2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Maintaining stable grasps during highly dynamic robot trajectories

Giandomenico Martucci[1,3], Joao Bimbo[2], Domenico Prattichizzo[1,3], Monica Malvezzi[1,3]

***Abstract*—One of the key advantages of robots is the high speeds at which they can operate. In industrial settings, increased velocities can lead to higher throughputs and improved efficiency. Some manipulation tasks might require the robot to perform highly dynamic operations such as shaking, or swinging while grasping an object. These fast movements may produce high accelerations and thus give rise to inertial forces that can cause a grasped object to slip. In this paper a method is proposed to determine the inertial forces that arise on a grasped object during a trajectory, find the instances at which the object might slip, and avoid these slippages by changing the trajectory, namely the orientation of the object. To exemplify the usage of this approach, two grasping tasks are realised: a prehensile and a non-prehensile grasp, and strategies to successfully perform these tasks without changing the overall duration of the trajectory are defined and evaluated.**

***Index Terms*—robot grasping, dynamics, contact modelling**

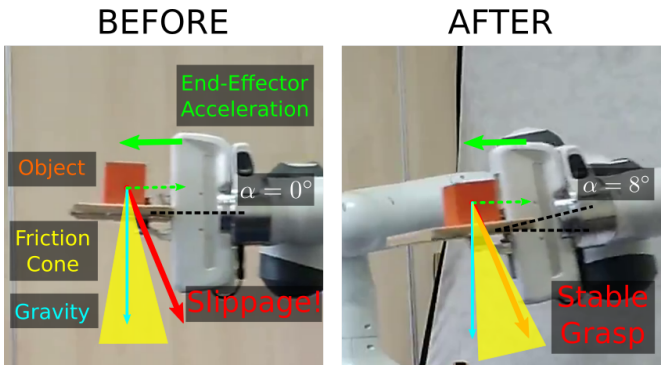


Fig. 1: As the robot accelerates, inertial forces arise that drive the resultant force (in red) to exit the friction cone (in yellow), causing the object to slip. By slightly modifying the trajectory, object slippage can be prevented.

I. Introduction

A. Motivation

The ability to grasp and manipulate objects is one of the indispensable skills required by modern robots. As robot manipulators become more pervasive in our lives, it is also expected that they are able to carry out a wide range of different tasks of increasing complexity. This flexibility in operation is likely to require generic robot hands or grippers, instead of tools that are tailored for a single task. Other than their precision, a key advantage of using robots is their ability to achieve high speeds. In industrial settings, higher speeds significantly increase the throughput of a production line. High speeds, generally involve high accelerations which introduce dynamical effects that cannot be neglected. Furthermore, complex tasks, such as shaking a bottle or flipping a pancake, are inherently dynamic, generating forces that may cause the object to slip away from the grasp.

In this paper, a framework that accounts for the inertial forces that arise on a grasped object during highly dynamic robot tasks, like the one in Fig. 1, is introduced. The objective is to evaluate if a trajectory can be completed or if slippage will occur, and modify the robot trajectory to prevent slippage while attempting to maintain the overall duration of the task.

[1]ADVR, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genoa, Italy. giandomenicomartucci@gmail.com
[2]Department of Mechanical Engineering and Materials Science, Yale University, 9 Hillhouse Avenue, 06511 New Haven, USA {joao.bimbo}@yale.edu
[3]Università degli Studi di Siena, Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Via Roma 56, 53100 Siena, Italy. {prattichizzo,malvezzi}@dii.unisi.it

B. Related Work

Robot grasping and manipulation has been one of the most fertile topics within the robotics community [1]. Most of the earlier work focused on hardware design, contact mechanics, and kinematic analysis [2]. Grasp planning and analysis have been, for the most part, based on quasi-static equilibrium equations to assess grasp stability [3]–[5]. The slippage of a grasped object can be predicted and prevented online, through the accurate measurement of tangential forces and using a dynamic friction model [6] or machine learning methods [7]. Woodruff *et al.* presented a framework to sequence and control a hybrid system containing a number of dynamic manipulation primitives for non-prehensile tasks. A recent review by Ruggiero *et al.* summarises the latest developments in dynamic manipulation for non-prehensile tasks [8].

When robot trajectory is generated and optimized, considering the dynamics is always paramount. Most trajectory generation and optimization algorithms aim at reducing the time to complete a task, the joint effort, or dynamical effects such as *jerk* [9], [10]. To the best of our knowledge, the problem of considering the slippage of a grasped object during the execution of a trajectory has not been previously featured in the literature in an explicit and structured way. Wang *et al.* [11] combine trajectory planning and grasp synthesis together, to find a collision-free trajectory to a good grasp. Mavrakis *et al.* [12] consider the object dynamics before grasping, in order to choose the grasp that will minimize the robot joint effort for the motions after the object is grasped.

978-1-7281-6211-9/20/$31.00 ©2020 IEEE

C. Structure

In the next section the preliminary notations and assumptions adopted in the rest of the paper are introduced, including a summary of spatial vectors, dynamics and grasp evaluation. The proposed approach for trajectory optimization is detailed in Section III. It follows a (chrono)logical order to explain each of the components of the framework. Two example tasks are presented in Section IV, and the capabilities of the proposed method to find a trajectory that successfully completes the task are evaluated. The final chapter discusses the results and anticipates some future developments of our method.

