Scalable Cooperative Transport of Cable-Suspended Loads With UAVs Using Distributed Trajectory Optimization

Brian E. Jackson, Taylor A. Howell, Kunal Shah, Mac Schwager, and Zachary Manchester

***Abstract*—Most approaches to multi-robot control either rely on local decentralized control policies that scale well in the number of agents, or on centralized methods that can handle constraints and produce rich system-level behavior, but are typically computationally expensive and scale poorly in the number of agents, relegating them to offline planning. This work presents a scalable approach that uses distributed trajectory optimization to parallelize computation over a group of computationally-limited agents while handling general nonlinear dynamics and non-convex constraints. The approach, including near-real-time onboard trajectory generation, is demonstrated in hardware on a cable-suspended load problem with a team of quadrotors automatically reconfiguring to transport a heavy load through a doorway.**

***Index Terms*—Path planning for multiple mobile robots or agents, optimization and optimal control.**

I. Introduction

MANY robotic tasks can be accomplished by a single agent, but often there are benefits to employing a team of robots. For example, transporting a heavy load can be accomplished by deploying a single powerful, expensive, and potentially dangerous aerial vehicle, or by deploying a group of smaller aerial vehicles that cooperatively transport the load. The potential benefits of a team approach, namely reduced cost, increased versatility, safety, and deployability of the system, are important but come at the cost of increased complexity. Motion planning for a single robot is already a challenging task, and coordinating a group of agents to cooperatively accomplish a task can be significantly more complicated due to increased degrees of freedom, more constraints, and the requirement to reason about the effects of distributed computation and communication.

Manuscript received September 10, 2019; accepted February 7, 2020. Date of publication February 24, 2020; date of current version March 9, 2020. This letter was recommended for publication by Associate Editor D. Lee and Editor Nak Young Chong upon evaluation of the reviewers' comments. This work was supported in part by NASA Early Career Faculty Award under Grant 80NSSC18K1503 in part by the Stanford-Ford Alliance, and in part by the National Science Foundation Graduate Research Fellowship Program under Grant DGE-1656518. *(B. E. Jackson and T. A. Howell contributed equally to this work.) (Corresponding author: Taylor Howell.)*

Brian E. Jackson, Taylor A. Howell, and Kunal Shah are with the Department of Mechanical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: bjack205@stanford.edu; thowell@stanford.edu; K2shah@stanford.edu).

Mac Schwager and Zachary Manchester are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA (e-mail: schwager@stanford.edu; zacmanchester@stanford.edu).

Digital Object Identifier 10.1109/LRA.2020.2975956

Approaches for multi-agent motion planning range from decentralized methods that consider local interactions of the agents to centralized methods that globally coordinate a system of agents. The decentralized approach has many advantages, such as robustness to agent removal or failure, scalability, computational efficiency and, possibly, reduced communication requirements among the network of agents. While decentralized approaches can achieve useful and interesting global behavior [1]–[5], they have important limitations. Often, they are limited to simple, homogeneous dynamics, such as single or double integrators. It is also difficult to enforce constraints, either at the agent or the system level.

Centralized approaches, in contrast, are able to reason about general dynamics and constraints. However, they typically necessitate increased computation and communication, and require solving large optimization problems. Centralized methods are, therefore, typically run offline. Additionally, since the optimization problems involved often exhibit cubic or exponential scaling of computation time with the number of decision variables, centralized approaches tend to scale poorly for systems with a large number of agents. Both centralized and decentralized approaches have been used in a large variety of fields, from controlling swarms of microscale robots [6], to modeling human crowds [7], to planning motion for teams of aerial vehicles transporting heavy loads.

Aerial vehicles have been used to transport slung loads since at least the 1960s [8] for applications including: delivering fire retardant to fight forest fires, carrying beams for civil infrastructure projects, moving harvested trees, carrying military vehicles, and transporting large animals. Today, quadrotors have become a standard testbed for such aerial vehicle applications, and there is an extensive literature on utilizing a single quadrotor to carry a slung load [9]–[12].

Teams of quadrotors have also been explored, oftentimes making significant simplifying assumptions. The most common simplification, and one also taken in the current work, is to model the cables as massless rigid links [13]–[17]. More recent approaches have relaxed this assumption by modeling the system as a hybrid dynamical system and then solving the problem as a mixed-integer program that takes nearly an hour to solve [10]. While most assume a simple point mass for the load, some also consider extensions to rigid bodies [13], [17]. A distributed controller for a group of quadrotors rigidly attached to a load, assuming no communication between agents [18] has also been developed.

