A Framework for Multi-Vehicle Navigation using Feedback-Based Motion Primitives

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

Abstract— We present a hybrid control framework for solving a motion planning problem among a collection of heterogenous agents. The proposed approach utilizes a finite set of low-level *motion primitives*, each based on a *piecewise affine feedback control*, to generate complex motions in a gridded workspace. The constraints on allowable sequences of successive motion primitives are formalized through a *maneuver automaton*. At the higher level, a control policy generated by a *shortest path non-deterministic algorithm* determines which motion primitive is executed in each box of the gridded workspace. The overall framework yields a highly robust control design on both the low and high levels. We experimentally demonstrate the efficacy and robustness of this framework for multiple quadrocopters maneuvering in a 2D or 3D workspace.

I. INTRODUCTION

This paper presents a modular, hierarchical framework for motion planning and control of robotic systems. Motion planning has received a great deal of attention by many researchers, and many hierarchical frameworks have already been presented. We begin by reviewing the important features of existing frameworks and then we highlight the unique properties of our approach.

First, a common control specification is called *reach-avoid* [2]–[4], in which robotic agents must navigate an environment cluttered with obstacles to reach a final goal configuration. More complex tasks expressed using *temporal logic* have also been considered [5], [6]. Second, the physical workspace is typically *abstracted* into a finite number of regions [2], [7], [8] or points [9]. Planning at the high-level is often based on applying a *shortest path algorithm* to the graph arising from the workspace abstraction. At the low-level, the control system is often assumed to be feedback linearizable or differentially flat [10], [11] so that standard *trajectory tracking* [4], [8], [12] or *feedback control* methods [3], [5]–[7], [13]–[15] can be applied. Recently the idea of *motion primitives* has further simplified the complexity of low-level control design [4], [8], [15]. Finally, feasibility of successive motion primitives can be formalized through a so-called *maneuver automaton* [12].

In this paper we present a modular, hierarchical framework for motion control which incorporates the following features: (i) we consider a reach-avoid control specification; (ii) we

Marijan Vukosavljev, Zachary Kroeze and Mireille E. Broucke are with the Dept. of Electrical and Computer Engineering, University of Toronto, Canada (e-mails: mario.vukosavljev@mail.utoronto.ca, zach.kroeze@mail.utoronto.ca, broucke@control.utoronto.ca). Angela P. Schoellig is with the University of Toronto Institute for Aerospace Studies (UTIAS), Canada (email: schoellig@utias.utoronto.ca). Supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Associated video at: http://tiny.cc/quad5scenes.

Fig. 1. Two quadrocopters navigating around an obstacle in the presence of wind disturbances from a fan. A video illustrating the results can be found at http://tiny.cc/quad5scenes.

abstract the workspace by partitioning it into rectangular regions; (iii) we assume the nonlinear control system has a translational symmetry in the output; (iv) we perform high-level motion planning using standard shortest path algorithms; (v) we employ a maneuver automaton to encode feasible sequences of motion primitives; (vi) we perform low-level control design of motion primitives based on reach control theory [13], [16]. We have selected these features because they provide an appealing balance between applicability and efficiency.

Despite the wealth of available literature, our framework offers important advantages over existing approaches, particularly due to the following three features:

Robustness and safety. Our framework uses low-level controllers based on *reach control theory*. It yields piecewise affine feedbacks that guarantee safety constraints on the states and outputs of the closed-loop system [13], [16] without the need to specify feasible open-loop trajectories. While other closed-loop methods such as potential methods [7] can be used in our framework, reach control is used for its guarantees on safety and for its ease of design.

Asynchronous motion of multiple agents. In a multi-agent system, each agent may execute a motion primitive independently of the actions of other agents. This freedom of action is modelled in our framework in terms of the *non-determinism* of a graph which captures feasible sequences of motion primitives of the collection of agents. Although each robot's trajectory is deterministic for a given initial condition, the graph abstracts low-level state information regarding specific initial conditions so that the order of completion of motion primitives arises as non-determinism. This feature allows for time-independent asynchronous motion,

which contrasts with many existing approaches where robot motion is typically synchronized or restricted to move one at a time [3], [5], [14].

Computational feasibility. One of our main modelling assumptions is that the system dynamics have a *translational symmetry* in the outputs [12]. This assumption implies that motion primitives can be first designed over a single box in the output space, and then be reapplied to other boxes since they are translations of each other. This feature significantly reduces the complexity of the high-level planning problem, allowing for a computationally feasible solution.

Finally, we highlight the differences between this paper and our previous work [1]. In [1], reach control was demonstrated on a single quadrocopter exhibiting a one-dimensional side-to-side motion. The difficult steps of performing a state space triangulation of the safety region and specifying a high-level sequence on this triangulation were done manually. In this paper, we have addressed these difficulties by introducing features (i)-(vi). This enabled us to extend the reach control design in [1] to higher dimensional systems, which in this paper is demonstrated on multiple quadrocopters moving in several degrees of freedom simultaneously.

