A force-mediated controller for cooperative object manipulation with independent autonomous robots

Nicole E Carey[1], Justin Werfel[2]

[1] Autodesk Robotics Lab, Autodesk Inc. nic.carey@autodesk.com
[2] Harvard John A. Paulson School of Engineering and Applied Sciences, Harvard University jkwerfel@seas.harvard.edu

Abstract. We consider cooperative manipulation by multiple robots assisting a leader, when information about the manipulation task, environment, and team of helpers is unavailable, and without the use of explicit communication. The shared object being manipulated serves as a physical channel for coordination, with robots sensing forces associated with its movement. Robots minimize force conflicts, which are unavoidable under these restrictions, by inferring an intended context: decomposing the object's motion into a task space of allowed motion and a null space in which perturbations are rejected. The leader can signal a change in context by applying a sustained strong force in an intended new direction. We present a controller, prove its stability, and demonstrate its utility through experiments with (a) an in-lab force-sensitive robot assisting a human operator and (b) a multi-robot collective in simulation.

Keywords: robot cooperation, distributed robotics, manipulation, force-based control

1 Background

Cooperative (or collaborative) manipulation is the task where two or more agents, which may be any combination of robots and humans, must work together to manipulate an object too large or unwieldy for one agent to handle alone. Often one agent may act as a leader, seeking to move the object along a privately known trajectory through space, aided by an arbitrary number of independent helper agents who have no direct knowledge of the path or destination.

Challenges associated with this task can include the following: (1) The details of the environment (topography of the ground, locations of obstacles, etc.) may not be known by some or all agents, and may change dynamically. (2) The number and locations of the agents participating in the task may not be known. (3) The properties of the object (mass, geometry, etc.) may not be known. (4) There may be unexpected disruptions (e.g., the object or agents may bump into obstacles). (5) Communication may be problematic, e.g., unreliable, undesirable, or unscalable.

We are interested in the case of an unanticipated need to move an object, with no opportunity for advance preparation. That is, the properties of the environment, object, set of helpers, and desired manipulation trajectory are all unknown, and some of these elements may also be changeable and dynamic. Further, if the leader is a human, then communicating the details of a desired trajectory to the helper robots is unlikely to be quick, easy, or detailed and accurate. From a usability standpoint, the ideal way for the robots to give help would be for the human to try to move the object in the desired way directly, by exerting forces on it, and for the robots to supplement the leader's force to enable the manipulation, without requiring any explicit instruction. An approach that works under these restrictive conditions could also be used in a scenario where more information or communication is available.

2 Related work

Work on collective manipulation typically assumes that all robots share knowledge about the intended trajectory, inertial parameters of the payload, and/or number and locations of teammates, and that explicit robot-to-robot communication is available [1, 7, 15, 17, 19]. Some work has considered the problem of obtaining such information in partially restricted cases, e.g., estimating inertial parameters using carefully coordinated actions and knowledge about the team [15]. Some studies rely on specialized gripper hardware that constrains freedom of movement in specific ways [10]. A subcategory of cooperative manipulation is collective transport, where movement is limited to a plane [4, 9, 10]. Here we consider cases where only limited payload information (local mass and relative center of mass (CM) location) can be obtained, no other information or explicit communication is available, movement in any dimension may be desired, and off-the-shelf hardware can be used.

3 Challenges and assumptions

In general, a helper robot cannot disambiguate wrenches that are due to guidance from a leader, collisions, other agents, or the load itself (Fig. 1B). Thus to successfully follow guidance forces and reject all others, each robot must have a contextual framework through which it can observe, filter, and classify sensed forces, and apply an appropriately calculated control torque to smoothly accomplish the task goal.

In previous work [4], we presented an adaptive control framework for collective transport, a special case of the more general cooperative manipulation problem, in which an unknown object is carried by an arbitrary number of robots with movement strictly in a horizontal plane. That work followed an operational space control paradigm [12] in which the robots allow movement in the plane and prevent movement out of the plane. That is, the contextual framework that robots used to interpret forces was the a priori knowledge that only movement in the plane would be intended.

In this paper, we modify our previously presented controller to permit more general collaborative manipulation tasks, encompassing movement in any direction. The central principle robots use to determine context is to interpret

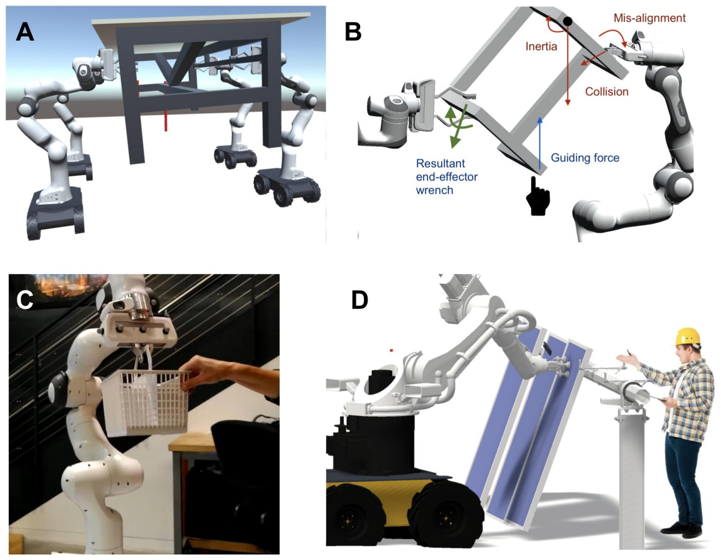


Fig. 1. A: A robot collective transports an unknown object following a leader's guidance. B: Two helper robots handle a small rigid object, guided by a human leader's force applied to one corner (blue arrow). The left-hand robot experiences a single multi-dimensional wrench at its end-effector, with no disambiguation of components resulting from the leader, the object's inertial properties, and forces due to the other agent. C: Physical testing of cooperative manipulation of a basket, using a Franka Emika Panda. D: An example application scenario: a robot helps a human manipulate a load in a challenging field situation of installing solar panels.

transient or weak forces as unintended (e.g., perturbations due to collisions), and sustained stronger forces as intentional control signals. We show that if each robot can estimate the payload's mass and relative CM location, then it can respond to control signals in any direction, switching between dimensional contexts without loss of controller stability or compliance. We demonstrate this framework in physical collaborative experiments using a compliant 7-DoF manipulator.

