Scalable Cooperative Transport of Cable-Suspended Loads with UAVs using Distributed Trajectory Optimization

Brian E. Jackson[1]*, Taylor A. Howell[1]*, Kunal Shah[1], Mac Schwager[2], and Zachary Manchester[2]

***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.**

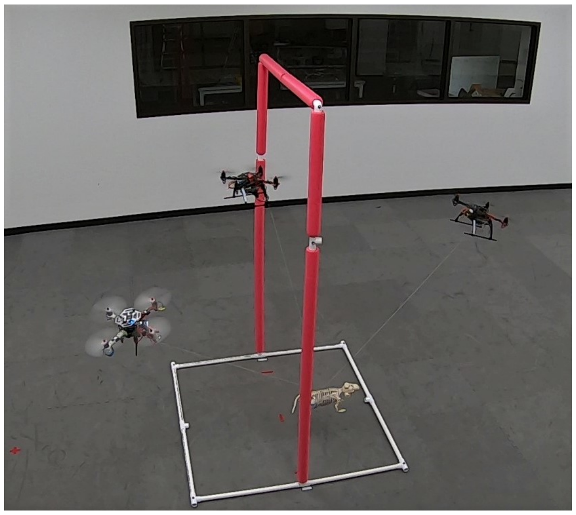

Fig. 1. Hardware experiment with a team of 3 quadrotors carrying a heavy load that a single quadrotor cannot lift. The team must reconfigure to proceed through the narrow doorway.

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.

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 communicational 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.

[1]Department of Mechanical Engineering, Stanford University {bjack205,thowell,k2shah}@stanford.edu
[2]Department of Aeronautics and Astronautics, Stanford University {schwager,zacmanchester}@stanford.edu
*These authors contributed equally to this work

Centralized approaches, in contrast, are able to reason about general dynamics and constraints. However, they typically necessitate increased computation and communication requirements, and require solving large optimization problems. Centralized methods are, therefore, nearly always run offline. Additionally, since the optimization problems involved typically 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.

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]. Of these, only [21] used a decentralized approach.

In this work, we present a scalable distributed approach for obtaining the 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). We formulate a trajectory optimization problem and consider the full nonlinear dynamics of the quadrotors and non-convex obstacle and collision constraints. We use the trajectory optimization solver ALTRO [23], along with some novel modifications to handle quaternion state variables, to find the state and control trajectories for the system of agents. To make the problem scalable in the number of agents, inspiration is taken from the alternating direction method of multipliers (ADMM) [24] to decompose the problem and solve agent-level subproblems in parallel. The result is a substantial reduction in solution time and superior scaling as the number of agents is increased.

The remainder of this paper is organized as follows: Section II provides background on quaternions and trajectory optimization. Section III presents modifications to ALTRO for states containing quaternions. Section IV formulates the cable-suspended-load problem as a trajectory optimization problem. Section V presents a decomposition scheme and an algorithm for solving the batch problem in parallel among agents. Section VI contains simulation results and Section VII contains details of the hardware experiments. Finally, we conclude with a discussion of the algorithm and results in Section VIII.

II. Background

A. Notation

We use $A \in \mathbb{S}^n_{++}$ for a $n \times n$ symmetric positive-definite matrix, and $A \in \mathbb{S}^n_{+}$ for an $n \times n$ symmetric positive-semidefinite matrix.

B. Quaternions

The unit quaternion is a four-parameter non-singular representation of rotations comprising a vector part $z \in \mathbb{R}^3$ and a scalar part $s \in \mathbb{R}$. To represent quaternion operations as linear-algebraic expressions, the following conventions are used: a quaternion q encodes the rotation from a vehicle's body frame to the world frame. The equivalent rotation matrix is given by,

$$R(q) = I + z^\times (z^\times + sI), \tag{1}$$

where I is the 3×3 identity matrix and $z^\times$ denotes the 3×3 skew-symmetric cross-product matrix:

$$z^\times = \begin{bmatrix} 0 & -z_3 & z_2 \\ z_3 & 0 & -z_1 \\ -z_2 & z_1 & 0 \end{bmatrix}. \tag{2}$$

