Interactive online choreography for a multi-quadrotor system

Ellen A. Cappo, Arjav Desai, Nima Dehghani, Ali Momeni, Nathan Michael
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract—We present a full system for online control of multiple quadrotors. Rather than following scripted directions, our system interprets user instructions and modifies quadrotor behavior online, allowing a user to issue instructions at any time to easily control and modify flight. Handling multiple quadrotors is done through grouping to allow a user to reduce a number of individual robots into scaleably sized collectives as needed. Instructions are specified from a simple enumeration of potential action combinations governing formation shape, flight speed and style, and overall group tasking (travel to target, aim spotlight, etc), which combinatorially yield over 8000 possible behaviors. The system interprets all user commands into dynamically feasible, non-colliding trajectories, and results are shown for over 100 behavior instructions issued online to a multi-quadrotor hardware system used for a theatrical performance.

I. INTRODUCTION

For a theatrical production, we desire a multi-quadrotor system which allows for easy interaction with the user. We therefore wish to ensure that it is possible for the system to be adapted or changed online. This is vital to the theatrical endeavor, where rather than simply execute a scripted series of motions, a performer must be able to interactively command the quadrotors as part of the production.

We propose here a system design where high-level, general actions specified by the user are interpreted into collision-free, dynamically feasible trajectories for the robots. To ensure that all generated trajectories are both safe—never allowing quadrotors to violate a specified inter-robot distance—and dynamically feasible, we allow the system to make controlled modifications to user instructions by incorporating an online search method for trajectory timing to enforce that trajectories remain within specified limits given the current system state. Status information on the resulting trajectories is communicated back to the user via system lighting and text output.

We evaluate our system by verifying over 64,000,000 state transitions and illustrate the system’s ability to adapt to online commands through performance of a six-robot, 100 command theatrical work, requiring timing changes, multi-formation splitting and merging, and multiple behavior combinations. We confirm that when responding online to unscripted commands, processing time does not disrupt flight control and that plans never violate the specified robot clearance distance or imposed dynamic limits.

II. RELATED WORK

Performances and light displays using quadrotors are becoming increasingly popular, as evidenced by recent shows by companies [1,2] as well as more dynamic choreography [11]. The trajectories, collision avoidance, and dynamic feasibility for these efforts, however, were computed offline and so did not require performer-system interaction or time-dependent motion planning methods.

To create an online planning framework for single-user, multi-quadrotor interaction, we present a full system extending the flight control and motion planning work presented in [6,8,12,13] and formulate a real-time trajectory generation methodology based on a user input “behavior” specification. Similar to previous work in user input for multi-robot systems, such as finding human-expressive gestures [3,4] or joystick-based multi-quadrotor formation control [15], our input representation allows a user to direct formation shape or movement without having to consider low-level tasks such as collision avoidance. We additionally incorporate an online search for transition times, allowing the system to modify user input to create time optimal, dynamically feasible flight trajectories. This widens the range of feasible instructions and facilitates interaction between the user and system.

We first describe the system design (Section III), including our choice of behavior-level input, user input options, and an overview of instruction translation to feasible flight trajectories and system feedback to the user. We then present results for a 100-instruction online performance done using a multi-quadrotor hardware system (Section IV), showing that...
minimum distance and dynamic requirements are respected, and link to video results.

III. SYSTEM DESIGN

Our desire to interactively involve quadrotors in a narrative storytelling at the direction of a performer directs the design of the quadrotor system. Foremost, online interaction between the user and the quadrotor system is critical to the performance objective. We have therefore chosen system commands which allow the user to easily command both actions (such as projecting a light or image onto a target) and acting behaviors (such as flying fast, slow, or in a repeating pattern). An important element of the direction is the use of formations. Creating and directing groups allows the user to issue commands for a few “macro” agents rather than for numerous individuals, as well as providing easy specification of visually appealing flight arrangements. Commands at this level allow for detailed direction using relatively few specifications, giving the user the ability to issue commands quickly while narrating the story.

A desired behavior command, along with information of the current system state, is used by a centralized planner to compute collision-free trajectories for all robots. The system handles all formation transitions, splitting, and merging actions. Further, the planner is able to modify timing elements of a user’s instruction in order to ensure that any generated trajectories are dynamically feasible given the current system state. Modifications are then communicated back to the user. Details of this process are provided in the following sections.

A. Behavior-level commands

In order to readily specify both choreography and tasks such as shining spotlight for multiple quadrotors, we have defined a set of basic information fields which describe a “behavior” command. These basic actions are broad enough to free the user from having to deal with low level consideration such as collision avoidance, while detailed enough to generate specific desired choreography, such as “drunken” or “nervous” characterized flight motions. These commands can be issued quickly and additionally combined online to create more complicated assembled behaviors.

