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

Scalable Collaborative Manipulation with Distributed Trajectory Planning

Ola Shorinwa[1], Mac Schwager[2]

***Abstract*— We present a distributed algorithm to enable a group of robots to collaboratively manipulate an object to a desired configuration while avoiding obstacles. Each robot solves a local optimization problem iteratively and communicates with its local neighbors, ultimately converging to the optimal trajectory of the object over a receding horizon. The algorithm scales efficiently to large groups, with a convergence rate constant in the number of robots, and can enforce constraints that are only known to a subset of the robots, such as for collision avoidance using local online sensing. We show that the algorithm converges many orders of magnitude faster, and results in a tracking error two orders of magnitude lower, than competing distributed collaborative manipulation algorithms based on Consensus alternating direction method of multipliers (ADMM).**

I. Introduction

In many situations, manipulating an object requires multiple robots. We present the Scalable Optimal Collaborative Manipulation with Local Constraints (SOCM LoCo) algorithm, a distributed algorithm by which a group of robots can collaboratively move an object to a desired configuration while avoiding obstacles in the environment. Each robot only communicates with its neighbors over a connected communication network, making the algorithm distributed. The algorithm's efficient scalability arises from its constant computational complexity, independent of the number of robots, allowing for an arbitrary number of robots to collaborate. The algorithm also allows for individual robots to sense obstacles online and impose collision avoidance constraints without other robots explicitly knowing about these constraints. In simulation, we show that SOCM LoCo provides 100 times better tracking performance and several orders of magnitude faster convergence rate than a distributed method based on Consensus alternating direction method of multipliers (ADMM) and performs comparably to leader-follower methods.

Our algorithm would be useful in a variety of collaborative manipulation applications, including in automated manufacturing or warehouse environments and in autonomous construction of buildings or structures in hazardous or remote environments such as in space. In these applications, a team of robots can work together to move large parts or sub-assemblies into place, providing greater scalability than traditional monolithic robots. In addition, our algorithm can be applied in disaster relief scenarios, where heavy and varied debris must be cleared by a group of robots to aid emergency workers in search of survivors. In all these cases, our algorithm enables the coordination of the group of robots in manipulating objects.

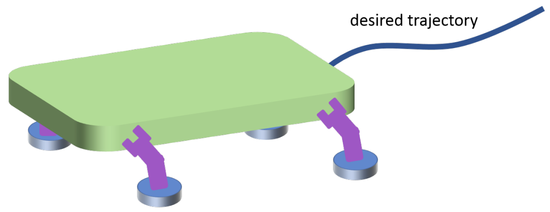

Fig. 1. A group of robots jointly manipulating an object along a desired trajectory. The robots compute a collision free trajectory, even if only a subset of the robots know about the presence of obstacles through local sensing.

The algorithm proceeds in a receding horizon fashion. Each robot solves a series of local optimization problems iteratively and communicates with its neighbors over a communication network, enabling the entire group to manipulate the object along an optimal trajectory respecting the object's dynamics, collision avoidance constraints, and input constraints of the robots. For convex constraints and affine object dynamics, our algorithm is guaranteed to converge to the globally optimal trajectory. In more practical cases with non-convex constraints such as collision avoidance and non-linear dynamics, including rotational dynamics, the algorithm gives locally optimal, collision free trajectories. The algorithm is inspired by the Separable Optimization Variable ADMM (SOVA) method [1]. Specifically, we use a separable optimization variable property unique to the collaborative manipulation problem to greatly reduce the number of optimization variables handled by each robot, thereby leading to faster convergence and better performance than other ADMM methods.

The contributions of this work are as follows. We derive the SOCM LoCo algorithm and prove its convergence to the globally optimal trajectory for an object with affine dynamics and convex constraints. Through iterative linearization, we extend the method to the case of non-linear dynamics and non-convex constraints. Further, we show the algorithm has a constant computational complexity, independent of the number of robots, while other methods scale cubically in the number of robots. In simulation, we show SOCM LoCo converges many orders of magnitude faster and attains a tracking error 100 times smaller than distributed methods based on Consensus ADMM.

[1,2] O. Shorinwa and M. Schwager are with the Department of Mechanical Engineering and Department of Aeronautics and Astronautics, Stanford University, USA.

This work was funded in part by DARPA YFA grant D18AP00064, NSF NRI grant 1830402, and ONR grant N00014-18-1-2830. We are grateful for this support.

This paper is organized as follows: in Section II, we note previous approaches and formulate the manipulation task as an optimization problem in Section III. We derive our distributed algorithm using the alternating direction method of multipliers in Section IV and apply our algorithm to objects with non-convex dynamics in Section V. In Section VI, we demonstrate our algorithm with as many as 100 robots collaboratively manipulating an object under different communication constraints. We provide concluding remarks in Section VII.

II. Related Works

Many approaches achieve collaborative manipulation using centralized control schemes which aggregate local information from each robot to compute individual motor commands [2], [3], [4]. These approaches require significant computation and communication and do not scale with the number of robots involved in the manipulation task. To allow for distributed schemes, previous approaches use potential fields to guide each robot to the object and subsequently manipulate the object by enclosing it, described as object closure or caging [5], [6]. These methods do not provide a mechanism for specifying local constraints for each robot which our method provides. To handle constraints, some methods plan for collision free trajectories in convex spaces of the environment centrally and follow these trajectories using local controllers [7], [8]. Our method does not require any centralized computation procedure, making it fully distributed.

Some other approaches employ a leader-follower architecture in which a single robot (leader) actively manipulates the object along a desired trajectory while the other robots (followers) infer the motion of the leader and move in consistency with the leader's motion. In some approaches [9], the followers communicate their motion constraints explicitly to the leader which shares the desired trajectory to all robots, incurring significant communication and poor scalability, while others require a human operator to issue commands for the group of robots [10]. Some other methods allow for collaboration between the leader and followers without communication. In these approaches, only the leader knows its desired trajectory. The followers estimate the leader's trajectory using force sensors and move along with the leader using non-linear feedback controllers [11], [12], [13], impedance controllers [14], [15], [16], and adaptive controllers [17]. For small groups of robots, the followers can estimate the leader's trajectory using impedance controllers without force sensors [18], with greater tracking errors for larger groups. All the above approaches allow the robots to follow the leader's trajectory but do not address trajectory planning for the group of robots. In contrast, our method finds the optimal trajectory for the object, avoiding collisions in the environment.

