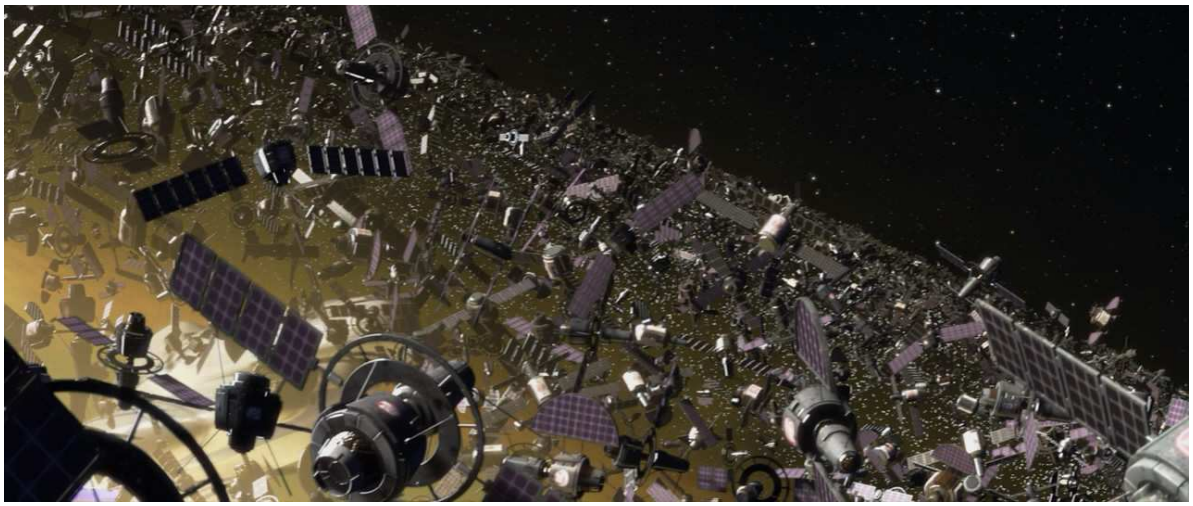
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Space Center
Prof. Herbert Shea

EPFL
Faculte Sciences et Techniques de l'ingenieur
(STI)
Institut d'ingenierie des systemes (I2S)

Orbital Debris Remediation: an approach using a 7-DOF robotic arm in space

Master Project – Spring 2011



Laurent Blanchet
Exchange Student
ENS Cachan, Paris

Muriel Noca
Advisor

clean-m

Space Center

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Contents

Acknowledgements

I would like to thank Aude Billard, Eric Sauser, Ashwini Shukla from LASA and Muriel Richard who helped me correct my mistakes.

1 Introduction

Due to mankind exploitation of space, many thousands of man-made objects are now in space. As we know, space is big and pretty much empty. However, due to mankind needs, the situation is very different in the vicinity of Earth. The situation degrades rapidly in GEO and worsen as we go toward Low Earth Orbit (LEO): there still is a lot of space, but for a first part the useful orbits are kind of the same, and there is a lot of objects, due to the commercial use of those orbits (see figure 1).

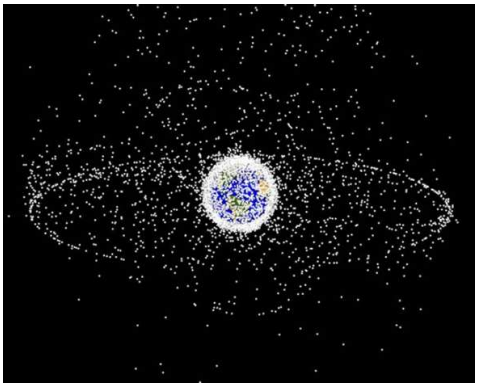


Figure 1: Objects >10cm in orbit: situation in 2010.

As of now, the situation is not good: satellites have to been armored to withstand small impacts from wandering fastenings, and have to be launched with large propellants tanks that will be used to avoid collisions with debris bigger than $10\,cm$. We can quote the example of the International Space Station (ISS), which orbit is really low in order to get it clear of debris. The energetic cost is high, and its orbit regularly needs to be raised. Even with this precaution, the ISS does collision avoidance manoeuvres all year long. Some debris are noticed too late to perform manoeuvres, and the people inhabiting the station have to get into the two Soyuz ships always docked to the station in case of emergency evacuation. Such a scenario happened on June, 28th 2011, while writing this document. The debris was at $250\,m$ from the station.

However contrary to the ISS, the satellites are not equipped to be serviced and when their tanks are empty, they are bound to collide with some other inert object in orbit and potentially create new inert objects (Kessler Syndrome) (see figure 2).

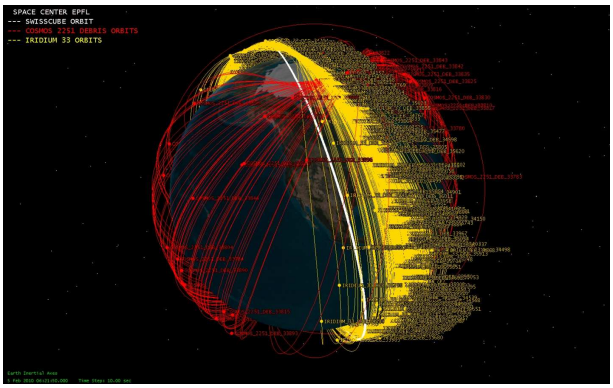


Figure 2: Collision event of Iridium and Cosmos satellites: situation in 2009.

The present situation is already a planned doom of space utilization: event without any more launches, the already present debris will collide and the quantity of debris will overcome the quantity of useful payloads in orbit by 2050 in the most optimist prediction (NASA LEGEND forecasts) for debris of size superior than 10 cm. However the reality is worst: although

the last years and the very short future did and will not see a lot of new launches due to the economic situation of the counties carrying the market of satellite and launchers, a study by the Northern Sky Research forecasts over 1600 new satellites built and launched by the year 2025 [1].

One overall question remains: what are those debris ? In the ideal situation where all satellites are de-orbited at the end of their lives, as specified by the regulations, instead of extending their mission until they can not be de-orbited, parts of the rocket stay in orbit, see figure 3. They are not only useless, but also very big and heavy.

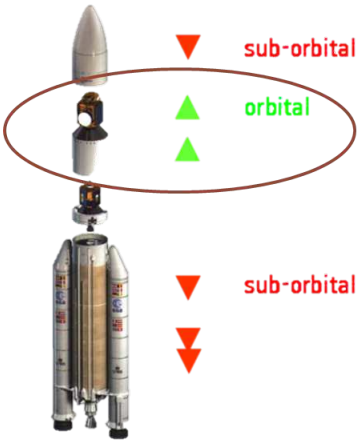


Figure 3: Parts of a rocket separated by post-deployment situation.

The scientific community is reviewing different solutions, and developing technologies to upgrade their Technology Readiness Level (TRL). As for now the most feasible solution is to get a chaser satellite in orbit, grab the debris and de-orbit it.

The Space Center at EPFL is putting together an Orbital Debris Removal (ODR) mission. This project is called Clean-mE, see reference [2].

A previous thesis [3] in the framework of this project, has determined a parametric MatLab/SIMULINK model for rendezvous and grasping manoeuvre simulation for the design of the Attitude Determination and Control Subsystem (ADCS). In this thesis, we are concerned with the grabbing part of the operation and the control of the spacecraft.

In a first part, we will review the context and requirements, defined by the most problematic debris, with the reasons they are the most problematic ones, and the space robotic state of the art and identification of the inherent problem of space robotics. Then we will move towards the remediation concept, with the solution proposed for the grabbing location and the scenario proposal for the ODR mission. Eventually we will get to the grabbing itself, how it has been intended to tackle it, how it has been implemented, and what results are obtained with this implementation.

2 Context and requirement

2.1 Debris

As debris remediation can be seen as Anti-SATellite (ASAT) activities, we can only retrieve objects from Switzerland or, for the partnerships that the final mission will need, our partners' orbital debris: let's consider the situation on the European scale. The first targets are non-operational, non-collided debris in Low Earth Orbit. One criteria has been worked out by ESA [4] to select priority targets, which is the mass times the probability of impact of an object. Within our list of European orbital debris, the Ariane upper stage (see figure 3 and 4 for the Ariane 4's) family have the highest priority.

Figure 4: H10 stage.

2.1.1 Why LEO ?

LEO contains 73% of the population of on-orbit objects, which corresponds to 40% of the total on-orbit mass [5] (GEO object are heavier). Also, the Kessler Syndrome mainly occurs in LEOs. Indeed, The present estimated number of collision fragments is around 500, whereas the predicted number of collision fragments in 2040 is about 2000. However those figures are for no more launches after 2007, thus without the FengYun ASAT test and the Iridium33/Cosmos2251 collision [6]. See figure 5.

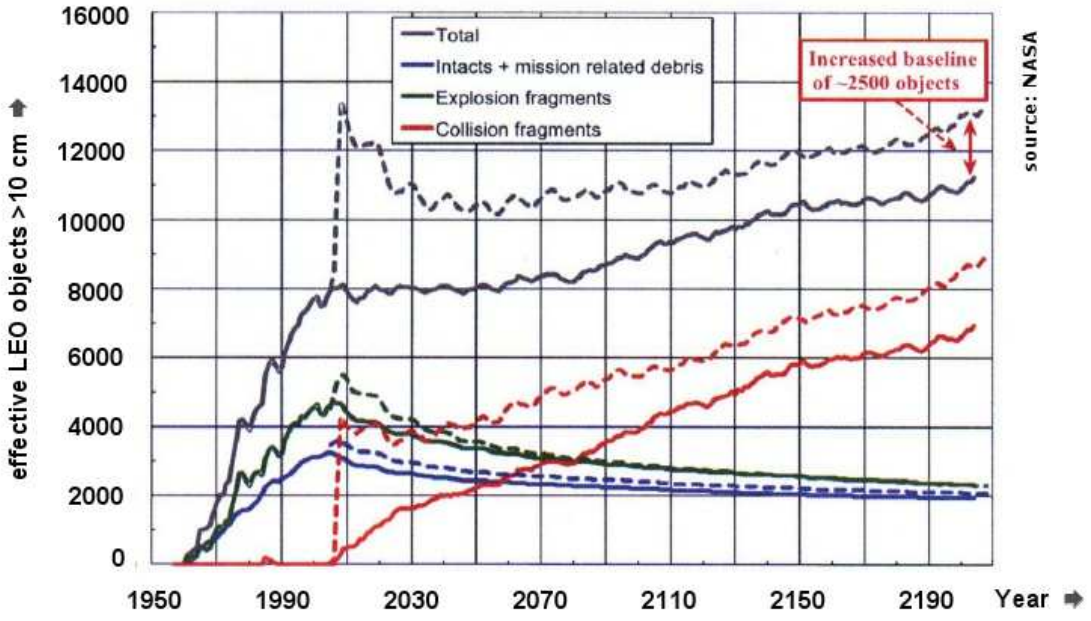


Figure 5: Long-term forecast of the effective number of orbital debris of 10cm or larger in LEO, doted lines are updates for the 2007 FengYun ASAT test and the 2009 Iridium33/Cosmos2251 collision. [5]

The prioritised orbits are [5]:

- $H = 900\,kms \pm 25$ & $i = 82^\circ \pm 1$: 300 tons of on-orbit debris;
- $H = 900\,kms \pm 25$ & $i = 98^\circ \pm 1$: 800 objects.

2.1.2 Why upper stages ?

The rockets' upper stages are the best choice for the Object Debris Retrieval (ODR) mission. Indeed, they represent 12% in quantity, or 25% in mass of all man-made objects in space, as well as not being of any use. The main concern of the international scientific community is LEO. Once again, those upper stages are the best choice, as they represent 61% of total LEO's objects mass.

Moreover, they have the highest mass times impact probability (directly related to area) product, which is the criteria to select optimal target for deorbiting, as Liou *et al.* showed [7].

One other reason is that upper-stages from Ariane launchers family are politically available for removal on a French(EADS)-European mission and upper-stages from Ariane launchers family are on the priority list of objects to be removed [4]. See figure 6 which displays the amount of upper stages left in LEO, as long as their ownership and their mass.

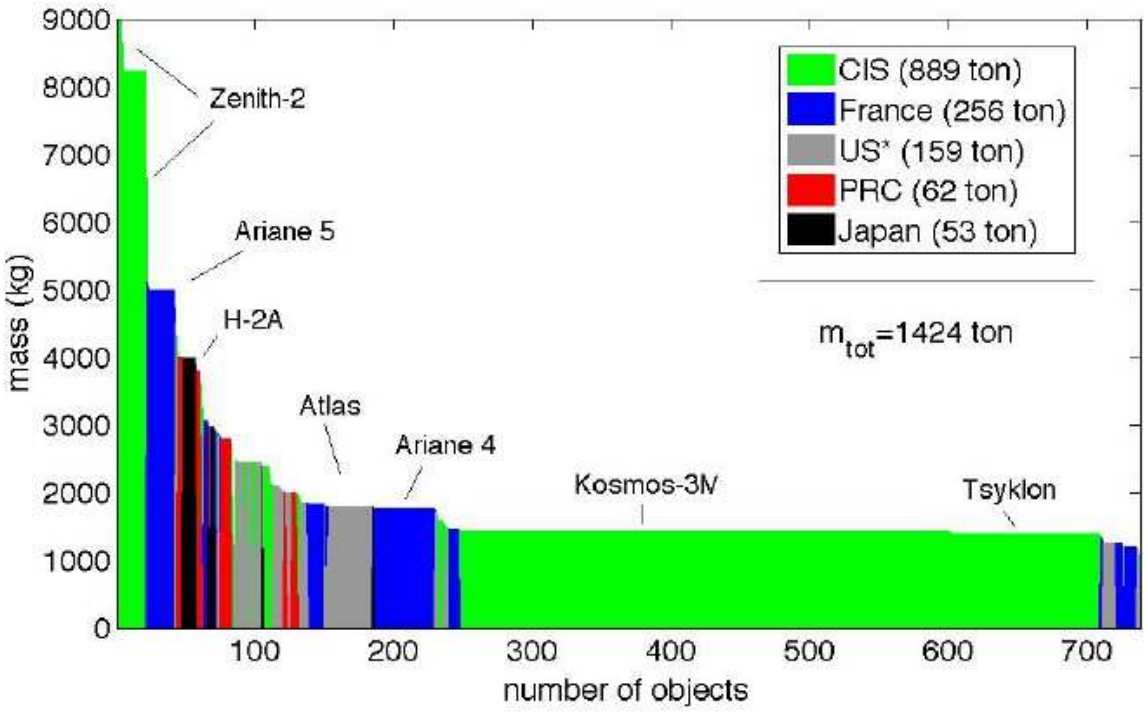


Figure 6: 1-ton and more upper stages in LEO sorted by quantities (x axis), masses (y axis), kinds (legend) and countries ownership. (CIS stands for Commonwealth of Independent States or in translated Russian: Sodruzhestvo Nezavisimykh Gosudarstv, SNG) Credits ETSIA/UPM [8].

Except for one (COSPAR ID: 86-19C), Ariane 4 and 5 upper stages are passivated. [4]. As they are passivated, there is no chance of them to explode from overheat of fuel left inside during re-entry phase, even if there still is some fuel inside all of them, see [9].

2.1.3 Targeted debris data

The upper stage rockets of the Ariane family are declined in several versions. In table 7 are displayed the envelope and the dry mass of the three first cases in term of quantity of on-orbit objects. More specifically, the targeted debris are upper stage from Ariane 4 launchers (H10). Within this family, the data of some individuals are displayed, figure 8 for their orbits, and table 1 for their identification, orbits and dynamical properties.

case	dimension	values
Ariane 4 (H10)	height × diameter	$10.2 \times 2.6\,m$
	dry mass	$1500\,kg$
Ariane 5 (EPS)	height × diameter	$3.4 \times 4\,m$
	dry mass	$1500\,kg$
Ariane 5 (ESC)	height × diameter	$4.7 \times 5.4\,m$
	dry mass	$4500\,kg$

Figure 7: Dimension of the envelope of the different upper stages cases considered as potential targets.

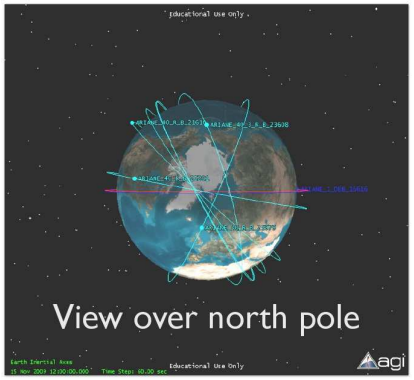


Figure 8: Typical orbits for spent H10 [10].

