ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/305369198

Reactionless Maneuvering of a Space Robot in Precapture Phase

Article *in* Journal of Guidance Control and Dynamics · July 2016

DOI: 10.2514/1.G001828

CITATIONS
0

READS
31

5 authors, including:

S. V. Shah
Indian Institute of Technology Jodhpur
42 PUBLICATIONS **68** CITATIONS
SEE PROFILE

Madhava Krishna
International Institute of Information Techno…
162 PUBLICATIONS **535** CITATIONS
SEE PROFILE

A. K. Misra
McGill University
253 PUBLICATIONS **2,224** CITATIONS
SEE PROFILE

All content following this page was uploaded by S. V. Shah on 15 December 2016.

The user has requested enhancement of the downloaded file. All in-text references underlined in blue are added to the original document and are linked to publications on ResearchGate, letting you access and read them immediately.

Reactionless Maneuvering of a Space Robot in Precapture Phase

Francis James[1], Suril V. Shah[2], Arun K. Singh[3], K. Madhava Krishna[4], and Arun K. Misra[5]

I. Introduction

Space robots can be used to perform several on-orbit tasks such as capturing space debris, servicing of satellites and refuelling. The proliferation of satellites as well as growing interest in debris capture makes it necessary to have robots that can perform these tasks autonomously [1].

The dynamics of robots in space differ from that of a grounded robot. The coupling of the arms and the base of a space robot creates reaction forces and moments on the base whenever the arms execute a maneuver, causing the base to rotate and translate in accordance with the laws of conservation of linear and angular momenta. However, it is generally desirable to keep the attitude of the base fixed relative to the sun and the earth (or other bodies) for navigation and communication purposes, or to maintain the target in the field of view of the sensors. A change in celestial orientation may also result in loss of communication with the data relay satellite or the ground station. While attitude control using thrusters may be used, such operations consume fuel which is mainly reserved for orbital maneuvers. It has also been shown that manipulation without the use of attitude controllers is more robust [2]. The translation of the base however, does not pose significant side effects [3]. Hence, researchers have focused on robotic manipulation with zero or minimal change in attitude, which is termed reactionless manipulation [4].

The capture of objects for servicing or of tumbling targets such as debris is divided into different

---

[1] M.S. Student (Robotics), College of Engineering, Oregon State University, Corvallis, OR 97330, USA.
[2] Assistant Professor, Mechanical Department, Indian Institute of Technology, Jodhpur, Rajasthan, 342011, India
[3] Postdoctoral Fellow, Bio-Medical Robotics Lab, Ben Gurion University, Beer Sheva, 8410501, Israel
[4] Associate Professor, Robotics Research Center, International Institute of Information Technology, Hyderabad, AP 500032, India
[5] Professor, Mechanical Department, McGill University, Montreal, Quebec, H3A 0C3, Canada

phases, namely the precapture, contact, post capture and compound stabilization [5]. This work focuses on reactionless manipulation of a space robot in the precapture phase. Each link of the robotic system is referenced by an index i, where index $i = 0$ corresponds to the base. The objective of a reactionless maneuver is to intercept an object by ensuring that the end-effectors reach a desired position with a desired velocity at a specified time. However, in the absence of thrusters for control, the non-integrability of the angular momentum makes the system nonholonomic. This complicates the planning and control of such systems [2].

Early strategies for zero reaction maneuvers have been tested on the single arm ETS VII space robot [6] to show their feasibility. This forms a seminal work that uses the concept of Reaction Null Space (RNS). RNS based methods have also been used in other works [7–9]. Apart from the ETS VII experiment, some experiments have also been conducted on parabolic flights [10] wherein the yaw angle of the base was controlled. Coordinate partitioning [11] offers an elegant solution for path planning using dependent variables which allows for planning using polynomials. However, both the RNS based method and Coordinate partitioning suffer from algorithmic singularities [12] when an end-effector trajectory is followed while maintaining constant attitude. These singularities are hard to predict and occur frequently enough to be a cause for concern. Other methods such as creating a disturbance map [13] have also been used to control attitude disturbances. Smooth planning using polynomials to control the configuration of the arms as well as the base was proposed in [14], but this method controls the orientation of the base only at the initial and final time and hence, can cause large base attitude disturbances during the maneuver. Recently, adaptive methods for reactionless manipulation have been developed [15] [8], however, these works focus on post-capture and do not specify a method to plan motion for the approach phase. Detumbling operations have been presented in [16] which are useful either after capture of an uncontrolled tumbling object or before the capture operation begins if the floating base system is spinning. Using intermediate points for reactionless manipulation, [9] has succeeded in avoiding singularities, but a systematic framework for selecting these points has not been provided. Yet another method focuses on using local optimization techniques where algorithmic singularities are avoided indirectly by introducing joint acceleration limits [17]. However, it may fail to find a path when the attitude disturbance is

strictly set to zero even when there is redundancy.

This paper presents a method that can be used to capture the target at the specified time while avoiding algorithmic as well as Jacobian singularities for a system that starts from rest. This method could alternatively be used to find a suitable initial configuration for a given desired motion state. Here, we define the motion state of the robot as the position and velocity of a point on the base as well as the joint positions and velocities of the arms. It allows additional constraints such as limits on angles, acceleration and jerk to be satisfied while avoiding collisions as well. The framework proceeds in two parts. In the first part, a reactionless trajectory is computed between the given start and the goal state by using Rapidly exploring Random Trees (RRT) [18] with control based sampling in which an optimization problem is solved for determining the control inputs. The objective function acts as a control equation that helps in biasing when a reverse time simulation [19] is performed. In most cases, this biasing is key to reducing the time required to find a solution. The constraints that must be satisfied while computing the final motion state are also described. The combination of these techniques for planning forms one of the contributions of this work.

