the two objects is such that no collision is possible. Otherwise, a further check is necessary and for each face of the obstacle the intersection between the straight line including the link and the plane containing the face of the obstacle is verified. If an intersection point is found and if this point belongs to the link, then it is to be verified whether the point lies on the obstacle face or not. So it is possible to say that there is a collision if and only if there is one point belonging both to the link and to one face of the polyhedron representing the obstacle.

It has to be noted that the case in which the segment lies on the plane including the face does not represent a collision situation. Moreover, the case of parallelism between the segment and the plane is not considered; in fact, in such a situation a possible collision is found checking for the other links the intersection between the segment and another face, adjacent to the present one. **Figure 4** depicts the situations in which a collision is detected and those in which there is not a collision.

The whole link-obstacle collision module is summarized in the flow-chart of **Figure 5.**

Link-Link Module

In the link-link collision module a link is modelled as a cylinder with two hemispheres at its ends. In this module the possibility of a collision between the arm and itself is verified, checking whether the distance between the two segments representing the links is less than the sum of the radia of the links. An example of collision-free situation is given in **Figure 6.**

Referring to Figure 2, only the following couples of links must be verified: 1-3, 1-4, 1-6, 2-4, 2-6 and 3-6.

Starting from the coordinates of the extremes of the links, the minimum distance between two segments in a 3D space is computed [8] through the derivatives, with respect to the two links equations parameters, of the distance equation and solving the resulting system. If its determinant is not zero, the points at minimum distance are computed. If both the two identified points belong to the segments representing the links then their coordinates and their relative distance are computed. If at least one of the points does not belong to the link, eight different cases, i.e. the extremes of one link against a point

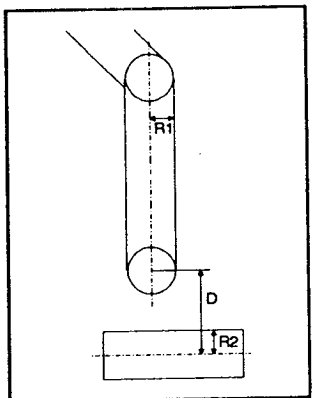

Fig. 6 - Link-Link Collision Test

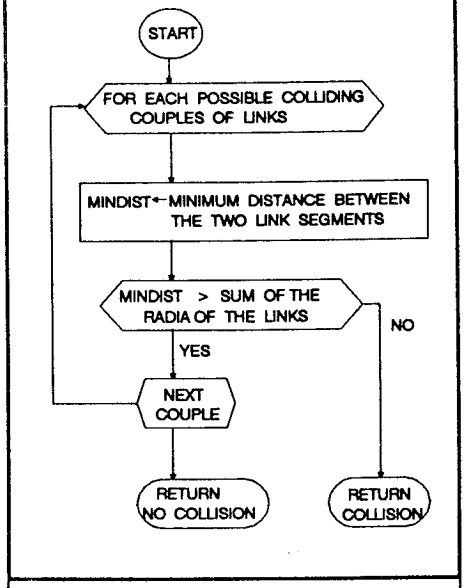

Fig. 7 - Link Link Module Flow-Chart

internal to the other link and the extremes of one link against the extremes of the other link, are to be examined. The eight distances so computed are then stored. Among all the above distances the minimum one and the corresponding points are extracted. Otherwise, i.e. if the determinant is zero, the links are parallell and their minimum distance and the relevant points are computed. A check is performed in order to verify whether the two axes are coincident or not. In the first case, the minimum distance is equal to zero if the two segments are overlapped and to the distance between the two nearest extremes, otherwise. In the second case, the minimum distance is equal to the distance between one extreme and the normal projection of the extreme of the other link. This is obtained by computing the intersection between the first link axis and the plane normal to the first link and containing the extreme of the second link.

The whole link-link collision module is summarized in the flow-chart of **Figure 7.**

Payload-Obstacle Module

In the payload-obstacle collision module the possibility of a collision between the payload and any obstacle is verified. For this reason, a preliminary test is performed in order to exclude those obstacles that have no possibilities of colliding with the payload. This test is performed checking whether the position of the payload is such that a collision with the obstacle under consideration can occur.

