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

Data Driven Online Multi-Robot Formation Planning

Ellen A. Cappo, Arjav Desai, and Nathan Michael

***Abstract*— This work addresses planning for multi-robot formations online in cluttered environments via a data-driven search approach. The user-specified objective function governing formation shape and rotation is expressed in terms of offline demonstrations of robot motions (performed in an obstacle free environment). We leverage the offline demonstration to inform online planning for coordinated motions in the presence of obstacles. We formulate planning as a discrete search over demonstrated multi-robot actions, and select actions using a best-first approach to minimize edge expansions for fast online operation. Actions are selected using a heuristic based on their probability distribution exhibited in the demonstration, and we show that this approach is able to recreate coordinated motions exhibited in the demonstration when navigating in the obstructed conditions of the cluttered test environments. We demonstrate results in simulation over environments with increasing numbers of obstacles, and show that resulting plans are collision free and obey dynamic constraints.**

I. Introduction

In this work we consider the problem of planning coordinated motions online for a group of robots in cluttered environments informed by the data generated through offline demonstration. Planning coordinated motions between agents is common across a wide range of multi-robot applications, including the cooperative transport or manipulation of objects [1], performing escort missions [2] or sensor coverage [3], and depicting characters or figures in entertainment applications [4–6]. However, planning kinodynamically feasible and collision free motions for multiple agents can be challenging because computational complexity scales with the number of agents and planning time is limited for applications that require online updates; increased environment complexity further complicates the planning problem because robots may not be able to reach or maintain the desired goal.

While the computational complexity of online multi-robot planning is a significant concern, the coordination problem itself can be challenging due to the difficulty in formulating a coordination objective. When the coordination goal is well understood or clearly expressed by quantifiable metrics, optimization approaches or rule-based policies are commonly used to plan team motions. For example, the common task of maintaining agents in formation is often formulated in terms of minimizing deviation to a desired formation, and can be solved via a constrained optimization employing sequential convex programming [1], using a vector field policy to avoid obstacles but maintain formation [7], or time-based polynomial trajectory planning over a receding horizon in order to maintain distance and bearing specifications between agents [8]. The coordination policy of these approaches (in this example, "minimize deviation from a desired reference") is specified by human understanding of the problem domain.

The authors are affiliated with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. <eacappo,adesai,nmichael>@cmu.edu We gratefully acknowledge support from industry.

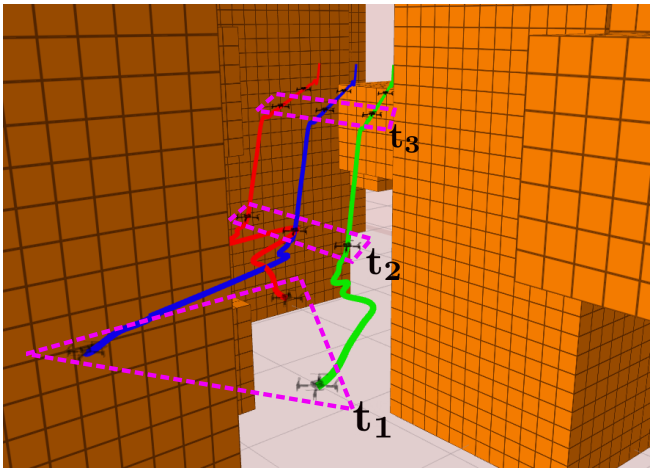


Fig. 1: Example of coordinated motions between robots in a formation being performed in a complex environment. The robot motions were selected during online search based on user-provided demonstration data. Robots are shown at sampled times, t_1, t_2, and t_3, along a set of trajectories generated through keyframes returned via the proposed search methodology. The robots transition from a triangle formation to a line formation in order to avoid collision with an obstacle (represented by the orange voxels).

However, some applications can require coordination policies that are difficult to express via optimizable metrics, or which do not have a clear best policy. In these cases, demonstrated solutions—if available—may be used to inform planning policies. In the example of team sports such as soccer where it may be unclear how players should perform defense, [9] uses video demonstration of human players to learn agent roles. In [10], no decentralized controller is available for the posed particle assignment problem, and so a centralized planning formulation is used as a demonstration to train decentralized policies for a multi-robot team.

In this work, we seek to plan coordinated motions reflecting user preferences for a team of robots in complex and cluttered environments without having to explicitly express user objectives through a rule-based methodology as in prior work [5, 6]. The methodology of [5, 6] allows a user to specify time-varying formation objectives online via parameterized input and create dynamically feasible and collision free multi-robot plans in the form of time-based polynomial trajectories in non-cluttered environments. However, directly finding dynamically feasible and non-colliding trajectories for the desired user input in cluttered environments is difficult to solve under online time constraints.

