Spacecraft Robotics Toolkit: an Open-Source Simulator for Spacecraft Robotic Arm Dynamic Modeling And Control

Conference Paper · March 2016

CITATIONS
8

READS
1,457

3 authors, including:

Josep Virgili-Llop
Naval Postgraduate School
45 PUBLICATIONS **731** CITATIONS

SEE PROFILE

Marcello Romano
Naval Postgraduate School
146 PUBLICATIONS **2,511** CITATIONS

SEE PROFILE

SPACECRAFT ROBOTICS TOOLKIT: AN OPEN-SOURCE SIMULATOR FOR SPACECRAFT ROBOTIC ARM DYNAMIC MODELING AND CONTROL.

Josep Virgili-Llop, Jerry V. Drew II, Marcello Romano

Spacecraft Robotics Laboratory
Naval Postgraduate School
Monterey CA USA

ABSTRACT

An open-source, six-degree-of freedom kinematic and dynamic software toolkit for a spacecraft with attached robotic arm has been developed and released. The toolkit is capable of simulating a floating or a flying base and can handle external forces, both in operational and in joint space. Based on a Newton-Euler approach which makes use of the Decoupled Natural Orthogonal Complement matrix, the forward and inverse dynamics are solved using an efficient, recursive $\mathcal{O}(n)$ algorithm. Recursive $\mathcal{O}(n)$ formulations to obtain the generalized inertia and convective inertia matrices have also been implemented. Written as a collection of function for MATLAB/ Simulink, the toolkit is very modular and it can be used for standalone MATLAB scripts or for Simulink models. The resulting Simulink models are suitable for code generation and thus can be readily compiled and executed into embedded hardware or integrated with third party tools. In addition to modeling the kinematics and dynamics, the software includes tools to help the user create control and analysis applications.

Index Terms— SPART, spacecraft robotics, toolkit, open-source, kinematics, dynamics, control

1. INTRODUCTION

A wide range of space missions require a robotic arm (e.g. satellite servicing, active debris removal and berthing). The kinematics and dynamics of space manipulators are highly non-linear and differ considerably from their terrestrial counterparts. The base-spacecraft is not anchored to the ground and thus it is free to react to the manipulator's motion, making the modeling and control of space-based manipulators a complex task. The base-manipulator interaction is stronger when the mass and inertia of the manipulator and of the base-spacecraft are comparable. Therefore, the difference between space and terrestrial manipulators tends to become more acute on manipulators mounted on small spacecraft.

There is a wealth of literature on the modeling and control of spacecraft manipulators, but each research group has typically developed its own software in order to simulate and validate control approaches. In an attempt to help speed the process and make space manipulators a more accessible research topic, the open-source SPAcecraft Robotics Toolkit (SPART) has been developed and released. SPART [1] computes the kinematics and dynamics of the spacecraft-manipulator system and thus can be used as a simulator (physics engine). Additionally, it includes control algorithms and tools to analyze the properties of a spacecraft manipulator system (e.g. workspace and manipulability analysis). The goal is to create an end-to-end toolkit to tackle spacecraft robotics problems.

This toolkit is originally written in MATLAB/Simulink, and it can be used as standalone MATLAB scripts or Simulink models. The Simulink models can subsequently be used for automatic code-generation and compiled to run in real-time on the selected target hardware (e.g. for embedded applications). This open-source code is thus suitable across the spectrum of system development from prototyping work to hardware implementation. Written as a collection of different functions the toolkit is very modular and flexible and thus can be integrated with other MATLAB/Simulink projects to rapidly create an application-specific code with little overhead.

The six-degree-of-freedom simulator can determine the system kinematics, such as the homogeneous coordinate transformation matrices and the velocity Jacobians, and also the systems dynamics, including the generalized inertia and convective inertia matrices. Starting with a Newton-Euler formulation and making use of the Decoupled Natural Orthogonal Complement matrix [2, 3] the forward and inverse dynamic problems are solved using an efficient, recursive $\mathcal{O}(n)$ algorithm. Recursive $\mathcal{O}(n)$ formulations to obtain the generalized inertia and convective inertia matrices have also been implemented. Several control manipulator and base-spacecraft control approaches have also already been implemented (for example, resolved motion-rate control and a zero reaction maneuver. More control tools are expected to be added in the near future.

As it currently stands, the toolkit has some limitations. Its underlying formulation is only valid for kinematic-tree topologies composed of rigid bodied. Closed-loop chains [4]

and flexible bodies [5] are not currently supported, although they could be implemented in the future.

This toolkit has been developed to support the teaching and experimental research activities conducted at the Naval Postgraduate School Floating Spacecraft Simulator (NPS-FSS) [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. The NPS-FSS test bed is a dynamic simulator consisting of a 4-by-4 meter granite monolith where Floating Spacecraft Simulators float via air-bearings (recreating a reduced friction and gravity environment). A recent addition of the test bed is a modular four-link manipulator arm [16].

Many mature and advanced general physics engines and libraries already exists, mainly the Open Dynamics Engine (ODE) [17], Bullet [18], Simbody [19] and Dynamic Animation and Robotics Toolkit (DART) [20]. These are mainly focused in simulation and do not provide, except DART, full access to internal kinematic and dynamic quantities (i.e. the generalized inertia matrix, generalized convective inertia or kinematic transformations, among others). In addition, these other physics engines and libraries are general in scope and thus are not focused on spacecraft robotics (floating-base) and do not include any control algorithms or tools to analyze the properties of a particular spacecraft-manipulator system configuration. As they are mainly written in C/C++ they are suitable for embedded execution but are difficult or cumbersome to use for rapid algorithm prototyping.

First, the main equations of a spacecraft with a robotic arm will be reviewed. Then, a brief introduction to the Decoupled Natural Orthogonal Complement matrix will be provided. Finally, the architecture and usage of the simulator will be presented as well as some examples that demonstrate how the simulator performs under some basic manipulator control applications.

2. KINEMATIC AND DYNAMIC MODELING

Historically, multiple methods to solve both the forward and inverse dynamics problems have been proposed. Most of these methods start by formulating the Newton-Euler (NE) or Euler-Lagrange (EL) equations.

Approaches based on NE have resulted in very computationally efficient $\mathcal{O}(n)$ [21, pp. 260-265] recursive algorithms for forward and inverse dynamics [22, 23, 24, 25, 26].

Of particular significance for the forward dynamics problem are the Composite Rigid Body method [23] and the Articulated-Body method [24]. The forward dynamics of the Composite Rigid Body method [23] results in a $\mathcal{O}\left(n^3\right)$ algorithm but with very small coefficients (i.e. $1/6n^3 + 11/2n^2 + 38/3n - 47$ multiplications), making it competitive for a small number of links (i.e. <12) [2].

A recursive algorithm based on Kane's equations [27] as well as a complex algorithm based on Kalman filtering and smoothing theory[28] have also been proposed. Parallel computing approaches that can reduce the complexity down to $\mathcal{O}(\log(n))$ [29, 30] have been developed although are only attractive for systems with a large number of coupled bodies.

