Abstract—We propose a distributed and cooperative motion and task control scheme for a team of mobile robots that are subject to dynamic constraints including inter-robot collision avoidance and connectivity maintenance of the communication network. Moreover, each agent has a local high-level task given as a Linear Temporal Logic (LTL) formula of desired motion and actions. Embedded graph grammars (EGGs) are used as the main tool to specify local interaction rules and switching control modes among the robots, which is then combined with the model-checking-based task planning module. It is ensured that all local tasks are satisfied while the dynamic constraints are obeyed at all time. The overall approach is demonstrated by simulation and experimental results.

I. INTRODUCTION

The control of multi-robot systems could normally consist of two goals: the first is to accomplish high-level system-wise tasks, e.g., formation and flocking [21], task assignment [20] and collaboration [19]; the second is to cope with constraints that arise from the inter-robot interactions, e.g., collision avoidance [5] and communication maintenance [21]. These two goals are often dependent and heavily coupled since it is essential to consider one when trying to fulfill another. For instance, it is unlikely that a multi-robot formation method would work if the inter-robot collision is not addressed, nor a collaborative task assignment scheme would work if the communication network among the robots is not ensured to be connected. Thus in this work, we tackle some aspects of both goals at the same time.

Regarding the high-level task, we rely on Linear Temporal Logic (LTL) as the formal language that can describe planning objectives more complex than the well-studied point-to-point navigation problem. The task is specified as an LTL formula with respect to an abstraction of the robot motion [1], [3]. Then a high-level discrete plan is found by off-the-shelf model-checking algorithms [2], which is then implemented through the low-level continuous control modes among the robots, which is then combined with the model-checking-based task planning module. It is ensured that all local tasks are satisfied while the dynamic constraints are obeyed at all time. The overall approach is demonstrated by simulation and experimental results.

II. PRELIMINARIES

A. Embedded Graph Grammars

Here we review some basics of Embedded Graph Grammars (EGGs). For a detailed description, see [14], [15]. Let $\Sigma$ be a set of pre-defined labels. A labeled graph is defined as the quadruple $G = (V, E, l, e)$, where $V$ is a set of vertices, $E \subset V \times V$ is a set of edges, $l : V \to \Sigma$ is a vertex labeling function, and $e : E \to \Sigma$ is an edge labeling function. Given a continuous state space $X$ for the vertices,
an embedded graph is given by $\gamma = (G, \gamma)$, where $G$ is a labeled graph and $x : V \rightarrow X$ is a realization function. We use $G_{\gamma}, x_\gamma$ to denote the labeled graph and continuous states associated with $\gamma$. The set of allowed embedded graphs being considered is denoted by $\Gamma$. Furthermore, an embedded graph transition is a relation $A \subseteq \Gamma \times \Gamma$ such that $(\gamma_1, \gamma_2) \in A$ implies $\gamma_1 = x_{\gamma_2}$ and $G_{\gamma_1} \neq G_{\gamma_2}$. The rules and conditions associated with the transitions are called graph grammars.

B. Linear Temporal Logic

The basic ingredients of a Linear Temporal Logic (LTL) formula are a set of atomic propositions $AP$ and several boolean or temporal operators, formed by the syntax [2]: $\varphi := \top \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \varphi_1 \lor \varphi_2$, where $a \in AP$ and $\top \in \{True, \}$. Other operators like $\Box$ (always), $\Diamond$ (eventually), $\Rightarrow$ (implication) and the semantics of LTL formulas can be found in Chapter 5 of [2]. There is a union of infinite words that satisfy $\varphi$: Words($\varphi$) = $\{ \sigma \in (2^{AP})^\omega \mid \sigma \models \varphi \}$, where $\models \subseteq (2^{AP})^\omega \times \varphi$ is the satisfaction relation. LTL formulas can be used to specify various control tasks, such as safety ($\Box \varphi_1, \text{globally avoiding } \varphi_2$), ordering ($\Diamond \varphi_1 \lor \Diamond \varphi_2 \land \Diamond \varphi_3$), $\varphi_1, \varphi_2, \varphi_3$ hold in sequence), response ($\varphi_1 \Rightarrow \varphi_2$, if $\varphi_1$ holds, $\varphi_2$ will hold in future), repetitive surveillance ($\Box \Diamond \varphi, \varphi$ holds infinitely often).

