Robust Execution of Contact-Rich Motion Plans by Hybrid Force-Velocity Control

Yifan Hou and Matthew T. Mason *Fellow, IEEE*

***Abstract*— In hybrid force-velocity control, the robot can use velocity control in some directions to follow a trajectory, while performing force control in other directions to maintain contacts with the environment regardless of positional errors. We propose an algorithm to compute hybrid force-velocity control actions automatically. We quantify the robustness of a control action and make trade-offs between different requirements by formulating the control synthesis as optimization problems. Our method can efficiently compute the dimensions, directions and magnitudes of force and velocity controls. We demonstrate the effectiveness of our method in several contact-rich manipulation tasks.**

I. INTRODUCTION

In the materials handling industry where robots pick up random objects from bins, it's generally difficult to pick up the last few objects, because they are usually too close to the bin walls, leaving no collision-free grasp locations. It's even harder if a flat object is lying in the corner. However, in such cases a human would simply lift the object up with only one finger by pressing on a side of the object and pushing against the bin wall. This is one of the many examples where humans can solve manipulation problems that are difficult for robots with surprisingly concise solutions. The human finger can do more than the robot finger because the human naturally utilizes the contacts between the object and the environment to create solutions.

Manipulation under external contacts is common and useful in human life, yet our robots are still far less capable of doing it than they should be. In the robot motion planning community, most works are focused on generating collision-free motion trajectories. There are planning methods that are capable of computing complicated, contact-rich robot motions [15], [16], however, the translation from a planned motion to a successful experiment turns out to be difficult. High stiffness controls, such as velocity control, are prone to positional errors in the model. Low stiffness controls such as force control are vulnerable to all kinds of inevitable force disturbances and noise, such as un-modeled friction.

In this work, we attempt to close the gap between contact-rich motion planning and successful execution with hybrid force-velocity control. We try to combine the good points of both worlds: high stiffness controls are immune to small force disturbances, while force controls (even somewhat inaccurate force controls) can comply with holonomic constraints under modeling uncertainties.

*This work was supported under NSF Grant No. 1662682.

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. yifanh@cmu.edu, matt.mason@cs.cmu.edu

Solving hybrid force-velocity control is more difficult than solving for force or velocity alone, because we need to compute directions for each type of control. It is challenging to properly formulate the problem itself, and the solution space is much higher dimensional. This is why most of the previous works on hybrid force-velocity control only analyzed simple systems with the robot itself (may include a firmly grasped object) and a rigid environment, without any free objects and more degree-of-freedoms.

In this work, we provide a hybrid force-velocity control problem formulation that works for systems with more objects, along with an algorithm to efficiently solve it. We quantify what it means for a constraint to be satisfied "robustly", and automate the control synthesis by formulating it as two optimization problems on the velocity/force controlled actions. The optimization automatically makes trade-offs between robustness and feasibility. In particular, we show that the velocity control directions do not have to be orthogonal to the holonomic constraints, leaving space for more solutions. Being closer to orthogonal does have benefits; it is considered in the cost function.

The rest of the paper is organized as follows. In the next section we review the related works. In section III, we introduce our modeling and problem formulation for hybrid force-velocity control problems. In section IV, we describe our algorithm for solving the problem. In section V and VI, we provide a step by step analysis and experimental results for one example.

II. RELATED WORK

A. Hybrid Force-Velocity Control

The idea of using hybrid force-velocity control for manipulation under constraints can date back to 1980s. Mason [12] introduced a framework for identifying force and velocity controlled directions in a task frame given a task description. Raibert and Craig [17] completed the framework and demonstrated a working system. Yoshikawa [23] investigated hybrid force-velocity control in joint space under Cartesian space constraints, and proposed to use gradient of the constraints to find the normal of the constraint surface in the robot joint space. There are also works on modeling the whole constrained robot system using Lagrange dynamics, such as analyzing the system stability under hybrid force-velocity control [13], or performing Cartesian space tracking for both positions and forces [14]. Most of these works modeled only the robot and a rigid environment without any un-actuated degree-of-freedoms in the system. As an exception, Uchiyama and Dauchez performed hybrid force-velocity

control for a particular example: two manipulators contacting one object [20].

There are lots of works on how to implement hybrid force-velocity controls on manipulators. For example, stiffness control can be used for this purpose. Velocity control is essentially a high stiffness control; force control can be implemented by low stiffness control with force offset. Salisbury [18] described how to perform stiffness control on arbitrary Cartesian axes with a torque-controlled robot. Raibert and Craig [17] divided Cartesian space into force/velocity controlled parts, then controlled them with separated controllers. The impedance control [7] and operational space control [8] theory provided detailed analysis for regulating the force related behaviors of the end-effector for torque-controlled robots. Maples and Becker described how to use a robot with position controlled inner loop and a wrist-mounted force-torque sensor to do stiffness control on Cartesian axes [11]. Lopes and Almeida enhanced the impedance control performance of industrial manipulators by mounting a high frequency 6DOF wrist [9]. Whitney [22] and De Schutter [3] provided overviews and comparisons for a variety of force control methods.