Other distributed methods describe the desired object trajectory using impedance without a designated leader [19]. The desired object impedance is distributed among the robots to compute individual control inputs using knowledge of the geometric relations between the robots. Each robot receives its desired trajectory before the task and follows the specified trajectory through impedance control. Like leader-follower approaches, these methods allow for following a desired trajectory without trajectory planning. In addition, these methods are suited to manipulation tasks involving a few robots (2 to 5 robots) with known grasp points and do not scale to large groups of robots.

Our method enables a group of robot to collaboratively plan an optimal trajectory for an object, responding to avoid obstacles in the environment known by a subset of the robots, while manipulating the object. In addition, with constant computational complexity, our algorithm scales to large groups of robots. We derive our algorithm using the alternating direction method of multipliers (ADMM), exploiting separability of the optimization variable to achieve lower computation and communication complexity. ADMM has been applied in receding horizon control, albeit in a centralized fashion [20]. Our distributed method provides greater tolerance to errors through feedback with a receding horizon approach.

III. Problem Formulation

We desire to manipulate an object from an initial configuration to a desired final configuration by controlling a group of robots to work in collaboration. Each robot grasps the object before manipulating it. We denote the object's configuration and velocities as $\mathbf{x}_{\text{obj}}$ which includes its position and translational velocity and its orientation and angular velocity. We consider a group of N robots manipulating the object. Robot i applies force F_i and torque Γ_i to the object. We can express the manipulation task as an optimization problem given by

$$\begin{aligned} \text{minimize} \quad & \int_{t=0}^{T} \Big(\psi(\mathbf{x}_{\text{obj}}(t)) + \sum_{i=1}^{N} \beta_i(F_i(t), \Gamma_i(t))\Big) dt \\ \text{subject to} \quad & g(\mathbf{x}_{\text{obj}}(t), F(t), \Gamma(t)) = 0 \quad \forall t \\ & h(\mathbf{x}_{\text{obj}}(t), F(t), \Gamma(t)) \le 0 \quad \forall t \end{aligned} \tag{1}$$

We include a desired trajectory tracking objective for the manipulated object $\psi(\cdot)$ and encode desired behaviors for robot i such as to minimize energy or fuel consumption in $\beta_i(\cdot)$. In addition, we consider dynamic constraints and equality constraints on the initial configurations and velocities of the object and robots given by $g(\cdot)$. We also incorporate additional convex constraints on the object's configuration and its derivatives denoted by $h(\cdot)$ which can include collision avoidance constraints represented by safe convex zones within the environment.

Communication Graph

We represent the robots as nodes in an undirected graph $\mathcal{G}$ described by a set of vertices $\mathcal{V} = \{i \mid i = 1, \cdots, N\}$ and a set of edges $\mathcal{E}$. An edge (i, j) exists in $\mathcal{E}$ if robot i can communicate with robot j. We assume the communication graph $\mathcal{G}$ is connected i.e. we can obtain a path linking every pair of nodes from the edges in $\mathcal{E}$. In addition, we denote the neighbor set of robot i as $\mathcal{N}_i = \{j \mid (i,j) \in \mathcal{E}\}$ consisting of robots which can communicate with robot i.

IV. Distributed Control

Distributed approaches to solving (1) typically involve communicating all relevant problem information to a single robot designated as a leader which computes and shares the solution to the other robots or followers. However, these approaches require significant communication especially for large groups of agents. With our approach, each robot computes its applied force and torque without a designated leader. To derive a scalable distributed scheme, we express problem (1) as

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N} \int_{t=0}^{T} \Big(\psi_i(\mathbf{x}_{i,\text{obj}}(t)) + \beta_i(F_i(t), \Gamma_i(t))\Big) dt \\
\text{subject to} \quad & g_i(\mathbf{x}_{i,\text{obj}}(t), F_i(t), \Gamma_i(t)) = 0 \quad \forall t,\ \forall i \in \mathcal{V} \\
& h_i(\mathbf{x}_{i,\text{obj}}(t), F_i(t), \Gamma_i(t)) \leq 0 \quad \forall t,\ \forall i \in \mathcal{V} \\
& \mathbf{x}_{i,\text{obj}}(t) = \mathbf{x}_{j,\text{obj}}(t) \quad \forall t,\ \forall j \in \mathcal{N}_i,\ \forall i \in \mathcal{V}
\end{aligned} \tag{2}$$

where each robot maintains a local copy of the trajectory of the object $\mathbf{x}_{\text{obj}}$ and $\psi_i(\mathbf{x}_{i,\text{obj}}(t))$ represents the tracking objective for the object trajectory associated with robot i. To retain the same objective function in (2), each robot receives the same tracking objective function for its local copy of the object trajectory. We introduce the equality constraints on local copies of the object's trajectory to ensure that all local object's trajectories coincide.

Theorem 1. *The optimization problem expressed in* (2) *is equivalent to* (1) *with the same optimal control inputs for each robot and associated object trajectory.*

Proof. Noting that the communication graph $\mathcal{G}$ is connected, the equality constraint in (2) ensures that all robots maintain the same object trajectory ($\mathbf{x}_{i,\text{obj}}(t) = \mathbf{x}_{\text{obj}}(t)\ \forall i, t$). Consequently, the objective function in (2) simplifies to the objective in (1). Both problems are thus equivalent with the same minimizers and optimal objective value. $\square$

With this formulation, the objective function in (2) is separable among the robots, and thus we can develop fully distributed schemes for solving the optimization problem. We discretize the optimization problem in (1) for transcription to a numerical optimization problem. The size of the optimization problem grows super-linearly with the duration of the manipulation task and the number of collaborating robots which makes solving the entire problem challenging in real-time. Meanwhile, many robotic tasks involve object manipulation over considerable time periods. Thus, we take a receding horizon control approach to solving (2). We solve the optimization problem over a smaller time horizon consisting of N_τ stages, apply the control inputs from the first stage, and repeat the procedure at each time interval. In addition, this approach provides added robustness to errors through feedback at each time interval. The resulting optimization problem at each time instance is given by