The key assumptions of the approach are: (1) Each agent can obtain an estimate of the load's relative CM location and its share of the mass (the latter may typically be m/n_a, for a system of n_a agents and total mass m, with both m and n_a unknown to the agents). (See §4.2 for a discussion of how this assumption can be realized.) (2) There are enough agents to handle the load (n_a is large enough that m/n_a is significantly less than the carrying capacity of any single agent). (3) The grasp or connection between each agent and the shared load is rigid, with no slippage or in-grasp rotation. (4) Agents have some compliance, permitting a degree of conflict between forces exerted by different agents without immediately causing instability. (5) Each agent knows the orientation of its base with respect to a shared reference vector (e.g., robots can sense the direction of

$\mathbf{d}$	Binary 6×1 dimensional constraint vector, $[x, y, z, \text{roll}, \text{pitch}, \text{yaw}]^T$.
D_b	Binary diagonal matrix s.t. $\text{diag}(D_b) = \mathbf{d}$.
Λ_x, Λ_ν	Inertial matrices associated with task and null spaces, respectively.
$\mathbf{p}$	The end-effector pose of the robot in cartesian space.
μ_x, μ_ν	Coriolis matrices associated with task and null spaces, respectively.
$\mathbf{x}$	An $n \times 1$ subvector of $\mathbf{p}$ consisting of the task-related pose dimensions.
J	the (non-square) standard manipulator Jacobian.
ν	An $m \times 1$ null-space velocity vector, where the total degrees of freedom of the manipulator can be written $n + m$.
J_E	The extended $(n + m) \times (n + m)$ manipulator Jacobian.
J_t	The task-space Jacobian.
η	A 6×1 vector describing the pose in non-task-related dimensions, with task dimensions (from $\mathbf{x}$) set to zero.
Z_ν	The null-space Jacobian.
$\mathbf{q}$	Joint positions.
$\mathbf{w}_t$	An $n \times 1$ vector of estimated external wrench along task-related dimensions.
K_e, K_ν, γ	Controller gain constants (stiffness).
B_n, D_ν, D_x	Controller gain constants (damping).
$\mathcal{T}$	Joint torque.
$\mathbf{f}$	Force (from force/torque sensors) expressed in the robot's base frame.

Table 1. Control variables and constants.

gravity while moving over uneven terrain). We do not assume other knowledge about the environment, task, agent numbers or locations, nor direct explicit communication.

We note that compliance can be helpful in multiple ways. For instance, it facilitates movement of the object in response to the guiding force, providing displacement as well as force cues that can be used by helper robots to infer the intended motion.

4 Control methodology

4.1 Closed loop control dynamics

The system dynamics for each agent take the standard form

$$\mathcal{T} = M\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathcal{T}_{ext} \tag{1}$$

where the load mass and inertia are modeled as connected rigid components of the end-effector, incorporated into the mass/Coriolis/gravitational matrices accordingly.

Each agent's (dynamic) context for the intended manipulation is encoded as a binary 6×1 vector $\mathbf{d}$, expressed in the robot's base frame. $d_i > 0$ indicates the agent should follow a guiding force along dimension i; $d_i = 0$ indicates a dimension where any external force is likely to be disturbance-related.

We can use $\mathbf{d}$ to decouple the instantaneous control torque applied in the robot's joint space into two components: a task controller, and a null or residual controller. Using $\mathbf{d}$ as a dimensional filter on the full manipulator Jacobian J

gives a task Jacobian J_t such that $\dot{\mathbf{x}} = J_t\dot{\mathbf{q}}$. The null space Jacobian $\mathcal{Z}_\nu$ can be derived as in [18]. This allows us to develop a task-specific control schema, and use the complementary space to enact a different form of control on the remaining null dimensions.

In a previous study [4], we considered a framework in which the task space was the set of dimensions in which external forces should be rejected, and the null space allowed free movement along other dimensions. Here we invert that division, so that instead of assigning a high priority task of stabilisation against unknown inertial dynamics (with the force-following component relegated to a lower priority null space), we assign the collaborative force-following goal to the high priority task space, while disturbance rejection of out-of-task forces is accomplished in the null space. This assignment allows the force-following behaviour of each agent (and hence the collective) to be more sensitive and responsive.

The task space controller takes the following form (a task passivity-based control):

$$\ddot{\mathbf{x}}_c = \Lambda_x^{-1}(\mathbf{w}_t - (\mu_x + D_x)\dot{\mathbf{x}} - \gamma K_e\tilde{\mathbf{x}}) \tag{2}$$

where $\mathbf{w}_t$ is an estimate of the guiding external wrench in the chosen task domain, expressed in the robot's base frame, and measured by the robot using either joint torque sensors or a wrist-based force/torque sensor; $(\mathbf{x}, \dot{\mathbf{x}})$ describes the trajectory of the end-effector in the task-space dimensions only (i.e., $\mathbf{x}$ is a subset of the full 6×1 end-effector pose vector $\mathbf{p}$); and Λ_x, μ_x are the task-space dynamic inertia and Coriolis matrices, respectively [20]. The error-correcting stiffness term K_e is explored in section 4.2.