B. Motion Planning through Contacts

Recently, a lot of works tried to solve manipulation under constraints without explicitly using force control. For holonomic constraints, De Schutter *et al*. proposed a constraint-based motion planning and state estimation framework [4]. Berenson *et al*. did motion planning on the reduced manifold of the constrained state space [1]. For non-holonomic constraints, the most popular example is pushing [10], [24], [5]. Chavan-Dafle *et al*. performed in-hand manipulation by pushing the object against external contacts [2]. In these works, the robots interacted with the objects in a way that force control was not necessary.

III. MODELING & PROBLEM FORMULATION

First of all, we introduce how we model a hybrid force-velocity control problem. We adopt quasi-static assumption throughout the work, *i.e*. inertia force and Coriolis force are negligible. Assume every object is rigid, including the robot. Note that the role of our work is to compute the exact force and velocity actions *after* motion planning, so we assume a motion trajectory is available such that the goal for our algorithm at any time step can be given as instantaneous velocities. We reuse several important concepts from [12] such as *natural constraints* and *artificial constraints*, but extend their meanings when necessary to better suit a more general problem formulation. Much to the second author's consternation.

A. Symbols

Consider a system of rigid bodies including the robot and at least one object. Denote $q \in \mathbb{R}^{n_q}$ as the configuration of the whole system. Denote $\tau \in \mathbb{R}^{n_q}$ as the corresponding force variables, *i.e*. if q denotes joint angles, τ denotes joint torques. Although the configuration space is enough to describe the system state, its time derivative may not make sense as a velocity, *e.g*. when q contains quaternions. We describe the system velocity in a different space, the selection of the variables is usually called the "generalized variables".

Denote $v = [v_u^T \ v_a^T]^T \in \mathbb{R}^n$ as the generalized velocity. We pick the variables of v in such an order that the first n_u entries $v_u \in \mathbb{R}^{n_u}$ denotes the dimension that are not controlled directly, such as the velocity of an object. The last n_a elements $v_a \in \mathbb{R}^{n_a}$ represent the robot degree-of-freedoms. $v \neq \dot{q}$ in general, but is related to $\dot{q}$ by a linear transformation: $\dot{q} = \Omega(q)v$, where $\Omega(q) \in \mathbb{R}^{n_q \times n}$. Denote $f = [f_u^T \ f_a^T]^T \in \mathbb{R}^n$ as the corresponding generalized force. The product of f and v is the work done by the robot. Note the uncontrolled part of f is always zero: $f_u = 0$. We will do most of the computations in the language of generalized variables.

B. Goal Description

We describe the goal for our control synthesis as an affine constraint on the generalized velocity:

$$Gv = b_G. \tag{1}$$

If a motion trajectory is available, the desired velocity can be obtained from its time derivatives. The goal (1) might be a desired generalized velocity itself, or it might only involve a part of it. For example, in regrasping problems, people only care about the in-hand pose of the object; the pose of the hand can be set free to allow for more space for other constraints.

C. Natural Constraints

The law of physics constrains the system variables in many ways. These constraints will never be violated, no matter what actions we take. We call them the *natural constraints*. Our definition of the natural constraints includes holonomic constraints and Newton's second law. The original definition in [12] did not contain the Newton's second law, because it is of no significance for fully actuated systems.

1) Holonomic Constraints: We consider holonomic constraints, which are bilateral constraints on q that are also independent of $\dot{q}$. Example of holonomic constraints are non-penetration constraints and sticking contact constraints. We describe them by

$$\Phi(q) = 0. \tag{2}$$

$\Phi(q) \in \mathbb{R}^{n_\Phi}$ can be computed from the problem description. Its time-derivative gives the constraint on instantaneous velocity:

$$J_\Phi(q)\dot{q} = J_\Phi(q)\Omega(q)v = 0. \tag{3}$$

If an action attempts to violate a holonomic constraint, *e.g*. pressing an object against a table, a reaction force will emerge to maintain the constraint. Denote $\lambda \in \mathbb{R}^{n_\Phi}$ as the reaction forces for $\Phi(q)$. Its contribution to the joint torque can be computed by the principle of virtual work [21]:

$$\tau_\lambda = J_\Phi^T(q)\lambda,$$

project τ_λ into the space of generalized force:

$$f_\lambda = \Omega^T(q) J_\Phi^T(q) \lambda \tag{4}$$

The positive directions for the reaction force is determined by how we define $\Phi(q)$: when both $\delta\Phi$ and reaction force λ are positive, they make positive work. Be careful when applying the rule to the contact force between two movable object, as the force would have a different direction for each body.

2) Newton's second law: For systems that are not fully-actuated, Newton's second law becomes necessary for computing forces in the system. Denote $F \in \mathbb{R}^n$ as the external force (gravity, magnetic force, *etc*.) in generalized force coordinates. For quasi-static system, the Newton's second law says all the forces imposed on the system must sum to zero:

$$\Omega^T(q) J_\Phi^T(q) \lambda + f + F = 0 \tag{5}$$

The three terms are reaction forces, control actions and external forces, respectively.

D. Velocity controlled actions and holonomic constraints

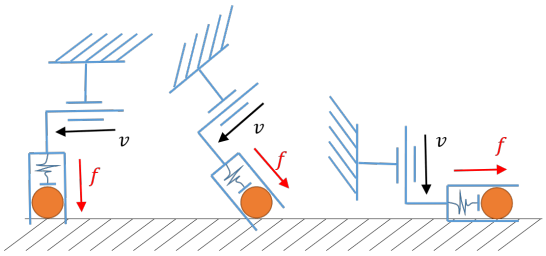

Fig. 1. Relation between velocity commands and holonomic natural constraints. The robot (blue) has a velocity controlled joint and a force controlled joint, which are orthogonal to each other. The table provides a natural constraint that stops the object from moving down. Assume no collision between the robot and the table. Systems in the left and middle are feasible. The right system is infeasible.