In this module both the payload and the obstacle are inscribed in a sphere. At the beginning the distance between the payload and the obstacle is computed and compared with the sum of the radia of the two spheres. If it is larger then there is no collision. If the spheres intersect each other the possibility of a collision between the payload and any face of the currently examined obstacle has to be verified computing the distance from the extremes of the axis of the payload to the plane containing the examined face. Some different cases can arise. Referring to **Figure 8a**, if $D_1 > R$ and $D_2 > R$ and the two extremes are not on the opposite side of the plane then there is no collision between the payload and the face of the obstacle. If such a condition is satisfied the next face is verified. Otherwise, it has to be checked whether the projections of the two extremes of the payload on the plane including the face of the obstacle are internal or external to the face. Referring to **Figure 8b**, if $D_1 < R$ and P_1 belongs to the face, or if $D_2 < R$ and P_2 belongs to the face, or if the extremes of the

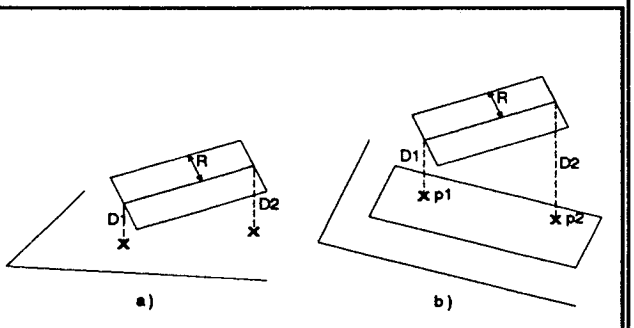

Fig. 8 - Payload Obstacle Collision Test

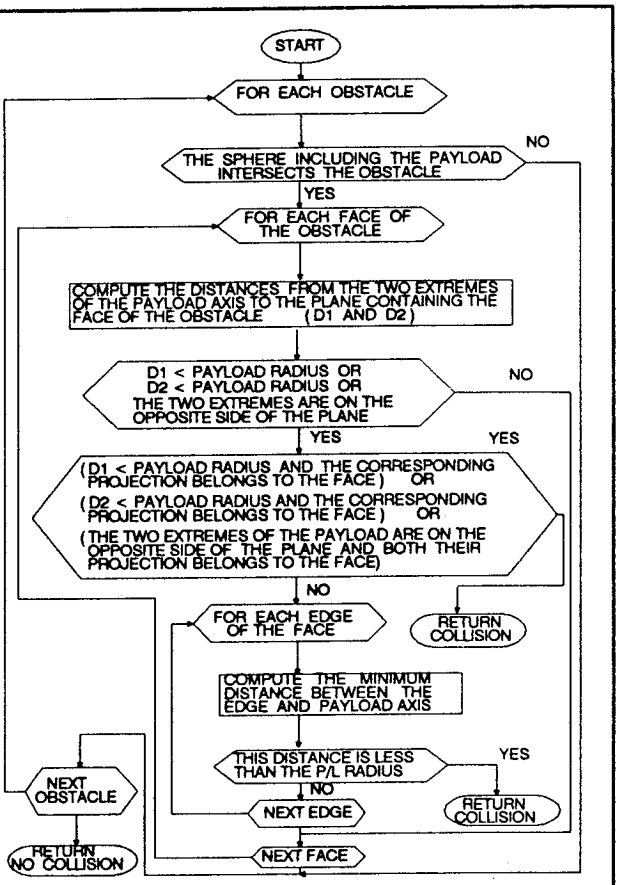

Fig. 9 - Payload Obstacle Module Flow-Chart

payload are on the opposite side of the plane and both their projections belong to the face then there is a collision. If none of these three conditions is verified a further check is required in order to establish whether a collision situation between each edge of the current face and of the examined solid is presented. For this reason the minimum distance between the two involved segments is computed; if it is less than the radius of the payload then there is a collision.