In a cluttered dynamic environment, it is likely that disturbance forces will be due to environmental collisions. Rigidly maintaining null-space target positioning during a collision risks potential damage to the robot or the colliding object/person. However, the control space of the null component is defined by the residual degrees of freedom that remain unused by the task controller (i.e., it is thus both task- and robot-dependent). Noting this, we can take advantage of the greater number of degrees of freedom offered by redundant manipulators, and use a redundant joint space to accommodate collisions compliantly while minimally impacting the task space behavior. (If, instead, a high rigidity null space is desired, or the robot has insufficient DOFs to accommodate all possible task space configurations while rejecting error, a high stiffness impedance controller based on joint space error could be used.)

We define ν as a null space velocity vector [4] [20] and create an impedance control-based input torque of the following form:

$$\mathcal{T}_\nu = -\mathcal{Z}_\nu^{\#T}\left[\mathcal{Z}^T K_\nu J_E\tilde{\eta} - B_\nu(\Lambda_\nu\dot{\mathcal{Z}}^{\#}\dot{\mathbf{q}} + (\mu_\nu + D_\nu)\nu)\right] \tag{3}$$

where μ_ν, Λ_ν are the null Coriolis and inertial matrices [20], K_ν is a stiffness term acting on offsets in non-task-space end-effector dimensions, and B_ν, D_ν represent damping gains. The null position error state variable $\tilde{\eta}$ can be calculated using $\bar{D}_b$, the binary matrix derived from the negation of $\mathbf{d}$, assuming that for small changes in η and $\mathbf{q}$, we can use the expanded (full-rank) Jacobian J_E to transform errors in joint space to the (non-task-constrained) end-effector frame dimensions:

$$\delta\eta = \bar{D}_b J_E \delta\mathbf{q} \tag{4}$$

(In fact for $\mathbf{x} = \mathbf{0}$, where the goal joint configuration $\mathbf{q}_d$ is governed solely by null space control input, $\tilde{\eta}$ is the 6x1 end-effector pose projection of the joint error $\tilde{\mathbf{q}}$.)

The input torque corresponding to the task control components can be written

$$\mathcal{T}_t = -J_t^T \left(\Lambda_x(\dot{J}\dot{\mathbf{q}}) + (\mathbf{w}_t - (\mu_x + D)\dot{\mathbf{x}} - \gamma K_e \tilde{\mathbf{x}}) \right) \tag{5}$$

and our total input torque becomes

$$\mathcal{T} = \mathcal{T}_t + \mathcal{T}_\nu + C(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}). \tag{6}$$

4.2 Inertial estimation and error compensation

Here we discuss the need for each agent to have an estimate of local load mass (m_i) and CM location ($\mathbf{r}_i$), the relationship between mass inertia matrix error and controller responsiveness, and how these estimates can be achieved.

A robot can calculate an estimated mass inertia matrix M^i_{est} using $\mathbf{q}$, m_i, and $\mathbf{r_i}$. The true (unknown) mass inertia matrix of the (local) system is M^i. We define the error in this inertial estimation as follows:

$$\Delta M^i = M^i - M^i_{est}. \tag{7}$$

When a follower agent senses an external force, we can assume that this force is predominantly composed of a wrench imparted by the guiding agent, and any residual uncompensated inertial elements of the load:

$$F_{ext} = W_t + J^T \Delta M^i(\mathbf{q})\ddot{\mathbf{q}}. \tag{8}$$

Assuming a constant load, M^i, and hence ΔM^i, are bounded above [21]. In fact, given our assumption of enough agents to handle the load, $\|M^i\| \leq m_{max}$, where m_{max} is the maximum load each agent can support. (Ideally it would be significantly less.) Then an absolute lower bound for the external guiding wrench can be written

$$W_{t,min} = \mathbf{f}_{ext} - J^T(M_{max}(\mathbf{q}) - M^i_{est}(\mathbf{q}))\ddot{\mathbf{q}} \tag{9}$$

where $M_{max}(\mathbf{q})$ is the mass matrix for a load of mass m_{max} at joint configuration $\mathbf{q}$. Based on this minimum guiding wrench input, and the current system state, we can project a minimum expected end-effector pose estimate $\mathbf{x}_g$, given a discrete time-window Δt. The difference between the actual pose and this forward projection can be written as an error term in the task domain, $\tilde{\mathbf{x}} = \mathbf{x}_g - \mathbf{x}$. This allows us to introduce an error correction term into our control acceleration equation, using a spring stiffness gain matrix (K_e) (see Eqn. 2), and our challenge then becomes choosing a suitable K_e so that it fully compensates for model error.

From Eqn. 9, suppose $\mathbf{f}(M_{max}, \mathbf{q})$ is the inertial force applied by the maximum load the robot could support (calculated given the joint configuration $\mathbf{q}$). Then a possible choice of the elements k_e of K_e is $|k_e^j| \geq \max\|(\mathbf{f}(M_{L,max}, \mathbf{q}))^j\|$, i.e., we estimate the maximum possible force exerted by the inertial mass matrix upper bound M_{max} in each task dimension $\{j\}$, and choose K_e to guarantee sufficient compensation force. Note that given Eqn. 9, the ratio of {maximum force it is reasonable for the guiding agent to exert} :: {upper bound of load estimate} represents a metric for the system responsiveness.

K_e need not be constant: an adaptive stiffness term can be used without loss of stability [14] (provided $\dot{K}_e$ is bounded), as shown in [4], to create regions or dimensions of high and low responsiveness based on the expected impact of errors in the inertial estimate. Adaptive stiffness can also be increased temporarily in response to unexpected motions of the load, to prevent instability that could otherwise result if load inertia estimates are poor [4].