COSPAR Id	NORAD Id	Description	Mass [kg]	I_1 $[kg.m^2]$	I_2 $[kg.m^2]$	altitude [km]	incl [deg]
86-19C	16615	Ariane1 (H8)	1318	1114	8662	781-797	98.89
90-5H	20443	Ariane40(H10)	1764	1491	19918	764-778	98.67
91-50F	21610	Ariane40(H10)	1764	1491	19918	759-763	98.72
93-61H	22830	Ariane40(H10)	1764	1491	19918	782-798	98.66
95-21B	23561	Ariane40(H10+)	1764	1491	19918	766-775	98.52
98-17B	25261	Ariane40(H10III)	1764	1491	18663	782-788	98.25
2-21B	27422	Ariane42P (H10)	1820	1538	22974	788-805	98.45

Table 1: Estimated masses and inertias with their orbital elements of potential targets.

Unfortunately, Arianespace is not willing to share a complete geometrical definition of the upper stages (H10 at least) because those are still french patrimony, even if they are not used anymore.

2.2 Space robotics

De-orbiting a satellite using a robotic arm to grab it can be classified as an On-Orbit Servicing (OOS). Two excellent reviews are available for the interested reader: from the Canadian Space Agency, the first part of [11] ("Related Work" section) presents an near-extensive review of space-servicing missions. In the second part ("Phases of a typical OOS Mission" section) are presented the different phases of a space-servicing mission, which is adapted to our case in section 3.3. The other reference, Yoshida and Wilcox, reviewed the historical advances of space robotics in the first part of [12] and mathematical modelling in the third part. In this section, OOS missions and the problem of space robotic are addressed.

2.2.1 OOS missions

Five flown or planned missions were designed to demonstrate OOS capability. Launch in November 1997, The Japanese Engineering Test Satellite VII (ETS VII, or Kiku-7) [13] depicted on figure 9, demonstrated the validity of two mathematical modelling of a space manipulator, carrying out a variety of on-board experiments. It was composed of Hikoboshi (name of the prince in a classic Japanese tale) equipped with a 6-DOF robotic manipulator for Orbit Replaceable Unit (ORU) exchange and grabbing of the target satellite Orihime (princess), the second part of the mission.

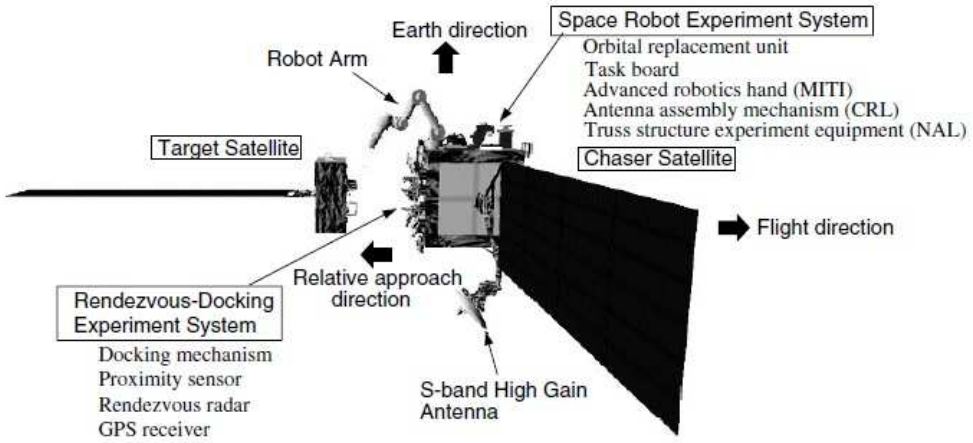


Figure 9: ETS VII.

Next was the NASA's DART, shown on figure 10 in its capture scenario, and Orbital Express, shown in orbit on figure 11, launched in April 2005 and March 2007, respectively. As explained in [14], DART collided with its intended target (MUBLCOM satellite) due to the miss of a waypoint acting as a transition in the entirely preprogrammed scenario.

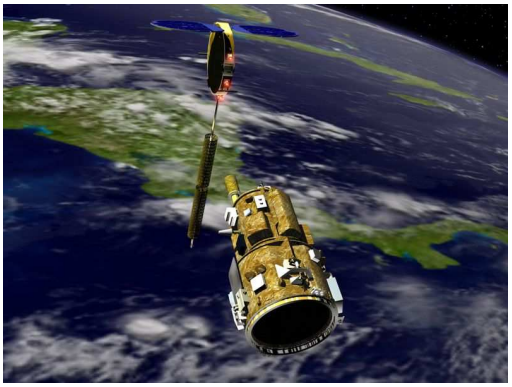


Figure 10: DART (below) closing on MUBLCOM. (artist's conception)

Figure 11: Artist conception of the assembled Orbital Express.

Orbital Express (see a self photo on figure 12) is composed of ASTRO, the chaser and NEXTSat, the target prefiguring OOS-enabled future satellites. After servicing tasks without the rendezvous/grabbing typical of OOS missions (see figure 13), ASTRO demonstrated OOS by an autonomous rendezvous and capture phase [15]. Three relevant scenarios are described here: in the first scenario (scenario 2-1) the exercise was entirely autonomous, around 2h of unmated operations at a maximum distance of 12m; in a second scenario (scenario 8-1), the exercise was almost autonomous with little help from ground, and lasted around 1 day at 7km apart. The last exercise was a recovery from exercise 3-1. It resulted in the longest apart exercise, with almost 8days and 6km apart.

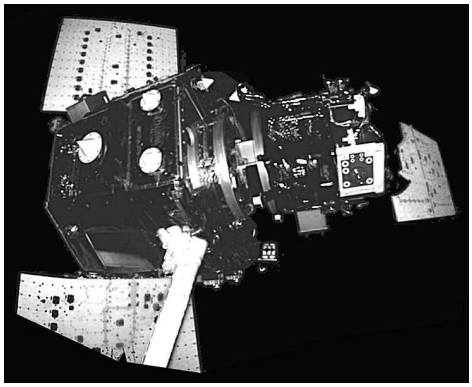


Figure 12: Orbital Express: ASTRO taking a picture of itself using the camera of the robotic arm.

Figure 13: Orbital Express: ASTRO servicing NEXTSat (units replacement, no grabbing). (artist's conception)

In Europe, there are two missions designed to demonstrate OOS capability: the Swedish PRISMA [16] shown in figure 14 and the German TECSAS, redefined in the DEOS mission (see [17] for updated information about the program), depicted in figure 15. The DEOS mission is still in design process, but the objectives are OOS demonstrations and satellite de-orbiting. DEOS will rely on an ultra-light, efficient and powerful robotic arm to accomplish its mission, specially developed by the DLR for such a mission and already space qualified (some details about it in [4]).

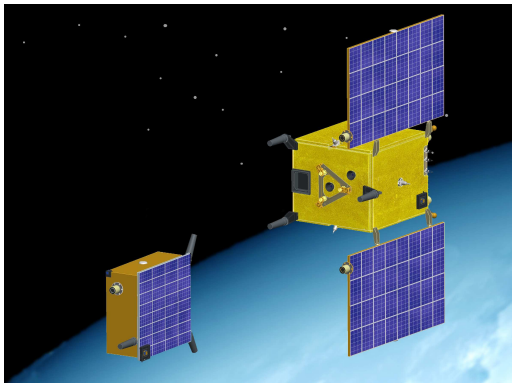


Figure 14: PRISMA, formation flying of Mango (main) and Tango.

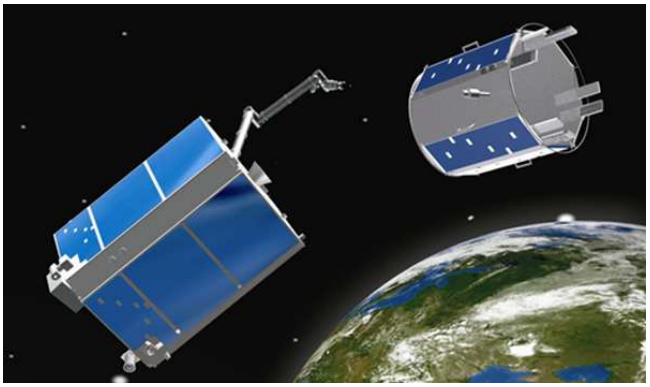


Figure 15: DEOS: Capture of a tumbling satellite. (artist's conception)

PRISMA is composed of two satellites, Mango being the chaser and Tango the target satellite. They demonstrated formation flying and rendezvous based on different sensors (notably, GPS), as long with subsidiary technologies (such as a new propellant (HPGP), micropropulsion, and enhancements of common satellite subsystems). The mission is still on-going. Notably, they also demonstrated autonomous detection of Tango through vision sensors. See figures 16 and 17 for extremely different luminosity situations in space. Those two pictures of Tango have been taken by Mango while detecting it.

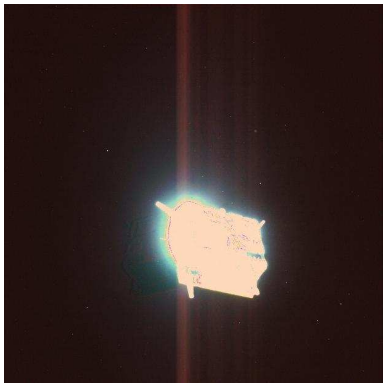


Figure 16: PRISMA, detection of Tango by Mango, highly illuminated.

Figure 17: PRISMA, detection of Tango by Mango, on a Earth background.

As the scientific community grows more and more concerned with the problem of space debris, it looks for a remediation concept. An example of that trend was the first European Workshop on Active Debris Removal, organised in the headquarters of the CNES in Paris on the June 22nd 2010. Few ideas and studies were presented, and in particular, few European universities presented de-orbiting mission studies based on current technologies. One noticeable was Cranfield University (United Kingdom). Their project, Debris Removal in LEO (DR LEO, [10]) led to the Yuki concept, figure 18. This concept is also one chaser-one big debris, but to cut the cost, 5 chasers are brought to orbit in one launch.

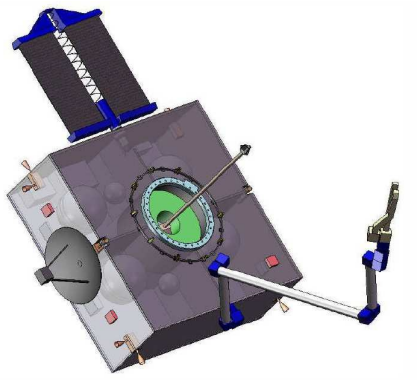


Figure 18: Yuki concept (CAD drawing by Ratcliffe, 2010), Cranfield University.

2.2.2 Space robotics' problem

There is one major problem for robotics in orbit: there is no fixed base! The problem is depicted on the following schematic (fig 19). This figure shows the trivial example of a 1-

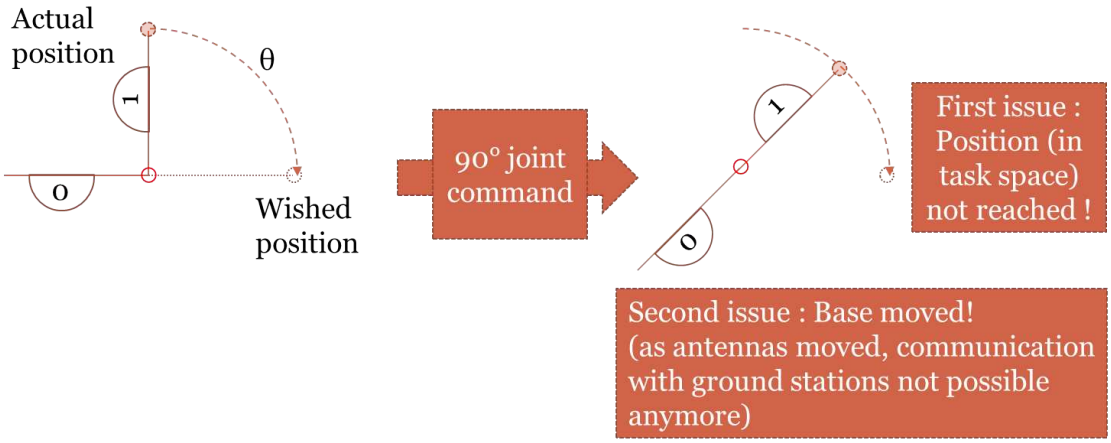


Figure 19: 1 DOF robot in space: simple depict of the problem, both link have the same inertia.

DOF space robot, trying to reach a goal depicted by the light circle on its right. On Earth, the manipulator would reach it by a 90° joint command, however in space there is no force compensating the reaction of the base. In this one problem, there are two issues: first, the end-effector did not reach the desired position, second, the base moved and the user is no longer able to communicate with the robot-satellite.

Of course this problem is not new and robotic systems have been used in space for a few decades now. However in most systems the base was way much more massive than the manipulated object (even the in the Shuttle (RMS)).

Lindberg, Longman, and Zedd (in the first article of [18]) addressed issues related to dy-

namics and kinematics of a space robot, which control depends on dynamics parameters as well as motion history. Longman, in the second paper of [18], showed how the forward and inverse kinematic problem become much more complicated when the mass of the load is not negligible compared to the one of the base. He also propose a scenario for attitude control of the base using the robotic arm instead of jet thrusters. This scenario is recycled in step 4 of the presented scenario section 3.3. One solution is developed for a point mass load problem. Vafa and Dubowsky developed the concept of Virtual Manipulator (see the eponymous paper in [18]), which is a kinematic representation of the real system as a fixed manipulator, by considering the immobility the center of mass of the entire system. Such concept begins to allow the use of control methods for fixed manipulators. Yoshida and Umetani present the concept of Generalized Jacobian Matrix (GJM), and two control methods based on this concept in [19]. This concept eliminates 6 degrees of freedom of the base to focus on the end-effector control. The GJM is an enhanced Jacobian, taking the reaction base motion in account. This allows to use control methods using Jacobian matrix, developed for fixed earth-based manipulators. In the original definition given in [19], it is assumed that no external forces nor torques were applied on the system, however this assumption is not necessary for the definition of the GJM, see reference [12]. Along with the GJM is defined the Generalized Inertia Matrix for space manipulators (GIM). The same authors, joined by Nenchev, worked out the Reaction Null Space (RNS) [20] formulation based on the opposite approach: this time the goal is to not disturb the spacecraft attitude while moving the end-effector point. This is achieved by specific trajectories. The validity and effectiveness of the last two formulations has been demonstrated in orbit, on the ETS VII mission [13].

This list of mathematical modelling is far away from being extensive. However the usefulness of the GJM has been demonstrated. This formulation is quite portable as it is compliant with ground-based manipulator controls, and it allows non-zero external momentum and forces - useful when the user of the robotic-armed chaser wants to use reaction wheels, at least to compensate the gravity torque.

3 Remediation Concept

3.1 Problem statement

The solution to the Kessler Syndrome is an Active Debris Removal mission, *ie* to get there, and take down the debris that would not come down by themselves (within a specified period of time). The scientific community is recommending that 5-15 large ($>$ 10 cm) debris objects be removed per year.

To do that, we need to launch a spacecraft, equipped with a subsystem to handle debris, and with a Attitude Determination and Control Subsystem (ADCS), able to withstand the needs of determination of the debris-handling subsystem, and the needs of control accuracy not only of this subsystem but also for the re-entry, carrying the debris. Previous studies have been made both on the handling subsystem [21] and on the ADCS that can be used for control of the spacecraft during approach, rendezvous, grabbing and de-orbiting/re-orbiting phases. The DR LEO concept [10] seemed adapted to the small mission, one big object retrieval demonstrator planned here at the Space Center (EPFL). The ADCS parameters used in simulation are those

of the study [22], which is widely based on the ATV architecture. The parameters worked out will be used as a basis for the following (as a first loop design). The reader should consult the requirement list in appendix A.

Based on the study [21] and [4], and the requirements list in appendix A, the preselected manipulator is the DLR robotic arm LWR III.

3.2 Location of the grabbing on the target by the manipulator

The location of the grabbing on the target is quite delicate. The first thing to notice is that when the thruster(s) for the de-orbiting phase are fired, the plume should not be deviated by the target, so the chaser should 'push' and not 'pull' the target, for an optimum use of the fuel. The direction of the thrust also should goes by the center of mass of the target for the same reason.

A good location has to allow a good final positioning of the chaser and its thruster(s) onto the target. There are mainly three different locations that could be used as a handle by a robotic manipulator (Ariane 4's third stage H10 case, see figure 21):

- The safer and more 'portable' choice would be the adaptor between the third stage and the satellite(s), as all third stage have almost the same connector's geometry, for commercial reasons. However, this adaptor is very distant to the center of mass of the stage, and thus either require a very long robotic arm, or an intensive usage of fuel in order for the chaser to track this adaptor.

