GNC architecture solutions for robust operations of a free-floating space manipulator via image based visual servoing

C. Marchionne[a], M. Sabatini[b]*, P. Gasbarri[c]

[a] *Department of Mechanical and Aerospace Engineering, Sapienza Università di Roma, chiamrc@yahoo.it*

[b] *Department of Astronautics, Electric and Energetics, Sapienza Università di Roma, marco.sabatini@uniroma1.it*

[c] *Department of Mechanical and Aerospace Engineering, Sapienza Università di Roma, paolo.gasbarri@uniroma1.it*

* Corresponding Author

Abstract

On-orbit servicing often requires the use of robotic arms, and a key asset in this kind of operations is autonomy. In this framework, the use of optical devices is a solution, already analyzed in many researches both for autonomous rendezvous and docking and for the evaluation of the control of the manipulator. In the present paper, simulations for assessing the controller performance are realized in a high-fidelity purposely developed software architecture, in which not only the selected 6 DOF space manipulator is modeled, but also a virtual camera, acquiring in the loop images of the target CAD model imported, is included in the GNC loop. This approach allows to emphasis several problems that would not emerge in simulations with images characterized by easily-identifiable, purposely-created markers. At the scope, a specific GNC architecture is developed, based on finite-state machine logic. According to this approach, two different Image Based Visual Servoing strategies are alternatively performed, commanding only linear or angular velocity of the camera, switching between the two control techniques when the “stack” or “divergence” condition is triggered. In this way a stable and robust accomplishment of the tasks is achieved for many configurations and for different target models.

Keywords: image based visual servoing; space manipulators; on-orbit servicing; advanced software simulations

1. Introduction

A class of advanced space activities, including on-orbit servicing and debris removal [1] , often requires the use of robotic arms. Autonomy of these operations is a key asset, since the necessary continuous monitoring of the scenario and real-time control cannot be provided by a ground station. In this framework, the use of optical devices for computing the arm control is a solution, already analyzed in many researches. Image Based Visual Servoing (IBVS) [2] is a control strategy that makes use of images (comparing the acquired image with the reference one) to compute the desired end-effector velocity, without the need of reconstructing the pose of the target object [3]. The final goal is reached when the acquired and reference images coincide, meaning that the end-effector has moved to the correct position. An early example of this technique applied to terrestrial manipulators can be found in [4].

Space manipulators present special features with respect to terrestrial applications. The movement of the robotic arm could lead to an excessive rotation of the platform, since it is unconstrained [5]. This problem has been already faced by many authors. In [6] an experimental set-up is developed to test applications specific to the space environment. In [7] a 3D model-based tracking algorithm has been studied and tested on a mock-up of a telecommunication satellite, using a 6-DOF robotic arm, with satisfactory results, in terms of precision of the pose estimation and computational costs, for a rendezvous mission. A visual servoing controller for a satellite mounted dual-arm space robot is proposed in [8]. The controller is designed to complete the task of servoing the robot's end-effectors to the desired pose, while regulating orientation of the base-satellite. Reference [9] presents an optimal control approach to guiding the free-floating satellite-mounted robot, using visual information and considering the optimization of the motor commands with respect to a specified metric along with chaos compensation.

In the great majority of the papers dealing with simulation of visual servoing, an important simplification is present: the image is not acquired, but reduced to some feature objects (points, curves, blobs…). Even in the case that experiments are performed, the target bodies are often characterized by easily identifiable markers, thus reducing to a quasi-ideal case. This approach leads to a significant over-estimate of the controller performance. In the present paper, simulations are performed in a high-fidelity purposely developed software architecture, in which not only the selected 6 DOF space manipulator is modeled, but also a virtual camera, acquiring images of the target CAD model imported in MATLAB Virtual Reality toolbox, is included in the GNC loop. This approach allows to emphasize several problems that would not emerge in simulations with ideal images (made of set of points or primitive shapes, as typically done). Errors on the image feature extraction and possible mismatches between reference and actual features heavily decrease the arm performance, which could be either stacked in a local minimum or even diverge. We have realized that computing the linear and angular velocity of the end-effector within a single computation, as usually performed, increases this undesired behavior. Therefore, a specific GNC architecture is developed, based on finite-state machine logic. According to this approach, two different IBVS strategies are alternatively performed, commanding only linear or angular velocity of the camera, switching between the two control techniques when the "stack" or "divergence" condition is triggered. In this way a stable and robust accomplishment of the tasks is achieved for many configurations and for different target models.

The remainder of the paper is organized as follows: in Section 2 the dynamics of the space manipulator are described. In Section 3 the main steps for simulating the IBVS technique (i.e. the camera model, the interaction matrix computation and control law evaluation) are reported. In Section 4 the results of the classic IBVS implementation for an ideal case (i.e. with ideal features) are compared with the results coming from the advanced simulation set-up, including rendered images. In Section 5 the algorithm that has been developed to overcome all the problems emerged in realistic simulations is described, and relevant result are shown. Final remarks can be found in Section 6.

2. Scenario and Dynamics

The on-orbit operation considered in this work is relevant to an inspection mission of a robotic arm, mounted on a satellite and composed by 6 links connected through revolute joints, with the task to acquire a certain configuration with respect to a target. This could be the case of a malfunctioning satellite that must be repaired, or a debris that must be grasped before removal. In both cases, the manipulator must be moved in a way that it reaches a certain desired position with respect to the target spacecraft. The scenario considered in this paper consists in a chaser satellite that, for avoiding interactions between robotic arm control and attitude control, will de-activate attitude control during arm motion.

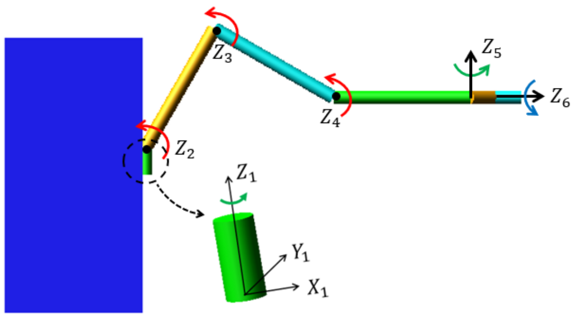

Fig. 1 The manipulator and the associated reference frames

For computing the dynamic equations of the space manipulator, three reference frames can be introduced:

- (O_I, X_I, Y_I, Z_I): inertial reference frame.
- (O_B, X_B, Y_B, Z_B): body reference frame of the bus centered in the spacecraft centre of mass with axes parallel to the fixed frame at simulation time $t_0 = 0$
- (O_i, X_i, Y_i, Z_i) with $i = 1, \dots, 6$: body reference frame of i-th link with origin in the i-th joint.

The coordinate vector describing the state of each single body is:

$$\vec{X} = [\vec{x}_B \quad \vec{q}_B \quad \vec{x}_1 \quad \vec{q}_1 \quad \dots \quad \vec{x}_i \quad \vec{q}_i \quad \dots \quad \vec{x}_6 \quad \vec{q}_6]^T$$

where:

- $\vec{x}_B$ is the position vector of main bus center of mass written in the inertial reference frame
- $\vec{x}_i$ is the position vector of i-th joint written in the inertial reference frame
- $\vec{q}_B$ and $\vec{q}_i$ with $i = 1, \dots, 6$ are the quaternions that define bus and links attitude with respect to inertial reference frame.

The minimum set of (lagrangian) coordinates of the system is:

$$\vec{Q} = [\vec{x}_B \quad \vec{q}_B \quad \theta_{j1} \quad \theta_{j2} \quad \theta_{j3} \quad \theta_{j4} \quad \theta_{j5} \quad \theta_{j6}]^T$$

where the base position vector and the base orientation vector are written in the inertial reference frame while θ_{ji} is the i-th joint angle calculated with respect to the (i-1)-th link body reference frame.

The rotation axis direction of i-th joints written in the inertial reference frame is:

$$^{I}\hat{z}_{ji} = {}^{I}\underline{\underline{R}}_{bi}\, {}^{bi}\hat{z}_{ji} \qquad (1)$$

where $^{I}\underline{\underline{R}}_{bi}$ is the rotation matrix from the i-th link body reference frame to the inertial reference frame and $^{bi}\hat{z}_{ji}$

is the unit vector identifying the direction of the i-th revolute joint rotation axis in the body reference frame. For all the revolute joints the following rotation axis is chosen:

$$^{bi}\hat{z}_{ji} = [0 \quad 0 \quad 1]^T \quad i = 1, \dots, 6$$

It's possible to get the motion equations for complex multibody systems with numerous degrees of freedom using Kane's method [10]. It allows to obtain the least number of differential equations that completely describe the multibody system without considering the forces due to constraints.