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N} \sum_{\tau=0}^{N_\tau - 1} \Big(\psi_i(\mathbf{x}_{i,\text{obj}}(\tau)) + \beta_i(F_i(\tau), \Gamma_i(\tau))\Big) \\
\text{subject to} \quad & g_i(\mathbf{x}_{i,\text{obj}}(\tau), F_i(\tau), \Gamma_i(\tau)) = 0 \quad \forall \tau,\ \forall i \in \mathcal{V} \\
& h_i(\mathbf{x}_{i,\text{obj}}(\tau), F_i(\tau), \Gamma_i(\tau)) \leq 0 \quad \forall \tau,\ \forall i \in \mathcal{V} \\
& \mathbf{x}_{i,\text{obj}}(\tau) = \mathbf{x}_{j,\text{obj}}(\tau) \quad \forall \tau,\ \forall j \in \mathcal{N}_i,\ \forall i \in \mathcal{V}
\end{aligned} \tag{3}$$

A naive method for solving the optmization in (3) involves distributing local copies of the entire optimization variable among the robots as in previous ADMM methods [21]. With this approach, each robot computes the control inputs for all robots within the group but only applies the control inputs relevant to it. The size of the optimization variables is $O(N)$ and thus scales linearly with the number of robots. Consequently, this approach incurs significant computation and communication complexity for larger groups of robots and number of stages N_τ.

To overcome the drawbacks of this naive approach, we employ the SOVA method in [1] which produces a distributed procedure with faster convergence rates and much lower computation and communication complexity, by noting the optimization variable is separable among the robots. Moreover, each robot has no explicit knowledge of the local constraints of the other robots with our approach. Figure 2 illustrates the distribution of the optimization variables among the robots. Each robot optimizes over variables relevant to its actions.

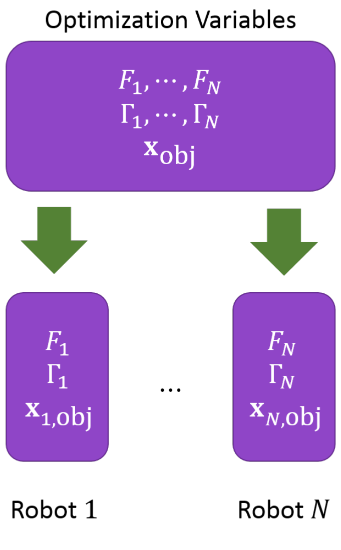
Optimization Variables
$F_1, \cdots, F_N$
$\Gamma_1, \cdots, \Gamma_N$
$\mathbf{x}_{\text{obj}}$
F_1
Γ_1
$\mathbf{x}_{1,\text{obj}}$
...
F_N
Γ_N
$\mathbf{x}_{N,\text{obj}}$
Robot 1
Robot N

Fig. 2. Distribution of the optimization variables among the robots.

In this approach, the optimization variable is distributed among the robots with equality constraints on corresponding elements within each robot's optimization variable. We denote robot i's contribution to the objective function in (3) as $J_i(\cdot)$ and drop the inputs to J_i. We represent the vertical concatenation of $\mathbf{x}_{i,\text{obj}}(\tau)$, $\tau = 0, \cdots, N_\tau - 1$, as $\mathbf{x}_{i,\text{obj}}$. Likewise, we represent the vertical concatenation of the control inputs over the same interval as $\mathbf{F}_i$ and $\mathbf{\Gamma}_i$. The augmented Lagrangian $\mathcal{L}_a$ of (3) is

$$\begin{aligned}\mathcal{L}_a = \sum_{i=1}^{N} \Big(& J_i(\cdot) + c_i^{\mathbf{T}} g_i(\cdot) + d_i^{\mathbf{T}} h_i(\cdot) \\ & + \sum_{j \in \mathcal{N}_i} \left(\lambda_{ij}^{\mathbf{T}} (\mathbf{x}_{i,\text{obj}} - v_{ij}) + \gamma_{ij}^{\mathbf{T}} (\mathbf{x}_{j,\text{obj}} - w_{ij}) \right) \\ & + \frac{\rho}{2} \sum_{j \in \mathcal{N}_i} \left(\|\mathbf{x}_{i,\text{obj}} - v_{ij}\|_2^2 + \|\mathbf{x}_{j,\text{obj}} - w_{ij}\|_2^2 \right) \Big)\end{aligned} \quad (4)$$

with the slack variables v_{ij} and w_{ij} for the equality constraints on the local object's trajectories and c_i, d_i, λ_{ij}, and γ_{ij} as dual variables on the problem constraints $g_i(\cdot)$, $h_i(\cdot)$, and the object trajectory constraints respectively with $d_i \geq 0$. The augmented Lagrangian (4) includes the penalty ρ on the residual in the equality constraints on the local object trajectories. Each robot's optimization variable is updated iteratively as the minimizers of the augmented Lagrangian while the dual variables associated with the problem constraints are updated through dual ascent on the augmented Lagrangian. The update step for the force and torque applied by each robot and the object trajectory reduces to the solution of a convex minimization problem for which efficient optimization solvers exist. We denote the vertically concatenated control inputs and trajectories at robot i as $\mathbf{r}_i = (\mathbf{x}_{i,\text{obj}}, \mathbf{F}_i, \mathbf{\Gamma}_i)$ and update $\mathbf{r}_i$ using

$$\begin{aligned}\mathbf{r}_i^{k+1} = \underset{\mathbf{x}_{i,\text{obj}}, \mathbf{F}_i, \mathbf{\Gamma}_i}{\text{argmin}} \max_{c_i, d_i} \Bigg\{ & J_i(\cdot) + c_i^{\mathbf{T}} g_i(\cdot) + d_i^{\mathbf{T}} h_i(\cdot) + q_i^{k\mathbf{T}} \mathbf{x}_{i,\text{obj}} \\ & + \rho \sum_{j \in \mathcal{N}_i} \left\| \mathbf{x}_{i,\text{obj}} - \frac{\mathbf{x}_{i,\text{obj}}^k + \mathbf{x}_{j,\text{obj}}^k}{2} \right\|_2^2 \Bigg\}\end{aligned} \quad (5)$$

with the dual variable q_i associated with the equality constraint on $\mathbf{x}_{i,\text{obj}}$ and subsequently update q_i with

$$q_i^{k+1} = q_i^k + \rho \sum_{j \in \mathcal{N}_i} \left(\mathbf{x}_{i,\text{obj}}^{k+1} - \mathbf{x}_{j,\text{obj}}^{k+1} \right) \quad (6)$$