The whole payload-obstacle collision module is summarized in the flow-chart of **Figure 9.**

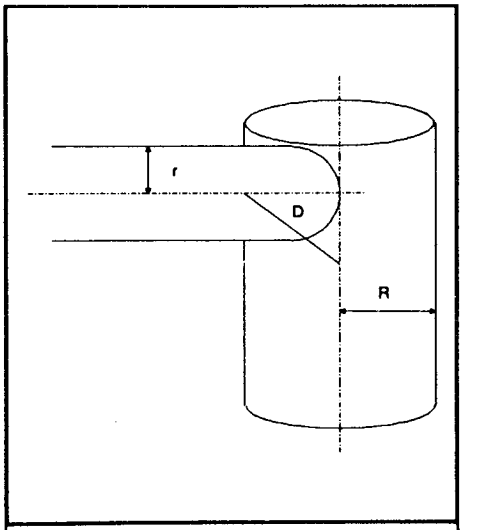

Fig. 10 - Payload Link Distance

Payload-Link Module

In the payload-link collision module the links which can collide with the payload, i.e. the first four links, are examined. If a sphere is circumscribed to the payload, modelled as a cylinder, the situation is the same as in the link-obstacle collision module. If there is the possibility of a collision the determinant of the second order equation formed by the intersection between the straight line connecting the extreme of the link with the medium point of the payload axis and the sphere is computed. If it is ≥ 0, i.e there is an intersection between the straight line and the sphere, the roots equation corresponding to the intersection points are computed. If there is an intersection, two cylinders, as in the link-link collision module, having different diameters are considered and the minimum distance between two segments, representing the link axis and the payload axis, in the space is computed, as shown in **Figure 10.** If the distance is less than the sum of the payload radius and the link radius, then there is a collision.

The whole payload-link collision module is summarized in the flow-chart of **Figure 11.**

Forbidden Values Method

The forbidden values method precomputes the forbidden joint ranges for the first three links and stores them, off-line, as a set of trees, depicted in **Figure 12.** The forbidden ranges represent the link-obstacle collision positions which are accessed, on-line, along the tree. Only a small subset of the data structure, which is represented by the neighbourhood of the current set, is loaded in the central memory. Each substructure represents the forbidden values for an interval of the first two joints, e.g., as depicted in **Figure 13,** when the first joint sweeps from 20 to 30 degrees and the second joint sweeps from 70 to 80 degrees. For each discretized position of the first link, which cannot collide with the obstacles, the intersection angles of the second link with the obstacles are found and the allowed ranges are stored. Then, for each discretized position of the second link within an allowed range, the intersection angles of the third link with the obstacles are computed and the allowed ranges are stored. In the same way the third link is checked for collisions with the first link.

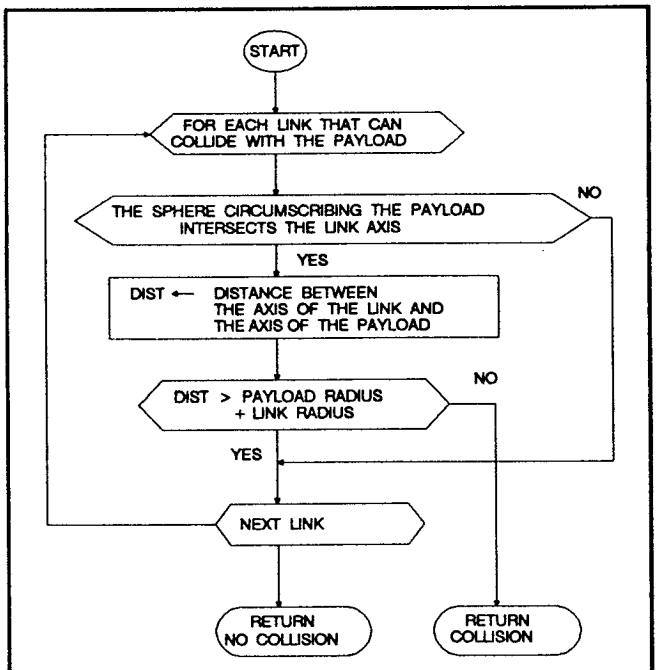

Fig. 11 - Payload Link Module Flow-Chart

In this method,based on a work of Lozano-Perez [9], the intersection between a link, modelled as a segment, and an obstacle, an enlarged polyhedron, are found computing the rotation center and the plane in which the second and third links rotate and finding the intersections between the sweeping circle and the obstacle faces.