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 control input, and $y \in \mathbb{R}^p$ is the output, with $p \leq n$. This model can represent the aggregate dynamics of a multi-robot system with the states as the positions and velocities, and the outputs as the positions, possibly after feedback linearization or differential flatness transformations. We denote $x(t)$ as the state trajectory and $y(t)$ as the output trajectory of (1) under some initial condition and control.

This paper considers a reach-avoid problem to find a feedback control causing the output trajectories of (1) to reach a goal set while avoiding an obstacle set. Let $\mathcal{P} \subset \mathbb{R}^p$ denote a feasible output space. Let $\mathcal{O} \subset \mathcal{P}$ denote a subset of obstacle regions, and let $\mathcal{G} \subset \mathcal{P}$ be a nonempty goal set. See Figure 3 for an example. We consider the following problem.

Problem 2.1: Given the system (1), output space $\mathcal{P} \subset \mathbb{R}^p$, obstacle set $\mathcal{O} \subset \mathcal{P}$, and goal set $\mathcal{G} \subset \mathcal{P}$, find a feedback control strategy $u(x)$ and a set of initial conditions $\mathcal{X}_0 \subset \mathbb{R}^n$ such that for each $x(0) \in \mathcal{X}_0$ we have

(i) **Avoid**: for all $t \geq 0$, $y(t) \in \mathcal{P} \setminus \mathcal{O}$.
(ii) **Reach**: there exists $T \geq 0$ such that for all $t \geq T$, $y(t) \in \mathcal{G}$.

We make a standing assumption regarding the outputs of the system (1) in order to exploit symmetry; see [12] for an exposition on nonlinear control systems with symmetries. This assumption is satisfied for many robotic systems, for example, when the outputs are positions. Also, it significantly simplifies our control design; see Section III-C.

Assumption 2.1: For simplicity, the outputs are the first p states, that is, $y_i = x_i$ for $i = 1, \ldots, p$. Moreover, the system
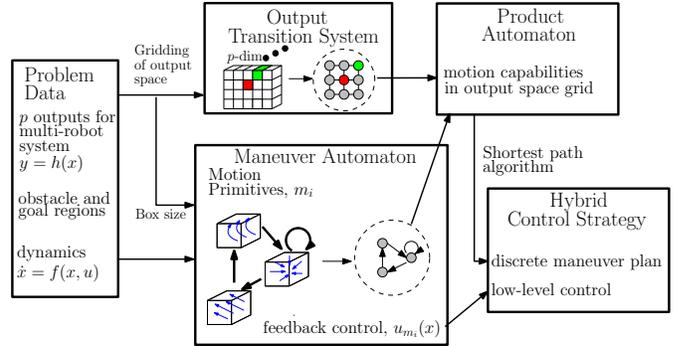

Fig. 2. Our hybrid control framework consists of five modules.

has a *translational invariance* with respect to its outputs, that is, for all $u \in \mathbb{R}^m$ and $x, x' \in \mathbb{R}^n$ with $x'_j = x_j$, $j = p+1, \ldots, n$, we have $f(x, u) = f(x', u)$. ◁

III. METHODOLOGY

In this section we present our methodology. It can be broken down into five main modules, as depicted in Figure 2.

- The *problem data* include the system (1) with p outputs satisfying Assumption 2.1 and a reach-avoid task to be executed in the output space.
- The *output transition system* (OTS) is a graph that captures an abstraction of the output space based on a partition of the output space into boxes. The reach-avoid task is translated in terms of desired behaviors on the OTS.
- The *maneuver automaton* (MA) is a hybrid system whose nodes correspond to all possible control modes, called *motion primitives*, that can be implemented over any single box in the output space. The edges of the MA represent feasible successive motion primitives.
- The *product automaton* (PA) is a graph which is the synchronous product of the OTS and the discrete part of the MA. It represents the combined constraints on feasible motions in the output space and feasible successive motion primitives.
- The *hybrid control strategy* is a combination of the high-level plan and the low-level controllers obtained from the design of motion primitives. The high-level plan is obtained by applying a shortest path algorithm adapted to non-deterministic graphs on the PA [17].

Now we describe in detail each of these modules.

A. Output Transition System

The output transition system (OTS) provides an abstract description of the *workspace* or *output space* associated with the system (1). Consider a canonical p-dimensional box $Y^* := \prod_{i=1}^p [0, d_i]$ with edge lengths $d_i > 0$. Let $\{Y_j\}_{j=1}^{n_L}$ be a collection of boxes that covers the output space $\mathcal{P}$. We assume that each box Y_j is a translation of Y^* by an integer multiple of d_i in the i-th output coordinate. This yields a gridded output space as in Figure 3. The OTS is a graph that abstracts this gridded space by associating a label l_j to each
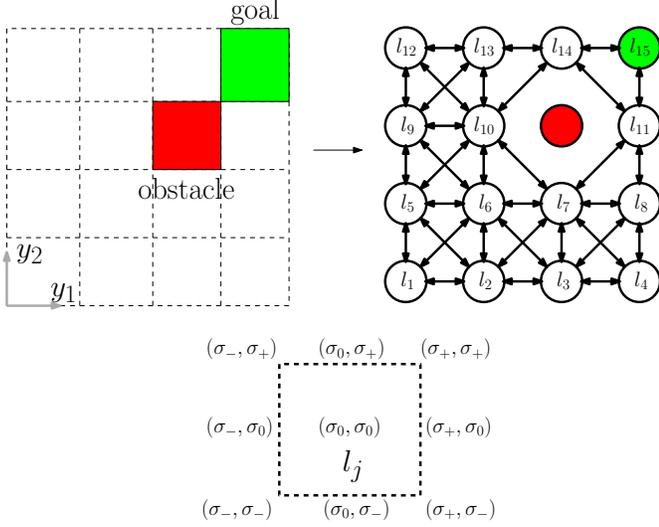
Fig. 3. A two-output, $p = 2$, example of a reach-avoid task. Shown on the left is the feasible space $\mathcal{P}$ consisting of 16 boxes, the obstacle region $\mathcal{O}$ (red), and the goal region $\mathcal{G}$ (green). The output transition system (OTS), which abstracts the 15 non-obstacle box regions and their neighbor connectivity, is shown on the right. Shown below, the faces of a box are labelled using $\Sigma = \prod_{i=1}^{2} \{\sigma_0, \sigma_+, \sigma_-\}$, which includes the box edges, vertices, and interior.

