Hierarchically Consistent Motion Primitives for Quadrotor Coordination

Marijan Vukosavljev, Angela P. Schoellig, and Mireille E. Broucke

Abstract—We present a hierarchical framework for motion planning of a large collection of agents. The proposed framework starts from low level motion primitives over a gridded workspace and provides a set of rules for constructing higher level motion primitives. Our hierarchical approach is highly scalable and robust making it an ideal tool for planning for multi-agent systems. Results are demonstrated experimentally on a collection of quadrotors that must navigate a cluttered environment while maintaining a formation.

I. INTRODUCTION

This paper proposes a hierarchical construction of motion primitives for motion planning of multi-vehicle systems. The framework allows designers to incrementally build up more complex motion primitives from simpler ones in a systematic, rigorous way, without the need to redesign feedback controllers. This approach dramatically reduces the overall complexity of motion planning.

The main idea is simple. Consider a control system with two outputs. Suppose the output space has been gridded into boxes, and we have feedback controllers for a finite set of *atomic motion primitives*. In Figure 2 we show five atomic motion primitives. Motion primitives at level 1 are formed from sets of sequences of Level 0 motion primitives. For example, two successive $Right$ atomic motion primitives from level 0 form the $TwoRight$ motion primitive at level 1. Progressively more complex behaviors can be similarly defined, as shown in Figure 2.

Hierarchy is a common theme to simplify and scale up control design and has many applications [4], [9], [12], [19]. Motion primitives often serve as an important layer in the hierarchy and have been applied in various forms [6], [7], [16]. Abstraction as it relates to hierarchy has been studied in more general settings [1], [3], [8], [21], [25], but can be difficult to apply to practical applications. One of our motivating applications is efficient formation coordination of quadrotors, which boasts a broad literature [2], [5], [10], [14], [15], [18], [20].

Though our main idea is simple, and while many concepts and methods involving hierarchy, abstractions, and quadrotor control have been well studied, we place extra demands on our design that take it a step beyond what has been done. First, we allow for an arbitrary number of hierarchical levels, and the design of any level depends in the same way only
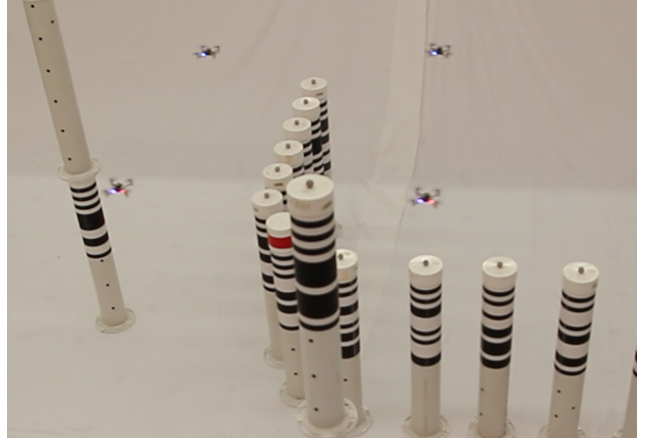
Fig. 1. Four tiny Crazyflie quadrotors navigate a cluttered environment while maintaining a square formation. A video illustrating the results can be found at http://tiny.cc/hier-moprim.

on the level below. One of the benefits of this uniformity of the architecture is that a designer can apply a planning algorithm at any level to obtain a control synthesis to the problem. Second, motion primitives at any level must be implementable by the low level continuous dynamics, for instance by adhering to safety and continuity constraints on positions and velocities. Effectively, we are implementing a notion of hierarchical consistency, but with the addition of a continuous state feedback at the lowest level [8]. As a final constraint, we expect a design that gives dramatic computational advantages, especially in the multi-agent setting. To this end, we construct hierarchical motion primitives on a grid, as shown in Figure 2.

There are three main contributions of this paper. First, we provide an architectural formulation of hierarchical motion primitives, which involves careful attention to suitable, parsimonious data structures and relationships. Second, we solve a reach-avoid problem with *behavior constraints*, in which the vehicles must safely reach goal locations while maintaining predefined sequences of motions; here our analysis employs fairly standard notions from hybrid control theory. Finally, we supply an elegant and highly efficient design to address the formation control of quadrotors. For example, we are able to navigate a formation of quadrotors interleaving them around a set of obstacles, while maintaining strict safety guarantees.

This research is an outgrowth of our modular framework for motion planning based on motion primitives [22], [23]; the notions of atomic motion primitives and the *maneuver automaton* were introduced there. As those constructions
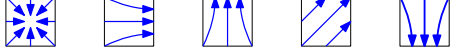
Level 0

Motion primitives

"Hold" "Right" "Up" "Right-Up" "Down"
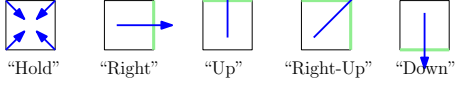
Implementation (closed-loop vector fields)

Level 1

Motion primitives

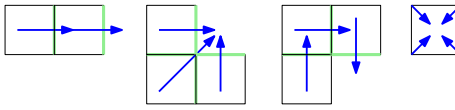"Two Right" "Right and Up" "U-Turn" "Hold"
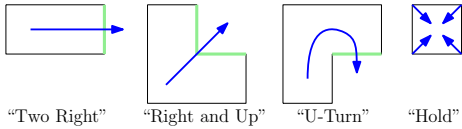
Implementation (using Level 0)

Fig. 2. An example of motion primitives designed at two hierarchical levels over a gridded two dimensional output space.

were limited only to level 0, this paper concerns itself with describing how to build hierarchical motion primitives at higher levels and illustrating the computational advantages of employing a multi-level hierarchy. Moreover, we have introduced the notion of behavior constraints in addition to the usual reach-avoid problem as a mechanism to address the formation control problem of quadrotors in a hierarchical manner. At the same time we suppress theoretical details including proofs, which will be made available in a forthcoming journal paper.