In the second stage, the final time constraint is imposed by subjecting the obtained reactionless trajectory to a time scaling transformation. The temporal profile of the trajectory is scaled nonlinearly, thus changing the velocity and its derivatives along a geometric path to meet the time constraints. The time scaling transformation used is built on our earlier work on non-linear time scaling [20]. However in contrast to this cited work, we modify the definition of the time scaling function and describe a method for choosing its parameters. The proposed time scaling can be applied to any geometric path where the velocities $\dot{x}$ follow a constraint equation of the form $A\dot{x} = 0$ or $A\dot{x} \leq c$, where A is a matrix and c is a vector. Since this is not specific to a space system, this forms another, perhaps more widely applicable contribution of this work.

A. Basic Assumptions

The method assumes that the whole system starts from rest, the desired motion state with which to capture a target is known, and that a reactionless path between the initial and desired final motion states of the robot exists.

II. Mathematical Preliminaries: Reactionless Manipulation

For an n Degrees-Of-Freedom (n-DOF) manipulator arm mounted on a floating-base, the angular momentum ($\boldsymbol{l}$) can be written as [6]

$$\boldsymbol{l} = \tilde{\boldsymbol{I}}_b \boldsymbol{\omega}_0 + \tilde{\boldsymbol{I}}_{bm} \dot{\boldsymbol{\theta}} + \boldsymbol{r_0} \times \boldsymbol{p}. \tag{1}$$

where $\tilde{\boldsymbol{I}}_{bm}\dot{\boldsymbol{\theta}}$ is the coupling angular momentum, $\boldsymbol{\omega}_0$ is the angular velocity of the base, $\boldsymbol{r_0}$ is the position of the Centre of Mass (COM) or fixed point of the base and $\boldsymbol{p}$ is the linear momentum. The constraint for reactionless manipulation can then be obtained by substituting $\boldsymbol{\omega_0} = 0$ in (1). If the system starts from rest, $\boldsymbol{l} = \boldsymbol{p} = 0$. This results in

$$\tilde{\boldsymbol{I}}_{bm} \dot{\boldsymbol{\theta}} = 0. \tag{2}$$

Even though the method presented in this work uses planning in joint space, the desired state is often specified in terms of the end-effectors' position and velocity at the instant of capture. The Generalized Jacobian Matrix (GJM) is used in order to obtain the joint velocities that correspond to the end-effector velocities ($\boldsymbol{t}_e$) at the instant of capture [21], i.e.,

$$\boldsymbol{t}_e = \boldsymbol{J}_g \dot{\boldsymbol{\theta}}, \ \textit{where} \ \boldsymbol{J}_g = \left(\boldsymbol{J}_{me} - \boldsymbol{J}_{be} \boldsymbol{I}_b^{-1} \boldsymbol{I}_{bm} \right). \tag{3}$$

In (3), $\boldsymbol{J}_g \in R^{m \times n}$ is the GJM, and $\boldsymbol{J}_{be}$ and $\boldsymbol{J}_{me}$ are the Jacobian matrices for the base and manipulator, respectively. $\boldsymbol{J}_g$ has a similar interpretation to the Jacobian ($\boldsymbol{J}_{me}$) of an earth-based robot. However, $\boldsymbol{J}_g$ contains additional inertia terms associated with the system's dynamics. Using (3), the joint velocities are obtained as $\dot{\boldsymbol{\theta}} = \boldsymbol{J}_g^{+} \boldsymbol{t}_e$, where $\boldsymbol{J}_g^{+}$ is the pseudoinverse of $\boldsymbol{J}_g$, assuming the DOF of the arms allows the desired Cartesian velocity to be attained, i.e., $n \geq m$ and $rank[\boldsymbol{J}_g] = rank[\boldsymbol{J}_g \ \boldsymbol{t}_e]$. It may be noted that for $rank[\boldsymbol{J}_g] = n = m$, $\boldsymbol{J}_g^{+} = \boldsymbol{J}_g^{-1}$.

III. Motion Planning for Reactionless Manipulation

Most of the path planning methods proposed in literature suffer from algorithmic singularities. Algorithmic singularities [12] are different from dynamic singularities associated with space robots and occur when the end- effector's motion conflicts with constraints for reactionless manipulation. These singularities are difficult to predict. Apart from this, obstacles in the workspace may necessitate algorithms that can find trajectories lying exclusively in the free space.

In this work, we present a method which takes care of algorithmic singularities while allowing us to specify the pose of the end-effectors, and to satisfy additional constraints such as collision avoidance. For this, we first identify the final desired configuration and velocity and use a planning algorithm to connect it to the initial configuration.

A. Desired Configuration

Since the system starts from rest, we can use a simple position error based control law to bias the planner while doing a reverse time simulation. First however, the final state corresponding to the final end-effectors' pose and velocity needs to be found.

If the robot moves in a reactionless manner, the rotation of the base is always zero. So, the kinematic equations of the space robot under study are an exclusive function of the joint angles, $\boldsymbol{\theta}$, and the position of a point on the base, say, its COM, $\mathbf{c}_0$. Since no external force or torque acts on the system, the centre of mass of the entire system remains fixed irrespective of the maneuver executed. A valid solution must keep the COM fixed and attain the desired end-effector position and orientation. Since the COM is fixed, the base will have a unique position if the joint angles are fixed. These equations can be written as

$$\begin{bmatrix} f(\mathbf{c}_0, \theta) \\ h(\mathbf{c}_0, \boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \mathbf{a}_e \\ \mathbf{c}_{com} \end{bmatrix} \tag{4}$$

where $f(\mathbf{c}_0, \theta)$ represents the kinematic equations, $h(\mathbf{c}_0, \boldsymbol{\theta})$ represents the centre of mass equations, $\mathbf{a}_e$ contains the position and orientation of the end-effectors and $\mathbf{c}_{com}$ is the constant vector corresponding to the COM of the system. Not all solutions are guaranteed to be reachable from the initial configuration. However, since the trajectory planning uses a sampling method that is probabilistically complete, it ensures that if a path exists, it can be found. The method proceeds under the assumption that the selected solution is attainable from the initial state, which is generally true when the manipulator has a high degree of redundancy. Once the initial and final states are found in joint space, they need to be connected. Since the planning is done in joint space, Jacobian singularities do not arise.