box Y_j. Edges in the OTS encode contiguous boxes in the output space, where two boxes are said to be contiguous if they share any face of dimension between 1 and $p-1$. Notice that the OTS does not describe the actual evolution of output trajectories of (1).

The *output transition system* (OTS) is a tuple $\mathcal{A}_{\text{OTS}} = (L_{\text{OTS}}, \Sigma, E_{\text{OTS}})$ with the following components:

Locations L_{OTS} denotes a finite set of locations. Each location $l_j \in L_{\text{OTS}}$ is associated with a non-obstacle box Y_j in the output space.

Labels $\Sigma := \prod_{i=1}^{p} \{\sigma_0, \sigma_+, \sigma_-\}$ is a finite set of labels. Each label $\sigma \in \Sigma$ uniquely identifies the face of a box using the components σ_0, σ_+, σ_-. Refer to Figure 3 for a $p = 2$ example.

Edges $E_{\text{OTS}} \subset L_{\text{OTS}} \times \Sigma \times L_{\text{OTS}}$ is a set of edges. The edge $e = (l, \sigma, l')$ denotes that the face σ of box l is shared between l and l'.

Figure 3 shows an example OTS for a simple scenario. The OTS locations are the 15 non-obstacle boxes, which includes a goal box for the reach-avoid task. The OTS edges are shown as arrows; for example, $e = (l_6, (\sigma_-, \sigma_-), l_1) \in E_{\text{OTS}}$.

B. Maneuver Automaton

The maneuver automaton (MA) is a hybrid system whose discrete states correspond to a finite set of motion primitives, and whose edges capture both constraints on feasible successive motion primitives and constraints on which faces of a box can be reached using a specific motion primitive. The concept of a MA was introduced in [12] as a way to simplify the control design by invoking canonical high-level behaviors. By concatenating motion primitives, complex behaviors such as reach-avoid can be achieved.
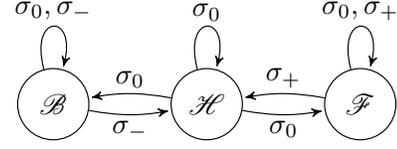

Fig. 4. A manuever automaton for double integrator dynamics, $p = 1$, with three motion primitives: Hold ($\mathcal{H}$), Forward ($\mathcal{F}$), and Backward ($\mathcal{B}$).

The *maneuver automaton* (MA) is a tuple $\mathcal{H}_{\text{MA}} = (Q_{\text{MA}}, \Sigma, E_{\text{MA}}, X_{\text{MA}}, I_{\text{MA}}, G_{\text{MA}}, R_{\text{MA}}, Q_{\text{MA}}^0)$, where

State Space $Q_{\text{MA}} = M \times \mathbb{R}^n$ is the hybrid state space, where M is a finite index set of *motion primitives*.

Labels Σ is the same set of labels used in the OTS, denoting the faces of a box.

Edges $E_{\text{MA}} \subset M \times \Sigma \times M$ is a set of edges. The edge $e = (m, \sigma, m') \in E_{\text{MA}}$ denotes that the motion primitives m and m' may be executed in sequence by the system (1) when the output trajectory crosses the face σ of the current box.

Vector Fields X_{MA} assigns a closed-loop vector field of (1) for each motion primitive. Associated with this closed-loop vector field is a feedback controller that implements the desired motion primitive.

Invariants I_{MA} assigns an invariant region in the state space over which each motion primitive's feedback controller is defined. The projection of each invariant region onto the output space is contained in a box.

Enabling and Reset Conditions Associated with each edge $e \in E_{\text{MA}}$ of the MA is an enabling condition $g_e \in G_{\text{MA}}$ that defines the set of states for which the edge can be taken. Likewise, the reset condition $r_e \in R_{\text{MA}}$ determines the new state value after the edge is taken.

Initial Conditions $Q_{\text{MA}}^0 \subset Q_{\text{MA}}$ is a set of initial states.