In some works of quasi-static analysis, the rows of Newton's second law for velocity-controlled dimensions are ignored because the force has no influence on other parts of the system, as shown in Fig. 1, left. We keep these rows in (5, because in general the axes of velocity commands may not lie completely in the null space of natural constraints, then the force generated from a velocity command could get involved in force computation in other parts of the system. One such system is illustrated in Fig. 1, middle.

An interesting question is, can we set velocity commands in any directions? One apparent fact is that our velocity command should never align with holonomic constraints (3 in the space of generalized velocity. Otherwise the hyperplanes they formed will be parallel and have no solution. Physically it means the velocity action is trying to pierce a wall, as illustrated in Fig. 1, right. Additionally, the velocity command is preferred to be close to the null space of holonomic constraints, in which case the two hyperplanes in the generalized velocity space will be less likely to get aligned under disturbances. If the system is holonomic, *i.e.* fully actualized, the velocity commands can be chosen from within the null space of holonomic constraints [23]. This is not always possible in general.

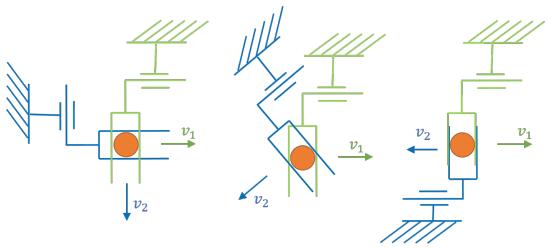

Fig. 2. Relation between different velocity commands. The blue robot and green robot are applying different velocity commands on the object. Assume no collision between the two robots. Systems in the left and middle figures are feasible. The right system is infeasible.

We can do the same analysis for different velocity commands. As shown in Fig. 2, different velocity controlled actions in generalized velocity space should not be co-linear. The system would be more robust to disturbances if the velocities are more perpendicular to each other.

E. Guard Conditions

A motion plan usually assumes a certain contact mode for a contact at any given time. In hybrid control theory, the term *guard conditions* refers to conditions for transitions between discrete modes. In our problem, we also need to apply guard conditions to make sure our robot action will not change the contact modes in the motion plan.

In this work, we consider guard conditions that can be expressed as linear or affine inequalities on force variables. Examples of this type are friction cone constraints and lower/upper bound on forces.

$$\Lambda \begin{bmatrix} \lambda \\ f \end{bmatrix} \leq b_\Lambda \tag{6}$$

F. Problem Formulation

The directions of force and velocity controlled actions form linear subspaces of generalized coordinates. To clearly describe the actions, we introduce *transformed generalized velocity* $w = [w_u^T \ w_{af}^T \ w_{av}^T]^T \in \mathbb{R}^n$, where $w_u = v_u$ is the un-actuated velocity, $w_{af} \in \mathbb{R}^{n_{af}}$ is the velocity in the force controlled directions, $w_{av} \in \mathbb{R}^{n_{av}}$ is the velocity controlled actions. Denote $\eta = [\eta_u^T \ \eta_f^T \ \eta_v^T]^T \in \mathbb{R}^n$ as the *transformed generalized force*, where $\eta_u = f_u = 0$ is the un-actuated force, $\eta_f \in \mathbb{R}^{n_{af}}$ is the force controlled actions, $\eta_v \in \mathbb{R}^{n_{av}}$ is the force in the velocity controlled directions. The robot action of the system (or *artificial constraints* [12]) is (w_v, η_f).

We use matrix T to describe the directions of force/velocity controlled axes: $w = Tv$, $\eta = Tf$. $T = diag(I_u, R_a) \in \mathbb{R}^{n \times n}$, where $I_u \in \mathbb{R}^{n_u \times n_u}$ is an identity matrix, $R_a \in \mathbb{R}^{n_a \times n_a}$ is an invertible matrix (not necessarily orthogonal).

Now we are ready to define the robust execution problem mathematically. At any time during the execution of a motion plan, the task of hybrid force-velocity control is to find out:

1) the dimensions of force controlled actions and velocity controlled actions, n_{af} and n_{av}, and
2) the directions to do force control and velocity control in the space of generalized variables, described by matrix T, and
3) the magnitude of force/velocity controlled actions: η_f and w_v,

such that:

- the goal (1) is satisfied as a result of velocity controlled actions and holonomic constraints (2);
- the guard conditions (6) are satisfied as a result of force controlled actions and the Newton's law (5).

Usually the problem described above has more than one solutions. As discussed in section III-D, we prefer velocity commands that are perpendicular to each other, and are close to the null space of holonomic natural constraints.

The formulation ensures that the two types of control are doing what they are good at, which explains the robustness of our method. The satisfaction of goals is ensured by velocity controlled actions, which are precise and immune of force uncertainties; the holonomic natural constraints are satisfied by selecting non-conflicting directions for velocity controlled actions, it won't be easy for a disturbance to make them conflict again. The guard conditions are basically maintaining contacts, which do not require the force controlled actions to be super precise.