We compute the applied forces and torques at each time interval and execute only the first control input associated with the first stage of the optimization problem in (3). For each problem instance, the robots communicate their local object trajectories only, allowing each robot to independently select control forces and torques which satisfy its dynamic constraints and other robot-specific desired behaviors. Hence, each robot does not need to exchange information relating to its local constraints. Each robot updates its applied force, torque, and object trajectory over the problem horizon using (5). The dual variable updates involve simple algebraic operations which are efficient to execute. At the beginning of the manipulation task at $t = 0$, the initial values of these optimization variables are initialized as the solution to the optimization problem given by

$$\mathbf{r}_i^0 \leftarrow \underset{\mathbf{x}_{i,\text{obj}}, \mathbf{F}_i, \mathbf{\Gamma}_i}{\text{argmin}} \max_{c_i, d_i} \left\{ J_i(\cdot) + c_i^{\mathbf{T}} g_i(\cdot) + d_i^{\mathbf{T}} h_i(\cdot) \right\} \quad (7)$$

At subsequent time instances $t > 0$, the optimization variables are initialized using the solution to the previous problem instance. This choice of initialization improves computation efficiency especially in manipulation tasks with slowly changing problem constraints as prior problem information is passed to the problem instance at the present time t.

Theorem 2. *The torques and forces applied by the robots* $\mathbf{F}_i$, $\mathbf{\Gamma}_i$ $\forall i \in \mathcal{V}$ *converge to their optimal values in* (3). *Likewise,* $\mathbf{x}_{i,obj}$ $\forall i \in \mathcal{V}$ *converges to the corresponding optimal trajectory in* (3).

Proof. The augmented Lagrangian of (3) is closed, finite, and convex. Hence, a minimizer exists for the update steps in (5) at all iterations k. As the algorithm progresses as in [1], the residuals reduce to zero, and the iterates from (5) and (6) converge to the optimal control inputs, trajectories, and dual variables. □

Remark 1. *Each robot has no explicit knowledge of the local constraints of other robots, a significantly useful feature of our method. In problems where each robot has local sensors for collision avoidance, this feature allows for fully distributed sensing without any sharing of local collision avoidance constraints.*

Algorithm 1 summarizes the distributed control scheme. The algorithm is executed for the duration of the manipulation task, and the control inputs are applied at intervals of δt. In the method *ForwardIntegrate*, we initialize the optimization variables at the last stage of the problem instance by integrating the values of these variables at the previous stage.

Algorithm 1 Scalable Optimal Collaborative Manipulation with Local Constraints (SOCM LoCo)

```
while manipulation task in progress do
    for $i = 1, \cdots, N$ do
        $(\mathbf{x}_{i,\text{obj}}, \mathbf{F}_i, \mathbf{\Gamma}_i) \leftarrow$ DistributedControl$(t, \mathbf{x}_{i,\text{obj}}(t))$
        $F_i(t + \delta t) \leftarrow \mathbf{F}_i(0)$
    end
end
```

V. Planar Manipulation with Quadratic Objective

Given the physical limitations of robots, large object manipulation requires synergy between multiple robots. The robots apply forces at different locations on the object which contribute to translation and rotation of the object shown in Figure 3. The relationship between these forces and the resulting object motion depends on the object mass and inertia properties. Here, we consider a network of N robots collaborating to manipulate the object. The robots carry the object without friction effects. However, we can incorporate the effects of friction into the dynamics of the object simply in (8). The total effect of these forces on the object's linear motion is described by the translational dynamics of the

Function DistributedControl(t,$\mathbf{x}_{i,\text{obj}}(t)$)

```
Initialization:
$q_i^0 \leftarrow 0$
$k \leftarrow 0$
$\mathbf{r}_i^k := (\mathbf{x}_{i,\text{obj}}^k, \mathbf{F}_i^k, \mathbf{\Gamma}_i^k)$
if $t = 0$ then
    $\mathbf{r}_i^0 \leftarrow \underset{\mathbf{x}_{i,\text{obj}},\mathbf{F}_i,\mathbf{\Gamma}_i}{\text{argmin}}\ \underset{c_i,d_i}{\max} \{J_i(\cdot) + c_i^{\mathbf{T}} g_i(\cdot) + d_i^{\mathbf{T}} h_i(\cdot)\}$
else
    $(\mathbf{x}_{i,\text{obj}}^{\text{prev}}, \mathbf{f}_i^{\text{prev}}, \mathbf{\Gamma}_i^{\text{prev}}) \leftarrow$ previous optimal solution
    for $\tau = 0, \cdots, N_\tau - 2$ do
        $\mathbf{x}_{i,\text{obj}}^0(\tau) \leftarrow \mathbf{x}_{i,\text{obj}}^{\text{prev}}(\tau+1)$
        $\mathbf{F}_i^0(\tau) \leftarrow \mathbf{F}_i^{\text{prev}}(\tau+1)$
        $\mathbf{\Gamma}_i^0(\tau) \leftarrow \mathbf{\Gamma}_i^{\text{prev}}(\tau+1)$
    end
    $\mathbf{F}_i^0(N_\tau - 1) \leftarrow \mathbf{F}_i^{\text{prev}}(N_\tau - 1)$
    $\mathbf{\Gamma}_i^0(N_\tau - 1) \leftarrow \mathbf{\Gamma}_i^{\text{prev}}(N_\tau - 1)$
    $\mathbf{x}_{i,\text{obj}}^0(N_\tau - 1) \leftarrow$ ForwardIntegrate($\mathbf{F}_i^0, \mathbf{\Gamma}_i^0$)
end
do in parallel $i = 1, \cdots, N$
    $\mathbf{r}_i^{k+1} \leftarrow$ Equation (5)
    $q_i^{k+1} \leftarrow$ Equation (6)
    $k \leftarrow k + 1$
while not converged or stopping criterion is not met;
return $(\mathbf{x}_{i,obj}, \mathbf{F}_i, \mathbf{\Gamma}_i)$
```

object given by

$$M\ddot{p} = \sum_{i=1}^{N} F_i \tag{8}$$

where M denotes the mass of the object, $p \in \mathbb{R}^3$ denotes the $(x,\ y,\ z)$ position of the object, and robot i applies force $F_i \in \mathbb{R}^3$ to the object.