II. Theoretical framework

A. Preliminaries

The algebra of spatial vectors is used in this paper as mathematical framework, as it presents significant advantages for describing and calculating rigid-body dynamics. A guide to spatial vectors by Roy Featherstone [13]–[15] provides a detailed overview of this notation, and will give the unfamiliar reader a better understanding of this paper. Nevertheless, this section summarizes the main elements of spatial vectors employed in this paper.

Let us assume a stiff object that can be treated as a rigid body, denoted with symbol B. Let W be a fixed Cartesian frame with its coordinate system $Oxyz$ defined by the Euclidean basis $\mathcal{C} = \{\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}\}$; assume B is a local frame with the origin placed in the center of mass of the body. Thereafter, we will use B to refer to either the body or the frame, as convenient. To express the body-fixed frame orientation, the quaternion notation is adopted in this paper, since it is compact, allows easy computations and avoids some of the issues related with Euler angles [16], [17]; for positions, Euclidean coordinates are used. Furthermore, for a given time instant $t \in \mathbb{R}^+$, the pose $\boldsymbol{p}(t)$ of B wrt the frame W is defined as

$$\boldsymbol{p}(t) = \begin{bmatrix} q_w(t) & q_x(t) & q_y(t) & q_z(t) & x(t) & y(t) & z(t) \end{bmatrix}^T \tag{1}$$

such that $\boldsymbol{q} = [q_w, q_x, q_y, q_z]^T$, $\boldsymbol{q} \in H$, where H denotes the set of unit quaternions, $q_w \in \mathbb{R}$ and $(q_x, q_y, q_z) \in \mathbb{R}^3$ constitute the real and imaginary part of quaternion, respectively [18]; $\boldsymbol{c_m} = (x, y, z)$ is a 3D Euclidean vector denoting the position of the origin of B.

Once the orientation and position of the body frame are set, the spatial inertia is then defined as $\boldsymbol{I}$

$$\boldsymbol{I} = \begin{bmatrix} \overline{\boldsymbol{I}}_{\boldsymbol{c_m}} + m\, \boldsymbol{c_m} \times \boldsymbol{c_m}\times^T & m\, \boldsymbol{c_m}\times \\ m\, \boldsymbol{c_m}\times^T & m\, \mathbf{1} \end{bmatrix} \tag{2}$$

where m, $\boldsymbol{c_m}$, $\overline{\boldsymbol{I}}_{\boldsymbol{c_m}}$ denote the mass, the vector locating the center of mass and the rotational inertia with respect to $\boldsymbol{c_m}$, respectively.

Let $\mathbf{M}^6$ and $\mathbf{F}^6$ be the motion and force vector space, respectively; let $\boldsymbol{P}$ be a body-fixed point coincident with the origin $\boldsymbol{O}$ at the current instant, the Plücker coordinates [19] for spatial velocity are denoted with symbols $\boldsymbol{v}$, and for spatial force with $\boldsymbol{f}$, defining them as in (3), (4) [1].

$$\boldsymbol{v} = \begin{bmatrix} \omega_x & \omega_y & \omega_z & v_{Ox} & v_{Oy} & v_{Oz} \end{bmatrix}^T \tag{3}$$

$$\boldsymbol{f} = \begin{bmatrix} n_{Ox} & n_{Oy} & n_{Oz} & f_x & f_y & f_z \end{bmatrix}^T \tag{4}$$

Throughout this paper, the couples (rotational part of the spatial force) are indicated with symbol $\boldsymbol{\tau}$. Defining the Plücker coordinate system for ${}^B\boldsymbol{v}$ and ${}^B\boldsymbol{f}$ vectors, the Plücker coordinate transform (5) and (6) from W to B are

$${}^B\boldsymbol{X}_W = \begin{bmatrix} \boldsymbol{R} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ -\boldsymbol{c}\times & \mathbf{1} \end{bmatrix} \tag{5}$$

$${}^B\boldsymbol{X}^*_W = \begin{bmatrix} \boldsymbol{R} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\boldsymbol{c}\times \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{6}$$

for motion and force vector respectively, such that

$$\begin{aligned} {}^B\boldsymbol{v} &= {}^B\boldsymbol{X}_W\, {}^W\boldsymbol{v} \\ {}^B\boldsymbol{f} &= {}^B\boldsymbol{X}^*_W\, {}^W\boldsymbol{f} \end{aligned}$$

where $\boldsymbol{R}$ is the rotation matrix and $\boldsymbol{c}\times$ is the skew-symmetric matrix of the vector $\boldsymbol{c}$, which locates the origin of frame B wrt the frame W[2].

The symbol $\boldsymbol{s}(t)$ represents the state of the system, composed of pose $\boldsymbol{p}$, spatial velocity $\boldsymbol{v}$ and spatial acceleration $\boldsymbol{a}$, and we define a trajectory $\{S\}$ as the set of states over time. As such:

$$\boldsymbol{s}(t) = \begin{bmatrix} \boldsymbol{p}(t) \\ \boldsymbol{v}(t) \\ \boldsymbol{a}(t) \end{bmatrix}$$

$$\{S\} = \begin{bmatrix} \boldsymbol{s}(1),\, \boldsymbol{s}(2), \ldots, \boldsymbol{s}(t_{stop}) \end{bmatrix}$$

B. Inverse Dynamics

When an object is accelerating or rotating, inertial forces appear to act on the object. This effect can be seen in Fig. 1, where the robot is holding an object as it accelerates to the left. In the non-inertial frame of the robot end-effector, a force seemingly appears to push the object to the right.

