Shared-control teleoperation methods for a cable-suspended dual-arm unmanned aerial manipulator

Mario Selvaggio, Federico Esposito, Vincenzo Lippiello, and Fabio Ruggiero

***Abstract*— This paper introduces two shared-control teleoperation methods for remotely executing long-reach tasks with a cable-suspended dual-arm unmanned aerial manipulator. The proposed techniques aim to improve task performance and user experience during remote tasks involving interaction with the environment. Two application scenarios are envisioned: pushing against a flat surface to emulate in-contact inspection tasks of infrastructures, and object grasping to simulate debris removal in cluttered environments. The effectiveness of the two shared-control teleoperation methods is evaluated through a human-subjects study involving 10 participants commanding the simulated robot via a joystick interface. Statistical analysis demonstrates significant enhancements in task performance and system usability when using the proposed methods compared to standard teleoperation.**

I. Introduction

Unmanned Aerial Manipulators (UAMs) hold great promise for performing in-contact tasks in high-altitude areas that are hard or dangerous for humans to reach [1]. The interaction capability of these robotic platforms allow them to maintain or repair high-voltage lines [2] or clear disaster scenarios from clutter and debris [3], [4]. However, interaction involves making intentional contact with the environment and this generates impulsive forces that may be dangerous or even destabilize the aerial platform. To avoid this problem, a recent trend in aerial manipulation is the inclusion of long, flexible cables between the suspension and the manipulation platforms [5]. On one hand, this configuration (also denoted as *long reach*) keeps the rotors' blades at a larger distance from the interaction spot, minimizing their risk of colliding while also absorbing the effects of unavoidable impacts experienced by the manipulator(s). On the other hand, it increases the control complexity as unactuated and flexible links cause pendulum-like oscillations of the manipulation system.

Despite notable strides in technology and algorithms enabling autonomous operations, aerial manipulation tasks performed with such systems remain challenging to execute entirely autonomously. The complexity of unstructured and dynamic environments further compounds the difficulty of these tasks requiring human-like capabilities in terms of perception and cognition. In such scenarios, human-in-the-loop teleoperation control is still the unique viable solution [6], [7]. However, it may easily become cognitively demanding for a human to online control the dynamics of such a complex flying robot along its many degrees of freedom (DOFs). Designing optimal human interfaces for aerial manipulation tasks is paramount to solve this problem for their fast deployment in the wild [8]. In shared-control teleoperation architectures, direct control of the human is usually put beside some autonomous control: the operator is in charge of controlling the lowest amount possible of the system's DOFs, which are relevant to the task execution, while the autonomy chooses the rest of the control inputs to accomplish the goal while improving the user's experience and the usability of the robot [9] (see Fig. 1).

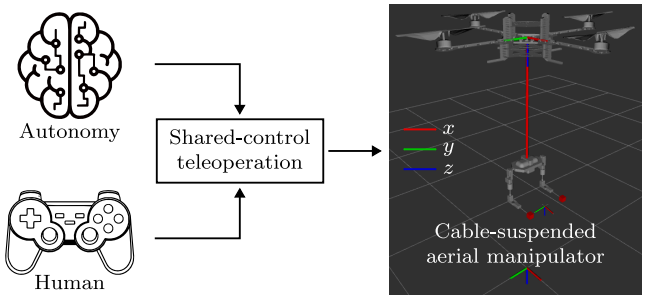

Fig. 1. Visual representation of the shared-control teleoperation architecture for cable-suspended aerial manipulation considered in this work. The human user specifies the arms centre point position and their relative distance while the autonomy decides the rest of the control inputs based on the task to be performed.

Inspired by this, this article aims to develop and evaluate shared-control teleoperation methods for a cable-suspended dual-arm UAM performing in-contact tasks. The robotic manipulation system considered in this work is the Lightweight & Compliant Arms for Service (LiCAS), shown in Figure 1, attached to a quadrotor platform by means of cables [10], [11], [12]. For this robot, two application scenarios are considered: *i) Pushing* against a flat surface: emulating in-contact infrastructure inspection situations where the robot has to push a sensor against a flat surface with its end-effectors and reach a desired value of the force exchanged to remain in contact during measurements [13]; *ii) Grasping* and transport of an object: emulating a debris removal task where the robot is supposed to approach, grasp, and then move away a box-shaped target object [14]. For each scenario, we propose a shared-control approach that we hypothesize would enhance the task performance while lowering the human physical/cognitive effort. The shared-control

This project has received funding from the European Union's Horizon Europe Framework Programme under grant agreement No 101070596 (euROBIN project), the AI-DROW project, in the frame of the PRIN 2022 research program, grant number 2022BYSBYX, funded by the European Union Next-Generation EU, and the the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie (grant agreement n. 953454). The authors are solely responsible for its content.

The authors are with the department of Electrical Engineering and Information Technology, University of Naples Federico II, 80125, Napoli, Italy. Corresponding author e-mail: mario.selvaggio@unina.it

modalities are systematically compared to baseline teleoperation methods along a human-subjects study involving 10 participants commanding the simulated robot via a joystick interface. The results of the statistical study show that task performance and system usability are significantly enhanced by our proposed methods.

II. Related Works

The field of aerial manipulation is incredibly vast, and several different robotic systems belonging to this branch have been developed so far [1]. Despite notable advancements in the field of perception and cognition, these robots are still far from achieving human-like interaction capabilities. For difficult tasks, the inclusion of a human in the control provides the necessary perception and cognition abilities to achieve superior eye-hand coordination. However, it may easily become cognitively demanding for a human to remotely control a complex mechanism, with many DOFs, which is kinematically different from the human embodiment.