We propose to use plans reflecting user specified multi-

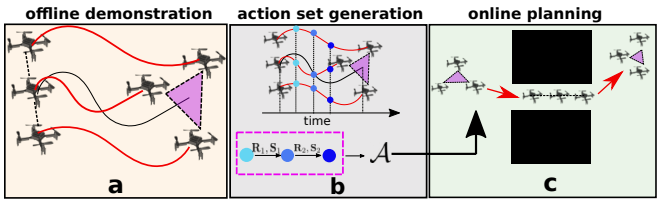


Fig. 2: This figure illustrates an overview of the approach. Block (a) shows an offline demonstration of a group of robots changing formation from a line to a polygon. Block (b) illustrates the demonstration, sampled at uniform time discretizations, represented as actions performed by the formation, described as rotations and shape transformations. These actions are stored in an action set, $\mathcal{A}$. Block (c) shows a group of robots in a cluttered environment, executing actions from action set $\mathcal{A}$ to transition from polygon to line to navigate between obstacles.

robot coordination preferences generated in an open environment via [5, 6] to inform online plan generation in cluttered and complex environments. The contribution of this approach is that it allows a system to reproduce user-preferred coordination strategies in non-demonstrated scenarios (the transfer of actions from open to cluttered environments), as well as perform coordinated multi-robot motions without requiring the direct specification of a rule-based objective function or direct online user-input.

II. Methodology Overview

This section provides a conceptual overview of the proposed approach, as visually depicted in Fig. 2. Our approach is divided into Offline and Online portions. We formulate multi-robot planning as a Markov Decision Process, where robots transition between states via actions, and introduce notation to describe a discrete representation of multi-robot states and actions (Sect. III). A demonstration provided by a user, showing user-desired motions for a formation of robots, is given in the form of a time series of robot positions. Sect. IV describes the representation of the demonstration as state-action tuples, and the creation of an action set composed of actions experienced in the demonstration.

Sect. V describes the online navigation of a group of robots through a cluttered environment. Finding a sequence of actions that allows the robot group to navigate the desired path can be formulated as search over a tree of possible actions, and to minimize edge expansions for fast search online, we propose a selection heuristic for use with a Best-First Search approach. We regard the provided demonstration as a user-generated probability distribution of actions, and propose the use of a simple, probability-based heuristic to select actions during search.

Sect. VI discusses metrics for evaluating action sequences compared to a demonstration, and Sect. VII evaluates the proposed approach through a series of experiments performed in varying simulated environments.

III. Notation and Problem Formulation

This section introduces notation and problem formulation. In the following descriptions, indexing elements are given as superscripts, while time elements are written as subscripts. For example, the state x of robot i at time t is written as x_t^i.

A. Robot, formation, and obstacle notation

The state of an individual robot, s, with respect to a local, formation reference frame $\mathbb{S}$ is given by its position, $\mathrm{s} = [x, y, z]^\mathrm{T} \in \mathbf{R}^3$. The state of a group of n robots is a vector containing the states of the member robots, $\mathbf{S} = [\mathrm{s}^1, \ \ldots, \ \mathrm{s}^n] \in \mathbf{R}^{3\times n}$. A valid formation, or shape $\mathbf{S}$, is one in which no robots are in collision with each other: i.e., a minimum separation distance, dr, is required between all robots in a formation: $\| \ \mathrm{s}^i - \mathrm{s}^j \ \|_2 \geq dr \ \forall \left(\mathrm{s}^i, \mathrm{s}^j\right) \in \mathbf{S}$, and $\| \cdot \|_2$ describes the Euclidean distance or l^2-norm.

Similar to state in the group reference frame, we define the state of a robot in world frame $\mathbb{W}$ as $\mathrm{x} = [x, y, z]^\mathrm{T}, \mathrm{x} \in \mathbf{R}^3$, and the state of a group of n robots as $\mathbf{X} = [\mathrm{x}^1, \ \ldots, \ \mathrm{x}^n], \ \mathbf{X} \in \mathbf{R}^{3\times n}$. The formation reference frame, $\mathbb{S}$, is related to $\mathbb{W}$ by a positional offset, $C \in \mathbf{R}^3$, and a rotation, $R \in \mathrm{SO}(3)$, so that state s of a robot in frame $\mathbb{S}$ is expressed in frame $\mathbb{W}$ as $\mathrm{x} = C + R\mathrm{s}$, and likewise for a formation of robots, $\mathbf{X} = C + R\mathbf{S}$.

We use a voxel-based environment representation, letting $\mathcal{O} \subset \mathbb{R}^3$ be the set of static obstacles (occupied voxel cells). However, the methodology described in this work holds for any chosen environment or obstacle representation, for example, defining obstacles $\mathcal{O}$ as a point set of observed surface data or collection of convex polygons, etc. We define $\mathcal{P}(\mathbf{X}_t)$ as the set of voxels occupied by the convex polytope defined by robot poses $\mathbf{X}$ at time t. A formation of robots is collision free if the set of voxels covered by the formation does not intersect the obstacle set, i.e., the intersection of the two sets is the empty set: $\mathcal{P}\left(\mathbf{X}\right) \cap \mathcal{O} = \{\}$.