IV. APPROACH

Now we introduce an algorithm to efficiently solve the problem defined in section III-F. The algorithm first solves for velocity commands, during with the dimensions and directions of both velocity control and force control are also determined. Then we fix the directions and solve for force controlled actions.

A. Solve for Velocity Controlled Actions

In this section, we design the velocity command (solve for n_{af}, n_{av}, T and w_v), so as to satisfy all the velocity-level conditions. We use a $n_{av} \times n$ selection matrix S_v to select the velocity commands out of the generalized variables: $w_v = S_v w$.Equations of interest to this section are: (Use $\dot{q} = \Omega v, v = Tw$, omitting argument q)

- Holonomic natural constraint $J_\Phi \Omega v = 0$. Denote $N = J_\Phi \Omega$, the constraint becomes $Nv = 0$;
- Goal condition $Gv = b_G$;
- Velocity command $S_v T v = w_v$. Denote $C = S_v T$, $b_C = w_v$, we can write the velocity command as $Cv = b_c$.

Denote the solution set of each equation as $Sol(N), Sol(G)$ and $Sol(C)$. We want to design the velocity command $C,\ b_c$ such that the resulted generalized velocity (the solution set of natural constraints and velocity commands) becomes a non-empty subset of the desired generalized velocity (the solution set of natural constraints and goal condition):

$$Sol(N\&C) \ \in \ Sol(N\&G) \tag{7}$$

1) Determine dimensions of velocity control: Denote $r_N = rank(N), r_{NG} = rank(\begin{bmatrix} N \\ G \end{bmatrix})$. The minimum number of independent velocity commands we must enforce is

$$n_{av}^{\min} = r_{NG} - r_N. \tag{8}$$

This condition makes sure the dimension of $Sol(N\&C)$ is smaller or equal to the dimension of $Sol(N\&G)$, so that their containing relationship is possible. The maximum number of independent velocity commands we can enforce is

$$n_{av}^{\max} = n - r_N = Dim(null(N)), \tag{9}$$

where $null(N)$ denotes the null space of N. This condition ensures the system will not be overly constrained to have no solution. We choose the minimal number of necessary velocity constraints:

$$n_{av} = n_{av}^{\min} = r_{NG} - r_N. \tag{10}$$

This choice makes it harder for the system to get stuck. As will be shown in the next section, it also leaves more space for solving force controlled actions.

2) Solve for directions and magnitude: With our choice of n_{av}, we know $rank([N;C]) = rank([N;G])$. Then the condition $Sol(N\&C) \ \in \ Sol(N\&G)$ implies

$$Sol(N\&C) \ = \ Sol(N\&G), \tag{11}$$

i.e. the two linear systems share the same solution space. This can be achieved by firstly choosing C such that the homogeneous linear systems $\begin{bmatrix} N \\ C \end{bmatrix} v = 0$ and $\begin{bmatrix} N \\ G \end{bmatrix} v = 0$ become equivalent (share the same solution space). Compute a basis for the solution of $[N^T G^T]^T v = 0$: $[\sigma_1, ..., \sigma_{n-r_{NG}}]$, then we just need to ensure C satisfies:

$$C\sigma_i = 0, \quad i = 1, ..., n - r_{NG} \tag{12}$$

Then we can compute b_C from any specific solution of $\{Nv = 0, Gv = b_G\}$. The original non-homogeneous systems then become equivalent.

Beside equation (12), we impose a few more requirements on C based on the discussions in section III-D:

- Rows of C must be linearly independent from each other. And we prefer to have them as orthogonal to each other as possible.
- Each row of C is also independent from rows of holonomic natural constraint N. And we prefer to pick the rows as close to $null(N)$ as possible.

To solve for C, denote $c^T \in \mathbb{R}^{1\times n}$ as any row in C. From $C = S_v T$ we know the first n_u columns in C are zeros, rewrite this and equation (12) as a linear constraint on c:

$$\begin{bmatrix} \sigma_1^T \\ \vdots \\ \sigma_{n-n_{NG}}^T \\ \begin{bmatrix} I_{n_u} & \mathbf{0}_{n_u \times n_a} \end{bmatrix} \end{bmatrix} c = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{13}$$

Its solution space has dimension of $n_c = n_a - n + r_{NG} = r_{NG} - n_u$. Since we need n_{av} independent constraints, we

require $n_c = r_{NG} - n_u \geq n_{av} = r_{NG} - r_N$, which gives $r_N \geq n_u$, *i.e.*

$$r_N + n_a \geq n. \tag{14}$$

The physical interpretation is, for our method to work on a system, it must be possible for the actions and constraints to fully constrain the system. Denote matrix $\mathbb{B}_c = [c^{(1)} \ \cdots \ c^{(n_c)}]$ as a basis of the solution space of equation (13). We can find a C that satisfies all the conditions by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{k}_1,\cdots,\mathbf{k}_{n_{av}}} & \sum_{i\neq j} ||c_i^T c_j|| - \sum_i ||null(N)^T c_i|| \\ s.t. \quad & c_i^T c_i = 1, \quad \forall i \\ & c_i = \mathbb{B}_c \mathbf{k}_i, \quad \forall i \end{aligned} \tag{15}$$

The solution $C = (\mathbb{B}_c \ [\mathbf{k_1} \ \ ... \ \ \mathbf{k_{n_{av}}}])^T$. The optimization problem (15) is non-convex and non-linear. However, notice that it is straightforward to sample from feasible solutions by:

1) Randomly sample $\mathbf{k} = [\mathbf{k_1} \ \ ... \ \ \mathbf{k_{n_{av}}}]$ from $\mathbb{R}^{n_c \times n_{av}}$;
2) compute $C = (\mathbb{B}_c \mathbf{k})^T$, and
3) normalize the rows of C to unit length.