The evaluation of the interaction forces acting between the robot and the grasped object generated by inertial effects, for a given motion, is done through the calculation of the Inverse Dynamics. In spatial vector algebra this is obtained through the formula

$$\boldsymbol{f} = \boldsymbol{I}\boldsymbol{a} + \boldsymbol{v} \times^* \boldsymbol{I}\boldsymbol{v}, \tag{7}$$

where the spatial acceleration is obtained as

$$\boldsymbol{a} = \dot{\boldsymbol{v}} = \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \ddot{\boldsymbol{c}} - \boldsymbol{\omega} \times \dot{\boldsymbol{c}} \end{bmatrix},$$

and $\boldsymbol{v}\times^*$ is the spatial cross product for forces, which can be represented by the 6×6 matrix

$$\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_O \end{bmatrix} \times^* = \begin{bmatrix} \boldsymbol{\omega}\times & \boldsymbol{v}_O\times \\ \mathbf{0} & \boldsymbol{\omega}\times \end{bmatrix}.$$

[1]Note that in the literature, the vectors may be denoted using the symbols $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{f}}$, respectively.

[2]The vector $\boldsymbol{c}$ corresponds to the vector $\boldsymbol{c_m}$ if the origin of body frame B coincides with the body center of mass

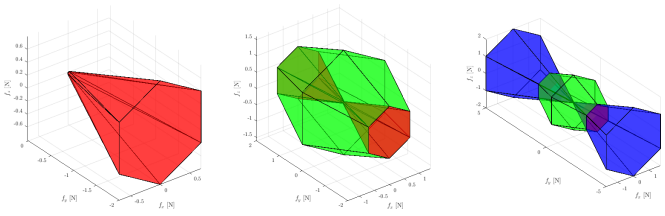


Fig. 2: Grasp Spatial Force Space. Left: Cone Spatial Force Space (CSFS) for one finger; Middle: Cones and resulting Grasp Spatial Force Space (GSFS) for a two finger grasp; Right: GSFS augmented with actuator limits

If the object B is being moved by a robot, the spatial velocity and acceleration can also be obtained directly from the joint angles of the robot $\boldsymbol{\theta}$, using the Jacobian $\boldsymbol{J}$ that maps joint velocities to the spatial velocity of body B as

$$\boldsymbol{v} = \boldsymbol{J}\dot{\boldsymbol{\theta}},$$

$$\boldsymbol{a} = \boldsymbol{J}\ddot{\boldsymbol{\theta}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{\theta}}.$$

C. Grasp Spatial Force Space

A commonly used tool in grasp planning, synthesis, and analysis is the Grasp Wrench Space [5], defined by Ferrari and Canny as a metric for grasp quality [3]. This space is defined as the forces that a grasp can resist given a set of contacts and a friction coefficient. Geometrically, this corresponds to a 6D polytope built as the *Convex Hull* [20] of the Minkowski sum of the friction cones for each contact.

In this paper, the algorithm is extended to be used with spatial vectors, defined similarly to [21], and indicated as *Grasp Spatial Force Space* (*GSFS*). The evaluation starts by calculating the cone spanning the spatial forces $\boldsymbol{f}_i$ that the i^{th} finger can generate (and therefore resist), when touching the body $\boldsymbol{B}$ at contact point $\boldsymbol{c_{p_i}}$. *This cone is defined as the Cone Spatial Force Space* (CSFS) In this paper, a soft-finger contact model is employed [2], [22] (that can generate moment about the contact normal), and the center of mass of body is assumed to be located in the origin of body local frame.

$$\begin{aligned} CSFS_i = \{\, \boldsymbol{f}_i | \boldsymbol{f}_i = \begin{pmatrix} \bar{\boldsymbol{\tau}}_{\boldsymbol{O}i} \\ \bar{\boldsymbol{f}}_i \end{pmatrix},\ \bar{\boldsymbol{\tau}}_{\boldsymbol{O}i} = \boldsymbol{c_{p_i}} \times \bar{\boldsymbol{f}}_i \ , \\ \left\| \bar{\boldsymbol{f}}_i - \bar{\boldsymbol{f}}_{\boldsymbol{n}i} \right\| \leq -\mu \left\| \bar{\boldsymbol{f}}_{\boldsymbol{n}i} \right\|,\ \left\| \bar{\boldsymbol{\tau}}_{\boldsymbol{n}i} \right\| \leq \mu_r \left\| \bar{\boldsymbol{f}}_{\boldsymbol{n}i} \right\| \,\} \end{aligned} \tag{8}$$

where $\bar{\boldsymbol{f}}_i$ are the linear coordinates of $\boldsymbol{f}_i$, $\bar{\boldsymbol{f}}_{\boldsymbol{n}i}$ and $\bar{\boldsymbol{\tau}}_{\boldsymbol{n}i}$ are the force and torque normal to the surface at contact point $\boldsymbol{c_{p_i}}$, whereas μ and μ_r are the linear and torsional friction coefficients. In practice, the cone is approximated to a pyramid to speed up calculations [23]. An example using 6 edges is depicted in Fig. 2 (left).