B. Control Input Using Optimization

The planning stage must ensure that only reactionless maneuvers are executed. For this, the selected control input should result in velocities that lie within the null space of the modified coupling inertia matrix in (2). It is clear that for the existence of this null space, the manipulator must be redundant. For any given configuration, a linear combination of the basis vectors that span this null space will result in joint velocities that ensure zero base rotation.

$$\dot{\boldsymbol{\theta}} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + ... + \alpha_u \mathbf{v}_u. \tag{5}$$

Here, α_i are scalars, $\mathbf{v}_i \in R^{n\times 1}$ are vectors that span the null space and are a function of the joint angles. u is the dimension of the null space. In matrix form, we can express this equation as

$$\dot{\boldsymbol{\theta}} = \mathbf{V}\boldsymbol{\alpha}, \tag{6}$$

where $\mathbf{V} \in R^{n\times u}$ is the matrix containing the basis vectors $\mathbf{v}_i$'s and $\boldsymbol{\alpha} \in R^{u\times 1}$ is the vector containing the control inputs α_i's.

The actual joint velocities are chosen to minimize the square of the norm of the difference between the current and required velocity vectors.

$$\min ||\mathbf{V}\boldsymbol{\alpha} - \dot{\boldsymbol{\theta}}_{req}||^2 \tag{7}$$

Equation (7) reduces the determination of the control input $\boldsymbol{\alpha}$ to an optimization problem. By integrating these velocities over time t, we can evolve the state of the system. In (7), $\dot{\boldsymbol{\theta}}_{req}$ is the required velocity at a point of time, and is obtained as

$$\dot{\boldsymbol{\theta}}_{req} = -\lambda(\boldsymbol{\theta}_{current} - \boldsymbol{\theta}_{desired}), \tag{8}$$

where λ is a constant which controls the rate of convergence, and $\boldsymbol{\theta}_{current}$ is the set of current joint angles. For a chosen desired configuration, $\boldsymbol{\theta}_{desired}$ is a constant. $\dot{\boldsymbol{\theta}}_{current}$ (distinct from $\dot{\boldsymbol{\theta}}_{req}$) is equal to $\mathbf{V}\boldsymbol{\alpha}$ obtained from (7). The use of (7)-(8) as a control law allows the system to reach the desired state with zero terminal velocities when the desired joint trajectory is tracked.

In the optimization problem, it is possible to impose constraints such as limits on acceleration and higher order derivatives of the velocity. In general, by constraining the $(k+2)^{th}$ derivative, we

can ensure that the $(k+1)^{th}$ derivative is continuous and the k^{th} derivative is both differentiable and continuous. In the simulations presented in this work, a decision was taken to keep the acceleration continuous and the velocity both continuous and differentiable by imposing a limit on the jerk. An additional constraint on acceleration was imposed keeping physical limitations of the actuators in mind. These constraints are expressed as

$$|\boldsymbol{\theta}''|_{t_i} = |\frac{\dot{\boldsymbol{\theta}}(t_i) - \dot{\boldsymbol{\theta}}(t_{i-1})}{t_i - t_{i-1}}| \leq \ddot{\boldsymbol{\theta}}_{max}, \tag{9}$$

$$|\boldsymbol{\theta}'''|_{t_i} = |\frac{\dot{\boldsymbol{\theta}}(t_i) - 2\dot{\boldsymbol{\theta}}(t_{i-1}) + \dot{\boldsymbol{\theta}}(t_{i-2})}{(t_i - t_{i-1})^2}| \leq \dddot{\boldsymbol{\theta}}_{max}, \tag{10}$$

where $\dot{\boldsymbol{\theta}}(t_i) = \mathbf{V}\boldsymbol{\alpha}$. In (9) and (10), $\ddot{\boldsymbol{\theta}}_{max}$ and $\dddot{\boldsymbol{\theta}}_{max}$ are the maximum allowed angular acceleration and jerk of the joints respectively.

Algorithm 1 Initial Trajectory Planning Algorithm

procedure TRAJECTORY PLANNING
  $Tree \leftarrow final\_State$
  $count \leftarrow 1$
  $k \leftarrow bias$
  while $Tree.latest\_node\_state \neq desired\_state$ **do**
    if modulo(count,k)=0 **then**
      $\theta_{rand} \leftarrow desired\_Configuration$
    else
      $\theta_{rand} \leftarrow random\_Configuration$
    $[\theta_{near}, \dot{\theta}_{near}] \leftarrow nearest\_State(\theta_{rand})$
    $\dot{\theta}_{req} \leftarrow compute\_Required(\theta_{rand}, \theta_{near})$
    $V\alpha \leftarrow optimize\_Algorithm(\theta_{near}, \dot{\theta}_{near}, \dot{\theta}_{req})$
    $\theta_{new} \leftarrow new\_State(\theta_{near}, V\alpha, \delta t)$
    $check = check\_No\_Collisions(\theta_{new})$
    if *check* **then**
      $Tree.add\_Vertex \leftarrow .add\_vertex(\theta_{new})$
      $Tree.add\_edge \leftarrow (\theta_{near}, \theta_{new}, V\alpha)$
    $count \leftarrow count + 1$
  return *Tree*

C. Trajectory Planning Algorithm

Sampling based approaches such as Rapidly Exploring Random Trees (RRT) [18] have proved to be very effective in finding a trajectory between two desired states, especially when the motion is highly constrained or obstacles are present in the workspace. RRT is probabilistically complete, it ensures that if a path exists, it can be found.