Other work has shown approaches for multiple robots to collaboratively estimate load parameters [15], [8]. In future work we plan to extend these approaches to the case without explicit communication. In the reductive case of a single guiding agent and a single follower (as in §5.1), an upper bound on ΔM^i can be found by examining the force residuals $\Delta\mathbf{f}(\mathbf{q})$ at the extremes of inertial offset, while only the follower agent is carrying the load (call this a stiffness tuning phase). Since in this case it is also relatively trivial to obtain a high degree of accuracy in inertial load estimation [16], in practice these residual force errors are minimal, and a small constant K_e suffices to fully compensate for model error.

4.3 Controller stability

In this section, we consider the stability of the full operational space controller, including the model-error correction component described above. Let our state variable be $\mathbf{z} = \{\tilde{\mathbf{q}}, \nu, \tilde{\mathcal{T}}, \tilde{\mathbf{f}}, \tilde{\mathbf{x}}\}$. . From [11], we can demonstrate asymptotic stability for the system if there exists a function $V(\mathbf{z})$ such that in a neighborhood Ω of an equilibrium point $\mathbf{z} = \mathbf{0}$, the following three conditions are satisfied:

1. $V(\mathbf{z}) \geq 0 \quad \forall \mathbf{z} \in \Omega$, and $V(0) = 0$;
2. $\dot{V}(\mathbf{z}) \leq 0 \quad \forall \mathbf{z} \in \Omega$;
3. the system is asymptotically stable for the largest positive invariant set $\mathcal{L}$ in $\{\mathbf{z} \in \Omega \quad | \quad V(\mathbf{z}) = 0\}$.

The total wrench acting on the system along the task dimensions can be written $\mathbf{f} = \mathbf{w}_t + \tilde{\mathbf{f}}$, where $\mathbf{w}_t$ is the estimated input guiding input wrench as before, and $\tilde{\mathbf{f}}$ is the total force estimation error, now including not only inaccuracies in the system model as above, but also other sources of error such as sensor noise. (Note that we are not considering the constrained null dimensions, so this error term is not confounded by disturbance rejection). At equilibrium, $\mathbf{f} = \tilde{\mathbf{f}}$, and we can introduce a disturbance observer to track error in the force domain. Using the task state error defined previously, we desire $\dot{\mathbf{x}}_g = 0$, so $\dot{\tilde{\mathbf{f}}} = -\Gamma_f^{-1}\dot{\mathbf{x}}$, where Γ_f is symmetric and positive definite.

Let $V(\mathbf{x}, \mathbf{f})$ be a candidate Lyapunov sub-function that considers only task-related energy terms:

$$V(\mathbf{x}, \mathbf{f}) = \tilde{\mathbf{x}}^T K_e \tilde{\mathbf{x}} + \frac{1}{2}\dot{\mathbf{x}}^T \Lambda_x \dot{\mathbf{x}} + \frac{1}{2}\tilde{\mathbf{f}}^T \Gamma_f \tilde{\mathbf{f}} \tag{10}$$

This function is positive definite along $(\mathbf{x}, \mathbf{f})$ and positive semi-definite on the full state $\mathbf{z}$ (condition 1 above). Differentiating, we find

$$\dot{V} = 2\tilde{\mathbf{x}}^T K_e \dot{\mathbf{x}} + \dot{\mathbf{x}}^T \Lambda_x \ddot{\mathbf{x}} + \frac{1}{2}\dot{\mathbf{x}}^T \dot{\Lambda}_x \dot{\mathbf{x}} + \tilde{\mathbf{f}}^T \Gamma_f \dot{\tilde{\mathbf{f}}} \tag{11}$$

Substitute the closed form dynamics from Eqn. 2 to obtain

$$\dot{V} = 2\tilde{\mathbf{x}}^T K_e \dot{\mathbf{x}} + \dot{\mathbf{x}}^T \Lambda_x \left[\Lambda_x^{-1}(\mathbf{w}_t - (\mu_x + D)\dot{\mathbf{x}} - \gamma K_e \tilde{\mathbf{x}})\right] + \frac{1}{2}\dot{\mathbf{x}}^T \dot{\Lambda}_x \dot{\mathbf{x}} + \tilde{\mathbf{f}}^T \Gamma_f \dot{\tilde{\mathbf{f}}} \tag{12}$$

We can substitute the disturbance observer defined in Eqn. 4.3 and obtain an estimate of the task dimension force error by examining the position error in combination with the environmental stiffness. Then, considering the skew-symmetry of $\dot{\Lambda}_x - 2\mu_x$ [20], we find condition 2 above on $\dot{V}$ is satisfied provided

$$2\tilde{\mathbf{x}}^T K_e \dot{\mathbf{x}} - \gamma \tilde{\mathbf{x}}^T K_e \dot{\mathbf{x}} - \dot{\mathbf{x}}^T D \dot{\mathbf{x}} < 0 \tag{13}$$

and since D is constant and positive definite, we require only that the controller gain $\gamma > 2$ to ensure conditional stability.

Furthermore, under the set condition $\mathbf{x} = \mathbf{0}$, our null state error term $Z^T K_n J \tilde{\eta}$ reduces to $Z^T K_n \tilde{\mathbf{q}}$, and the null space control torque is analogous to that used in [20]. By using a similar subset function candidate $V_{\mathcal{L}}$ on the set $\mathcal{L} = \{\tilde{\mathbf{q}}, \nu, \tilde{\mathcal{T}}, \tilde{\mathbf{f}} = \mathbf{0}, \tilde{\mathbf{x}} = \mathbf{0}\}$, it can be shown that the system is asymptotically stable conditional to $\mathcal{L}$ (condition 3 above). Hence all conditions for overall system stability are fulfilled.