The *Grasp Spatial Force Space*, spanning the set of forces that a grasp can resist is then constructed using the convex hull of the Minkowski sum of all *CSFS*:

$$GSFS = \text{Conv}(\{\boldsymbol{f} | \boldsymbol{f} = \sum_{i=1}^{k} \boldsymbol{f}_i \ ,\ \boldsymbol{f}_i \in CSFS_i\}) \tag{9}$$

and an example is shown in Fig. 2 (middle). In order to represent this 6-dimensional polytope as a three-dimensional polyhedron, the *GSFS* was intersected with the polytope with torque $\bar{\boldsymbol{\tau}} = \boldsymbol{0}$. An important parameter to consider when building the *CSFS* is the height at which to truncate it. In grasp synthesis it is common to give an arbitrary value (e.g. a unit vector). Other approaches are possible, such as using the actuator limits [24]. For the application presented in this paper, the normal force that the finger is exerting as the height of the cone is used. In reality, this may not be entirely accurate, since one must consider also the forces that a finger can resist on its own. Hence, the union of the computed *GSFS* with another set of *CSFS*, constructed for each finger and bounded by actuation limits, shown in Fig. 2 (right), may be a better approximation of the forces that the grasp can resist. The experienced disturbance forces can be assumed to be smaller than the grasp force, in this case the polytope in Fig. 2 (middle) can be used in the calculations.

III. Methods

This section describes the procedure that enables the modification of a trajectory such that the inertial forces due to object dynamics can be resisted. While this approach is generalizable for a number of different problems, for the sake of brevity here a particular application where a serial robot is grasping an object and needs to move it quickly through a number of via-points while maintaining a stable grasp is presented. The developed procedure is based on the following assumptions: 1) a serial robot with a gripper as an end-effector is used; 2) the task of generating a smooth trajectory given a set of via-points is performed by an existing motion planner; 3) the original trajectory is maintained as closely as possible and only in the object orientation is changed; 4) the grasp contact points and forces are given as an input and do not change throughout the task.

The *Motion Planner* generates joint trajectories that track a desired set of Cartesian via-points $\boldsymbol{p}$. This plan takes into account robot kinematics and joint limits, and outputs a set of joint positions $\boldsymbol{\theta}$, velocities $\dot{\boldsymbol{\theta}}$, and accelerations $\ddot{\boldsymbol{\theta}}$. The joint trajectory is then transformed into its corresponding cartesian trajectory using *Forward Kinematics (FK)*.

The *Inverse Dynamics (ID)* module transforms these vectors into their spatial representation according to the equations in Section II-B, computing the inertial forces $\boldsymbol{f}$ that arise during the trajectory.

The *GSFS* polytope is calculated for the given grasp, as described in Section II-C.

The spatial forces $\boldsymbol{f}$ at every point of the trajectory are then tested by the *Force Evaluator* to check if they are contained inside the *GSFS* polytope [25]. When a force is not within the *GSFS*, we modify this point $\boldsymbol{p}(t = t_s)$ through optimization.

The *Optimizer* takes in the spatial force $\boldsymbol{f}(t_s)$ that is outside the *GSFS*, and the pose $\boldsymbol{p}(t_s)$. It then modifies the orientation $\boldsymbol{q}(t_s)$ to reorient the object such that the *GSFS* is able to resist $\boldsymbol{f}(t_s)$. For this purpose a local search is used, based on the Levenberg-Marquardt (LM) algorithm that tries to minimize the cost function in (10). This function takes into

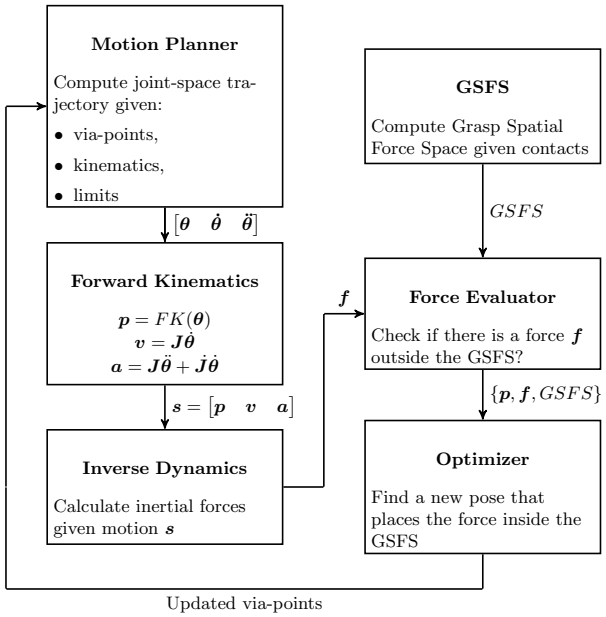


Fig. 3: Trajectory optimization procedure: The loop between planner and optimizer is iterated until all forces are within the Grasp Spatial Force Space

account the distance between the force and the surface of the *GSFS*. A penalty is added in order to keep the new rotation as close as possible to the previous point $\boldsymbol{f}(t_{s-1})$ in the trajectory. In order to reduce the number of calculations, the spatial vector $\boldsymbol{f}(t_s)$ is rotated instead of the *GSFS* polytope to find the inverse rotation. We chose the LM local solver to find solutions inside the *GSFS* which are as close as possible to the original orientation $\boldsymbol{q}(t_s)$.