Quaternion multiplication is denoted $q_2 \otimes q_1$, and the quaternion conjugate, representing the opposite rotation, is denoted q^*. For $r \in \mathbb{R}^3, \hat{r}$ is a quaternion with $s = 0$ and $z = r$. $H \in \mathbb{R}^{4\times 3}$ is the matrix "hat" operator, such that $\hat{r} = Hr$ and $r = H^T \hat{r}$. The matrix $M(q) \in \mathbb{R}^{4\times 4}$ is defined such that $M(q_2)q_1 = q_2 \otimes q_1$.

Quaternion rotations require special consideration in optimization problems to properly account for the unit-norm constraint. We define $\phi \in \mathbb{R}^3$ to be a three-dimensional differential rotation, such that $\hat{\phi} = \delta q$, where $q = q_0 \otimes \delta q$, when δq is a small rotation. We seek the Jacobian $G(q) \in \mathbb{R}^{3\times 4}$ that maps from the quaternion to our three-dimensional differential, ϕ. From above, we have $q = q_0 \otimes \delta q = q_0 \otimes \hat{\phi} = M(q_0)H\phi$, so $G(q_0)^T = M(q_0)H$.

If $h(q) : q \mapsto \mathbb{R}^p$ is a function that maps a quaternion to a real-valued vector, the linearization of $h(q)$ gives

$$\delta h \approx G(q) \frac{\partial h(q)}{\partial q} G(q)^T \phi. \tag{3}$$

Similarly, for a real-valued function $g(q) : q \mapsto \mathbb{R}$, the second-order Taylor series expansion is

$$\delta g \approx \frac{\partial g(q)}{\partial q} G(q)^T \phi + \frac{1}{2} \phi^T G(q) \frac{\partial^2 g(q)}{\partial q^2} G(q)^T \phi \tag{4}$$

C. Trajectory Optimization

Numerical trajectory optimization algorithms solve variations of the following problem,

$$\begin{aligned} \underset{x_{0:N},\, u_{0:N-1},\, \Delta t}{\text{minimize}} \quad & \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k, \Delta t) \\ \text{subject to} \quad & x_{k+1} = f_k(x_k, u_k, \Delta t), \\ & c_k(x_k, u_k) \, \{\leq, =\} \, 0, \end{aligned} \tag{5}$$

where N is the number of time steps, k is the time-step index, ℓ_N and ℓ_k are the terminal and stage costs, $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ are the states and control inputs, Δt is the duration of a time step, $f_k(x_k, u_k, \Delta t)$ is the discrete dynamics, and $c_k(x_k, u_k)$ are inequality and equality constraints. For notational simplicity, we use $X = x_{0:N} = \{x_0, \ldots, x_N\}$ and $U = u_{0:N-1} = \{u_0, \ldots, u_{N-1}\}$.

III. Modified Augmented Lagrangian iLQR for Quaternions

In this section we present modifications to ALTRO, a constrained trajectory optimization solver that uses iterative LQR [25] within an augmented Lagrangian framework (AL-iLQR), in order to optimize states that contain quaternions.

With standard commercial solvers, like those used in direct trajectory optimization methods, quaternions can only be treated as vectors in $\mathbb{R}^4$, often resulting in poor performance. Here we present a method for correctly optimizing with respect to a unit quaternion using AL-iLQR. Extensions of this method to states with multiple quaternions are easily obtained through simple concatenation.