The dynamic equations of the multibody system written according to Kane approach are:

$$\underline{\underline{J}}^T \underline{\underline{M}}\, \underline{\underline{J}}\, \ddot{\vec{Q}} + \underline{\underline{J}}^T \underline{\underline{M}}\, \dot{\underline{\underline{J}}}\, \dot{\vec{Q}} = \underline{\underline{J}}^T \vec{F} + \underline{\underline{J}}^T \vec{C} + \vec{\tau} \qquad (2)$$

where:

- $\underline{\underline{M}}$ is the mass matrix of the system

$$\underline{\underline{M}} = \begin{bmatrix} \underline{\underline{M}}_B & \underline{\underline{0}}_{6\times6} & \cdots & \cdots & \cdots & \underline{\underline{0}}_{6\times6} \\ \underline{\underline{0}}_{6\times6} & \underline{\underline{M}}_1 & & & & \vdots \\ \vdots & & \ddots & & & \vdots \\ \vdots & & & \underline{\underline{M}}_i & & \vdots \\ \vdots & & & & \ddots & \underline{\underline{0}}_{6\times6} \\ \underline{\underline{0}}_{6\times6} & \cdots & \cdots & \cdots & \underline{\underline{0}}_{6\times6} & \underline{\underline{M}}_6 \end{bmatrix}$$

$$\underline{\underline{M}}_i = \begin{bmatrix} \underline{\underline{m}}_i & \underline{\underline{m}}_i\, \underline{\tilde{r}}_{CMi}^T \\ \underline{\underline{m}}_i\, \underline{\tilde{r}}_{CMi} & {}^I\underline{\underline{J}}_{O_i} \end{bmatrix}$$

m_i is the mass and ${}^I\underline{\underline{J}}_{O_i}$ the inertia matrix of the i-th body, whereas $\vec{r}_{CMi}$ is the position vector of the body center of mass with respect to the origin of the body reference frame written in the inertial frame. $\underline{\tilde{r}}_{CMi}$ represents the skew-symmetric form of the vector $\vec{r}_{CMi}$:

$$\underline{\tilde{r}}_{CMi} = \begin{bmatrix} 0 & -z_{CMi} & y_{CMi} \\ z_{CMi} & 0 & -x_{CMi} \\ -y_{CMi} & x_{CMi} & 0 \end{bmatrix}$$

- $\vec{C}$ is the Coriolis and centrifugal forces vector

$$\vec{C} = [\vec{C}_B \quad \vec{C}_1 \quad \dots \quad \vec{C}_i \quad \dots \quad \vec{C}_6]^T$$

$$\vec{C}_i = \begin{bmatrix} -\underline{\underline{m}}_i \underline{\underline{\tilde{\omega}}}_i \underline{\underline{\tilde{\omega}}}_i\, \vec{r}_{CMi} \\ -\underline{\underline{\tilde{\omega}}}_i\, {}^I\underline{\underline{J}}_{O_i}\, \vec{\omega}_i \end{bmatrix}$$

$\vec{\omega}_i$ is the angular velocity of i-th body, and $\underline{\underline{\tilde{\omega}}}_i$ the skew-symmetric form of the vector $\vec{\omega}_i$:

$$\underline{\underline{\tilde{\omega}}}_i = \begin{bmatrix} 0 & -\omega_{i,z} & \omega_{i,y} \\ \omega_{i,z} & 0 & -\omega_{i,x} \\ -\omega_{i,y} & \omega_{i,x} & 0 \end{bmatrix}$$

- $\vec{F}$ is the external forces vector
- $\vec{\tau}$ is the vector of control torques applied to the floating base and to the joints
- $\underline{\underline{J}}$ is the Jacobian matrix that relates the vectors $\vec{X}$ and $\vec{Q}$:

$$\dot{\vec{X}} = \underline{\underline{J}}\, \dot{\vec{Q}} \qquad (3)$$

The geometric relations between $\vec{X}$ and $\vec{Q}$ can be written and derived with respect to time to calculate $\underline{\underline{J}}$. In general form, the time derivative of i-th joint position and the angular velocity of i-th link can be written as follows:

$$^I\dot{\vec{x}}_i = {}^I\dot{\vec{x}}_B - \left({}^I\underline{\underline{\tilde{D}}}\right) {}^I\overrightarrow{\omega_B} - \sum_{k=1}^{i-1} {}^I\underline{\underline{\tilde{l}_k}}\; {}^I\vec{\omega}_k \qquad (4)$$

$$^I\vec{\omega}_i = {}^I\overrightarrow{\omega_B} + \sum_{k=1}^{i} {}^I\underline{\underline{R}}_{bk}\; {}^{bk}\hat{z}_{jk}\; \dot{\theta}_{jk} \qquad (5)$$

with $i = 1, \dots, 6$.

${}^I\vec{D}$ is the vector from base reference frame to first joint reference frame while ${}^I\vec{l_\iota}$ is the vector from i-th joint to (i+1)-th joint written in inertial reference frame, where ${}^I\underline{\underline{R}}_{bk}$ is the rotation matrix from the k-th link body reference frame to the inertial reference frame; ${}^I\underline{\underline{\tilde{D}}}$ and $\underline{\underline{\tilde{l}_\iota}}$ represent the skew-symmetric form of the vector ${}^I\vec{D}$ and ${}^I\vec{l_\iota}$, respectively:

$$^I\underline{\underline{\tilde{D}}} = \begin{bmatrix} 0 & -{}^ID_z & {}^ID_y \\ {}^ID_z & 0 & -{}^ID_x \\ -{}^ID_y & {}^ID_x & 0 \end{bmatrix}$$

$$^I\underline{\underline{\tilde{l}_\iota}} = \begin{bmatrix} 0 & -{}^Il_{i,z} & {}^Il_{i,y} \\ {}^Il_{i,z} & 0 & -{}^Il_{i,x} \\ -{}^Il_{i,y} & {}^Il_{i,x} & 0 \end{bmatrix}$$

Using these equations allows to write the expression for the Jacobian matrix.

3. Image Based Visual Servoing (IBVS)

The task of moving the manipulator is realized using the IBVS technique, which requires, of course, the presence of a camera, acquiring the image of the target and extracting the most important information from it, so that the control law can be computed.

3.1 The camera

A camera is a system that can reproduce on an image the portion of space in its field of view through a projection. The simplest model of camera is the basic pinhole model [16].

Consider a point in the 3D space:

$$P = (x, y, z)$$

whose coordinates are written in a frame (O_C, X_C, Y_C, Z_C) attached to the camera and consider its projection on a plane called *image plane I*:

$$p = (u, v)$$

whose coordinates are written in a frame attached to the plane and with the axis u and v parallel to X_C and Y_C.

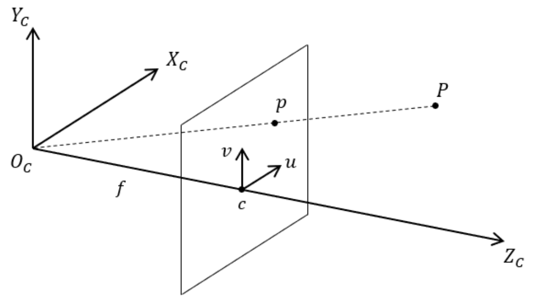

Fig. 2 Basic pinhole model of camera

All the rays meet in a single point, called *optical center* O_C .The plane I is placed in $Z = f$ from the optical center, where f is the focal length of camera. $\hat{Z}_C$, named *optical axis*, is the straight line passing through the optical center and perpendicular to the image plane. The point c is the projection of optical center on image plane and is called *principal point*.

The point in the camera frame is transformed into a point in the image plane via the perspective transformation:

$$\begin{cases} u = f\dfrac{x}{z} \\ v = f\dfrac{y}{z} \end{cases} \qquad (6)$$

This nonlinear transformation can be written in a linear form by resorting to the homogeneous representation of the points:

$$p = \begin{bmatrix} u \\ v \end{bmatrix} \rightarrow \tilde{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \tilde{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The relations (6) become:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fx/z \\ fy/z \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underline{\underline{\Omega}} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (7)$$

where $\underline{\underline{\Omega}}$ is the camera calibration matrix that connects the homogeneous coordinates of image plane $\tilde{p}$ and the coordinates of point P in the camera frame.

Eq.(7) can be written considering also the homogeneous coordinates of P :

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underline{\underline{\Omega}}\,\underline{\underline{\Pi}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \underline{\underline{Q}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (8)$$

$\underline{\underline{Q}}$ is the perspective projection matrix that connects the homogeneous coordinates of image plane $\tilde{p}$ and the homogeneous coordinates $\tilde{P}$ in the camera reference frame.

An infinite number of points exist in an image plane, so a spatial sampling is needed. The spatial sampling unit is the pixel and thus the coordinates (u, v) of a point in the image plane are to be expressed in pixel, (u_I, v_I).

The pixel coordinates of the point are related to coordinates in metric units through two scale factors α_x and α_y:

$$\begin{cases} u_I = \alpha_x\, u \\ v_I = \alpha_y\, v \end{cases}$$

where:

$$\begin{cases} \alpha_x = 1/\Delta u \quad [pixel/m] \\ \alpha_y = 1/\Delta v \quad [pixel/m] \end{cases}$$

Δu and Δv are respectively the horizontal and vertical size of a pixel.

Moreover, the image reference frame (X_{im}, Y_{im}) is translated and rotated with respect to the frame (u_I, v_I) with origin in the principal point, so it is necessary to perform a coordinates transformation:

$$\begin{cases} X_{im} = u_0 - u_I = u_0 - \alpha_x\, f\; x/z \\ Y_{im} = v_0 - v_I = v_0 - \alpha_y\, f\; y/z \end{cases} \qquad (9)$$

where $c = (u_0, v_0)$ are the coordinates of principal point written in the image reference frame.

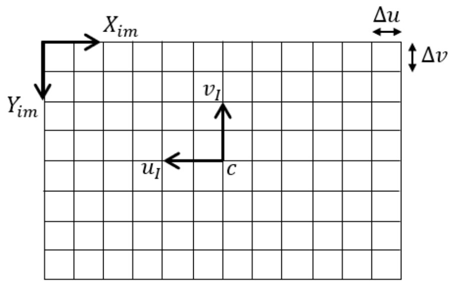

Fig. 3 Image Reference Frame

The perspective projection matrix $\underline{\underline{Q}}$ becomes:

$$\underline{\underline{Q}} = \begin{bmatrix} -\alpha_x f & 0 & u_0 & 0 \\ 0 & -\alpha_y f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad (10)$$

P is a point of the target that the manipulator wants to capture. Generally, the position vector of the point P is written in the reference frame attached to the target (O_O, X_O, Y_O, Z_O), called object reference frame. So, it is needed to realize a roto-translation from object reference frame to camera reference frame:

$${}^{C}\vec{r}_P = {}^{C}\underline{\underline{R}}_O\, {}^{O}\vec{r}_{P/O_O} + {}^{C}\vec{r}_O \qquad (11)$$

where ${}^{C}\underline{\underline{R}}_{O}$ is the rotation matrix from the object reference frame to camera reference frame and ${}^{C}\vec{r}_{O}$ is the position vector of object reference frame origin with respect to camera refence frame origin, written in camera reference frame.