Notation. Let $|\cdot|$ denote the cardinality of a set. If A is a set, let $\mathcal{P}(A)$ denote its power set. The set difference of A and B is denoted $A \setminus B$. For a collection of sets $\{A_i\}_{i=1}^n$, the cartesian product is denoted $\prod_{i=1}^n A_i$; when $n = 2$, we may write $A_1 \times A_2$, and when $A_i = A$ for all $i = 1, \ldots, n$, we may write A^n (which in general should not be confused with superscripts for indexing in other contexts). Let $\mathbb{Z}$ and $\mathbb{R}$ denote the integers and the real numbers, respectively.

II. PROBLEM STATEMENT

Consider the general nonlinear control system

$$\dot{x} = f(x, u), \qquad y = h(x), \qquad (1)$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^\mu$ is the input, and $y \in \mathbb{R}^p$ is the output. Let $y(\cdot, x_0)$ denote the output trajectory of (1) starting at initial condition $x_0 \in \mathbb{R}^n$ and under some open-loop or feedback control. We make a standing assumption that the outputs of (1) are a subset of the states and that the vector field is invariant to the value of the output. This symmetry property is satisfied for many robotic systems, for example when the outputs are positions, and it allows motion

primitives to be designed only over a single box in a gridded output space [6], [23].

Fix a *grid length vector* $d = (d_1, \ldots, d_p)$ where $d_i > 0$. The gridded output space is constructed by translating the canonical box $Y^\star := \prod_{i=1}^p [0, d_i]$. That is, associated with each $l \in \mathbb{Z}^p$ is a shifted box $Y_l := \prod_{i=1}^p [l_i d_i, (l_i + 1) d_i]$. We call $l \in \mathbb{Z}^p$ the *box index* of Y_l. We identify a (non-empty) set of feasible boxes in terms of their box indices $L_f \subset \mathbb{Z}^p$. The feasible boxes arise from control specifications including obstacle avoidance, collision avoidance, communication constraints, and others. Similarly, we identify a (non-empty) set of goal boxes in terms of their box indices $L_g \subset L_f$.

Now consider an output trajectory $y(\cdot, x_0)$ for some control and initial condition $x_0 \in \mathbb{R}^n$. We associate to $y(\cdot, x_0)$ a discretized trajectory called a *run* that records the boxes that the output trajectory visits; see [11] for a formal definition. The run is denoted as $y_r := l_1 l_2 \cdots$, where $l_i \in \mathbb{Z}^p$ is a box index. We define the *behavior* induced by $y(\cdot, x_0)$ to be the sequence of box index increments $y_b := \kappa_1 \kappa_2 \cdots$, where $\kappa_i := l_{i+1} - l_i$. Because y_r records every box visited by the output trajectory, we have that $\kappa_i \in \{-1, 0, 1\}^p \setminus \{0\}$. That is, the increment in any coordinate is of magnitude at most 1, and the overall increment is never 0. Let $\mathcal{B}$ denote the empty sequence and all finite and infinite sequences on $\{-1, 0, 1\}^p \setminus \{0\}$. We define a *behavioral constraint* $\widehat{\mathcal{B}} \subset \mathcal{B}$ to be any non-empty subset of $\mathcal{B}$.

Problem 2.1 (Behavior-Constrained Reach-Avoid): Consider the system (1) with a gridded output space in terms of grid length vector $d \in \mathbb{R}^p$. We are given goal boxes L_g, feasible boxes L_f, and a behavioral constraint $\widehat{\mathcal{B}} \subset \mathcal{B}$. Find a feedback control $u(x)$ and a set of initial conditions $\mathcal{X}_0 \subset \mathbb{R}^n$ such that for each $x_0 \in \mathcal{X}_0$:

(i) **Avoid**: $y(t, x_0) \notin \mathbb{R}^p \setminus \left(\bigcup_{l \in L_f} Y_l \right)$ for all $t \geq 0$,

(ii) **Reach**: there exists $T \geq 0$ such that for all $t \geq T$, $y(t, x_0) \in \bigcup_{l \in L_g} Y_l$,

(iii) **Behavior**: $y_b \in \widehat{\mathcal{B}}$.

III. HIERARCHICAL MOTION PRIMITIVES

In this section we present the hierarchical construction of motion primitives. First we recall the definition of a maneuver automaton (MA) from our previous work [22], [23], as it serves as the level 0 or base case of the construction.

A. Maneuver Automaton

A *maneuver automaton* (MA) is a hybrid system whose discrete modes correspond to motion primitives. Each level 0 motion primitive has associated to it a continuous time closed-loop vector field obtained by applying a state feedback law to (1). The edges of the MA model feasible successive motion primitives. A motion primitive generates some desired behavior of the output trajectories of the closed-loop system over a box in the output space. Because of the uniform gridding of the output space into boxes and because of the symmetry assumption on the outputs, level 0 motion primitives are designed only over $Y^\star$. The MA applies state resets to ensure that output trajectories remain in $Y^\star$. Real,

physical trajectories clearly do not undergo such resets and instead output trajectories move continuously from box to box.

Definition 3.1: Consider the system (1) and the box $Y^\star$ with grid length vector d. The level 0 *maneuver automaton* (0-MA) is a tuple $\mathcal{H}^0 = (Q^0, \Sigma^0, E^0, I^0, Q^{0,0}, X^0, G^0, R^0)$, where

State Space $Q^0 = M^0 \times \mathcal{X}^0$ is the hybrid state space, where $M^0 = \{m_1^0, m_2^0, \ldots\}$ is a finite set of level 0 motion primitives, and $\mathcal{X}^0 = \mathbb{R}^n$ is the continuous state space.

Labels $\Sigma^0 = \{(0, \kappa) \mid \kappa \in \{-1, 0, 1\}^p\}$ is a finite set of event labels, where $\sigma = (0, \kappa) \in \Sigma^0$ describes a possible neighbouring direction κ from the p-dimensional box $Y^\star$ at location 0.

Edges $E^0 \subset M^0 \times \Sigma^0 \times M^0$, is a finite set of edges, describing allowable concatenations of motion primitives.

Invariants $I^0 : M^0 \to \mathcal{P}(\mathbb{R}^n)$ assigns a bounded invariant set $I^0(m)$ to each $m \in M^0$ that defines the region on which the vector field $X^0(m)$ is defined.