B. MDP over states and actions

We formulate planning for a group of robots as a Markov decision process (MDP), where our state at time t is the tuple $\mathbf{s}_t = \{C_t, R_t, \mathbf{S}_t\}$, an action at time t is the tuple $\mathbf{a}_t = \{c_t, \Omega_t, A_t\}$, and the transition function $\mathcal{T}()$ gives $\mathcal{T}(\mathbf{s}_{t+1} | \ \mathbf{a}_t, \mathbf{s}_t)$ by:

$$C_{t+1} = C_t + c_t \tag{1}$$

$$R_{t+1} = \Omega_t R_t \tag{2}$$

$$\mathbf{S}_{t+1} = A_t \mathbf{S}_t. \tag{3}$$

In the above set of equations, $c_t \in \mathbf{R}^3$ describes a translation in $\mathbb{W}$ in x, y, and z; $A_t \in \mathbf{R}^{3\times3}$ describes a linear transformation of the n robot formation, $\mathbf{S}_t \in \mathbf{R}^{3\times n}$; and $\Omega_t \in \mathrm{SO}(3)$ describes an incremental rotation that takes R_t to R_{t+1}. We let $\mathcal{T}()$ be deterministic, transitioning $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$ given $\mathbf{a}_t$ with probability 1.

The problem definition is as follows: given the current system state $\mathbf{s}_t$ at time t, we seek to complete a path specification given only as components c and Ω of an action with the most appropriate shape space transform A to form the complete action tuple: $\mathbf{a}_t = \{c_t, \Omega_t, A_t\}$ to take the system to the desired (and feasible) state at the next timestep, $\mathbf{s}_{t+1}$. This may be stated as:

$$A_t = \underset{A_t \in \mathcal{A}}{\operatorname{argmax}} \ \mathrm{Reward}(A_t) \tag{4}$$

$$\text{s.t. } \mathcal{P}\left(\mathbf{X}_{t+1}\right) \cap \mathcal{O} = \{\} \tag{5}$$

$$\| \ \mathrm{s}^i - \mathrm{s}^j \ \|_2 \geq dr \quad \forall \left(\mathrm{s}^i, \mathrm{s}^j\right) \in \mathbf{S}_{t+1}, \ \mathrm{s}^i \neq \mathrm{s}^j. \tag{6}$$

Reward(A_t) is a reward function in terms of A_t, the linear transform describing the transition of $\mathbf{S}_t$ to $\mathbf{S}_{t+1}$, drawn from a set $\mathcal{A}$ of demonstrated transforms. We discuss the specifics concerning the generation of set $\mathcal{A}$ and details of reward calculation Reward(A_t) to Sects. IV and V, respectively.

IV. Demonstration data as $\{\mathbf{s}, \mathbf{a}\}$ tuples

To select a shape transform A_t as in Eqn. (4) in line with user expectations, we require a demonstration of user-directed robot motions represented as a series of state-action tuples. Here, we generate demonstrations using the methodology of [5, 6], which takes detailed user input to produce dynamically feasible, time-based polynomial trajectories for all robots. We therefore sample the demonstration at a chosen, uniform time discretization to create a time-series of robot states. The dynamic feasibility of the demonstration trajectories means that (1) no states contain robot-robot collisions, and (2) sampling at a fixed time resolution provides actions within a bounded set, as the demonstration trajectories generated by [5, 6] respect the provided actuator limits of the physical robot platform.

We begin with a demonstration of the form $\mathbf{X}_{1:T}$ in $\mathbb{W}$ for all robots in a group, where $\mathbf{X}_t$ has been sampled at $t = \{1, \ldots, T\}$, and $t_{i+1} = t_i + dt$. Given two sequential poses, $\mathbf{X}_t$ and $\mathbf{X}_{t+1}$, and the knowledge of either $\mathbf{S}_t$ or R_t, we can decompose $\mathbf{X}_t$ and $\mathbf{X}_{t+1}$ to MDP states $\mathbf{s}_t$ and $\mathbf{s}_{t+1}$ and the action $\mathbf{a}_t$ which transitions $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$. This is a Procrustes problem [11], where we seek a transformation that maps robots from $\mathbf{X}_t$ to $\mathbf{X}_{t+1}$, such that transformation Y minimizes $\| Y\mathbf{X}_t - \mathbf{X}_{t-1} \|$.