Let $x \in \mathbb{R}^n$ $(n \geq 4)$ be a state vector containing a unit quaternion of the form,

$$x = \begin{bmatrix} y \\ q \end{bmatrix} \tag{6}$$

where $y \in \mathbb{R}^{n-4}$ is a standard vector and q is the quaternion. Using the results from section II-B, the difference between two states, x and x_0, is

$$\delta x = \begin{bmatrix} \delta y \\ \phi \end{bmatrix} = \begin{bmatrix} y - y_0 \\ H^T(q_0^* \otimes q) \end{bmatrix} \in \mathbb{R}^{n-1}, \tag{7}$$

from which we get the Jacobian,

$$G(x) = \begin{bmatrix} I_{n-4} & 0 \\ 0 & G(q) \end{bmatrix} \tag{8}$$

where $G(x) \in \mathbb{R}^{n-1 \times n}$. The next sections detail how to modify the AL-iLQR backward and forward passes to correctly take expansions with respect to quaternion states.

Backward pass: The backward pass of iLQR linearizes the dynamics at each time step and approximates the cost-to-go as a quadratic function by taking the second-order Taylor series expansion of the cost-to-go. $P_N \in \mathbb{S}^n_{++}$, $p_N \in \mathbb{R}^n$ are the Hessian and gradient of the terminal cost-to-go and the corrected terms are,

$$\tilde{P}_N = G(x_N) P_N G(x_N)^T \tag{9}$$

$$\tilde{p}_N = G(x_N) p_N \tag{10}$$

Q_{xx}, Q_{ux}, and Q_x are the terms from the approximate[1] quadratic expansion of the action-value function at time step k that must be modified. The corrected terms from the expansion are:

$$\tilde{Q}_{xx} = G(x_k) Q_{xx} G(x_k)^T \tag{11}$$

$$\tilde{Q}_{ux} = Q_{ux} G(x_k)^T \tag{12}$$

$$\tilde{Q}_x = G(x_k) Q_x. \tag{13}$$

The rest of the backward pass proceeds as normal, using the corrected expansion terms.

Forward pass: The only required modification to the forward pass is the correct calculation of the difference between states x and x_0 using (7). The modified change in control, where $K \in \mathbb{R}^{m \times (n-1)}$ and $d \in \mathbb{R}^m$ are the feedback and feedforward gains from the backward pass, is:

$$\delta u_k = K_k \delta x_k + d_k. \tag{14}$$

Objective: Directly subtracting states that contain quaternions (or any rotational representation) is incorrect. A good alternative is to again use the state difference Jacobian, as demonstrated in the following objective,

$$\begin{aligned} &\tilde{J}(X,U) = \\ &(x_N - x_f)^T G(x_N)^T \tilde{W}_N G(x_N)(x_N - x_f) \\ &+ \Delta t \sum_{k=0}^{N-1} \{ (x_k - x_f)^T G(x_k)^T \tilde{W}_k G(x_k)(x_k - x_f) \\ &+ (u - u_r)^T V_k (u - u_r) \} \end{aligned} \tag{15}$$

where $\tilde{W} \in \mathbb{S}^{n-1}_+, V \in \mathbb{S}^m_{++}$ are cost matrices. Here the three-parameter angular representation ϕ is penalized.

IV. Batch Problem

We formulate a trajectory optimization problem for the team cable-suspended load problem with L quadrotors attached to a single 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 of the system is critical for getting good performance using trajectory optimization, and is described in detail in Section IV-A. Section IV-B then describes the optimization problem.

A. Dynamics

1) Quadrotor Model with Quaternions: The quadrotor dynamics presented in [27] are modified to use quaternions for angular representation and incorporate the force generated by a suspension cable:

$$\begin{aligned} \dot{x} = \begin{bmatrix} \dot{r} \\ \dot{q} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} &= \begin{bmatrix} v \\ \frac{1}{2} q \otimes \hat{\omega} \\ g + \frac{1}{m^i}(R(q)F(u) + F_c(u_5, x, x^\ell)) \\ J^{-1}(\tau(u) - \omega \times J\omega) \end{bmatrix} \\ &= f(x, u; x^\ell) \end{aligned} \tag{16}$$

where $r \in \mathbb{R}^3$ is the position, q is a unit quaternion, $v \in \mathbb{R}^3$ 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. The forces and torques $F(u)$, $\tau(u) \in \mathbb{R}^3$ in the body frame are