- The nozzle of the stage is much more closer to the center of mass point and thus require as shorter arm. However in the case of the H10 stage for example, this distance is still quite prohibitive: the end of the nozzle is at $4\,m$ of the center of mass point, leading at least to a four meters long arm. Moreover, as the chaser should push the target, there is a big issue: as soon as the engine stops its burn, the pressure in the actuators maintaining the position and orientation of the nozzle, with reference to the tanks of the stage, is released. Hence pushing on the nozzle does not imply a pushing direction through the center of mass. Even more, after 725 to 780 (H10 to H10-3) seconds of burn, the nozzle is highly heated, and then heavily cooled as it is into the cold of space. Because of that, we can have absolutely no certainty about the shape of the nozzle, nor its state. To get an idea of the heating phase of the nozzle, see figure 20.

- A rod connecting the re-pressure tanks to the fuels tanks (see figure 21). Each rocket stage needs a re-pressure tank (generally of helium) as it uses its fuel. Those re-pressure tank(s) are generally beneath the tanks and above the motor, making the rod(s) real close to the center of mass of the stage. The re-pressure and fuel tanks are linked by rods that are designed and tested to handle the acceleration of several g of the launch, with the full mass of the fuels tanks weighting on it.

  The Ariane 4 rocket had to handle $4,5g$ at first stage cut-off. From the tank configuration of the H10 in [9], the mass supported by the rod is of the re-pressure tank only. From the tank configuration, we can estimate the diameter of the helium tank, about $650\,mm$; and we know the helium stocking temperature and pressure, from the Ariane4 user's

manual [23], to be $100\,K$ and $200\,bar$. Knowing those two parameters we get the density of helium[1]: $74,76\,kg/m^3$. A quick estimate of the helium's mass, assuming a perfect sphere shaped tank of volume $4/3.\pi.r^2$ is $10,75\,kg$. An estimate of the tank's mass, assuming a steel of a medium density of $7800\,kg/m^3$ for a spherical tank of volume $4.\pi.r^2.e$, with a $10\,mm$ thickness e, is $414,12\,kg$. The sum of those masses leads to a force handled by the rod slightly under $19\,kN$. According to this figure, we can be quite sure that this rod will handle the forces that a grasping process and manipulation can generate.

Figure 20: Heated nozzle of a third stage, cooling down.

3.3 Scenario

The goal of the ADCS + grabbing subsystem design is to reduce the use of the ADCS during the grabbing phase in order to reduce the consumption of fuels. The optimal reduction is to not use it at all. That is satisfied by the scenario of the mission as it follows (see figure 23):

1. Launch, deployments, orbit positioning (circular orbit below the target).
2. Analysis of the target (as many orbits as necessary, until the target is well identified, as well as its movements).
3. Rendezvous (closing gap between target and chaser: the chaser approach closer than one meter the target moving zone (see figure 21: the big black arrow is the axis of rotation of the target, its origin is the center of mass. The target pictured is the H10 upper stage (Ariane 4), our main targets. The wide-spaced dotted frame depict the window in which the target moves, and that the chaser will avoid, the norrow-spaced dotted frame is the same window but for the x angle to 0 degrees (worst case)).
4. Emptying momentum of the reaction wheels (no perturbing reactions during catching phase that can be avoided! This phase can be done with the arm to save some fuel, see Longman's scenario for base attitude control in [18]. According to projections from the Yuki's scenario [22], the emptying of reaction wheels momentum is the task that use most of the on-board fuel).

[1] Thanks to WolframAlpha

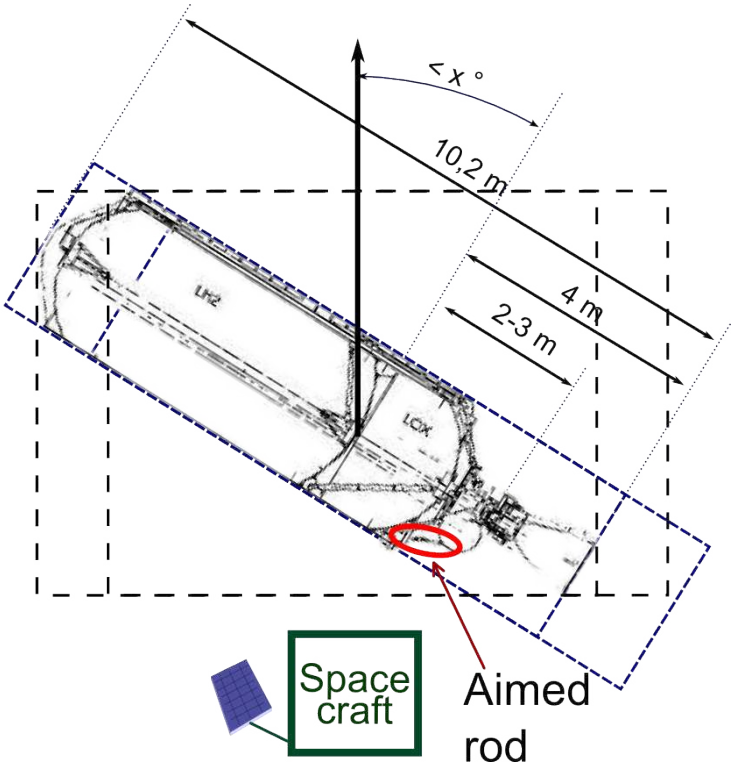


Figure 21: Approach and positioning of the chaser with reference to the target.

5. Grabbing (see section 4): the chaser is placed within the axis of rotation of the target, and from there the arm follow the point aimed for the grabbing while the target rotate (see figure 21). This point aimed at for the grabbing phase is the middle of the exposed connecting rod between the LOX tank and the re-pressure tank.

6. Autonomous phase:

   (a) (passive sub-phase) Target grabbed, chaser adopts the movement of the target (inertia of the target way over the chaser's).

   (b) Reversing the kinematics, the chaser places itself onto the target, in a location that will allow it to control the trajectory, communicate with ground stations and have an efficient thrust for the group formed by itself and the target (see figure 22).

   The favoured scenario is the nozzle atop configuration. In this configuration, the alignment does not rely on the tanks-nozzle alignment because of two problems. First, as soon as the rocket's thrust is shut down, this alignment is disabled[2]. Second, after the burn, the geometrical definition of the nozzle is no longer known. Hence, the idea is to control the alignment *via* the robotic arm, still grabbing the rod, while the main thrust goes by the nozzle.

[2]source: Chistophe Bonnal, chef de projets senior, Direction des Lanceurs du CNES

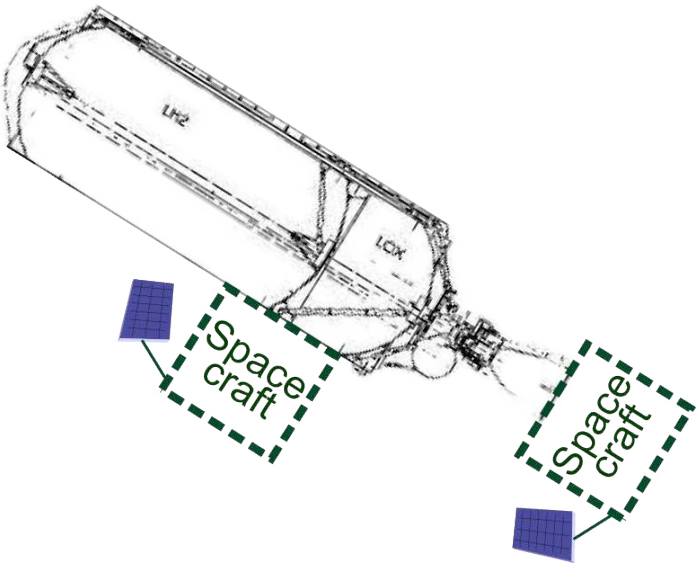


Figure 22: Detail of step 6b : two possible positions of the chaser onto the target. The reader's attention is directed towards the nozzle position: in this configuration, the chaser is aligned with the center of mass of the target by the manipulator still connected to the rod, it will produce an efficient thrust.

(c) Recovering ground station liaison after/while stabilizing the target. If the position onto the target of the spacecraft is the top or rear of the stage, it may put the both of them into stabilizing spin along the axis of revolution of the target.

7. Decelerating thrust to get on a correct re-entry path.

8. controlled re-entry.

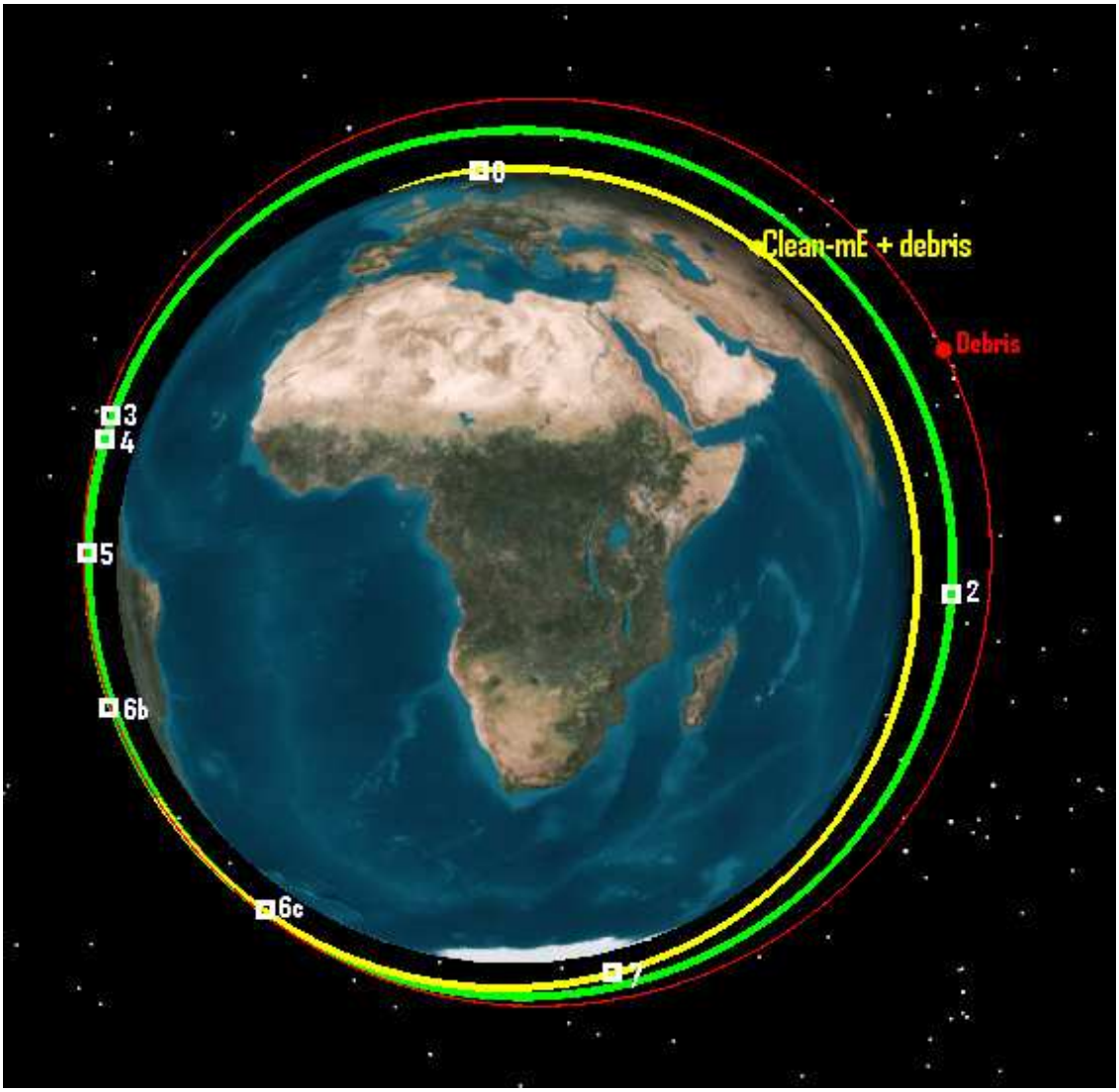


Figure 23: Deorbit scenario.

4 Grabbing

4.1 End effector positioning problem

On ETS-VII [13] were tested three different control strategies: conventional manipulation (meaning the position and attitude of the hand are commanded in the satellite-fixed coordinate frame, while the attitude of the base should be maintained by reaction wheels: compensation of the lack of static inertia of the base (like on Earth) by dynamic inertia), Generalized Jacobian Matrix (GJM) [19] and Reaction Null Space (RNS) [18].

Two exercises put the differences and utilities of the different formulations in evidence: in exercise A the trajectory asked to the end-effector was a straight line in the inertial frame, and

the exercise B was a two-phase approach of an initial $400\,mm$ distant target, drifting with a velocity of $10\,mm/s$. The target constrain was to keep the target in the range of the camera. Those two phases consisted of a first rough approach, and then switching to a more exact dynamic of motion when reaching either $100\,mm$ distance to the target or, for the conventional approach as it was clear it would not reach the arbitrary limit, a time limit. The data of this latest experiment is shown for two cases: first, conventional with base compensation figure 24; second with a RNS approach switched at $100\,mm$ from the target for a GJM final approach and capture on figure 25.

It turns out that the base compensation only is neither accurate, neither does it keep the base at a constant attitude: for the exercise A, the hand had one degree of error in pitch, whereas for exercise B, the base attitude was disturbed more than one degree. The RNS is quite good for constant base attitude, as it is designed to, but not that good at end effector accuracy, while the GJM does not keep the attitude of the base constant, but is really good for the end effector accuracy: exercise A resulted in less than 0.2° in pitch error.

Those values may seem quite low but one has to keep in mind that the manipulator length is $2\,m$ and it operated at less than half a meter, furthermore the base was quite heavy ($2.5\,T$, order of magnitude of the inertia ratio between base and links is about 10^3). One other fact stood out of the experiments: the gravity gradient was not negligible, opposed to the usual assumption when using a manipulator, for its reaction is dominant over a short-time range.

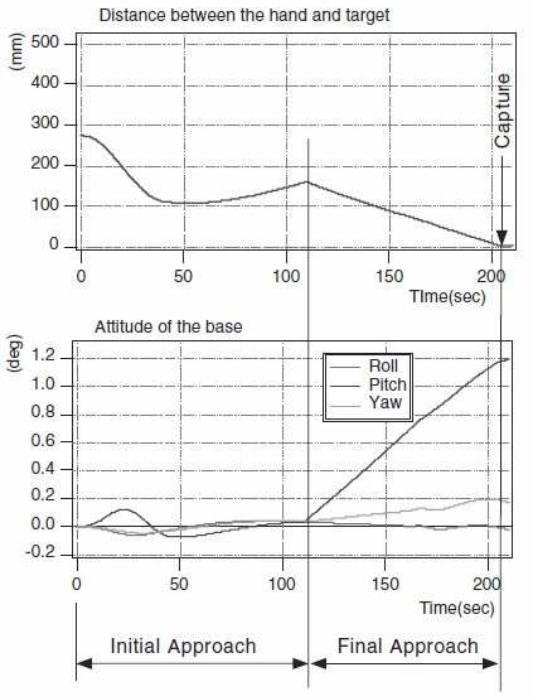


Figure 24: ETS VII: autonomous target capture: conventional satellite-frame fixed trajectories, with base attitude compensation with reaction wheels.

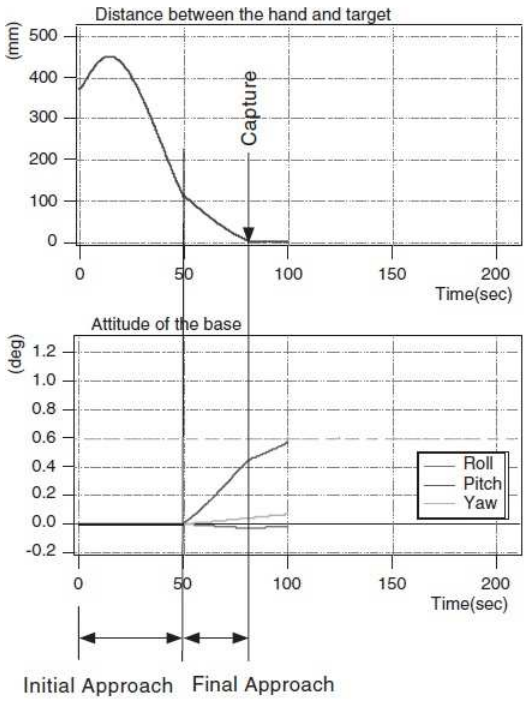


Figure 25: ETS VII: autonomous target capture: RNS-based reactionless approach switched to GJM-based inertial manipulation when closer to target than 100 mm.

The duration of the project allowed me to implement only one of the two strategies, and as the GJM was fundamental to effectively grab the target (final approach phase, figure 25), I chose to implement it instead of the RNS. Moreover one could project a truly autonomous mission, where constant attitude of the base would no longer matter, as the spacecraft is no longer controlled by ground stations - it would control itself with reference to the target. Furthermore, thanks to Generalized Jacobian and Inertia Matrices, base attitude reaction can be estimated beforehand and a feed-forward command can be generated.