Remark 3.1: Formally, an automaton is termed non-deterministic if there exists a state with more than one outgoing edge with the same label. In this way, the MA is non-deterministic as there are generally multiple selections for successive motion primitives. However, there is another relevant notion of non-determinism. Viewing only the discrete part of the MA, a given motion primitive may also have multiple edges with different labels, which has the interpretation that output trajectories can reach different faces non-deterministically. In contrast, in the full MA the face reached is a function of the initial condition.

C. Motion Primitives

A motion primitive is a closed-loop vector field that achieves a desired canonical behavior in the output space. The closed-loop vector field of each motion primitive is formed by invoking a pre-computed feedback controller associated with that motion primitive and applying it to the system (1).

Our design of motion primitives exploits three simplifications. First, we invoke Assumption 2.1 and the fact that all the boxes in the OTS are translations of each other to design motion primitives over the canonical box Y^* only. Second, we assume any feedback linearizations or transformations are
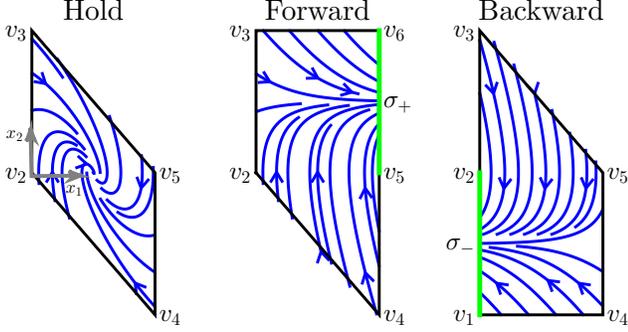
Fig. 5. The closed-loop vector fields for the Hold, Forward, and Backward motion primitives over their respective invariant regions in the (x_1, x_2) axes.
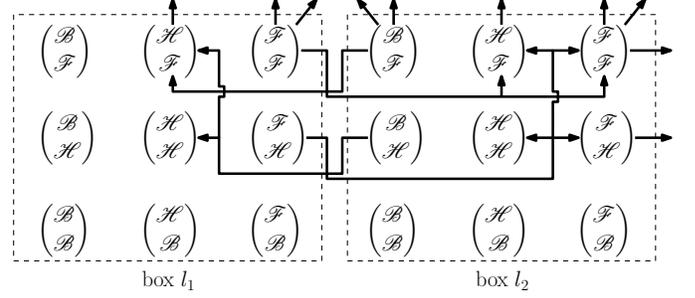

Fig. 6. A snapshot of the product automaton (PA) showing the system's motion capabilities over the two lower left boxes l_1 and l_2 from Figure 3. The motion primitives shown in Figure 4 are composed to obtain 9 motion primitives. A product state is a box equipped with a particular motion primitive. The arrow denotes an edge in the PA, where the outgoing location of the arrow corresponds to the face reached in a product state.

done separately, in order to focus our design of motion primitives to double integrator dynamics in each output coordinate, noting that Assumption 2.1 is satisfied. Finally, to achieve an overall desired behavior in the p-dimensional output space, we first design *atomic motion primitives* for a single output coordinate and then compose them. Typical canonical motion primitives for robotic agents operating in a 2D workspace include *Right*, *Left*, *Up*, and *Down*. For example, if the desired motion primitive in the output space is Right, then we want an atomic motion primitive for y_1 to increase in value, and another atomic motion primitive for y_2 to hold its value.

Let (1) be the double integrator system

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = u, \quad y = x_1, \tag{2}$$

where $x := (x_1, x_2) \in \mathbb{R}^2$, $u \in \mathbb{R}$, and the output y is the position, so that $p = 1$. The MA, depicted in Figure 4, consists of three atomic motion primitives, called *Hold*, *Forward*, and *Backward* and denoted as $\mathscr{H}$, $\mathscr{F}$, and $\mathscr{B}$, respectively. Since $p = 1$, the canonical box Y^* is simply a segment of fixed length. In the Forward motion primitive, output trajectories must exit the right face of Y^* corresponding to the label $\sigma_+ \in \Sigma$, whereas in Backward it is the left face with $\sigma_- \in \Sigma$. For all three motion primitives, the label $\sigma_0 \in \Sigma$ corresponds to output trajectories having not yet crossed a face of Y^*. These behaviors are exhibited in the closed-loop vector fields shown in Figure 5.

We synthesize controllers to achieve these closed-loop vector fields using reach control theory. In particular we employ standard design tools for reachability to a facet on a polytope for affine systems [16], [18]. Each motion primitive's invariant region is a polytopic subset constructed as the convex hull of the vertices v_k, $k \in \{1, \dots, 6\}$, see Figure 5. The vertices are determined by the segment length $d > 0$, maximum control value $u^* > 0$, and the derived maximum speed $v^* = \sqrt{du^*} > 0$. Specifically, $v_1 = (0, -v^*)$, $v_2 = (0, 0)$, $v_3 = (0, v^*)$, $v_4 = (d, -v^*)$, $v_5 = (d, 0)$, and $v_6 = (d, v^*)$.