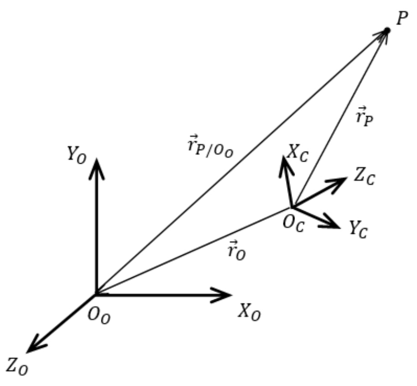

Fig. 4 Object and camera reference frame

Now, it is possible to write the relationship between the target's point in homogeneous coordinate written in the object reference frame and the its projections on the image plane.

$$\begin{bmatrix} X_{im} \\ Y_{im} \\ 1 \end{bmatrix} = \underline{\underline{Q}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_C = \underline{\underline{Q}}\, {}^{C}\underline{\underline{T}}_{O} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_O = \underline{\underline{\Sigma}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_O \qquad (12)$$

where

$${}^{C}\underline{\underline{T}}_{O} = \begin{bmatrix} {}^{C}\underline{\underline{R}}_{O} & {}^{C}\vec{r}_{O} \\ \underline{\underline{0}}_{3\times 3} & 1 \end{bmatrix}$$

is the homogeneous transformation matrix that allows to pass from the homogeneous coordinates written in the object frame to the homogeneous coordinates written in the camera frame.

The matrix $\underline{\underline{\Sigma}}$ contains all the parameters that characterize the camera. In particular, two kinds of parameters can be recognized:

- The intrinsic parameters that depend on the sensor's characteristics.
- The extrinsic parameters that depend on the camera relative position with respect to the object reference frame.

3.2 *Interaction matrix*

Consider a point in the 3D space:

$${}^{C}P = (x, y, z)$$

whose coordinates are expressed in camera reference frame (see Fig. 5).

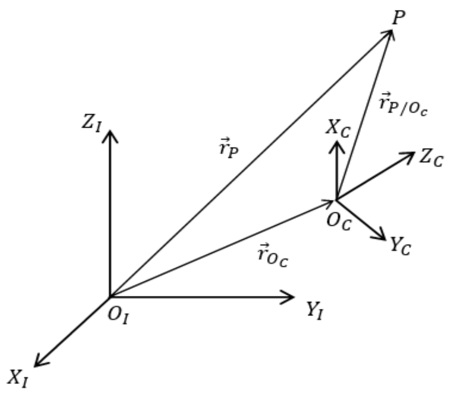

Fig. 5 Inertial and camera reference frame

The point's projection on the image plane is given by the following relations:

$$\begin{cases} u = f\dfrac{x}{z} = -X_{im} + u_O \\ v = f\dfrac{y}{z} = -Y_{im} + v_O \end{cases} \qquad (13)$$

where f is the focal length of camera in pixels. Deriving Eq. (13) with respect to time, it is found:

$$\begin{cases} \dot{u} = f\dfrac{\dot{x}\,z - x\dot{z}}{z^2} = f\dfrac{\dot{x}}{z} - u\dfrac{\dot{z}}{z} \\ \dot{v} = f\dfrac{\dot{y}z - y\dot{z}}{z^2} = f\dfrac{\dot{y}}{z} - v\dfrac{\dot{z}}{z} \end{cases} \qquad (14)$$

where

$${}^{C}\dot{\vec{r}}_{P/O_C} = [\dot{x} \quad \dot{y} \quad \dot{z}]^T$$

is the relative velocity vector of point P with respect to the camera, written in camera reference frame.

Eq. (14) can be written in matrix form:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \dfrac{f}{z} & 0 & -\dfrac{u}{z} \\ 0 & \dfrac{f}{z} & -\dfrac{v}{z} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \underline{\underline{J}}_1 \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \qquad (15)$$

The absolute velocity vector of point P written in camera reference frame is the following:

$${}^{C}\dot{\vec{r}}_P = {}^{C}\dot{\vec{r}}_{O_C} + {}^{C}\dot{\vec{r}}_{P/O_C} + {}^{C}\vec{\omega}_{O_C} \wedge {}^{C}\vec{r}_{P/O_C} \qquad (16)$$

${}^{C}\dot{\vec{r}}_{O_C}$ and ${}^{C}\vec{\omega}_{O_C}$ are the linear and angular velocity of the camera, written in camera reference frame. Here and in the following, symbol "$\wedge$" represent the cross product.

Eq. (16) can be written in extended form as:

$${}^{C}\dot{\vec{r}}_P = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \wedge \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (17)$$

In this paper, the interaction matrix and relevant IBVS is developed under the hypothesis of fixed target; it is true that a residual relative velocity is always present between chaser and target. However, the focus of the paper is on the serious performance degradation caused by the processing of "real" images with respect to the ideal performance of the control. In real cases, the target velocity will be not null, but actually very low, therefore the residual motion of the target is considered a negligible disturbance in the following simulations:

$$^{C}\dot{\vec{r}}_P = 0$$

Hence, Eq. (17) becomes

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = -\begin{bmatrix}v_x\\ v_y\\ v_z\end{bmatrix} - \begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix} \wedge \begin{bmatrix}x\\ y\\ z\end{bmatrix} = -\begin{bmatrix}v_x\\ v_y\\ v_z\end{bmatrix} + \begin{bmatrix}x\\ y\\ z\end{bmatrix} \wedge \begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}$$

In matrix form:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}-1 & 0 & 0 & 0 & -z & y\\ 0 & -1 & 0 & z & 0 & -x\\ 0 & 0 & -1 & -y & x & 0\end{bmatrix} {}^{C}\vec{v}_C$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = \underline{\underline{J_2}}\ {}^{C}\vec{v}_C \qquad (18)$$

where ${}^{C}\vec{v}_C$ is a 6×1 vector that contains both linear and angular velocity of the camera, written in camera reference frame.

Combining Eq. (15) and Eq. (18), the relationship between the time variation of the feature coordinates on image plane and the camera velocity vector is obtained:

$$\begin{bmatrix}\dot{u}\\ \dot{v}\end{bmatrix} = \underline{\underline{J_1}}\,\underline{\underline{J_2}}\ {}^{C}\vec{v}_C = \underline{\underline{J_{int}}}\ {}^{C}\vec{v}_C \qquad (19)$$

with:

$$\underline{\underline{J_{int}}} = \begin{bmatrix}-\frac{f}{z} & 0 & \frac{u}{z} & \frac{u\,v}{f} & -\left(f + \frac{u^2}{f}\right) & v\\ 0 & -\frac{f}{z} & \frac{v}{z} & f + \frac{v^2}{f} & -\frac{u\,v}{f} & -u\end{bmatrix}$$

$\underline{\underline{J_{int}}}$ can be written for each point detected on the image. Then, a system of $2k$ equations (being k the number of considered feature points) in 6 unknow variables is found:

$$\begin{bmatrix}\dot{u}_1\\ \dot{v}_1\\ \vdots\\ \dot{u}_k\\ \dot{v}_k\end{bmatrix} = \begin{bmatrix}\underline{\underline{J_{int}^1}}\\ \vdots\\ \underline{\underline{J_{int}^k}}\end{bmatrix} {}^{C}\vec{v}_C = \underline{\underline{L_s}}\ {}^{C}\vec{v}_C \qquad (20)$$

$\underline{\underline{L_s}} \in \mathbb{R}^{2k\times 6}$ is the interaction matrix that depends not only on the features $\vec{s} = [u \quad v]^T$ but also depends on the coordinates z of points with respect to the camera:

$$\underline{\underline{L_s}} = \underline{\underline{L_s}}\,(\vec{s}, {}^{C}\vec{z})$$

where

$${}^{C}\vec{z} = [z_1 \quad \cdots \quad z_k]$$

Usually, ${}^{C}\vec{z}$ is not known so an estimated value or a constant value can be used, for example the depth value in the initial configuration or that in the desired pose. Using an approximate value for the point's depth is equal to use an estimate of interaction matrix $\underline{\underline{\hat{L}_s}}$.

3.3 IBVS control

The purpose of visual servoing control scheme is to minimize an error that is defined as:

$$\vec{e}(t) = \vec{s}(t) - \vec{s}^* \qquad (21)$$

where $\vec{s}(t)$ is the $2k \times 1$ vector containing the feature parameters of the current image while $\vec{s}^*$ is the $2k \times 1$ vector containing the desired feature parameters.