The recursive approach to be used in this paper is based on the Decoupled Natural Orthogonal Complement matrices and theories of linear algebra [31, 32, 2, 3]. This particular method exhibits a $\mathcal{O}(n)$ complexity when solving both the forward and inverse dynamics problems [2, 3] and allows one to obtain recursive $\mathcal{O}(n)$ formulations for the generalized inertia H and the convective inertia C matrices. The actual computational complexity coefficients are implementation dependent and and in general, can be improved by optimizing the code.

The state of the spacecraft-manipulator system is fully described by the joint variables q and their time derivatives $\dot{q}$ and $\ddot{q}$. These joint variables q define the *joint space,* in contrast to the *operational space* which uses the Cartesian representation. Manipulator tasks (e.g. capturing or manipulating an object) are usually defined in operational space but the control of the manipulator occurs in joint space (e.g. forces applied at the manipulator joints). It is worth noting that the generalized joint variables q include the position and attitude of the base-spacecraft q_0 and the manipulator joint states q_m and so $q = \begin{bmatrix} q_0 & q_\mathrm{m} \end{bmatrix}^T$. The equation of motion can be expressed in a compact form by Eq. (1). The τ vector denotes the generalized forces applied to the system in joint space, H denotes the generalized inertia matrix, and C is the generalized matrix of convective inertia, which contains the Coriolis and centrifugal forces.

$$H(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} = \tau \tag{1}$$

If the inertia matrices are known, Eq. (1) can be used to obtain a straightforward solution to the forward and inverse dynamics problems. However, recursive methods are significantly more computationally efficient and thus are commonly used for these tasks. Equation (1) is mainly used to gain insight on the system and to derive other useful expressions when formulating and solving different control problems. In particular, the generalized inertia matrix H is very much used to formulate and solve control problems, and thus it is important to obtain it in an efficient manner.

2.1. Kinematics

Each node (i.e rigid body) of the spacecraft-manipulator system will be referred as a link and identified by a number using a regular numbering scheme [26]. In such a scheme the base-spacecraft (root node) is given the number 0. Each link and its associate joint connecting it to its parent will have a higher number than its parent. In the case of branched manipulators, multiple numbering options will exist and can be chosen arbitrarily among them. Figure 1 schematically shows a generic spacecraft manipulator system and an example of the regular numbering scheme.

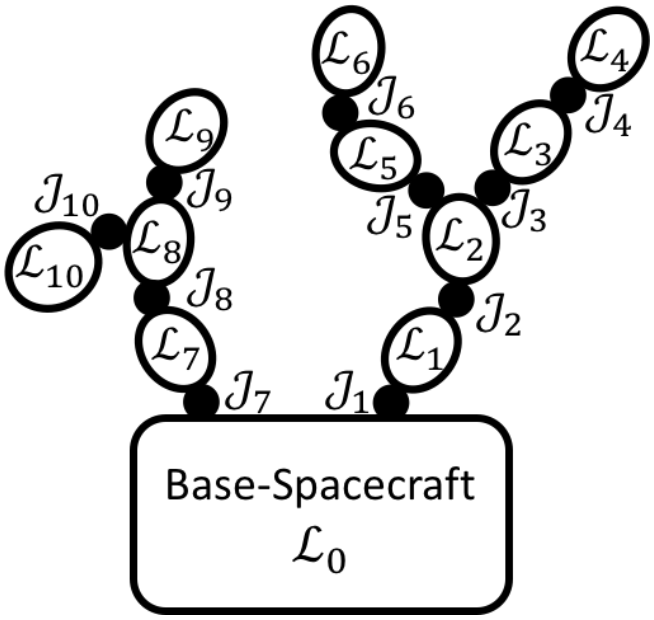


Fig. 1: Generic spacecraft-manipulator system.

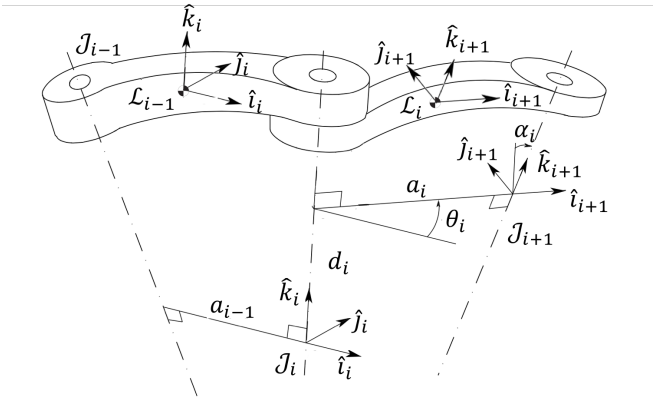


Fig. 2: Denavit-Hartenberg parameters (adapted from [33]).

2.1.1. Reference frames and coordinate transformations

An inertial reference frame $\mathcal{I}$ with arbitrary origin and orientation is defined and, unless otherwise noted, all magnitudes will be specified in this frame. Link $\mathcal{L}_i$ and joint $\mathcal{J}_i$ reference frames are also defined.

The origin and orientation of the joint reference frame $\mathcal{J}_i$ will be chosen using the Denavit-Harteneberg (DH) convention shown in Fig. 2 [33, 34].

It is important to clarify that the notation $i+1$ and $i-1$ will be used to denote a child and the parent link or joint irrespective of their numbering. Additionally the last joint and link on a branch will be generally denoted by n. This is done to simplify the notation while accommodating the kinematic tree topology where links can branch out as it is shown in Fig. 1.

In the DH convention, the $\hat{k}_i$ axis of a generic joint $\mathcal{J}_i$ is chosen along the joint rotation (revolute joint) or sliding axis (prismatic joint). The origin of the $\mathcal{J}_i$ frame is located at the intersection of the $\hat{k}_i$ axis with the common normal between $\hat{k}_{i-1}$ and $\hat{k}_i$. The $\hat{\imath}_i$ axis is usually chosen to point at the next joint origin. In the case of a link with multiple children, thus with multiple $\mathcal{J}_{i+1}$, one of the following joints is arbitrarily selected in order to align $\hat{\imath}_i$. Finally the $\hat{j}_i$ axis is laid out completing the right-hand triad.

In such convention there are four different parameters that univocally describe the relationship between the $\mathcal{J}_{i+1}$ and $\mathcal{J}_i$ frames. These parameters are shown in Fig. 2, and their geometric definition is provided in Table 1. Note that these parameters need to be defined for each pair of connected joints.

The θ and d parameters become the joint variables of revolute and prismatic joint respectively. As each joint is assumed to only be revolute or prismatic, the generic joint variable q is used to denote both (irrespective of the joint type).

The link reference frame $\mathcal{L}_i$ will have its origin located, without loss of generality, at the link's center-of-mass and their axis orientation will match one of their children joints $\mathcal{J}_{i+1}$ (to be selected arbitrarily). For convenience, the base-spacecraft reference frame $\mathcal{L}_0$ can also be defined arbitrarily, but to make the analysis simpler, it will be assumed that the origin of $\mathcal{L}_0$ is located in the base-spacecraft center-of-mass. In general, the magnitudes related to the base-spacecraft will be denoted by the subscript $_0$.