For each $m \in \{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$, we employ an affine feedback

$$u_m(x) = K_m x + g_m. \tag{3}$$

We set $K_{\mathscr{H}} = \begin{bmatrix} k_1 & k_2 \end{bmatrix}$, $K_{\mathscr{F}} = K_{\mathscr{B}} = \begin{bmatrix} 0 & k_2 \end{bmatrix}$, $g_{\mathscr{H}} = g_{\mathscr{F}} = u^*$, $g_{\mathscr{B}} = -u^*$, $k_1 := -2u^*/d$, and $k_2 := -2u^*/v^*$.

These controllers can be derived by solving reach control problems on a triangulation of the polytopic subsets [16], [18] and imposing continuity constraints. We omit further details, but we refer to our previous work [1] for similar controllers.

In essence, these polytopic subsets are designed precisely so that the closed-loop vector fields over them can be pieced together and faithfully represent the discrete transitions defined in the MA shown in Figure 4. For example, the transition $(\mathscr{F}, \sigma_+, \mathscr{H}) \in E_{\text{MA}}$ is possible because when an output trajectory exits the right face of Forward as shown in green on Figure 5, it reappears on the left side of Hold (activating the enabling and reset conditions). The transitions labelled with σ_0 can be ignored when $p = 1$ and only play a role in correctly generating a MA for $p > 1$, which is described next.

The p-dimensional MA is obtained by parallel composing the single output MA shown in Figure 4 p-times. Our three single-output motion primitives, $\{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$, give rise to 3^p p-dimensional motion primitives, denoted as $\prod_{i=1}^{p}\{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$. For example, $(\mathscr{F}, \mathscr{H})$ would implement the motion primitive Right in 2D. The composed transitions are obtained by considering all possible combinations of one-dimensional transitions in each output. Consequently, this gives rise to non-deterministic motion primitives, such as $(\mathscr{F}, \mathscr{F})$, but these may be optionally pruned out in the parallel composition.

D. Product Automaton

The product automaton (PA) is the synchronous product of the OTS and the discrete part of the MA. It models the combined constraints on successive locations in the OTS and successive motion primitives. As such, it captures the overall feasible motions of the system. Any high-level plan must adhere to these feasible motions.

We define the *product automaton* (PA) to be the tuple $\mathcal{A}_{\text{PA}} = (Q_{\text{PA}}, \Sigma, E_{\text{PA}}, Q_{\text{PA}}^f, D_{\text{PA}}, H_{\text{PA}})$, where

State Space $Q_{\text{PA}} \subset L_{\text{OTS}} \times M$ is a finite set of states, denoting a box equipped with a motion primitive.

Labels Σ is the same set of labels used by the OTS and MA.

Edges $E_{\text{PA}} \subset Q_{\text{PA}} \times \Sigma \times Q_{\text{PA}}$ is a set of edges. In particular, an edge $e = (q, \sigma, q') \in E_{\text{PA}}$, where $q = (l, m)$ and $q' =$
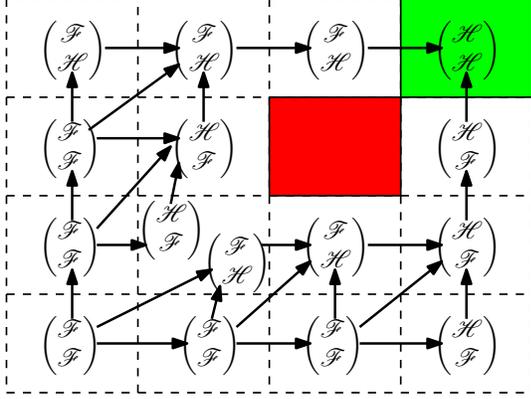
Fig. 7. This figure shows a discrete control policy for the scenario shown in Figure 3.

(l', m'), if $(l, \sigma, l') \in E_{\text{OTS}}$ and $(m, \sigma, m') \in E_{\text{MA}}$.

Final Condition $Q_{\text{PA}}^f \subset Q_{\text{PA}}$ is the set of final states. A product state (l, m) is a final state if the box l is labelled as a goal.

Discrete and Terminal Cost D_{PA} is the instantaneous cost associated with each edge in E_{PA} and H_{PA} is the terminal cost associated with each final state in Q_{PA}^f.

Referring to Figure 6 for some examples, box l_1 equipped with $(\mathscr{F}, \mathscr{H})$ is a product state, which will reach box l_2 in finite time through the face (σ_+, σ_0). As can be verified, this product state has four edges each with the label (σ_+, σ_0), corresponding to the feasible successive motion primitives $(\mathscr{H}, \mathscr{H})$, $(\mathscr{F}, \mathscr{H})$, $(\mathscr{H}, \mathscr{F})$ and $(\mathscr{F}, \mathscr{F})$. Each edge leads to a product state on the box l_2. On the other hand, box l_1 equipped with $(\mathscr{F}, \mathscr{F})$ is another product state; however, only one of the three faces (σ_+, σ_0), (σ_0, σ_+), and (σ_+, σ_+) will be reached in finite time. For each possibility there is a different set of edges. The PA inherits its non-deterministic properties from the MA, see Remark 3.1.