4. Computational Results

In this Section the performances of the proposed collision avoidance system are analyzed and some conclusions regarding trade-off between precision, memory and time are presented.

Different running tests of the algorithms have been performed on the basis of 1,000 configurations, randomly chosen in the working area of the arm, to be checked for collision. The main sampled data, related to the geometric method, are the following:

- Average # of multiplications : 1651
- Standard Deviation : 195
- Maximum # of multiplications : 6064

It is important to emphasize that the maximum and minimum number of multiplications are related, respectively, to the payload-obstacle and payload-link tests. Moreover, the maximum and minimum number of collisions are detected, respectively, in the link-obstacle and payload-obstacle tests.

Assuming that a multiplication requires 1 time unit, a sum 1/3 time unit, a square root 3 time units and a trascendental call 7 time units, a conservative evaluation of the number of FLOP required by the overall algorithm is, in the worst case, about 11,000.

Using a variable stopping distance, an obstacle growing algorithm is also used. For the Hermes model described above, this algorithm requires 325 FLOP. This number grows linearly with the complexity of the model. As the FLOP required by the obstacle growing is about 3% of the total, and because the frequency of activation of this program can be 1/10 of the frequency of activation of the collision avoidance checks, this contribution can be neglected.

The computational cost C (kFLOPS) of the geometric algorithms is given by:

$$C = 11 * f \tag{4-1}$$

where f (Hz) is the frequency of activation of the algorithms, related to the discretization error through the speed by:

$$Ed = v/f \tag{4-2}$$

where v is the speed (m/sec) of the fastest part of the arm and Ed is the discretization error (mm). The discretization error is due to the fact that all the checks are performed only in discrete positions. Assuming a computer power of about 18 kFLOPS, which allows a maximum frequency of activation of 1.7 Hz, it is possible to evaluate this error on the basis of the formulas 4-1 and 4-2. The discretization error for different arm speeds is herebelow reported. The values are related to all the algorithms running together, included the kinematic transform:

50 (mm/sec) ⇒ 29 (mm)
100 (mm/sec) ⇒ 59 (mm)
200 (mm/sec) ⇒ 118 (mm)

As it has been already mentioned, the precision of the method is affected also by the stopping distance and the geometric model.

The stopping distance is treated as a stepwise function of speed and load of the arm. Letting the maximum value of the stopping distance be 100 mm, i.e. the typical value for maximum speed movements, and using 10 ranges, the error due to this contribution can be assumed, in the worst case, as 10 mm.

The chosen geometric model approximates the curve surfaces of the obstacles with polyhedra; the error due to this contribution depends on the exact shape of the obstacle and on the chosen polyhedrical approximation and can be reduced using a more detailed model of the obstacle and rounding sharp edges and vertices. In the model used for the experiments the worst case error can be estimated as about 200-250 mm. This is a very large value: the use of a more detailed model, that can be easily computed on the baseline of Section 3, is hence recommended.

It is worth to remark that a cylindric model of the payload is too difficult to be treated with a good precision without a too heavy computational charge. Infact, a more detailed model increases the required computing power. This increase cannot be evaluated a priori but only a careful trade-off between precision and time can give acceptable results. The preliminary test (pruning) plays a foundamental role in this trade-off, allowing to obtain a better resolution without a great increase in the required computing power.