4.4 Task-frame switching

The guiding agent can indicate a change in $\mathbf{d}$ (e.g., switching from lifting to horizontal carrying) to the other agents using the payload as the medium for a physical signal. For this paper, we note that in the absence of slow changes in uncompensated force driven by unknown load inertia (and with the assumption of a joint-compliant null-space control), disturbance forces are likely to be abrupt in onset and of limited duration. We can use the dynamic characteristics of such disturbances to define an admittance/rejection filter, establishing a set of time- and magnitude-based characteristics that signify a force imposed by the knowledgeable agent intended to change the dimensional domain of the task, and thus adjust the vector $\mathbf{d}$ on the fly. §6.2 discusses examples.

5 Experimental validation

The approach requires a force-sensitive manipulator with (ideally) > 6 degrees of joint freedom. The Franka Emika Panda cobot meets these needs (7 DOF) and is readily commercially available, making it an ideal platform with which to undertake proof of concept experiments. In §5.1 we discuss hardware testing/demonstration of the approach, using the single physical Panda available to us to assist a human leader. In §5.2 we discuss simulations with four Pandas assisting a leader. All code is available at [2].

5.1 Implementation in hardware

We implemented an initial parameter estimation phase without a human in the loop, using a two-stage recursive least squares method [13] to estimate approximate mass and CM location of the load, without requiring any foreknowledge of the load parameters or grasp point. Although the inclusion of off-diagonal inertias will in theory improve the stability region [14], we are interested in exploring the case where inertial estimation is imprecise or not all parameters can be determined. The estimation trajectory comprised a series of sinusoidal joint velocity commands (phase offset to each other to minimize inversion errors due to singularity points), operating symmetrically around the robot's default neutral pose. The joint velocity control during this inertial estimation phase was defined as: $\dot{\mathbf{q}}(t) = a_i \sin\left(2\pi \frac{t-t_L}{\omega_i}\right)$, with amplitudes $\mathbf{a} = [0.3265, -0.3265, 0.225, -0.3, -0.4435, 0.3, -0.3625]$, cycle period scaling factor $\omega = [3.68, 2.04, 2.98, 1.75, 4.43, 2.749, 1.4]$, t_L a phase offset constant.

The parametric convergence of the overall mass and CM location (relative to the end-effector grasp centroid) was usually accomplished in $< 5s$, with a mass estimation accuracy within $\pm5\%$ and centre of mass accuracy $\pm10\%$. We did not seek to ensure highly accurate parameter estimation, because we anticipate precise inertial information to be difficult to obtain in the multi-agent case, and did not want this to be a necessary pre-condition when assessing the success of this proof of concept.

When calculating the null-space Jacobian projection $\mathcal{Z}$, we must take particular care to ensure the devolved sub-Jacobian J_m [18] is well-conditioned, by using a robust inversion function; this necessarily increases the computation time required. The joint control torque is thus updated asynchronously, leading to non-zero errors in torque and projected position estimation; however, we find that the small correction stiffness K_e (described in §4.2) serves to accommodate these errors and ensure stability without impacting performance.

The Franka Panda is extremely sensitive to discontinuities in control inputs. Abrupt changes in torque input can result in a safety system lock, and aside from the need to low pass filter the raw torque sensor data due to a high noise level, there is also the potential for such discontinuities during task switching. Therefore, when a request for task dimension change has been registered, the real-time controller (a) sends a short burst of torque commands ($\sim$50ms) comprising mass and Coriolis compensating elements only, in order to bring the system smoothly to equilibrium, and (b) re-establishes a new null space origin at the current joint and end-effector positions. This ensures a relatively smooth transition between task contexts.

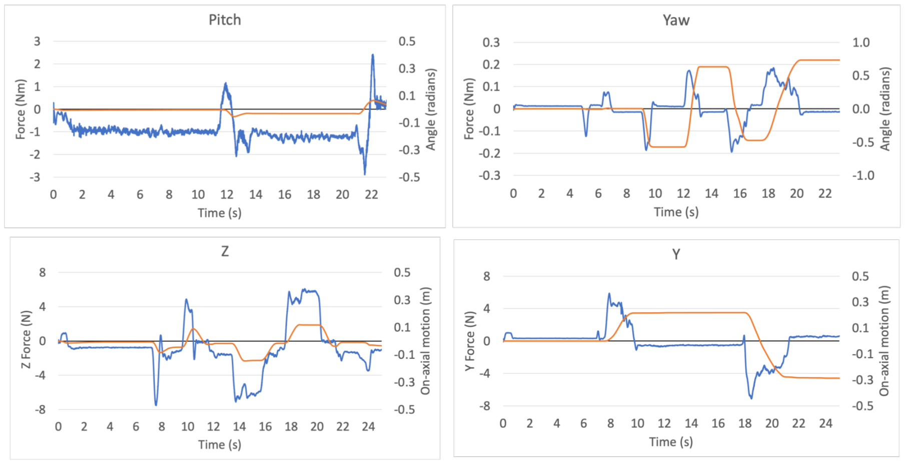


Fig. 2. Hardware implementation: behaviour in null-space (constrained; left) vs. task-space (unconstrained; right) dimensions. Top left: torque was applied around the pitch axis (relative to the robot's base). Bottom left: a force was applied in the Z (vertical) dimension. Top right: torque was applied around the yaw axis (relative to the robot's base). Bottom right: a force was applied in the Y (lateral) dimension. In all cases, blue indicates the applied force or torque, orange indicates the system response along or around the axis in question. Controller gains: $D_x = 5.0I_3, K_e = 0.2I_3, \gamma = 2.5, K_\nu = diag(240, 120, 120, 120, 120, 120, 120), B_\nu = 0.2, D_\nu = 4.0I_4$.