A. Position and translation components of states and actions

We equate the position component of state, C, to the geometric mean of robot positions: $\bar{\mathbf{X}}$. Given position components $C_t = \bar{\mathbf{X}}_t$ and $C_{t+1} = \bar{\mathbf{X}}_{t+1}$ at neighboring timesteps, translation term c_t is found by subtracting the geometric mean of the robot group at time t from $t+1$, a rearrangement of Eqn. (2): $c_t = C_{t+1} - C_t$.

B. Rotation component, Ω_t, of action $\mathbf{a}_t$

Knowledge of the position component lets us group the remaining state components, R_t and $\mathbf{S}_t$, together as P_t:

$$P_t = \quad R_t\mathbf{S}_t \quad = X_t - C_t \tag{7}$$

$$P_{t+1} = R_{t+1}\mathbf{S}_{t+1} = X_{t+1} - C_{t+1} \tag{8}$$

Given P_t and P_{t+1}, the rotation matrix Ω_t is the solution to the orthogonal Procrustes problem [11] : $\Omega = \text{argmin}_\Omega \| \Omega P_t - P_{t+1} \|_F$ subject to $\Omega^{\mathrm{T}}\Omega = \Omega\Omega^{\mathrm{T}} = I$ with $\det(\Omega) = 1$, where $\| \cdot \|_F$ denotes the Frobenius norm.

This problem is equivalent to finding the nearest special orthogonal matrix[1] to a given matrix $M = P_{t+1}P_t^{\mathrm{T}}$. To find Ω_t, singular value decomposition is used to write:

$$P_{t+1}P_t^{\mathrm{T}} = U\Sigma V^{\mathrm{T}} \tag{9}$$

$$\Omega_t = U\Sigma' V^{\mathrm{T}} \tag{10}$$

[1]An orthogonal matrix is a square matrix whose columns and rows are orthogonal unit vectors, i.e., $R^{\mathrm{T}}R = RR^{\mathrm{T}} = I$; a special orthogonal matrix is an orthogonal matrix whose determinant equals 1, $\det(R) = 1$.

where Σ' is a modified Σ with the smallest singular value replaced by sign($\det(UV^{\mathrm{T}})$), i.e. +1 or -1, and the other singular values replaced by 1, so that the determinant of Ω is guaranteed to be positive [12].

C. Discussion of R_0 and $\mathbf{S}_0$

Poses P_t and P_{t+1} are known from Eqns. (7) - (8), and we would like to decompose this information into four state components, R_t, $\mathbf{S}_t$, R_{t+1}, $\mathbf{S}_{t+1}$. We have so far expressed only one constraint: that R_t evolves to R_{t+1} via pure rotation. This allows us to express R_{t+1} fully in terms of R_t via Ω_t (Eqn. (10)) exactly as in Eqn. (3). We therefore have two knowns, P_t and P_{t+1}, and three unknowns, R_t, $\mathbf{S}_t$, and $\mathbf{S}_{t+1}$, and so require one additional piece of information which may be provided by the knowledge of either the starting shape, $\mathbf{S}_0$, or the starting orientation, R_0. The demonstration may be assumed to begin with R_0 equal to identity, or alternatively, $\mathbf{S}_0$ may be defined via knowledge of the user's desired shape or by matching the observed positions of the robots at $t = 0$ to a library of desired formations.

D. Rotation and shape components of state

Without loss of generality, we use poses $\mathbf{P}_{t=0}$ and $\mathbf{P}_{t=1}$ to clarify the explanation of determining state rotation and shape components.

We first examine the case where R_0 has been specified. Given R_0, Eqn. (7) may be directly solved for $\mathbf{S}_{t=0}$:

$$\mathbf{S}_{t=0} = R_0^{\mathrm{T}}P_0, \tag{11}$$

and the evolution of $R_{t=0}$ by $\Omega_{t=0}$ (Eqn. (10)) gives $R_{t=1}$, yielding $\mathbf{S}_{t=1}$:

$$\mathbf{S}_{t=1} = R_1^{\mathrm{T}}P_1. \tag{12}$$

If $\mathbf{S}_0$ has been specified, R_0 may be found similarly as to Eqn. (10), as:

$$P_0\mathbf{S}_0^{\mathrm{T}} = U\Sigma V^{\mathrm{T}} \tag{13}$$

$$R_0 = U\Sigma' V^{\mathrm{T}}, \tag{14}$$

and the solution of $\mathbf{S}_{t=0}$ and $\mathbf{S}_{t=1}$ proceeds as in Eqns. (11) and (12).

While we have used poses at $t = 0$ and $t = 1$ for explanation, it should be clear that knowledge of R or S at any time t enables the calculation of all components of state at $t+1$. States at all times may therefore be calculated sequentially from knowledge state $\mathbf{s}_{t=0}$.