Shared-control techniques combine the perception and cognition abilities of human operators with the autonomy provided by intelligent control algorithms to reduce the operating pressure of a human remotely operating a robot [15]. So far, shared-control techniques have been widely used for the control of fixed-base robot manipulators [16], [17], [18], however, there have been some attempts to make the teleoperation of aerial manipulators more effective. Coelho et al. propose a passivity-based control framework for multi-task time-delayed bilateral teleoperation and shared control of kinematically redundant robots with applications to aerial manipulation [7]. The suspended aerial manipulator presented in [19] is controlled to not only fulfill primary end-effector task but also provide the operator with the capability of steering the flying base to achieve a desired view from the camera attached to it (secondary task). Probine et al. propose shared-control strategies for the teleoperation of miniature indoor robotic airships, paired with an autonomous landing and charging system [20]. A series of experiments involving user-guided indoor exploration and autonomous landing validate the proposed framework. Masone et al. propose a novel bilateral shared-control framework for a cooperative aerial transportation and manipulation system composed of a team of micro aerial vehicles with a cable-suspended payload. The human operator is in charge of steering the payload and change online the desired shape of the formation of the robots. At the same time, an obstacle avoidance algorithm is in charge of avoiding collisions with the environment. The signals from the user and from the obstacle avoidance are blended in the trajectory generation module [21]. Kong et al. present the design and the teleoperation of a suspended aerial manipulation avatar for physical interaction in unstructured environments [3], which consists of a humanoid torso attached to a hexacopter. The proposed system is validated via indoor experiments reproducing post-disaster scenarios.

However, none of the above-listed works have ever considered the development of shared-control teleoperation interfaces for a cable-suspended aerial manipulator, as the one considered in this work. The main objective of this study is to propose and evaluate two shared-control methods for such a system along two tasks involving a consistent interaction with the environment.

III. Control Framework

A. Low-level Control

The considered UAM uses a decentralized control architecture. The hierarchical control adopted for the UAV is detailed in Sec. III-A.1. It compensates for the wrench due to the suspended manipulation platform that is estimated via a momentum-based observer described in Sec. III-A.2. Finally, arms position controllers are shown in Sec. III-A.3.

1) UAV control: The UAV is commanded by the hierarchical control described in [22]. It implements a slower outer loop, that generates the total thrust $u_T \in \mathbb{R}^+$ used for translational movements and the reference values for the faster inner loop, that controls the orientation variables through a torque $\tau_b \in \mathbb{R}^3$.

We assume the UAV dynamics evolves according to the following dynamic model:

$$\begin{cases} m\ddot{p}_b = mge_3 - u_T R_b e_3 + f_e \\ M\ddot{\eta}_b = -C\dot{\eta}_b + Q^T\tau^b + \tau_e \end{cases}, \tag{1}$$

where, in the first equation, $m \in \mathbb{R}^+$ is the mass of the UAV, p_b and $\dot{p}_b \in \mathbb{R}^3$ its translational state variables, $g = 9.81$ is the gravity constant, $e_3 = [0,\ 0,\ 1]^T$, in the second equation, $\eta_b\,(3\times1)$ represents the UAV's orientation, expressed through the roll, pitch and yaw angles ϕ, θ and ψ, $\omega_b^b \in \mathbb{R}^3$ is the angular velocity of the UAV with respect to the fixed world frame expressed in the body frame, Q indicates the (3×3) transformation matrix that links the derivatives of the Euler's angles to the angular velocity of the drone, i.e. $\omega_b^b = Q\dot{\eta}_b$, $M = Q^T I_b Q\,(3\times3)$, where $I_b \geq 0$ is the UAV's (3×3) diagonal inertia matrix, the (3×3) Coriolis matrix C is computed as $Q^T S(Q\dot{\eta}_b) I_b Q + Q^T I_b \dot{Q}$, where the (3×3) $S(\cdot)$ indicates the skew-symmetric operator. The terms f_e, $\tau_e \in \mathbb{R}^3$ are the resultant of all the external forces and torques which are applied to the drone; in our case, the main contribution to this terms comes from the presence of the cable-suspended dual-arm manipulator.

The outer loop control decides u_T, i.e. the total thrust generated by the UAV's propellers which is multiplied by the body rotation matrix $R_b \in \mathrm{SO}(3)$ in (1); it implements a PID-like controller with the feedforward of the desired acceleration and of the estimated external forces to compute the desired acceleration $\mu_d \in \mathbb{R}^3$ as in the following:

$$\mu_d = -K_p \begin{bmatrix} e_p \\ \dot{e}_p \end{bmatrix} - K_i \int_0^t e_p dt + \ddot{p}_{b,d} - \frac{f_{est}}{m}, \tag{2}$$

$$u_T = m\sqrt{{\mu_{d_x}}^2 + {\mu_{d_y}}^2 + (\mu_{d_z} - g)^2}, \tag{3}$$

$$\phi_d = \arcsin\left[\frac{m}{u_T}\left(\mu_{d_y}\cos\psi_d - \mu_{d_x}\sin\psi_d\right)\right], \tag{4}$$

$$\theta_d = \arctan\left[\frac{\mu_{d_x}\cos\psi_d + \mu_{d_y}\sin\psi_d}{\mu_{d_z} - g}\right]. \quad (5)$$

Exploiting the differential flatness property of the UAV we can compute reference values for the roll (4) and pitch (5) angles, that are input to the inner loop. In (2) we have defined the error variables as $e_p = p_b - p_{b,d} \in \mathbb{R}^3$ for the linear position and $\dot{e}_p = \dot{p}_b - \dot{p}_{b,d} \in \mathbb{R}^3$ for the linear velocity, with $p_{b,d} \in \mathbb{R}^3$ and $\dot{p}_{b,d} \in \mathbb{R}^3$ being the reference values for the position and velocity. We have used the subscript x, y and z to indicate the first, second and third component of the desired acceleration vector $\mu_d \in \mathbb{R}^3$ in equations (3), (4) and (5).

The inner loop, instead, generates τ^b that is the torque generated by the UAV's propellers. The structure of the inner loop controller is described by the following equations:

$$\tilde{\tau} = -K_e \begin{bmatrix} e_\eta \\ \dot{e}_\eta \end{bmatrix} - K_a \int_0^t e_\eta dt + \ddot{\eta}_{b,d}, \quad (6)$$

$$\tau^b = I_b Q \tilde{\tau} + Q^{-T} C \dot{\eta}_b - Q^T \tau_{est}, \quad (7)$$

where the error variables are defined as $e_\eta = \eta_b - \eta_{b,d} \in \mathbb{R}^3$ for the Euler's angles and $\dot{e}_\eta = \dot{\eta}_b - \dot{\eta}_{b,d} \in \mathbb{R}^3$ for their time derivatives, with $\eta_{b,d}$ and $\dot{\eta}_{b,d} \in \mathbb{R}^3$ being reference values, $\tau_{est} \in \mathbb{R}^3$ is the estimate of the external torque. Finally $K_i > 0\,(3\times 6)$ and $K_a > 0\,(3\times 3)$ are gain matrices.