The features are some specific points detected in the image. Hence, extracting an i-th point (X_i, Y_i) from the acquired image at time t and considering the coordinates transformation (9), the features vector associated to this point is:

$$\vec{s}_i = \begin{bmatrix}u_i\\ v_i\end{bmatrix}$$

So, if k points are detected then the features vector is given by:

$$\vec{s} = \begin{bmatrix}\vec{s}_1\\ \vdots\\ \vec{s}_k\end{bmatrix}$$

Deriving the features vector with respect to time, Eq. (20) can be written as follow:

$$\dot{\vec{s}} = \underline{\underline{L_s}}\ {}^{C}\vec{v}_C \qquad (22)$$

Usually, the system of equations (22) is overdetermined ($2k > 6$) and therefore the matrix $\underline{\underline{L_s}}$ is not square and the inverse cannot be calculated. Eq. (22) can be solved using least-squares technique, whose solution is given by:

$${}^{C}\vec{v}_C = \underline{\underline{L_s^+}}\ \dot{\vec{s}} \qquad (23)$$

where $\underline{\underline{L_s^+}}$ is the pseudo-inverse matrix defined as:

$$\underline{\underline{L_s^+}} = \left(\underline{\underline{L_s^T}}\,\underline{\underline{L_s}}\right)^{-1}\underline{\underline{L_s^T}}$$

Within the hypothesis that the target is fixed, the time derivative of the error is directly related to the velocity of the image points as follows:

$$\dot{\vec{e}}(t) = \dot{\vec{s}}(t)$$

The control actions can be now evaluated, for instance, by imposing an exponential decrease of the error, through an assigned gain matrix $\underline{\underline{K_s}}$:

$$\dot{\vec{e}}(t) = \dot{\vec{s}}(t) = -\underline{\underline{K_s}}\,\vec{e} \qquad (24)$$

Combining Eq. (22) and Eq. (24), it results as follows:

$$-\underline{\underline{K_s}}\,\vec{e} = \underline{\underline{L_s}}\,{}^{C}\vec{v}_C$$

In the case $2k > 6$, the solution is the following:

$$^{C}\vec{v}_C = -\underline{\underline{K_s}}\,\underline{\underline{L_s^{+}}}\,\vec{e} \qquad (25)$$

The camera is mounted on the end effector of the manipulator and therefore the desired camera's velocity coincides with the desired end effector's velocity:

$$\vec{v}_C^{\,des} = \vec{v}_{EE}^{\,des}$$

The motion of the end effector needs to be related to the commands provided to the actuators ("internal control" in Fig. 6), depending on the specific architecture of the manipulator. It is necessary to establish how the torques that must be applied to the joints vary over time to reach the desired behavior of the manipulator. The easiest command to execute is proportional in each instant to the error between the current joint angular velocity $\dot{\vec{\theta}}_j$ and the desired joint angular velocity $\dot{\vec{\theta}}_j^{des}$. So, it can be seen as a purely derivative control, even if $\dot{\vec{\theta}}_j^{des}$ is at end dependent on the end effector position and attitude error.

$$\vec{\tau} = \begin{bmatrix} \underline{\underline{0}}_{6\times1} \\ -\underline{\underline{K_d}}\left(\dot{\vec{\theta}}_j - \dot{\vec{\theta}}_j^{des}\right) \end{bmatrix} \qquad (26)$$

where $\underline{\underline{K_d}}$ is the gain diagonal matrix.

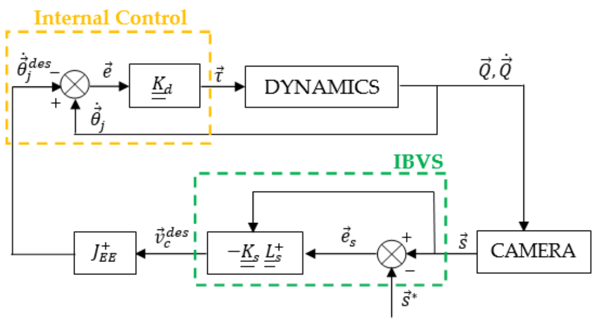

Fig. 6 IBVS control loop scheme

IBVS provides as output the velocity of the camera to reach the desired configuration of the manipulator rather than the desired angular velocity of the joints. So, it is necessary to establish the relation that connects end effector velocity and joint angular velocity.
The following relation can be written:

$$\vec{v}_{EE} = \begin{bmatrix} \underline{\underline{J_B}} & \underline{\underline{J_M}} \end{bmatrix} \begin{bmatrix} \dot{\vec{x}}_B \\ \vec{\omega}_B \\ \dot{\vec{\theta}}_j \end{bmatrix} \qquad (27)$$

where $\underline{\underline{J_B}}$ is the portion of Jacobian matrix referred to the linear and angular velocity of the base while $\underline{\underline{J_M}}$ is the portion referred to the angular velocity of the joints.
The geometric relationships between the position of end effector and the Lagrangian variables can be written and derived with respect to time to obtain $\underline{\underline{J_B}}$ and $\underline{\underline{J_M}}$ (the same procedure has been already done to obtain the matrix $\underline{\underline{J}}$, eq. (4) and eq. (5)).
The desired angular velocity of the revolute joints is easily found if the camera velocity is well-known from visual servoing control:

$$\vec{v}_{EE}^{\,des} - \underline{\underline{J_B}} \begin{bmatrix} \dot{\vec{x}}_B \\ \vec{\omega}_B \end{bmatrix} = \underline{\underline{J_M}}\,\dot{\vec{\theta}}_j^{des} \Rightarrow$$

$$\dot{\vec{\theta}}_j^{des} = \underline{\underline{J_M^{+}}} \left(\vec{v}_{EE}^{\,des} - \underline{\underline{J_B}} \begin{bmatrix} \dot{\vec{x}}_B \\ \vec{\omega}_B \end{bmatrix} \right) \qquad (28)$$

4 Straightforward IBVS application

4.1 Ideal simulations

A preliminary test of the proposed IBVS algorithm is strongly advisable, both to provide confidence in the approach and to define the dynamic behavior to be expected during the tests.
The numerical computing environment MATLAB has been used to execute the dynamical model of 3D manipulator and the control scheme IBVS. As a first study, only ideal cases have been considered in which the images processing is supposed already done.
The chosen target body is a cube and the features are the edges of one facet (see Fig. 7).

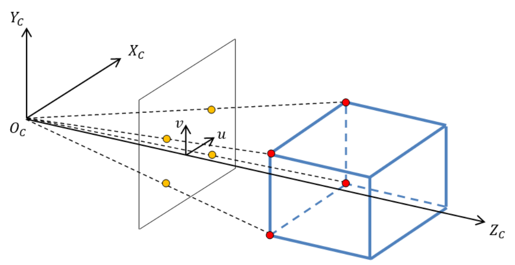

Fig. 7 Features of the ideal target

The coordinates of the target body and the initial position of the end-effector are introduced as input.
At $t = 0\,s$ the space system is at rest and the initial conditions of manipulator are the following:

$$\vec{x}_B = [-0.525 \quad 0 \quad -0.2]^T m$$

$$\vec{q}_B = [1 \quad 0 \quad 0 \quad 0]^T$$

$$\vec{\theta}_j = [0 \quad \pi/3 \quad -2\,\pi/3 \quad \pi/3 \quad 0 \quad 0]^T\ rad$$

Performing the required transformations, the relative position of the features, with respect to the camera written in the camera reference frame, can be obtained. Afterwards, the current and the desired final position of the pixels can be identified. Based on the error between the current and the desired position, the required joints angular velocities are evaluated, and the manipulator kinematics is propagated.
The gain matrixes $\underline{\underline{K_s}}$ and $\underline{\underline{K_d}}$ have been chosen after a tuning in such a way that the features error goes to zero in 60 seconds.

$$\underline{\underline{K_s}} = 0.16\ \underline{\underline{I}}_{6\times 6}$$

$$\underline{\underline{K_d}} = \begin{bmatrix} 200 & 0 & 0 & 0 & 0 & 0 \\ 0 & 200 & 0 & 0 & 0 & 0 \\ 0 & 0 & 200 & 0 & 0 & 0 \\ 0 & 0 & 0 & 200 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

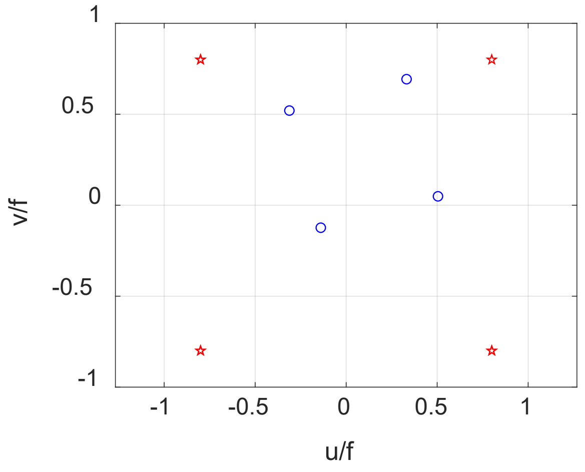

Fig. 8 Features in the desired position (red stars) and in the initial position (blue circles)

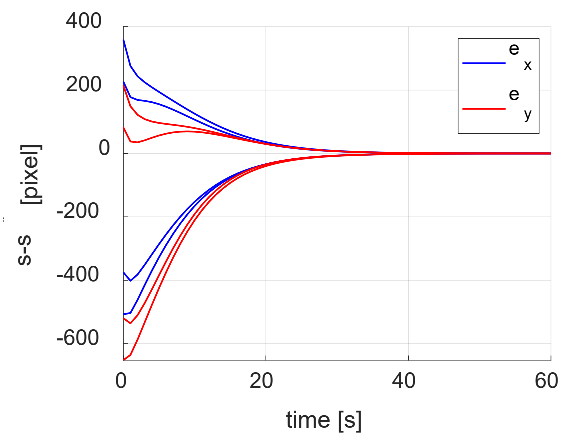

Fig. 9 Behavior of components x and y of error

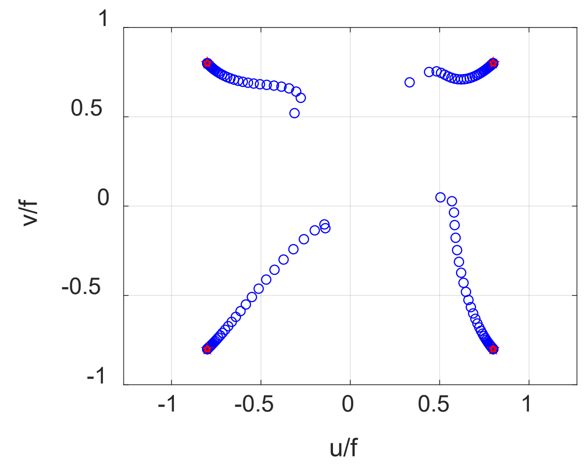

Fig. 10 Trajectories in the image plane of the features from the initial to the desired configuration

The features error converges to zero in almost 40 seconds, as shown in Fig. 9. Fig. 10 shows the trajectories of the points in the image plane: the features detected at each step move on the image plane until the desired location is reached.