4.2 Implementation

4.2.1 Structure and user's guide

I used RobotToolKit [24] (RTK) for the implementation in C++. RobotToolKit is an open-source robot simulator developed for researchers and robot hackers under GNU licence. It offers a C++ environment for the control of a robot, with a set of mathematical and robot-interacting classes.

I went for the implementation of the Resolved Acceleration Control method together with the use of Generalized Jacobian Matrix (GJM) [19] for the control of the robot, with an error feedback loop through the video feedback, as the reader can see on the schematic on figure 26. In the simulation, the video feedback is replaced by the actual position and orientation of the end-effector, specified as a `Pu`-named link in the structure of the robot, and the position and orientation of the target.

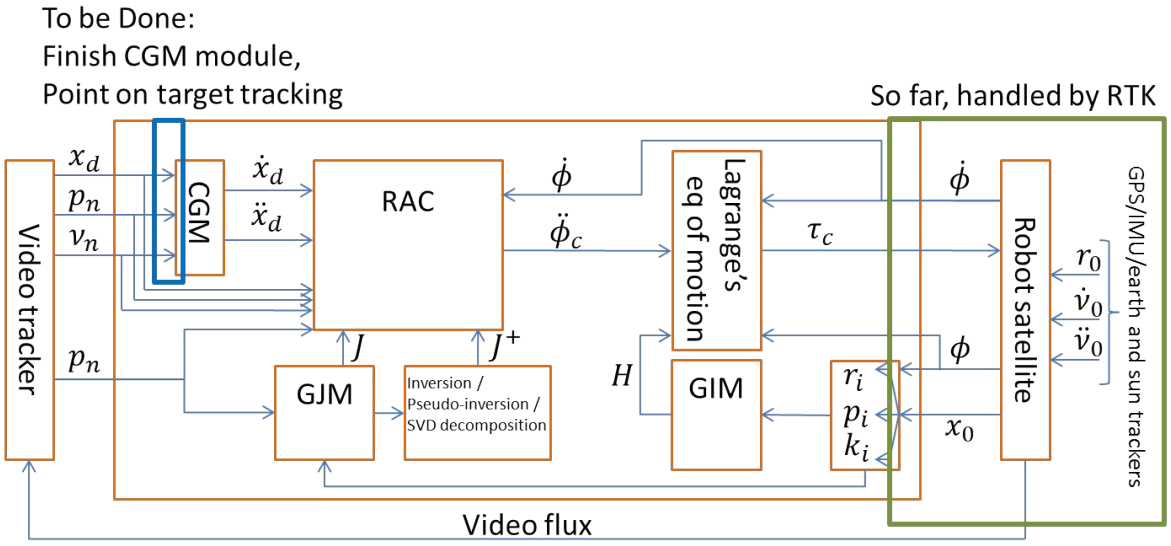


Figure 26: Implementation schematic. The bold blue frame is what is left to do, explicitly the tracking of the target.

Each of the boxes are implemented methods in the C++ class `generalizedjacobianmethod`, which is constructed with the robot used in input, or `CMGv0` for the CMG box:

- box "r_i, p_i, k_i" is decomposed in two methods, `GJMupdate`, collecting for each link the position of the center of mass r_i, of the center of the joint p_i, and the axis of rotation of the joint k_i and updating them. Then the method `submatrices` computes the submatrices necessary to the definition of the GJM and GIM, and update them.

- box "GJM" correspond to the `jacobian` method which computes the GJM, updating it and giving it back.

- box "Inversion/Pseudo-inversion/SVD decomposition" is decomposed in many methods, corresponding to the different algorithms used for the kinematic inversion problem: on one hand we have two methods working together: `pseudoInverse`, fed with the matrix to inverse and the resulting matrix, used to separate the case of a square matrix, broad or wide rectangular matrix. Then the matrix is send to the `DegeneratedSVD` which begins by computing the rank of the "squared matrix" (in fact, of the matrix times the transpose, or the transpose times the matrix depending on the case, see section 4.2.3 first and second paragraphs), and then computes a degenerated SVD inverse (see third paragraph of section 4.2.3) if the matrix is not of full rank, or simply send back an information of a non necessary degenerated SVD inverse computation so the first method can deal with less computationally expensive inverse methods (pseudo-inverse or simple inversion as implemented in RTK). On a second hand were implemented the Damped Least Squares algorithm in the `DLS` method. This is fed with the matrix to inverse, the matrix resulting of the inversion, and a damping constant λ. This method is computationaly cheap as it does not any SVD decomposition. On a third hand were implemented the Selective Damped Least Squares in the `SDLS` method. This is fed with the matrix to inverse and the matrix resulting of the inversion. This method requires an SVD decomposition at each step. On a last hand were implemented an intermediate method, the `MixedDLS` method, computing an SVD decomposition each dozen or hundred of steps and DLS the rest of the time using as damping constant the previously determined damping constant. More information about those three algorithms and the reason of the redundant implementation in section 4.2.3.

- box RAC is in fact two methods, but the first one, `update` is transparent as its only use is to deal with the quantity of informations we have on the trajectory. When called with the integer 1 specifying the use of the RAC method and the given vectors among the end-effector position, attitude, speed, acceleration, rotational speed, and rotational acceleration (in this order), it transmits the vectors to the `RAC` method. Then the `RAC` method compute the command acceleration of each joint, calling the update of the base vectors, submatrices, and Jacobian, as well as the inversing method. *Still in debugging.*

- RMRC is a simplification of the RAC method. In the same way it is composed of two methods: the first one, `update` being transparent as its only use is to deal with the quantity of informations we have on the trajectory. When called with the integer 2 specifying the use of the RMRC method and the given vectors among the end-effector position and attitude (in this order), it transmits the vectors to the `RMRC` method. If the attitude vector is not specified, the initial attitude is kept. Then the `RMRC` method compute the command position of each joint, calling the update of the base vectors, submatrices, and Jacobian, as well as the inversing method.

- box "GIM", implemented as the `inertiaTensor` method, computes the GIM, updating it and giving it back.
- box "Lagrange's equation of motion", implemented as `lagrangeForGJM`, computes the command torque of each joint, using the updated joint accelerations. *Still in debugging.*
- box "CGM" is the implementation of the first version of this command generator module (see section 4.2.7) in the class `CMGv0`. It needs an initialization through the constructor (which need two boolean, specifying if the user wishes null initial and final speeds and accelerations), and then the `calcul_traj` method where the user should give the final position, attitude, the initial/current position, attitude and last, if he wants, the duration of the run to reach the destination, else one is automatically calculated depending on the length between initial and final positions.

  The user can thus get the Trajectory, Speed, Acceleration and Attitude of the hand at the time t through the accessors `GetHand`[...]. The user can also use a dummy trajectory being a circle in the XY, YZ or ZX planes for test purposes by the `CMGv0DummyTraj` method, given the step (integer related to current instant t), the current position, the base position and the plane, as a two character string (`'XY'`, `'XZ'` or `'YZ'`, not regarding on the order of the letters).
- alternatively, the box "CGM" can be replaced by an external list of points, with or without the associated speeds, acceleration and attitude (given as 3 euler angles vector or as a 4 components quaternion). This list is fetched by the `Load` method of the first class. The user wishing to use this method to enter the commands should put the list named `CleanMeCommands.xml` in the `RTK/data/Misc` folder, formatted as an XML file such as the short example 1.

Accessors are available to access the matrices and vectors, and are quite straightforward:

- `GetCurrentHandPosition`
- `GetCurrentBasePosition`
- `GetCurrentHandAttitude`
- `GetJacobian`
- `GetInertiaTensor`
- `GetRAC`
- `GetRMRC`
- `GetTorques`

```
<Commands>
   <trajectory>
      <point> 4.610000 2.510000 0.282000 </point>
                        ...
      <point> 1.000000 3.000000 1.000000 </point>
   </trajectory>
   <speed>
      <point> 0.000000 0.000000 0.000000 </point>
                        ...
      <point> -0.000152 -0.000304 0.000000 </point>
   </speed>
   <acceleration>
      <point> 0.000000 0.000000 0.000000 </point>
                        ...
      <point> 0.303378 0.606757 -0.000139 </point>
   </acceleration>
   <rotation>
      <point> 0.000000 0.000000 -1.000000 0.000000 </point>
                        ...
      <point> 0.000000 -0.999999 -0.001571 0.000000 </point>
   </rotation>
</Commands>
```

Code fragment 1: Example of the formatting of an external list of commands. Only the "trajectory" points are necessary. The number of points should correspond between the different inputs, and should be selected based on the length of the trajectory and the time step chosen in the simulator (config.xml file). This file was generated with MatLab.

4.2.2 Generalized Jacobian Matrix

This enhanced Jacobian matrix depends only on the joint positions and on the base attitude. See its definition in appendix B.

Below are presented the useful vectors and matrices used thereafter:

- $\underline{\phi} = {}^t(\phi_1, \phi_2, \ldots, \phi_n)$ the joints positions vector,
- $\underline{X} = \begin{bmatrix} \underline{p_r} \\ \underline{\theta_n} \end{bmatrix}$ the end-effector position vector in task space in inertial frame,
- $\underline{X_b}$ the base position vector in inertial frame,
- $\underline{\tau} = {}^t(\tau_1, \ldots, \tau_n)$ the vector of the torques,
- $\underline{\underline{J^*}}$ the Generalized Jacobian Matrix for Space Manipulators [13] (GJM) (see below),
- $\underline{\underline{H^*}}$, the Generalized Inertia Matrix for Space Free-Flying Manipulators [13] (GIM).

The velocity of the manipulator hand in the inertial frame is expressed as:

$$\dot{X} = J_m.\dot{\phi} + J_s.\dot{X}_b \tag{1}$$

On Earth we would have $\dot{X}_b = 0$ but not in space. To be able to use the Jacobian-based algorithms, we have to get rid of the second term.

Now, there is no external forces applied on the spacecraft, hence the momenta of the system remains constant, and we have:

$$\begin{bmatrix} \mathsf{P} \\ \mathsf{L} \end{bmatrix} = I_s.\dot{X}_b + I_m.\dot{\phi} \tag{2}$$

Assuming nil initial momenta, equation 2 express $\dot{x_b}$ as a function of $\dot{\phi}$ and allows to get rid of it in 1 equation, and defining the generalized Jacobian:

$$\underline{\dot{X}} = \underline{\underline{J^*}}.\underline{\dot{\phi}} \tag{3}$$

With J^* being:

$$\underline{\underline{J^*}} = \left[\underline{\underline{\widehat{J_m}}} - \underline{\underline{\widehat{J_s}}}.\underline{\underline{I_s}}^{-1}.\underline{\underline{I_m}}\right] \tag{4}$$

In the process of establishing the generalized jacobian, from equation 2 and using the angular momentum part, we got:

$$I_s.\dot{\omega_b} + I_m.\dot{\phi} = \mathsf{L} = 0 \quad \Leftrightarrow \quad \dot{\omega_b} = -I_s^{-1}.I_m.\dot{\phi} \tag{5}$$

The equation 5 gives the attitude of the base resulting from the movement of the manipulator, allowing a correction of this attitude before the actual manipulation in order to keep contact with the satellite.

4.2.3 Matrix inversion

There are several matrices to inverse, some are square, but most are not and in both cases some are not of full rank. Indeed, in the general case, a robot has not exactly 6 DOF. In the case considered, it should have 7 (as we consider the DLR manipulator). This implies a non-square Jacobian, and thus a non-invertible one. Moreover, some singular cases may diminish the rank of the matrix. In the case of full-rank square matrix, I use the already implemented in RTK [24] Inverse method. In the case of full rank but non-square matrix $\underline{\underline{M}} \in M_{m,n}(\mathbb{R})$ with $m < n$, I will use the pseudo-inverse for broad matrices, noted M^+:

$$\underline{\underline{M^+}} = \underline{\underline{{}^tM}}.\left(\underline{\underline{M}}.\underline{\underline{{}^tM}}\right)^{-1} \tag{6}$$

In the case of full rank but non-square matrix $\underline{\underline{M}} \in M_{m,n}(\mathbb{R})$ with $m > n$, I will use the pseudo-inverse for tall matrices, noted M^+:

$$\underline{\underline{M^+}} = \left(\underline{\underline{{}^tM}}.\underline{\underline{M}}\right)^{-1}.\underline{\underline{{}^tM}} \tag{7}$$

Finally, the non-full-rank case occurring at singularities. In this case I first used singular values decomposition. Thus for $\underline{\underline{M}} \in M_{m,n}(\mathbb{R})$ with $r < \min(m,n)$ the rank of $\underline{\underline{M}}$:

$$\underline{\underline{M^+}} = \underline{\underline{V_r}}.\underline{\underline{D_r}}^{-1}.\underline{\underline{{}^tU_r}} \tag{8}$$

Then in a first approach, I deleted near-zero singular values and their left and right singular vectors (Generalized Singular Values Decomposition [25]), in a second, instead of deleting the singular values, set their 'inverse' to 0.

This second algorithm was much more robust to singularities than the previous ones, but not enough. Indeed the stability of the inversing algorithm should be very high because we try to control 9 degrees of freedom (3 for the hand position, 3 for the hand attitude and 3 for the base attitude. The base attitude is not strictly speaking controlled, but as to stay within small motion range, which forbids high dynamics on the arm which would led to big movements of the base).

In 2009 Samuel Buss made a review of algorithms to solve the Inverse Kinematics problem [26], on which is based the following. The SDLS algorithm is based on a previous work of the same author [27].

The next method implemented was the Damped Least Squares method which minimises $||J.\Delta\theta - \Delta e|| + \lambda^2.||\Delta\theta||$ instead of just the first part. In words, it means it minimises the tracking error and the joint velocities instead of simply the tracking error without any regards for the joint velocities.

The problem with this algorithm is double: the damping constant λ has to be tuned up for the application, and is identical for all joints. The SDLS algorithm do a SVD decomposition and adjusts the damping factor separately for each singular value of the Jacobian based on the difficulty of reaching the target positions.

A last implementation is a trade-off between those last two: called MixedDLS, it does a SDLS inversion and stores the damping factors. Then it uses those stored values for DLS inversion in the next hundred or thousand steps, and then re-do an SDLS inversion before the stored values are not adequate anymore. As in this gap between two SDLS inversion may occurs a singular position, the damping factors determined have a minimal value, very low, but not null.

In our application, the DLS is preferred as the tune up of the damping value can be done before hand.

It should be noted that some of the inverted matrices in the generalized jacobian algorithm are intricately invertible as $\underline{\underline{I_s}}$ and $\underline{\underline{H_{b\omega}}}$ are (as proved in 4.2.6).

4.2.4 Control algorithm

Resolved Motion Rate Control:

$$\begin{aligned} \dot{X} = J^*(\Omega_0, \phi)\,.\dot{\phi} \quad &\Rightarrow \quad X_{next} - X_{previous} = J^*.(\phi_{next} - \phi_{previous}) \\ &\Rightarrow \quad \phi_{next} = \phi_{previous} + J^{*-1}.(X_{next} - X_{previous}) \end{aligned} \tag{9}$$

This control needs only the inversion of the Jacobian, but needs a controller to ensure the following of the consign.

Resolved Acceleration Control (RAC), feedback loop: As described in [19], in order to be able to control the dynamics of the robot, we differentiate the relation between the workspace position and the jointspace position - the d subscript indicating wished or calculated positions and variables.

$$\begin{aligned} \dot{X} = J^*(\Omega_0, \phi).\dot{\phi} \quad &\Rightarrow \quad \ddot{X}_d = \dot{J}^*.\dot{\phi}_d + J^*.\ddot{\phi}_d \\ &\Rightarrow \quad \ddot{\phi}_d = J^{*+}.\left(\ddot{X}_d - \dot{J}^*.\dot{\phi}_d\right) \end{aligned} \tag{10}$$

Furthermore, this formulation allow a feedback from the error in position and in velocity.

$$\underline{\ddot{\phi}_d} = \underline{\underline{J^{*+}}}.\left[\underline{\ddot{X}_d} + \underline{\underline{K_1}}.\left(\underline{\dot{X}_d} - \underline{\dot{X}}\right) + \underline{\underline{K_2}}.\left(\underline{X_d} - \underline{X}\right) - \underline{\underline{\dot{J}^*}}.\underline{\dot{\phi}_d}\right] \tag{11}$$

There is the matter of the gain matrices K_1 and K_2 that are to be tuned.

Attitude Control: The error function to be minimised used is:

$$e = 1/2 * [i \wedge i_d + j \wedge j_d + k \wedge k_d] = n.\sin(\alpha) \tag{12}$$

This function was determined by Luh *et al.*, [28] and is used since then, particularly for RAC.