Initial Conditions $Q^{0,0} \subset Q^0$ is a set of initial conditions.

Vector Field $X^0 : M^0 \to \{f_m^0\}_{m \in M^0}$ assigns to each $m \in M^0$ a globally Lipschitz closed-loop vector field of the form $f_m^0(\cdot) = f(\cdot, u_m(\cdot))$, where $u_m(\cdot)$ is the state feedback controller associated with $m \in M^0$.

Enabling Conditions $G^0 : E^0 \to \{g_e^0\}_{e \in E^0}$ assigns to each edge $e = (m, \sigma, m') \in E^0$ a non-empty enabling or guard condition $g_e^0 \subset I^0(m)$.

Reset Conditions $R^0 : E^0 \to \{r_e^0\}_{e \in E^0}$ assigns to each edge $e = (m, \sigma, m') \in E^0$ a reset map $r_e^0 : \mathbb{R}^n \to \mathbb{R}^n$. Resets of states are determined by the event $\sigma \in \Sigma^0$, and they only affect the output coordinates in order to maintain output trajectories inside the canonical box $Y^\star$. ◁

The semantics of a level 0 MA involve the standard notion of an *execution* of a hybrid system [13], [23]. Informally, the continuous state evolves over the invariant of the current discrete mode $m \in M^0$ according to the vector field assigned to m, and an edge is taken when the continuous state reaches an enabling condition, thereby updating the discrete mode as well as resetting the continuous state. We shall denote a level 0 execution as $\chi^0 = (\tau^0, m^0, x^0)$, where τ^0 is a *hybrid time domain* consisting of time intervals (e.g. $\tau^0 = \{[0, 2], [2, 3.5], \ldots\}$), m^0 is a sequence of level 0 motion primitives recording the motion primitive over each time interval, and x^0 is the continuous state as a function of time.

B. Higher Level Maneuver Automata

We want to extend the notions of motion primitives, maneuver automata, and executions of an MA from level 0 to higher levels by a hierarchical construction that builds level k using only information from level $k-1$. We begin with the discrete part of $\mathcal{H}^0$ since its extension to level $k > 0$ is straightforward.

The discrete part of $\mathcal{H}^0$ is given by the tuple (M^0, Σ^0, E^0), representing a graph with nodes M^0 (the level 0 motion primitives), edges E^0, and labels Σ^0. Analogously, the discrete part of the level $k > 0$ MA, $\mathcal{H}^k$, is a tuple (M^k, Σ^k, E^k),

representing a graph over level k motion primitives. Because level k motion primitives may be defined over more boxes than just $Y^\star$, the event labels Σ^k on transitions must be accordingly generalized.

For example, consider the level 1 motion primitive in Figure 2 called *U-Turn*. It is constructed by concatenating three level 0 motion primitives: *Up*, *Right*, and *Down*. *U-Turn* may, in turn, be concatenated with other level 1 motion primitives encoded by an edge $e = (m, s, m') \in E^1$ where $m \in M^1$ denotes *U-Turn* and $m' \in M^1$ denotes a possible next level 1 motion primitive. First we attach a reference frame in the gridded output space for primitive m with an arbitrary base point o_m, say the lower left box. The role of $s = (l, \kappa) \in \Sigma^1 \subset \mathbb{Z}^2 \times \{-1, 0, 1\}^2$ is then to encode the fact that the transition from m to m' occurs downwards from the upper right box Y_l. Therefore $s = ((1, 1), (0, -1))$.

Now consider the continuous time part of $\mathcal{H}^0$. It consists of a continuous state space $\mathcal{X}^0 = \mathbb{R}^n$, a set of continuous time closed-loop vector fields X^0, a set of invariants I^0 that specify the domain of each vector field, and the enabling conditions G^0 and reset maps R^0 that specify how continuous states must be reset upon reaching an enabling condition. These notions are extended to level k as follows.

The continuous state space $\mathcal{X}^0$ is reinterpreted at level k as a (discrete) state space $\mathcal{X}^k = \mathbb{Z}^p \times M^{k-1}$, consisting of the pairs (l, μ), with $l \in \mathbb{Z}^p$ identifying a box index (or offset) and $\mu \in M^{k-1}$ identifying a level $k-1$ motion primitive. Similarly, the notion at level 0 of a continuous state trajectory, which is a function of time, is replaced at level k by a discrete sequence of (l, μ) pairs.

The continuous time vector fields at level 0 are replaced by a set of discrete maps. For motion primitive $m \in M^k$, the discrete map $f_m^k : \mathcal{X}^k \times \Sigma^{k-1} \to \mathcal{X}^k$ defines the *internal transitions* on $(l, \mu) \in \mathbb{Z}^p \times M^{k-1}$ pairs that make up motion primitive m. These discrete maps must adhere to the constraints on successive level $k-1$ motion primitives as specified in $\mathcal{H}^{k-1}$ (formal details are below). The invariant $I^k(m)$ is the set of (l, μ) pairs that the discrete map f_m^k acts on. The index l in the pair (l, μ) identifies the origin of the frame for $\mu \in M^{k-1}$ w.r.t. the frame for $m \in M^k$.

The enabling conditions and reset maps take the analogous meanings as in $\mathcal{H}^0$. They define the *external transitions* from primitive $m \in M^k$ to other level k motion primitives (including m itself). To properly define reset maps, it is necessary to keep track of reference frames for the various primitives involved. Suppose we have three frames for the three primitives $m, m' \in M^k$, and $\mu \in M^{k-1}$, respectively. When an internal or external transition occurs, the next primitive effectively starts from a shifted frame. This shift is recorded in terms of the base point of each frame, denoted o_m, $o_{m'}$, and o_μ, respectively. For example, the origin of the frame for μ w.r.t. the frame for m is denoted by o_μ^m. Similarly, the origin of the frame for m' w.r.t. the frame for m is denoted by $o_{m'}^m$.