$$\boldsymbol{q}'(t_s) = \underset{q}{\operatorname{argmin}} \exp(d(\boldsymbol{f}', GSFS) + k_1|\alpha|), \qquad (10)$$

where $\boldsymbol{f}' = \boldsymbol{q}'\boldsymbol{f}\boldsymbol{q}'^*$ is the the spatial force $\boldsymbol{f}$ rotated by quaternion $\boldsymbol{q}'$. The function $d(\boldsymbol{f}', GSFS)$ denotes the distance between $\boldsymbol{f}'$ and the *GSFS* polytope, with the condition that if the force is inside the polytope, the distance becomes negative. This calculation was done using the Multi-Parametric Toolbox 3.0 [26]. k_1 is a gain that regulates α, which is the angular distance between $\boldsymbol{q}'(t_s)$ and the previous point in the trajectory $\boldsymbol{q}(t_{s-1})$. This penalty is enforced to prevent large angular distances between subsequent points, which would generate high angular velocities and accelerations, and in turn increase inertial forces.

Once a trajectory point is optimized and the force is within the *GSFS* polytope, the via-points are updated and sent back to the motion planner to generate a new trajectory. As mentioned, this change in one trajectory point will modify the inertial forces both before and after that point, and so the whole process of planning, finding the Inverse Dynamics, and checking the forces needs to be repeated. In the case that a solution cannot be found, then several options are possible, such as increasing the grasp forces, or slowing down the trajectory in that region. A diagram summarizing this system is shown in Fig. 3.

IV. Results

To validate the proposed approach two tasks were defined, consisting of a grasp and a trajectory to be followed. In both trajectories, the inertial forces that arise due to the high accelerations rule out the successful completion of the task. The experiments were carried out using Franka Emika 7 DOF Panda manipulator robot[3], with a gripper. The robot was controlled using ROS[4] and its MoveIt planning framework. All other calculations and analyses are done in MATLAB R2019b[5]. The object in both tasks was a 3D-printed 5×5×5cm orange cube, weighing 37g. The spatial velocities are represented in the fixed global frame W, while the spatial forces are expressed in the local object frame B.

A. Task 1: Non-prehensile transport

The first task consisted of the robot gripping a flat wood surface on top of which a small cube was placed. The robot moved its end-effector down, then up, and then quickly accelerated laterally, while maintaining the surface in a flat orientation. This sequence of motions is shown in the upper row of Fig. 4. In the fourth frame of this sequence, the orange cube slides off from the surface due to the inertial forces caused by the sudden acceleration, which cannot be resisted by the friction between the wood and plastic surfaces.

Figures 5a and 5b show the motion and the forces during this task in dotted lines. Since the object does not change its orientation, the angular velocities are all zero, and thus also the moments. As for the forces, it can be observed that at $t = 1.05$s the normal force (z) is reduced to around -0.279N as a result of the vertical deceleration, while the tangential force (x) increases to -0.105N due to the lateral acceleration. This is the point at which the object starts sliding away from the surface, and slips off.

Applying the method presented in this paper, this slippage can be predicted and prevented. First, calculating the Inverse Dynamics and the GSFS with an estimated friction coefficient $\mu = 0.35$, the slippage points can be determined. Fig. 6a shows the spatial forces as a solid line, with the lighter areas corresponding to the regions where the grasp would remain stable if the other components were to remain the same. This area is obtained from the intersection of the GSFS with the line along the dimension of that component. At time $t = 1.05$s, the lines are no longer inside this safe region, since the magnitude of the tangential force $f_x = -0.105$N is above the maximum that the friction could resist ($|f_x| \leq 0.096$N). On the other hand, the blue region shows that this lateral force would be resisted if $f_z \leq -0.317$N. As for the spatial moments and f_y, no modification to these could make this grasp stable.

[3]https://www.franka.de
[4]https://www.ros.org
[5]https://www.mathworks.com/products/matlab.html