So instead of solving the optimization problem exactly, we sample a bunch of Cs and pick the one C^* that gives the lowest cost. Don't care too much about optimality here: all the sampled solutions are feasible, the cost is only a matter of quality.

After we obtain C^*, we know the last n_{av} rows of R_a. Denote the last n_a columns of C^* as R_{C^*}, we can expand it into a full rank R_a:

$$R_a = \begin{bmatrix} null(R_{C^*})^T \\ R_{C^*} \end{bmatrix}, \tag{16}$$

it encodes the axes of the force controlled directions. Then we have $T = diag(I_u, R_a)$.

The algorithm is summarized in algorithm 1.

Algorithm 1 Solve for velocity controlled actions

1: Check equation (14). Declare infeasibility if check fails.
2: Compute n_{av} from equation (10).
3: Compute a basis of $[N^T G^T]^T v = 0$.
4: Compute a basis $\mathbb{B}_c$ of the solution of equation (13).
5: Sample N_s sets of coefficients $\mathbf{k} \in \mathbb{R}^{n_c \times n_{av}}$
6: **for** each sample $\mathbf{k}$ **do**
7: Compute matrix $C = (\mathbb{B}_c \mathbf{k})^T$, normalize its rows.
8: Compute the cost of C from equation (15).
9: **end for**
10: Pick the C^* with lowest cost.
11: Use equation (16) to compute R_a. Then $T = diag(I_u, R_a)$.
12: Compute one solution v^* for $Nv = 0, Gv = b_G$.
13: Compute $w_v = b_C = C^* v^*$.

B. Solve for Force Controlled Actions

Next we compute the force command (solve for η_f) so as to satisfy all the force-level requirements. Equations of interest to this section: (Use $\eta = Tf$, omitting argument q)

- Newton's second law: express (5) in the transformed generalized force space:

$$T\Omega^T(q) J_\Phi^T(q) \lambda + \eta + TF = 0. \tag{17}$$

- Guard conditions: express it as a constraint on λ, η:

$$A \begin{bmatrix} \lambda \\ f \end{bmatrix} = [A_\lambda \quad A_f T^{-1}] \begin{bmatrix} \lambda \\ \eta \end{bmatrix} \leq b_A. \tag{18}$$

The unknowns are the force variables λ, η. Remember $\eta = [\eta_u, \eta_f, \eta_v]$. Our choice of η_f and Newton's laws (17) will determine the value for all the forces; we need to make sure the resulted forces satisfy the guard conditions (18).

Remember also $f_u = 0$. Express it as $Hf = HT^{-1}\eta = 0$, and combine it with Newton's second law into one affine constraint:

$$\begin{bmatrix} 0 & HT^{-1} \\ T\Omega^T J_\Phi^T & I \end{bmatrix} \begin{bmatrix} \lambda \\ \eta \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -TF \end{bmatrix}. \tag{19}$$

Due to the limitation of rigid body modeling, the free forces $f_{free} = [\lambda^T \ \eta_u^T \ \eta_v^T]^T$ may not have a unique solution given a force action η_f. Rewrite the constraints (19) and move η_f to the right hand side, we find one solution for f_{free} by penalizing the sum-of-squares norm of the free forces:

$$\begin{aligned} \min_{f_{free}} \quad & f_{free}^T f_{free} \\ s.t. \quad & M_{free} f_{free} = \begin{bmatrix} \mathbf{0} \\ -TF \end{bmatrix} - M_{\eta_f} \eta_f. \end{aligned} \tag{20}$$

This is a quadratic programming (QP) problem. Denote f_{free}^* as the dual variables of f_{free}, the KKT condition says the solution to the QP can be found by solving the following linear system:

$$\begin{bmatrix} 2I & M_{free}^T \\ M_{free} & \mathbf{0} \end{bmatrix} \begin{bmatrix} f_{free} \\ f_{free}^* \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \begin{bmatrix} \mathbf{0} \\ -TF \end{bmatrix} - M_{\eta_f} \eta_f \end{bmatrix}. \tag{21}$$

This linear system determines the value for free forces given force action η_f. Rewrite it as

$$\begin{bmatrix} 2I & M_{free}^T & \mathbf{0} \\ M_{free} & \mathbf{0} & M_{\eta_f} \end{bmatrix} \begin{bmatrix} f_{free} \\ f_{free}^* \\ \eta_f \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ -TF \end{bmatrix}. \tag{22}$$

This linear equation encodes the unique solution for Newton's law. Finally we solve (22) together with guard conditions (18) to compute all forces. The procedure is summarized in algorithm 2.

Algorithm 2 Solve for force controlled actions

1: From Newton's laws, write down M_{free}, M_{η_f} in (20).
2: Write down coefficient matrices for equation (22).
3: Solve the linear programming problem (18)(22) for η_f.