Definition 3.2: Suppose we are given a level $k-1$ MA $\mathcal{H}^{k-1}$, where $k \geq 1$. A *MA* at level k is a tuple $\mathcal{H}^k =$

$(Q^k, \Sigma^k, E^k, I^k, Q^{k,0}, X^k, G^k, R^k)$, where

State Space $Q^k = M^k \times \mathcal{X}^k$ is the hybrid state space, where $M^k = \{m_1^k, m_2^k, \ldots\}$ is a finite set of motion primitives at level k, and $\mathcal{X}^k = \mathbb{Z}^p \times M^{k-1}$ is the analogue of the notion of a continuous state space at level 0.

Labels $\Sigma^k = \{s_1^k, s_2^k, \ldots, \mid s_i^k \in \mathbb{Z}^p \times \{-1,0,1\}^p\}$ is a finite set of event labels. If $s = (l_s, \kappa_s) \in \Sigma^k$, then $l_s \in \mathbb{Z}^p$ is the box index w.r.t. a frame for some primitive $m \in M^k$ from which an external transition occurs, and κ_s identifies the face of the box through which the transition occurs.

Edges $E^k \subset M^k \times \Sigma^k \times M^k$ is a finite set of edges describing which level k motion primitives can be concatenated.

Invariants $I^k : M^k \to \mathcal{P}(\mathcal{X}^k)$ assigns to each motion primitive $m \in M^k$ a non-empty, finite set of states $I^k(m) \subset \mathcal{X}^k$ on which the transition function $f_m^k : \mathcal{X}^k \times \Sigma^{k-1} \to \mathcal{X}^k$ (defined below) acts. The box indices $l \in \mathbb{Z}^p$ for pairs $(l, \mu) \in I^k(m)$ correspond to the origin of the frame for $\mu \in M^{k-1}$ w.r.t. the frame for $m \in M^k$. The invariant is different from the *envelope* of $m \in M^k$, denoted $L^k(m)$, which is the collection of all box indices w.r.t. the frame for m accumulated from the envelopes of the lower level primitives that constitute m.

Initial Conditions $Q^{k,0} \subset Q^k$ assigns a set of initial states, satisfying: if $(m, x) \in Q^{k,0}$, then $x \in I^k(m)$.

Transition Functions $X^k : M^k \to \{f_m^k\}_{m \in M^k}$ assigns to each $m \in M^k$ a transition function $f_m^k : \mathcal{X}^k \times \Sigma^{k-1} \to \mathcal{X}^k$ that specifies all the allowable *internal transitions* between $(l, \mu) \in I^k(m)$ pairs that together constitute the level k motion primitive. Consider $(l, \mu) \in I^k(m)$ and $\sigma = (l_\sigma, \kappa_\sigma) \in \Sigma^{k-1}$, such that $l + l_\sigma + \kappa_\sigma \in L^k(m)$ (identifying that this is an internal transition) and such that $(\mu, \sigma, \mu') \in E^{k-1}$ for some $\mu' \in M^{k-1}$. If $f_m^k((l, \mu), \sigma) = (l', \mu')$, then (i) $(l', \mu') \in I^k(m)$; (ii) $(\mu, \sigma, \mu') \in E^{k-1}$; and (iii) $l = o_\mu^m$, $l' = o_{\mu'}^m$, and $l' - l = o_{\mu'}^\mu$.

Enabling Conditions $G^k : E^k \to \{g_e^k\}_{e \in E^k}$ assigns to each edge $e = (m, s, m') \in E^k$ a non-empty enabling condition $g_e^k \subset I^k(m)$. If $s = (l_s, \kappa_s) \in \Sigma^k$, the first requirement is that $l_s + \kappa_s \notin L^k(m)$, identifying that this is an external transition. Then g_e^k consists of all those pairs $(l, \mu) \in I^k(m)$ for which an *external transition* to a consecutive level k motion primitive can occur. That is, $(l, \mu) \in g_e^k$ if there exists (a unique) $\sigma = (l_\sigma, \kappa_\sigma) \in \Sigma^{k-1}$ such that (i) there exists $\tilde{\mu} \in M^{k-1}$ such that $(\mu, \sigma, \tilde{\mu}) \in E^{k-1}$; (ii) $l = o_\mu^m$, $l_s = l_s^m$ is the index w.r.t. the frame for m for the box from which the external transition occurs, and $l_\sigma = l_\sigma^\mu$ is the index w.r.t. the frame for μ for the same box. Therefore $l_s = l_s^m = o_\mu^m + l_s^\mu = l + l_\sigma$ and $\kappa_\sigma = \kappa_s$.

Reset Conditions $R^k : E^k \to \{r_e^k\}_{e \in E^k}$ assigns to each edge $e = (m, s, m') \in E^k$ a reset map $r_e^k : \mathcal{X}^k \times \Sigma^{k-1} \to \mathcal{X}^k$ that characterizes the *external transitions* of $m \in M^k$. Associated with e is a unique transformation $o_{m'}^m \in \mathbb{Z}^p$ relating the frames of m and m' for this external transition. Consider $s = (l_s, \kappa_s) \in \Sigma^k$, $\sigma = (l_\sigma, \kappa_\sigma) \in \Sigma^{k-1}$, and $(l, \mu) \in g_e^k$ such that $l_\sigma = l_\sigma^\mu = l_\sigma^m - o_\mu^m = l_s - l$ and $\kappa_\sigma = \kappa_s$. If $r_e^k((l, \mu), \sigma) = (l', \mu')$, then (i) $(l', \mu') \in$
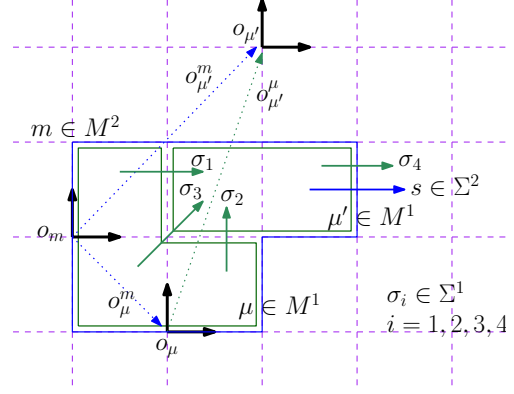

Fig. 3. An example of level 1 and level 2 motion primitive reference frames, inspired from the $p = 2$ output example from Figure 2. The primitive $m \in M^2$ consists of two primitives $\mu, \mu' \in M^1$.