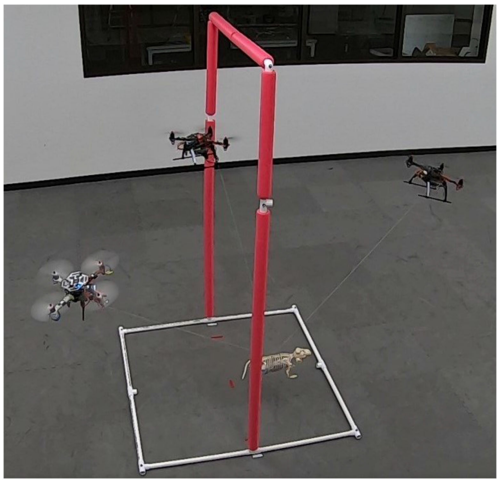


Fig. 1. Hardware experiment with a team of 3 quadrotors carrying a heavy load that a single quadrotor cannot lift. The team must reconfigure to proceed through the narrow doorway.

Beyond the suspended-load problem, collision-free trajectories for a swarm of quadrotors can be calculated using sequential convex programming, but the computational complexity scales exponentialy with the number of agents [19]. Other notable examples of coordinated teams of quadrotors include throwing and catching a ball with quadrotors connected by a net [20], performing a treasure hunt [21], and manipulating flexible payloads [22]. Distributed approaches for motion planning have been proposed using alternating direction method of multipliers (ADMM) and mixed integer programs [23]–[26], but they utilize simplified dynamics or constraints, and don't consider interaction between agents (e.g., forces).

In this work, we present a scalable distributed approach for obtaining a solution to the centralized "batch" motion-planning problem for a team of quadrotors carrying a cable-suspended load and demonstrate the algorithm in hardware (Fig. 1). This approach combines the generality of centralized methods, while approaching near real-time performance that would be impossible without distributed, parallel computation. We formulate a trajectory optimization problem and consider the nonlinear rigid body dynamics of the quadrotors and non-convex collision and obstacle avoidance constraints. We use the trajectory optimization solver ALTRO [27] to find the state and control trajectories for the system of agents. To make the problem scalable in the number of agents, we take an intuitive approach of decomposing the problem by agent and solving the resulting subproblems in parallel. The result is a substantial reduction in solution time and superior scaling as the number of agents is increased. The novelty of this approach lies in the use of a trajectory optimization solver (in our case, ALTRO) to solve a sequence of smaller constrained problems, that include optimizing interactions (i.e., forces) between agents, instead of solving a single large trajectory optimization problem.

The remainder of this letter is organized as follows: Section II formulates the cable-suspended-load problem as a trajectory optimization problem. Section III presents a decomposition scheme and an algorithm for solving the batch problem in parallel among agents. Section IV contains simulation results and Section V contains details of the hardware experiments. Finally, we conclude with a discussion of the algorithm and results in Section VI.

II. Batch Problem

We formulate a trajectory optimization problem for the team cable-suspended load problem with L quadrotors attached to a point-mass load. The suspension cables are assumed to pass through the center of mass of each quadrotor, and are modeled as massless rigid links. The dynamics model formulation of the system is critical for achieving good performance using trajectory optimization, and is described in detail in Section II-A. Section II-B then describes the optimization problem.

A. Dynamics

1) Quadrotor Model With Quaternions: The quadrotor dynamics presented in [28] are modified to use quaternions for angular representation and incorporate the force generated by a suspension cable:

$$\begin{aligned}\dot{x} &= \begin{bmatrix}\dot{r}\\ \dot{q}\\ \dot{v}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix} v \\ \frac{1}{2} q \otimes \hat{\omega} \\ g + \frac{1}{m^i}(R(q)F(u) + F_c(u_5, x, x^\ell)) \\ J^{-1}(\tau(u) - \omega \times J\omega)\end{bmatrix} \\ &= f(x, u; x^\ell) \end{aligned} \tag{1}$$

where $r \in \mathbb{R}^3$ is the position, q is a unit quaternion, $R(q) \in \mathbb{SO}(3)$ is a quaternion-dependent rotation matrix from body frame to world frame, $v \in \mathbb{R}^3$ is the linear velocity in the world frame, $\omega \in \mathbb{R}^3$ is the angular velocity in the body frame, $x \in \mathbb{R}^{13}$ is the state vector, $u \in \mathbb{R}^5$ is the control vector with the last component being the magnitude of the cable force, $x^\ell \in \mathbb{R}^6$ is the state vector of the load (defined below), $g \in \mathbb{R}^3$ is the gravity vector, and $m^i \in \mathbb{R}$ is the mass of the i-th quadrotor, $J \in \mathbb{S}^3$ is the moment of inertia tensor, $q_2 \otimes q_1$ denotes quaternion multiplication, and $\hat{\omega}$ denotes a quaternion with zero scalar part, and ω vector part. Quaternions are used to easily allow for large angular displacements and extensions to more aggressive maneuvers. The forces and torques F, $\tau \in \mathbb{R}^3$ in the body frame are

$$F(u) = \begin{bmatrix} 0 \\ 0 \\ k_f(u_1 + u_2 + u_3 + u_4) \end{bmatrix} \tag{2}$$