Path planning in the RRT method consists of a tree that grows from the start node, exploring the configuration space, until a branch reaches the goal node. Creation of a new node (corresponding to a state) consists of an extend operation performed by selecting a random target node within the configuration space, finding the node on the tree closest to the target node and moving direction of the target as far as possible. The RRT algorithm allows fast exploration and inclusion of motion-related constraints. These constraints could be motion constraints of the system itself or obstacles in the workspace. The growth of the tree can be influenced by biasing regions within the configuration space. For example, in goal biasing, the goal state is chosen as target instead of a random node in the extend operation. This helps in guiding the search towards biased regions.

Compared to optimal control strategies which often require several hours of computation time, such sampling based planners can find a solution more quickly. For instance, the results shown in Fig. 3 were obtained in 81.84 s using MATLAB on i5-3317U CPU. Despite the reduction in computation time, trajectories were not computed in real time except in the simplest of cases due to the highly restricted range of motions. However, considerable improvement in run time can be obtained using C++. Additionally, the method assumes that there exists a path which satisfies all constraints. Determining if a path, that satisfies all constraints, exists remains an open problem.

Here, we show how an RRT algorithm can be implemented using the control input and the optimization problem presented in the previous subsection. By using (7) for control based sampling, we evolve a tree that consists of reactionless paths. Since we use reverse time simulations, we first choose the final state of the system to be the starting node. Next, $\boldsymbol{\theta}_{desired}$ in (8) is replaced with $\boldsymbol{\theta}_{rand}$, which is chosen randomly. Adding a bias to the desired initial configuration can reduce the time taken to find a trajectory. Since our system starts from rest and (8), which gives us the required velocity, has a tendency to drive a system to a given $\boldsymbol{\theta}_{desired}$ with zero terminal velocity,

we can leverage this to effectively add a bias by a combination of using reverse time simulations and setting $\boldsymbol{\theta}_{desired}$ to the initial configuration. In simple cases, we can often use a hundred percent bias to the initial configuration in the sampling based path planning algorithm for obtaining quick results. Then we find the node from which the state is to be evolved by using a suitable metric. The determination of an ideal metric is not a simple problem, but the Euclidean distance between configurations seems to suffice for our purpose. The node with the minimal Euclidean distance to the desired configuration is then selected for evolving the state. Equation (7) is used to generate reactionless velocities subject to the constraints in (9) and (10) to evolve the state over a small time interval by integration. If no obstacles are encountered, the new node is added to the tree. The process is repeated till the state is close enough to the desired initial state. When the number of task constraints is high, variants of this algorithm such as LQR-RRT* [22] may be used for faster convergence to a solution. The methodology for planning an initial trajectory with obstacle avoidance using the basic RRT formulation is summarized in Algorithm 1.

The proposed methodology is easy to implement and bypasses singularities as well as obstacles. However, conventionally, RRT or other similar sampling based approaches are not designed to perform planning based on time considerations. In the next section, we describe the concept of non-linear time scaling, which changes the temporal specification of the trajectory obtained through RRT, to accomplish the objective of reaching the final state at a desired instant of time and thus ensure successful capture.

IV. Time Scaling

Time scaling involves changing the current time scale, t to a new time scale, τ in the trajectory definition, $\mathbf{x}(t) = [x(t)\ y(t)]^T$. These transformations do not change the geometric path followed by the joints of the robot, but change the profile of the velocity and its derivatives so the time required for traversing the path changes. Once an initial trajectory connecting the initial and final states is found as a parametric function $\boldsymbol{\theta}(t)$, we can transform it to a function in variable τ, which is the actual time taken after scaling using the equation

$$\frac{dt}{d\tau} = g(.). \tag{11}$$

This transformation allows us to express $\boldsymbol{\theta}(\tau)$ and its derivatives in terms of t.

$$\dot{\boldsymbol{\theta}}(\tau) = \boldsymbol{\theta}'(t)\frac{dt}{d\tau}. \tag{12}$$

A. Scaling Functions

The simplest scaling function is a scalar constant, proposed by Hollerbach [23]. However, it induces a discontinuity between the scaled and un-scaled velocities at the boundary points. Consequently, other scaling functions have been proposed to overcome this issue. Some choices for the scaling function include the B-Spline [24][25] and exponential functions [20]. All these functions must ensure that the scaling value is always positive since time needs to be a positive, monotonically increasing function (i.e $\frac{dt}{d\tau} > 0$). For B-spline functions, which are not naturally monotonic, extra conditions have to be enforced on its control points to ensure positive definiteness. Exponential functions do not require this additional condition which makes them well suited for certain problems such as minimization of time required to traverse a path. However, neither of these functions is suitable when the geometric path has to be traversed in a specified interval of time. The exponential scaling function makes it difficult to find the constants that result in a desired final time, especially when the trajectory is divided into subintervals.The scaling function given in (13) on the other hand, produces a quadratic equation upon integration, which is easy to solve for a desired scaled time τ even with multiple subintervals.