The hierarchical control generates inputs in the form of the total thrust u_T and the control torque τ_b that are produced directly by the UAV rotors, commanding their desired speed ω, computed thanks to the inverse of the quadrotor's allocation matrix.

2) Momentum-based estimator: To avoid using heavy and expensive external sensors, the observation of the generalized momentum and the total energy of the robot is used to detect the unexpected occurrences, such as collisions, using only the already available proprioceptive measurements [23]. Such a collision identification technique has been used in [24] as an estimator of external wrenches and unmodeled dynamics that helps the UAV to reject the disturbance associated with aerodynamic effects, payloads, and physical interactions. We have implemented a momentum-based estimator that is able to quantify the external forces and torques applied to the robot, so that we can then use them to compute appropriately robust control inputs using (2) and (7). We have decided to use a second order estimator because the literature shows that it is an acceptable solution when it comes to UAVs [24]. The expressions of the generalized momentum and of its time derivatives are:

$$q = \begin{bmatrix} mI_3 & 0_3 \\ 0_3 & I_b \end{bmatrix} \begin{bmatrix} \dot{p}_b \\ \omega_b^b \end{bmatrix}, \quad \dot{q} = \begin{bmatrix} mge_3 - u_T R_b e_3 + f_e \\ C^T \dot{\eta}_b + Q^T \tau^b + \tau_e \end{bmatrix}, \quad (8)$$

while the equation that describes the evolution of the estimated disturbances f_{est} and τ_{est} is:

$$\begin{bmatrix} f_{est} \\ \tau_{est} \end{bmatrix} = K_2 \left(\int_0^t - \begin{bmatrix} f_{est} \\ \tau_{est} \end{bmatrix} + K_1 \left(q - \int_0^t \begin{bmatrix} f_{est} \\ \tau_{est} \end{bmatrix} + \right. \right. \left. \left. + \begin{bmatrix} mge_3 - u_T R_b e_3 \\ C^T \dot{\eta}_b + Q^T \tau^b \end{bmatrix} dt \right) dt \right). \quad (9)$$

When picking the transfer function to use to choose the gains for the estimator, we looked at Butterworth's polynomials, as they offer the flattest response possible in the passband.

3) Arms kinematic control: LiCAS A1 arms' movements are position-controlled and are realized through servomotors mounted on the manipulators' joints. This allows using kinematic-based algorithms to compute joint values starting from Cartesian pose of the arms' tips. To reduce the amount of user's commands, we opted to give the user the control of the arms center point, which is located at the middle of the two end-effectors, while keeping a constant orientation. The left and right arm tip reference position are extracted adding/subtracting a distance that can be chosen by the user to open or close the arms around an object. Denoting by $x_d \in \mathbb{R}^3$ the desired position of the arms center point (expressed in the shoulders reference frame, see Fig. 1) the left and right arms tips reference positions are computed as:

$$x_{d_{l,r}} = x_d \pm \left[0, L/2, 0\right], \quad (10)$$

where $x_{d_{l,r}} \in \mathbb{R}^3$ are the desired positions for the left and right arm tips in the same reference frame, and L represents the desired distance between the two, controlled by the user.

Since the arms only have 4 DoFs, we can not define both the position and the full orientation of their tips: in this case we consider a 3-DoFs task that only concerns tip positioning, so that we can exploit the arms' redundancy to allow for some optimization. The joint velocities command from Cartesian references are computed exploiting the differential inverse kinematics relation using the pseudo-inverse of the manipulators Jacobian, i.e.:

$$\dot{q}_{l,r} = J_{l,r}^\dagger \left[K(x_{d_{l,r}} - x_{l,r}) + \dot{x}_{d_{l,r}}\right] + N_{l,r}\dot{\hat{q}}_{l,r} \quad (11)$$

where $x_{d_{l,r}}$ and $\dot{x}_{d_{l,r}}$ are computed from (10) and contribute to construct the PD error in (11), $J_{l,r}(q_{l,r}) = \frac{\partial x_{l,r}}{\partial q_{l,r}} \in \mathbb{R}^{(3\times 4)}$ is the tip position task Jacobian, $J_{l,r}^\dagger$ its Moore–Penrose inverse, $K\ (3\times 3)$ is a gain matrix, $N_{l,r} = I - J_{l,r}^\dagger J_{l,r}\ (4\times 4)$ is the null-space projector of the joint velocity $\dot{\hat{q}}_{l,r}$, that can be used to optimize the arm configuration. For this have chosen to minimize at each instant the current distance between the arms' center of mass CoM and its initial position CoM_n; this task, however, is put at a lower priority level compared to the correct positioning of the arm's tips, so the distance will not always reach the absolute minimum value. This minimization is performed by choosing:

$$\dot{\hat{q}}_{l,r} = K\frac{\partial w}{\partial q_l}, \quad w = -\frac{1}{6}||CoM(q_{l,r}) - CoM_n(q_{l,r})||. \quad (12)$$

B. Teleoperation and Shared Control

The theory presented above is valid in the 3D space. For the rest of the paper, we restrict our focus to the 2D (plane x-z in Fig. 1). For each task the robotic system has to perform, two control techniques have been devised: a pure *teleoperation* one, in which only the human operator can specify both $p_{b,d}$ (the desired position for the drone, 2-DOFs), x_d (the desired position for the arms center point, 2-DOFs) and L (the distance between the arms tips, 1-DOF),

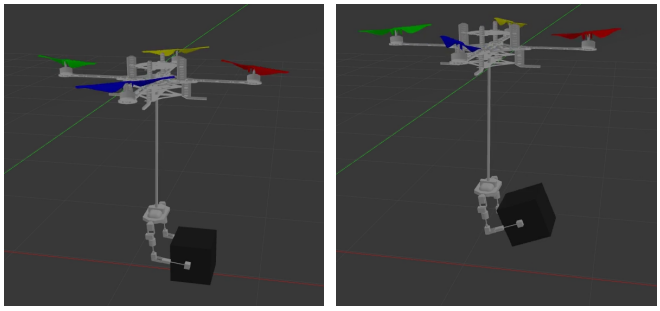

Fig. 2. UAM grasping task. Left: the arms approach the object and grasp it reducing their relative distance. Right: the UAM picks the object from the floor and transports it to a desired location.