$$\tau(u) = \begin{bmatrix} k_f d_{\text{motor}}(u_2 - u_4) \\ k_f d_{\text{motor}}(u_3 - u_1) \\ k_m(u_1 - u_2 + u_3 - u_4) \end{bmatrix} \tag{3}$$

where k_f, k_m are motor constants, d_{motor} is the distance between motors, and $u_{1:4}$ are motor thrusts. Forces from the cables, modeled in the world frame, are calculated as:

$$F_c(\gamma, x, x^\ell) = \gamma \frac{r^\ell - r}{\|r^\ell - r\|_2}, \tag{4}$$

where $\gamma \in \mathbb{R}$ (u_5 for each quadrotor) is the magnitude of the tension in the cable and r^ℓ is the three-dimensional position of the load.

2) Load: The dynamics of a load being transported by L quadrotors are:

$$\begin{aligned}\dot{x}^\ell = \begin{bmatrix}\dot{r}^\ell \\ \dot{v}^\ell\end{bmatrix} &= \begin{bmatrix} v^\ell \\ g + \frac{1}{m^\ell} F^\ell(x^\ell, u^\ell, x^1, \ldots, x^L)\end{bmatrix} \\ &= f^\ell(x^\ell, u^\ell; x^1, \ldots, x^L)\end{aligned} \tag{5}$$

where r^ℓ is the three-dimensional position, v^ℓ is the linear velocity in the world frame, m^ℓ is the mass of the load, $x^\ell \in \mathbb{R}^6$ is the state vector, $u^\ell \in \mathbb{R}^L$ is the force acting on the load (not a direct control input), x^i is the state vector of quadrotor i, and

$$F^\ell(x^\ell, u^\ell, x^1, \ldots, x^L) = -\sum_{i=1}^{L} F_c(u_i^\ell, x^i, x^\ell). \tag{6}$$

B. Optimization Problem

The batch problem is formulated by concatenating the states and controls of L quadrotors and the load:

$$\bar{x} \in \mathbb{R}^{13L+6} = \begin{bmatrix} x^1 \\ \vdots \\ x^L \\ x^\ell \end{bmatrix}, \quad \bar{u} \in \mathbb{R}^{5L+L} = \begin{bmatrix} u^1 \\ \vdots \\ u^L \\ u^\ell \end{bmatrix}. \tag{7}$$

where $\mathcal{I}_L = \{1, \ldots, L\}$ are the indices of the quadrotors and $\mathcal{I}_A = \{1, \ldots, L, \ell\}$ are the indices of all the agents (including the load). We can pose the team cable-suspended-load problem as a single trajectory optimization problem:

$$\underset{\mathbf{X}, \mathbf{U}}{\text{minimize}} \; J^\ell(X^\ell, U^\ell) + \sum_{i=1}^{L} J^i(X^i, U^i) \tag{8a}$$
$$x_{k+1}^i = f_k^i(x_k^i, u_k^i, \Delta t; x_k^\ell), \quad \forall i \in \mathcal{I}_L \tag{8b}$$
$$x_{k+1}^\ell = f_k^\ell(x_k^\ell, u_k^\ell, \Delta t; x_k^1, \ldots, x_k^L) \tag{8c}$$
$$x_0^i = x(0)^i, \quad \forall i \in \mathcal{I}_A \tag{8d}$$
$$x_N^\ell = x(t_f)^\ell \tag{8e}$$
$$r_{\text{min}}^i \leq r_k^i \leq r_{\text{max}}^i, \quad \forall i \in \mathcal{I}_A \tag{8f}$$
$$0 \leq (u_k^i)_j \leq u_{\text{max}}^i, \quad \forall i \in \mathcal{I}_L, j \in \{1, \ldots, 4\} \tag{8g}$$
$$u_k^\ell \geq 0 \tag{8h}$$
$$(u_k^i)_5 = (u_k^\ell)_i, \quad \forall i \in \mathcal{I}_L \tag{8i}$$
$$\|r_k^i - r_k^\ell\|_2 = d_{\text{cable}}, \quad \forall i \in \mathcal{I}_L \tag{8j}$$
$$2d_{\text{quad}} - \|p_k^i - p_k^j\|_2 \leq 0, \quad \forall i, j \in \mathcal{I}_L, i \neq j \tag{8k}$$
$$d_{\text{quad}} + d_{\text{obs}} - \|p_k^i - p_{\text{obs}}^j\|_2 \leq 0, \quad \forall i \in \mathcal{I}_L, \; \forall j \tag{8l}$$
$$d_{\text{load}} + d_{\text{obs}} - \|p_k^\ell - p_{\text{obs}}^j\|_2 \leq 0, \quad \forall j, \tag{8m}$$

where $X = [x_0, \ldots, x_N]$ is a state trajectory of length N, $U = [u_0, \ldots, u_{N-1}]$ is a control trajectory of length $N - 1$, $\mathbf{X} = [X^1, \ldots, X^L, X^\ell]$ and $\mathbf{U} = [U^1, \ldots, U^L, U^\ell]$ are sets of trajectories for the system, $p^i \in \mathbb{R}^2$ is the two-dimensional position of the quadrotor or load (discarding height), $x(0)$ and $x(t_f)$ are initial and final conditions, d is a scalar dimension (e.g., quadrotor radius), Δt is the time step duration, and all constraints apply at each time steps k.