$$\frac{dt}{d\tau} = \frac{1}{at + b}, \tag{13}$$

where a and b are the constants to be solved for. Upon integration, the equation yields

$$\tau = \frac{a}{2}t^2 + bt + c, \tag{14}$$

B. Scaling Subintervals

To gain more flexibility in scaling, we can divide the initial trajectory into subintervals. Note however, that this can make the velocity non differentiable at the boundary points, although within each subinterval the differentiablility introduced by constraining jerk is retained. The trajectory

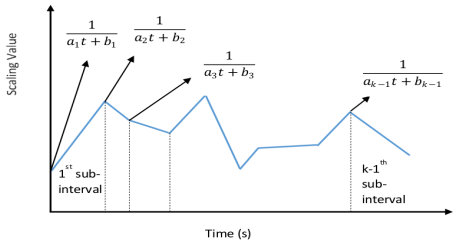

Fig. 1: A Time Scaling Function

found is divided into $(k-1)$ subintervals by k points, and the scaling function is determined for each subinterval. For the r^{th} subinterval, the scaling function is given by

$$\frac{dt}{d\tau} = \frac{1}{a_r t + b_r}. \tag{15}$$

The scaling function $\frac{dt}{d\tau}$ is expressed as a piecewise function of the form (15). The interval $[t_1, t_f]$ is divided into $(k-1)$ subintervals by the points $t_1, t_2, t_3 .... t_k$, where $t_k = t_f$. Fig. 1 shows an example of such a scaling function. As depicted, the piecewise scaling function is continuous, but not differentiable.

To meet our velocity requirements at the end points, the boundary value of the scaling variable $\frac{dt}{d\tau}$ at the beginning of the first subinterval should be s_1 and its value at the end of the last subinterval should be s_2. If it is necessary to keep either initial or final velocity unchanged, we set the corresponding scaling value to one. The value must also be positive at all the boundary points. In order to ensure continuity between adjacent intervals, the value at the end of any subinterval should be equal to the value at the beginning of the next. These requirements are formulated as

$$\frac{dt}{d\tau}(t_1) = s_1 \Rightarrow \frac{1}{a_1 t_1 + b_1} = s_1 > 0,$$
$$\frac{dt}{d\tau}(t_f) = s_2 \Rightarrow \frac{1}{a_{f-1} t_f + b_{f-1}} = s_2 > 0,$$
$$\frac{1}{a_{r-1} t_r + b_{r-1}} = \frac{1}{a_r t_r + b_r} > 0, \forall r \in [2, k-1].$$

Since the scaling function selected is positive at both the boundary points of any subinterval, it is either monotonically increasing or monotonically decreasing throughout the subinterval. This gives us the flexibility to put restrictions on the maximum value of the joint velocity by choosing local maxima of velocity as boundary points and constraining the maximum scaling value at these points if necessary. However, the same cannot be guaranteed for the accelerations. Hence, it becomes

necessary to check scaled acceleration values at points within a subinterval with a suitable time resolution. The accelerations are given by

$$|\ddot{\boldsymbol{\theta}}(\tau)| \leq \ddot{\boldsymbol{\theta}}_{max} \Rightarrow |\ddot{\boldsymbol{\theta}}(t)\left(\frac{dt}{d\tau}\right)^2 + \dot{\boldsymbol{\theta}}(t)\frac{d^2t}{d\tau^2}| \leq \ddot{\boldsymbol{\theta}}_{max}. \tag{16}$$

Using equation (15), it is easy to verify that equation (16) in the r^{th} subinterval can be expressed as

$$|\ddot{\boldsymbol{\theta}}(t)\left(\frac{1}{a_r t + b_r}\right)^2 - \dot{\boldsymbol{\theta}}(t)\frac{a_r}{(a_r t + b_r)^2}| \leq \ddot{\boldsymbol{\theta}}_{max}. \tag{17}$$

The total time taken to traverse the path is given by

$$T = \sum_{i=1}^{k-1} \frac{a_i}{2}(t_{i+1} - t_i)^2 + b_i(t_{i+1} - t_i). \tag{18}$$

This must be imposed as an additional constraint to traverse the path in time T.

In order to find the constants of the piecewise scaling function, we can either use iterative methods to solve a set of equality and inequality constraints, or formulate our requirements as a feasibility program as shown in (19). This allows us to use nonlinear optimization algorithms to find solutions that satisfy all constraints. However, since there is no function that needs to be optimized, the objective function can be chosen to be any arbitrary constant.

$$\begin{aligned}
&\min\ 0 \\
&\text{s.t. } T - \sum_{i=1}^{k-1} \frac{a_i}{2}(t_{i+1} - t_i)^2 + b_i(t_{i+1} - t_i) = 0, \\
&\quad \frac{1}{a_r t_{r+1} + b_r} - \frac{1}{a_{r+1} t_{r+1} + b_{r+1}} = 0, \\
&\quad \frac{dt}{d\tau}(t_1) = s_1,\ \frac{dt}{d\tau}(t_f) = s_2, \\
&\quad |\boldsymbol{\theta}(t)\left(\frac{1}{a_r t + b_r}\right)^2 - \dot{\boldsymbol{\theta}}(t)\frac{a_r}{(a_r t + b_r)^2}| \leq \ddot{\boldsymbol{\theta}}_{max}, \\
&\quad \frac{1}{a_r t_{r+1} + b_r} > 0.
\end{aligned} \tag{19}$$

V. Results and Discussion

The proposed methodology described in this work was tested through simulations of a dual-arm space robot shown in Fig. 2. Each arm is comprised of three rigid links making each arm a 3-DOF

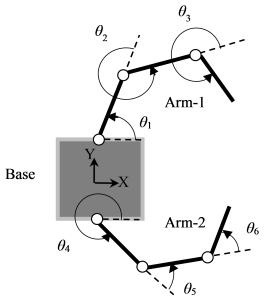

Fig. 2: Dual-arm Planar Space Robot

system. The COM of the base lies at $(0m, 0m)$ at the start of each simulation and corresponds to the geometric centre of the base. The results were tested for both planar and spatial cases. Given an initial configuration, the final state was computed and then the path was planned as described in section III. In each case, the maximum joint position error for termination of the algorithm is about two degrees.