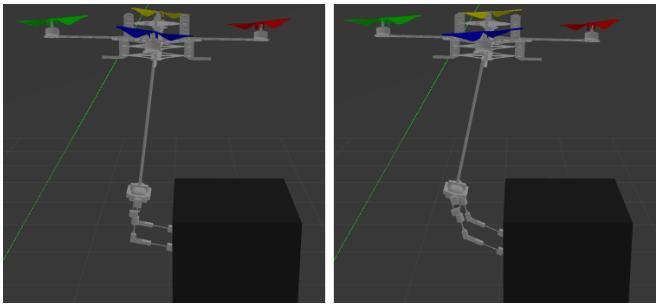

Fig. 3. UAM pushing against the flat surface task. Left: arms in the nominal configuration; Right: arms have been moved forward; the angle between the cables and the world's vertical axis is increased.

and a *shared-control* one, in which the autonomy takes over the control of $p_{b,d}$ while the human still takes care of x_d and L (see Fig. 1). As previously said, the user commands are specified via a joystick's interface, and the position of the two sticks on each of their axes is scaled and mapped to velocities $s = [s_x,\, s_z]^{\mathrm{T}}$ along the x and z axes, while a button is used to control the opening/closing movement of the arms. The human user can additionally choose to specify arm tip movements with respect to the shoulders or to the world depending on the situation. As already stated, in pure teleoperation the user controls the two parts of the robot (UAV and arms) independently, with a total of 5 reference commands to be imparted.

For positional commands specified with respect to the world frame, we compute the vector $\delta = [\delta_x,\, \delta_z]^{\mathrm{T}}$ representing the positional displacement of the drone/arms from the position they were in at the beginning of the task, as follows:

$$\begin{bmatrix}\delta_x \\ \delta_z\end{bmatrix} = \begin{bmatrix}\delta_x \\ \delta_z\end{bmatrix} + \frac{1}{f} R_s^w \begin{bmatrix}s_x \\ s_z\end{bmatrix}, \tag{13}$$

where $f \in \mathbb{R}^+$ is the rate at which the control runs and the rotation matrix $R_s^w \in \mathrm{SO}(2)$ is used to bring the velocities $s \in \mathbb{R}^2$ from the shoulder reference frame to the world reference frame[1]. The components of δ are initialized at zero. To avoid potential discontinuity in the robot's commanded position the two displacements δ_x and δ_z are filtered and mapped before altering the UAV's reference position. The filtered values δ'_x and δ'_z are computed as:

$$\begin{bmatrix}\delta'_x \\ \delta'_z\end{bmatrix} = K_o \begin{bmatrix}\delta'_x \\ \delta'_z\end{bmatrix} + K_n \begin{bmatrix}\delta_x \\ \delta_z\end{bmatrix}. \tag{14}$$

where $K_o = 0.9$ and $K_n = 0.1$ are the chosen filter gains.

1) Grasping an object: Executing a grasping action with the considered system in teleoperation mode requires continuously switching between controlling the drone and the arms and it might be too cognitively demanding and time consuming for a user. More specifically, the teleoperation method requires specifying commands using (13) alternatively for the UAV (in the world frame) and for the arms (in the shoulder frame). Our proposed shared-control method accommodates the user-specified arms linear movements (in the world frame), trying to restore the initial arms configuration by means of drone movements. This choice is made since arm movements are faster and more intuitive for the user with respect to specifying drone movements while accounting for the current arms configuration. To this end, the same δ', constructed from (13), is added to both the initial drone position and arms tip center point to get $p_{b,d}$ and x_d according to the following equations:

$$x_d(t) = x_d(0) + \delta'(t), \quad p_{b,d}(t) = p_{b,d}(0) + \delta'(t). \tag{15}$$

The effect of this shared-control action is a joint command of the drone/arms system useful for grasping tasks (see Fig. 2). In addition, a button is used to control the opening/closing movement of the arms incrementing or decrementing L in (10) by a certain amount at each press.

2) Pushing against a flat surface: The other shared-control algorithm we have designed deals with force control during pushing interactions with the environment: the human user maintains control over the arms' movements specifying x_d, while an autonomous force controller can be activated to alter the position of the drone in order to regulate the force exchanged to a desired value. The idea behind this controller stems from the observation that the amount of exchanged force is proportional to the sine of the angle formed by the cables with the vertical direction (see Fig. 3). There are two ways to modify this angle: changing the arms configuration or changing the drone position. The first method might cause the arms to straighten leading to singular configurations. The second method avoids such occurrence exploiting pivoting of the arms around the interaction point. Jointly commanding the drone and the arms to accomplish the force regulation task, avoiding the explained problems is too demanding for the user. Our shared-control method is responding to this need by autonomously allocating movements to the drone ($p_{b,d}$) in response to user commands of the arms tips (x_d). We conduct this task in quasi-static conditions so that the acceleration of the UAV does not have a large effect on the estimated force.

The autonomous controller consists of a PI loop. The input that the user gives to the algorithm is the absolute value of the desired pushing force. The PI controller acts on the error between the desired and the estimated force that the UAV

[1]Note: when s is specified with respect to the world frame R_s^w is the identity matrix.

experience along its $x-y$ axes, as well as on its integral, We then compute the displacements of the drone as:

$$\begin{bmatrix}\delta_x\\ \delta_y\end{bmatrix} = -\begin{bmatrix}K_p & 0\\ 0 & K_p\end{bmatrix}\begin{bmatrix}e_x\\ e_y\end{bmatrix} - \begin{bmatrix}K_i & 0\\ 0 & K_i\end{bmatrix}\int_0^t \begin{bmatrix}e_x\\ e_y\end{bmatrix} \mathrm{d}t, \quad (16)$$

where K_p and K_i are positive scalar gains, while e_x and e_y are the force errors, e.g. $e_x = f_{est,x} - f_{x,d}$. Our force control algorithm has been implemented by closing a (sensorless) force feedback loop around the existing position control loop that was offered by the hierarchical controller of the UAV.