The constraints are, from top to bottom: discrete quadrotor dynamics (8b) from (1), discrete load dynamics (8c) from (5), initial conditions (8d), final condition for the load (8e), workspace constraints (i.e., floor and ceiling constraints) (8f), quadrotor motor constraints (8g), positive cable tension (8h), equal tension force on quadrotor and load (8i), cable length (8j), collision avoidance (8k), and obstacle avoidance for the quadrotors (8l) and load (8m). The obstacles are modeled as cylinders of infinite height, which reduces collision checking to a plane. These simple constraints can be combined to form narrow doorways or slots and enables fast, analytical collision checking. The self-collision constraints also model the quadrotors as cylinders of infinite height to help prevent unmodeled prop wash effects from disturbing the system. The objective for each agent was a quadratic cost, having the form: $(x_k - x_{\text{ref}})^T Q_k (x_k - x_{\text{ref}})$ and $(u_k - u_{\text{ref}})^T R_k (u_k - u_{\text{ref}})$ for the states and controls, respectively, at each time step k.

We found it beneficial to both initialize the solver and include in the objective state and control references: x_{ref}, u_{ref}, based on trim conditions that produce static hovering of the system. These conditions were found using trajectory optimization by setting the initial and final positions and velocities of the system to be the same. Large costs were used for all states except the orientation, which had a relatively small cost. Solving this problem, the system naturally finds an equilibrium state where the quadrotors "lean away" from the load in order to create thrust in a direction that compensates for the tension in the cable (this behavior can be seen in Fig. 1). These quadrotor orientations are then used for the initial and final orientations and the trim controls are used as reference controls. Regularizing the control values to these trim controls, rather than to zero, had a significant effect on the convergence of the optimization. This hover condition was also used as the initial control trajectory provided to ALTRO.

III. Distributed Formulation

We present a scalable approach for solving (8a) by decomposing the problem by quadrotor. Of the given constraints, only the collision avoidance constraint (8k) directly couples the states of the quadrotors (i.e., only the collision avoidance constraints are functions of the states of different quadrotors). The quadrotors' dynamics are only indirectly coupled (through the load) via the cable constraints (8i) and (8j). The objective, by design, is separable by agent, i.e., there is no cost coupling between the state and controls of the quadrotors or the quadrotors and load.

From these observations, a decomposition is quite apparent: solve a trajectory optimization problem for each quadrotor independently, treating all other quadrotor and load trajectories as static. After each quadrotor has optimized independently, the updated trajectories are collected and used to optimize the load

Algorithm 1: Distributed Trajectory Optimization.

```
1:  function DIST-TRAJ-OPT($\mathbf{X}_0, \mathbf{U}_0$, tol.)
2:      $\tilde{\mathbf{X}} \leftarrow \mathbf{X}_0$, $\tilde{\mathbf{U}} \leftarrow \mathbf{U}_0$
3:      while MAX-VIOLATION($\tilde{\mathbf{X}}, \tilde{\mathbf{U}}$) $>$ tol. do
4:          for $i = 1, \ldots, L$ do in parallel
5:              $X^i, U^i \leftarrow$ SOLVE-QUAD($X^i, U^i; \tilde{\mathbf{X}}, \tilde{\mathbf{U}}$)
6:              send $X^i, U^i$ to central agent
7:          end for
8:          $X^\ell, U^\ell \leftarrow$ SOLVE-LOAD($X^\ell, U^\ell; \tilde{\mathbf{X}}, \tilde{\mathbf{U}}$)
9:          send $\tilde{\mathbf{X}}, \tilde{\mathbf{U}}$ to all agents
10:     end while
11: return $\tilde{\mathbf{X}}, \tilde{\mathbf{U}}$
12: end Function
```

trajectory. The updated load trajectory, along with the updated quadrotor trajectories, are then communicated to each quadrotor and the process is repeated.

While this update for the load trajectory can be considered a consensus update, it is distinct from approaches like ADMM that perform primal updates over agents followed by a collective dual update. In our approach, each agent performs primal and dual updates until convergence before communicating. In practice, we found this approach to converge faster and more reliably. Our procedure is summarized in Algorithm 1.

The "central" agent (the one that computes the load trajectory) is, in practice, randomly assigned before the solve to one of the quadrotors, and is computed in a separate process with a separate memory space.