A user specifies basic actions such as “move” or “turn lights on.” These commands take parameters, such as which quadrotors should perform the action and the target location to where they should move or aim a spotlight. A user may also indicate different speeds (“fast,” “slow,” “moderate”); flight mannerisms (“nervous,” “drunk”); or describe the relationship between the quadrotors, asking them to move in a line, regular polygon, or freeform. Quadrotors can also be asked to perform repetitive motions such as moving in continuous circles or back and forth along lines, which allows the quadrotors to continually move without the performer needing to direct every minute of flight.

Further, not every field must be specified for any given message. Default values are specified for commonly used fields, allowing common instructions to be issued quickly. This input structure also allows for easy, custom formation specification online. A specific example can be seen in Figure 2, where two groupings of quadrotors, formed as triangles, have been formed into a single group rotating around a center location. Rather than forming all six quadrotors into a single regular polygon, by leaving the grouping argument blank, the system understands to apply the specified “rotate” command to all quadrotors as a single entity but to leave the groups in their respective triangle formations.

B. User input methods

“Behavior” commands may be generated through a variety of input methods. As a primary training tool for the performer, we have developed a tablet-based user interface which uses feedback from the motion-capture system to show realtime positions of the quadrotors, and allows a performer to specify robots and target locations through simple touch-based selection. Individual components of a behavior message may be specified from a selection of options, which update to present legal potential combinations as fields are selected.

The tablet-based interface is highly convenient for training, but focuses a performer’s attention on a screen rather than on the audience or system itself. Therefore, we have also worked to define a gesture-based input method, where gesture recognition technology is used to read the performer’s movements as reported by a motion capture system. Gestures are a natural way for the performer to interact with the physical system; when selecting which robots should perform an action, it is convenient to simply point at the desired individual quadrotor or group. Gestures can also allow the performer-system interaction to be easily viewed by the audience, and the use of full-bodied gestures gives a physical choreography to the performance which underlines the performer’s vocal narration.

While gestures are theatrically preferable, there is a burden on the performer to learn the required gesture language and issue commands while narrating. While refinement of this interface is still being explored, we have therefore generated results for this work using behavior-instructions issued online through a GUI-based system.

C. Trajectory design from behavior commands

Behavior-instruction fields such as the target, specified robots, and any desired formation specifications give positioning information used to locate the starting and ending states of our trajectories, while any desired speed is used to find an initial nominal trajectory duration. This allows us to completely define commanded behaviors as time-based polynomial trajectories in the \( x, y, z \), and heading, \( \psi \), dimensions as in \([10],[12]\). Because \( x, y, z \), and \( \psi \) are the flat outputs for the differentially flat quadrotor system, \([8]\), specifying fifth- or higher-order polynomials in these dimensions allows us to calculate the continuous control inputs necessary to achieve a state trajectory, and allow us to easily scale trajectories with respect to time. These are important aspects which allow us to evaluate behaviors for dynamic feasibility \([5]\). The following sections present a high-level overview of how a desired
behavior may be composed into polynomial trajectories for multiple quadrotors; detailed explanation for finding time optimal transitions, merging, and splitting quadrotor formations may be further referenced as in \cite{6}.

We note that to plan for quadrotors to transition between desired behaviors, we make the following assumptions:

1) all robots are homogeneous and interchangeable with no specific preference to goal locations within a formation;
2) the region defined by the convex hull of all source and goal locations is obstacle free;
3) all start and goal positions are located a predefined minimum clearance distance apart;
4) the number of specified goal locations equals the total number of specified robots.

To therefore transition between behaviors, the following steps are performed:

- **Shape definition:** The starting configuration for the desired behavior is taken from the system state based on the time at which the instruction is received and the ending configuration calculated based on the desired target location and behavior specification. These configurations are specified with respect to the local shape origin frame. A group formation is represented as a "shape," the specification of relative states between robots, given with respect to a local group reference frame. A shape can be described as a time varying shape vector $s(t)$, which contains the spatial and orientation relationships between each pair of robots in the formation, and between every robot and the origin of the local shape reference frame.

- **Optimal Assignment:** The desired starting and ending goal locations are then specified in the local shape reference frame. Assignment is performed based on \cite{13} and as explained in \cite{6}, where costs are computed based on distances between start and goal states, and the Hungarian Algorithm \cite{7} is used to perform optimal assignment, minimizing the total distance travelled by all robots. Using a nominal transition time, the local shape vector $s(t)$ between all robots in the group is designed by finding local trajectories which take robots from their starting locations to their goal states in the local frame.

- **Ensuring dynamic feasibility:** The movement of the robots in their local frame is composed onto a trajectory designed to move the group in the inertial frame, similarly as in \cite{14} and as further detailed in \cite{6}. The nominal required control thrusts necessitated by these resulting trajectories for quadrotor motion in the inertial frame can be found as detailed in \cite{5} \cite{6} and checked against dynamic constraints to ensure dynamic feasibility of the resulting final trajectories. If the required thrusts violate actuator limits, trajectories are time-scaled as in \cite{8} to decrease the needed actuator effort. If the trajectory duration must be extended in order to lower the needed actuator thrusts to feasible values, the system issues a warning to the user notifying them of the amended instruction timing. If, however, the group is able to feasibly transition to the desired formation in the given amount of time, the system performs a search for the minimal-time feasible transition as described in \cite{6}.