IV. Experiments and Results

In this section, we first describe in details the software that we have used to build the simulator and to control the UAM (Sec. IV-A), then we describe the two tasks that the user must execute and present the experiments performed to compare our shared-control methods against baseline teleoperation, and finally we analyse the outcomes (Sec. IV-B).

A. Experimental Setup

All the simulations and following tests were carried out in the ROS/Gazebo environment at a frequency of $f = 100$ Hz; the control schemes for the UAV and for the arms were developed in Matlab/Simulink, and converted into C++ code thanks to the Simulink's automatic code generation tool. The generated C++ files were wrapped to exchange inputs and outputs through ROS topics with the UAM simulated in Gazebo and with the rest of the ROS nodes. We have utilised the RotorS library to accurately simulate the dynamics of the UAV system. The UAM was created following the approach in [25]. We modified one of the xacro file of the library changing the drone's inertial and geometrical parameters to reach a size and weight appropriate for carrying the dual-arm cable-suspended manipulator, then adjusted the positioning and enumeration of the rotors in accordance to the allocation matrix. The xacro file was completed by inserting in it the information regarding the LiCAS A1 arms, connecting the arms' to the UAV's body link via a 6-DoF kinematic chain representing the cable, as well as with the definition of some frames centered in the end-effectors, which are used by the inverse kinematics described earlier. RotorS library is used to control the drone: it simulates the rotors and the wrench they apply to the robot setting the speeds of the actuators through a ROS message; it also simulates, through Gazebo plugins, the odometry and IMU sensors that we can use to localize the robot in the world and to close the feedback loop on the position and attitude controller. The RotorS package for control of the multicopters' position and orientation lacked the robustness

TABLE I
Inertial and Geometrical Parameters of the Simulated UAV

Mass	Width	Height	Length
20.269 kg	0.6 m	0.24 m	0.645 m

Rotor radius	I_x	I_y	I_z
0.3 m	0.845 kgm^2	0.845 kgm^2	0.912 kgm^2

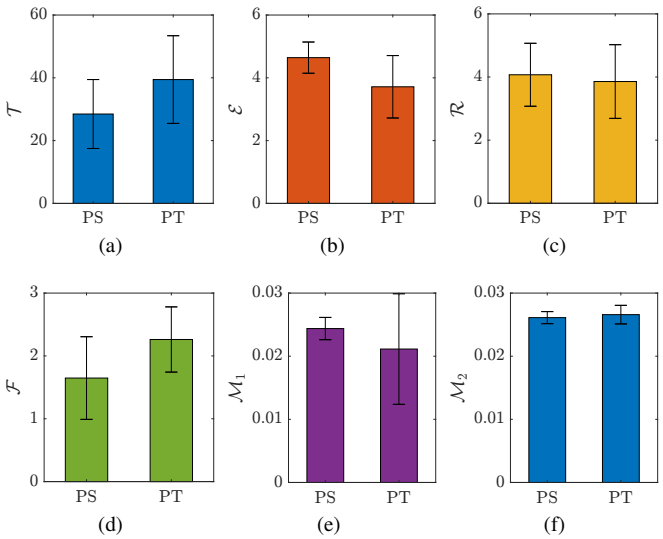

Fig. 4. Results of the wall pushing task. A standard deviation has been added and subtracted to show a realistic range of values around the means. (a) Time $\mathcal{T}$, (b) Ease of use $\mathcal{E}$, (c) Responsiveness $\mathcal{R}$, (d) Root mean square force error $\mathcal{F}$, (e) Minimum manipulabity measure $\mathcal{M}_1$, (f) Mean manipulabity measure $\mathcal{M}_2$.

needed to control a UAM rather than a UAV, so to solve this problem we decided to use our estimator-based controller presented earlier. The arms are PID controlled via ROS position/effort controllers. The joystick we used is the F710 Wireless Logitech Game Controller.

B. Human-subjects Study

To validate the proposed shared-control algorithms, we performed a comparative human-subjects study with a sample size of 10 people (8 male, 2 female, average age 27), well-accustomed to the field of robotics but who had never used our specific system before. Each of the subjects performed four tasks in a randomized order to decrease biases associated with novelty or with the increased experience that a user has on the second and following tests. Before the experiments, we explained to the subjects how to use the joystick to command the UAM, the situations to avoid (mainly singularity configurations caused by the over-extension of the arms) and what their goal was for each task; in case of simulation-breaking human errors we let the subject retry the test. At the end of each pair of tasks the subjects were compiling a survey in which they were questioned about their satisfaction i.e. the *ease of use* of the control interface and about its *responsiveness* on a scale from "Very Unsatisfied" (associated with a numerical value of 1) to "Very Satisfied" (associated instead with a value of 5). The tests have also been timed by the examiner, while "bag" files from ROS were recorded in order to have information about all the messages exchanged by the system during the tasks. To finalize the analysis of the data gathered during the experiments, we have implemented the Mann–Whitney U test (also known as Wilcoxon rank-sum test), which is a nonparametric test of the null hypothesis.

1) Pushing: The first task consists in using the UAM to push with arm tips against a flat surface and generate

a desired force. The subject receives the feedback in the form of the pushing force itself, which is presented both with its current numerical value and with a graph showing its recent evolution. In the meantime the simulation is also shown, so the user can watch how their commands affect the robot configuration. The two analysed cases are Pushing with Teleoperation (PT) and with Shared control (PS) devised as explained in Sec. III-B. In the PT task, the subject is told to try and keep the pushing force against the flat surface at an absolute value of 7 N by moving the robot's arms. In the PS task, the goal of the subject remains the same, as does the way they receive information from the system about the pushing force and the configuration of the robot. What changes is that the force control described previously is active, and the subjects are told that, as they move the robot's arms, a controller will also try to reach the desired force of $7\,\mathrm{N}$ by means of drone movements.