The adopted DH convention still leaves some undefined magnitudes. The origin and the direction of the $\hat{\imath}$ axes of the reference frames of the joints connected to the base-spacecraft $\mathcal{J}_{0+1}$ are arbitrary. Additionally, the last joint of each branch does not have a subsequent joint $\mathcal{J}_{n+1}$ and thus the direction of $\hat{\imath}$ could also be defined arbitrarily. The same occurs with the last link of a branch $\mathcal{L}_n$ where there is no subsequent joint $\mathcal{J}_{n+1}$ to inherit its orientation. Generally at the end of each branch an end-effector will be located, and a convenient reference frame for that end-effector will be defined. This end-effector reference frame will be denoted as $\mathcal{J}_{EE}$ and can be used as the $\mathcal{J}_{n+1}$ reference frame and thus keep the DH convention for the last link and joint of a branch. Figure 3 shows the disposition of the links and joints and the definition of their geometric parameters.

The DH parameters are the input to the simulator and describe the geometric relationship between the different rigid bodies of the system. With this input, the kinematics function is able to compute the transformation matrices which relate the different joint and links. This homogeneous transformation matrices from the joint and link frames to the inertial reference frame (${}^{\mathcal{I}}T_{\mathcal{J}_i}$ and ${}^{\mathcal{I}}T_{\mathcal{L}_i}$) can then be computed recursively moving though the tree topology from the base-spacecraft outwards as shown in Eq. (2) and (3).

Table 1: Denavit-Hartenberg parameters and their geometric definition.

DH parameter	Geometric definition
$d_{i,i+1}$	Distance between the $\mathcal{J}_i$ and $\mathcal{J}_{i+1}$ origins along the $\hat{k}_i$ axis. It is also the prismatic joint variable.
$\theta_{i,i+1}$	Rotation from $\hat{i}_i$ to $\hat{i}_{i+1}$ along $\hat{k}_i$. It is also the revolute joint variable.
$\alpha_{i,i+1}$	Rotation from $\hat{k}_i$ to $\hat{k}_{i+1}$ along $\hat{j}_{i+1}$.
$a_{i,i+1}$	Distance along the common normal between $\hat{k}_i$ and $\hat{k}_{i+1}$

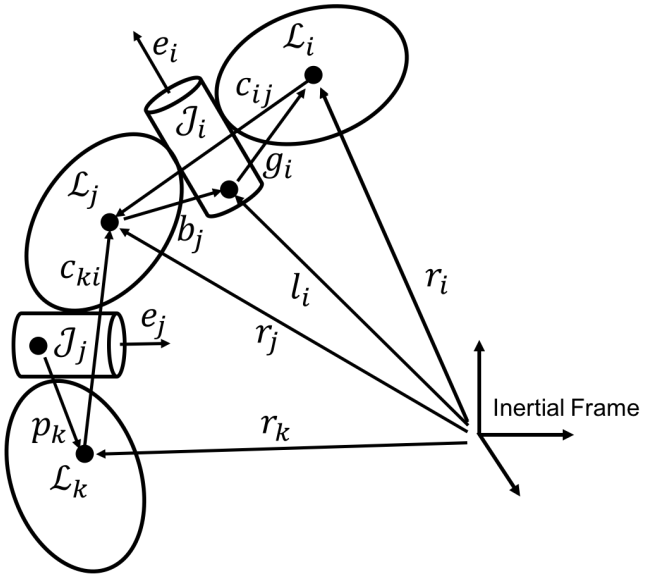


Fig. 3: Schematic disposition of links and joints.

$$ {}^{\mathcal{I}}T_{\mathcal{J}_i} = {}^{\mathcal{I}}T_{\mathcal{L}_0}\,{}^{\mathcal{L}_0}T_{\mathcal{J}_{0+1}} \prod_{j=2}^{i} {}^{\mathcal{J}_{j-1}}T_{\mathcal{J}_j} = {}^{\mathcal{I}}T_{\mathcal{J}_{i-1}}\,{}^{\mathcal{J}_{i-1}}T_{\mathcal{J}_i} \quad (2) $$

$$ {}^{\mathcal{I}}T_{\mathcal{L}_i} = {}^{\mathcal{I}}T_{\mathcal{J}_i}\,{}^{\mathcal{J}_i}T_{\mathcal{L}_i} = {}^{\mathcal{I}}T_{\mathcal{J}_{i+1}}\,{}^{\mathcal{J}_{i+1}}T_{\mathcal{L}_i} \quad (3) $$

A homogeneous transformation matrix is just the combination of the rotation matrix R and the displacement vector s written in a compact as the 4×4 matrix. If Eq. (4) represents a frame transformation of the vector $v^{\mathcal{J}_{i+1}}$, its equivalent homogeneous transformation matrix can then be constructed by Eq. (5).

$$ v^{\mathcal{J}_i} = {}^{\mathcal{J}_i}R_{\mathcal{J}_{i+1}} v^{\mathcal{J}_{i+1}} + {}^{\mathcal{J}_i}s_{\mathcal{J}_{i+1}} \quad (4) $$

$$ {}^{\mathcal{J}_i}T_{\mathcal{J}_{i+1}} = \begin{bmatrix} {}^{\mathcal{J}_i}R_{\mathcal{J}_{i+1}} & {}^{\mathcal{J}_i}s_{\mathcal{J}_{i+1}} \\ 0_{1,3} & 1 \end{bmatrix} \quad (5) $$

The transformation matrix from the first joint of the branch to the base-spacecraft ${}^{\mathcal{L}_0}T_{\mathcal{J}_{0+1}}$ is fixed by the spacecraft-manipulator geometry. The ${}^{\mathcal{L}_0}s_{\mathcal{J}_{0+1}}$ will then be the vector from the origin of $\mathcal{L}_0$ to the arbitrary origin of $\mathcal{J}_{0+1}$ in the $\mathcal{L}_0$ frame (i.e. location of that first joint in the $\mathcal{L}_0$ body axis). This transformation is also required as an input.

$$ {}^{\mathcal{L}_0}T_{\mathcal{J}_{0+1}} = \begin{bmatrix} {}^{\mathcal{L}_0}R_{\mathcal{J}_{0+1}} & {}^{\mathcal{L}_0}s_{\mathcal{J}_{0+1}} \\ 0_{1,3} & 1 \end{bmatrix} \quad (6) $$

If the mass of each link is denoted by m_i, the spacecraft-manipulator system center-of-mass position r_c can be easily computed using Eq. (7).

$$ r_c = \frac{\sum_{i=0}^{n} m_i r_i}{\sum_{i=0}^{n} m_i} \quad (7) $$

The rotation axis of each joint in the inertial frame e_i can be computed using Eq. (8) where ${}^{\mathcal{I}}R_{\mathcal{J}_i}$ denotes the rotation sub-matrix in ${}^{\mathcal{I}}T_{\mathcal{J}_i}$. Obviously e_i is simply the last column of the ${}^{\mathcal{I}}R_{\mathcal{J}_i}$ rotation matrix.

$$ e_i = \hat{k}_{\mathcal{J}_i} = {}^{\mathcal{I}}R_{\mathcal{J}_i} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (8) $$