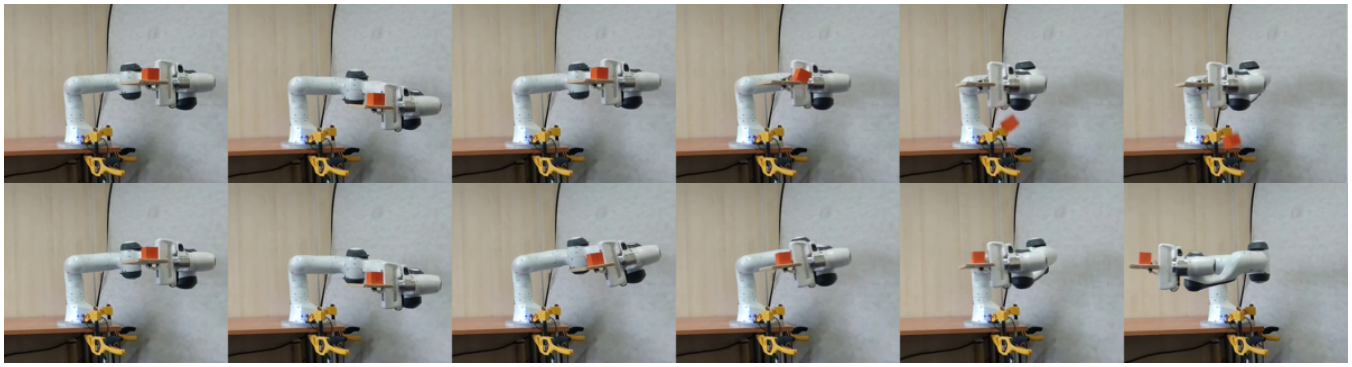
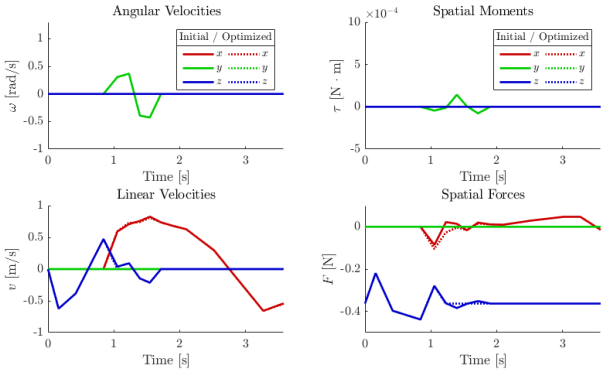
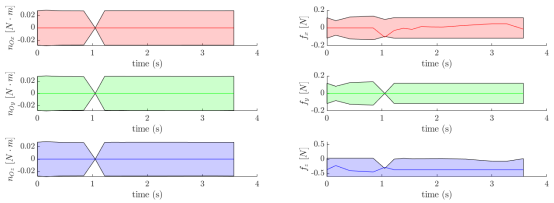


Fig. 4: Task 1 – The object is placed on a surface and quickly moved laterally. Top row: Initial trajectory; Bottom row: Optimized trajectory

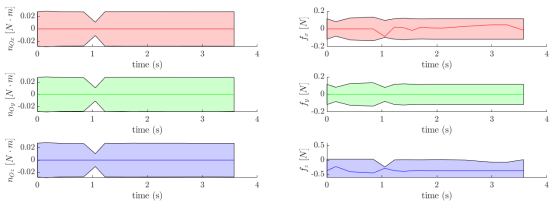


(a) Spatial velocities (b) Spatial forces

Fig. 5: Spatial velocities and forces for task 1



(a) Initial trajectory



(b) After optimization

Fig. 6: Inertial forces and safety regions for task 1

Feeding this slippage point $\{\boldsymbol{p}, \boldsymbol{f}\}(t = 1.05)$ and the GSFS into the optimizer, a new orientation can be obtained that keeps the force within the safe, non-slipping region. The result of this optimization is shown in the second row of Fig. 4. It can be seen that the trajectory was modified such that, around the third frame ($t = 0.85$s), the robot tilts the surface just enough to be able to resist this lateral force ($\alpha \approx 8°$). This new trajectory is again passed through the planner, *ID*, and *FK*, and the result is plotted as solid lines in Fig. 5a and 5b. While the linear part of the motion remains almost indistinguishable from the previous, there is an added angular velocity, which in turn generates some moment. As for the linear forces, they are very similar to the forces in the initial trajectory but, at the point where it previously slipped, the forces are now $f_x = -0.086$N and $f_z = -0.279$N. Checking these forces against the GSFS, the forces in this new trajectory are now all inside the stable region, as plotted in Fig.6b. In fact, as previously seen in the bottom row of Fig. 4, when the optimized trajectory is executed, the cube stays on top of the wooden surface.

B. Task 2: Highly dynamic grasp

In the second task the robot started by moving forwards and down ($\boldsymbol{v}_O = [0.33, 0, -0.42]^T$m/s) and then suddenly changed direction, swinging up ($\boldsymbol{v}_O = [0, 0, 0.42]^T$m/s). The robot gripper's fingers were covered in plastic tape to add compliance and reduce friction, allowing the very light object to slip. This prevented having to use a heavier object which might compromise safety. The sequence of pictures in the first row of Fig. 7 shows how the object slips away when this sudden change of direction happens (frame 4).

The procedure is similar to the one in the previous task, where the slippage point is detected, this time at $t = 0.9$s. Since the friction is reduced in this case ($\mu = 0.15$), the optimization required a large rotation in order to maintain the object within the GSFS, practically aligning the finger with the inertial force vector ($\alpha = 85°$).

Unlike the previous task, this trajectory presented an additional problem: after optimizing this point of high acceleration, as the robot returns up while rotating back, there is an additional force that is generated by this rotation, which again would cause the object to slip at $t = 1.44$s. For this reason the algorithm required two steps to optimize the whole trajectory. Figures 8a and 8b show the initial and final optimized trajectory. In Fig. 9 the forces and safe regions for all three trajectories are plotted. It can be noticed that the forces are slightly outside the stable region in 9b at $t = 1.44$s.