$I^k(m')$; (ii) $(\mu, \sigma, \mu') \in E^{k-1}$; (iii) $l = o_\mu^m$, $l' = o_{\mu'}^{m'}$, and $(l' + o_{m'}^m) - l = o_{\mu'}^\mu$. $\triangleleft$

Example 3.1: Returning to the two level hierarchy of Figure 2, suppose we want to design motion primitives at level 2 based on the existing motion primitives at level 1. Consider the candidate motion primitive $m \in M^2$ formed by the concatenation of the two level 1 motion primitives $\mu = Right\&Up$ and $\mu' = TwoRight$. Figure 3 shows all the geometric information to understand how μ and μ' are laid out relative to each other to construct m.

The invariant for m is $I^2(m) = \{(l, \mu), (l', \mu')\}$, where $l = (-1, 1)$ and $l' = (2, 2)$ are the indices w.r.t. to the frame for m for the boxes to which the frames for μ and μ', respectively, are attached. Three internal transitions are encoded in the transition function f_m^2, corresponding to the level 1 edges $e_i = (\mu, \sigma_i, \mu') \in E^1$, $i = 1, 2, 3$. For example, the level 2 internal transition $f_m^2((l, \mu), \sigma_1) = (l', \mu')$ is implemented by a level 1 edge $e_1 \in E^1$ with associated event $\sigma_1 = (l_{\sigma_1}, \kappa_{\sigma_1}) \in \Sigma^1$. In the frame of o_μ, the transition occurs from the box $l_{\sigma_1} = (-1, 1)$ through its right face $\kappa_{\sigma_1} = (1, 0)$.

There is one level 2 external transition for $m \in M^2$ with an associated edge $e = (m, s, m') \in E^2$. In turn, it corresponds to a level 1 edge $e_4 = (\mu', \sigma_4, \mu'') \in E^1$ and event $\sigma_4 \in \Sigma^1$. Thus, $r_e^2((l', \mu'), \sigma_4) = (l'', \mu'')$ for some $(l'', \mu'') \in I^2(m')$. The event $\sigma_4 = (l_{\sigma_4}, \kappa_{\sigma_4})$ is expressed in the frame of $o_{\mu'}$, while the event $s = (l_s, \kappa_s) \in \Sigma^2$ is expressed in the frame of o_m. Thus, $l_{\sigma_4} = (0, -2)$, $\kappa_{\sigma_4} = (1, 0)$, $l_s = (2, 0)$, and $\kappa_s = (1, 0)$. $\triangleleft$

For $k \geq 1$ we define the notation $\mathcal{H}^{k-1} \preceq \mathcal{H}^k$ to mean that $\mathcal{H}^k$ is an abstraction built up from $\mathcal{H}^{k-1}$ according to Definition 3.2. In analogy with $\mathcal{H}^0$, we define an *execution at level* $k > 0$, denoted as $\chi^k = (\tau^k, m^k, x^k)$. There are two differences from χ^0. First, the hybrid time domain τ^k is a sequence of sets of discrete times (e.g. $\tau^k = \{\{0, 1, 2\}, \{3\}, \{4, 5\}, \ldots\}$), where each set consists of the discrete times when internal transitions occur. Second, continuous and discrete transitions at level 0 are replaced by internal and external transitions, respectively, at level k.

C. Hierarchical Maneuver Automaton

Now consider a collection of maneuver automata, $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$, where $K \geq 0$, such that for all $k = 1, \ldots, K$, $\mathcal{H}^{k-1} \preceq \mathcal{H}^k$. We call $\mathcal{H}$ a *hierarchical maneuver automaton* (HMA).

Next, we can also define an overall *hierarchical execution* by stacking together each execution at level k. Suppose that a level 0 event occurs from the current level 0 motion primitive. This event is either interpreted as an internal or external transition at level 1. If it is internal, the transition function X^1 determines the next motion primitive at level 0 (and the continuous state is reset with R^0). Otherwise, this event is propagated up the hierarchy until it is registered as internal at some level $1 \leq K' \leq K$ or external at level K. In the former case, the transition function $X^{K'}$ determines the next motion primitive at level $K' - 1$, while in the latter case the reset map R^K determines the next motion primitive at level K. An internal transition at level K' is always implemented as an external transition at the levels $K' - 1$ to 0. The reset maps update the motion primitive at each level below. For example, if there is an internal transition at level 3, then at level 2, R^2 determines the next level 1 primitive, R^1 determines the next level 0 primitive, and R^0 resets the continuous state.

Recalling our definition of the behavior induced by an output trajectory $y(\cdot, x_0)$, we can likewise define the behavior induced by a hierarchical execution.

Definition 3.3: Let $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$ be an HMA and let $\chi = \{\chi^k\}_{k=0}^K$ be a hierarchical execution of $\mathcal{H}$. The *behavior* of χ is defined to be the behavior of its associated χ^0; namely it is a sequence $\kappa_1 \kappa_2 \ldots$ where $\sigma_i = (0, \kappa_i)$ is the i-th event at level 0. The *language* of $\mathcal{H}$ denoted by $\mathcal{L}(\mathcal{H})$ is the set of all behaviors induced by the hierarchical executions of $\mathcal{H}$.
◁

IV. SOLUTION OF PROBLEM 2.1

In this section we give the main elements of the solution to Problem 2.1 using hierarchical motion primitives.

Definition 4.1: Let $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$ be a HMA. We say that $\mathcal{H}$ is *well-posed* if (i) each hierarchical execution χ induces a unique behavior in $\mathcal{B}$, and (ii) for all initial conditions of a hierarchical execution, there exists a maximal extension to a hierarchical execution such that its level 0 execution is infinite.
◁

Ideally, we would provide a collection of checkable conditions in terms of HMA parameters, and then *prove* that we obtain a well-posed HMA. See [23] where such proofs are supplied for $\mathcal{H}^0$ only. We have adopted Definition 4.1 in order to focus this paper on high level ideas.