The inertia of the ith link in inertial frame can be computed using Eq. (9).

$$ I_i = {}^{\mathcal{I}}R_{\mathcal{L}_i} I_i^{\mathcal{L}_i} {}^{\mathcal{I}}R_{\mathcal{L}_i}^T \quad (9) $$

2.1.2. Velocities and accelerations

The angular ω_i and linear $\dot{r}_i$ velocities of the ith link of the chain in operational space can be encapsulated into a six-dimensional *twist* vector t_i.

$$ t_i = \begin{bmatrix} \omega_i \\ \dot{r}_i \end{bmatrix} \quad (10) $$

The twist-propagation equations can be encapsulated in a more compact form as follows.

$$ t_i = B_{ij} t_j + p_i \dot{q}_i \quad (11) $$

Where the 6×6 $B_{i,j}$ matrix denotes the twist-propagation matrix and the 6-dimensional p_i vector represents the twist-propagation vector. Let t denote the $6\,(n+1)$ generalized twist vector and $\dot{q}$ the $(n+6)$ generalized joint-rate variables vector.

$$t = \begin{bmatrix} t_0 & t_1 & \dots & t_n \end{bmatrix}^T \tag{12}$$

$$\dot{q} = \begin{bmatrix} \dot{q}_0 & \dot{q}_1 & \dots & \dot{q}_n \end{bmatrix}^T \tag{13}$$

The base-spacecraft $\dot{q}_0$ state is composed of its angular velocity in body axis and its linear velocity in inertial frame, thus making it a six-dimensional vector. The rest of the generalized joint variables will then be part of the manipulator and will be generically denoted by the n-dimensional q_m vector.

$$\dot{q}_0 = \begin{bmatrix} \omega_0 & \dot{r}_0 \end{bmatrix}^T \tag{14}$$

$$\dot{q} = \begin{bmatrix} \dot{q}_0 & \dot{q}_m \end{bmatrix}^T \tag{15}$$

Using the generalized joint variables $\dot{q}$, a velocity transformation or natural orthogonal complement $6\,(n+1) \times (n+6)$ matrix N can be defined. The velocity transformation matrix maps the velocities in joint space $\dot{q}$ to velocities in the operational space t.

$$t = N\dot{q} \tag{16}$$

This matrix, first introduced by Angeles and Lee [31], can be decoupled to obtain the decoupled natural orthogonal complement [32] with N_l being the $6\,(n+1) \times (n+6)$ lower block triangular matrix and N_d the $6\,(n+1) \times (n+6)$ block diagonal matrix.

$$N = N_l N_d \tag{17}$$

$$N_l = \begin{bmatrix} 1_{6.6} & 0_{6,6} & \cdots & 0_{6,6} \\ B_{10} & 1_{6,6} & \cdots & 0_{6,6} \\ \vdots & \vdots & \ddots & \vdots \\ B_{n0} & B_{n1} & \cdots & 1_{6,6} \end{bmatrix} \tag{18}$$

$$N_d = \begin{bmatrix} P_0 & 0_{6,1} & \cdots & 0_{6,1} \\ 0_{6,6} & p_1 & \cdots & 0_{6,1} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{6,6} & 0_{6,1} & \cdots & p_n \end{bmatrix} \tag{19}$$

For the base-spacecraft the the usual p_i 6-dimensional propagation vector becomes a 6×6 matrix that contains the base-spacecraft rotation matrix as shown in Eq. (20). The velocity transformation matrix is then simplified as follows.

$$P_0 = \begin{bmatrix} {}^{\mathcal{I}}R_{\mathcal{L}_0} & 0_{3,3} \\ 0_{3,3} & 1_{3,3} \end{bmatrix} \tag{20}$$

$$t_0 = P_0 \dot{q}_0 \tag{21}$$

It is important to note that the twist-propagation matrix will be zero $B_{i,j} = 0$ if the two links are not in the same branch, producing branch induced sparsity on the N_l matrix. A single branch manipulator will produce a full lower trigonal N_l matrix. Additionally as all the links share a branch with the base-spacecraft then $B_{i0} \neq 0$.

The accelerations can then be computed by differentiating the twist-propagation equations as follows.

$$\dot{t}_0 = \Omega_0 P_0 \dot{q}_0 + P_0 \ddot{q}_0 \tag{22}$$

$$\dot{t}_i = B_{i,i-1}\dot{t}_{i-1} + \dot{B}_{i,i-1}t_{i-1} + \Omega_i p_i \dot{q}_i + p_i \ddot{q}_i \tag{23}$$

With $\dot{B}_{ij}$ being the 6×6 time derivative of the twist-propagation matrix, Ω_i being the 6×6 angular velocity matrix. This twist-rate vector can also be expressed using the velocity transformation time-derivative as follows.

$$\dot{t} = N\ddot{q} + \dot{N}\dot{q} \tag{24}$$

2.1.3. Jacobians

The individual velocity transformation matrices that compose N are also known as Jacobians [33] which is the common denomination used when formulating spacecraft robotics control problems [35, 36].

Instead of being used to obtain the generalized twist vector t, Jacobians are used to obtain the velocities of an arbitrary point on the manipulator. The $6 \times (n+6)$ J_i Jacobian can be used to obtain the ith link velocities t_i as shown in Eq. (25). It can be easily obtained as an horizontal slice of the velocity transformation matrix N.

$$t_i = J_i \dot{q} \tag{25}$$

More generically, a Jacobian that maps the velocities in joint space $\dot{q}$ to the velocities in operational space of a generic point on the ith link x_i can be also be defined and will be denoted by J_{x_i}. The contributions to the operational space velocity from the base-spacecraft motion $\dot{q}_0$ and from the manipulator motion $\dot{q}_m$ can be made explicit by splitting the J_{x_i} Jacobian into the 6×6 base-spacecraft Jacobian J_{0x_i} and the $6 \times n$ manipulator Jacobian J_{mx_i}. The operational space velocities can be simply obtained by Eq. (27).

$$J_{x_i} = \begin{bmatrix} J_{0x_i} & J_{mx_i} \end{bmatrix} \tag{26}$$

$$t_i = J_{0i}\dot{q}_0 + J_{mi}\dot{q}_m \tag{27}$$

A particular point of interest is the origin of the end-effector reference frame $\mathcal{J}_{EE}$ denoted by x_{EE}. The end-effector Jacobians J_{0EE} and J_{mEE} can be also obtained and thus, the end-effector velocities can then be computed as follows.

$$t_{EE} = J_{0EE}\dot{q}_0 + J_{mEE}\dot{q}_m \tag{28}$$

2.2. Dynamics

The torques n_i and forces f_i in operational space applied to the ith link center-of-mass can also be encapsulated into a six-dimensional wrench vector w_i.

$$w_i = \begin{bmatrix} n_i \\ f_i \end{bmatrix} \tag{29}$$

The NE equations can then be written for each link as follows.