E. Shape transform A_t

The final component of action $\mathbf{a}_t$ undefined is transform A_t. This is, again, a Procrustes problem, although here unrestricted to a rotation matrix and so may be solved through the least squares minimization given by [11] (additionally as noted in the example implementation of [13]):

$$A_t = \mathbf{S}_{t+1}\mathbf{S}_t^{\mathrm{T}}(\mathbf{S}_t\mathbf{S}_t^{\mathrm{T}})^{-1}. \tag{15}$$

Due to the described extraction of rotation matrix Ω_t from P_t, P_{t+1}, transform A_t may be interpreted as scaling and shearing elements, where scale may be given along the x, y, and z dimensions and with the off-diagonal components

describing shearing as:

$$\begin{bmatrix} s_{xx} & 0 & 0 \\ 0 & s_{yy} & 0 \\ 0 & 0 & s_{zz} \end{bmatrix}, \quad \begin{bmatrix} 1 & s_{xy} & s_{xz} \\ s_{yx} & 1 & s_{yz} \\ s_{zx} & s_{zy} & 1 \end{bmatrix}. \tag{16}$$

We make no assumptions about the uniformity of scaling, meaning that $s_{xx} \neq s_{yy} \neq s_{zz}$, nor symmetry of shearing, i.e., $s_{xy} \neq s_{yx}$, etc, reiterating that $A \in \mathbf{R}^{3\times3}$ or equivalently $A \in \mathbf{R}^9$ for the values $\{s_{xx},\ s_{yy},\ s_{zz},\ s_{xy}\, s_{yx},\ s_{xz}\, s_{zx},\ s_{yz}\, s_{zy}\}$.

Transforms A_t observed from demonstration $\mathbf{X}_{1:T}$ may be discretized (for example, casting all values to a desired resolution, dA) so that actions can be recognized as discrete and unique. A set of possible actions, $\mathcal{A}$ used in Eqn. (4), may then be formed of all unique actions from the demonstration.

V. Online action search

This section describes the formulation of action selection as a tree search among possible actions, and the introduction of a simple reward measure for estimating user-preferred actions.

A. MDP as tree search

The process of selecting and evaluating the possible actions from a given state may be formulated as search over a directed graph, specifically a tree, where nodes represent states and edges represent actions. A node stores state $\mathbf{s} = \{C, R, \mathbf{S}\}$ and collision information with respect to obstacle set $\mathcal{O}$. Each edge represents a single action, $A \in \mathcal{A}$.

The root node is initialized at the start position in the world, $C_{t=0}$, with the starting rotation, $R_{t=0}$, and formation state, $S_{t=0}$. At each node, a maximum of $|\mathcal{A}|$ possible decisions can be made (i.e., a node has a maximum of $|\mathcal{A}|$ possible children), where $|\mathcal{A}|$ is the size of action set $\mathcal{A}$.

Child node state is found by applying the transition function (Eqns (1)-(3)). Therefore children share action components c_t and Ω_t, resulting in shared state components C_{t+1}, R_{t+1}, but have different shape transforms A_t and so different shapes $\mathbf{S}_{t+1}$. Every child node is checked for collisions with the environment as well as for collisions between robots, and nodes in collision are marked as inadmissible.

In order to find actions quickly to meet online planning time requirements, we propose a Best First Search (BFS) approach [14] to determine a sequence of shape transforms, $A_{1:T}$, for a user input $\{c_{1:T}, \Omega_{1:T}\}$. BFS for our application is summarized in Alg. 1. Search may be terminated under three conditions: (1) when the end of the horizon is reached, $t = T$; (2) the frontier list (Algorithm 1) is empty, meaning all non-colliding nodes have been explored; (3) we have reached a maximum number of node expansions, capped so as to limit online planning time.

Algorithm 1: Best First Search [14] formulated for our application.

returns `TreePath` or failure
$root \longleftarrow$ node with formation start state and zero reward
frontier $\longleftarrow$ $root$ // queue ordered by cumulative reward
explored $\longleftarrow$ {} // empty list
while $i < limit$ **do**
  if `Empty`(frontier) **then return** failure
  $node \longleftarrow$ `Pop`(frontier) // highest cumulative reward
  if $node.depth == T$ **then return** `TreePath`$(root, node)$
  explored $\longleftarrow$ $node$
  $Children \longleftarrow$ `Expand`$(node)$
  `CollisionCheck`$(Children)$
  `CumulativeReward`$(Children)$
  frontier $\longleftarrow$ `Insert`$(Children \setminus (explored \cup frontier))$
end

B. Selection policy

We suggest that the reward of a node representing transform A_t be based on the probability of seeing A_t as part of a string of actions from the demonstration. We express this by defining a window around timestep t using two parameters, m and h, where m defines the number of actions prior to, and h defines the number of actions following, A_t.

$$\text{Reward}(A_t) = p(A_{t:t+h}|A_{t-m:t-1}) + t. \tag{17}$$

Given many nodes with similar or equal reward, we prefer to expand nodes that move closer to the end of the path. As search depth is correlated with timestep t, we therefore add t to the reward based on $p(A_t)$.