The data gathered shows us that the mean of the time required to complete the task ($\mathcal{T}$) in PT was 35.4380 s with a standard deviation of 10.7952 s, while in PS these values are brought to a mean of 29.7620 s and a standard deviation of 12.3565 s, as shown in Fig. 4a.

The PS task also received much better results in terms of ease of use and responsiveness, compared to the PT one: the mean of the ease of use ($\mathcal{E}$) for PT was 3.5, with a standard deviation of 1.0801, while the mean of the ease of use for PS was 4.6, with a standard deviation of 0.5164, which shows us that the shared-control approach was consistently preferred to the teleoperated one, as presented in Fig. 4b; the mean of the responsiveness ($\mathcal{R}$) for PT was 3.4, with a standard deviation of 1.0750, while the mean of the responsiveness for PS was 3.9, with a standard deviation of 1.1005, as shown in Fig. 4c.

To show that the shared-control approach is not only faster and better received but also more accurate, we show data about the root mean square error between the measured force ($\mathcal{F}$) and the desired force of $7\,\mathrm{N}$, both in the case of PT and PS. The computation is done on the whole simulation period, starting from the first user input, in order to reward attempts that quickly decreased the initial error and attempts that ended the simulation with a low final error in the steady state. Figure 4d shows how the addition of the shared-control algorithm makes the force control task much more accurate: on average the PT task has a root mean square error of $2.3592\,\mathrm{N}$ with a standard deviation of $0.5528\,\mathrm{N}$, while the PS task has an average root mean square error of $1.5850\,\mathrm{N}$ with a standard deviation of $0.7406\,\mathrm{N}$.

The data presented show that the addition of a shared-control technique in the pushing task has clear advantages: the presence of autonomy increases the accuracy and makes it so humans do not have to provide small and precise joystick inputs to get closer to the desired force value and without overshooting, while the addition of the human user can speed up the transient and thus aid the controller as well.

One last factor that we have studied from the recorded data of the experiment is the measure of the manipulability of the robotic arms, that tells us how far the configuration of the manipulator is from a kinematic singularity. The formula used for this computation at each simulation step is:

$$\mathcal{M}\left(q_{l,r}\right) = \sqrt{\det\left(J_{l,r}(q_{l,r}) J_{l,r}^{T}(q_{l,r})\right)}. \tag{17}$$

We have considered both the minimum value of the manipulability measure reached during the experiment ($\mathcal{M}_1$), in order to compare the worst-case scenarios (see Figure 4e), and the mean of the manipulability measure during the experiment ($\mathcal{M}_2$) (see Figure 4f).

In the first case we have that the mean value of the minimum manipulability measure in the PT task is 0.0196 with a standard deviation of 0.0101, while for the PS task it is 0.0246 with a standard deviation of 0.0016; if we instead consider the mean manipulability measures we have that it is 0.0263 with a standard deviation of 0.0017 for the PT task and 0.0264 with a standard deviation of 0.0009 for the PS task: in Fig. 4e we can see that the addition of the shared-control algorithm results in an improvement of the worst-case scenario, but its effects are barely noticeable in the average cases shown in Fig. 4f.

Table II summarizes the previously mentioned test results analyzed via Mann–Whitney U test. Statistically significant values ($p < 0.05$) are highlighted in bold.

2) Grasping: The second task consists in approaching, grasping, and moving an object to a desired goal location. In analogy to what presented for the pushing task we perform grasping in teleoperation (GT), and in shared control (GS) devised as explained in Sec. III-B. During the task the simulation is shown to the user, who can thus see how the drone, the arms, and the object are moving. In the GT task the goal of the subject is to bring the cube on top of the goal location by moving the robotic system and using the dual-arm manipulator to grasp and release the object. At the beginning of the task the subject can use the joystick to control the position of the arms, while the drone hovers in position; by pressing a button on the joystick the user can switch the state of the joystick wrapper, assuming the control of the UAV and leaving the arms in their last configuration or vice versa. An approach to avoid switching would involve two coordinated operators but it is not always practical. In

TABLE II
RESULTS FOR THE PT/PS TASK

Task: pushing against a flat surface				
Controller	**Metric**	**Average**	**Standard deviation**	***p***
PT	$\mathcal{T}$	$35.4380\,\mathrm{s}$	$10.7952\,\mathrm{s}$	0.2116
PS	$\mathcal{T}$	$29.7620\,\mathrm{s}$	$12.3565\,\mathrm{s}$	
PT	$\mathcal{E}$	3.5	1.0801	**0.0083**
PS	$\mathcal{E}$	4.6	0.5164	
PT	$\mathcal{R}$	3.4	1.0750	0.3248
PS	$\mathcal{R}$	3.9	1.1005	
PT	$\mathcal{F}$	$2.3592\,\mathrm{N}$	$0.5528\,\mathrm{N}$	**0.0173**
PS	$\mathcal{F}$	$1.5850\,\mathrm{N}$	$0.7406\,\mathrm{N}$	
PT	$\mathcal{M}_1$	0.0196	0.0101	0.6232
PS	$\mathcal{M}_1$	0.0246	0.0016	
PT	$\mathcal{M}_2$	0.0263	0.0017	0.4274
PS	$\mathcal{M}_2$	0.0264	0.0009	

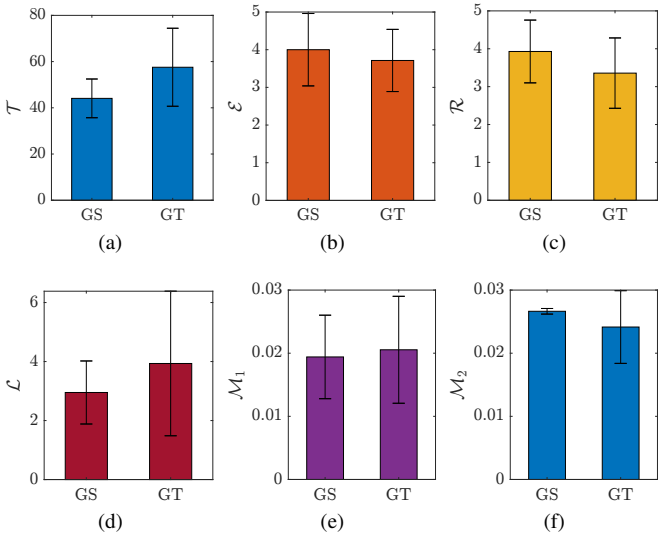

Fig. 5. Results of the grasping task. A standard deviation has been added and subtracted to show a realistic range of values around the means. (a) Time $\mathcal{T}$, (b) Ease of use $\mathcal{E}$, (c) Responsiveness $\mathcal{R}$, (d) Length of the path $\mathcal{L}$, (e) Minimum manipulabity measure $\mathcal{M}_1$, (f) Mean manipulabity measure $\mathcal{M}_2$.

the GS task the goal and the visualization protocol remain the same. At the beginning of the task the user activates the shared-control algorithm presented previously with the press of a button: from that point on, the joystick controls the arms while the drone tries to accommodate the arms' movements.