$$M_i = \begin{bmatrix} I_i & 0_{3,3} \\ 0_{3,3} & m_i 1_{3,3} \end{bmatrix} \tag{30}$$

$$M_i \dot{t}_i + \dot{M}_i t_i = w_i \tag{31}$$

$$\dot{M} = \begin{bmatrix} \omega_i \times I_i & 0_{3,3} \\ 0_{3,3} & 0_{3,3} \end{bmatrix} \tag{32}$$

The compact NE equations can then be generalized for all the system links as follows.

$$M\dot{t} + \dot{M}t = w \tag{33}$$

$$w = \begin{bmatrix} w_0 & w_1 & \dots & w_n \end{bmatrix}^T \tag{34}$$

$$M = \mathrm{diag}\left(\begin{bmatrix} M_0 & M_1 & \dots & M_n \end{bmatrix}^T \right) \tag{35}$$

The wrenches can be decomposed in non-contributing wrenches w^n, that do not contribute the system motion and contributing wrenches w^c. The contributing wrenches can be further divided by the ones provided by the joint actuators w^q (control wrenches) and the wrenches due to external or internal interactions w^F.

$$w = w^c + w^n = w^q + w^F + w^n \tag{36}$$

As the non-contributing wrenches w^n do not perform any work, they can be eliminated by premultiplying the NE Eq. (33) by N^T as $t^T w^n = \dot{q}^T N^T w^n = 0$.

$$N^T \left(M\dot{t} + \dot{M}t \right) = N^T w^q + N^T w^F \tag{37}$$

Applying the twist (Eq. (16)) and the twist-rate (Eq. (24)) vector into Eq. (37) recovers the generalized equation of motion already shown in Eq. (1).

$$H\ddot{q} + C\dot{q} = \tau \tag{38}$$

$$H = N^T M N \tag{39}$$

$$C = N^T \left(M\dot{N} + \dot{M}N \right) \tag{40}$$

$$\tau = \tau^q + \tau^F \tag{41}$$

$$\tau^F = N^T w^F \tag{42}$$

$$\tau^q = N^T w^q \tag{43}$$

The $(n+6) \times (n+6)$ generalized inertia matrix H is symmetric and positive semi-definite and can be subdivided in the 6×6 base-spacecraft inertia matrix H_0, the $n \times n$ manipulator inertia matrix H_m and the $6 \times n$ base-manipulator coupling inertia matrix H_{0m}. Similarly the generalize convective inertia, which is not symmetric, can also be subdivided in analogous subparts (although in this case the C matrix in not symmetric.

$$H = \begin{bmatrix} H_0 & H_{0m} \\ H_{0m}^T & H_m \end{bmatrix} \tag{44}$$

$$C = \begin{bmatrix} C_0 & C_{0m} \\ C_{m0} & C_m \end{bmatrix} \tag{45}$$

3. SOFTWARE ARCHITECTURE

The toolkit source code can be found at [1] and it is being released with a GNU GPL v3 license [37]. The core of the toolkit is a collection of MATLAB functions that provide the basic kinematic and dynamic quantities – mainly the kinematic transformations, the flying- and floating-base Jacobians, and the flying- and floating-base generalized inertia and convective inertia matrices.

These core functions are then assembled to create the $\mathcal{O}(n)$ forward and inverse dynamic algorithms. These algorithms can solve for a floating or a flying base. These forward and inverse dynamic solvers are then used as the basis to create a dynamic simulator. These already implemented simulators are also available as Simulink blocks (the system's plant).

Another set of MATLAB functions is used for control. Multiple types of controllers can be build easily using these functions. A resolved motion, a zero and desired reaction maneuver, and a transpose Jacobian have already been developed. The control input required to counteract the base reaction due to the manipulator motion can also be computed. These controllers are also available as Simulink blocks, which can then be used to control the plant. Using the Simulink automatic code generation capabilities, a C version of these controllers can be generated and later compiled for a specific embedded hardware target running a Real-Time Operating System Environment.

Finally, another set of functions are used to analyze the properties of a spacecraft with a robotic arm. Algorithms used to analyze the workspace and the kinematic manipulability have already been implemented. SPART aims to become an end-to-end solution to tackle spacecraft robotics problems (from analysis to simulation and control).

The toolkit as it currently stands is mainly standalone in an effort to facilitate its interaction with other third-party tools

is being made. Integration with other robotic frameworks as ROS (a set of software libraries and tools that help you build robot applications) [38], GAZEBO (a multi-robot simulator for outdoor environments) [39] and MoveIt! (software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation) [40] is being carried out. These other frameworks could complement SPART adding visualization, path-planning, navigation and networking capabilities.

Currently the spacecraft-manipulator description is provided to SPART using a custom structure. Integration with the commonly used URDF, SDF, and VSK file formats is envisioned in the near-future.

4. EXAMPLE APPLICATIONS

A Desired Reaction Maneuver and a workspace analysis will be used to briefly illustrate the capabilities of the toolkit.

4.1. Desired-Reaction-Maneuver

If the manipulator has additional degrees-of-freedom, these can be used to partially or totally impose a desired base-spacecraft reaction. This concept was first proposed by Yoshida [41] as a means to avoid base-spacecraft reaction and is commonly known as Zero Reaction Maneuver (ZRM). This concept is extended here to obtain any desired base reaction maneuver.

In general, to transmit a desired reaction to the base Eq. (46), which includes the requested trajectory t_{x_i} and the desired base reaction $\dot{q}_0$ can be used. In Eq. (46) $\mathcal{M}^{'}$ denotes the system's initial angular and linear momentum.

$$\begin{bmatrix} q_0 \\ t_{x_i} \end{bmatrix} - \begin{bmatrix} H_0^{-1} \\ J_{0x_i} H_0^{-1} \end{bmatrix} \mathcal{M}^{'} = \begin{bmatrix} -H_0^{-1} H_{0m} \\ J_{mx_i} - J_{0x_i} H_0^{-1} H_{0m} \end{bmatrix} \dot{q}_m \tag{46}$$

Although Eq. (46) assumes that $\dot{q}_0$ and t_{x_i} have to be fully controlled, partial control is also an option. Partial control can be imposed by only including the portions of interest of the state vector ($\dot{q}_0$ and t_{x_i}) and its associated matrices portions into Eq. (46).

To obtain the joint velocities $\dot{q}_m$ required to obtain the requested t_{x_i} motion while transmitting a reaction to the base that forces the prescribed $\dot{q}_0$ motion, Eq. (47) can be employed.

$$\dot{q}_m = \begin{bmatrix} -H_0^{-1} H_{0m} \\ J_{x_i}^{\star} \end{bmatrix}^{-1} \left(\begin{bmatrix} \dot{q}_0 \\ t_{x_i} \end{bmatrix} - \begin{bmatrix} H_0^{-1} \\ J_{0x_i} H_0^{-1} \end{bmatrix} \mathcal{M}^{'} \right) \tag{47}$$

To solve this Desired Reaction Maneuver (DRM) an inverse needs to be computed, and the process is susceptible to encounter singularities (in even less intuitive locations than when inverting only Jacobians). To obtain stable solutions Eq. (48), which uses the adjugate of the G matrix and in which k is an arbitrary scalar, can be used. If $k = 1/\det(G)$ the same results are recovered as when using the inverse, but imposing an upper limit on k will bound $\dot{q}_m$ in the vicinity of the singularity. This method is recommended when computing the solutions.