The forces applied to the object produce angular motion about the object's center of mass since the forces have a non-zero orthogonal component to the relative position vector between the object's center of mass and the point of application of these forces. The resulting rotation is described by

$$J\dot{\omega} = \sum_{i=1}^{N} \big(r_i \times F_i\big) + \Gamma_i \tag{9}$$

with $J \in \mathbb{R}^3$ the mass moment of inertia of the object and $r_i \in \mathbb{R}^3$ the relative position of robot i in a fixed coordinate frame with the object's local coordinate frame located at its center of mass. We denote the angular acceleration of the object as $\dot{\omega} \in \mathbb{R}^3$ and its angular velocity as $\omega \in \mathbb{R}^3$. The torque $\Gamma_i \in \mathbb{R}^3$ applied by robot i contributes to the rotation of the object.

For 2D manipulation, the robots apply no force along the z axis, $F_{i,z} = 0$, $p \in \mathbb{R}^2$, and $r_i \in \mathbb{R}^2$. Likewise, the object's rotation occurs about the z-axis with $\omega \in \mathbb{R}$. The object's rotational dynamics reduces to

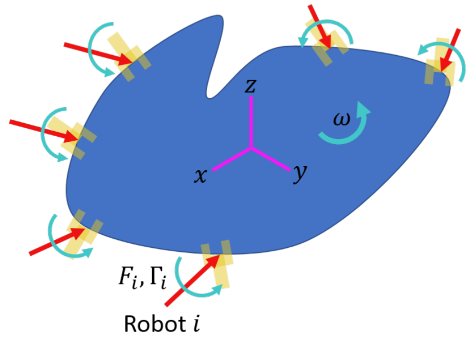

Fig. 3. Collaborative manipulation of an object by a group of robots. Each robot applies a force to the object, producing translation and rotation of the object.

$$J\dot{\omega} = \sum_{i=1}^{N} \big(\mathbf{R}r_i \times F_i\big) + \Gamma_i \tag{10}$$

where $\mathbf{R} \in \mathbb{R}^{2\times 2}$ denotes the rotation matrix transforming vectors in the local object frame to a fixed coordinate frame. We obtain $\mathbf{R}$ from

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{11}$$

and note the dependence of $\mathbf{R}$ on the current orientation of the object about the z-axis denoted by θ. As described in Section V, we formulate the manipulation task as an optimization problem and discretize the continuous-time problem using trapezoidal integration over a time interval of δt to obtain a numerical optimization problem. At each time instant t, the optimization problem over a receding horizon of N_τ stages is

$$\begin{aligned} \underset{\mathbf{p},\dot{\mathbf{p}},\boldsymbol{\theta},\boldsymbol{\omega}}{\text{minimize}} & \sum_{i=1}^{N} \psi_i(\mathbf{p}_i, \dot{\mathbf{p}}, \boldsymbol{\theta}, \boldsymbol{\omega}) \\ \text{subject to } & A_{v,i}\dot{\mathbf{p}}_i = B_{v,i}\dot{\mathbf{p}}_i + K_{v,i}\mathbf{F}_i \\ & A_{p,i}\mathbf{p}_i = B_{p,i}\mathbf{p}_i + K_{p,i}\dot{\mathbf{p}}_i \\ & A_{\omega,i}\boldsymbol{\omega}_i = B_{\omega,i}\boldsymbol{\omega}_i + K_{\omega,i}\mathbf{F}_i + K_{r,i}\mathbf{\Gamma}_i \\ & A_{\theta,i}\boldsymbol{\theta}_i = B_{\theta,i}\boldsymbol{\theta}_i + K_{\theta,i}\boldsymbol{\omega}_i \\ & \dot{\mathbf{p}}_i(0) = \dot{\bar{\mathbf{p}}}_i(t), \quad \mathbf{p}_i(0) = \bar{\mathbf{p}}_i(t) \\ & \boldsymbol{\omega}_i(0) = \bar{\boldsymbol{\omega}}_i(t), \quad \boldsymbol{\theta}_i(0) = \bar{\boldsymbol{\theta}}_i(t) \\ & \dot{\mathbf{p}}_i = \dot{\mathbf{p}}_j, \quad \mathbf{p}_i = \mathbf{p}_j \quad \forall j \in \mathcal{N}_i \\ & \boldsymbol{\omega}_i = \boldsymbol{\omega}_j, \quad \boldsymbol{\theta}_i = \boldsymbol{\theta}_j \quad \forall j \in \mathcal{N}_i \end{aligned} \tag{12}$$