We now present the results obtained starting from a review of the data gathered from the surveys and the recorded execution times. The mean of the time required to grasp, lift and move the cube to the goal position ($\mathcal{T}$) with teleoperation is $59.1920\,\mathrm{s}$ with a standard deviation of $17.3281\,\mathrm{s}$, while the mean of the time needed to perform the same task using shared control is $45.1300\,\mathrm{s}$ with a standard deviation of $9.7709\,\mathrm{s}$, as shown in Fig. 5a.

The results from the surveys are less polarizing: in terms of ease of use ($\mathcal{E}$), the PT task GT received on average a grade of 3.6 with a standard deviation of 0.9661, while GS receives an average grade of 3.7 with a standard deviation of 0.9487, so we can conclude that the two control modes were perceived as fairly similar in terms of complexity of use, even though the shared-control one ended up having slightly better grades, as shown in Fig. 5b. One of the subjects has commented that they felt more confident about the control of the robot while using the teleoperation method, because they did not have to think about how the movement of the arms would affect the movements of the drone through the shared-control system; it can be noted, however, that the same subject required a time almost three times as long to complete the GT task compared to the GS task. Another interesting detail that we can take from the subjects comments and behavior is that two of them, while performing the teleoperation task, barely adjusted the arms position and only focused on the control of the drone, thus avoiding having to switch between the two control modes presented in the GT task.

The difference perceived between the two tasks in terms

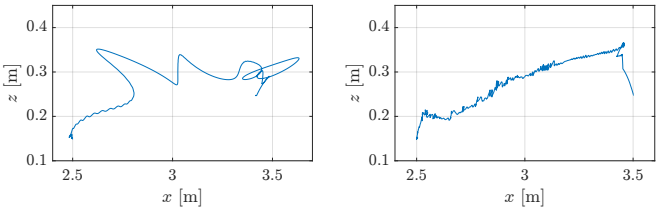

Fig. 6. Example of the cube's path on the $x-z$ plane using the GT (on the left) and the GS architectures (on the right).

of responsiveness ($\mathcal{R}$) is larger: GT received a mean value of 3.3 with a standard deviation of 0.9487, while GS received a mean value of 3.8 with a standard deviation of 0.9189, as shown in Fig. 5c.

Continuing the analysis of the results of the grasping task, we now show objective results about the length of the path followed by the cube ($\mathcal{L}$), from the moment of its grasping to the last moment of the recorded simulations; of course a shorter path relates to the task being executed more efficiently. As previously stated, we have let subjects retry the test in case the simulation broke; despite that, in the recorded data, we still have a subject that caused a singularity in the arms while executing the GT task and one that had similar problems with the GS task.

We see that the average path length ($\mathcal{L}$) for the GT task is $4.0645\,\mathrm{m}$ with a standard deviation of $2.6162\,\mathrm{m}$, while for the GS task the average is $3.1571\,\mathrm{m}$ with a standard deviation of $1.1837\,\mathrm{m}$. In the path length computation we have also considered the distance that the cube travels if it falls off the goal platform, as this gives us an indication of the precision with which the cube has been positioned above the platform.

We can see that the use of the shared-control algorithm lets the user travel a shorter path while completing the task; if we compare an example of the plotted path that the cube follows in the two tasks performed by the same subject (Figure 6), we can also see that the addition of shared control lets the user command a more intuitive trajectory to the robot. It is important to highlight that in both GT and GS the control scheme of the UAV remained the same, and that the trend of the path in the GS being much more linear compared to that in the GT task is seen in all the experiments recorded.

The manipulability analysis presented in Fig. 5e shows how, in the GT and GS tasks, the addition of shared control can hardly be appreciated when considering the worst-case scenario of the manipulability measure during the experiment ($\mathcal{M}_1$): in the first case in fact we have that GT has a mean value of 0.0186 with a standard deviation of 0.0094, while GS has a mean value of 0.0196 with a standard deviation of 0.0078. Moving our attention to the average case ($\mathcal{M}_2$) presented in Fig. 5f, instead, we see that the presence of the shared control results in some noticeable improvements, as the GT task has a mean value of 0.0230 with a standard deviation of 0.0066 while the GS task has a mean value of 0.0267 and a standard deviation of just 0.0005.

One last thing to consider is that, during the experiments, the task in which the subjects more often reached kinematic singularities and thus needed to restart was GS. As we have

TABLE III

RESULTS FOR THE GS/GT TASK

Task: grasping a box				
Controller	**Metric**	**Average**	**Standard deviation**	***p***
GT	$\mathcal{T}$	$59.1920\,\mathrm{s}$	$17.3281\,\mathrm{s}$	**0.0283**
GS	$\mathcal{T}$	$45.1300\,\mathrm{s}$	$9.7709\,\mathrm{s}$	
GT	$\mathcal{E}$	3.6	0.9661	0.9351
GS	$\mathcal{E}$	3.7	0.9487	
GT	$\mathcal{R}$	3.3	0.9487	0.2332
GS	$\mathcal{R}$	3.8	0.9189	
GT	$\mathcal{L}$	$4.0645\,\mathrm{m}$	$2.6162\,\mathrm{m}$	0.4274
GS	$\mathcal{L}$	$3.1571\,\mathrm{m}$	$1.1837\,\mathrm{m}$	
GT	$\mathcal{M}_1$	0.0186	0.0094	0.6232
GS	$\mathcal{M}_1$	0.0196	0.0078	
GT	$\mathcal{M}_2$	0.0230	0.0066	**0.0257**
GS	$\mathcal{M}_2$	0.0267	0.0005	

said previously, the shared-control algorithm controls the arms by publishing the reference frame with respect to the world, and then moves the drone to accommodate the arms' movements; while at steady state this results in a translation of the whole UAM in the direction commanded by the user, during the transient it is possible, by giving commands to the arms in rapid succession or by mixing commands along two or more axes, to bring the arms in a configuration that either reaches or approaches a singularity. As Figures 5e and 5f show, however, by eliminating these swift or complex commands, the manipulability measure results to be higher than the one seen in teleoperation.