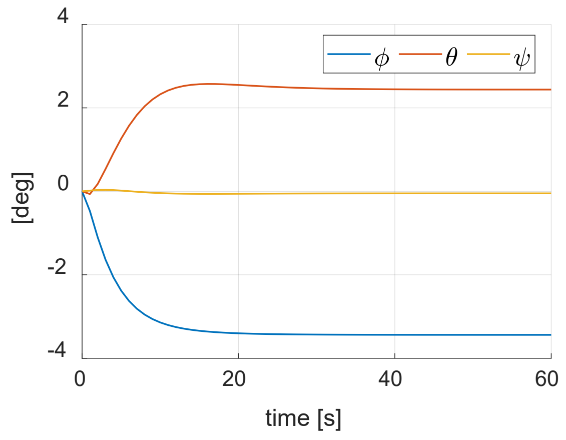

Fig. 11 Base Attitude

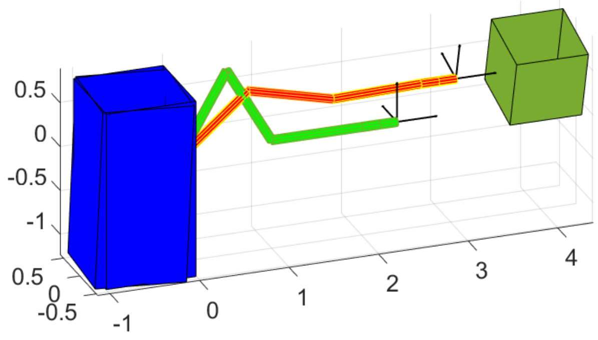

Fig. 12 Initial (green) and final (orange) configuration of the robotic arm

Fig. 11 shows the change in the satellite attitude over simulation time. Indeed, the movements of the robotic arm cause the rotation of main bus if the attitude control system is not in working. The variation in attitude can be seen in Fig. 12 as well, where the initial and final configuration of manipulator are illustrated.

4.2 Advanced Simulations

4.2.1 VR world

Previous simulations supposed well-known feature coordinates. However, in a more realistic environment, in which the target body is not characterized by purposely created visual markers, the image processing must be performed to extract the information about features. The digital image processing concerns the use of algorithms to create, elaborate, transmit and visualize digital images. Features can be detected through an appropriate image analysis.

MATLAB Virtual Reality Toolbox (VR) can be used to introduce digital images in our model. The Virtual Reality Toolbox is a solution for viewing and interacting with dynamic systems in a 3D virtual reality environment. It extends the capabilities of MATLAB and Simulink into the world of virtual reality graphics.

Hence, a virtual world can be built through this toolbox and a CAD model of the target, that must be captured, can be imported into it. A rocket engine (see Fig. 13) has been chosen as target because it could be a spatial debris such as the last stage of a launcher, but this analysis can be done with other objects as well.

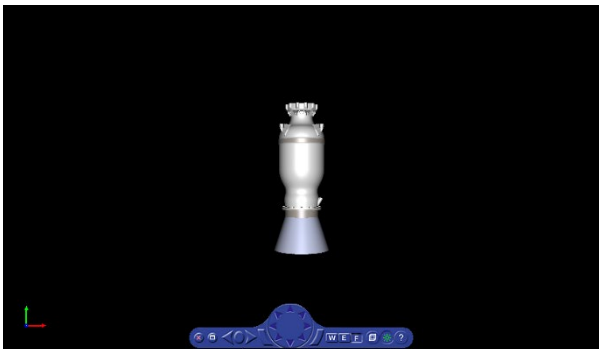

Fig. 13 Target into the virtual world

The engine is located with the center of mass in the origin of VR reference frame that corresponds to the object reference frame introduced in Section 2.

In the virtual reality world, it is possible to observe the engine from different points of view that can be modified via a MATLAB code. Varying the point of view means to change the position and orientation of a camera that focuses the object. This camera is not real because its intrinsic parameters cannot be changed.

4.2.2 Feature detection and description

Features such as points, lines, edges, corners or other elements that characterize the object can be detected in the image acquired at time t and compared with the desired features.

The feature extraction is divided in two steps [12]: feature detection and feature description. In the first phase, the purpose is to find a set of distinctive and stable points of interest while, in the second phase, the goal is to build vectors named descriptors that contain information about the neighborhood of the detected points. Descriptors are fundamental to connected features of different images.

In the literature, a large variety of feature extraction methods have been proposed to compute reliable descriptors. KAZE [13] is the detector that has been chosen for our simulations because it is more efficient in terms of number of features extracted than others.

The choice of KAZE detector is not necessarily the best one in every circumstance [14]. In fact, rendered images are still different from real images, that are usually more detailed. So, for real images or different targets, the KAZE detector could be less efficient than other detectors.

4.2.3 Feature matching

Feature matching consists in finding corresponding points between the current image and the desired image. An example of image acquisition, comparison with the desired one (Fig. 14), and matching of the relevant features (Fig. 15) is shown for clarity. This process allows to write the error defined as the difference between the position of matched features and executes the IBVS control.

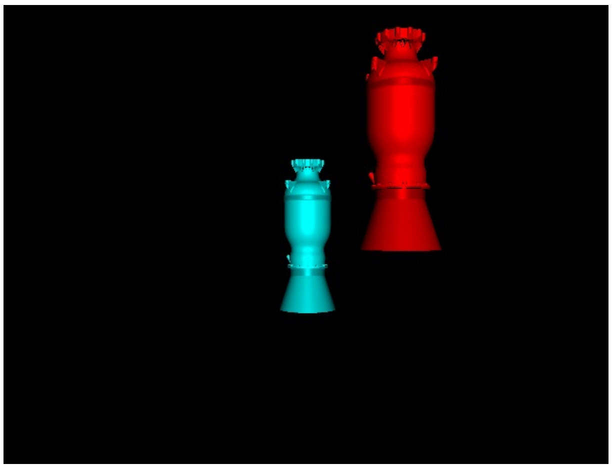

Fig. 14 Desired image (red target) and current image (cyan target)

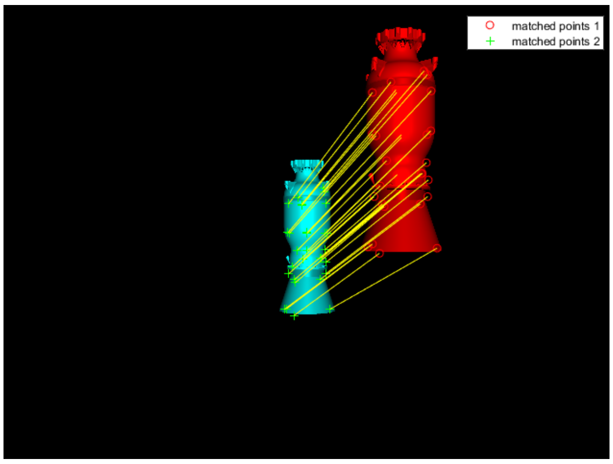

Fig. 15 Feature matching

To realize feature matching, the descriptors of the two images are compared.

Consider p, one of the m features detected from the desired image, and its related descriptor $\vec{\phi}$. The purpose is finding the best match in the second image in which n features have been detected. The Euclidean distance between the descriptor of feature p and all the n descriptors of features detected in the image acquired at time t must be calculated to find the best match:

$$d_i = \sum |\vec{\phi}(p) - \vec{\phi}(q_i)| \quad i = 1, \dots, n$$

The points detected in the second image are organized in ascending order from the descriptor closer to $\vec{\phi}(p)$ to the furthest one. So, the matched features are those that have the distance d_i smaller.

Feature matching is a delicate phase because it can lead to false matches. In fact, it is probable that a point of interest in the first image has more than one corresponding point in the second image and it is necessary to choose the best match based on some criterions. Tuning some values in the algorithm, false matches can be reduced. Clearly, no algorithm can guarantee that there are not false matches for all the time of simulation.

For our analysis, we choose descriptors made of 128 elements instead of usual 64 elements to improve feature matching. In addition, at each iteration we selected only the 10 best matches.

4.3 Simulation set-up architecture

The introduction of virtual camera and images processing modify the control loop as depicted in Fig. 16.

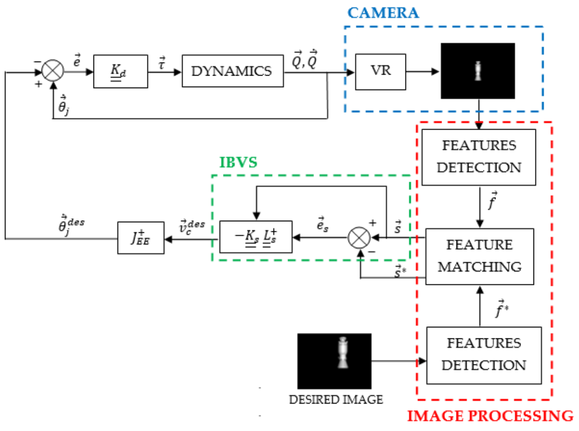

Fig. 16 Simulation set-up scheme

The camera is inside the loop. At each step, the manipulator changes its configuration (according to the vectors $\vec{Q}$ and $\dot{\vec{Q}}$ provided in output by the dynamics) and the end effector reaches a new position.