A. Path Planning for Planar Robot

The path planning algorithm was tested for desired goal positions and velocities. A small, randomly selected non-zero end-effector velocity was required to be achieved at the instant of capture. For this simple case, a hundred percent bias in the RRT algorithm was able to provide a solution in $81.84s$. Fig. 3 (a) depicts the initial and final configurations along with the path taken by the end-effectors. The displacement of the base as the robot achieves the final configuration can also be seen. The required initial and final angular values are specified in Table 1. The low angular velocities of the order $10^{-4}\ rad\ s^{-1}$ in Fig. 3(b) show that the motion is reactionless, demonstrating the efficacy of the proposed methodology. However, these angular velocities are larger than numerical roundoff errors since the sampling based method maintained a constant reactionless velocity over a fixed time period even though integration was done using a variable time step solver. Fig. 3 (a) also shows that there is no noticeable rotation of the base. Fig. 4 shows the velocities and accelerations of the joints. It is evident from Figs. 4 (b) and (d) that the acceleration values are within the imposed limit of 1 rad/s^2.

Table 1: Desired Initial and Final Joint Angles (rad) and Velocities (rad/s)

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Initial Position	2.3562	-1.5708	-1.5708	-2.3562	1.5708	1.5708
Final Position	0.7854	-0.7854	0.5236	-0.7854	1.0472	0.5236
Final Velocities	-0.0264	0.0085	0.0203	0.0181	-0.0064	-0.0152

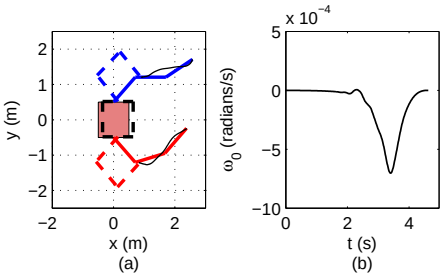

Fig. 3: System Configurations and Base Velocities: (a) Initial Configuration (dashed lines), Final Configuration (solid lines), end-effector paths (solid black line) (b) Base Angular Velocities

B. Path Planning with Obstacle Avoidance for Planar Robot

Here we present results when obstacles are present in the workspace. Reactionless maneuvering in itself reduces the allowable motion, and the further addition of obstacles makes the motion highly

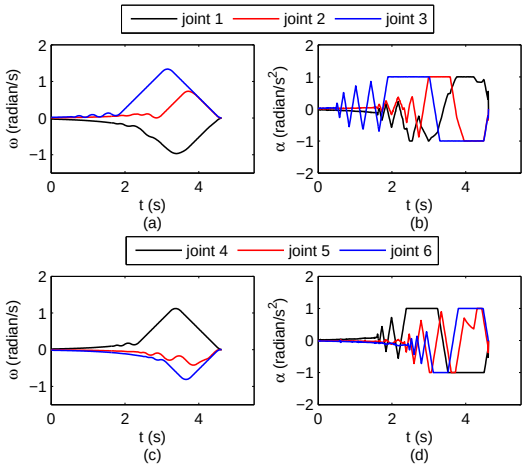

Fig. 4: Joint Motions: (a) Angular velocities of joints 1-3 (b) Angular accelerations of joints 1-3 (c) Angular velocities of joints 4-6 (d) Angular accelerations of joints 4-6

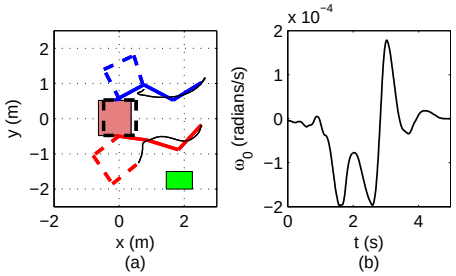

Fig. 5: System Configurations and Base Velocities: (a) Initial Configuration (dashed lines), Final Configuration (solid lines), end-effector paths (solid black line) and Obstacle (light green) (b) Base Angular Velocities

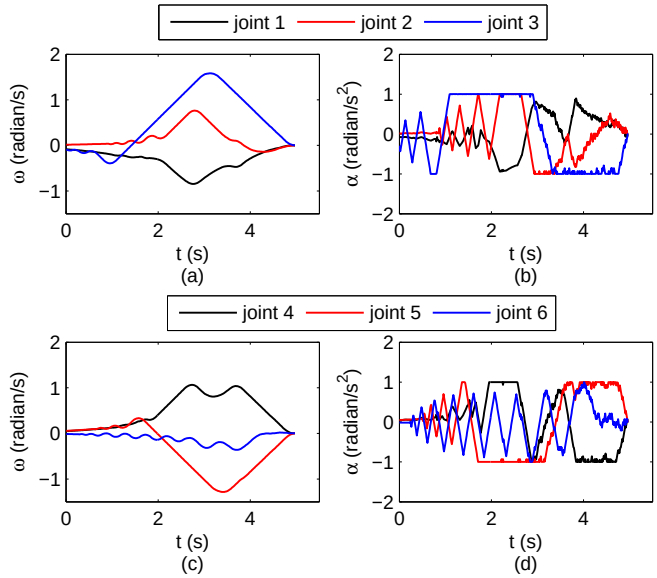

Fig. 6: Joint Motions: (a) Angular velocities of joints 1-3 (b) Angular Accelerations of joints 1-3 (c) Angular velocities of joints 4-6 (d) Angular Accelerations of joints 4-6

constrained. Sampling based methods are very effective for such cases as demonstrated by the results obtained. The time taken to compute this path was 391.46 s.