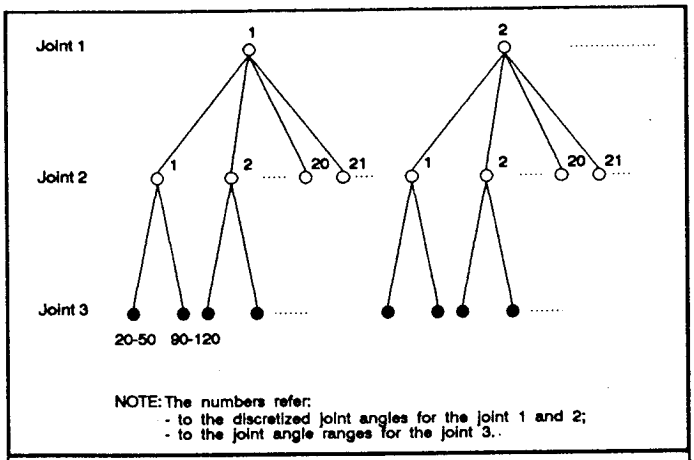

Fig. 12 - Data structure of the forbidden values method

The memory required for the code, using the Fortran 77 language, is about 20 kbyte on a VAX. The memory required for the database, in the case of the Hermes model described above, is about 3 kbytes. Then the use of a quite refined and complex model does not appear to cause problems in terms of memory occupation.

As far as the forbidden values method is concerned, the memory required for the code is about 18 kbyte. The overall structure for the first three joints, with a discretization of 1 degree for the first two joints sweeping over the ranges 0 to 360 and -30 to 210 respectively, has required about 500 MFLOP. The memory required to store this structure is about 600 kbyte.

Also in this case, the dimension of this structure depends on the chosen discretization and on the number of convex obstacles. The dependence on the number of convex obstacles is quite complex and is mainly related to the position and to the size of the obstacles. A reasonable estimate of the memory required by a convex obstacle is 75 kbytes. Then the dimension D of the world model (kbyte) is given by the following formula:

$$D = 75 * N * (1/n1) * (1/n2) \tag{4-3}$$

where n1 and n2 are, respectively, the discretization in degrees of the first and second joint angles and N is the number of obstacles.

The chosen discretization affects the precision of the method. The error due to this contribution is given by:

$$Ed = 2((L3 * \tan(n2/2))^2 + (L2 * \cos(n2) * \tan(n1/2))^2)^{1/2} \tag{4-4}$$

where L2 and L3 are the length of the second and third link, respectively.

Assuming $n1 = n2 = n$ and considering cos(n) about equal to 1 and tan(n/2) about equal to $n\pi/360$, the discretization error Ed (cm) is given by:

$$Ed = n\pi/180 * (L2^2 + L3^2)^{1/2} = 11.4 * n \tag{4-5}$$

In the proposed model the overall structure is composed of 864 substructures, each of them describing a range of 10x10 degrees (see Fig. 13). The maximum dimension of this structure sufficient to store all the ranges is 310 integer pointers and 400 angular values, which implies a memory request for each substructure of about 1.5 kbytes; assuming to have in memory 9 substructures, according to the chosen loading strategy, 13.5 kbytes are hence required.

The central memory required for some typical values of the discretization error are herebelow reported, considering the case of 8 obstacles:

5 (cm)	⇒	70 (kbyte)
10 (cm)	⇒	18 (kbyte)
15 (cm)	⇒	8 (kbyte)
20 (cm)	⇒	4 (kbyte)

It has to be remarked that, because the forbidden values structure is built off-line, the stopping distance must be assumed as the worst value.