$$F(u) = \begin{bmatrix} 0 \\ 0 \\ k_f(u_1 + u_2 + u_3 + u_4) \end{bmatrix} \tag{17}$$

$$\tau(u) = \begin{bmatrix} k_f d_{\text{motor}}(u_2 - u_4) \\ k_f d_{\text{motor}}(u_3 - u_1) \\ k_m(u_1 - u_2 + u_3 - u_4) \end{bmatrix} \tag{18}$$

where k_f, k_m are motor constants, d_{motor} is the distance between motors, and $u_{1:4}$ are the motor thrusts. The forces from the cables are calculated as:

$$F_c(\gamma, x, x^\ell) = \gamma \frac{r^\ell - r}{\|r^\ell - r\|_2}, \tag{19}$$

[1]iLQR computes the Gauss-Newton quadratic approximation of the Hessian of the cost-to-go, whereas Differential Dynamic Programming [26] computes the complete quadratic approximation Hessian.

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:

$$\begin{aligned}\dot{x}^\ell = \begin{bmatrix}\dot{r}^\ell \\ \dot{v}^\ell\end{bmatrix} &= \begin{bmatrix} v^\ell \\ g + \frac{1}{m^\ell} F^\ell(x^\ell, u^\ell, x^1, ..., x^L)\end{bmatrix} \\ &= f^\ell(x^\ell, u^\ell; x^1, \dots, x^L)\end{aligned} \tag{20}$$

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 control vector, x^i is the state vector of quadrotor i, and

$$F^\ell(x^\ell, u^\ell, x^1, ..., x^L) = -\sum_{i=1}^{L} F_c(u_i^\ell, x^i, x^\ell). \tag{21}$$

B. Optimization Problem

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

$$\bar{x} \in \mathbb{R}^{13L+6} = \begin{bmatrix} x^1 \\ \vdots \\ x^L \\ x^\ell \end{bmatrix}, \quad \bar{u} \in \mathbb{R}^{5L+L} = \begin{bmatrix} u^1 \\ \vdots \\ u^L \\ u^\ell \end{bmatrix}. \tag{22}$$

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

$$\underset{\mathbf{X}, \mathbf{U}}{\text{minimize}} \quad J^\ell(X^\ell, U^\ell) + \sum_{i=1}^{L} J^i(X^i, U^i) \tag{23a}$$

subject to

$$x_{k+1}^i = f_k^i(x_k^i, u_k^i, \Delta t; x_k^\ell), \quad \forall i \in \mathcal{I}_L, \tag{23b}$$

$$x_{k+1}^\ell = f_k^\ell(x_k^\ell, u_k^\ell, \Delta t; x_k^1, \dots, x_k^L), \tag{23c}$$

$$x_0^i = x_0^i, \quad \forall i \in \mathcal{I}_A, \tag{23d}$$

$$x_N^\ell = x_f^\ell, \tag{23e}$$

$$r_{\text{min}}^i \le r_k^i \le r_{\text{max}}^i, \quad \forall i \in \mathcal{I}_A, \tag{23f}$$

$$0 \le (u_k^i)_j \le u_{\text{max}}^i, \quad \forall i \in \mathcal{I}_L, j \in \{1, \dots, 4\}, \tag{23g}$$

$$u_k^\ell \ge 0, \tag{23h}$$

$$(u_k^i)_5 = (u_k^\ell)_i, \quad \forall i \in \mathcal{I}_L, \tag{23i}$$

$$\left\| r_k^i - r_k^\ell \right\|_2 = d_{\text{cable}}, \quad \forall i \in \mathcal{I}_L, \tag{23j}$$

$$2d_{\text{quad}} - \left\| p_k^i - p_k^j \right\|_2 \le 0, \quad \forall i, j \in \mathcal{I}_L, i \ne j, \tag{23k}$$

$$d_{\text{quad}} + d_{\text{obs}} - \left\| p_k^i - p_{\text{obs}}^j \right\|_2 \le 0, \quad \forall i \in \mathcal{I}_L, \forall j, \tag{23l}$$

$$d_{\text{load}} + d_{\text{obs}} - \left\| p_k^\ell - p_{\text{obs}}^j \right\|_2 \le 0, \quad \forall j, \tag{23m}$$

where $\mathbf{X} = [X^1, \dots, X^L, X^\ell], \mathbf{U} = [U^1, \dots, U^L, U^\ell]$, $p^i \in \mathbb{R}^2$ is the two-dimensional position of the quadrotor or load (discarding height), d is a scalar dimension (e.g., quadrotor radius), and all constraints apply at each time steps k.