Fig. 5 (a) shows the motion when an obstacle is present. Fig. 5 (b) shows that the rate of rotation of the base is negligible, thus verifying that the robot executes a reactionless maneuver. Fig. 6 shows that the limits on accelerations are also satisfied while executing maneuvers to reach the target while avoiding the obstacle.

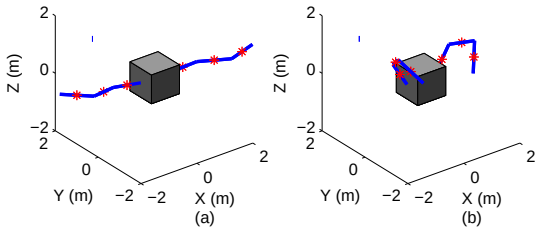

Fig. 7: Spatial Case: (a) Initial Configuration (b) Final Configuration

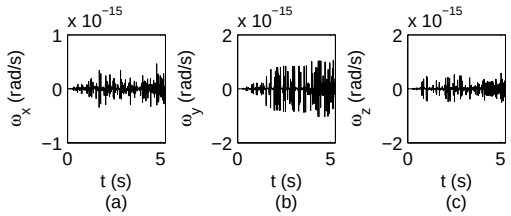

Fig. 8: Base Rotation Rates: (a) About X-axis(b) About Y-axis(c) About Z-axis

C. Path Planning for Spatial Robot

Next, the algorithm was tested for a spatial dual-arm robot with the same number of arms and links as in the planar case, but with the three links on an arm rotating about mutually perpendicular axes. In the spatial case, rotation about three axes needs to be controlled and therefore, the motion has a greater number of constraints than the planar case, thus reducing the degree of redundancy as well. In fact, using methods such as coordinate partitioning, it would not be possible to specify the end-effector positions for both arms for this system due to the number of dependent variables required.

Fig. 7 shows the initial and final configurations of the spatial robot for the tested case. The red stars mark the centre of mass of each link. Fig. 8 shows that the base reactions are negligible.

D. Time Scaling

For capture of a tumbling object, the path has to be traversed in a fixed time, which necessitates time scaling. It can be seen from the graphs of accelerations in Figs. 4 and 6 that the values are well below the limit towards the beginning. This indicates that the velocities could probably be scaled up at least in this region to meet time requirements.

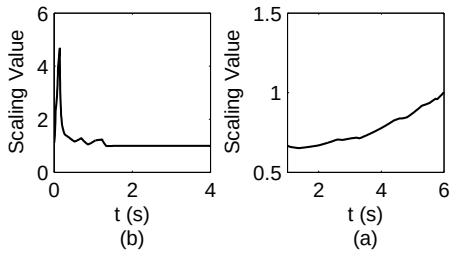

Fig. 9: Scaling Function: (a) Scaling up to traverse a path in 4 s (b) Scaling down to traverse the same path in 6 s

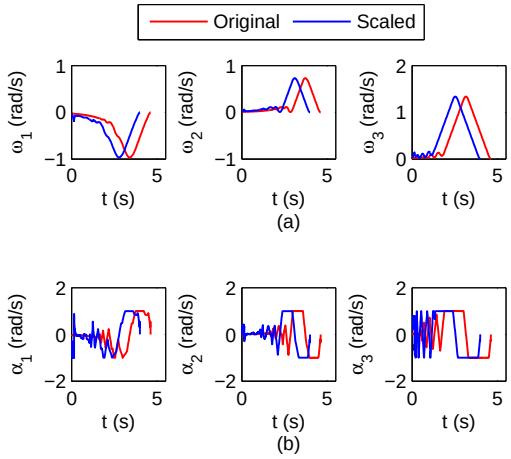

Fig. 10: Original and Scaled (a) angular velocities and (b) angular acceleratains of arm 1

Fig. 9 shows the piecewise scaling function used to scale the path shown in Fig. 3. In Fig. 3, the time taken to traverse the path was 4.62 s. Time scaling was used to traverse the same path in 4 s and 6 s as shown in Fig. 9. While we can postpone the execution of motion instead of scaling it down, if necessary, scaling can be useful in avoiding dynamic obstacles [20]. Fig. 10 corresponds to the scaling function in Fig. 9 (a). The scaled value and unscaled angular velocity and acceleration for the joints of the one of the arms of the robot are shown in Fig. 10, demonstrating the efficacy of the proposed methodology.

VI. Conclusions

The method proposed in this paper allows a space robot to autonomously reach a desired state at the desired instant while ensuring zero base rotation. Faster convergence in the RRT algorithm

is obtained through a simple strategy for biasing in reverse time simulations. The results obtained demonstrate how these objectives are achieved while satisfying additional constraints such as limits on acceleration and obstacle avoidance. The path planning framework is generic and can be applied to any system with nonholonomic constraints where the system is initially at rest and the final state can be found. The use of nonlinear time scaling to traverse a path in specified time was demonstrated and can be readily applied to alter the traversal time for any given path planning problem. It can also be used to check a given path for time optimality by simply checking if it can be scaled up with the imposed system constraints.

Acknowledgements

This work was supported by the DST fast-track research grant No: SB/FTP/ETA-0132/2013 of the second author. The authors would like to thank V. V. Anurag, MS (Research) student at IIIT Hyderabad, for setting up some of the functions for the spatial simulations.