Definition 4.2: Let $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$ be a HMA with $K \geq 1$. We say that $\mathcal{H}$ is *complete* if the following hold:

$$M^K = \{m^\star\}, \ E^K = \emptyset, \ Q^{K,0} = \{(m, x) \in Q^K \mid x \in I^k(m)\}.$$

That is, at level K there is only one motion primitive, there are no external transitions, and all states are valid initial conditions.
◁

Completeness provides a hierarchy $\mathcal{H}$ whose top level acts as a *control policy* dictating the assignment of motion primitives one level below, with those motion primitives dictating the assignment of motion primitives below that, and so on. In practice, the control policy yielding a complete $\mathcal{H}$ is obtained by running a planning algorithm at level $K - 1$ using only the discrete part $(M^{K-1}, \Sigma^{K-1}, E^{K-1})$, which is a (non-deterministic) graph with labeled transitions. For the reach-avoid problem, one may adapt our algorithm from [23] for planning at level $K - 1$, or other graph search methods may be employed.

There are two main advantages associated with planning at higher levels. First, a policy produced at a higher level would be much more efficiently computed and represented, composed of only a few high level motion primitives and the transitions between them. Second, the application of planning algorithms to generate policies at higher levels would not need modifications to enforce behavioral constraints explicitly, provided that the concatenation of the constituent high level motion primitives automatically preserve the desired behavioral constraint.

Our main result, Theorem 4.1, states conditions to solve Problem 2.1. We can see that conditions (i), (ii), and (iii) correspond to the avoid, reach, and behavior specifications, respectively. The significance of the result is that if a planning algorithm can meet these conditions, then they satisfy the requirements of Problem 2.1 at the lowest, continuous time level.

Theorem 4.1: Consider the system (1) with a gridded output space in terms of grid length vector $d \in \mathbb{R}^p$. We are given a feasible and goal set of boxes, $L_g \subset L_f \subset \mathbb{Z}^p$ and a behavior constraint $\widehat{\mathcal{B}} \subset \mathcal{B}$. Consider a HMA $\{\mathcal{H}^k\}_{k=0}^K$, which is well-posed and complete. Suppose that

(i) $L^K(m^\star) \subset L_f$,
(ii) all hierarchical executions of $\mathcal{H}$ reach some subset of boxes of L_g and never leave,
(iii) $\mathcal{L}(\mathcal{H}) \subset \widehat{\mathcal{B}}$.

Then there exists a set of initial conditions $\mathcal{X}_0$ and an associated feedback control strategy $u(x)$ solving Problem 2.1.

V. FORMATION CONTROL

In this section we show how hierarchical motion primitives can be used to solve a formation control problem. To illustrate ideas, we consider two agents, each modeled as a double integrator with a scalar output. The behavioral constraint to maintain a formation of two agents, each with one output, is that if one agent incurs an increment in its output value due to the application of a motion primitive, then the other agent experiences the same increment in the successive motion primitive. Consider a behavior $y_b = \kappa^1 \kappa^2 \ldots \in \mathcal{B}$ of the multi-agent system, where $\kappa^i = (\kappa_1^i, \kappa_2^i)$. We require that for all $i \geq 1$,

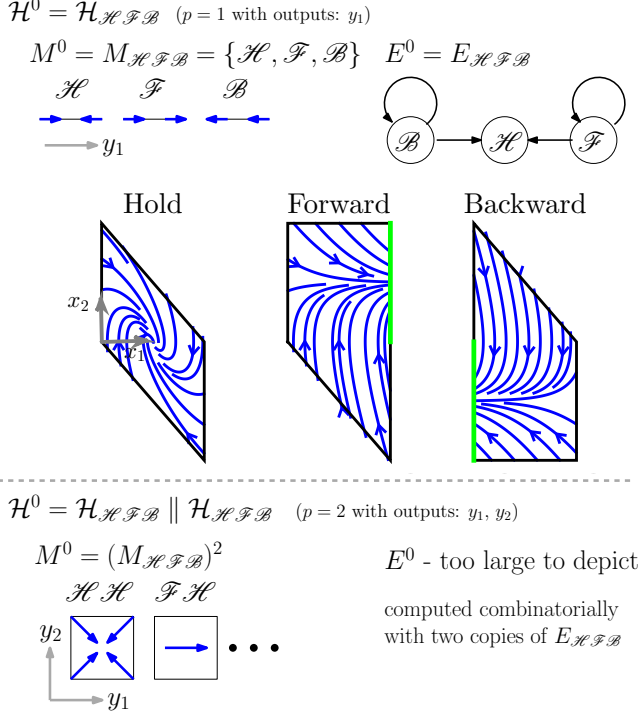$$\left| \sum_{l=1}^i (\kappa_1^l - \kappa_2^l) \right| \leq 1 . \tag{2}$$

Fig. 4. This figure illustrates a level 0 MA design $\mathcal{H}^0$ for a system of p double integrators. The top part shows the atomic motion primitives M^0 and their edges E^0 for one double integrator ($p = 1$), which are *Hold* ($\mathcal{H}$), *Forward* ($\mathcal{F}$), and *Backward* ($\mathcal{B}$). Also shown are the associated invariants, vector fields, and enabling conditions (green) in the position-velocity (x_1, x_2) state space. The bottom part shows that two copies of the $p = 1$ design may be composed to yield motion primitives for $p = 2$. This procedure extends easily to an arbitrary number of p outputs.
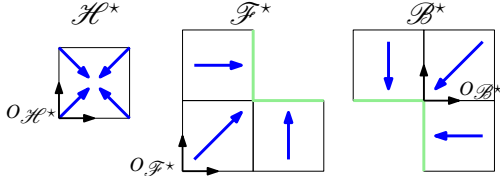

Fig. 5. This figure illustrates a level 1 MA design $\mathcal{H}^1$ to achieve the formation behavior constraint. It is built as an abstraction of the level 0 MA $\mathcal{H}^0$ shown in Figure 4. Although this figure illustrates the case of two outputs $p = 2$, this design is easily scalable with respect to any $\mathcal{H}^0$ (of the form shown in Figure 4), which may have any number of p outputs.