The constraints are, from top to bottom: discrete quadrotor dynamics from (16) (23b), discrete load dynamics from (20) (23c), initial conditions (23d), final condition for the load (23e), workspace constraints (i.e. floor and ceiling constraints) (23f), quadrotor motor constraints (23g), positive cable tension (23h), equal tension force on quadrotor and load (23i), cable length (23j), collision avoidance (23k), and obstacle avoidance for the quadrotors (23l) and load (23m). The self-collision constraints model the quadrotors as cylinders to help prevent unmodeled prop wash effects from disturbing the system. The objective for each quadrotor is (15) and objective of the load was a simple quadratic cost function.

We found it beneficial to initialize the problem with 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 of 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 given to ALTRO.

V. Distributed Formulation

We present a scalable approach for solving (23a) by decomposing the problem per-quadrotor. Of the given constraints, only the collision avoidance constraint (23k) directly couples the states of the quadrotors. The load, on the other hand, is directly coupled to all of the quadrotors via the cable constraints (23i) and (23j) and dynamics (23c). The objective, by design, is separable by agent, i.e. there is no coupling between the state and controls of the quadrotors or the quadrotors and load.

From these observations, the decomposition is quite apparent: solve a trajectory optimization problem for each quadrotor independently, treating all other quadrotor and load trajectories as constant. After each quadrotor has solved independently, the updated trajectories are collected and used to optimize the load trajectory. The updated load trajectory, along with the updated quadrotor trajectories, is then communicated to each quadrotor and the process is repeated. This idea is summarized in Algorithm 1.

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 (23k) or cable constraints (23i) and (23j)). 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