References

[1] Andrew Long, Matthew Richards, and Daniel E Hastings. On-orbit servicing: a new value proposition for satellite design and operation. *Journal of Spacecraft and Rockets*, 44(4):964–976, 2007.

[2] Kazuya Yoshida. Engineering test satellite vii flight experiments for space robot dynamics and control: theories on laboratory test beds ten years ago, now in orbit. *The International Journal of Robotics Research*, 22(5):321–335, 2003. doi:10.1177/0278364903022005003.

[3] Farhad Aghili. Optimal control of a space manipulator for detumbling of a target satellite. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3019–3024, 2009. doi:10.1109/ROBOT.2009.5152235.

[4] Dragomir N Nenchev, Kazuya Yoshida, Prasert Vichitkulsawat, and Masaru Uchiyama. Reaction null-space control of flexible structure mounted manipulator systems. *IEEE Transactions on Robotics and Automation*, 15(6):1011–1023, 1999.

[5] Satoko Abiko and Gerd Hirzinger. On-line parameter adaptation for a momentum control in the post-grasping of a tumbling target with model uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 847–852, 2007. doi:10.1109/robot.2009.5152235.

[6] K. Yoshida, K. Hashizume, and S. Abiko. Zero reaction maneuver: flight validation with ETS-VII space robot and extension to kinematically redundant arm. In *Proceedings 2001 ICRA. IEEE International*

Conference on Robotics and Automation (Cat. No.01CH37164). Institute of Electrical & Electronics Engineers (IEEE). doi:10.1109/robot.2001.932590.

[7] Satoko Abiko and Kazuya Yoshida. Adaptive reaction control for space robotic applications with dynamic model uncertainty. *Advanced Robotics*, 24(8-9):1099–1126, 2010.

[8] Thai-Chau Nguyen-Huynh and Inna Sharf. Adaptive reactionless motion for space manipulator when capturing an unknown tumbling target. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4202–4207, 2011. doi:10.1109/ICRA.2011.5980398.

[9] Suril V Shah, Inna Sharf, and Arun K Misra. Reactionless path planning strategies for capture of tumbling objects in space using a dual-arm robotic system. In *AIAA Guidance, Navigation, and Control Conference, Boston*, 2013. doi:10.2514/6.2013-4521.

[10] C Menon, A Aboudan, S Cocuzza, A Bulgarelli, and F Angrilli. Free-flying robot tested on parabolic flights: kinematic control. *Journal of Guidance, Control, and Dynamics*, 28(4):623–630, 2005. doi:10.2514/1.8498.

[11] Dimitar Dimitrov and Kazuya Yoshida. Utilization of holonomic distribution control for reactionless path planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3387–3392, 2006. doi:10.1109/IROS.2006.282574.

[12] Giacomo Marani, Jinhyun Kim, Junku Yuh, and Wan Kyun Chung. Algorithmic singularities avoidance in task-priority based controller for redundant manipulators. In *IEEE/RJS International Conference on Intelligent Robots and Systems (IROS)*, volume 4, pages 3570–3574, 2003. doi:10.1109/IROS.2003.1249709.

[13] Miguel A Torres and Steven Dubowsky. Minimizing spacecraft attitude disturbances in space manipulator systems. *Journal of Guidance, Control, and Dynamics*, 15(4):1010–1017, 1992. doi:10.2514/3.20936.

[14] Evangelos Papadopoulos, Ioannis Tortopidis, and Kostas Nanos. Smooth planning for free-floating space robots using polynomials. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4272–4277, 2005.

[15] Thai Chau Nguyen-Huynh and Inna Sharf. Adaptive reactionless motion and parameter identification in postcapture of space debris. *Journal of Guidance, Control, and Dynamics*, 36(2):404–414, 2013. doi:10.2514/1.57856.

[16] Farhad Aghili. Time-optimal detumbling control of spacecraft. *Journal of Guidance, Control, and Dynamics*, 32(5):1671–1675, 2009. doi:10.2514/1.43189.

[17] Silvio Cocuzza, Isacco Pretto, and Stefano Debei. Least-squares-based reaction control of space manip-

ulators. *Journal of Guidance, Control, and Dynamics*, 35(3):976–986, 2012. doi:10.2514/1.45874.

[18] Steven M LaValle. Rapidly-exploring random trees a new tool for path planning. 1998.

[19] Steven M LaValle. Planning algorithms. Cambridge, U.K., 2006. Cambridge University Press. Available at http://planning.cs.uiuc.edu/.

[20] Balasubramanian Gopalakrishnan, Arun Kumar Singh, and K Madhava Krishna. Time scaled collision cone based trajectory optimization approach for reactive planning in dynamic environments. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4169–4176. IEEE, 2014. doi:10.1109/iros.2014.6943150.

[21] Yoji Umetani and Kazuya Yoshida. Resolved motion rate control of space manipulators with generalized jacobian matrix. *IEEE Transactions on Robotics and Automation*, 5(3):303–314, 1989. doi:10.1109/70.34766.

[22] Alejandro Perez, Robert Platt, George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, pages 2537–2542, 2012. doi:10.1109/icra.2012.6225177.

[23] John M Hollerbach. Dynamic scaling of manipulator trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 106(1):102–106, 1984. doi:10.1115/1.3149652.

[24] Y Bouktir, M Haddad, and T Chettibi. Trajectory planning for a quadrotor helicopter. In *16th Mediterranean Conference on Control and Automation*, pages 1258–1263, 2008. doi:10.1109/MED.2008.4602025.

[25] Fumio Kanehiro, Wael Suleiman, Florent Lamiraux, Eiichi Yoshida, and J-P Laumond. Integrating dynamics into motion planning for humanoid robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 660–667, 2008. doi:10.1109/iros.2008.4650950.

View publication stats