V. Example

Next we illustrate how our method works with a concrete example. Consider the "block tilting" task shown in Fig. 3. The robot hand is a point. The robot needs to tilt and flip a square block about one of its edges by pressing on the block's top surface. We use a simple motion plan: the robot hand moves along an arc about the rotation axis, and all contacts in the system are sticking. If we only use velocity control to execute the plan, the robot can easily get stuck since the modeling or perception of the block may not be perfect.

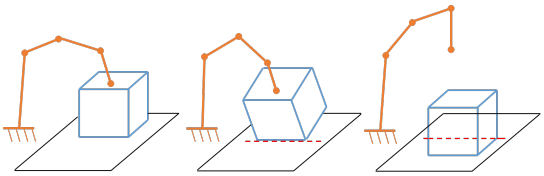

Fig. 3. Block tilting example. From left to right, the robot use one point contact to rotate the block.

A. Variables

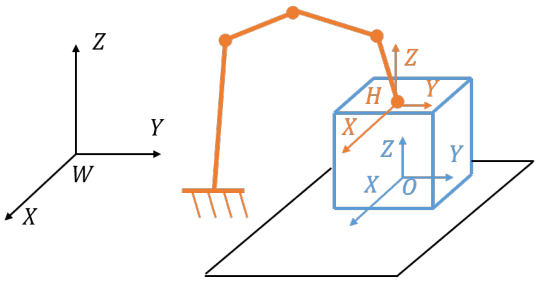

Fig. 4. illustration of the coordinate frames.

Denote W, H and O as the world frame, the hand frame and the object frame respectively. In the following, we use the form of ${}^A_B X$ to represent a symbol of frame B as viewed from frame A. We do Cartesian control for the robot, so we ignore the joints and only model the hand. The state of the system can be represented by the 3D pose of the object and the position of the hand as viewed in the world frame:

$$q = [{}^W_O p^T,\ {}^W_O q^T,\ {}^W_H p^T]^T \in \mathbb{R}^{10}. \tag{23}$$

Define the generalized velocity for the system to be the object body twist ${}^O_O \xi \in \mathbb{R}^6$ and the hand linear velocity ${}^W_H v \in \mathbb{R}^3$:

$$v = [{}^O_O \xi^T,\ {}^W_H v^T]^T \in \mathbb{R}^9. \tag{24}$$

The benefit of choosing body twist over spatial twist for representing generalized velocity of rigid body is that the expression of the mapping $\dot{q} = \Omega(q)v$ becomes simple:

$$\Omega(\mathbf{q}) = \begin{bmatrix} {}^W_O R & & \\ & E({}^W_O q) & \\ & & I_H \end{bmatrix} \in \mathbb{R}^{10\times 9}, \tag{25}$$

where ${}^W_O R \in SO(3)$ denotes the rotation matrix for ${}^W_O q$, $E({}^W_O q)$ is the linear mapping from the body angular velocity to the quaternion time derivatives [6]:

$$E({}^W_O q) = \frac{1}{2}\begin{bmatrix} -{}^W_O q_1 & -{}^W_O q_2 & -{}^W_O q_3 \\ {}^W_O q_0 & -{}^W_O q_3 & {}^W_O q_2 \\ {}^W_O q_3 & {}^W_O q_0 & -{}^W_O q_1 \\ -{}^W_O q_2 & {}^W_O q_1 & {}^W_O q_0 \end{bmatrix}. \tag{26}$$

The generalized force corresponding to our choice of generalized velocity is the object body wrench together with the hand pushing force:

$$f = [{}^O_O w^T,\ {}^W_H f^T]^T \in \mathbb{R}^9 \tag{27}$$

B. Goal Description

In the motion plan, the object rotates about the line of contact on the table. The goal for control at any time step is to let the object follow this motion. Now we try to write down the generalized velocity for such motion. Denote ${}^W p_{tc}$ as the location of any point on the line of contact, ${}^W \omega_g$ as the axis of rotation, $\dot{\theta}_g$ as the desired object rotation speed. We can firstly write down the spatial twist for the object motion as ${}^W \xi_g = (-{}^W \omega_g \times {}^W p_{tc},\ {}^W \omega_g)\dot{\theta}_g \in \mathbb{R}^6$. The corresponding body twist can be computed as

$${}^O \xi_g = Ad_{{}^W_O g^{-1}}\, {}^W \xi_g \tag{28}$$

where $Ad_{{}^W_O g^{-1}} = \begin{bmatrix} {}^W_O R^T & -{}^W_O R^T {}^W_O \hat{p} \\ 0 & {}^W_O R^T \end{bmatrix}$ is the adjoint transformation associated with ${}^W_O g^{-1} = \begin{bmatrix} {}^W_O R & {}^W_O p \\ 0 & 1 \end{bmatrix}^{-1}$. Then the goal for our controller can be specified as

$$G\mathbf{v} = b_G, \tag{29}$$

where $G = \begin{bmatrix} I_6 & 0_{6\times 3} \end{bmatrix}$, $b_G = {}^O \xi_g$.

C. Natural constraints

1) Holonomic constraints: The contact between the object and the hand is a sticking point contact, which constrains the system states by

$${}^W_O Q({}^O p_{hc}) + {}^W_O p = {}^W p_{hc}, \tag{30}$$

where ${}^W p_{hc}, {}^O p_{hc}$ denote the location of the contact point, function ${}^W_O Q(p)$ rotates vector p by quaternion ${}^W_O q$.