4.2.5 Torques: equation of motion

However, the motors of the arm are controlled by torques. From the RAC, we get the accelerations that we should have on the motor to get it to described the desired motion. To get torques from accelerations, we use an equation of motion derived from a Lagrange function [19].

$$\underline{\underline{H^*}}.\underline{\ddot{\phi}_d} + \underline{\underline{\dot{H}^*}}.\underline{\dot{\phi}} - \frac{\partial}{\partial \phi}\left\{\frac{1}{2}.\,{}^t\underline{\dot{\phi}}.\underline{\underline{H^*}}.\underline{\dot{\phi}}\right\} = \underline{\tau_d} \tag{13}$$

4.2.6 Solving joints gradient from equation of motion

In equation 13, the third product on the first hand of the equation is problematic. I settled to find out explicit derivatives of the ${}^t\dot{\phi}.H^*.\dot{\phi}$ product with help of a formal algebra software, Maple©. First, I simplified the algebra by reducing the inversion of the 6 by 6 block matrix $H_{b\omega}$ (see appendix B), to the inversion of one 3 by 3 matrix, using Volker Strassen formula for the inversion of block matrices :

$$H_{b\omega}{}^{-1} = \begin{bmatrix} \dfrac{I_{d_3}}{m_{0n}} + {}^t\tilde{r}_{0g}.H_\omega^{*-1}.\tilde{r}_{0g} & -{}^t\tilde{r}_{0g}.H_\omega^{*-1} \\ -H_\omega^{*-1}.\tilde{r}_{0g} & H_\omega^{*-1} \end{bmatrix} \tag{14}$$

With H_ω^* being the Schur complement of $H_{b\omega}$:

$$\underline{\underline{H_\omega^*}} = \underline{\underline{H_\omega}} - m_{0n}.\underline{\underline{\tilde{r}_{0g}}}.\underline{\underline{{}^t\tilde{r}_{0g}}} \tag{15}$$

Furthermore, this H_ω^* matrix left to inverse is invertible because of its definite positive character, shown in the following demonstration.

Proof. On a first hand we know that $\tilde{\underline{\underline{r}}}_{0g}$ are antisymmetric matrices, and thus they are of alternate form, thus semi definite positive:

$$\forall \underline{x} \in \mathbb{R}^3, {}^t\underline{x}.\tilde{\underline{\underline{r}}}_{0i}.\underline{x} = 0 \quad \Rightarrow \quad {}^t\underline{x}.\tilde{\underline{\underline{r}}}_{0i}.\underline{x} \geqslant 0 \tag{16}$$

Furthermore, the square of an antisymmetric matrix is definite negative :

$$\forall \underline{x} \in \mathbb{R}^3, {}^t\underline{x}.\tilde{\underline{r}}^2.\underline{x} = ||\underline{x}.\underline{r}||^2 - ||\underline{x}||^2.||\underline{r}||^2 \leqslant 0 \tag{17}$$

On a second hand, we use the triangular inequality : $\forall \underline{x} \in (\mathbb{R}+)^3$,

$${}^t\underline{x}.\left(\sum_{i=1}^{n} m_i.\tilde{\underline{\underline{r}}}_{0i}\right)^2.\underline{x} \geqslant \sum_{i=1}^{n} {}^t\underline{x}.m_i^2.\tilde{\underline{\underline{r}}}_{0i}{}^2.\underline{x} \tag{18}$$

which is true because all elements of the sum are positive, as $\forall i \in [\![1;n]\!], m_i \geq 0$ and as shown beforehand for $\tilde{r}_{0i}$.

On a last hand, we can work out a low limit for ${}^t x.H_\omega^*.x$:

$$H_\omega^* = \sum_{i=0}^{n} I_i + \sum_{i=1}^{n} \left(m_i.{}^t\tilde{r}_{0i}.\tilde{r}_{0i}\right) - m_{0n}.\tilde{r}_{0g}.{}^t\tilde{r}_{0g} \tag{19}$$

$$\begin{aligned}
H_\omega^* - \sum_{i=0}^{n} I_i &= -\sum_{i=1}^{n} \left(m_i.\tilde{r}_{0i}^2\right) + m_{0n}.\tilde{r}_{0g}^2 \\
H_\omega^* - \sum_{i=0}^{n} I_i &= -\sum_{i=1}^{n} m_i.\tilde{r}_{0i}^2 + \frac{\left(\sum_{i=1}^{n} m_i.\tilde{r}_{0i}\right)^2}{m_{0n}} \\
H_\omega^* - \sum_{i=0}^{n} I_i &\geq \underbrace{-\sum_{i=1}^{n} m_i.\tilde{r}_{0i}^2}_{\geq 0} + \frac{\sum_{i=1}^{n} m_i^2.\tilde{r}_{0i}^2}{m_{0n}}
\end{aligned} \tag{20}$$

We multiply each product of the first sum of the second hand by $\frac{m_i}{m_{0n}} < 1$:

$$H_\omega^* - \sum_{i=0}^{n} I_i \geq \frac{-\sum_{i=1}^{n} m_i^2.\tilde{r}_{0i}^2 + \sum_{i=1}^{n} m_i^2.\tilde{r}_{0i}^2}{m_{0n}} = 0 \tag{21}$$

Thereby, we have H_ω^* definite positive:

$${}^t x.H_\omega^*.x \geq \sum_{i=0}^{n} {}^t x.I_i.x > 0 \tag{22}$$

□

For I_s, which is the sum of the inertia matrices and the sum of the opposite of $m_i.\tilde{r}_{gi}^2$, the demonstration is even more straightforward as it uses only the first part of the demonstration for H_ω^*.

Then, as we are sure that H_ω^* is invertible, we use a 3 by 3 matrix inversion formula. We now have the definition of the GIM that is definite.

What is left to do is to derive the GIM with respect to the joint positions. Eventually I would use Maple to solve this problem, so far without success.

4.2.7 Command Generator Module (CGM)

The goal of the CGM is to generate commands to get from the current position (to be understood : 6 components vector composed of the 3 components of the trajectory of the end-effector point and of the 3 components of the attitude of the end-effector link) to the wished position. Eventually this wished position will be the grabbing goal on the moving target, for now it is a fixed position in the inertial space. As for the trajectory requirements, we look for smoothed trajectory, as well as smoothed profiles for the joints speeds and accelerations as we want to reduce the perturbing effect on the base. One way to satisfy those requirements is to have smoothed trajectory, speed profile and acceleration profile on the end-effector. As for the attitude requirement, we look for the shortest rotation to have the end-effector in the final attitude, ready to grab the target. We would also like to have smooth rotation rate and acceleration.

In this first version of the CGM, I used BSplines for the commanded trajectory, and spherical linear interpolation (Slerp) for the commanded attitude.

BSplines are polynomial forms defined piecewise, linked at the abscissa u_i called knot, with an order of continuity of $(m-1)$ for a BSpline of degree m. The control polygon PG_n, formed by $(n+1)$ points $\underline{P_i}$ guides the spline:

$$\underline{P(u)} = \sum_{i=0}^{n} N_{im}(u).\underline{P_i} \tag{23}$$

The abscissa u must be within the range of the knots sequence, $u_0 \leq u_1 \leq \cdots \leq u_p$ such as $u \in [u_i, u_p]$. I set the knots by an uniform pattern. The mixing functions N_{im} are m-degree polynomial functions, with a local support (allowing to modify one point without modifying the whole spline). They are defined recursively:

$$\begin{array}{ll} N_{i0}(u) = 1 & \text{if } u \in [u_i, u_{i+1}[\\ N_{i0}(u) = 0 & \text{else.} \end{array} \tag{24}$$

$$N_{ij} = \frac{u - u_i}{u_{i+j} - u_i}.N_{i,j-1}(u) + \frac{u_{i+j+1} - u}{u_{i+j+1} - u_{i+1}}.N_{i+1,j-1}(u) \tag{25}$$

For more detailed informations about the BSplines, see section 5 "Les courbes Bspline" (BSplines curves) of reference [29] (pages 14-15), and about the setup pattern of the knots, see section 6.1 paragraphs "Paramétrage uniforme" (Uniform setting) and "Choix de la séquence nodale" (Knots sequence choice) in reference [29] (page 16).

In order to get smoothed profiles, I use a high order spline. In the case the arm touches the target, we want the reaction to be minimal. Thus, we want the speeds and accelerations next to null as we approach in really close range the target. Moreover, as speeds and accelerations are null before the movement towards the target, I used a trick to define the control polygon of the trajectory: in addition to the control points, the first and last ones are duplicated twice (depending on the user's will: when he calls for the constructor of the class he can specify to discard this trick, partially by giving 0 and 1 to the two boolean-inputs of completely by giving one or two 1). The effect is that speed and acceleration profiles are tangential to zero in initial and final positions. That way we get a plot for a demonstration trajectory on figure 27.

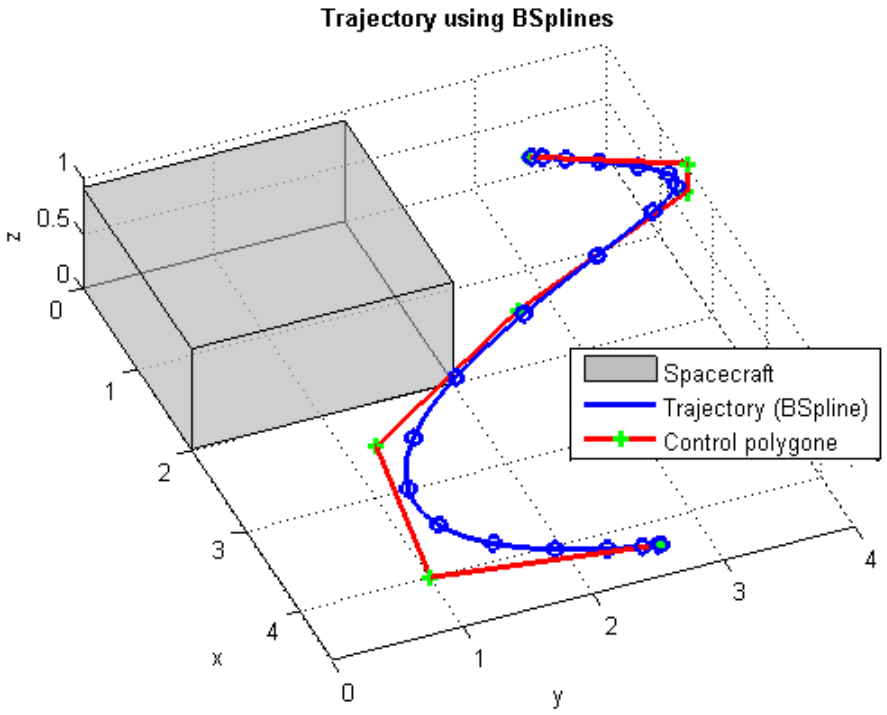


Figure 27: Plot of a example trajectory.

In order to get the command speeds, the trajectory spline is derived, which is equivalent to deriving the mixing functions N_{im}:

$$\underline{C'(u)} = \sum_{i=0}^{n} {N'}_{im}(u).\underline{P_i} \tag{26}$$

$${N'}_{im} = \frac{m}{u_{i+m} - u_i}.N_{i,m-1}(u) - \frac{m}{u_{i+m+1} - u_{i+1}}.N_{i+1,m-1}(u) \tag{27}$$

The command speeds, plotted, give the figure 28 for the speeds on all three axis for the previous trajectory.

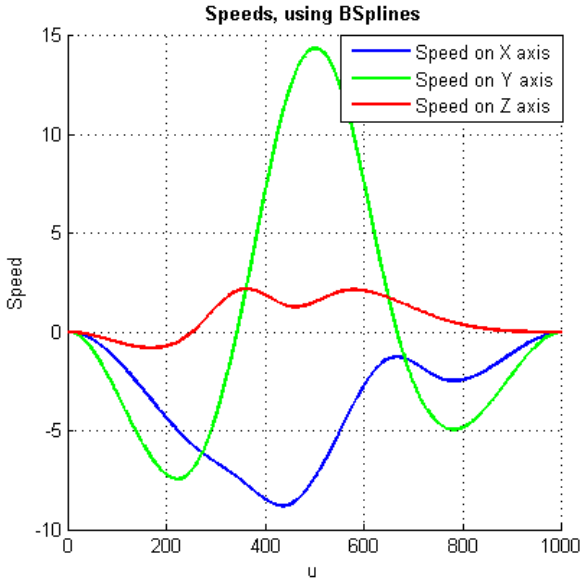


Figure 28: Plot of the speeds on the 3 axis for the previous trajectory (figure 27).

Last, the speeds are derived, or equivalently the N'_{im}, in order to get the accelerations commands:

$$\underline{C''(u)} = \sum_{i=0}^{n} N''_{im}(u).\underline{P_i} \tag{28}$$

$$N''_{im} = m.(m-1).\left(\frac{N_{i,m-2}(u)}{(u_{i+m}-u_i)\,.\,(u_{i+m-1}-u_i)} - \frac{(u_{i+m}-u_i+u_{i+m+1}-u_{i+1})\,.N_{i+1,m-2}(u)}{(u_{i+m}-u_i)\,.\,(u_{i+m}-u_{i+1})\,.\,(u_{i+m+1}-u_{i+1})} + \frac{N_{i+2,m-2}(u)}{(u_{i+m+1}-u_{i+1})\,.\,(u_{i+m+1}-u_{i+2})}\right)$$

This second derivative of the trajectory spline, plotted, give the figure 29 for the 3 accelerations, the same way than for speeds.

However those speeds and accelerations are not the command speeds and accelerations. Indeed, with $u = f(t)$, the trajectory $C(u)$ is also $C(f(t)) = (C \circ f)(t)$. To get command speed and acceleration, the trajectory is derived with respect to the time :

$$S(t) = C'(f(t)).f'(t) \tag{29}$$

$$A(t) = C''(f(t)).(f'(t))^2 + C'(f(t)).f''(t) \tag{30}$$

As I choose the f function to be linear and satisfy the limits, $f(0) = 0$ and $f(T_f) = 1$, which result in $u = f(t) = \frac{t}{T_f}$, its second derivative is identically null. The duration T_f for

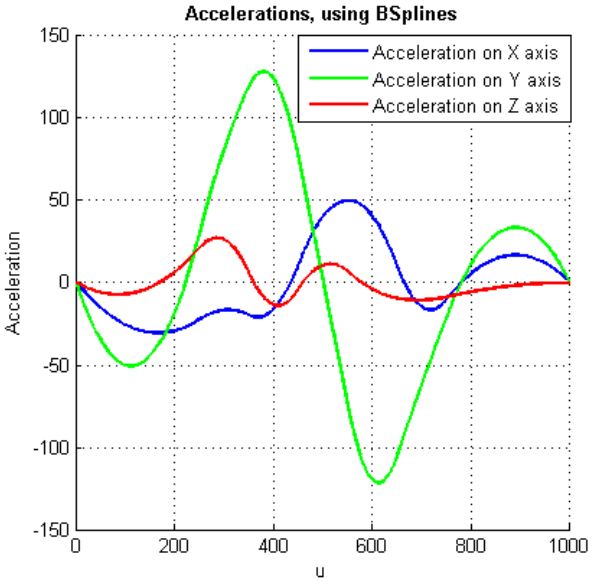


Figure 29: Plot of the accelerations on the 3 axis for the previous trajectory (figure 27).

the simulation and experiment is determined either by the user or by the ratio between the distance from initial position to goal position with the maximum curvilinear speed wished, $T_f = \frac{d}{v_c}$. Moreover, as the trajectory is sampled by NoP number of points, the equation $T_f = NoP.h$ comes out, h being the time-step for an update. This way, with the duration of the experiment T_f, the number of points in which the trajectory is sampled is $NoP = \frac{T_f}{h}$. Therefore, the command speeds are:

$$\underline{S(t)} = \frac{1}{NoP.h}.\sum_{i=0}^{n} N'_{im}(u).\underline{P_i} \tag{31}$$

And the command accelerations are:

$$\underline{A(t)} = \frac{1}{(NoP.h)^2}.\sum_{i=0}^{n} N''_{im}(u).\underline{P_i} \tag{32}$$

As for the attitude of the end-effector, the Slerp define the shortest rotation to go from one attitude to another [30]. I chose to use it. To do so, the attitudes are first converted in quaternions and then the Slerp formula for quaternion is applied. An example is plotted on figure 30.

Then, the commanded attitude at instant t is converted again into euler angles, carefully because of the gimbal lock situation. This determinates the joint positions. Command laws for rotational speeds and accelerations are derived from the slerp function. The interpolation function is a 5th degree curve to meet the conditions (nul initial and final speeds and acceleration, equal to the unit at T_f).