As discussed in section 4.4, we use a force input from the guiding agent to signal task dimension switches (changes in $\mathbf{d}$). For each test case discussed in the next section, we established a set $\mathbf{d}_i$ of pre-coded potential task domain vectors, and time and wrench magnitude thresholds $(t_{th}, w_{i,th})$ such that any external force or torque $> w_{i,th}$ applied for $> t_{th}$ along a non-task dimension would signal a switch in the task dimension vector. In our case, the number of pre-coded states $\mathbf{d}_i$ is small; however, with a sufficiently granular set of $\mathbf{d_i}$, we can attain as much dimensional control flexibility as desired.

5.2 Implementation in simulation

To demonstrate the efficacy of this method for multi-robot cooperation without direct communication, we used a simulated Franka Emika Panda model created in the Unity platform [3], and implemented the controller described above on four simulated robots manipulating a large shared load. A separate simulated leader applies forces and torques to the load. Each Panda is mounted on an omnidirectional wheeled base; it uses lateral motion of its end-effector in the task space as a control input to its base, thereby moving along with the leader's guidance in the plane, and extending its effective workspace.

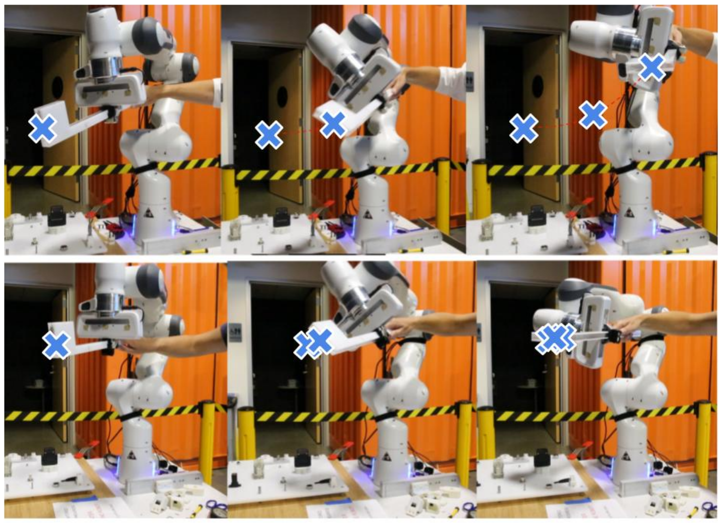


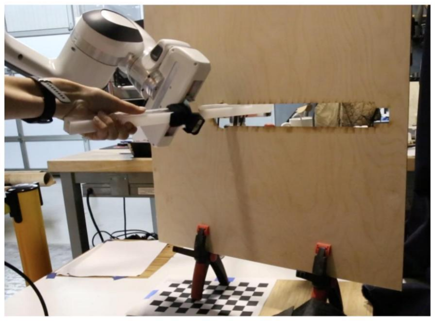
Fig. 3. Robot guidance without and with contextual information. The human guide applies a torque to rotate the end-effector through 90 degrees, around the roll axis. In the first sequence of images (top), the robot control torque compensates for gravitational load, but otherwise is compliant in all dimensions. In the second sequence (bottom), we implement our decoupled force/impedance algorithm with the task constraint vector $\mathbf{d} = \{0, 0, 0, 1, 1, 0\}$, rejecting wrenches in any direction other than roll and pitch. X's show the position of the end of the object across successive frames. Video: [5].

6 Results

6.1 Controller performance within static task domain

To investigate the performance of the two control elements (force following vs. impedance-control), we defined our task domain as $\mathbf{d} = \{1, 1, 0, 0, 0, 1\}$, applied a set of short disturbance forces to a load held by a single robot, and examined the resulting end-effector trajectory. A sample output is shown in Fig. 2, with motion and force expressed relative to the robot's base frame of reference. Disturbances in the pitch or vertical motion are rejected by the null controller, while those around the yaw axis or along the lateral Y axis result in motion for as long as the wrench is applied, and do not return to the origin position afterwards.

To demonstrate the behavioural difference between our chosen decoupled control schema and a more conventional compliant control that might be used for, e.g., a learning by demonstration context, we consider a task whose goal is a 90 degree rotation of a manipulated object around the base's roll axis. Fig. 3 compares the motion of the robot and object for each of the two control schemes in response to the same user input wrench. In the top set of images, the robot is under a free-floating (mass-compensated) control without contextual information about which wrench dimensions should be followed. Since the user input—as



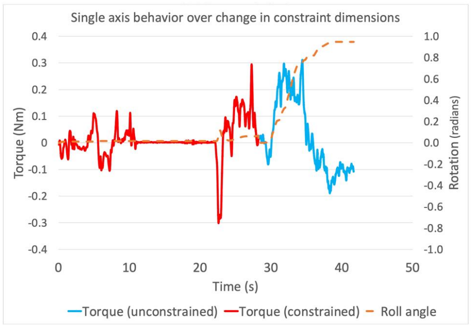


Fig. 4. Performing a collaborative task where force following and rejection dimensions change, under the guidance of a human operator. The human guides the robot to first insert a key into a slot, then turn it 90° and slide it along the slot. Left: Still image after insertion, during the turning phase. Right: The force and motion mapping for one dimension (roll). Force following vs force rejection phases are indicated through colour on the torque input (blue = force follow, red = force reject). Controller gains: $D_x = 5.0I_3, K_e = 0.2I_3, \gamma = 2.5, K_\nu = diag(240, 120, 120, 120, 120, 120, 120), B_\nu = 0.2, D_\nu = 4.0I_4$. Task-switch thresholding: $(t_{th} = 1.5s, w_{th} = (5N, 0.5Nm))$. Video: [5].

expressed through the shared load—has wrench components in multiple dimensions, the robot's end-effector position not only rotates, but also moves in the Y and Z dimensions. In the bottom image series, the robot is controlled by our decoupled controller and supplied with a dimensional context which restricts motion in Y and Z ($\mathbf{d} = \{0, 0, 0, 1, 1, 0\}$). This controller decouples the wrenches imparted by human manipulation of the load and responds only to those along or around the task dimensions, resulting in an end-effector pose that changes only along the roll axis, as desired. In other words, with this control framework and given a suitable context, we can shape the robot's resultant trajectory with much greater precision. See video at [5] (part 1).