A summary of the results of the grasping task is shown in Table III where statistically significant values ($p < 0.05$), found via the Mann–Whitney U test, are highlighted.

V. CONCLUSION AND FUTURE WORK

This paper presented two shared-control teleoperation algorithms for cable-suspended aerial manipulation systems, aiming to enhance performance of remotely executed interaction tasks. The proposed methods facilitate the generation of desired pushing forces during interactions with flat surfaces and enable seamless approach, grasping, and manipulation of lightweight objects. Experimental comparisons between the shared-control solutions and conventional teleoperation methods were conducted for both surface-pushing and grasping tasks. The performance of 10 subjects testing the system was analyzed using the Wilcoxon rank-sum test to identify significant effects on various metrics. Results indicate that shared control outperforms simple teleoperation based on both subjective (user surveys) and objective (time measurements, force error measurements, and distance traveled measurements) criteria. Analysis of manipulability demonstrates significant improvements in worst-case scenarios for the pushing task and in mean values for the grasping task. The statistical analysis also highlights significant differences in ease of use and force error for pushing, and in execution time and mean manipulability for grasping.

Future work may focus on comparing with other baseline methods (e.g. involving two coordinated operators) or devising other shared control schemes to further enhance user performance and task effectiveness.

REFERENCES

[1] F. Ruggiero, *et al.*, "Aerial manipulation: A literature review," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, 2018.

[2] A. Ollero, *et al.*, "Aerial-core: Ai-powered aerial robots for inspection and maintenance of electrical power infrastructures," in *arXiv preprint*, Jan. 2024.

[3] F. Kong, *et al.*, "A suspended aerial manipulation avatar for physical interaction in unstructured environments," *CoRR*, vol. abs/2310.03586, 2023.

[4] D. Mellinger, *et al.*, "Design, modeling, estimation and control for aerial grasping and manipulation," in *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2011, pp. 2668–2673.

[5] R. Miyazaki, *et al.*, "Long-reach aerial manipulation employing wire-suspended hand with swing-suppression device," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3045–3052, 2019.

[6] A. Coelho, *et al.*, "Whole-body bilateral teleoperation of a redundant aerial manipulator," in *2020 IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 9150–9156.

[7] ——, "Whole-body teleoperation and shared control of redundant robots with applications to aerial manipulation," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 1, p. 14, Apr 2021.

[8] B. B. Kocer, *et al.*, "Immersive view and interface design for teleoperated aerial manipulation," in *2022 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2022, pp. 4919–4926.

[9] M. Selvaggio, *et al.*, "Autonomy in physical human-robot interaction: A brief survey," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7989–7996, 2021.

[10] A. Suarez, *et al.*, "Lightweight and human-size dual arm aerial manipulator," in *2017 Int. Conf. on Unmanned Aircraft Systems*, 2017, pp. 1778–1784.

[11] ——, "Design of a lightweight dual arm system for aerial manipulation," *Mechatronics*, vol. 50, pp. 30–44, 2018.

[12] ——, "Lightweight and compliant long reach aerial manipulator for inspection operations," in *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 6746–6752.

[13] A. Jimenez-Cano, *et al.*, "Aerial manipulator for structure inspection by contact from the underside," in *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015, pp. 1879–1884.

[14] H. Seo, *et al.*, "Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control," in *2017 IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 6362–6368.

[15] G. Li, *et al.*, "The classification and new trends of shared control strategies in telerobotic systems: A survey," *IEEE Transactions on Haptics*, vol. 16, no. 2, pp. 118–133, 2023.

[16] M. Selvaggio, *et al.*, "Passive task-prioritized shared-control teleoperation with haptic guidance," in *2019 Int. Conf. on Robotics and Automation*, 2019, pp. 430–436.

[17] ——, "A shared-control teleoperation architecture for nonprehensile object transportation," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 569–583, 2022.

[18] F. Stroppa, *et al.*, "Shared-control teleoperation paradigms on a soft-growing robot manipulator," *Journal of Intelligent & Robotic Systems*, vol. 109, no. 2, p. 30, Sep 2023.

[19] Y. S. Sarkisov, *et al.*, "Development of sam: cable-suspended aerial manipulator," in *2019 Int. Conf. on Robotics and Automation*, 2019, pp. 5323–5329.

[20] C. Probine, *et al.*, "A shared control teleoperation framework for robotic airships: Combining intuitive interfaces and an autonomous landing system," in *2021 IEEE Int. Conf. on Systems, Man, and Cybernetics*, 2021, pp. 1028–1034.

[21] C. Masone *et al.*, "Shared control of an aerial cooperative transportation system with a cable-suspended payload," *Journal of Intelligent & Robotic Systems*, vol. 103, no. 3, p. 40, Oct 2021.

[22] K. Nonami, *et al.*, *Autonomous flying robots*. Tokyo, Japan: Springer, Mar. 2010.

[23] A. D. Luca, *et al.*, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," *2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1623–1630, 2006.

[24] F. Ruggiero, *et al.*, "Passivity-based control of vtol uavs with a momentum-based estimator of external wrench and unmodeled dynamics," *Robotics and Autonomous Systems*, vol. 72, pp. 139–151, 2015.

[25] G. D'Ago, *et al.*, "Modelling and identification methods for simulation of cable-suspended dual-arm robotic systems," *Robotics and Autonomous Systems*, p. 104643, 2024.