The new camera position and attitude are defined in the object reference frame and insert in the VR setting. Hence, the field of view is modified, and a new image can be acquired.

To make the simulation more realistic, the camera does not capture images at each integration step (fixed at $h = 0.01\ s$) but every $\Delta t = 0.2\ s$.

4.4 Results for advanced simulations

We consider as first situation a simple case illustrated in Fig. 17, where the two images differ only in the vertical coordinate of the camera position.

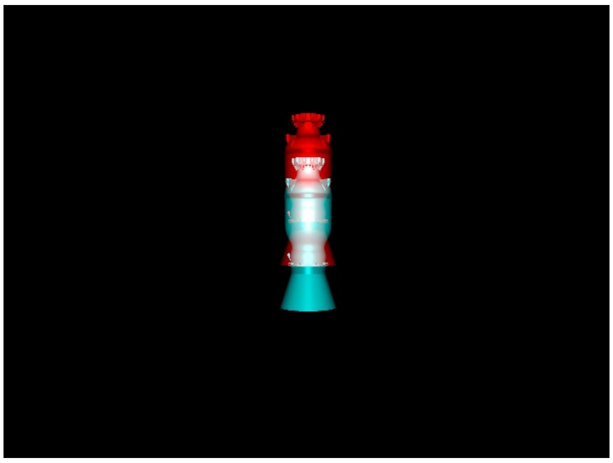

Fig. 17 Desired image (red target) and initial image (cyan target)

The simulation gives the following result.

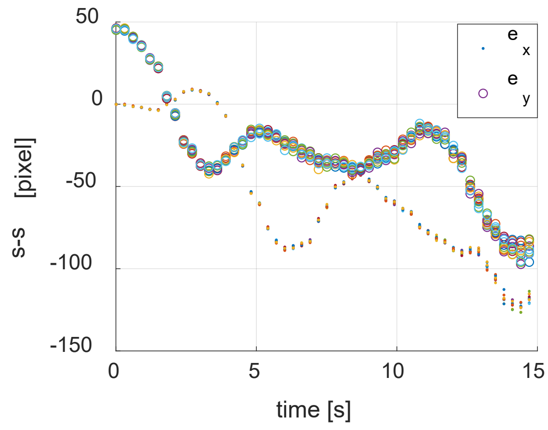

Fig. 18 Behavior of components x and y of error

Fig. 18 shows a totally unacceptable result. The behavior of error is unstable and diverges over time.
Performing numerous simulations for different target positions and for different gains, the features error never converges to zero. So, the IBVS provides undesirable results with the introduction of a virtual image in the control loop while it perfectly works in ideal case. For this reason, the observed problems in these simulations are not due to incorrect design of IBVS but to the differences between ideal and realistic case, that is the introduction of images.
In the ideal simulation, the error vector is generated by always comparing the same four points chosen like features. Now, every Δt KAZE does not detect all the same points of previous step and, consequently, the feature matching gives different results. So, at each step the vector error is calculated by comparing different features. This is the first difference with the ideal case.
The second big difference is that the location of detected points is not accurate as the ideal case. Small errors in the vector $\vec{e}$ lead to discordant camera velocity. To understand this issue, consider the previous example. The y-coordinate is the only difference between the two vectors. We expect that the camera obtains only a negative velocity in y direction to reach the target. Hence, features are detected from the two images and the feature matching is executed.

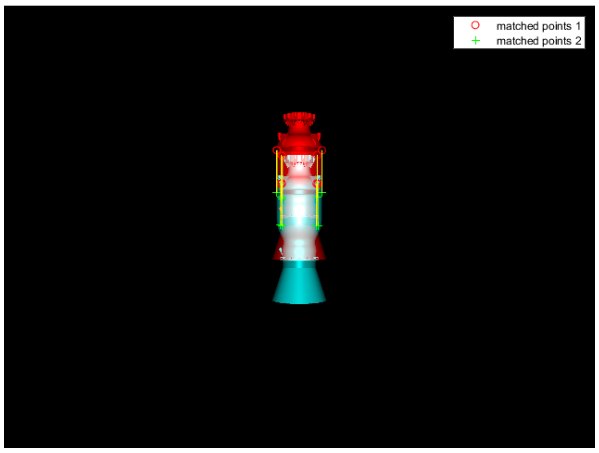

Fig. 19 Feature matching

Observing the feature matching (Fig. 19), we expect that features error will be zero for the x components and will be different from zero for the y components.
The feature vector is defined in the following way:

$$\vec{e} = \vec{s} - \vec{s}^* = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix} - \begin{bmatrix} x_1^* \\ y_1^* \\ \vdots \\ x_n^* \\ y_n^* \end{bmatrix} = \begin{bmatrix} e_{x1} \\ e_{y1} \\ \vdots \\ e_{xn} \\ e_{yn} \end{bmatrix} \quad (29)$$

The resulting vector has the expected form reported in Table 1.

Table 1 Errors using real images.

e_x	e_y
-0.0029	45.0611
-0.1309	45.7435
0.0452	45.8896
0.0580	45.8530
-0.0509	45.8560
-0.0045	45.0383
0.1259	45.7331
-0.0427	45.8884
0.0046	45.9581
-0.1546	45.8007

Now, thanks to the vector $\vec{e}$ and the interaction matrix calculated with the features of initial image, it is possible to obtain the desired velocity of the camera following eq. (25), with the results reported in Table 2.

Table 2 Desired velocity of the camera using real images.

$v_x = 3.6 \cdot 10^{-4}\, m/s$	$\omega_x = 2.41 \cdot 10^{-2}\, rad/s$
$v_y = 7.8 \cdot 10^{-4}\, m/s$	$\omega_y = -1.8 \cdot 10^{-4}\, rad/s$
$v_z = 2.6 \cdot 10^{-3}\, m/s$	$\omega_z = -5.37 \cdot 10^{-4}\, rad/s$

The result is totally different from the one supposed:

- v_y is positive and has the same order of magnitude of v_x
- v_z is the most important component of linear velocity
- ω_x is the biggest velocity component

The camera moves towards positive y and performs a positive rotation around x axis rather than moves towards negative y and preserves the attitude.

We want to perform another calculation. We modify the vector error setting the x components to zero and the y components to 45 pixels while we leave the interaction matrix unvaried. The desired camera velocity becomes as reported in Table 3.

Table 3 Desired velocity setting factiously to zero the errors due to noisy image processing

$v_x = 1.58 \cdot 10^{-15}\, m/s$	$\omega_x = 2.77 \cdot 10^{-17}\, rad/s$
$v_y = -0.047\, m/s$	$\omega_y = 9.85 \cdot 10^{-16}\, rad/s$
$v_z = -3.5 \cdot 10^{-16}\, m/s$	$\omega_z = 2.29 \cdot 10^{-17}\, rad/s$

The results agree with the hypothesized values: v_y is the only component of the velocity different from zero and it is negative.

From this test, it evident that errors smaller than 1 pixel cause important variations in the velocity components. So, another control law must be executed to converges features error to zero.

5. Sequential Partial Control

An alternative control law is found to perform a suitable control for the manipulator such that the feature error converges to zero.

Regarding previous example, the camera must only move towards positive direction of y axis to reach the target. For this reason, we can think to set equal to zero all the elements of interaction matrix that are referred to camera angular velocity. We obtain the following matrix:

$$\underline{\underline{L}}_T = \begin{bmatrix} \underline{\underline{J}}_T^1 \\ \vdots \\ \underline{\underline{J}}_T^k \end{bmatrix} \tag{30}$$

where:

$$\underline{\underline{J}}_T^i = \begin{bmatrix} -\frac{f}{z} & 0 & \frac{u_i}{z} & 0 & 0 & 0 \\ 0 & -\frac{f}{z} & \frac{v_i}{z} & 0 & 0 & 0 \end{bmatrix}$$

The desired camera velocity is given by:

$${}^{C}\vec{v}_C^{des} = -\underline{\underline{K}}_s\, \underline{\underline{L}}_T^{+}\, \vec{e} \tag{31}$$

Using this relationship in the previous example, we find the results of Table 4.

Table 4 Desired velocity with the sequential partial approach.

$v_x = 3.73 \cdot 10^{-5}\, m/s$	$\omega_x = 0\, rad/s$
$v_y = -0.0475\, m/s$	$\omega_y = 0\, rad/s$
$v_z = 0.0044\, m/s$	$\omega_z = 0\, rad/s$

Hence, these results are more acceptable than the ones obtained using the total interaction matrix. In fact, the component v_y assumes the expected value while both v_x and v_z have a lower order of magnitude compared to v_y. The control law executed with this partial interaction matrix limits the end-effector movements because the camera cannot acquire angular velocity around its axis. Therefore, if the initial image acquired is simply rotated with respect to the desired one, this control does not work. For example, consider the following image Fig. 20.

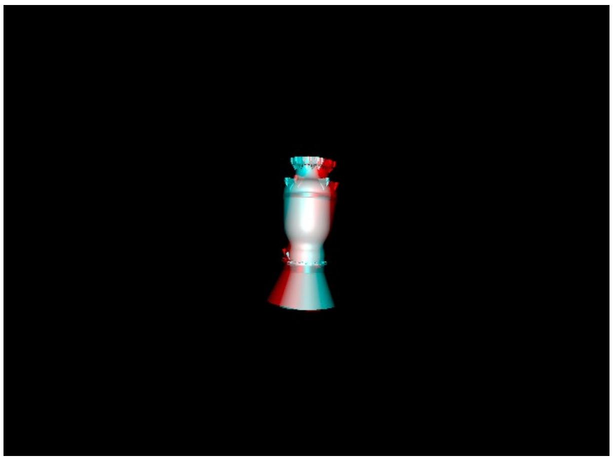

Fig. 20 Desired image (red target) and initial image (cyan target)

In this case, the camera must just perform a rotation of 10° around its optical axis to reach the goal and therefore the matrix $\underline{\underline{L}}_T$ cannot be used.