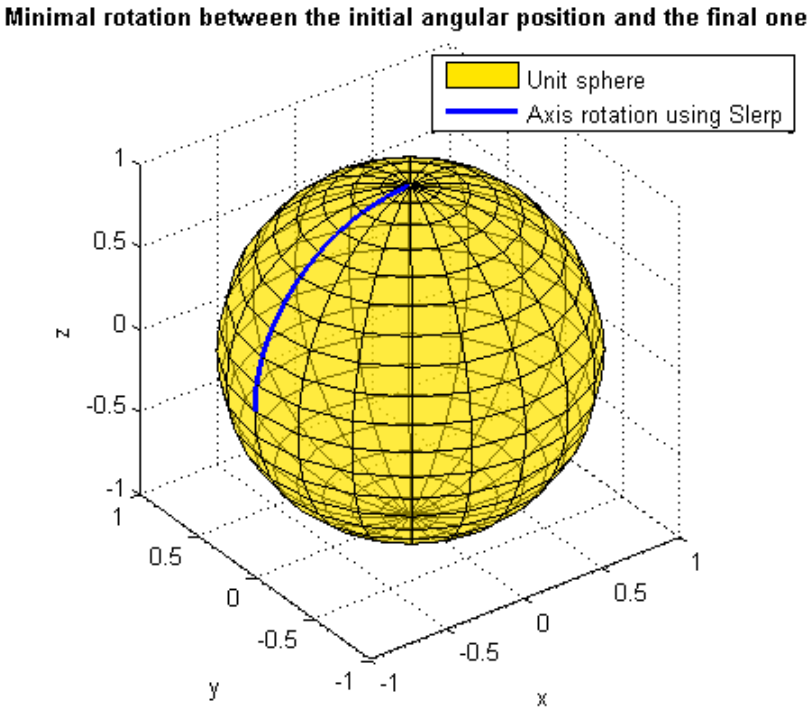


Figure 30: Plot of initial and final attitude on the unit sphere, and the slerp-determined 'trajectory' to go from one to the other attitude. (Only the tip of the axis of rotation is plotted)

4.2.8 Tests

Means:

For debugging purpose, a simplified robot was used (2 DOFs, fixed base) on a simple trajectory (circle in the X-Y plane), see figure 32. The real robot is used on dummy trajectories (circles in XY, XZ, YZ planes), see figure 33 and on complex trajectories, see figure 31 and next paragraph for details.

Nota Bene about figures 31, 32 and 33: The manipulator seems to be not linked to the base, but that is just a display simplification of the base.

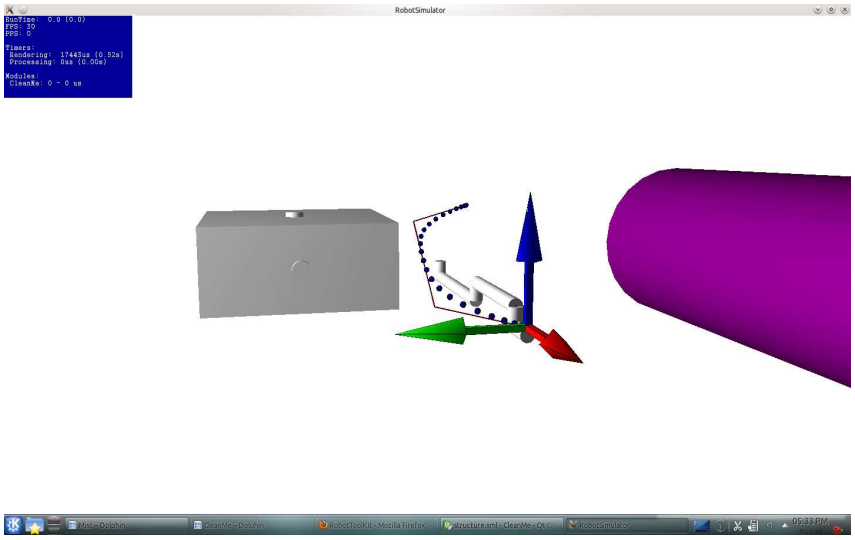


Figure 31: The spacecraft and its manipulator, with the trajectory displayed by small spheres and the control polygon.

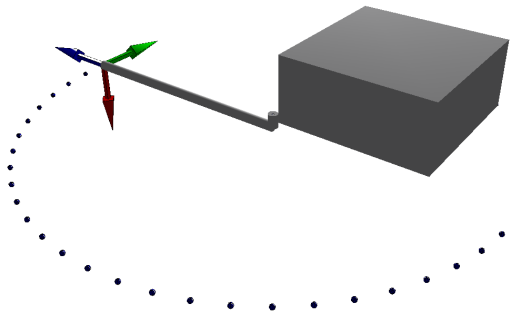


Figure 32: The spacecraft and its simplified manipulator, with the trajectory displayed by small spheres. This dummy trajectory is a simple circle centred on the origin of the arm.

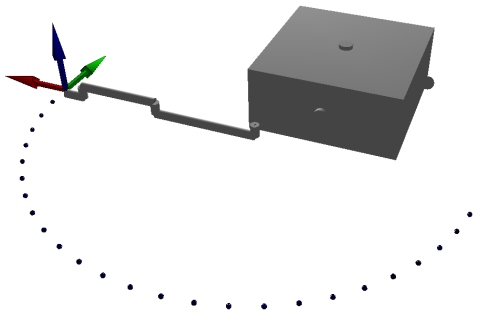


Figure 33: The spacecraft and its manipulator, with the trajectory displayed by small spheres. This dummy trajectory is a simple circle centred on the origin of the arm.

Analysis:

I made two analysis for the last three implementations of the inversion process: during the travel, following a path to its destination, and at destination.

The numerical analysis are available in appendices C and D.

The trajectory set for the tests are: (smoothed means 4th-degree curve, with nil initial and final speeds and accelerations)

- 3 radius:
  - 1m, too close to the base with attitude control
  - 2m, within working range
  - 3m, too far, unreacheable (length of the fully extended arm: $2.78\,m$).
- Each radius for 3 circles in orthogonal planes [inertial XY, inertial XZ, inertial YZ] to test for the entire half sphere of action,
- Straight line: smoothed (BSpline with a control polygon of only 2 points), difficult for a serial manipulator
- Complex BSpline/ smoothed, the actual kind of trajectory aimed at, set for control of acceleration and speed.

Arm length and radius length:
The arm length is $2.78\,m$, so the $3\,m$ radius trajectories are purposely out of reach. The goal is to see if the controller will follow the trajectory minus the unreachable length, and observe the stability. The two main links of the arm are $1,4\,m$ and $1,1\,m$ long so the $1\,m$ radius trajectories are purposely out of reach with the attitude control. Will the controller make a trade-off between the attitude and the tracking target?

The result depends on the method used for the inversion. With the DLS, the 1m radius path is followed with a trade off between the attitude and the tracking error. In the SDLS and MSDLS cases, the tracking error is globally lower than the the attitude error. The same goes for the 3m radius: the DLS keeps the attitude error restrained and follows the path with a certain distance (corresponding to the distance between the end-point of the arm and the trajectory) when the SDLS and MSLDS try to minimize the tracking error through unsteady behaviour.

Definition of statistical criteria used:

$$NRMSE = \frac{\sqrt{\left(\frac{\sum_{i=1}^{n}\left(x\left[i\right]-x_d\left[i\right]\right)^2}{n}\right)}}{x_{max}-x_{min}} \tag{33}$$

with $x_{max} = max(x, x_d)$ and $x_{min} = min(x, x_d)$.

$$RMSE = \sqrt{\left(\frac{\sum_{i=1}^{n}\left(x[i]-x_d[i]\right)^2}{n}\right)} \tag{34}$$

Definition of error for attitude:
The problem with the attitude is the domain of the values. In degrees, we can have successively 179° and -179° with a gap of only 2 degrees but the computed difference will be 358°. I used the same error function to compute the attitude error than in the code, equation 12, which evaluate the angle between actual and desired position: for (i, j, k) the three vectors representing the frame with respect to the hand.

Evaluation of $||q||$ at target:
On an earth-based manipulator, it would not make sense to qualify the steadiness of the joints, as the hand should have reached its target. However, on a space manipulator, every movement of the arm induces an effort on the base, resulting in movement of the said base. As there is no other forces, this movement is not compensated. Because of this, the base moves when the arm is on target, and qualifying the steadiness of the joints is a mean to qualify the steadiness of the base.

Quality analysis:
Pseudo-Inverse, transpose Jacobian as inverse and the first implementation of the tweaked SVD inversion (with erasing the too close to 0 eigenvalues) are way too unstable at singularities for our problem. The first encouraging results came with the second implementation of the SVD inversion (with inverse eigenvalues set to 0) but for fixed robot instead of the free floating base, because the joint velocities were too high to keep the base at a close to steady attitude of the base. The last three inversion methods consider the joints velocities and for those the results are encouraging. That is why the numerical analysis (appendices C for tracking and D at target) is only about those, the other ones are not exploitable at all as the base's attitude does not allow the arm to reach its goal.

- SDLS: too noisy but quite accurate. It is not usable in our case.
- DLS: good results, probably due to the good set up of the damping constant. The more stable of the three, does not generate too high joint or hand speeds even in the unreachable cases.
- MSDLS: Settings: By only retriving the optimal damped constants from SDLS, too unstable (SDLS inversion leads to the worst dq/dt, visible by a lot of oscillation in the simulation) so I added a small minimum constant (1/10th of the damping constant of the DLS method). Observations: With those settings, it gets the stability of the DLS and the recovery capacity of SDLS. However, there is even more oscillations than the SDLS on unreachable targets (see 3-meter circular trajectories). In all cases the accuracy is better than with the other methods.

This analysis is illustrated on table 2 with data extracted from appendix C for targeted case: BSpline. For the most stable case, DLS, the trajectory is projected in XY and ZY planes figures 34 and 35, for a starting point at $[x = 4.48, y = 2.51, z = 0.342]$, and hand attitude, decomposed on the 3 axis, is depicted on figure 36. The stability of the motion is evaluated from the joint speeds, figure 37.

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|\dot{q}\|\|$
DLS	0.362% ie 0.008	1.230% ie 0.917	58.821
min	3.063	180.0	0.0
max	5.150	254.6	305.7
MSDLS	0.047% ie 0.001	1.117% ie 0.833	86.375
min	3.063	180.0	0.0
max	5.147	254.6	3106.1
SDLS	0.090% ie 0.002	1.292% ie 0.963	566.268
min	3.063	180.0	0.0
max	5.147	254.6	1703.9

Table 2: Complex smoothed BSpline trajectory: nul initial and final speeds and accelerations

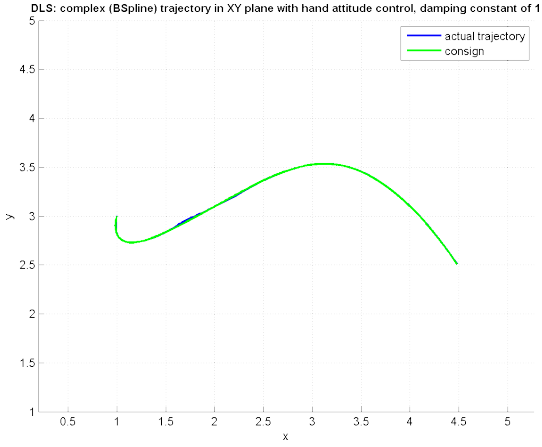


Figure 34: BSpline-DLS case: trajectory projected on XY plane.

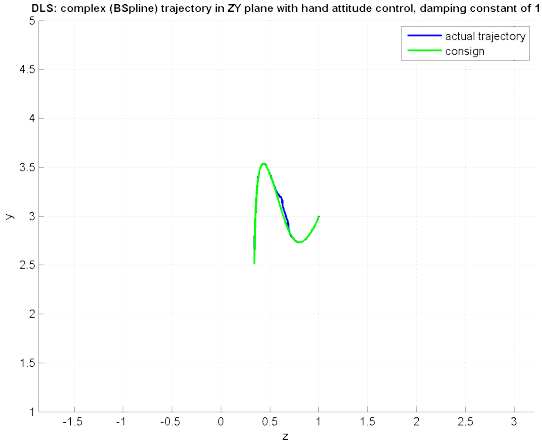


Figure 35: BSpline-DLS case: trajectory projected on ZY plane.

The tracking error of the DLS method is higher than with the other methods, but it is more stable.

All data, graphs and video of the tests are available at this url: `http://perso.crans.org/blanchet/epfl/results/`.

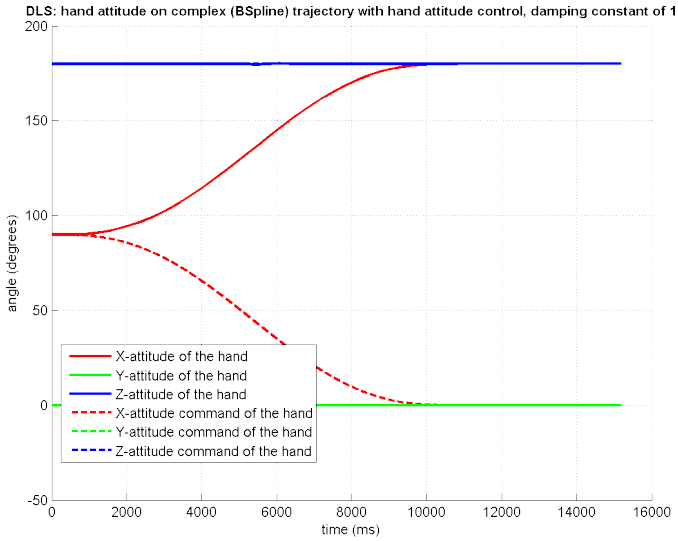


Figure 36: BSpline-DLS case: hand attitude decomposed on 3 axis, modulo 180°.

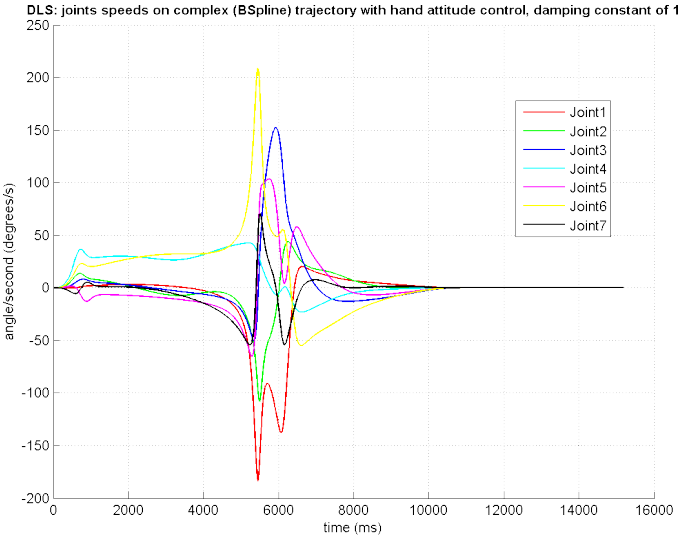


Figure 37: BSpline-DLS case: joint speeds.

5 Conclusion

In this project's report, we proposed an original scenario for an ODR mission. There is room for improvements, if only for the fact that one chaser spent for each target is quite prohibitive. Furthermore, the mathematical modelling is not good enough. Ideally, the end-effector should be positioned in a target-fixed coordinate frame, and the satellite base in the inertial coordinate frame.

Nonetheless, on this first implementation, there is still work to do, but for reachable tra-

jectories the accuracy requirements are met. Even though the DLS inversion method does not always meet the accuracy requirements, I recomend its use as it is the more stable inversion method.

Moreover, The concept and scenario seem promising as the grabbing process does not uses fuel to have the chaser track the target like any other scenario, which is quite expensive.

6 Left to do

Two methods in the `generalizedjacobianmethod` are left to debug; the tracking of the goal-point onto the target is also not implemented.

On another scale, next step is to add noise on the joint positions and velocities that are measured and fed back to the control loop, corresponding to the inaccuracy of the dedicated sensors. In the same line of thought, the position, orientation, velocity and rotation rate of the end-effector, as well as the position, orientation, rotation rate of the target should be modified by a noise corresponding to the inaccuracy of the video feedback.