As with most nonlinear optimization algorithms, the performance of our algorithm improves dramatically with a good initial guess. We designed our guess by solving an initial set of trajectory optimization problems for each quadrotor and the load separately, without any of the system-level constraints (i.e., self-collision (8k) or cable constraints (8i) and (8j)). Since these problems are completely de-coupled, they can be solved in parallel. Getting these initial trajectories to be sufficiently close to the final solution was key in getting the fast convergence demonstrated in the results that follow. The time to solve these initial trajectory optimization problems is included in the overall solution time. These optimizations were initialized with the same trim conditions given to the batch problem, described previously. However, for these initial optimization problems, the initial control and control reference for the cable tension was set to zero, since the cable constraints weren't considered during these problems. This resulted in slightly different initializations for the batch and distributed algorithms.

IV. Simulations

A. Scenarios

Three scenarios using the batch problem formulation are considered: 1) point-to-point transfer of a load using L quadrotors (see Fig. 2), 2) a load transfer through a narrow slot, followed by a narrow doorway, which requires the quadrotors to spread apart and then gather together (see Fig. 4), and 3) a load transfer through a narrow doorway using 3 quadrotors (see Fig. 6). All scenarios solve a 10 s trajectory. We assume perfect knowledge of the obstacles.

In the first and second scenarios, the stage costs were identical for all time steps. The costs for the second and third scenarios were identical; however, in the third scenario, the cost was altered at the middle time step $k_m = (N-1)/2$ to encourage the quadrotors to "line up" when passing through the door. This stage cost simply placed a high cost (about equal in magnitude to the cost at the terminal time step) for deviations from an intermediate configuration for passing through the doorway. The desired positions were calculated directly from the initial position of the load and the location of the center of the door. The load was assumed to be at the center of the door, at the same height it started. The quadrotors are then evenly distributed on an arc of α degrees. This encourages the quadrotors to "fan" out around the load, see Fig. 1).

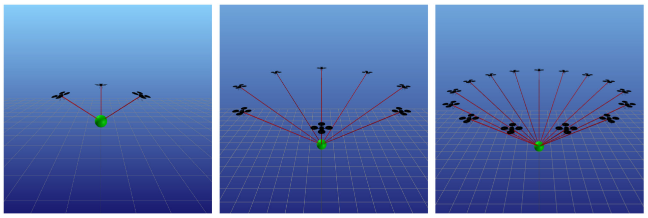


Fig. 2. Simulation of teams with 3, 8, and 15 quadrotors (left, center, right) in final configuration after a point-to-point load transfer. To maintain the final system configuration, quadrotors orient to produce thrust that maintains hover despite a force resulting from the load.

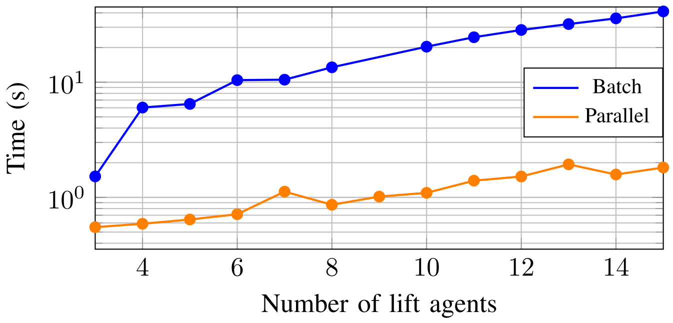


Fig. 3. Timing result comparing batch and parallel algorithms for a point-to-point load transfer using L quadrotors. The parallel algorithm scales favorably compared to the batch approach as the number of quadrotors is increased.

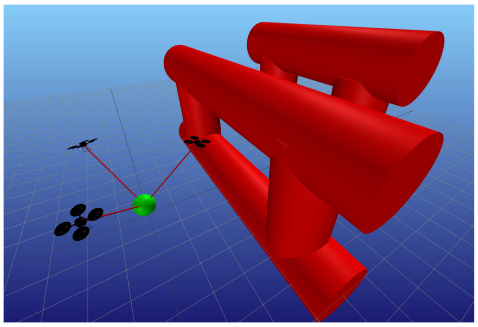
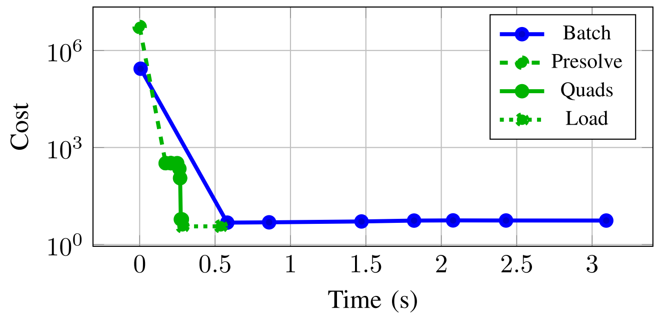
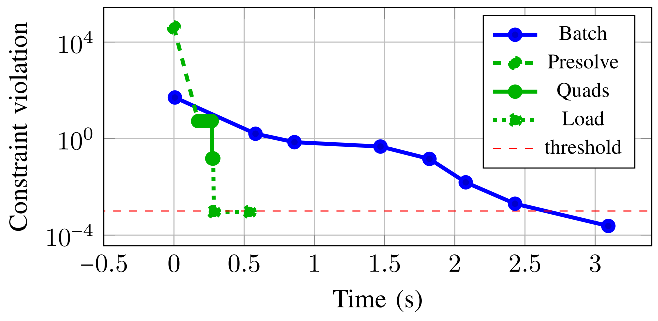


Fig. 4. Slot scenario. The quadrotors have to automatically reconfigure to pass through a narrow horizontal slot, followed by a narrow doorway.