- **Collision checking and relative time scaling:** Given dynamically feasible (and minimal) transitions for all robots, a final check is performed to ensure all trajectories preserve a minimum separation distance between robots. This is done through prioritization and timescaling as in \cite{13} and \cite{6}, where a prioritization order is established based on start and goal locations, and each robot is checked for collisions against all robots of higher priority. In the event that a collision occurs between robots, the trajectory of the lower prioritized robot is assigned a small, positive time offset. This process is repeated iteratively, guaranteeing collision free trajectories for all the robots in the formation based off the minimal transition time for the highest prioritized robot \cite{13}. In the event no trajectories intersect, all robots transition between formations in the minimal feasible time.

\textbf{D. Feedback to User}

While the user can readily determine the current system state by observing the quadrotors' physical motion, the system provides additional feedback to the user in the way of light patterns and written status messages.

Each quadrotor's LED arrangement is mapped to the quadrotor's velocity, allowing a user to clearly see from LED intensity when quadrotors are starting or finishing a behavior. More detailed system output is displayed as text at the system console. After every instruction, the system reports the validity of the given instruction based on the state of the system. All instructions which are valid, however, may not be dynamically
feasible given the current system state. For example, robots may be able to move from the current state to a desired target, but not at the specified speed. In the case that the instruction is infeasible, the system either reports an error message explaining the limitation, or performs a modified version of the instruction and outputs an explanatory message, such as the fact that speed was modified to produce feasible trajectories.

IV. Evaluation

The described system is being used for a theatrical production where commands for desired actions, formation specifications, motion characteristics, and desired speeds have been combinatorially combined to describe over 8,000 behaviors. A state transition is defined as any time the system receives a command to transition from performing its current behavior to a new behavior. Including self-transitions, this yields over 8,0002 potential transitions. The system’s ability to handle all potential behavior transitions was validated by randomly generating transitions and ensuring that every one of the over 64,000,000 state transitions was performed at a minimum once.

A theatrical scene enacting a selection of roughly 100 of these behaviors, spanning approximately fifteen minutes of flight time (excluding battery changes), for between two and six quadrotors has been performed on a physical system in a motion capture arena. The quadrotors are built using the Pixhawk flight controller running custom flight controller software, and built using commercially available frames, motors, and motor controllers. The system planner is implemented in Matlab, and messages are passed between Matlab and the Pixhawk controller using the ROS framework over wifi. Each quadrotor receives position feedback from the motion capture system.

For the performance, the behavior commands are sent online to the system; while the rough order of events is dictated by story chronology, the performer-issued commands are given with no pre-determined time specification. The online planning framework described in this work is able to correctly modify trajectories for the half-dozen quadrotors used in the performance, validating that modification of current behaviors and planning time does not disrupt flight control. We also show that the generated plans never violate the specified safe inter-robot clearance distance or exceed specified acceleration limits.

All trajectories generated by the system must maintain a 0.5m center-center distance between all robots at all times as well as ensuring all robots stay within the boundaries of the designated flight volume. Further, generated trajectories must not violate maximum values of linear and angular acceleration, chosen based on hardware and flight volume limitations. Plots of minimum clearance distance and actuator limits as a result of the plans created by a user’s online instructions for the performance are shown in Figure 3.

Still images of the performance are shown in Figures 1 and 2 and a time lapsed video of the performance is available at http://www.andrew.cmu.edu/user/ecappo/rss16.mp4.

V. Conclusion and Future Work

The system outlined in this work is able to handle real-time demands for online choreography of multiple quadrotors, and the choice of the described behavior-level input allows users to interact at a level which allows detailed but relatively general command specification, appropriate for online use. A key feature of the described system is that the planner is allowed to modify user commands in order to keep the general intent even if the specific command may be dynamically infeasible (for example, moving specified quadrotors to a target location, but altering the timing of the instruction). This is critical for online use, as a user cannot easily determine specific dynamic constraints, and having the planner reject infeasible commands outright leads to large delays while the performer attempts to adjust the input command, which is undesirable from a theatrical performance perspective. We plan to use the work described here as a baseline system to further explore the question of how to understand a user’s intent, and how to communicate necessary modification back to the user using the described feedback methods. We are interested in learning how a performers input changes based on increased system feedback, as well as how to establish a learning methodology so that over prolonged use, the system can learn more intent-aware modifications for maintaining dynamic feasibility, to better facilitate online interaction between a human user and a dynamic multi-robot system.

Acknowledgments

We would like to thank the BeckonBot team, especially Eric Fang and Kevin Apolo for their assistance in assembling the quadrotor hardware and Eric Adlam and James Laney for their work with the gesture and user input systems.
REFERENCES