The contact between the object and the table is a sticking line contact. We approximate it with two point contacts at the two ends. Use subscript tc to denote the table contacts, the sticking constraints can be approximated by requiring the two points to be sticking:

$${}^W_O Q({}^O p_{tc,i}) + {}^W_O p = {}^W p_{tc,i}, \quad i = 1, 2. \tag{31}$$

Equation (30) and (31) together form the holonomic constraints for our system:

$$\Phi(q) = \begin{bmatrix} {}^W_O Q({}^O p_{hc}) + {}^W_O p = {}^W p_{hc} \\ {}^W_O Q({}^O p_{tc,1}) + {}^W_O p = {}^W p_{tc,1} \\ {}^W_O Q({}^O p_{tc,2}) + {}^W_O p = {}^W p_{tc,2} \end{bmatrix} = 0 \tag{32}$$

This example does not have face to face contacts; they can be handled similarly by multi-point-contacts approximation.

2) Newton's second law: The reaction forces $\lambda = [{}^W\lambda_{hc}^T, {}^W\lambda_{tc,1}^T, {}^W\lambda_{tc,2}^T]^T \in \mathbb{R}^9$ associated with the holonomic constraints (32) are the three contact forces as viewed in world frame. In Newton's second law (5):

$$\Omega^T(q)J_\Phi^T(q)\lambda + f + F = 0$$

Ω is known, $J_\Phi(q)$ is computed by symbolic derivation from $\Phi(q)$, we refrain from showing its exact expression to save pages. The external force F contains the gravity of the object G_O and the robot hand G_H, the reference frames of which should be consistent with the generalized force:

$$F = \begin{bmatrix} {}^OG_O \\ 0 \\ {}^HG_H \end{bmatrix} \in \mathbb{R}^9. \tag{33}$$

${}^HG_H$ should be zero if the robot force controller already compensates for self weight.

D. Guard Conditions

The motion plan requires all contacts to be sticking. Assume Coulomb friction model, we translate this requirement into two constraints on the force variables:

1) The normal forces at all contacts must be greater than a threshold n_{min}.
2) All contact forces must be within their friction cones.

To express 3D friction cone constraints linearly, we approximate the cone with eight-sided polyhedron [19] with $d_i = [\sin(\pi i/4),\ \cos(\pi i/4),\ 0]^T$ being the unit direction vectors for each ridge. Denote μ_{hc}, μ_{tc} as the estimated minimal possible friction coefficient, $z = [0\ 0\ 1]^T$ as the unit Z vector, the friction cone constraints becomes

$$\begin{aligned} &\mu_{hc} z^T ({}^O_W R\, {}^W\lambda_{hc}) \geq d_i^T ({}^O_W R\, {}^W\lambda_{hc}), \quad i = 1, ..., 8 \\ &\mu_{tc} z^T {}^W\lambda_{tc,1} \geq d_i^T {}^W\lambda_{tc,1}, \quad i = 1, ..., 8 \\ &\mu_{tc} z^T {}^W\lambda_{tc,2} \geq d_i^T {}^W\lambda_{tc,2}, \quad i = 1, ..., 8 \end{aligned} \tag{34}$$

The normal force lower bound can be written as

$$\begin{aligned} &z^T ({}^O_W R\, {}^W\lambda_{hc}) \geq n_{\min} \\ &z^T {}^W\lambda_{tc,1} \geq n_{\min} \\ &z^T {}^W\lambda_{tc,2} \geq n_{\min} \end{aligned} \tag{35}$$

Equation (34) and (35) are affine constraints on λ, together they form the guard condition (6).

E. Solve the problem

At each time step, given the object and the hand poses we can use algorithm 1 and 2 to solve for the hybrid force-velocity control numerically. You can find our Matlab implementation of the step by step derivations in our GitHub repository (see section VI-A).

Here we briefly describes the solved actions. The solution to the block tilting problem has one dimensional velocity controlled action, which points in the tilting direction and is roughly perpendicular to the line from the hand to the rotation axis. The other two dimensions are under force control. The Y component of the force command is close to zero, which makes sense as forces in Y direction don't do anything useful. The force in other component is roughly pressing against the rotation axis to maintain sticking.

VI. EXPERIMENTS

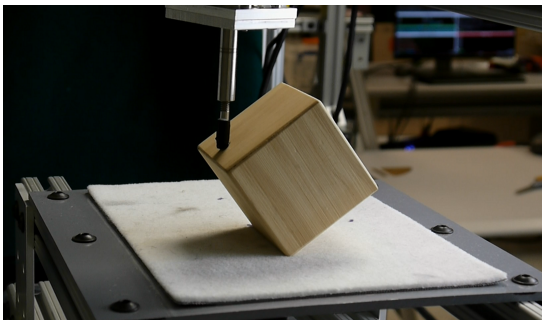

Fig. 5. Our experiment setup.

We implement the block tilting example. The object is a wooden block with edge length 75mm. The robot hand is a metal bar with tip covered by vinyl tapes to increase friction.

Low level control can be implemented in many ways. We implemented translational hybrid force-velocity control according to [11], and added functionality for choosing axes in any orientation. We use an ABB IRB 120 industrial robot with 250Hz position control loop, and a wrist-mounted force torque sensor, Mini-40 from ATI, to measure contact forces at 1000Hz. We place a 2mm-thick piece of cloth on the table to introduce some passive compliance, so that the control loop won't easily explode with impact forces from hard contacts. The cloth can be removed if we use a robot with passive compliance, like SEA joints or direct-drive joints.