(a) Cost convergence



(b) Constraint convergence

Fig. 5. Convergence comparison of covergence of the cost function (a) and constraint violation (b) for the batch and distributed solves. The distributed solve is broken into three phases: 1) Presolve, where an initial trajectory for each agent is obtained by solving each problem independently, 2) Quads, where each quadrotor solves it's own problem in parallel, and 3) Load, where the trajectory for the load is solved using the updated quadrotor trajectories.

The dimensions of the quadrotor, load cables, and doorway were based on the actual values for the hardware experiment (see Section V). In simulation, the slot and doorways are modeled using horizontally and vertically oriented cylinder obstacle constraints. The doorway in Fig. 6 was constructed from two cylinders 2 m apart with a radius of 0.5 m, 0.6 m smaller than the actual doorway (1.0 m vs 1.6 m). The slot scenario is similarly dimensioned. To encourage the quadrotors to get in position before reaching the door, we made the doorway artificially "deep" by adding a second set of cylinders. The width of the quadrotor was set to 0.55 m. This scenario is challenging since, although a single quadrotor can fit through the doorway easily, two cannot. The quadrotors must coordinate a way to carry the load through the doorway while passing through one at a time.

B. Results

We define two methods for solving (8a):

- Batch - solve directly
- Parallel - solve using Algorithm 1 with multiple cores on the same or distributed machines

All trajectory optimization sub-problems were solved using ALTRO to a maximum constraint violation of 1e-3 and performed either on a desktop computer with an AMD Ryzen Threadripper 2950x processor and 16 GB RAM or an ODROID-XU4 microcomputer with a 32-bit Samsung Exynos5 Octa ARM Cortex-A15 2Ghz processor and 2 GB of RAM onboard the quadrotors. Algorithm 1 is implemented in the Julia programming language, uses the ALTRO trajectory optimization solver [27], and leverages the language's convenient methods for working with distributed computation. The solver hyper-parameters were tuned to each algorithm (batch vs. distributed) but kept the same across all scenarios.

Fig. 3 contains the timing results for the first scenario with L ranging from 3 to 15 quadrotors, clearly demonstrating the scalability of Algorithm 1 compared to explicitly solving the batch problem (8a).

When solving the second scenario, the batch version took 4.0 seconds to solve, whereas the distributed version took only 2.0 seconds. To test sensitivity of the algorithm to initial conditions,

TABLE I
RUNTIME PERFORMANCE: DOORWAY SCENARIO

Computer	Batch	Parallel
Desktop	3.6 s	0.8 s
1 Quadrotor	28.7 s	5.4 s
3 Quadrotors	-	5.7 s

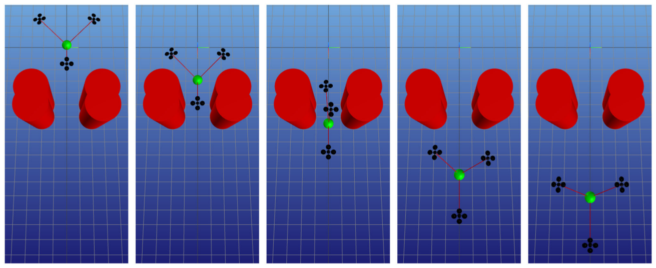


Fig. 6. Simulation results of a team with 3 quadrotors transporting a load, that a single agent cannot lift, through a doorway during a 10 s trajectory. The system reconfigures to travel through the doorway and is shown at time instances t = 0.0, 2.4, 5.0, 7.6 10.0 s (left to right).

we varied the x position by ±1.5 m in the x and y direction ($+x$ is closer to the goal, and $+z$ is vertical). The distributed version could solve problems within ±0.5 m in x and ±1.5 m in y. The batch version solved problems within ±1.5 m in x and ± 1.5 m in y. The batch version can handle a larger set of initial conditions since it jointly optimizes all of the trajectories, so will naturally be more robust.