Under this behavioral constraint, the relative value of the two outputs is maintained while at the same time a sequence of motion primitives to achieve a reach-avoid specification is executed by the multi-agent system. The behavioral constraint can be easily generalized to N agents.

To implement the formation behavioral constraint using hierarchical motion primitives, we propose a two level hierarchy consisting of the MA's, $\mathcal{H}^0$ and $\mathcal{H}^1$. Level 0 is based on our previous design presented in [22], [23], which we now briefly summarize.

First, the level 0 (atomic) motion primitives for each agent are: *Hold* ($\mathcal{H}$), *Forward* ($\mathcal{F}$), and *Backward* ($\mathcal{B}$), corresponding to the agent's output holding its value (remaining in the current box of the gridded 1D output space), increasing its value (moving to a box to the right), or decreasing its value (moving to a box to the left). Each agent i is equipped with a level 0 maneuver automaton denoted $\mathcal{H}^{0,i} = \mathcal{H}_{\mathcal{H}\mathcal{F}\mathcal{B}}$, $i = 1, \ldots, N$, as shown in Figure 4.

Second, to obtain a level 0 maneuver automaton for the two agent system, we take the parallel composition of the individual MA's to form the MA $\mathcal{H}^0 = \mathcal{H}^{0,1} \| \mathcal{H}^{0,2}$. Informally, parallel composition is based on the cartesian product to account for the asynchronous possibilities that arise when combining two independent subsystems. More generally, for p agents we would form $\mathcal{H}^0 = \|_{j=1}^{p} \mathcal{H}^{0,j}$. Figure 4 illustrates some of the level 0 motion primitives for the two agents, including $\mathcal{H}\mathcal{H}$, $\mathcal{F}\mathcal{H}$, and so forth for all the possible neighboring directions in 2D.

Finally, the level 1 motion primitives for the two agent system are: *Formation Forward* ($\mathcal{F}^\star$), *Formation Backward* ($\mathcal{B}^\star$), and *Formation Hold* ($\mathcal{H}^\star$). These are depicted in Figure 5. It is easy to see that the concatenation of these level 1 motion primitives satisfies the desired behavior (2). Once again, this design is easily generalized to N agents.

VI. EXPERIMENTAL RESULTS

It is well known that the model of a quadrotor is differentially flat, and so the nonlinear dynamics can be effectively treated as three independent double integrators for each of the world frame coordinates (x_w, y_w, z_w) [18]. Suppose we have a collection of N quadrotors. The corresponding multi-vehicle system (1) has $p = 3N$ outputs with $n = 6N$ states. We use our formation hierarchical motion primitives to solve Problem 2.1 with a formation behavior constraint.

First, we grid the 3D physical space and identify obstacles. A formation configuration is specified in terms of the relative box offsets between the vehicles to a chosen representative vehicle and a goal box is specified for the representative. Although the full gridded output space for the N agent system is obtained by composing the gridded 3D physical space N times, we do not compute it explicitly because N may be large. Moreover, instead of specifying the feasible boxes $L_f \subset \mathbb{Z}^p$, we define virtual obstacles in 3D based the representative vehicle locations, the formation configuration, and physical obstacles. The behavioral constraint $\widehat{\mathcal{B}}$ on the full gridded space consists of the constraint (2) applied to each physical directions (x_w, y_w, z_w) independently on the N vehicles.

Next, we group together all outputs of the N agents corresponding to each workspace direction x_w, y_w, and z_w, and to each such group we appropriate one level 1 MA, $\mathcal{H}_{\mathcal{H}^\star \mathcal{F}^\star \mathcal{B}^\star}$. Then we take the parallel composition of these three MA's for the three workspace directions to obtain $\mathcal{H}^1 = \|_{j=1}^{3} \mathcal{H}_{\mathcal{H}^\star \mathcal{F}^\star \mathcal{B}^\star}$, which has $3^3 = 27$ motion primitives, independent of N. The resulting HMA $\mathcal{H} = \{\mathcal{H}^0, \mathcal{H}^1\}$ is guaranteed to generate the correct behavior, $\mathcal{L}(\mathcal{H}) \subset \widehat{\mathcal{B}}$, because the individual components $\mathcal{H}_{\mathcal{H}^\star \mathcal{F}^\star \mathcal{B}^\star}$ satisfy (2).

Finally, we generate a control policy $\mathcal{H}^2$ to yield a complete HMA $\mathcal{H}' = \{\mathcal{H}^0, \mathcal{H}^1, \mathcal{H}^2\}$, which also preserves the forma-
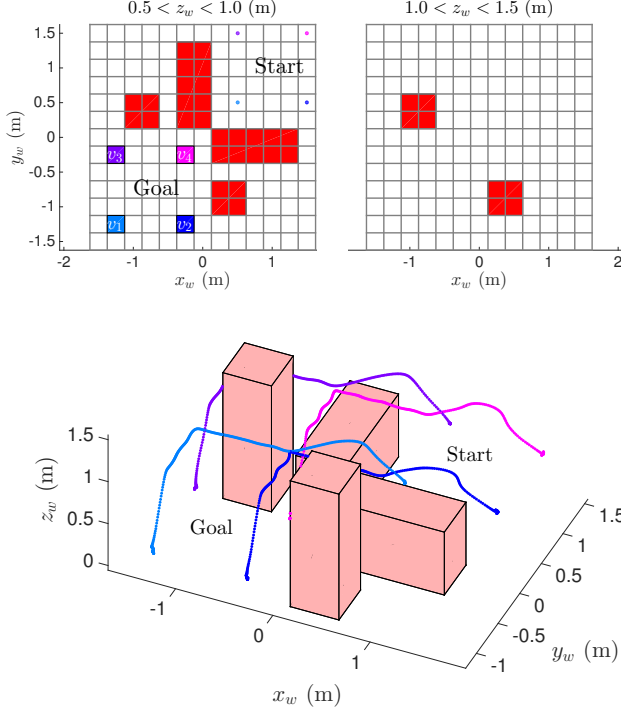
Fig. 6. Experimental results for the square formation scenario shown in Figure 1.

tion. The control policy can be generated efficiently based on the 27 motion primitives of $\mathcal{H}^1$ and the box locations of the representative vehicle using any shortest path planning algorithm. More specifically, the computation of the control policy is equivalent to the complexity of planning as with a single agent, and the computation of the virtual obstacles is linear in the number of agents. Thus we satisfy our aim to dramatically improve the overall computation time.