A Requirements

Description		Value	Status	Justification
Debris retrieval capability	Dims (h x d)[m]	10.2x2.6	A4 H10 case	Target: Ariane upper stage, we shall be able to collect them all.
	Mass [kg]	4500	A5 ESC case	
	Quantity	1	Proposed	Rather large debris...
Debris retrieval system	Target damaging	No	Proposed	Goal is to stop object multiplication
	Target selection	Yes	Proposed	Target shape & degradation state
Target/SC relative rotation velocity		$60^\circ/s$	Proposed	ESA & NASA estimated max feasible
Target/hand relative...	rotation velocity	$< 1^\circ/s$	Proposed	Velocity values: do not damage the target nor perturb it
	translation velocity	$< 0.02\,m/s$	Proposed	
	pointing accuracy	$< 1^\circ$	Proposed	Accuracy values: achieved by Yoshida (ETS VII), depend of grabbing zone
	position accuracy	$< 0.01\,m$	Proposed	
Robotic arm length	Extended	$> 2\,m$	Proposed	A4 C_g-rod $\approx 2\,m$
	Folded [m]	< 2x0.9x0.5	Proposed	Fits Yuki design
Robotic arm characteristics	DOF	7	Proposed	[RNS: 3] + [Target tracking: 3] + [Rotation of the target: 1]
	Mass	$20\,kg - 45\,kg$	Proposed	lightest space qualified arm: DLR ($45\,kg$)
	Power max [W]	$50 - 120$	Proposed	Wished :< 50; DLR arm needs: $120\,W$

B Generalized Jacobian Matrix

This enhanced Jacobian matrix depend only on the joint positions and on the base attitude. Before defining the working variables and matrices, we define the following elementary variables (reefer to the schematic figure 38) along with one operator:

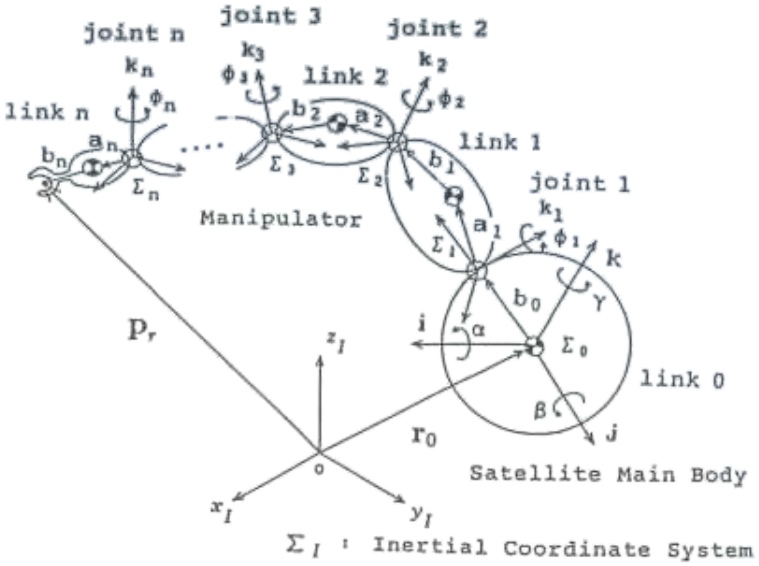


Figure 38: Schematic of a space manipulator with the parametrisation associated.

- $\forall i \in [\![1;n]\!]$, $\underline{a_i}$ the vector between $\underline{p_i}$ and $\underline{r_i}$ for each link (intricate to the link and its orientation, constant norm),
- $\forall i \in [\![0;n]\!]$, $\underline{b_i}$ the vector between $\underline{r_i}$ and $\underline{p_{i+1}}$ for each link (intricate to the link and its orientation, constant norm),
- $\forall i \in [\![1;n]\!]$, $\underline{\underline{R_{\alpha i}}}$, rotation matrix between $\underline{p_i p_{i+1}}$ and $\underline{a_i}$ for the link i (intricate to the link, constant),
- $\forall i \in [\![0;n]\!]$, $\underline{\underline{R_{\beta_i}}}$, rotation matrix between $\underline{p_i p_{i+1}}$ and $\underline{b_i}$ for the link i (intricate to the link, constant),
- $\forall i \in [\![1;n]\!]$, $\underline{\underline{R_i}}$, rotation matrix between $\underline{k_i}$ and $\underline{k_{i+1}}$ for the link i (intricate to the link, constant),
- $\forall i \in [\![1;n]\!]$, $\underline{\underline{R(\phi_i)}}$, matrix of the ϕ_i rotation on the $\underline{k_i}$ axis (z-axis rotation of ϕ_i matrix formatted),
- $\underline{\underline{\tilde{x}}} = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$, the antisymmetric matrix associated with the vector $\underline{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$.

Thanks to those, we define the working variables:

- $\forall i \in [\![0;n]\!]$, m_i the mass of each link,
- $\forall (i,j) \in [\![0;n]\!]^2$, $m_{ij} = \sum_{k=i}^{j} m_k$ the compact sum of masses, in particular m_{0n}, representing the mass of the whole spacecraft,
- $\forall i \in [\![0;n]\!]$, $\underline{\underline{I_i}}$ the inertia matrix of each link,
- $\forall i \in [\![0;n]\!]$, $\underline{r_i}$ the position of the center of mass of each link,
- $\underline{r_g}$ the position of the center of mass of the whole spacecraft $\left(m_{0n}.\underline{r_g} = \sum_{i=0}^{n} m_i.\underline{r_i}\right)$,
- $\forall i \in [\![1;n]\!]$, $\underline{p_i}$ the position of the center of the joint i,
- $\underline{p_r}$ the position of the end-effector point,
- $\forall i \in [\![0;n]\!]$, $\underline{\underline{\Omega_i}}$ the attitude of each link,
- $\underline{\theta_n}$ the orientation of the end-effector point,
- $\forall i \in [\![0;n]\!]$, $\underline{k_i}$ the axis vector of each joint $\left(\underline{k_i} = \underline{\underline{\Omega_i}}.\underline{z}\right)$,
- $\forall (i,j) \in ([\![0;n]\!] \cup \{g\})^2$, $r_{ij} = r_j - r_i$ the vector between 2 center of mass,
- $\forall (i,j) \in ([\![0;n]\!] \cup \{r\})^2$, $p_{ij} = p_j - p_i$ the vector between 2 joints.

And how they are linked to the elementary variables: $\forall i \in [\![1;n]\!]$,

- $\underline{p_{i+1}} = \underline{r_i} + \underline{b_i}$,
- $\underline{r_i} = \underline{a_i} + \underline{p_i}$,
- $\underline{k_i} = \underline{\underline{R\,(\phi_i)}}.\underline{\underline{R_i}}.\underline{k_{i-1}}$,
- $\underline{a_i} = a_i.{}^t\underline{\underline{R_{\alpha i}}}.\underline{\underline{R\,(\phi_i)}}.\underline{\underline{R_i}}.\dfrac{\underline{p_i p_{i-1}}}{||\underline{p_i p_{i-1}}||}$,
- $\underline{b_i} = b_i.{}^t\underline{\underline{R_{\beta_i}}}.\underline{\underline{R\,(\phi_i)}}.\underline{\underline{R_i}}.\dfrac{\underline{p_i p_{i-1}}}{||\underline{p_i p_{i-1}}||}$.

Thereby we can define the useful vectors and matrices:

- $\underline{\phi} = {}^t(\phi_1, \phi_2, \ldots, \phi_n)$ the joints positions vector,
- $\underline{X} = \begin{bmatrix} \underline{p_r} \\ \underline{\theta_n} \end{bmatrix}$ the end-effector vector in task space,

- $\underline{\tau} = {}^t(\tau_1, \ldots, \tau_n)$ the vector of the torques,
- $\underline{\underline{J^*}} = \left[\underline{\underline{\widehat{J_m}}} - \underline{\underline{\widehat{J}_s}}.\underline{\underline{I_s}}^{-1}.\underline{\underline{I_m}}\right]$ the Generalized Jacobian Matrix, such as $\underline{\dot{X}} = \underline{\underline{J^*}}.\underline{\dot{\phi}}$,
- $\underline{\underline{H^*}} = \underline{\underline{H_\phi}} - {}^t\underline{\underline{H_{bm}}}.\underline{\underline{H_{b\omega}}}^{-1}.\underline{\underline{H_{bm}}}$, with: $\underline{\underline{H_{bm}}} = \begin{bmatrix} \underline{\underline{J_{T\omega}}} \\ \underline{\underline{H_{\omega\phi}}} \end{bmatrix}$, and $\underline{\underline{H_{b\omega}}} = \begin{bmatrix} m_{0n}.\underline{\underline{I_{d_3}}} & m_{0n}.{}^t\underline{\underline{\tilde{r}_{0g}}} \\ m_{0n}.\underline{\underline{\tilde{r}_{0g}}} & \underline{\underline{H_\omega}} \end{bmatrix}$.

With the underlining matrices of the jacobian:

- $\underline{\underline{\widehat{J_m}}} = \underline{\underline{J_m}} + \underline{\underline{J_s}}.\begin{bmatrix} \underline{\underline{J_{T\omega}}}/m_{0n} \\ \underline{\underline{0_3}} \end{bmatrix}$,
- $\underline{\underline{J_m}} = \begin{bmatrix} \underline{k_1} \wedge (\underline{p_r} - \underline{p_1}) & \ldots & \underline{k_n} \wedge (\underline{p_r} - \underline{p_n}) \\ \underline{k_1} & \ldots & \underline{k_n} \end{bmatrix}$,
- $\underline{\underline{J_s}} = \begin{bmatrix} \underline{\underline{I_{d_3}}} & \underline{\underline{-\tilde{p}_{0r}}} \\ \underline{\underline{0_3}} & \underline{\underline{I_{d_3}}} \end{bmatrix}$,
- $\underline{\underline{\widehat{J}_s}} = \underline{\underline{J_s}}.\begin{bmatrix} \underline{\underline{\tilde{r}_{0g}}} \\ \underline{\underline{I_{d_3}}} \end{bmatrix}$,
- $\underline{\underline{I_s}} = \underline{\underline{I_0}} + m_{0n}.\underline{\underline{\tilde{r}_g}}.\underline{\underline{\tilde{r}_{0g}}} + \sum_{i=1}^{n}\left(\underline{\underline{I_i}} - m_i.\underline{\underline{\tilde{r}_i}}.\underline{\underline{\tilde{r}_{0i}}}\right) = \sum_{i=0}^{n}\left(\underline{\underline{I_i}} - m_i.\underline{\underline{\tilde{r}_{gi}}}^2\right)$,
- $\underline{\underline{I_m}} = \sum_{i=1}^{n}\left(\underline{\underline{I_i}}.\underline{\underline{J_{Ri}}} + m_i.\underline{\underline{\tilde{r}_i}}.\underline{\underline{J_{Ti}}}\right) - \underline{\underline{\tilde{r}_g}}.\underline{\underline{J_{T\omega}}}$.

And of the inertia tensor:

- $\underline{\underline{H_\omega}} = \underline{\underline{I_0}} + \sum_{i=1}^{n}\left(\underline{\underline{I_i}} + m_i.{}^t\underline{\underline{\tilde{r}_{0i}}}.\underline{\underline{\tilde{r}_{0i}}}\right)$,
- $\underline{\underline{H_{\omega\phi}}} = \sum_{i=1}^{n}\left(\underline{\underline{I_i}}.\underline{\underline{J_{Ri}}} + m_i.\underline{\underline{\tilde{r}_{0i}}}.\underline{\underline{J_{Ti}}}\right)$,
- $\underline{\underline{H_\phi}} = \sum_{i=1}^{n}\left({}^t\underline{\underline{J_{Ri}}}.\underline{\underline{I_i}}.\underline{\underline{J_{Ri}}} + m_i.{}^t\underline{\underline{J_{Ti}}}.\underline{\underline{J_{Ti}}}\right)$.

And finally the common matrices to both the jacobian and the inertia tensor:

- $\underline{\underline{J_{Ti}}} = \left[\ \underline{k_1} \wedge (\underline{r_i} - \underline{p_1}) \quad \ldots \quad \underline{k_i} \wedge (\underline{r_i} - \underline{p_i}) \quad 0 \quad \ldots \quad 0\ \right]$,
- $\underline{\underline{J_{Ri}}} = \left[\ \underline{k_1} \quad \ldots \quad \underline{k_i} \quad 0 \quad \ldots \quad 0\ \right]$,
- $\underline{\underline{J_{T\omega}}} = \sum_{i=1}^{n} m_i.\underline{\underline{J_{Ti}}}$.

C Numerical analysis of the results - tracking case

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed position	NRMSE speed attitude
DLS	0.341% ie 0.006	1.183% ie 0.882	127.887		69.734	44.781%	62.882%
min	3.317	180.0	0.0		0.0	8.4	299.4
max	5.147	254.6	348.3		649.1	34.5	970.6
MSDLS	0.270% ie 0.005	1.241% ie 0.925	161.744		329.944	67.035%	87.064%
min	3.316	180.0	0.0		0.0	51.3	1673.2
max	5.147	254.6	445.7		9517.9	123.0	2275.9
SDLS	0.399% ie 0.007	1.104% ie 0.823	127.121		326.225	58.610%	61.595%
min	3.317	180.0	0.0		0.0	44.0	1044.5
max	5.147	254.6	347.8		1167.2	585.6	12063.8

Table 3: Straight smoothed trajectory: nul initial and final speeds and accelerations

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed position	NRMSE speed attitude
DLS	0.362% ie 0.008	1.230% ie 0.917	106.620		58.821	26.893%	61.549%
min	3.063	180.0	0.0		0.0	13.2	216.2
max	5.150	254.6	340.4		305.7	32.2	953.5
MSDLS	0.047% ie 0.001	1.117% ie 0.833	124.234		86.375	38.355%	66.226%
min	3.063	180.0	0.0		0.0	20.0	284.0
max	5.147	254.6	344.2		3106.1	34.4	978.9
SDLS	0.090% ie 0.002	1.292% ie 0.963	122.550		566.268	59.077%	65.543%
min	3.063	180.0	0.0		0.0	101.9	2298.8
max	5.147	254.6	338.5		1703.9	1066.9	20456.0

Table 4: Complex smoothed BSpline trajectory: nul initial and final speeds and accelerations

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	0.165% ie 0.004	0.730% ie 0.544	92.675		33.127	21.999%	58.073%
min	2.628	180.0	0.0		0.0	11.5	136.8
max	5.147	254.5	322.6		256.7	31.7	1340.1
MSDLS	0.048% ie 0.001	0.730% ie 0.544	92.398		44.010	23.814%	60.104%
min	2.628	180.0	0.0		0.0	12.7	293.2
max	5.147	254.6	323.0		1524.2	33.0	1346.6
SDLS	0.047% ie 0.001	0.730% ie 0.544	92.463		449.278	65.907%	62.959%
min	2.628	180.0	0.0		0.0	170.7	2582.4
max	5.147	254.6	323.6		1696.0	453.2	22227.0

Table 5: Circular trajectory of 1m radius in the XY plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	0.089% ie 0.002	1.119% ie 0.834	79.973		17.830	19.882%	59.141%
min	2.627	180.0	0.0		0.0	5.0	183.9
max	5.147	254.6	281.3		285.3	33.9	1298.7
MSDLS	0.044% ie 0.001	0.730% ie 0.544	79.546		23.873	19.850%	59.124%
min	2.628	180.0	0.0		0.0	5.1	219.2
max	5.147	254.6	281.4		989.8	34.6	1244.2
SDLS	0.040% ie 0.001	0.730% ie 0.544	79.504		290.582	54.466%	61.009%
min	2.628	180.0	0.0		0.0	56.2	1147.4
max	5.147	254.6	281.4		970.5	384.6	21771.5

Table 6: Circular trajectory of 1m radius in the XZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	0.130% ie 0.003	0.637% ie 0.475	98.530		32.776	65.460%	60.382%
min	3.102	180.0	0.0		0.0	11.7	181.5
max	5.147	254.5	361.4		401.0	32.9	1410.6
MSDLS	0.036% ie 0.001	0.637% ie 0.475	98.270		42.903	63.792%	60.803%
min	3.102	180.0	0.0		0.0	13.3	327.0
max	5.147	254.6	361.3		2189.7	34.3	1427.4
SDLS	0.055% ie 0.001	0.646% ie 0.482	97.934		1389.157	62.736%	58.108%
min	3.102	180.0	0.0		0.0	171.0	13639.3
max	5.147	254.6	360.6		6291.7	1060.5	267106.8

Table 7: Circular trajectory of 1m radius in the YZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	0.057% ie 0.001	1.077% ie 0.803	61.911		17.397	17.676%	57.636%
min	2.551	180.0	0.0		0.0	3.7	51.8
max	5.147	254.6	262.8		258.4	34.2	1010.1
MSDLS	0.015% ie 0.000	0.811% ie 0.604	65.596		22.459	16.853%	58.114%
min	2.551	180.0	0.0		0.0	4.6	75.8
max	5.147	254.6	262.8		920.3	37.8	1012.9
SDLS	0.029% ie 0.001	1.053% ie 0.785	61.475		684.255	67.218%	60.120%
min	2.551	180.0	0.0		0.0	353.3	3711.9
max	5.147	254.6	263.1		2212.4	926.0	23874.4

Table 8: Circular trajectory of 2m radius in the XY plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	0.052% ie 0.001	0.937% ie 0.698	54.402		26.398	22.972%	79.210%
min	2.551	180.0	0.0		0.0	14.6	609.8
max	5.147	254.5	247.0		262.3	40.1	1466.0
MSDLS	0.014% ie 0.000	0.833% ie 0.620	60.003		34.786	23.844%	71.802%
min	2.550	180.0	0.0		0.0	18.5	768.2
max	5.147	254.5	244.9		920.3	47.4	2144.0
SDLS	0.009% ie 0.000	1.342% ie 1.000	53.684		177.999	51.139%	63.192%
min	2.550	180.0	0.0		0.0	42.6	1115.7
max	5.147	254.5	245.0		970.3	306.2	6025.1