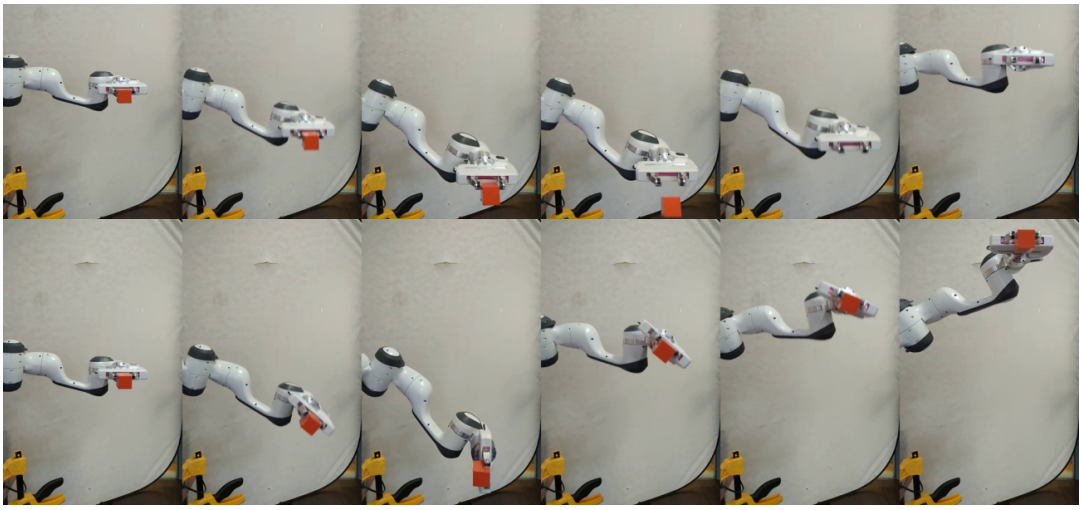
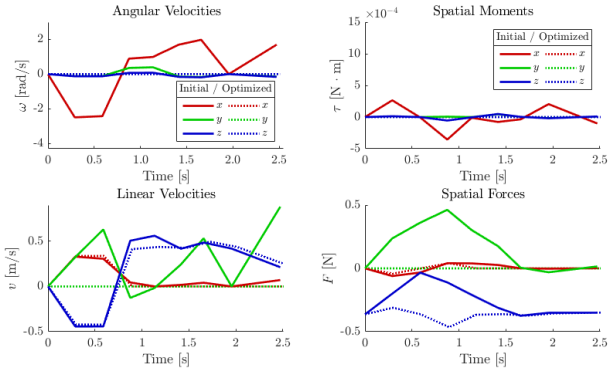
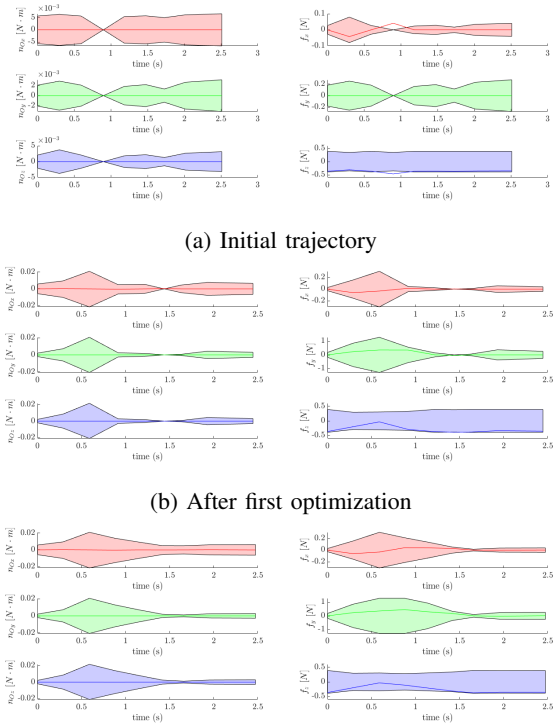


Fig. 7: Task 2 – A grasped object is moved down and quickly swung up. Top row: initial trajectory; Bottom row: optimized trajectory



(a) Spatial velocities (b) Spatial forces

Fig. 8: Spatial velocities and forces for task 2



(a) Initial trajectory

(b) After first optimization

(c) After final optimization

Fig. 9: Inertial forces and safety regions for task 2

The resulting optimized trajectory is shown in the bottom row of Fig. 7.

V. Conclusions

A. Discussion

In this paper a framework to allow a robot to move a grasped object at high velocities while maintaining the stability of the grasp is presented. Given a trajectory, this method calculates the inertial forces that the object experiences and checks them against the forces that the grasp can resist. If a force along that trajectory cannot be resisted, then an optimization algorithm modifies the pose of the object at that point such that this force can be counterbalanced by the grasp.

For the rigid body calculations spatial vector algebra was used, which enables compact and fast Inverse Dynamics computations. The same notation was used to formulate the forces that the grasp can resist, adapting the *Grasp Wrench Space* algorithm to construct a *Grasp Spatial Force Space* (GSFS), which is a 6D convex polytope that spans the spatial forces that the fingers can resist, given the fingers contact information (location, force, and friction properties).

Checking whether a force can be resisted by the grasp then becomes the geometric problem of finding whether the spatial force vector is contained inside the GSFS. If the force is predicted to be outside this polytope, an optimization algorithm modifies the orientation at that trajectory point such that the force stays inside. This change in the trajectory will, however, lead to a different set of forces, both before and

after that point, and so the process needs to be repeated until all the points are inside the GSFS.