E. Hybrid Control Strategy

Once the PA is constructed, we employ a variation of a standard non-deterministic Dijkstra algorithm [17], [19]. From this, we obtain a discrete feedback control policy, which assigns one successive motion primitive for each product state and for each possible face that can be reached from that product state. The hybrid control strategy consists of the discrete control policy and the low-level feedback controller associated with each motion primitive.

Figure 7 shows a possible control policy for the scenario from Figure 3. Although different routes may be taken from the same product state depending on the face reached, the control policy ensures that all paths lead to the goal.

IV. QUADROCOPTER APPLICATIONS

A. Quadrocopter Modeling

The standard quadrocopter dynamics model is ubiquitous in the literature; see, for example, [20]. The vehicle dynamics are described by six degrees of freedom and are nonlinear. It is well known that this model is differentially flat [11]. As a
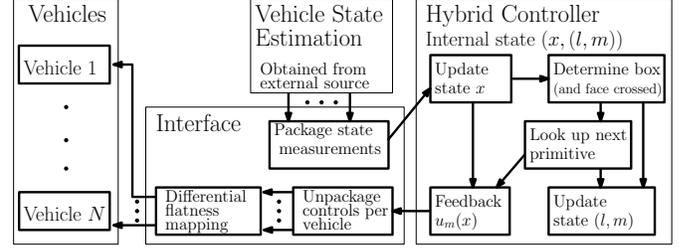

Fig. 8. Interface between multiple vehicles and the framework with $p = 3N$ outputs. The hybrid controller internal state consists of the joint state measurement of all the vehicles and includes the current (joint) box, l, and the current (joint) motion primitive, m. The internal state is updated via external state measurements (assumed to be given) and is used to compute the feedback controls.

result, the dynamics for the position (x_w, y_w, z_w) in the world frame each reduce to a double integrator and we are able to use the motion primitives from Section III-C.

B. Interfacing Multiple Quadrocopters

To apply our framework to a joint reach-avoid task among N quadrocopters, a copy of the gridded 3D workspace must be associated with each vehicle, resulting in a total of $p = 3N$ outputs. The p-dimensional MA representing the asynchronous motion capabilities of the multi-vehicle system is obtained by parallel composing the single-output MA from Section III-C. We employ an exhaustive search over the multi-vehicle configurations to label the p-dimensional obstacle boxes, which accounts for real-world obstacles and pairwise collisions between any two vehicles.

The multi-vehicle reach-avoid problem can be solved using our proposed methodology, following the steps shown in Figure 2. The output of the methodology is a hybrid controller, which is fully computed offline. We discuss its computational complexity in Section V. Once the hybrid controller is computed, the system can successfully execute the reach-avoid task from any starting configuration corresponding to a valid initial condition of the hybrid controller. The runtime workflow is depicted in Figure 8, showing how the hybrid controller interfaces with the multi-vehicle system. Due to the simplicity of a box partition and assuming that the next motion primitive can be looked-up in constant time, each runtime component requires a negligible amount of computation, even for a large number of vehicles and outputs.

C. Experimental Results

Our experimental platform is the Parrot AR.Drone 2.0 interfaced with the ROS *ardrone autonomy* package. We used an external motion capture system to obtain the state estimates for our feedback controller (3), which was run at 70 Hz.

Due to limited space, we illustrate one interesting scenario where two quadrocopters must coordinate switching places through a narrow passage, see Figure 9. Additional video examples include a simple, single quadrocopter maneuver, see http://tiny.cc/quadrocopterPlanar, and several 2D and 3D scenarios with two quadrocopters, see http://tiny.cc/quad5scenes.
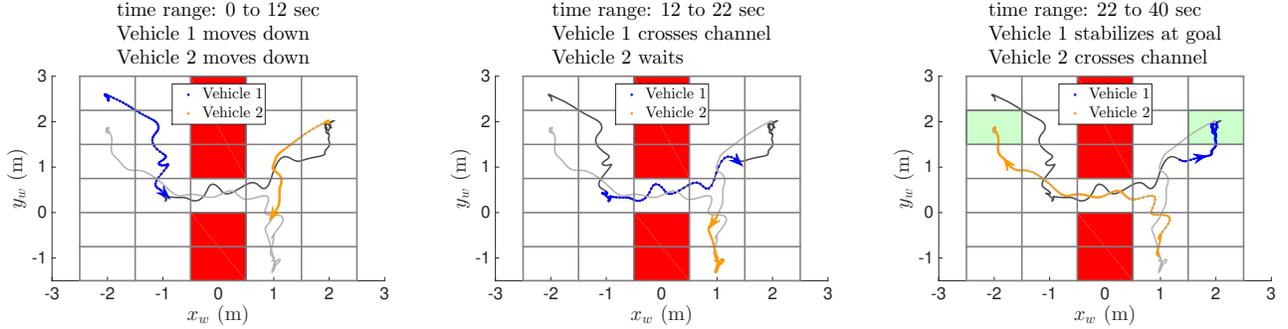
Fig. 9. Experimental results for the scenario where two quadrocopters must switch places through a narrow passage, where Vehicle 1 (blue) starts on the left and Vehicle 2 (orange) starts on the right. The grayed out trajectories show the full path followed by the vehicles; the color portion corresponds to the indicated time interval. The maneuver is shown over three time segments, with goals highlighted in green on the right. We observe that since Vehicle 2 gets to the passage first, it makes room and waits for Vehicle 1 to pass first. Noisiness in the trajectories is due to mutual aerodynamic effects, to be contrasted with Figure 11. Although these effects are not accounted for, our hybrid feedback controller safely completes the task.
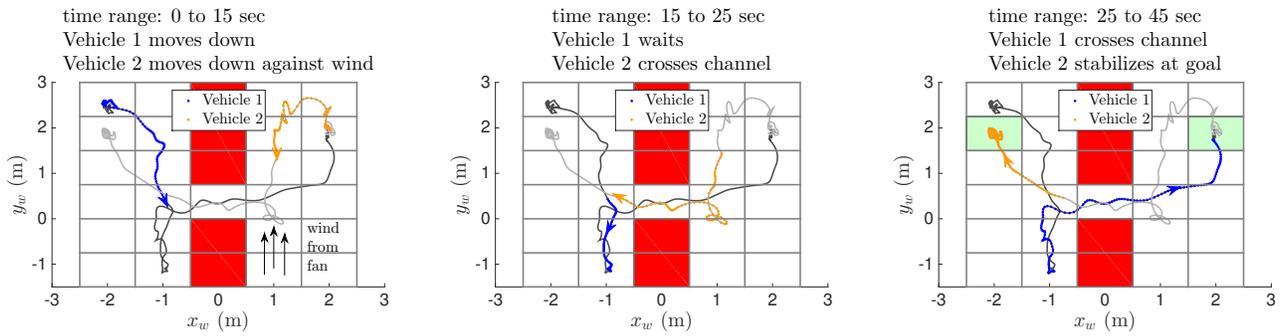