6.2 Controller performance with a changing task domain

To demonstrate switching between contexts when the motion requirements of a task have a clearly defined dimensional change, we consider a collaborative key/lock positioning/insertion and rotation task. We pre-code a set of dimensional constraints $\mathbf{d}_i$ corresponding to the key insertion ($i = 0$), and key turn/slide ($i = 1$):

$$\mathbf{d}_i = \begin{cases} \{1, 1, 1, 0, 0, 1\} & i = 0 \\ \{0, 1, 0, 1, 1, 0\} & i = 1 \end{cases} \tag{14}$$

Fig. 4 shows a snapshot during the process, and the roll input torque and end-effector angle over the interaction. Initially, the robot is free to move in (x, y, z, yaw), but in order to maintain the key in the correct orientation for insertion, constrained in (roll, pitch). Once the key has been inserted into the slot, the

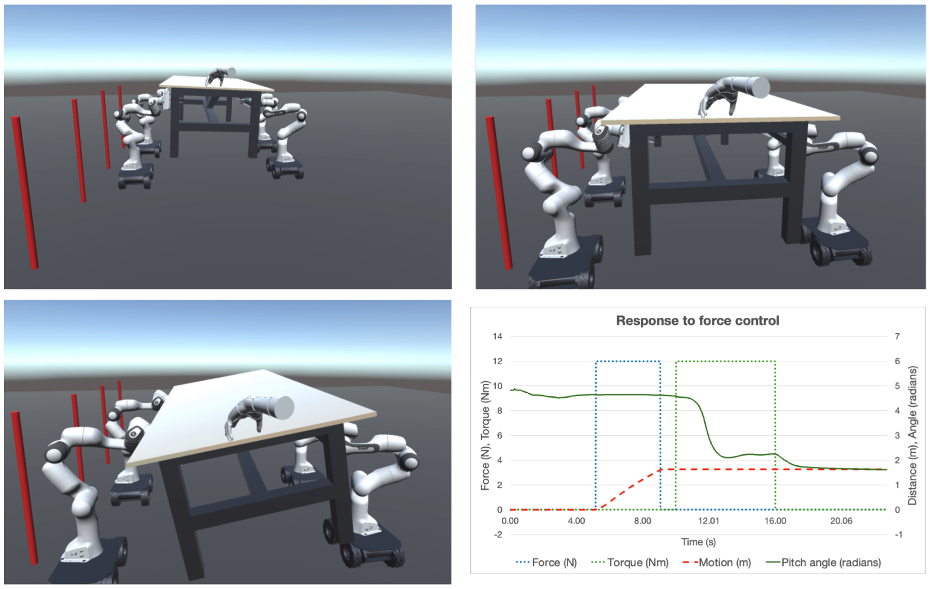


Fig. 5. Behaviour of a four robot collective using our controller, following a two-stage guiding force. Top left: stable state counteracting inertial load, before external force application. Top right: An external force is applied along the global X axis. Bottom left: An external torque is applied around the global X axis. Bottom right: The corresponding local Y and local pitch pose of one robot during this process. Video: [5].

human seeks to shift the robot's context to allow a 90 degree turn, and does this by applying a firm roll torque for 2s. The robot then shifts its constraints to a new dimensional set where it is free in (y, roll, pitch) but constrained in (x, z, yaw). See video at [5] (part 2).

The plot on the right of Figure 4 shows the roll torque applied to the robot through the shared load, and its corresponding roll angle. During the first phase (indicated by a red color on the input torque), the robot successfully rejects motion around its roll axis. Once the time/magnitude threshold for context switching has been passed, the contextual control dimension changes to d_1, and the roll input torque (now shown in blue) results in a corresponding angular rotation.

6.3 Multi-robot performance with changing task domain

To demonstrate that this control methodology can be extended to multiple robots with no direct communication, knowing only a (local) estimate of the load inertia, we implemented a simulated trial with four Franka Emika Panda robots using the Unity platform.

After grasping a shared load, each robot applies a joint torque sufficient to counteract the gravitational inertia of its portion of the manipulation object (assumed for this proof of concept to be known ahead of time). An external guiding force is then applied in two stages, first horizontally in the plane, then rotationally around the long axis of the load object.

The corresponding dimensional control constraints for each stage are:

$$\mathbf{d}_i = \begin{cases} \{1,1,0,0,0,1\} & i = 0 \\ \{1,0,1,0,1,0\} & i = 1 \end{cases} \tag{15}$$

Figure 5 shows that using only an external force input and a pre-calculated knowledge of load inertial parameters (note that the robots are not aware of the number, locations, or behaviour of other agents co-manipulating the object), the robot collective can ensure the load follows the guidance force. Video: [5] (part 3).

7 Discussion

A fundamental reason for deploying decoupled operational space control, in this task of robots performing collective manipulation with limited information, is to address disturbances imposed by the lack of shared knowledge. Without awareness of the others' poses or intentions, agents cannot disambiguate between interpretations of sensed forces, and thus in general will apply conflicting torques. This research demonstrates that (a) given a task context, these kinds of conflicts between agents can be resolved by using a compliant error-rejection framework; (b) if the agents are further provided with an estimation of load inertia, this task context can incorporate out-of-plane motion; and (c) force can be used as a control signal to enable contextual switching between control dimension subsets.