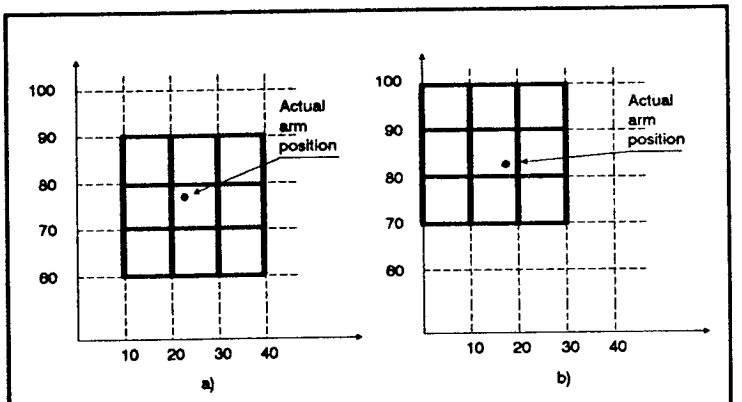

Fig. 13 - Loading Strategy

As a general performance analysis result of both the geometric and the forbidden values method, it is possible to state that:

- the processing power increases linearly with the number of solids
- the errors decrease linearly with the number of solids.

5. Conclusions and Future Work

The overall collision avoidance system architecture is shown in **Figure 14**: basically, it is a merging of the geometric method and the forbidden values method. This architecture applies to both the off-line and the on-line computation assuming a proper system initialization.

The current Tecnospazio study on collision avoidance has shown that such a collision avoidance software system is feasible with respect to the resources available on board, considering its performances.

On the basis of this assumption in the next future the proposed collision avoidance system will be enhanced and completed. Moreover, the generation of the HERA control system software for collision avoidance along with the implementation of a prototype on the HERA simulation facility will constitute the future activities of Tecnospazio in the area of collision avoidance with respect to the HERMES manipulator.

Furthermore, the developed software will be used, with the proper modifications, in the more complex problem of collision avoidance between two manipulator arms which operate within the same working area. This implementation will require a further refinement of specific algorithms together with a modeling improvement in order to allow tighter arm trajectories without falls or unwanted collision detections.

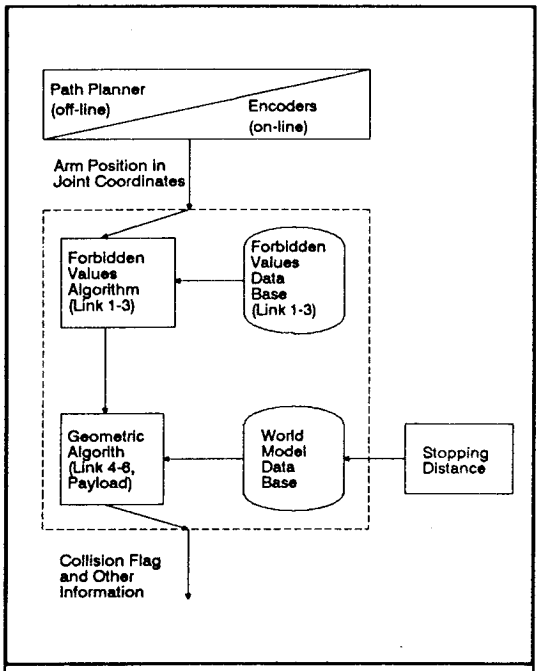

Fig. 14 - Overall System Architecture

REFERENCES

[1] Matra Espace, "Survey of the State of the Art in Robotics and Artificial Intelligence", Technical Report, 1981

[2] NASA, "Advancing Automation and Robotics Technology for the Space Station and for the U.S Economy", TM 87566, 1985

[3] Fokker Space & Systems, "FSS-N-88-HERA-042", 1988

[4] Fokker Space & Systems, "FSS-N-88-HERA-051", 1988

[5] Tecnospazio S.p.A, "HPP: Study on Collision Avoidance", Final Report, STUD005-87, 1987

[6] Tecnospazio S.p.A, "HERA Bridging Phase: Study on Collision Avoidance", Final Report, STUD009-88, 1988

[7] J. Denavitt, R.S. Hartemberg, "A Kinematic Notation for Lower-pair Mechanism Based on Matrices", ASME J. Apll. Mech., 22, 1955.

[8] D.G. Luenberger, "Introduction to Linear and Nonlinear Programming", Addison-Wesley Publishing Company, 1983

[9] T. Lozano-Perez, "A Simple Motion Planning Algorithm for General Robot Manipulators", IEEE J. of Robotics and Automation, June 1987.