		Base-spacecraft	Links
Mass	kg	100	10
Size	[cm]	75×75	50×5

Table 2: Parameters of the exemplary case.

$$\dot{q}_m = k \,\mathrm{adj}\,(\mathrm{G}) \left(\begin{bmatrix} \dot{q}_0 \\ t_{x_i} \end{bmatrix} - \begin{bmatrix} H_0^{-1} \\ J_{0x_i} H_0^{-1} \end{bmatrix} \mathcal{M}^{'} \right) \tag{48}$$

$$G = \begin{bmatrix} -H_0^{-1} H_{0m} \\ J_{x_i}^{\star} \end{bmatrix} \tag{49}$$

A ZRM is then a subset of the general DRM case presented here when the spacecraft-manipulator system is initially at rest $\mathcal{M}^{'} = 0$ and when no reaction on the base-spacecraft is desired $\dot{q}_0 = 0$ (or only no-angular-velocity reaction $\omega_0 = 0$).

It is interesting to note that the DRM can be used to counteract the presence of small initial momentum $\mathcal{M}^{'}$ (due to residual base-spacecraft angular motion for example) and provide the required manipulator motion which keeps the base-spacecraft and the x_i point fixed $\dot{q}_0 = t_{x_i} = 0$. Another important aspect is that these techniques can be also used for spacecraft with multiple manipulator arms. The second arm can provide the extra degrees-of-freedom to implement the ZRM or DRM.

An example of a DRM with a four-link manipulator is shown in Fig. 4. The end-effector describes a prescribed motion of constant orientation (zero degrees). During this maneuver, the additional degree-of-freedom is used to keep the base-spacecraft pointing towards the end-effector (imposing a desired reaction) while the base-spacecraft position is left uncontrolled. For this maneuver SPART has been used for the propagation of state variables and to generate the control inputs.

4.2. Workspace and Manipulability analysis

SPART also contains functions that analyze a particular spacecraft-manipulator robotics configuration. A planar example, considering identical links connected by revolute joints, is used (similar to what is used in other references [42]). The parameters used are provided in Table 2. The center-of-mass of each link will be located in its geometric center, and its inertia is obtained by assuming a homogeneous bodies. The first joint will have a ± 90 degree rotation limit and for subsequent joints a ± 115 degree limit is assumed.

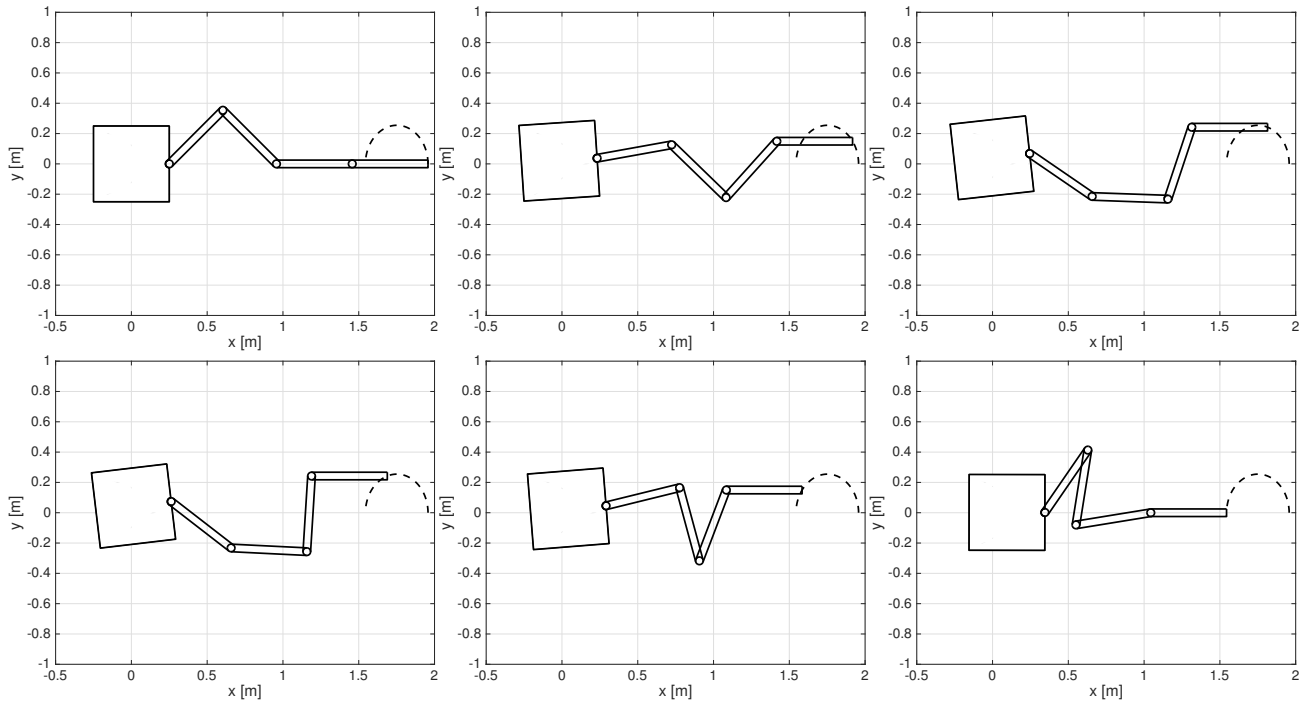


Fig. 4: DRM example evolution.

An example of reachable fixed vehicle workspace is shown in Fig. 5. Note that if a third link is added to the manipulator, the reachable workspace is greatly expanded as shown in Fig. 5b. Additionally with a third link the attitude of the end-effector can also be controlled (and a dextrous workspace could be defined).

Defining the workspace for a floating base is more challenging. The straight-path workspace is defined as the volume that can be reached if the point of interest (e.g. end-effector) is moved in a straight line from the starting configuration. This volume is obviously continuous and its boundary can be numerically obtained by subsequently moving the manipulator until it is no longer possible to maintain a straight line motion. Geometric and joint limits can be included when obtaining this workspace. An example of this workspace is shown in Fig. 6. This workspace considers the spacecraft-manipulator system in a certain configuration, so it is tied to this particular starting configuration.

The kinematic, static and dynamic manipulability ellipsoids provide quantitative measures of the ability to move a manipulator point x_i (usually the end-effector) in a particular direction and to obtain the velocity transmission ratio (kinematic manipulability ellipsoid) or the force transmission ratio (static and dynamic manipulability ellipsoids) from joint space to operational space along that direction. These ellipsoids are a tool to seek optimal manipulator configurations to perform a task in a certain direction (e.g. exert a force, an acceleration or a velocity to an object and achieve a certain accuracy on those magnitudes).