A sequence of frames from the doorway scenario is presented in Fig. 6, and the timing results for this scenario are included in Table I. The last row in the table was performed on the 3 ODROID-XU4 computers onboard the quadrotors used in the hardware demonstration. One of the quadrotors was used as the "central" agent, which used a separate core for solving the load problem. Communication between the computers was performed over WiFi. The cost and constraint convergence is compared in Figs. 5(a) and 5(b), respectively. The cost and constraint values for the parallel solve were calculated by concatenating the current values for the quadrotor and load trajectories. The distributed method starts with a slightly higher

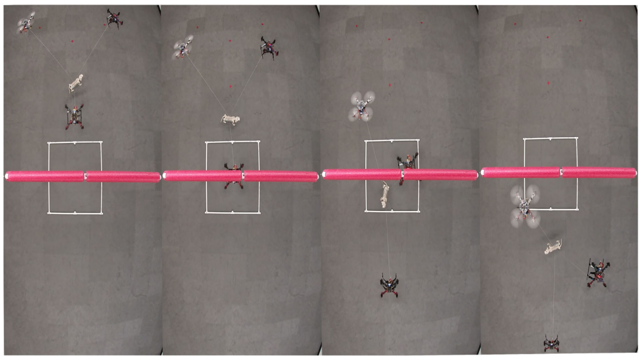


Fig. 7. Top view from hardware experiment with team of 3 quadrotors carrying a load through a doorway, progressing through time (left to right). The team reconfigures from an initial configuration that is wider than the doorway to a narrow configuration with the quadrotors nearly inline.

initial cost and constraint violation since the initial control trajectories assume zero force in the cable (since the initial "presolve" neglects these constraints), whereas the the initial guess for the batch method guesses the force from the trim conditions. Both methods achieve the same constraint satisfaction, but the distributed version achieves a slightly lower cost of 3.6 versus 4.6 for the batch version. While the resulting trajectories are similar, the distributed version results in a seemingly "smoother" trajectory, likely resulting from the guess provided by the "presolve" phase. The implementation and all simulation results are available at: https://github.com/RoboticExplorationLab/TrajectoryOptimization.jl/tree/distributed team lift.

V. Hardware Results

Hardware experiments were conducted with three custom-built quadrotor aerial robots based on the F330 frame [29] in a 16.5 m $\times$ 6.5 m $\times$ 2.7 m motion capture room. Each quadrotor ($m^i \approx 1$ kg) was equipped with a Pixfalcon, an open-source flight controller board, along with the PX4 open-source autopilot software (v1.7.3) to manage low-level control and real-time state estimation. Furthermore, each quadrotor used an ODROID-XU4 for high-level trajectory tracking and as bridge to the the Robot Operating System (ROS) network to interface with the Optitrack motion capture system. For this experiment, the 10 s doorway scenario trajectories from Section IV were computed offline using the onboard ODROID microcomputers networked over WiFi (see Table I) and a simple velocity-based trajectory tracking controller was used to follow the planned trajectories.

Despite each quadrotor being unable to individually lift a 0.9 kg load, the team was able to successfully transport it together through a 2.1 m $\times$ 1.6 m doorway. A sequence of frames from the experiment is shown in Fig. 7 and accompanying media. The experiments demonstrate that the method can be implemented on a team of resource-constrained quadrotors, making it practical for implementation onboard real systems.

VI. Discussion

The current work presents a novel method for solving cable-suspended load problems with quadrotors by posing them as nonlinear trajectory optimization problems. By decomposing the problem and solving each sub-problem in parallel, the algorithm is fast enough to generate new trajectories in a few seconds (faster than the time it takes to execute them). It also scales well to large numbers of agents, and is lightweight enough to run on resource-constrained onboard computers that can be carried by a quadrotor.

The presented approach has many advantages, including speed, scalability, and the ability to generate complex system-level behaviors via specification of general constraints or the objective of the optimization problem. However, there are also some important limitations worth noting: As with nearly all nonlinear optimization problems, convergence is not guaranteed. The results presented in Section V took careful tuning of the objective, selection of solver hyperparameters, and good initializations via trim conditions.

Our algorithm makes no assumptions about the dynamics of the system, and demonstrate that sub-problems can be solved in parallel across multiple agents to achieve dramatic reductions in compute time compared to a naive "batch" formulation. However, our approach benefits from the inherently sparse coupling between agents in the cable-suspended load problem, and it is unclear how well it will generalize to other multi-agent problems with more complicated coupling.

Several directions for future work remain: An implementation, specifically focusing on parallelization of the batch solve and communication between quadrotors, could likely execute faster than real-time, enable online re-planning and model-predictive control, and be more robust than the presented approach. Further extensions to the cable-suspended load scenario are also possible within our approach, including lifting rigid bodies instead of simple point masses, connecting the cables to arbitrary points on the quadrotor, and allowing for hybrid dynamics e.g., slack cables. While effective, the simplistic tracking controller used in the hardware demonstrations can also be improved, for example, by using the LQR feedback gains calculated by the trajectory optimization solver in the online control loop. Finally, generalizations to a variety of other multi-agent systems with different dynamics and constraints are possible.