We denote the vertical concatenation of p for all τ as $\mathbf{p}$ and do likewise for $\boldsymbol{\theta}$, $\boldsymbol{\omega}$, $\mathbf{F}$, and $\mathbf{\Gamma}$. The objective function $\psi_i(\cdot)$ is given by

$$\psi_i(\cdot) = \mathbf{e}_i^{\mathbf{T}}\mathbf{G}\mathbf{e}_i + \mathbf{F}_i^{\mathbf{T}}\mathbf{H}_f\mathbf{F}_i + \mathbf{\Gamma}_i^{\mathbf{T}}\mathbf{H}_r\mathbf{\Gamma}_i \tag{13}$$

where

$$\mathbf{e}_i = \begin{bmatrix} \mathbf{p}_i - \mathbf{p}_{\text{des}} \\ \dot{\mathbf{p}}_i - \dot{\mathbf{p}}_{\text{des}} \\ \boldsymbol{\theta}_i - \boldsymbol{\theta}_{\text{des}} \\ \boldsymbol{\omega}_i - \boldsymbol{\omega}_{\text{des}} \end{bmatrix}$$

and $\mathbf{p}_{\text{des}}$, $\dot{\mathbf{p}}_{\text{des}}$, $\boldsymbol{\theta}_{\text{des}}$, and $\boldsymbol{\omega}_{\text{des}}$ denote the desired object translation and rotation. We introduce the positive semi-definite matrices G_p, G_v, G_θ, and G_ω weighting the deviations in the object's position, orientation, and their derivatives, placed along the diagonal of $\mathbf{G}$. We have included the other terms in the objective function in (13) to obtain solutions that minimize actuator effort as a proxy for energy consumption and have introduced the positive definite matrices $\mathbf{H}_f$ and $\mathbf{H}_r$. In addition, we constrain the initial position, orientation, and their derivatives to the object's position, orientation and derivatives at time t which we denote as $\bar{\mathbf{p}}_i(t)$, $\dot{\bar{\mathbf{p}}}_i(t)$, $\bar{\boldsymbol{\theta}}_i(t)$, and $\bar{\boldsymbol{\omega}}_i(t)$. Subsequently, we denote $\bar{\mathbf{p}}_i(t)$, $\dot{\bar{\mathbf{p}}}_i(t)$, $\bar{\boldsymbol{\theta}}_i(t)$, and $\bar{\boldsymbol{\omega}}_i(t)$ as $\bar{\mathbf{p}}_i$, $\dot{\bar{\mathbf{p}}}_i$, $\bar{\boldsymbol{\theta}}_i$, and $\bar{\boldsymbol{\omega}}_i$ respectively.

We describe the translation dynamics of the object in (12) with $A_{v,i}$, $A_{p,i}$, $B_{v,i}$, $B_{p,i}$, $K_{v,i}$, and $K_{p,i}$ and the rotational dynamics of the object with $A_{\omega,i}$, $A_{\theta,i}$, $B_{\omega,i}$, $B_{\theta,i}$, $K_{\omega,i}$, $K_{\theta,i}$, and $K_{r,i}$. The object's angular acceleration depends on the rotation matrix $\mathbf{R}(\theta)$, a function of the object's orientation (10). Hence, the rotational dynamics constraints are non-convex as $K_{\omega,i}(\theta)$ depends on the object's orientation.

In general, distributed optimization methods have no convergence guarantees for non-convex optimization problems. Nonetheless, we linearize $K_{\omega,i}$ about the object's desired orientation at $t = 0$ and about the previous solution of the optimization problem at all other time which reduces the problem to a convex optimization problem. We provide the resulting algorithm in Algorithm 2. Following the procedure in Section IV, we distribute the problem among the robots. Robot i computes its control inputs, object's position, orientation, and dual variables from the system of linear equations (5):

$$\Lambda_i^{k+1}\mathbf{a}^{k+1} = \mathbf{d}^{k+1} \tag{14}$$

where $\mathbf{a} = (\dot{\mathbf{p}}_i,\ \mathbf{p}_i,\ \dot{\boldsymbol{\theta}}_i,\ \boldsymbol{\theta}_i,\ \mathbf{F}_i,\ \boldsymbol{\Gamma}_i,\ \mathbf{c}_i)$, $\Lambda_i \in \mathbb{R}^{12N_\tau \times 12N_\tau}$ denotes the Hessian of the optimization problem in (5), $\mathbf{c}_i$ represents the dual variables of the constraints in (12), and $\mathbf{d}$ incorporates information on the solution of other robots in $\mathcal{N}_i$, ensuring that all robots arrive at the same position and orientation along which to manipulate the object.

Algorithm 2 Manipulation with Non-Convex Dynamics

```
while manipulation task in progress do
    for $i = 1, \cdots, N$ do
        $K_{\omega,i} \leftarrow$ Linearize rotation dynamics (10) at $\mathbf{x}_{i,\text{obj}}^{\text{prev}}$
        $(\mathbf{x}_{i,\text{obj}}, \mathbf{F}_i, \boldsymbol{\Gamma}_i) \leftarrow$ DistributedControl$(t, \mathbf{x}_{i,\text{obj}}(t))$
        $F_i(t + \delta t) \leftarrow \mathbf{F}_i(0)$
    end
end
```

VI. Simulations

Now, we evaluate the Scalable Optimal Collaborative Manipulation with Local Constraints (SOCM LoCo) method in a manipulation task with a quadratic tracking objective on the desired object trajectory and control inputs for each robot as described in Section V. We examine the communication and computation resources required by our method in comparison to a leader-follower control architecture where the leader computes the control inputs for the entire group and consensus ADMM methods [21]. In addition, we evaluate the performance of our approach across different communication constraints represented by the communication graph $\mathcal{G}$. We consider a group of $N = 100$ robots collaborating to manipulate an object along the desired trajectory.

A. Computation Complexity

At time t, each robot solves the optimization problem in (12) for its control inputs iteratively. The update procedure for robot i's control inputs requires solving the linear system of equations given in (14). This linear system can be solved efficiently by factoring Λ_i. In the SOCM LoCo method, factoring $\Lambda_i \in \mathbb{R}^{14N_\tau \times 14N_\tau}$ requires $O(N_\tau^3)$ floating-point operations (flops). A subsequent back-solve step to obtain the control inputs incurs a cost of $O(N_\tau^2)$ flops. Likewise, the dual update procedure incurs a cost of $O(N_\tau)$ flops. Hence, the SOCM LoCo method requires a computation cost of $O(N_\tau^3)$ where we have ignored the less dominant terms in the computation cost.