VI. Evaluation Metrics

This section describes three measurable attributes that allow us to evaluate a search solution, a string of shape transforms over a time horizon T, $A_{1:T}$, with respect to a demonstration. The solution is a string (a finite sequence of symbols chosen from an alphabet, a finite set of symbols), where each symbol is a unique transform, $A^i \in \mathcal{A}$, and the size of the alphabet is the chirality of set $\mathcal{A}$, $|\mathcal{A}|$.

Longest Common Substring (LCS): One measure for determining the extent to which a solution exactly matches (a portion of) the demonstration can be expressed by the Longest Common Substring (LCS). Given two strings F and G, of lengths L_F and L_G respectively, the LCS is the longest string which is a substring of both F and G.

We additionally look at two possible metrics for scoring the "distance" of a solution from the demonstration.

Minimum Demonstrated Hamming Distance (MDHD): Edit distance describes the number of operations required to transform one string to another. The Hamming distance between two strings of equal length (we denote this as $d_{\text{H}}(\cdot,\cdot)$) is the number of positions at which the corresponding symbols differ [15], and in our application, gives a measure of how well the solution could have done (ignoring the feasibility requirements which constrained solution choice) to exactly replicate a portion of the demonstration.

The Hamming distance can only be computed between strings of equal length. For a demonstration string F of length L_F, and a solution string G with length L_G, there will be $(L_F - L_G + 1)$ possible L_G-length substrings from the demonstration. We denote the Minimum Demonstrated Hamming Distance[2] (MDHD) between the query string, G,

[2]Our defined "Minimum Demonstrated Hamming Distance," identifying the minimum distance of a query string to a string set, is unrelated to "minimum Hamming distance" as defined in the literature, the smallest Hamming distance between all possible pairs of strings in a set [15].

and the set of substrings, f^i with length $L_f = L_G$, drawn from the demonstration string, F, as:

$$f^i \subseteq F, \quad L_f = L_G \tag{18}$$

$$f^* = \underset{f^i \subseteq F}{\operatorname{argmin}} \ \mathrm{d_H}(f^i, G) \tag{19}$$

$$\mathrm{MDHD} = d_{\mathrm{H}}(f^*, G). \tag{20}$$

Minimum Demonstrated JSD (MDJSD): While it ignores action ordering, we can also view a demonstration and a solution as two probability distributions over actions. A common metric for measuring the similarity between two probability distributions is the Jensen–Shannon divergence (JSD), also known as the information radius or total divergence to the average [16]. The JSD is a symmetrized and smoothed version of the Kullback–Leibler divergence. We let F be the string of demonstration data and G the solution, and we use f_i, g_i, to denote the observed frequencies of A^i from the demonstration and the solution strings. The Kullback–Leibler divergence between strings F and G is then $\mathrm{KLD}(F||G) = \sum_{i=1}^{|\mathcal{A}|} f_i \log_2\left(\frac{f_i}{g_i}\right)$, and the JSD is given as:

$$F = \{f_i, \ \ldots, \ f_M\}, \ G = \{g_i, \ \ldots, \ g_M\} \tag{21}$$

$$M = \frac{1}{2}(F + G) \tag{22}$$

$$\mathrm{JSD}(F||G) = \frac{1}{2}\mathrm{KLD}(F||M) + \frac{1}{2}\mathrm{KLD}(G||M), \tag{23}$$

Comparing a solution to an entire demonstration is uninformative when $L_F \gg L_G$; we do not expect the solution to well represent an entire demonstration, but we would hope that the solution well represents some portion of the demonstration. Therefore, as in our calculation of the MDHD (Eqns. (19)–(20)), we calculate JSD with respect to the set of all L_G-length substrings of the demonstration to find a Minimum Demonstrated Jenson-Shannon divergence (MDJSD):

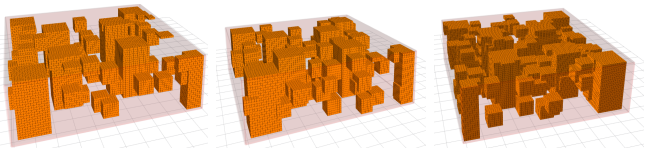
$$f^i \subseteq F, \quad L_f = L_G \tag{24}$$

$$f^* = \underset{f^i \subseteq F}{\operatorname{argmin}} \ \mathrm{JSD}(f^i||G) \tag{25}$$

$$\mathrm{MDJS} = \mathrm{JSD}(f^*||G). \tag{26}$$

VII. Evaluation

In this section we evaluate the proposed search approach using the metrics proposed in Sect. VI over several environments of varying complexity, and additionally examine the dynamic feasibility of trajectories interpolated through the discrete states returned from search.