We proceed as the previous example setting equal to zero all the elements of interaction matrix referred to camera linear velocity:

$$\underline{\underline{L}}_R = \begin{bmatrix} \underline{\underline{J}}_R^1 \\ \vdots \\ \underline{\underline{J}}_R^k \end{bmatrix} \tag{32}$$

where

$$\underline{\underline{J}}_R^i = \begin{bmatrix} 0 & 0 & 0 & \frac{u_i\, v_i}{f} & -\left(f + \frac{u_i^2}{f}\right) & v_i \\ 0 & 0 & 0 & f + \frac{v_i^2}{f} & -\frac{u_i\, v_i}{f} & -u_i \end{bmatrix}$$

The desired camera velocity is given by:

$$^{C}\vec{v}_C^{des} = -\underline{\underline{K}}_s\, \underline{\underline{L}}_R^{+}\, \vec{e} \tag{33}$$

We find the results of Table 5.

Table 5 Desired velocity obtained setting to zero the camera linear velocity

$v_x = 0\, m/s$	$\omega_x = 1.13 \cdot 10^{-4}\, rad/s$
$v_y = 0\, m/s$	$\omega_y = -4 \cdot 10^{-5}\, rad/s$
$v_z = 0\, m/s$	$\omega_z = -0.0342\ rad/s$

The most important component of velocity is ω_z as we expected.

In these examples, the initial and desired location of the camera are well-known and consequently also the desired velocity of the camera is known. However, in the real simulation, the desired image is only provided while the final position of camera is unknown, and we cannot know a priori what matrix is better to use. In a general case both a translation and a rotation could be necessary to get the desired position.

For this reason, a criterion valid for any desired target position and that alternates the use of $\underline{\underline{L}}_T$ and $\underline{\underline{L}}_R$ must be formulated.

The following solution is proposed. At the beginning of the simulation, the matrix $\underline{\underline{L}}_T$ is chosen to execute the visual servoing control (but equivalent results are found also taking the matrix $\underline{\underline{L}}_R$). A check on the behavior of the error is done to understand if the chosen matrix is suitable to realize the control. If the error diverges or converges to a value different from zero, the interaction matrix is changed into $\underline{\underline{L}}_R$. Otherwise, the simulation proceeds using $\underline{\underline{L}}_T$.

5.1 Algorithm

For $t = \Delta t = 0.2\, s$ (first time step), the image is captured, and the image processing is done. In this way, it is possible to calculate the error vector and consequently the desired camera velocity using the interaction matrix $\underline{\underline{L}}_T$ (Eq. 31).

In addition, the averages of error components x and components y is performed:

$$\bar{e}_x = \frac{1}{n}\,(e_{x1} + e_{x2} + \cdots + e_{xn}) \tag{34}$$

$$\bar{e}_y = \frac{1}{n}\,(e_{y1} + e_{y2} + \cdots + e_{yn}) \tag{35}$$

Then, the joints desired angular velocities are obtained from the knowledge of $^{C}\vec{v}_C^{des}$, and the control torques are calculated and applied to joints.

For $t = 2\,\Delta t$ a new image is captured. The feature error vector, the average of the error components and the desired camera velocity vector are calculated. New control torques are founded. The procedure is repeated until n images have been acquired and therefore n feature error vectors have been obtained. For the present simulation, n is fixed to 20. So, until $t = 4\, s$ we proceed as indicated.

For $t = 4\, s$, the averages of values $\bar{e}_x$ and $\bar{e}_y$ measured until now are calculated:

$$M_{1x} = \tfrac{1}{10}\left(\bar{e}_x(t_k) + \cdots + \bar{e}_x(t_{k-10})\right) \tag{36}$$

$$M_{2x} = \tfrac{1}{10}\left(\bar{e}_x(t_{k-11}) + \cdots + \bar{e}_x(t_{k-20})\right) \tag{37}$$

$$M_{1y} = \tfrac{1}{10}\left(\bar{e}_y(t_k) + \cdots + \bar{e}_y(t_{k-10})\right) \tag{38}$$

$$M_{2y} = \tfrac{1}{10}\left(\bar{e}_y(t_{k-11}) + \cdots + \bar{e}_y(t_{k-20})\right) \tag{39}$$

where in this case $t_k = 4\, s$, $t_{k-1} = 4\, s - \Delta t$, $t_{k-2} = 4\, s - 2\,\Delta t$ etc. Then, the following values are computed:

$$x_{cond} = |M_{1x} - M_{2x}|$$
$$y_{cond} = |M_{1y} - M_{2y}|$$

and a check on the trend of error is done.

- Stack Conditions (S)

  The stack conditions are given by:

  $$x_{cond} < x_{stack}^{th} \ \ \&\ \ y_{cond} < y_{stack}^{th} \tag{40}$$

  where x_{stack}^{th} and y_{stack}^{th} are thresholds fixed to $2\, pixel$. This value is found via a trial and error approach in such a way that it is not too small (it is never reached) or too large (the maneuver could have been performed longer because it had the possibility to further reduce the error).

If the conditions given by (40) are satisfied, we change the interaction matrix into $\underline{L}_R$. Otherwise, a check on divergence conditions is made.

- Divergence Conditions (D)

  The divergence conditions are given by:

  $$x_{cond} > x_{div}^{th} \mid y_{cond} > y_{div}^{th} \qquad (41)$$

  where x_{div}^{th} and y_{div}^{th} are thresholds fixed to $5\ pixel$. This value has been selected via a trial and error approach in such a way that the divergence alarm does not happen too early, and, on the other side, too much time has spent before changing the operation mode.

  If one or both conditions (41) are satisfied, the error is considered as diverging, and the interaction matrix must be changed.

After these checks, the camera velocity is calculated with the new interaction matrix (or with the previous one if the conditions are not respected) and the control loop is executed as long as other n images are acquired. Then, the behavior of the error must be examined again to choose the right matrix.

The process is repeated until the end of simulation, as represented by Fig. 21.

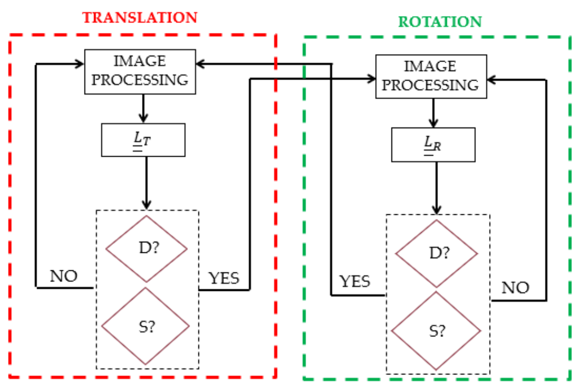

Fig. 21 New control loop: sequential partial control.

This strategy, from a conceptual point of view, is on the same line of the Task Priority (TP) [15] or Task Sequencing (TS) [16] strategies, which have been also experimentally tested in [17]. According to these strategies, the main task can be split into subtasks (TP approach) or in distinct phases so that an 'artificial' redundancy is introduced during task execution (TS approach). However, in the proposed Sequential Partial Control the logic states are iteratively changed, without any a priory schedule or intervention by an operator, and it can be therefore considered self-adapting to potential stall or divergence problems.

5.3 Results for the Sequential Partial Control

Numerous simulations have been executed to verify the new control law that works for all the studied cases.

The feature error always converges to zero and therefore every time the end effector reaches the desired pose. This kind of control has proven to be robust to false matches as well.

Consider the following example, where the target in the desired pose is rotated and translated with respect to the initial image (Fig. 22).

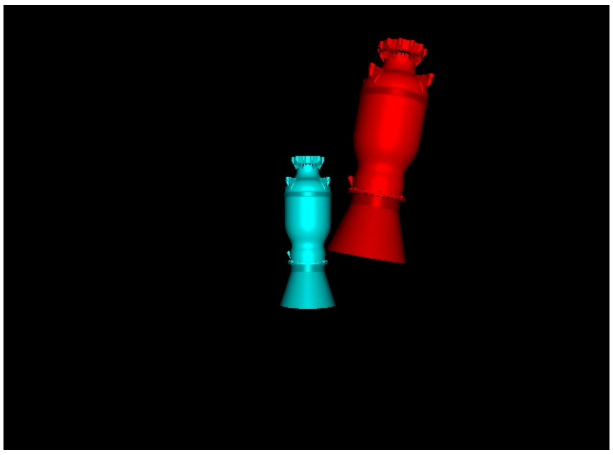

Fig. 22 Desired image (red target) and initial image (cyan target)

The following results are obtained with the new control law.

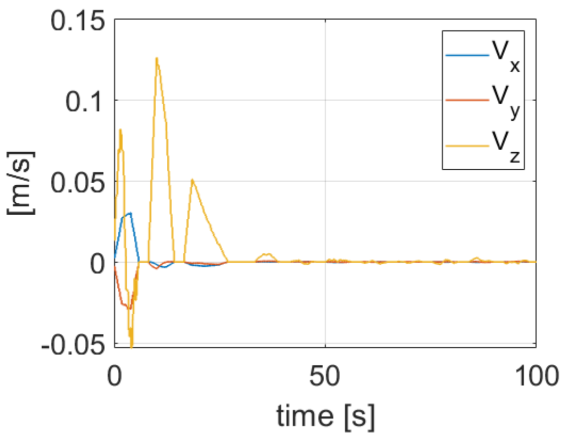

Fig. 23 Components of camera linear velocity written in camera reference frame

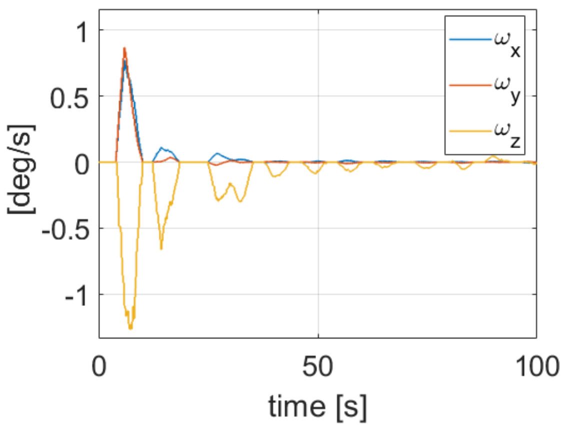

Fig. 24 Components of camera angular velocity written in camera reference frame

In Fig. 23 and Fig. 24 the linear and angular camera velocity are displayed and their particular behavior can be seen: when the components of translational velocity are different from zero, the components of angular velocity are null and vice versa. Nevertheless, the error exhibits a suitable trend and it converges in a short time (Fig. 25).