We implemented our algorithm 1 and 2 in Matlab without any particular optimization for speed. With sample size $N_s = 500$, we can solve the block tilting problem in single thread on a 3.1GHz CPU in about 200ms. Currently we only recompute controls every half a second, which is surely something can be improved by better engineering; but even with such low update rate the block tilting is still successful.

We run block tilting on the same block 51 times. The robot successfully tilted the block 44 times. Six of the experiments stopped prematurely because the low level force control loop broke down. This problem can be fixed in the future with more stable force control implementations. There is only one time in which the robot failed to maintain the contact on the object, and the object slipped away unexpectedly.

A. Resources

The Matlab implementation of the two algorithms, along with the derivations for several examples can be obtained from our GitHub repository. You can also download our implementation of the low level hybrid force-velocity controller from another GitHub repository.

VII. DISCUSSION AND FUTURE WORK

For a hybrid force-velocity control problem, people might be able to manually design a control strategy that works just fine. We insist that our method is valuable, because we can automate the process for new problems without manual design. Moreover, we can solve some problems

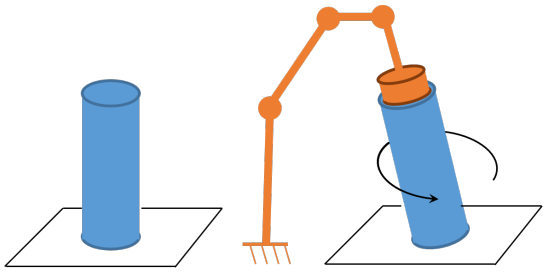

Fig. 6. Illustration of the bottle rotation problem.

that are unintuitive for a human. For example, consider the bottle rotation problem. Image a cylinder water bottle on a table, as shown in Fig. 6, left. Use a robot to press on its top surface with a face to face contact. If you apply force properly, you can tilt the bottle and rotate it on the table. The control strategy is not straightforward, since it involves hybrid actions in 6D wrench space. The Matlab code for solving this problem is also available in our GitHub repository. Unfortunately we don't have time to implement it on a robot.

Our algorithm has several limitations. Firstly, we haven't consider non-holonomic constraints in our current formulation. Secondly, since we use random sampling in algorithm 1, we are changing the velocity control directions at each time step with some randomness. This introduces oscillations and noise into the low level force controller, which may be a problem if we raise the control update rate. In the future we need to design a filter mechanism to avoid large changes in actions.

ACKNOWLEDGMENT

We would like to thank Hongkai Dai for helpful discussions on exponential coordinates and contact modeling.

REFERENCES

[1] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, and James J Kuffner. Manipulation planning on constraint manifolds. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 625–632. IEEE, 2009.

[2] N. Chavan-Dafle and A. Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6215–6222.

[3] Joris De Schutter, Herman Bruyninckx, Wen-Hong Zhu, and Mark W Spong. Force control: a bird's eye view. In *Control Problems in Robotics and Automation*, pages 1–17. Springer, 1998.

[4] Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré, Ruben Smits, Erwin Aertbeliën, Kasper Claes, and Herman Bruyninckx. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research*, 26(5):433–455, 2007.

[5] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. *Robotics: Science and systems VII*, 1, 2011.

[6] Basile Graf. Quaternions and dynamics. *arXiv preprint arXiv:0811.2889*, 2008.

[7] Neville Hogan. Impedance control: An approach to manipulation: Part iiimplementation. *Journal of dynamic systems, measurement, and control*, 107(1):8–16, 1985.

[8] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[9] António Lopes and Fernando Almeida. A force–impedance controlled industrial robot using an active robotic auxiliary device. *Robotics and Computer-Integrated Manufacturing*, 24(3):299–309, 2008.

[10] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

[11] J. Maples and J. Becker. Experiments in force control of robotic manipulators. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 695–702, Apr 1986.

[12] Matthew T Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):418–432, 1981.

[13] N Harris McClamroch and Danwei Wang. Feedback stabilization and tracking of constrained robots. *IEEE Transactions on Automatic Control*, 33(5):419–426, 1988.

[14] James K Mills and Andrew A Goldenberg. Force and position control of manipulators during constrained motion tasks. *IEEE Transactions on Robotics and Automation*, 5(1):30–46, 1989.

[15] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):43, 2012.

[16] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[17] Marc H Raibert and John J Craig. Hybrid position/force control of manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 103(2):126–133, 1981.

[18] J Kenneth Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, volume 19, pages 95–100. IEEE, 1980.

[19] David E Stewart and Jeffrey C Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[20] Masaru Uchiyama and Pierre Dauchez. A symmetric hybrid position/force control scheme for the coordination of two robots. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 350–356. IEEE, 1988.

[21] Luigi Villani and Joris De Schutter. Force control. In *Springer handbook of robotics*, pages 161–185. Springer, 2008.

[22] Daniel E Whitney. Historical perspective and state of the art in robot force control. *The International Journal of Robotics Research*, 6(1):3–14, 1987.

[23] Tsuneo Yoshikawa. Dynamic hybrid position/force control of robot manipulators–description of hand constraints and calculation of joint driving force. *IEEE Journal on Robotics and Automation*, 3(5):386–392, 1987.

[24] Jiaji Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377, 2016.