(a) "Low" clutter, ≈24% occupied **(b)** "Medium" clutter, ≈28% occupied **(c)** "High" clutter, ≈32% occupied

Fig. 3: Environments of varying complexity used in evaluation. Environment labels of "low, med, high" corresponding to Subfigures (a), (b), and (c), respectively, correspond to Table 1. Environments span 10m × 10m × 3m in height, and are shown here with a voxel resolution of 0.125m.

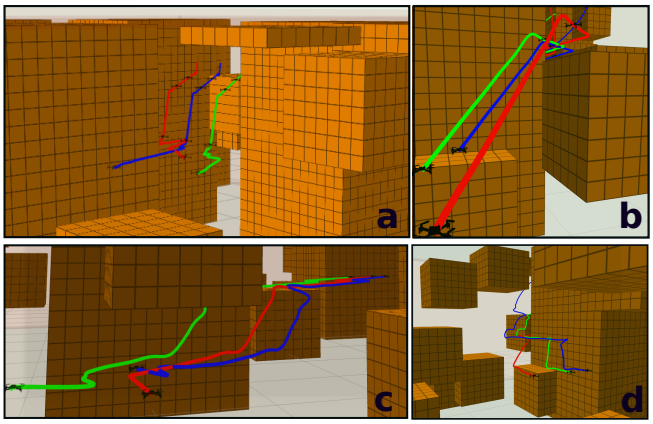


Fig. 4: These four images show representative examples of coordinated actions performed by a formation of robots during search evaluation.

Our methodology returns plans in the form of a non-colliding sequence of goal states (positions) for the robot group. To use the proposed approach with real robots, we require dynamically feasible plans trackable by a hardware platform. Some approaches use independent robot controllers to directly navigate to the goal destinations output from formation planning [1]; although this approach would also work for our problem formulation, we prefer to evaluate the dynamic feasibility of our plans by using the states returned from search as keyframes in a polynomial trajectory formulation. This allows us to quickly evaluate the dynamic feasibility and safety of the entire plan quickly online.

In the following experiments, we use the method of [17] to fit time-based polynomial trajectories to the discrete states returned from search. We evaluate plans for use with a quadrotor system requiring trajectories that are smooth and continuous up to the acceleration limits of the platform, and results are shown in Fig. 5. For each trial, robots are randomly placed in the environment and asked to follow a randomly-generated, set-length path. Table 1 shows the solution metrics evaluating the discrete solution returned from search (Sect. VI). We evaluated several values of m and h to describe the length of the windowed history used in Eqn. 17, and found that using a small window of past actions performed best in terms of solution quality and computation time; we therefore show results for trials conducted with search parameters $m = 1$ and $h = 0$.

The results in Table 1 illustrate that the proposed methodology is able to return solutions that exactly reflect portions of the demonstration even in mildly cluttered environments, as shown in Line 1. As environments become increasingly cluttered, we see that as expected, solutions are forced to deviate in greater degrees from any portion of the demonstration, and that there are a greater number of failure cases where a formation cannot fit along an intended path. However, even in the most cluttered environment, approximately 80% of the solution sequence reflects a portion of the demonstration (the LCS of Line 5 is approximately 80% of the action sequence length).

We additionally show results for the same environments represented at alternate voxel resolutions. We anticipated

Table 1: Performance over varying environments. Action sequence length = 19, $|\mathcal{A}| = 92$, $dA = .01$, $dt = .25$, $m = 1$, $h = 0$. Environments "low", "med", and "high" are shown in Figs. 3a, 3b, 3c. The voxel resolution of "coarse" is set to 0.25m, and "fine" is set to 0.125m. We report the average values over 25 trials at each environment and voxel resolution.

	Average LCS	Average MDHD	Average MDJSD	Average search time [s]	Average graph size [nodes]	Failed trials
1. Env: low, res: fine	19.000	0.000	0.000	2.196	1157.818	3
2. Env: low, res: coarse	17.227	1.545	0.044	2.184	1124.455	3
3. Env: med, res: fine	18.619	0.381	0.012	2.096	1140.286	4
4. Env: med, res: coarse	17.095	1.476	0.041	2.180	1103.476	4
5. Env: high, res: fine	15.579	2.947	0.079	2.060	1052.895	6
6. Env: high, res: coarse	11.263	6.684	0.196	1.998	1081.368	6

that searching a coarser resolution would perform faster, and we do see that in general, coarser representations show slightly lower search times and graph sizes. However, the performance differences in search time and graph size with respect to voxel resolution are minor, showing that the method maintains performance over higher-fidelity environment representations. This is beneficial in avoiding the decreasing solution quality incurred by coarser voxelization; with coarser representations, fewer solutions are found due to the reduced available free space.