Table 9: Circular trajectory of 2m radius in the XZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	0.041% ie 0.001	0.732% ie 0.546	65.599		8.088	53.766%	61.376%
min	3.455	180.0	0.0		0.0	16.4	117.5
max	5.147	254.5	276.3		230.2	29.3	1060.8
MSDLS	0.174% ie 0.003	0.737% ie 0.550	67.269		1420.381	61.460%	57.777%
min	3.455	180.0	0.0		0.0	682.9	7068.7
max	5.147	254.6	314.6		14000.1	2824.8	275579.3
SDLS	0.197% ie 0.003	0.738% ie 0.550	66.756		1482.743	62.046%	59.063%
min	3.455	180.0	0.0		0.0	770.3	28250.2
max	5.147	254.6	317.0		6782.1	2909.6	313469.0

Table 10: Circular trajectory of 2m radius in the YZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	8.628% ie 0.286	0.935% ie 0.725	152.741		44.846	38.311%	62.923%
min	2.727	176.3	0.0		0.0	26.4	892.5
max	6.041	253.9	500.9		2695.1	47.6	3875.6
MSDLS	5.038% ie 0.164	0.283% ie 0.603	96.532		4757.373	68.338%	81.042%
min	2.782	33.4	0.0		0.0	2742.0	276285.2
max	6.041	246.2	572.5		57292.5	8455.5	659270.2
SDLS	8.614% ie 0.285	0.628% ie 0.740	54.422		1389.128	76.235%	59.397%
min	2.729	134.6	0.0		0.0	2907.2	45379.2
max	6.041	252.4	235.9		5750.9	4760.9	340017.4

Table 11: Circular trajectory of 3m radius (unreachable) in the XY plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	3.769% ie 0.099	0.979% ie 0.728	59.570		49.801	45.497%	60.408%
min	2.736	180.0	0.0		0.0	33.4	690.5
max	5.358	254.4	358.0		2517.0	35.7	3410.2
MSDLS	3.209% ie 0.084	0.485% ie 0.845	35.635		5383.372	67.011%	63.468%
min	2.736	78.0	0.0		0.0	1117.2	218073.0
max	5.358	252.2	370.5		24277.8	7243.6	897758.3
SDLS	3.914% ie 0.103	0.995% ie 0.739	79.603		1654.753	77.970%	58.119%
min	2.739	180.0	0.0		0.0	1219.0	22570.2
max	5.358	254.3	351.6		5681.5	1834.2	368959.0

Table 12: Circular trajectory of 3m radius (unreachable) in the XZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE $\|\|q\|\|$		RMSE $\|\|\dot{q}\|\|$	NRMSE speed position	NRMSE speed attitude
DLS	16.974% ie 0.362	0.962% ie 0.782	65.706		260.144	71.902%	84.971%
min	3.656	172.7	0.0		0.0	220.6	3111.5
max	5.788	254.0	381.8		6019.0	308.2	5905.5
MSDLS	9.559% ie 0.181	0.503% ie 1.252	87.064		8341.245	76.067%	86.506%
min	3.898	10.6	0.0		0.0	2312.3	559659.5
max	5.788	259.4	493.7		73124.7	4016.3	739638.5
SDLS	17.377% ie 0.366	0.307% ie 0.768	71.529		1576.707	78.129%	61.368%
min	3.683	2.2	0.0		0.0	4919.0	100099.4
max	5.788	252.5	396.3		8715.6	7512.4	445786.3

Table 13: Circular trajectory of 3m radius (unreachable) in the YZ plane

D Numerical analysis of the results - at target case

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	5.629% ie 0.000	0.000% ie 0.000	0.000		0.001	5.526%	5.576%
min	3.317	180.000	348.270		0.000	0.000	0.000
max	3.317	254.558	348.270		0.017	0.000	0.002
MSDLS	28.945% ie 0.000	0.002% ie 0.002	11.445		20.162	46.189%	44.887%
min	3.316	180.000	420.726		26.239	0.025	16.129
max	3.317	254.556	457.319		530.185	1.500	68.270
SDLS	1.449% ie 0.000	0.000% ie 0.000	0.000		0.000	1.515%	1.541%
min	3.317	180.000	347.806		0.000	0.000	0.000
max	3.317	254.558	347.806		0.000	0.000	0.000

Table 14: Straight smoothed trajectory: nul initial and final speeds and accelerations

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	2.286% ie 0.000	0.000% ie 0.000	0.000		0.000	2.653%	2.675%
min	3.317	180.000	251.792		0.000	0.000	0.000
max	3.317	254.558	251.792		0.001	0.000	0.000
MSDLS	2.287% ie 0.000	0.000% ie 0.000	0.000		0.000	2.395%	2.438%
min	3.317	180.000	344.169		0.000	0.000	0.000
max	3.317	254.558	344.169		0.000	0.000	0.000
SDLS	1.533% ie 0.000	0.000% ie 0.000	0.000		0.000	1.656%	1.649%
min	3.317	180.000	338.496		0.000	0.000	0.000
max	3.317	254.558	338.496		0.000	0.000	0.000

Table 15: Complex smoothed BSpline trajectory: nul initial and final speeds and accelerations

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	1.657% ie 0.000	0.000% ie 0.000	0.001		1.467	2.467%	3.180%
min	2.628	180.000	322.649		0.000	0.000	0.000
max	2.629	254.558	322.697		41.119	0.523	5.452
MSDLS	1.478% ie 0.000	0.000% ie 0.000	0.000		0.954	2.141%	2.167%
min	2.628	180.000	322.961		0.000	0.000	0.000
max	2.629	254.558	322.979		40.968	0.522	2.859
SDLS	1.425% ie 0.000	0.000% ie 0.000	0.000		0.896	2.126%	2.593%
min	2.628	180.000	323.550		0.000	0.000	0.000
max	2.629	254.558	323.565		41.081	0.522	1.574

Table 16: Circular trajectory of 1m radius in the XY plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	3.942% ie 0.000	0.000% ie 0.000	0.003		2.262	2.375%	3.423%
min	2.627	180.000	281.265		0.000	0.000	0.000
max	2.628	254.558	281.356		66.683	0.358	7.493
MSDLS	2.208% ie 0.000	0.000% ie 0.000	0.000		1.527	2.287%	2.173%
min	2.628	180.000	281.359		0.000	0.000	0.000
max	2.628	254.558	281.377		61.851	0.351	7.125
SDLS	1.305% ie 0.000	0.000% ie 0.000	0.000		1.361	2.348%	2.773%
min	2.628	180.000	281.406		0.000	0.000	0.000
max	2.628	254.558	281.418		61.519	0.350	4.328

Table 17: Circular trajectory of 1m radius in the XZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	2.227% ie 0.000	0.000% ie 0.000	0.001		0.993	2.950%	3.132%
min	3.102	180.000	361.361		0.000	0.000	0.000
max	3.103	254.558	361.389		31.869	0.481	6.606
MSDLS	1.293% ie 0.000	0.000% ie 0.000	0.000		0.603	1.845%	1.956%
min	3.102	180.000	361.253		0.000	0.000	0.000
max	3.103	254.558	361.265		31.853	0.481	3.802
SDLS	1.976% ie 0.000	0.000% ie 0.000	0.000		1.026	2.997%	3.869%
min	3.102	180.000	360.641		0.000	0.000	0.000
max	3.103	254.558	360.653		31.809	0.481	3.022

Table 18: Circular trajectory of 1m radius in the YZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	1.337% ie 0.000	0.000% ie 0.000	0.001		0.426	1.809%	1.654%
min	2.551	180.000	262.813		0.000	0.000	0.000
max	2.552	254.558	262.854		21.589	0.506	6.694
MSDLS	1.632% ie 0.000	0.000% ie 0.000	0.000		0.512	2.314%	2.129%
min	2.551	180.000	262.818		0.000	0.000	0.000
max	2.551	254.558	262.839		21.587	0.506	2.451
SDLS	1.228% ie 0.000	0.000% ie 0.000	0.000		0.403	1.798%	2.221%
min	2.551	180.000	263.119		0.000	0.000	0.000
max	2.551	254.558	263.139		21.628	0.506	1.426

Table 19: Circular trajectory of 2m radius in the XY plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	20.609% ie 0.003	0.011% ie 0.008	18.137		65.969	50.023%	52.268%
min	2.544	180.000	207.229		27.767	0.126	15.887
max	2.558	254.557	281.641		448.192	2.721	111.156
MSDLS	36.118% ie 0.016	0.226% ie 0.168	19.654		313.626	23.870%	4.217%
min	2.507	180.000	191.534		27.321	0.171	22.404
max	2.552	254.551	303.131		2531.803	6.533	1977.462
SDLS	10.876% ie 0.023	0.091% ie 0.068	42.534		149.944	17.343%	12.921%
min	2.376	180.000	225.014		23.748	0.287	14.221
max	2.587	254.558	359.983		1653.941	9.569	467.543

Table 20: Circular trajectory of 2m radius in the XZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	2.172% ie 0.000	0.000% ie 0.000	0.004		1.084	2.387%	3.142%
min	3.455	180.000	276.273		0.000	0.000	0.000
max	3.456	254.558	276.357		24.531	0.427	6.893
MSDLS	1.748% ie 0.000	0.000% ie 0.000	0.001		1.218	2.436%	2.068%
min	3.455	180.000	314.602		0.000	0.000	0.000
max	3.456	254.558	314.663		42.700	0.421	7.048
SDLS	1.602% ie 0.000	0.000% ie 0.000	0.001		1.105	2.530%	3.296%
min	3.455	180.000	316.968		0.000	0.000	0.000
max	3.456	254.558	317.003		39.579	0.424	3.095

Table 21: Circular trajectory of 2m radius in the YZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	99.937% ie 0.122	0.229% ie 0.165	2.689		2.018	2.970%	3.212%
min	2.726	180.000	487.655		1.657	0.000	0.007
max	2.847	252.144	497.386		17.826	0.449	3.504
MSDLS	62.564% ie 0.066	2.110% ie 0.808	0.502		2263.810	12.250%	11.061%
min	2.779	180.000	484.312		0.227	0.000	0.097
max	2.884	218.279	488.297		20772.849	299.327	23797.906
SDLS	99.961% ie 0.119	0.501% ie 0.348	0.293		0.674	2.717%	3.186%
min	2.728	180.000	235.867		0.919	0.000	0.006
max	2.847	249.453	236.883		15.426	0.453	1.836

Table 22: Circular trajectory of 3m radius (unreachable) in the XY plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	99.996% ie 0.111	0.227% ie 0.164	0.420		1.252	2.817%	3.195%
min	2.736	180.000	357.987		0.014	0.000	0.000
max	2.847	252.152	359.768		15.726	0.317	4.659
MSDLS	99.881% ie 0.112	0.289% ie 0.209	0.159		489.037	13.300%	12.082%
min	2.736	180.000	359.017		0.037	0.000	0.000
max	2.847	252.213	360.427		4143.995	14.823	3987.039
SDLS	99.980% ie 0.108	0.464% ie 0.324	1.659		1.413	3.245%	3.511%
min	2.739	180.000	351.581		1.111	0.000	0.004
max	2.847	249.867	357.454		15.360	0.319	3.077

Table 23: Circular trajectory of 3m radius (unreachable) in the XZ plane

	NRMSE+RMSE position [% & m]	NRMSE+RMSE attitude [% & °]	RMSE ||q||		RMSE $||\dot{q}||$	NRMSE speed linear	NRMSE speed rotational
DLS	99.167% ie 0.375	0.150% ie 0.108	0.529		12.429	3.497%	4.625%
min	3.654	180.000	381.416		0.000	0.000	0.000
max	4.032	251.993	383.687		102.044	0.510	6.406
MSDLS	88.110% ie 0.135	18.255% ie 1.321	1.316		2017.890	11.317%	10.790%
min	3.879	172.765	366.599		2.356	0.001	3.233
max	4.032	180.000	374.114		19065.719	121.820	19200.171
SDLS	98.022% ie 0.351	0.525% ie 0.362	17.285		214.274	90.815%	87.924%
min	3.674	180.000	393.286		3992.226	93.230	1793.336
max	4.032	248.888	448.212		4661.262	107.567	2236.540

Table 24: Circular trajectory of 3m radius (unreachable) in the YZ plane

References

[1] Northern Sky Research. New NSR report projects over 1,600 satellites worth $250 billion to be launched over the next 15 years. Technical report, NSR, June 2011. Link to the article on www.nsr.com.

[2] Reto Wiesendanger. Clean-me: concept for an orbital debris removal mission : Preliminary evaluation. Seed-Money Activity, 2010. EPFL.

[3] Manuel Klein. Attitude determination and control system for a orbital debris removal spacecraft. Master Thesis, 2010. EPFL.

[4] Simon Hyde. Approach and control of uncooperative orbital debris. ESTEC (ESA) draft, 2010.

[5] ESA & NASA. *Active Debris Removal with Electrodynamic Tethers*, June 2010. 1st European Workshop on Active Debris Removal, CNES HQ (Paris).

[6] H. Klinkrad and N. L. Johnson. Space debris environment remediation concepts. In *European Space Agency, (Special Publication) ESA SP*, volume 672 SP, 2009.

[7] J. C. Liou, N. L. Johnson, and N. M. Hill. Controlling the growth of future leo debris populations with active debris removal. *Acta Astronautica*, 66(5-6):648–653, 2010.

[8] ETSIA & UPM. *Active Debris Removal with Electrodynamic Tethers*, June 2010. 1st European Workshop on Active Debris Removal, CNES HQ (Paris).

[9] C. Bonnal, M. Sanchez, and W. Naumann. Ariane Debris Mitigation Measures. In B. Kaldeich-Schuermann, editor, *Second European Conference on Space Debris*, volume 393 of *ESA Special Publication*, pages 681–+, 1997.

[10] Cranfield University. *Debris Removal in Low Earth Orbit*, June 2010. 1st European Workshop on Active Debris Removal, CNES HQ (Paris).

[11] I. Rekleitis, E. Martin, G. Rouleau, R. L'Archeveque, K. Parsa, and E. Dupuis. Autonomous capture of a tumbling satellite. *Journal of Field Robotics*, 24(4):275–296, 2007.

[12] Kazuya Yoshida and Brian Wilcox. Space robots and systems. In *Springer Handbook of Robotics*, pages 1031–1063. 2008.

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

[14] Mishap Investigation Board (MIB). Overview of the dart mishap investigation results. Accident report, NASA, May 2006.

[15] Boeing. Orbital express - program summary and program flight results. Technical report, Boeing, 2007.

[16] S. Persson, P. Bodin, E. Gill, J. Harr, and J. Jörgensen. PRISMA - an autonomous formation flying mission. In ESA SP, editor, *European Space Agency, (Special Publication) ESA SP*, volume 625 SP, 2006.

[17] DLR (German Space Agency). TECSAS/DEOS mission. Website, 2011.

[18] Y. Xu and T. Kanade, editors. *Space Robotics: Dynamics and Control*. Kluwer Academic Publishers, 1993.

[19] K. Yoshida and Y. Umetani. *Space Robotics: Dynamics and Control*, chapter Control of Space Manipulators with Generalized Jacobian Matrix. Kluwer Academic Publishers, 1993.

[20] Dragomir Nenchev, Yoji Umetani, and Kazuya Yoshida. Analysis of a redundant free-flying spacecraft/manipulator system. *IEEE Transactions on Robotics and Automation*, 8(1):1–6, 1992.

[21] Sylvain Gallay. Clean-me: Orbital debris removal mission : Preliminary mechanical concept evaluation. Semester Thesis, 2010. EPFL.

[22] Francois Caullier. DR.LEO - AOCS and rendezvous. Master's thesis, Cranfield University, 2010.

[23] Arianespace. *ARIANE 4: user's manual*. Arianespace, February 1999.

[24] Eric Sauser. RobotToolKit, an open-source robot simulator developed for researchers and robot hackers. Freely distributed under the terms of the GNU General Public Licence by Eric Sauser, 2011. Developed at the LASA, EPFL. See also `http://lasa.epfl.ch/RobotToolKit/`.

[25] Singular value decomposition (SVD) and generalized singular value decomposition (GSVD). Wikipedia article. `http://en.wikipedia.org/wiki/Singular_value_decomposition`.

[26] Samuel R Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Buss personal webpage `http://math.ucsd.edu/~sbuss/ResearchWeb/index.html`, October 2009.

[27] Samuel R Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, October 2005.

[28] I. Y. S. LUH, M. W. WALKER, and R P. C. PAUL. Resolved acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, AC-25(3):468–474, June 1980.

[29] Christophe Tournier. Usinage des courbes et surfaces. Course about splines and surfaces drilling, October 2006. Process mechanics course, First year of M.Sc.

[30] Ken Shoemake. Animating rotation with quaternion curve. *SIGGRAPH '85*, 19(3):245–254, 1985.