Fig. 10. The same scenario as in Figure 9 above, but now an unmodelled wind disturbance causes a delay in Vehicle 2. To compensate, Vehicle 1 makes room and waits for Vehicle 2 to pass first, opposite to the nominal case. The same hybrid controller was used as in the nominal run of Figure 9.

For the example shown here, the 3D workspace is partitioned into a $(5 \times 6 \times 1)$ grid consisting of boxes having side lengths of 1, 0.75, and 3 meters, respectively. The planar (x_w, y_w)-view is shown in Figure 9, where the red boxes represent the physical obstacles forming the passage. The z_w-direction is stabilized using the Hold motion primitive, so the overall number of outputs is effectively $p = 4$.

A nominal experimental run is shown in Figure 9, depicting the (x_w, y_w)-trajectories of each quadrocopter. The trajectories are divided into several time slots in order to show how the vehicles coordinate moving through the passage. As a consequence of our non-deterministic motion primitives, both vehicles move simultaneously for most of the run. Notice that one vehicle (Vehicle 2 in this case) moves into a corner and waits to give space for the other vehicle to pass through the passage in order to maintain a sufficient safety margin.

In contrast, an experimental run with disturbances induced by a fan is shown in Figure 10. Starting from the same configuration as the nominal run, Vehicle 2's motion was delayed due to a persistent wind disturbance, see the annotation. As such, the quadrocopters automatically executed a significantly different path compared to the nominal run (Vehicle 1 moved into a corner this time). In Figures 9 and 10 mutual aerodynamic effects resulted in "wobbly" trajectories, which can be contrasted to the single vehicle run in Figure 11.

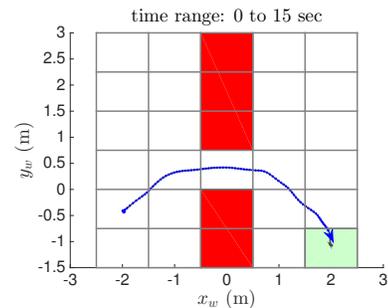As long as disturbances are not too severe, the underlying

Fig. 11. A single-vehicle run for a similar scenario as considered in Figures 9 and 10. The trajectory shown in blue is smooth and efficient, which illustrates that the noisy trajectories in the multi-vehicle case arise from mutually caused aerodynamic effects.

feedback-based motion primitives will compensate for it and, if necessary, the high-level control policy will guide the vehicles along a new path to the goal configuration. Thus, our hybrid control strategy achieves a robust maneuver that does not require any online recomputation or timing estimates in the trajectories.

V. DISCUSSION

We analyze the complexity of the computations associated with our methodology. Using the MA shown in Figure 4 for each output, Figure 12 shows the total time to compute the OTS (with one goal and no obstacles), the parallel composed
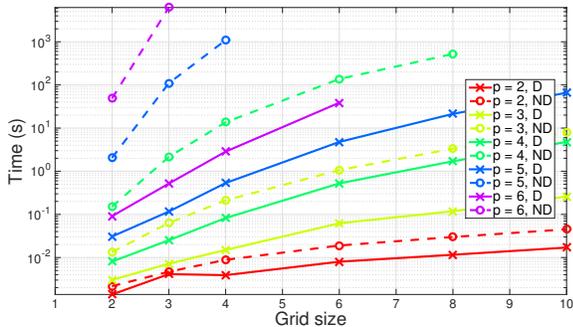
Fig. 12. Offline computation times for various scenario sizes. The grid size refers to the number of boxes in each output; for example, if $p = 3$ and the grid size is 4, there are a total of 4^3 OTS boxes. We compare the computation using non-deterministic (ND) versus deterministic (D) motion primitives.