Search times reported in Table 1 are reported as the (average) total search time to find actions along the entire specified path (including all collision checks). Trajectories generated from the discrete states returned from search roughly span a time duration on the order of sample resolution, dt, multiplied by the number of actions in the solution sequence. The search time required is therefore sufficiently less than the travel time of the robots that the methodology can safely evaluate and generate trajectories for online use. We further note that our implementation was performed in MATLAB, and that transitioning to alternate programming languages or libraries will decrease search time.

VIII. Conclusion and Future Work

The proposed approach contributes the capability to coordinate multi-robot formation planning guided by demonstration, allowing us to reflect user preferences or instructions that are not easily input during online operation. This affords us the ability to coordinate actions with respect to a wider variety or more unstructured representation of user preferences–as represented by the distribution of actions in a demonstration–rather than being restricted to minimizing a rule-specified objective function. Simulation results validate that the proposed method is able to find high-resolution solutions over an extended time horizon within a computation budget that allows for online operation, even in complex and cluttered environments.
function proved difficult to evaluate within the desired online

As future work, we intend to incorporate formation state and environment features into action selection. We anticipate that we can improve the quality of action selection and increase search speed by reducing the number of required collision checks, as we learn which actions are likely to produce collisions with respect to environment features. We additionally intend to evaluate alternate search approaches which might improve action selection while remaining within the desired online time budget. For example, while incorporating a look-ahead parameter in our proposed reward

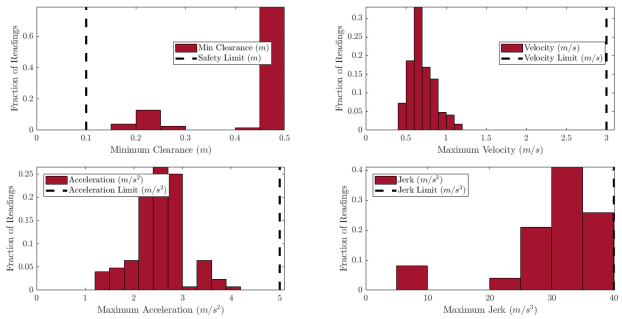


Fig. 5: Trajectory characteristics showing dynamic feasibility: tests employ robots of radius 0.1 m, and acceleration and jerk limits of 5 m/s^2, and 40 m/s^3, respectively.

operating time for the size of the search set, intelligent selection of likely nodes might mean that we can afford a limited look-ahead over a subset of options to improve plan quality.

References

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

[2] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The entrapment/escorting mission," *IEEE Robot. Autom. Mag.*, vol. 15, no. 1, pp. 22–29, 2008.

[3] P. Dasgupta, T. Whipple, and K. Cheng, "Effects of multi-robot team formations on distributed area coverage," *Intl. J. of Swarm Intelligence Research (IJSIR)*, vol. 2, no. 1, pp. 44–69, 2011.

[4] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Multi-robot system for artistic pattern formation," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2011, pp. 4512–4517.

[5] E. A. Cappo, A. Desai, and N. Michael, "Robust coordinated aerial deployments for theatrical applications given online user interaction via behavior composition," in *Int. Symp. on Dist. Auton. Robotics Syst.* Springer, Cham, 2016, pp. 665–678.

[6] E. A. Cappo, A. Desai, M. Collins, and N. Michael, "Online planning for human – multi-robot interactive theatrical performance," *Auton. Robots*, vol. 42, pp. 1771–1786, Dec. 2018.

[7] D. Zhou, Z. Wang, and M. Schwager, "Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 916–923, 2018.

[8] M. Turpin, N. Michael, and V. Kumar, "Decentralized formation control with variable shapes for aerial robots," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.* IEEE, 2012, pp. 23–30.

[9] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *Proc. of the 34th Intl. Conf. on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1995–2003.

[10] Q. Li, X. Du, Y. Huang, Q. Sykora, and A. P. Schoellig, "Learning of coordination policies for robotic swarms," *arXiv preprint arXiv:1709.06620*, 2017.

[11] J. C. Gower and G. B. Dijksterhuis, *Procrustes problems*. Oxford University Press on Demand, 2004, vol. 30.

[12] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-d rigid body transformations: a comparison of four major algorithms," *Machine vision and applications*, vol. 9, no. 5-6, pp. 272–290, 1997.

[13] M. K. Chung, *Statistical and computational methods in brain image analysis*. CRC press, 2013.

[14] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*. Malaysia; Pearson Education Limited, 2016.

[15] T. Yamada, "Principles of error detection and correction," in *Essentials of Error-Control Coding Techniques*. Elsevier, 1990, pp. 11–37.

[16] A. Mehri, M. Jamaati, and H. Mehri, "Word ranking in a single document by Jensen–Shannon divergence," *Physics Letters A*, vol. 379, no. 28-29, pp. 1627–1632, 2015.

[17] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Springer, 2016, pp. 649–666.