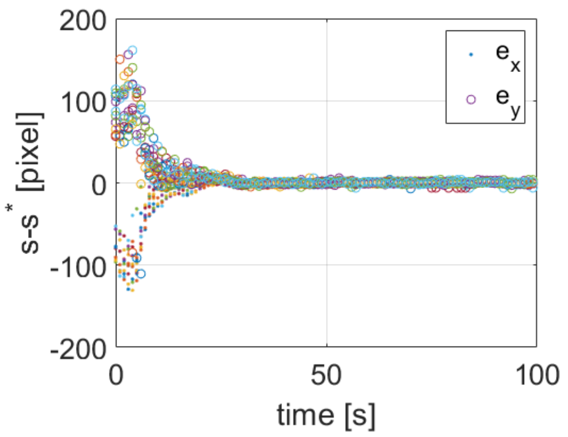

Fig. 25 Behavior of components x and y of error

This achieved by applying control torques of reasonable level, as reported in Fig. 7 for the first joint. Similar levels are recorded for the other joints. It must be noted that in these simulations the joints and relevant actuators are considered ideal.

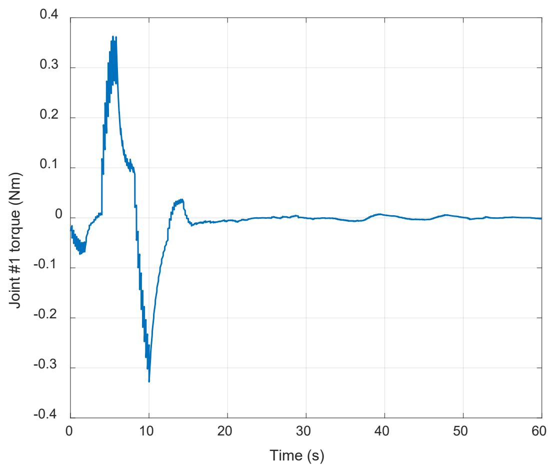

Fig. 26 Control torque for the first joint

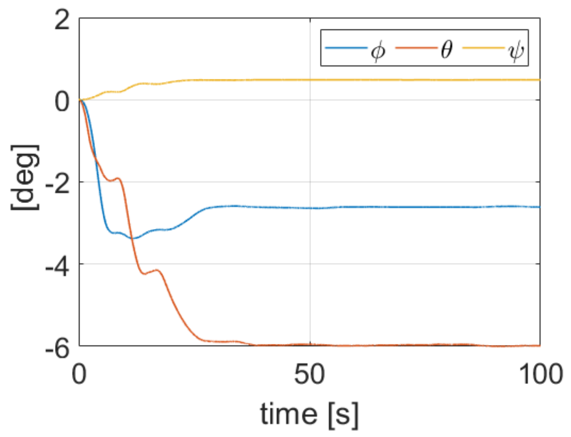

Fig. 27 Base Attitude

As in the ideal case, the satellite performs a change in attitude due to the movements of robotic arm (Fig. 26).

6. Conclusions

In this work, a study on the performance and limitation of the IBVS technique applied to space manipulators is presented. Advanced simulations, performed including rendered images in the GNC loop, have shown that small errors on the feature identification and matching process can produce large inaccuracies in the control computation, leading to the mission failure.

In order to limit this problem, a novel approach has been proposed, consisting in a sequential application of two different kinds of IBVS: one of these is focused on the linear motion of the end effector only, the other one is exclusively focused on the rotation of the end effector. Two switching conditions ("divergence" and "stack" conditions) have been introduced, so that the overall behavior of the manipulator is robust for a large number of considered scenarios.

Appendix 1

To execute the simulations, the spacecraft has been represented as a prism platform and the robotic arm has been represented as a series of empty cylinder with thickness t. The following tables show the geometrical and inertial properties of the bodies.

Spacecraft	
Area of prism base	$A = 1\,m \times 1m$
Height	$h = 2\,m$
Mass	$m_B = 600\,kg$
Inertia matrix in body frame	$\underline{\underline{{}^{B}J_{CM}}} = \begin{bmatrix} 250 & 0 & 0 \\ 0 & 250 & 0 \\ 0 & 0 & 100 \end{bmatrix} kg\,m^2$

Link 2 – Link 3 – Link 4	
Length	$l = 1\,m$
Circumference radius	$r = 0.05\,m$
Thickness	$t = 0.004\,m$
Mass	$m_i = 15\,kg$
Inertia matrix in body frame	$\underline{\underline{{}^{bi}J_{0_i}}} = \begin{bmatrix} 0.0346 & 0 & 0 \\ 0 & 5.0173 & 0 \\ 0 & 0 & 5.0173 \end{bmatrix} kg\,m^2$

Link 1 – Link 5 – Link 6	
Length	$l = 0.2\,m$
Circumference radius	$r = 0.05\,m$
Thickness	$t = 0.004\,m$
Mass	$m_i = 5\,kg$
Inertia matrix in body frame of link 1	$\underline{\underline{{}^{b1}J_{0_1}}} = \begin{bmatrix} 0.0724 & 0 & 0 \\ 0 & 0.0724 & 0 \\ 0 & 0 & 0.0115 \end{bmatrix} kg\,m^2$
Inertia matrix in body frame of link 5	$\underline{\underline{{}^{b5}J_{0_5}}} = \begin{bmatrix} 0.0115 & 0 & 0 \\ 0 & 0.0724 & 0 \\ 0 & 0 & 0.0724 \end{bmatrix} kg\,m^2$
Inertia matrix in body frame of link 6	$\underline{\underline{{}^{b6}J_{0_6}}} = \begin{bmatrix} 0.0724 & 0 & 0 \\ 0 & 0.0724 & 0 \\ 0 & 0 & 0.0115 \end{bmatrix} kg\,m^2$

References

[1] A. Ellery, “Tutorial Review on Space Manipulators for Space Debris Mitigation”, *Robotics*, No. 8, 2019.

[2] S. Hutchinson, G. D. Hager, P. I. Corke, “A Tutorial on Visual Servo Control”, *IEEE Transactions On Robotics And Automation*, Vol. 12, No. 5, Ottobre 1996 .

[3] F. Chaumette, S. Hutchinson, “Visual servo control. I. Basic approaches”, *IEEE Robotics Automation Magazine*, Vol. 13, No. 4, pp.82-90, Dicembre 2006.

[4] K. Hashimoto, T. Kimoto, T. Ebine, H. Kimura, “Manipulator Control with Image-Based Visual Servo”, *IEEE International Conference on Robotics and Automation*, Sacramento.

[5] K. Yoshida, B. Wilcox, *Space Robots and Systems*, Springer, 2008.

[6] M. Sabatini, R. Monti, P. Gasbarri, G. B. Palmerini “Adaptive and robust algorithms and tests for visual-based navigation of a space robotic manipulator”, *Acta Astronautica*, Vol. 83, pp. 65-84, 2013

[7] Petit, A., Marchand, E., Kanani, K. “Vision-based space autonomous rendezvous: A case study”, (2011) IEEE International Conference on Intelligent Robots and Systems, pp. 619-624.

[8] A.H.A. Hafez, V.V. Anurag, S.V. Shah, K.M. Krishna, C.V. Jawahar, Reactionless visual servoing of a dual-arm space robot, in: 2014 IEEE Int. Conf. Robot. Autom., IEEE, 2014, pp.4475–4480.

[9] J. Alepuza, M. R. Emami, J. Pomares, “Direct image-based visual servoing of free-floating space manipulators”, *Aerospace Science and Technology*, No. 55, pp. 1-9, 2016.

[10] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics: modelling, planning and control*, Springer, 2009

[11] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2° Edition, Cambridge University Press, 2003.

[12] A. I. Awad, M. Hassaballah, *Image Feature Detectors and Descriptors: Foundations and Applications*, Springer, Svizzera, 2016.

[13] P. F. Alcantarilla, A. Bartoli, A. J. Davison, “KAZE Features”, European Conference on Computer Vision, ECCV 2012, Part VI, LNCS 7577, pp. 214–227, Springer, 2012.

[14] S. A. K. Tareen, Z. Saleem, “A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK”, iCoMET, 2018.

[15] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, “Closed loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy,” Int. J. of Robotics Research, vol. 10, pp. 410–425, 1991.

[16] A. De Luca, G. Oriolo, and P. Robuffo Giordano, “Image-based visual servoing schemes for nonholonomic mobile manipulators,” Robotica, vol. 25, no. 2, pp. 131–145, 2007.

[17] A. De Luca, M. Ferri, G. Oriolo, P. Robuffo Giordano, "Visual Servoing with Exploitation of Redundancy: An Experimental Study", 2008 IEEE International Conference on Robotics and Automation Pasadena, CA, USA, May 19-23, 2008.