For the kinematic manipulability ellipsoid, joint velocities of unit norm $\dot{q}^T\dot{q} = 1$ are considered. Using the Jacobians to map the joint velocities into operational space the following ellipsoid equation is obtained [33].

$$t_{x_i}^T \left(J_{x_i} J_{x_i}^T\right)^{-1} t_{x_i} = 1 \tag{50}$$

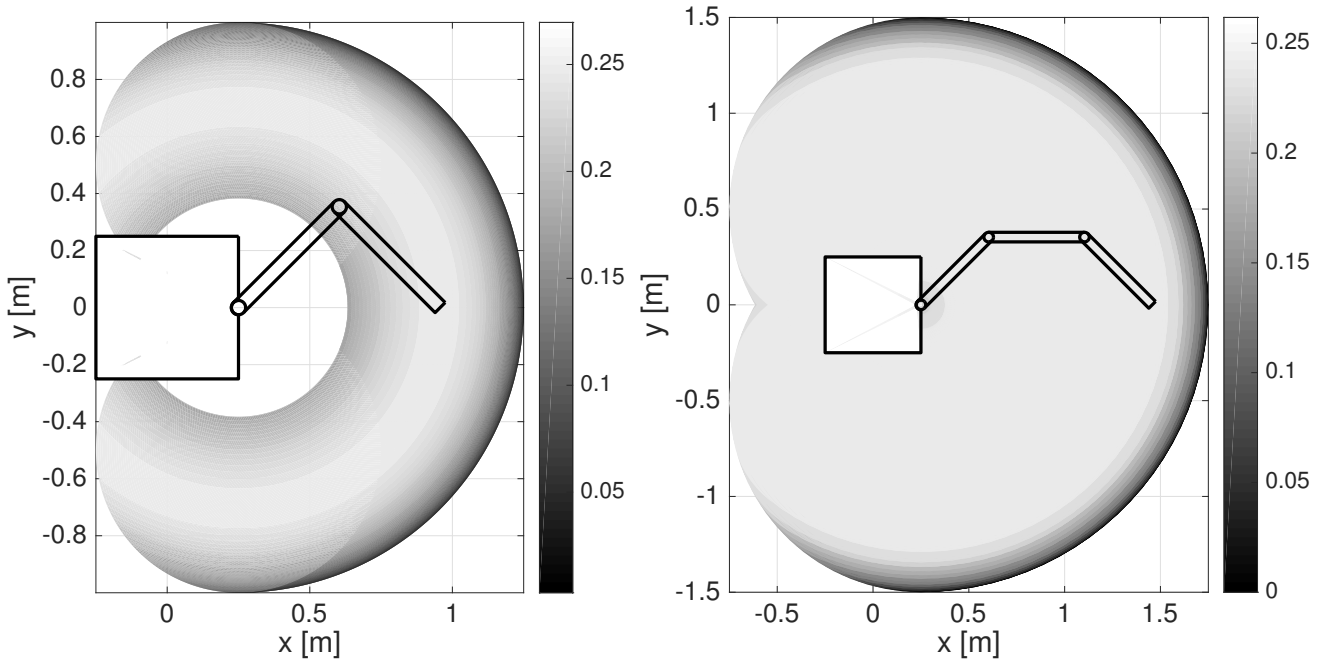
The quadratic $J_{x_i} J_{x_i}^T$ term defines the ellipsoid. The eigenvectors of this term define the principal axis of the ellipsoid and the eigenvalues are the a^2, b^2 and c^2 semi-axes of the ellipsoid. In this case the ellipsoid is centered around x_i. This ellipsoid only depends on the Jacobian, and thus it is only a function of the spacecraft-manipulator geometry and current configuration (joint state q). An example is provided in Fig. 7.

The volume of an ellipsoid is proportional to the product of its semi-axis and thus the volume of the ellipsoid is proportional to the kinematic manipulability measure km_{x_i}. Examples of this measure is provided in Fig. 5.

$$\mathrm{km}_{x_i} = \sqrt{\det\left[J_{x_i} J_{x_i}^T\right]} \tag{51}$$

5. CONCLUSIONS

An open-source spacecraft robotics toolkit has been developed and realized. The toolkit includes function that determine, using an $\mathcal{O}(n)$ algorithm, the kinematic and dynamic quantities based on the Decoupled Natural Orthogonal Complement matrix. This forms the basis of recursive forward



(a) (b)

Fig. 5: Kinematic manipulability measure and reachable workspace.

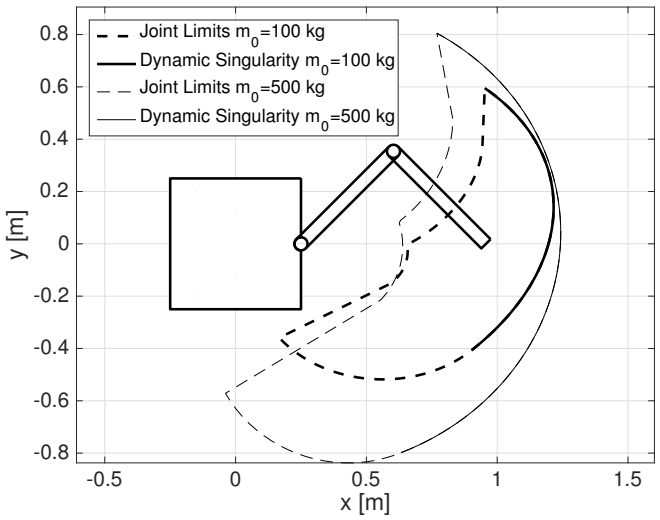


Fig. 6: Straight-path reachable workspace.

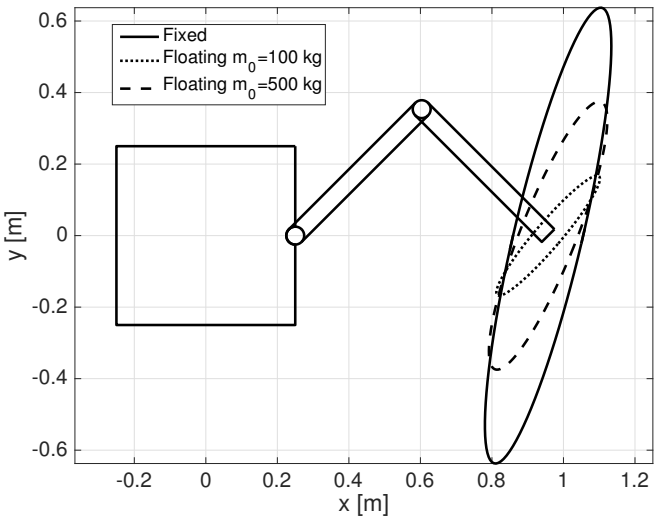


Fig. 7: Fixed-base kinematic manipulability ellipse.

and inverse dynamic algorithms. These are available as MATLAB functions or Simulink blocks. Multiple control algorithms, which build upon the kinematic and dynamic quantities have been implemented and are also available in the form of MATLAB functions or Simulink blocks. Using the Simulink's automatic code generation capability, a C implementation of these controllers can be obtained and later compiled to be is executed on embedded hardware. Tools to analyze the properties of the spacecraft-manipulator have also been implemented.

6. REFERENCES

[1] SPAcecraft Robotics Toolkit, "https://github.com/nps-srl/spart," .