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:          end for
7:          update $\tilde{\mathbf{X}}, \tilde{\mathbf{U}}$
8:          $X^\ell, U^\ell \leftarrow$ SOLVE-LOAD($X^\ell, U^\ell, \tilde{\mathbf{X}}, \tilde{\mathbf{U}}$)
9:          update $\tilde{\mathbf{X}}, \tilde{\mathbf{U}}$
10:     end while
11: return $\tilde{\mathbf{X}}, \tilde{\mathbf{U}}$
12: end function
```

follow. The time to solve these initial trajectory optimization problems is included in the overall solution time.

VI. Simulations

A. Scenarios

Two scenarios using the batch problem formulation are considered: 1) point-to-point transfer of a load using L quadrotors (see Fig. 2), and 2) a load transfer through a narrow doorway using 3 quadrotors (see Fig. 4). Both scenarios solve a 10 s trajectory. For the doorway scenario, we assume perfect knowledge of the location of the door.

In the first scenario, the stage costs (terms inside the sum of (15)) were identical for all time steps. In second 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 (essentially encouraging the quadrotors to "fan" out around the load, see Fig. 1).

B. Results

We define two methods for solving (23a):

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

All trajectory optimization 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 16GB RAM or an ODROID-XU4 microcomputer with a 32-bit Samsung Exynos5 Octa ARM Cortex-A15 2Ghz processor and 2GB of RAM onboard the quadrotors. Algorithm 1 is implemented in the Julia programming language and leverages the language's convenient methods for working with distributed computation.

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 (23a).

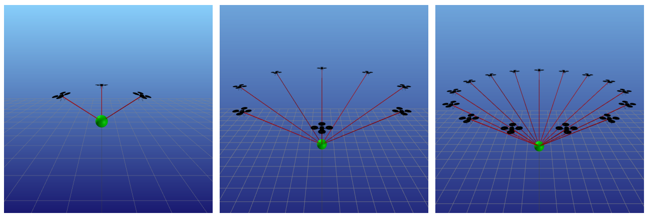
Fig. 2. Simulation of teams with 3, 8, and 15 quadrotors (left, center, right) in final configuration after a point-to-point load transfer. To maintain the final system configuration, quadrotors orient to produce thrust that maintains hover despite a force resulting from the load.

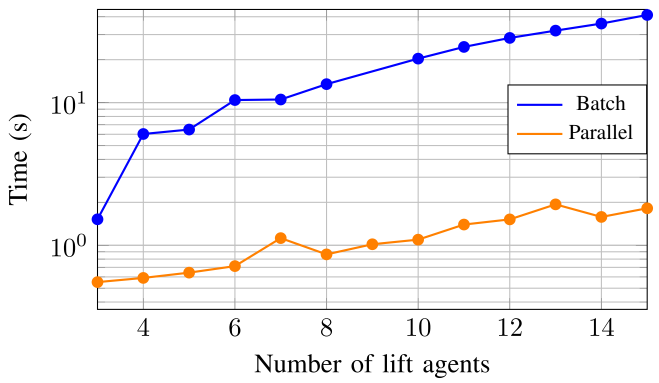

Fig. 3. Timing result comparing batch and parallel algorithms for a point-to-point load transfer using L quadrotors. The parallel algorithm scales favorably compared to the batch approach as the number of quadrotors is increased.

A sequence of frames from the doorway scenario is presented in Fig. 4, 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 set up a separate core for solving the load problem. Communication between the computers was performed over WiFi. Both methods achieve the same constraint satisfaction, but the batch version achieves a slightly lower cost of 2.9 versus 5.1 for the parallel version. As a result, the final trajectories have noticeable qualitative differences, with the batch solution producing a more visibly "fluid" motion. The implementation and all simulation results are available at: `https://github.com/RoboticExplorationLab/TrajectoryOptimization.jl/tree/ADMM`

VII. Hardware Results

Hardware experiments are conducted with three custom-built quadrotor aerial robots based on the F330 frame [28] in a 16.5 m × 6.5 m × 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

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

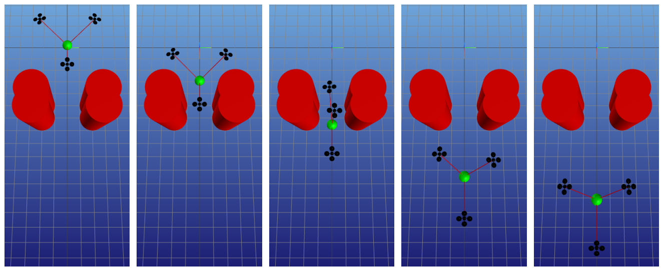
Fig. 4. Simulation results of a team with 3 quadrotors transporting a load, that a single agent cannot lift, through a doorway during a 10 s trajectory. The system reconfigures to travel through the doorway and is shown at time instances t = 0.0, 2.4, 5.0, 7.6 10.0 s (left to right).

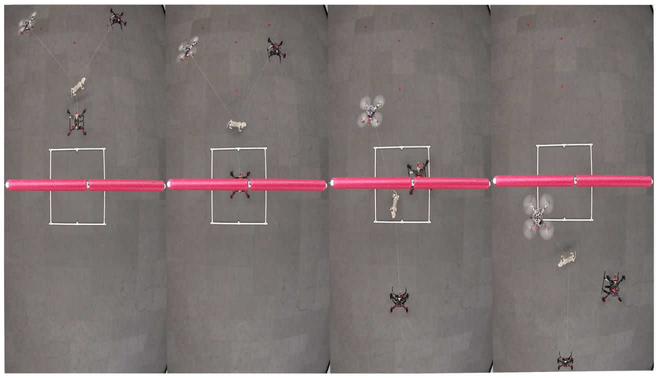
Fig. 5. Top view from hardware experiment with team of 3 quadrotors carrying a load through a doorway, progressing through time (left to right). The team reconfigures from an initial configuration that is wider than the doorway to a narrow configuration with the quadrotors nearly inline.

control and real-time state estimation. Furthermore, each quadrotor uses 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 VI 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. Given the solution times for the quadrotor computers, it would be possible to re-plan online in response to changes in the environment before the trajectory is completed.

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 × 1.6 m doorway. A sequence of frames from the experiment is shown in Fig. 5 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.

VIII. Discussion