The use case for such a system is likely to be in collaborative transport or manipulation in complex, unknown, or dynamic environments. As §6.3 demonstrates, this kind of spontaneous collaboration can be effectively deployed with very little foreknowledge or preparation, requiring only one knowledgeable guiding agent. This simplifies the navigational challenges of collaborative transport considerably, and minimizes the shared knowledge base required. In particular, using compliant control for both elements of the decoupled controller eliminates any need for strict global kinematic alignment between agents; hence such systems should be robust to perturbations such as rough terrain or minor collisions.

To fully realize the goal of collective manipulation without prior knowledge or communication, further work is needed for letting agents estimate inertial parameters online, potentially during an interaction [6]. Since direct communication is not necessary for localised mass estimation [15], it may be feasible, e.g., to use an adaptive high-gradient stiffness controller [4] to implement a CM estimation routine within a shallow 6DoF low-stiffness basin common to all agents, solving the dynamic equations iteratively to allow each agent to calculate an individual estimate of the approximate CM location. Future work will focus on investigating such methods.

Acknowledgements This work was supported by a Space Technology Research Institutes grant (number 80NSSC19K1076) from NASA's Space Technology Research Grants Program.

References

1. Bai, H. and Wen, J.T., 2010. Cooperative load transport: A formation-control perspective. IEEE Transactions on Robotics, 26(4), pp. 742–750.
2. Carey, N. (2022). Decoupled Controller for the Franka Panda (Version 1.08) [Computer software]. https://github.com/niccarey/panda_decoupled_controller
3. Carey, N. (2020). Unity implementation of the Franka Panda platform(version 1.0) [Computer software]. https://github.com/niccarey/FrankaPanda_Unity
4. Carey, N.E. and Werfel, J., 2021, May. Collective transport of unconstrained objects via implicit coordination and adaptive compliance. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 12603–12609). IEEE.
5. Carey, N.E. and Werfel, J., 2022, Sept. A force-mediated controller for cooperative object manipulation [Media file, video] https://youtu.be/VIoj6v_a-Gw
6. Ćehajić, D. and Hirche, S., 2017, May. Estimating unknown object dynamics in human-robot manipulation tasks. In 2017 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1730–1737). IEEE.
7. Elwin, M.L., Strong, B., Freeman, R.A. and Lynch, K.M., 2022. Human-multirobot collaborative mobile manipulation: the Omnid Mocobots. arXiv preprint arXiv:2206.14293.
8. Franchi, A., Petitti, A. and Rizzo, A., 2015, May. Decentralized parameter estimation and observation for cooperative mobile manipulation of an unknown load using noisy measurements. In 2015 IEEE International Conference on Robotics and Automation (ICRA) (pp. 5517–5522). IEEE.
9. Habibi, G., Kingston, Z., Xie, W., Jellins, M. and McLurkin, J., 2015, May. Distributed centroid estimation and motion controllers for collective transport by multi-robot systems. In 2015 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1282–1288). IEEE.
10. Hichri, B., Fauroux, J.C., Adouane, L., Doroftei, I. and Mezouar, Y., 2019. Design of cooperative mobile robots for co-manipulation and transportation tasks. Robotics and computer-integrated manufacturing, 57, pp. 412–421.
11. Iggidr, A. and Sallet, G., 2003. On the stability of nonautonomous systems. Automatica, 39(1), pp. 167–171.
12. Khatib, O., 1987. A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal on Robotics and Automation, 3(1), pp. 43–53.
13. Kubus, D., Kroger, T. and Wahl, F.M., 2007, October. On-line rigid object recognition and pose estimation based on inertial parameters. In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 1402–1408). IEEE.
14. Liang, X., Su, T., Zhang, Z., Zhang, J., Liu, S., Zhao, Q., Yuan, J., Huang, C., Zhao, L. and He, G., 2022. An adaptive time-varying impedance controller for manipulators. Frontiers in Neurorobotics, 16.
15. Marino, A., Muscio, G. and Pierri, F., 2017, May. Distributed cooperative object parameter estimation and manipulation without explicit communication. In 2017 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2110–21116). IEEE.
16. Mavrakis, N. and Stolkin, R., 2020. Estimation and exploitation of objects' inertial parameters in robotic grasping and manipulation: A survey. Robotics and Autonomous Systems, 124, p. 103374.
17. Montemayor, G. and Wen, J.T., 2005, April. Decentralized collaborative load transport by multiple robots. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (pp. 372–377). IEEE.

18. Ott, C., 2008. Cartesian impedance control of redundant and flexible-joint robots. Springer.
19. Petitti, A., Franchi, A., Di Paola, D. and Rizzo, A., 2016, May. Decentralized motion control for cooperative manipulation with a team of networked mobile manipulators. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 441–446). IEEE.
20. Sadeghian, H., Villani, L., Keshmiri, M. and Siciliano, B., 2013. Task-space control of robot manipulators with null-space compliance. IEEE Transactions on Robotics, 30(2), pp. 493–506.
21. Sariyildiz, E., Sekiguchi, H., Nozaki, T., Ugurlu, B. and Ohnishi, K., 2018. A stability analysis for the acceleration-based robust position control of robot manipulators via disturbance observer. IEEE/ASME Transactions on Mechatronics, 23(5), pp. 2369–2378.
22. Wang, Z. and Schwager, M., 2016, May. Kinematic multi-robot manipulation with no communication using force feedback. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 427–432). IEEE.