References

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.
[2] L. Krick, M. E. Broucke, and B. A. Francis, "Stabilisation of infinitesimally rigid formations of multi-robot networks," *Int. J. Control*, vol. 82, no. 3, pp. 423–439, 2009.
[3] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.
[4] L. C. Pimenta, V. Kumar, R. C. Mesquita, and G. A. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *Proc. 47th IEEE Conf. Decis. Control*, 2008, pp. 3947–3952.
[5] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 1928–1935.
[6] H. Woern, M. Szymanski, and J. Seyfried, "The i-swarm project," in *Proc. ROMAN-15th IEEE Int. Symp. Robot Human Interactive Commun.*, 2006, pp. 492–496.

[7] A. Bera and D. Manocha, “Realtime multilevel crowd tracking using reciprocal velocity obstacles,” in *Proc. 22nd Int. Conf. Pattern Recognit.*, 2014, pp. 4164–4169.

[8] T. Lancashire, R. T. Lytwyn, G. Wilson, and D. Harding, “Investigation of the mechanics of cargo handling by aerial crane-type aircraft,” Boeing Co Morton Pa Vertol Div, Tech. Rep., 1966.

[9] K. Sreenath, T. Lee, and V. Kumar, “Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load,” in *Proc. 52nd IEEE Conf. Decis. Control*, 2013, pp. 2269–2274.

[10] S. Tang and V. Kumar, “Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2216–2222.

[11] C. De Crousaz, F. Farshidian, and J. Buchli, “Aggressive optimal control for agile flight with a slung load,” in *Proc. IROS Workshop Mach. Learn. Planning Control Robot Motion*, 2014.

[12] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza, “Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload,” in *Proc. Robot.: Sci. Syst.*, 2017, pp. 1–10.

[13] N. Michael, J. Fink, and V. Kumar, “Cooperative manipulation and transportation with aerial robots,” *Auton. Robots*, vol. 30, no. 1, pp. 73–86, 2011.

[14] M. Bernard, K. Kondak, I. Maza, and A. Ollero, “Autonomous transportation and deployment with aerial robots for search and rescue missions,” *J. Field Robot.*, vol. 28, no. 6, pp. 914–931, 2011.

[15] S. Tang, V. Wüest, and V. Kumar, “Aggressive flight with suspended payloads using vision-based control,” *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1152–1159, 2018.

[16] T. Lee, K. Sreenath, and V. Kumar, “Geometric control of cooperating multiple quadrotor uavs with a suspended payload,” in *Proc. IEEE 52nd Annu. Conf. Decis. Control*, 2013, pp. 5510–5515.

[17] T. Lee, “Geometric control of multiple quadrotor UAVs transporting a cable-suspended rigid body,” in *Proc. IEEE 53rd Annu. Conf. Decision and Control*, 2014, pp. 6155–6160.

[18] Z. Wang, S. Singh, M. Pavone, and M. Schwager, “Cooperative object transport in 3D with multiple quadrotors using no peer communication,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1064–1071.

[19] F. Augugliaro, A. P. Schoellig, and R. D’Andrea, “Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1917–1922.

[20] R. Ritz, M. W. Müller, M. Hehn, and R. D’Andrea, “Cooperative quadrocopter ball throwing and catching,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 4972–4978.

[21] V. Spurny *et al.*, “Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles,” *J. Field Robot.*, vol. 36, no. 1, pp. 125–148, 2019.

[22] R. Ritz and R. D’Andrea, “Carrying a flexible payload with multiple flying vehicles,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 3465–3471.

[23] R. Van Parys and G. Pipeleers, “Online distributed motion planning for multi-vehicle systems,” in *Proc. Eur. Control Conf.*, 2016, pp. 1580–1585.

[24] G. Inalhan, D. M. Stipanovic, and C. J. Tomlin, “Decentralized optimization, with application to multiple aircraft coordination,” in *Proc. 41st IEEE Conf. Decis. Control*, 2002, vol. 1, pp. 1147–1155.

[25] Y. Kuwata and J. P. How, “Cooperative distributed robust trajectory optimization using receding horizon milp,” *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 2, pp. 423–431, Mar. 2011.

[26] S.-S. Park, Y. Min, J.-S. Ha, D.-H. Cho, and H.-L. Choi, “A distributed ADMM approach to non-myopic path planning for multi-target tracking,” *IEEE Access*, vol. 7, pp. 163 589–163 603, 2019.

[27] T. A. Howell, B. E. Jackson, and Z. Manchester, “ALTRO: A fast solver for constrained trajectory optimization,” in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2019, pp. 7674–7679.

[28] D. Mellinger, N. Michael, and V. Kumar, “Trajectory generation and control for precise aggressive maneuvers with quadrotors,” *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 664–674, 2012.

[29] Z. Wang, R. Spica, and M. Schwager, “Game theoretic motion planning for multi-robot racing,” in *Proc. Distrib. Auton. Robot. Syst.*, 2019, pp. 225–238.