The current work presents a novel method for solving cable-suspended load problems with quadrotors by posing them as a nonlinear trajectory optimization problem. 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, such as speed, scalability, and the ability to change the resulting system behaviors by modifying the constraints or 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 current results took careful tuning of the objective, selection of solver hyperparameters, and finding good trajectory 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: a more careful implementation, specifically focusing on the parallelization and communication between quadrotors, could likely execute faster than real-time, enabling online re-planning and model-predictive control. Several 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 slack cables. While effective, the simplistic tracking controller used in the hardware demonstrations can also be improved, for example, by using the 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 should be possible.

Acknowledgements

This work was partially supported by a NASA Early Career Faculty Award (Grant Number 80NSSC18K1503) and the Stanford-Ford Alliance. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1656518. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[2] L. Krick, M. E. Broucke, and B. A. Francis, "Stabilisation of infinitesimally rigid formations of multi-robot networks," *International Journal of 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 Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[4] L. C. Pimenta, V. Kumar, R. C. Mesquita, and G. A. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008, pp. 3947–3952.

[5] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 1928–1935.

[6] H. Woern, M. Szymanski, and J. Seyfried, "The i-swarm project," in *ROMAN 2006-The 15th IEEE International Symposium on Robot and Human Interactive Communication*, IEEE, 2006, pp. 492–496.

[7] A. Bera and D. Manocha, "Realtime multilevel crowd tracking using reciprocal velocity obstacles," in *2014 22nd International Conference on Pattern Recognition*, IEEE, 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 *CDC*, Citeseer, 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 *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 2216–2222.

[11] C. De Crousaz, F. Farshidian, and J. Buchli, "Aggressive optimal control for agile flight with a slung load," in *IROS 2014 Workshop on Machine Learning in Planning and Control of Robot Motion*, Citeseer, 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 *Robotics: Science and Systems*, 2017, pp. 1–10.

[13] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous 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," *Journal of Field Robotics*, 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 Robotics and Automation Letters*, 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 *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, IEEE, 2013, pp. 5510–5515.

[17] T. Lee, "Geometric control of multiple quadrotor uavs transporting a cable-suspended rigid body," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, IEEE, 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 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 *2012 IEEE/RSJ international conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 1917–1922.

[20] R. Ritz, M. W. Müller, M. Hehn, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 4972–4978.

[21] V. Spurny, T. Báča, M. Saska, R. Pěnička, T. Krajník, J. Thomas, D. Thakur, G. Loianno, and V. Kumar, "Cooperative autonomous search, grasping, and delivering in a treasure hunt scenario by a team of unmanned aerial vehicles," *Journal of Field Robotics*, vol. 36, no. 1, pp. 125–148, 2019.

[22] R. Ritz and R. D'Andrea, "Carrying a flexible payload with multiple flying vehicles," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 3465–3471.

[23] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A fast solver for constrained trajectory optimization," in *2019 IEEE International Conference on Intelligent Robots and Systems*, IEEE, 2019.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[25] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.," in *ICINCO (1)*, 2004, pp. 222–229.

[26] D. H. Jacobson and D. Q. Mayne, "Differential dynamic programming," 1970.

[27] D. Mellinger, N. Michael, and V. Kumar, “Trajectory generation and control for precise aggressive maneuvers with quadrotors,” *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[28] Z. Wang, R. Spica, and M. Schwager, “Game theoretic motion planning for multi-robot racing,” in *Distributed Autonomous Robotic Systems*, Springer, 2019, pp. 225–238.