MA (with and without non-deterministic motion primitives), the PA, and a control policy (employing a non-deterministic Dijkstra algorithm) for various scenario sizes. Our implementation was done in Python 2.7.10 and computations were performed on a 64-bit Lenovo ThinkPad with a 2.6 GHz Intel Core i7 processor and 15.3 GiB RAM.

From Figure 12, it is evident that the offline computation time is mild for moderate problem sizes. For small problems involving one or two vehicles, the computation often takes under a second, and all cases shown take less than two hours. For the scenario shown in Section IV-C, the total computation time was about 8 seconds. A significant reduction in computation time is possible by restricting to use only deterministic motion primitives, at the expense of limiting simultaneous motion capabilities. Finally, since substantial motion can be robustly achieved through a single box, often the grid size can be quite coarse.

While it may be expensive to compute the full solution to a larger multi-vehicle system, it only needs to be performed offline once. Therefore, it may be an effective way to coordinate multiple vehicles through critical areas. Alternatively, the small computation times for $p \leq 3$ suggest a decentralized approach, in which each agent treats the other vehicles as obstacles and recomputes its hybrid control strategy over a small neighbourhood of boxes. Finally, faster computation times may be possible through specialized shortest path algorithms that do not explore the full product automaton structure.

VI. CONCLUSION

We have developed a modular, hierarchical framework for motion planning of heterogenous agents in known environments. It consists of various modules that describe the partitioned workspace, low-level control design, and high-level motion plan. Overall we obtain a two-level control design which is highly robust, modular, and conceptually elegant. We presented a specific maneuver automaton for a double integrator system, and we illustrated its effectiveness to coordinate a maneuver between two quadrocopters. Extensions of this work include a decentralized implementation,

a specialized shortest path algorithm to better exploit the modularity of our framework, and a generalization to control specifications expressed in linear temporal logic.

REFERENCES

[1] M. Vukosavljev, I. Jansen, M. E. Broucke, and A. P. Schoellig, "Safe and Robust Robot Maneuvers Based on Reach Control," in *Proc. of the International Conference on Robotics and Automation*, 2016.

[2] S. R. Lindemann and S. M. LaValle, "Simple and Efficient Algorithms for Computing Smooth, Collision-free Feedback Laws Over Given Cell Decompositions," *International Journal of Robotics Research*, vol. 28, no. 5, pp. 600-621, 2009.

[3] N. Ayanian and V. Kumar, "Decentralized Feedback Controllers for Multiagent Teams in Environments With Obstacles," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 878-887, 2010.

[4] D. Heß, M. Althoff, and T. Sattel, "Formal Verification of Maneuver Automata for Parameterized Motion Primitives," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2014.

[5] 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.

[6] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343-352, 2009.

[7] D. C. Conner, A. A. Rizzi, and H. Choset, "Composition of Local Potential Functions for Global Robot Control and Navigation," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2003.

[8] 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," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2014.

[9] P. Mihail, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices." *Journal of Field Robotics*, vol. 26, no. 3, pp. 308-333, 2009.

[10] J. Liu, U. Topcu, N. Ozay, and R. M. Murray, "Synthesis of Reactive Control Protocols for Differentially Flat Systems," in *Proc. of the Conference on Decision and Control*, 2012.

[11] D. Zhou and M. Schwager, "Vector field following for quadratures using differential flatness," in *Proc. of the International Conference on Robotics and Automation*, 2014.

[12] 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.

[13] L.C.G.J.M. Habets, P.J. Collins, and J.H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Transactions on Automatic Control*, vol. 51, no. 6, pp. 938-948, 2006.

[14] V. Raman and H. Kress-Gazit, "Synthesis for multi-robot controllers with interleaved motion," in *Proc. of the International Conference on Robotics and Automation*, 2014.

[15] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Motion and Action Planning under LTL Specifications using Navigation Functions and Action Description Language," in *Proc. of the International Conference on Intelligent Robots and Systems*, 2013.

[16] B. Roszak and M. E. Broucke, "Necessary and sufficient conditions for reachability on a simplex," *Automatica*, vol. 42, no. 11, pp. 1913-1918, 2006.

[17] M. E. Broucke, S. Di Gennaro, M. Di Benedetto, and A. Sangiovanni-Vincentelli, "Efficient solution of optimal control problems using hybrid systems," *SIAM Journal on Control and Optimization*, vol. 43, no. 6, pp. 1923-1952, 2005.

[18] G. Ashford and M. E. Broucke, "Design of reach controllers on simplices," in *Proc. of the Conference on Decision and Control*, 2013.

[19] E. M. Wolff, U. Topcu, and R. Murray "Optimal Control of Non-deterministic Systems for a Computationally Efficient Fragment of Temporal Logic." in *Proc. of the Conference on Decision and Control*, 2013.

[20] S. Lupashin, M. Hehn, M. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: the Flying Machine Area," *Mechatronics*, vol. 24, no. 1, pp. 41-54, 2014.