To demonstrate the applicability of our method, two different grasping tasks were realised, one prehensile and one non-prehensile, where the initial trajectory would cause the object to slip away. Then, applying the proposed framework, the trajectory was modified and the robot successfully completed both tasks without changing the overall duration of the trajectory.

In the case where a solution cannot be found, the algorithm will still modify the trajectory to be closer to the safe, non-slipping orientation. Then either the grasping can be increased to cope with larger forces, or the overall speeds and accelerations of the trajectory can be reduced.

B. Future Work

While this framework is generic in terms of how the trajectories are generated and how they can be modified, in this paper only the use case of a serial manipulator was demonstrated and only the orientation along the trajectory was modified. Several extensions of this work can be envisioned, for instance applying this method in other types of tasks such as aerial robots. Additionally, if one allows the trajectory to be modified also in its position, and not only in orientation, other features can be included, such as robot workspace, obstacles in the environment, etc.

In the future customized planning methods, allowing a more flexible control over how the trajectory is generated, will be developed. This can yield a more efficient re-planning of the trajectory when one point is modified, restricting also the number of trajectory points before and after the modified point for which the Inverse Dynamics needs to be re-computed and the force checking needs to be done.

Finally, the computation of the GSFS polytope will be improved. In this work, the GSFS was assumed not to change during the trajectory. This is not entirely accurate, since the changes in the contact between the fingers and the object throughout the trajectory, both in force and location, may deform this polytope. As a future improvement, the GSFS will be dynamically recalculated to cope with these changes.

REFERENCES

[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, April 2000, pp. 348–353 vol.1.

[2] J. K. Salisbury and B. Roth, "Kinematic and Force Analysis of Articulated Mechanical Hands," *Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 105, no. 1, pp. 35–41, 03 1983. [Online]. Available: https://doi.org/10.1115/1.3267342

[3] C. Ferrari and J. Canny, "Planning optimal grasps," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. 2290–2295.

[4] M. Pozzi, M. Malvezzi, and D. Prattichizzo, "On grasp quality measures: Grasp robustness and contact force distribution in underactuated and compliant robotic hands," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 329–336, Jan 2017.

[5] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3d object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326 – 336, 2012, autonomous Grasping.

[6] X. Song, H. Liu, J. Bimbo, K. Althoefer, and L. D. Seneviratne, "A novel dynamic slip prediction and compensation approach based on haptic surface exploration," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 4511–4516.

[7] F. Veiga, H. van Hoof, J. Peters, and T. Hermans, "Stabilizing novel objects by learning to predict tactile slip," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 5065–5072.

[8] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, July 2018.

[9] Y. Zhao, H. Lin, and M. Tomizuka, "Efficient trajectory optimization for robot motion planning," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Nov 2018, pp. 260–265.

[10] M. G. Ardakani, A. Robertsson, and R. Johansson, "Online minimum-jerk trajectory generation," 09 2015, pp. 1086–1091.

[11] L. Wang, Y. Xiang, and D. Fox, "Manipulation trajectory optimization with online grasp synthesis and selection," 2019.

[12] N. Mavrakis, R. Stolkin, L. Baronti, M. Kopicki, M. Castellani *et al.*, "Analysis of the inertia and dynamics of grasped objects, for choosing optimal grasps to enable torque-efficient post-grasp manipulations," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 171–178.

[13] R. Featherstone, "A beginner's guide to 6-d vectors (part 1)," *IEEE robotics & automation magazine*, vol. 17, no. 3, pp. 83–94, 2010.

[14] ——, "A beginner's guide to 6-d vectors (part 2)[tutorial]," *IEEE robotics & automation magazine*, vol. 17, no. 4, pp. 88–99, 2010.

[15] ——, *Rigid body dynamics algorithms*. Springer, 2014.

[16] J. Funda, R. H. Taylor, and R. P. Paul, "On homogeneous transforms, quaternions, and computational efficiency," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, pp. 382–388, June 1990.

[17] I. Aguilar and D. Sidobre, "On-line trajectory planning of robot manipulator's end-effector in cartesian space using quaternions," 2005.

[18] E. B. Dam, M. Koch, and M. Lillholm, "Quaternions, interpolation and animation," Tech. Rep., 1998.

[19] M. Joswig and T. Theobald, *Plücker Coordinates and Lines in Space*. Springer, 01 2013, pp. 193–207.

[20] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, vol. 22, no. 4, pp. 469–483, 1996.

[21] C. Borst, M. Fischer, and G. Hirzinger, "Grasp planning: How to choose a suitable task wrench space," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 1. IEEE, 2004, pp. 319–325.

[22] D. Prattichizzo and J. C. Trinkle, "Grasping," in *Springer handbook of robotics*. Springer, 2008, pp. 671–700.

[23] Li Han, J. C. Trinkle, and Z. X. Li, "Grasp analysis as linear matrix inequality problems," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 663–674, Dec 2000.

[24] R. Orsolino, M. Focchi, C. Mastalli, H. Dai, D. G. Caldwell, and C. Semini, "Application of wrench-based feasibility analysis to the online trajectory optimization of legged robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3363–3370, Oct 2018.

[25] J. A. De Loera, "Actually doing it: Polyhedral computation and its applications," *Manuscript in progress*, 2010.

[26] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, pp. 502–510, http://control.ee.ethz.ch/~mpt.