[2] S. K. Saha, "Dynamics of serial multibody systems using the decoupled natural orthogonal complement matrices," *Journal of Applied Mechanics*, vol. 66, no. 4, pp. 986–996, 12 1999.

[3] S.V. Shah, S.K. Saha, and J.K. Dutt, "Modular framework for dynamic modeling and analyses of legged robots," *Mechanism and Machine Theory*, vol. 49, pp. 234 – 255, 2012.

[4] Subir Kumar Saha and Werner O Schiehlen, "Recursive kinematics and dynamics for parallel structured closed-loop multibody systems*," *Mechanics of Structures and Machines*, vol. 29, no. 2, pp. 143–175, 2001.

[5] Santosha Kumar Dwivedy and Peter Eberhard, "Dynamic analysis of flexible manipulators, a literature review," *Mechanism and Machine Theory*, vol. 41, no. 7, pp. 749 – 777, 2006.

[6] Marcello Romano and Jason Hall, "A testbed for proximity navigation and control of spacecraft for on-orbit assembly and reconfiguration," in *Proceedings of the AIAA Space 2006 Conference and Exhibit*, 2006, pp. 1–11.

[7] Marcello Romano, David A. Friedman, and Tracy J. Shay, "Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target," *Journal of Spacecraft and Rockets*, vol. 44, no. 1, pp. 164–173, 2007.

[8] Jason S. Hall and Marcello Romano, "Novel robotic spacecraft simulator with mini-control moment gyroscopes and rotating thrusters," in *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*. 2007, pp. 1–6, IEEE.

[9] Riccardo Bevilacqua, Jason S. Hall, James Horning, and Marcello Romano, "Ad hoc wireless networking and shared computation for autonomous multirobot systems," *Journal of Aerospace Computing, Information, and Communication*, vol. 6, no. 5, pp. 328–353, 2016/01/07 2009.

[10] Riccardo Bevilacqua, Andrew Caprari, Jason Hall, and Marcello Romano, "Laboratory experimentation of multiple spacecraft autonomous assembly," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2009.

[11] Claudio Lugini and Marcello Romano, "A ballistic-pendulum test stand to characterize small cold-gas thruster nozzles," *Acta Astronautica*, vol. 64, no. 5, pp. 615–625, 2009.

[12] Fabio Curti, Marcello Romano, and Riccardo Bevilacqua, "Lyapunov-based thrusters' selection for spacecraft control: analysis and experimentation," *Journal of guidance, control, and dynamics*, vol. 33, no. 4, pp. 1143–1160, 2010.

[13] Jason S. Hall and Marcello Romano, *Laboratory experimentation of guidance and control of spacecraft during on-orbit proximity maneuvers*, INTECH Open Access Publisher, 2010.

[14] Marco Ciarcia, Alessio Grompone, and Marcello Romano, "A near-optimal guidance for cooperative docking maneuvers," *Acta Astronautica*, vol. 102, pp. 367–377, 2014.

[15] Richard Zappulla II, Hyeongjun Park, Josep Virgili Llop, and Marcello Romano, "Experiments on autonomous spacecraft rendezvous and docking using an adaptive artificial potential field approach," in *26th AAS/AIAA Space Flight Mechanics Meeting, Napa, CA. 14-18 February 2016. AAS 16-459*, 2016.

[16] Josep Virgili-Llop, Jerry Drew, and Marcello Romano, "Design and parameter identification by laboratory experiments of a prototype modular robotic arm for orbiting spacecraft applications," in *6th International Conference on Astrodynamics Tools and Techniques. 14-17 March 2016, Darmstad, Germany.*, 2016.

[17] Open Dynamics Engine, "http://www.ode.org," .

[18] Bullet, "http://bulletphysics.org/wordpress/," .

[19] Simbody, "https://simtk.org/home/simbody/," .

[20] Dynamic Animation and Robotics Toolkit, "http://dartsim.github.io," .

[21] Christos H Papadimitriou, *Computational complexity*, John Wiley and Sons Ltd., 2003.

[22] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line computational scheme for mechanical manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 102, no. 2, pp. 69–76, 06 1980.

[23] M. W. Walker and D. E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, no. 3, pp. 205–211, 09 1982.

[24] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *The International Journal of Robotics Research*, vol. 2, no. 1, pp. 13–30, 1983.

[25] C.A. Balafoutis, R.V. Patel, and P. Misra, "Efficient modeling and computation of manipulator dynamics using orthogonal cartesian tensors," *Robotics and Automation, IEEE Journal of*, vol. 4, no. 6, pp. 665–676, Dec 1988.

[26] Roy Featherstone, *Rigid Body Dynamics Algorithms*, Springer, 2009.

[27] Thomas R. Kane and David A. Levinson, "The use of kane's dynamical equations in robotics," *The International Journal of Robotics Research*, vol. 2, no. 3, pp. 3–21, 1983.

[28] G. Rodriguez, "Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 6, pp. 624–639, December 1987.

[29] Roy Featherstone, "A divide-and-conquer articulated-body algorithm for parallel o(log(n)) calculation of rigid-body dynamics. part 1: Basic algorithm," *The International Journal of Robotics Research*, vol. 18, no. 9, pp. 867–875, 1999.

[30] Rudranarayan M. Mukherjee and Kurt S. Anderson, "Orthogonal complement based divide-and-conquer algorithm for constrained multibody systems," *Nonlinear Dynamics*, vol. 48, no. 1, pp. 199–215, 2006.

[31] Jorge Angeles and Sang Koo Lee, "The formulation of dynamical equations of holonomic mechanical systems using a natural orthogonal complement," *Journal of Applied Mechanics*, vol. 55, no. 1, pp. 243–244, 03 1988.

[32] S.K. Saha, "A decomposition of the manipulator inertia matrix," *Robotics and Automation, IEEE Transactions on*, vol. 13, no. 2, pp. 301–304, Apr 1997.

[33] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo, *Robotics Modelling, Planning and Control*, Springer, 2009.

[34] Jacques Denavit and Richard Scheunemann Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, 1955.

[35] S. Ali A. Moosavian and Evangelos Papadopoulos, "Free-flying robots in space: an overview of dynamics modeling, planning and control," *Robotica*, vol. 25, pp. 537–547, 9 2007.

[36] Angel Flores-Abad, Ou Ma, Khanh Pham, and Steve Ulrich, "A review of space robotics technologies for on-orbit servicing," *Progress in Aerospace Sciences*, vol. 68, pp. 1 – 26, 2014.

[37] Free Software Foundation, "GNU General Public License version 3," http://www.gnu.org/licenses/gpl.html, 2007.

[38] Robot Operating System, "http://www.ros.org," .

[39] GAZEBO, "http://gazebosim.org," .

[40] MoveIt!, "http://moveit.ros.org," .

[41] K. Yoshida, K. Hashizume, and S. Abiko, "Zero reaction maneuver: flight validation with ets-vii space robot and extension to kinematically redundant arm," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, 2001, vol. 1, pp. 441–446 vol.1.

[42] Yoji Umetani and Kazuya Yoshida, "Workspace and manipulability analysis of space manipulator," *Trans. Soc. Instrum. Control Eng*, vol. E-1, no. 1, pp. 116–123, 2001.