Experimental results are demonstrated for four Crazyflie quadrotors flown indoors using a VICON motion capture system. The top row of Figure 6 shows the gridded 3D workspace, the physical obstacles (red), and the goal boxes (colored). The behavioral constraint is imposed to maintain a square formation. This scenario is challenging because of the intermingling of obstacles amidst the moving formation and because 3D maneuvers are required for obstacle avoidance. The middle row of Figure 6 shows the resulting 3D trajectories in the workspace, demonstrating that the reach-avoid task was successfully executed. A video is available at http://tiny.cc/hier-moprim. It also shows different scenarios involving more vehicles in order to demonstrate the applicability to a variety of formations and settings as well as the computational scalability of our approach.

VII. Conclusion

In conclusion, we have introduced hierarchical motion primitives to solve the motion planning problem for a large collection of agents. Hierarchical motion primitives are constructed recursively using a hierarchical maneuver automaton. The framework was applied to multi-vehicle formations,

yielding novel behaviors such as the ability to pass "through" obstacles while maintaining a formation. Future work includes developing an automated procedure to generate efficient structures of hierarchical motion primitives for arbitrary reach-avoid behavior-constrained problems and discovering additional applications of our hierarchical approach.

References

[1] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas, Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, vol. 88, no 7, pp. 971-984, 2000.

[2] N. Ayanian and V. Kumar, Abstractions and controllers for groups of robots in environments with obstacles. *Proc. of the IEEE International Conference on Robotics and Automation*, 2010.

[3] P. E. Caines and Yuan-Jun Wei, Hierarchical hybrid control systems: a lattice theoretic formulation. *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 501-508, 1998.

[4] M. Denia and A. Ude, Synthesis of new dynamic movement primitives through search in a hierarchical database of example movements. *International Journal of Advanced Robotic Systems*, vol. 12, no. 10, pp. 137-150, 2015.

[5] M. Egerstedt and X. Hu, Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 947-951, 2001.

[6] E. Frazzoli, M. A. Dahleh, and E. Feron, Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077-1091, 2005.

[7] D. J. Grymin, C. B. Neas, and M. Farhood, A hierarchical approach for primitive-based motion planning and control of autonomous vehicles. *Robotics and Autonomous Systems*, vol. 62, no. 2, pp. 214-228, 2014.

[8] P. Hubbard and P. E. Caines, Dynamical consistency in hierarchical supervisory control. *IEEE Transactions on Automatic Control*, vol. 47, no. 1, pp. 37-52, 2002.

[9] L. P. Kaelbling and T. Lozano-Prez, Hierarchical task and motion planning in the now. *Proc. of the IEEE International Conference on Robotics and Automation*, 2011.

[10] M. Kloetzer and C. Belta, Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 320-330, 2007.

[11] M. Kloetzer and C. Belta, A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*. Vol. 53, no. 1, pp. 287-297, 2008.

[12] D. Kulic, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, Incremental learning of full body motion primitives and their sequencing through human motion observation. *International Journal of Robotics Research*, vol. 31, no. 3, pp. 330-345, 2012.

[13] J. Lygeros, K.H. Johansson, S. Sastry, and M. Egerstedt. On the existence of executions of hybrid automata. *IEEE Conference on Decision and Control*, 1999.

[14] S. Nagavalli, N. Chakraborty, K. Sycara, Automated sequencing of swarm behaviors for supervisory control of robotic swarms. *Proc. of the IEEE International Conference on Robotics and Automation*, 2017.

[15] K-K. Oh, M-C. Park, and H-S. Ahn, A survey of multi-agent formation control. *Automatica*, vol 53, no. 3, pp. 424-440, 2015.

[16] M. Pivtoraiko and A. Kelly, Kinodynamic motion planning with state lattice motion primitives. *Proc. of the International Conference on Intelligent Robots and Systems*, 2011.

[17] S. J. Russell and P. Norvig, Artificial Intelligence, a Modern Approach. Prentice Hall, third edition, 2010.

[18] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, Automated composition of motion primitives for multi-robot systems from safe LTL specifications. *International Conference on Intelligent Robots and Systems*, 2014.

[19] L. Sentis and O. Khatib, Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505-518, 2005.

[20] X. Sun and C. G. Cassandras, Optimal dynamic formation control of multi-agent systems in constrained environments. *Automatica*, vol. 73, no. 11, pp. 169-179, 2016.

[21] P. Tabuada, G. J. Pappas, and P. Lima, Compositional abstractions of hybrid control systems. *Proc. of the IEEE Conference on Decision and Control*, 2001.

[22] M. Vukosavljev, Z. Kroeze, M. E. Broucke, and A. P. Schoellig. A Framework for Multi-Vehicle Navigation Using Feedback-Based Motion Primitives. *Proc. of the International Conference on Intelligent Robots and Systems*, 2017.

[23] M. Vukosavljev, Z. Kroeze, A. P. Schoellig, and M. E. Broucke. A modular framework for motion planning using safe-by-design motion primitives. *IEEE Transactions on Robotics*, accepted January 2019.

[24] E. M. Wolff, U. Topcu, and R. Murray. Optimal control of non-deterministic systems for a computationally efficient fragment of temporal logic. *Proc. of the IEEE Conference on Decision and Control*, 2013.

[25] W.M. Wonham and K. Cai, "Supervisory Control of Discrete-Event Systems," [Online]. Available: http://www.control.utoronto.ca/~wonham/Research.html.