As described in Section II, leader-follower control schemes require the designation of a leader which computes the control inputs for all robots in the team. Each robot's control input is obtained by solving a linear system of equations similar to (14) with a positive definite Hessian matrix $H \in \mathbb{R}^{(N+12)N_\tau \times (N+12)N_\tau}$. We can factor H with $O(N^3N_\tau^3)$ flops and solve for the control inputs with $O(N^2N_\tau^2)$ flops. Consequently, leader-follower methods require $O(N^3N_\tau^3)$ flops with the less dominant terms ignored.

In consensus ADMM methods, each robot computes the control inputs for the entire group. Factoring the Hessian and solving for all the control inputs takes $O(N^3N_\tau^3)$ and $O(N^2N_\tau^2)$ respectively. Thus, consensus ADMM methods have the same complexity as leader-follower methods with $O(N^3N_\tau^3)$ flops.

The computation complexity of both leader-follower and consensus ADMM methods scale cubically with the number of robots involved in the manipulation task. As a result, these methods require significant computation resources for manipulation tasks with large groups of robots. The SOCM LoCo method reduces computation complexity of leader-follower and consensus ADMM methods by a factor of N^3. Notably, complexity of the SOCM LoCo method is independent of the number of robots. As such, performance of SOCM LoCo does not degrade with large groups of robots, unlike leader-follower and consensus ADMM methods. The lower complexity of SOCM LoCo enhances its performance in object manipulation problems which require high frequency control inputs.

B. Communication Complexity

In the SOCM LoCo method, each robot shares its local copy of the object trajectory over N_τ stages with its neighbors defined in $\mathcal{G}$. The object trajectory can be represented with $48N_\tau$ bytes in double precision floating-point format. As such, $O(N_\tau)$ bytes of information flows over the peer-to-peer network between each robot and its neighbors at each communication round.

In leader-follower methods, the leader robot computes and shares the control inputs for the entire team. Each communication round involves sharing $O(NN_\tau)$ bytes of information. Likewise, each communication round in consensus ADMM methods takes $O(NN_\tau)$ bytes per robot. The amount of communication scales linearly with the size of the group, presenting a challenge with large groups. However, the SOCM LoCo method provides about a factor of N lower communication complexity, constant for all group sizes, which allows the robots to collaborate on the manipulation task with minimal communication.

C. Communication Constraints and Convergence

We examine the SOCM LoCo method in manipulating an object along the desired trajectory shown in Figure 4 with 20 robots. We note that the desired object trajectory is not required to be dynamically feasible for the group of robots; hence, perfect tracking is not always achievable. Larger tracking errors arise in infeasible regions of the desired trajectory. Moving the object to its desired final position requires rotating the object to enable it to fit through narrow passages, shown in the provided video. Figure 5 shows the tracking errors in the position and orientation of the object during the task. With the SOCM LoCo method, the position tracking error remains less than 10^{-5} m^2 with its orientation error less than 10^{-4} rad^2 as the object follows its desired trajectory.

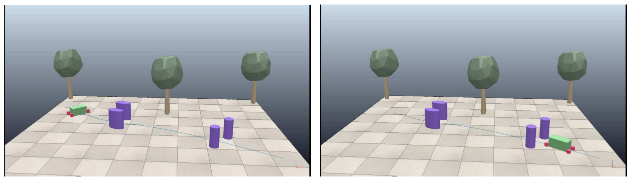
Fig. 4. A group of 20 robots (in red) manipulate an object (in green) to a desired final position and orientation. The robots rotate the object to enable it to fit through narrow passages. Three robots are shown for visualization purposes.

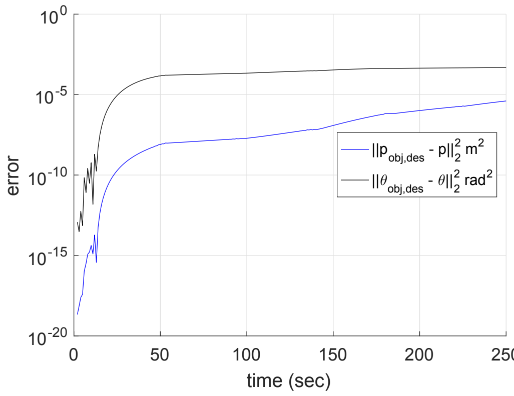

Fig. 5. With the SOCM LoCo method, the robots manipulate an object along a desired trajectory with small tracking errors in the object's position and orientation.

Next, we evaluate the performance of the SOCM LoCo method on fully-connected communication graphs, randomly generated connected graphs, and chain graphs (the least-connected acyclic graph with each robot having at most two neighbors). We examined the consensus ADMM method on a fully-connected graph. We summarize the examined methods and associated communication constraints in Table I.

We examine the tracking error on the desired object trajectory as the number of collaborating robots increases. Figure 6 shows the mean tracking error of each method for different number of robots. The SOCM LoCo method outperforms consensus ADMM methods by almost two orders of magnitude in tracking the desired trajectory. The tracking performance of consensus ADMM methods degrade with the larger groups of robots as the size of each agent's optimization variable increases. However, the tracking error in the SOCM LoCo method remains relatively constant for groups greater than 40 robots. Leader-follower methods require more stringent communication conditions as all follower robots must communicate with the designated leader robot. This demand makes leader-follower approaches unsuitable in environments with limited communication infrastructure which are prevalent in robotics. In contrast, the SOCM LoCo method provides desirable performance with small tracking errors within 10^{-5} m^2 in these environments even in extreme cases with chain communication graphs for all group sizes.

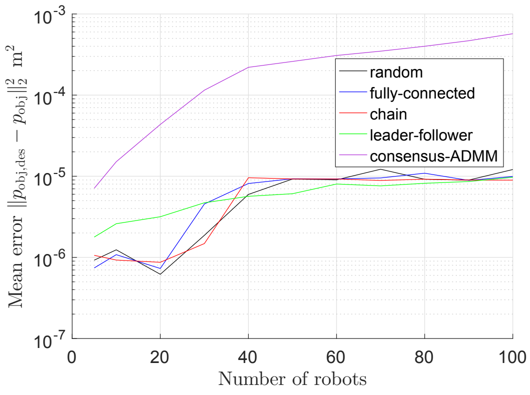

Fig. 6. Mean tracking error on the desired manipulated object trajectory for different number of collaborating robots. SOCM LoCo attains smaller tracking errors by two orders of magnitude compared to consensus ADMM and provides comparable tracking performance with the leader-follower method across different communication graphs.

TABLE I
EVALUATED METHODS AND COMMUNICATION CONSTRAINTS

Name	Method	Communication Graph
random	SOCM LoCo	randomly-connected
fully-connected	SOCM LoCo	fully-connected
chain	SOCM LoCo	chain
leader-follower	leader-follower	all followers to leader
consensus-ADMM	consensus-ADMM	fully-connected

Further, we examine the convergence rate of SOCM LoCo and consensus ADMM methods for a manipulation task with

100 robots. SOCM LoCo converges within 600 iterations and attains a significantly lower tracking error less than 10^{-3} m^2 as shown in Figure 7. In contrast, consensus ADMM fails to attain the same magnitude of error within 1200 iterations. The large number of iterations required for convergence in consensus ADMM method hinders its application to receding horizon problems where an optimization problem is solved at every timestep.

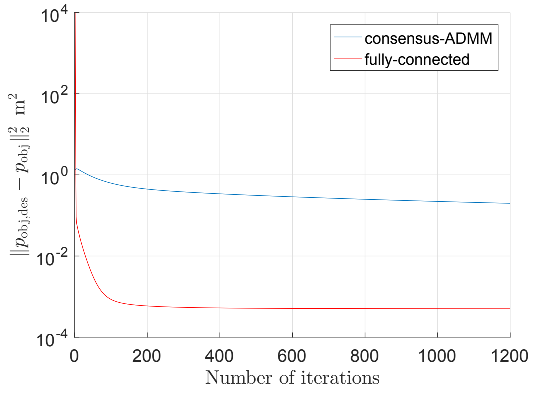

Fig. 7. The SOCM LoCo method provides faster convergence compared to consensus ADMM methods with $N = 100$ robots, and SOCM LoCo converges to a significantly lower tracking error less than 10^{-3} m^2 within a few hundred iterations. Consensus ADMM methods require much greater number of iterations for convergence to the same error magnitude.

VII. Conclusion

Object manipulation requires the collaboration of multiple robots in many situations such as autonomous construction. In this work, we develop the SOCM LoCo method for optimal collaborative manipulation by robots without a fully-connected communication graph or a designated leader. SOCM LoCo enables large groups of robots to jointly manipulate objects with minimal communication, requiring no communication of each robot's planned trajectory and control inputs. The complexity of our method is independent of the number of robots, leading to superior improvements in computation and communication complexity compared to earlier distributed control schemes. Future work will demonstrate the application of the SOCM LoCo method in manipulation tasks in $SE(3)$ space with non-convex objective functions and constraints.

References

[1] O. Shorinwa, T. Halsted, and M. Schwager, "Scalable distributed optimization with separable variables in multi-agent networks," in *Proceedings of the 2020 American Control Conference*, 2020.

[2] B. Hichri, L. Adouane, J.-C. Fauroux, Y. Mezouar, and I. Doroftei, "Cooperative mobile robot control architecture for lifting and transportation of any shape payload," in *Distributed Autonomous Robotic Systems*. Springer, 2016, pp. 177–191.

[3] A. Z. Bais, S. Erhart, L. Zaccarian, and S. Hirche, "Dynamic load distribution in cooperative manipulation tasks," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2380–2385.

[4] D. Ortenzi, R. Muthusamy, A. Freddi, A. Monteriù, and V. Kyrki, "Dual-arm cooperative manipulation under joint limit constraints," *Robotics and Autonomous Systems*, vol. 99, pp. 110–120, 2018.

[5] W. Wan, R. Fukui, M. Shimosaka, T. Sato, and Y. Kuniyoshi, "Cooperative manipulation with least number of robots via robust caging," in *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2012, pp. 896–903.

[6] Y. Kobayashi and S. Hosoe, "Cooperative enclosing and grasping of an object by decentralized mobile robots using local observation," *International Journal of Social Robotics*, vol. 4, no. 1, pp. 19–32, 2012.

[7] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[8] G. Montemayor and J. T. Wen, "Decentralized collaborative load transport by multiple robots," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 372–377.

[9] A. Petitti, A. Franchi, D. Di Paola, and A. Rizzo, "Decentralized motion control for cooperative manipulation with a team of networked mobile manipulators," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 441–446.

[10] I. Mas and C. Kitts, "Object manipulation using cooperative mobile multi-robot systems," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, 2012.

[11] Z. Wang and M. Schwager, "Force-amplifying n-robot transport system (force-ants) for cooperative planar manipulation without communication," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1564–1586, 2016.

[12] S. G. Faal, S. T. Kalat, and C. D. Onal, "Towards collective manipulation without inter-agent communication," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016, pp. 275–280.

[13] M. Bernard, K. Kondak, I. Maza, and A. Ollero, "Autonomous transportation and deployment with aerial robots for search and rescue missions," *Journal of Field Robotics*, vol. 28, no. 6, pp. 914–931, 2011.

[14] A. Tagliabue, M. Kamel, R. Siegwart, and J. Nieto, "Robust collaborative object transportation using multiple mavs," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1020–1044, 2019.

[15] A. Tsiamis, C. K. Verginis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Cooperative manipulation exploiting only implicit communication," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 864–869.

[16] A. Marino and F. Pierri, "A two stage approach for distributed cooperative manipulation of an unknown object without explicit communication and unknown number of robots," *Robotics and Autonomous Systems*, vol. 103, pp. 122–133, 2018.

[17] P. Culbertson and M. Schwager, "Decentralized adaptive control for collaborative manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 278–285.

[18] Y. Kume, Y. Hirata, Z.-D. Wang, and K. Kosuge, "Decentralized control of multiple mobile manipulators handling a single object in coordination," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2758–2763.

[19] S. Erhart, D. Sieber, and S. Hirche, "An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 315–322.

[20] M. Annergren, A. Hansson, and B. Wahlberg, "An admm algorithm for solving ℓ_1 regularized mpc," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 4486–4491.

[21] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010.