III. PROBLEM FORMULATION

A. Robot Dynamics

Consider a team of $N$ mobile robots (agents) in an obstacle-free 2D workspace, indexed by $N = \{1, 2, \ldots, N\}$. Each agent $i \in N$ satisfies the unicycle dynamics:

$$\begin{align*}
\dot{x}_i &= v_i \cos(\theta_i), \\
\dot{y}_i &= v_i \sin(\theta_i), \\
\dot{\theta}_i &= w_i,
\end{align*}$$

where $s_i = (x_i, y_i, \theta_i) \in \mathbb{R}^3$ is the state with position $p_i = (x_i, y_i)$ and orientation $\theta_i$; and $u_i = (v_i, w_i) \in \mathbb{R}^2$ is the control input as linear and angular velocities, bounded by $v_{\text{max}}$ and $w_{\text{max}}$. Agent $i$ has reference linear and angular velocities $V_i < v_{\text{max}}$ and $W_i < w_{\text{max}}$, respectively. Each agent occupies a disk area of $\|p_i - p_j\| \leq r$, where $r > 0$ is the radius of its physical volume. A safety distance $d > 2r$ is the minimal inter-agent distance to avoid collisions. Moreover, agents $i, j \in N$ can only communicate if $\|p_i - p_j\| \leq d$, where $d > d$ is the communication radius.

Definition 1: Agents $i, j \in N$ are: in collision if $\|p_i(t) - p_j(t)\| < d$; neighbors if $\|p_i(t) - p_j(t)\| \leq d$.

Given the agent states, an embedded graph $\gamma(t)$ is defined as $\gamma(t) = (G(t), p(t))$, where $G(t) = (N, E(t))$ with $(i, j) \in E(t)$ if $\|p_i(t) - p_j(t)\| < d$, $\forall i, j \in N$, $i \neq j$; $p(t)$ is the stack vector of all $p_i(t)$. Then we define the set of allowed embedded graphs $\Gamma_d$ as follows:

Definition 2: An embedded graph $\gamma(t) = (G(t), p(t))$ is allowed that $\gamma(t) \in \Gamma_d$ if (i) $\|p_i(t) - p_j(t)\| < d$, $\forall i, j \in N$, $i \neq j$; and (ii) the graph $G(t)$ is connected.

B. Local Task Specification over Motion and Actions

For each agent $i \in N$, there is a set of points of interest in the workspace, denoted by $Z_i = \{z_{i1}, z_{i2}, \ldots, z_{iM_i}\}$, where $z_{i\ell} \in \mathbb{R}^2, \forall \ell = 1, 2, \ldots, M_i$, where $M_i > 0$. Each point satisfies different properties. Furthermore, it is capable of performing a set of actions, described by the action primitives $\Sigma_i = \{a_{i1}, a_{i2}, \ldots, a_{iK_i}\}$. Combining these two aspects, we can derive a complete motion and action model for agent $i$ as a finite transition system (FTS) $M_i = (\Pi_i, \rightarrow_i, (\Pi_i,0,AP_i, L_i))$, where $\Pi_i = Z_i \times \Sigma_i$ is the set of states; $\rightarrow_i : \Pi_i \rightarrow 2^{\Pi_i}$ is the transition relation; $\Pi_i \circ 0 \subseteq \Pi_i$ is the set of initial states; $AP_i$ is the set of atomic propositions over workspace property and action primitives; $L_i : \Pi_i \rightarrow 2^{AP_i}$ is the labeling function that returns the set of propositions satisfied at each state. We omit the details about how to construct $M_i$ here due to limited space and refer the readers to [8], [10]. The local task for each agent $i \in N$ is specified as an LTL formula $\varphi_i$ over $AP_i$ as described in Section II-B.

Definition 3: The task $\varphi_i$ is satisfied if there exists a sequence of time instants $t_0, t_1, t_2, \ldots$ and a sequence of states $\pi_{t_0}, \pi_{t_1}, \pi_{t_2}, \ldots$ such that $\pi_{t_k} = (z_{i\ell_k}, a_{i\ell_k})$ where $z_{i\ell_k} \in Z_i$ and $a_{i\ell_k} \in \Sigma_i$; at time $t_k, \|p_i(t_k) - z_{i\ell_k}\| \leq c_i$ where $c_i > 0$ is a given threshold for reaching a point of interest and the action $a_{i\ell_k}$ is performed at $z_{i\ell_k}, \forall k = 0, 1, 2, \ldots$; and $L_i(\pi_{t_0})L_i(\pi_{t_1})L_i(\pi_{t_2}) \cdots \models \varphi_i$.

C. Problem Statement

Design a distributed motion control scheme such that $\varphi_i$ is satisfied, $\forall i \in N$, while at the same time $\gamma(t) \in \Gamma_d, \forall t \geq 0$.

IV. SOLUTION

The proposed solution consists of two major parts: the embedded graph grammars (EGGs) design and the local task coordination, of which the details are given in the sequel. Then we combine them as the complete solution, where we also prove the correctness formally.

A. EGGs Design

The design of EGGs involves: (i) the workspace discretization; (ii) essential building blocks; (iii) graph grammars.

1) Workspace Discretization: The 2-D workspace is discretized into uniform grids by a quantization function, through which we transform the collision avoidance and connectivity constraints into relative-grid positions.

Definition 4: Given a point $(x, y) \in \mathbb{R}^2$, its grid position is given by the function $\text{GRID} : \mathbb{R}^2 \rightarrow \mathbb{Z}^2$:

$$(g_x, g_y) \equiv \text{GRID}(x, y) \equiv ([x/d], [y/d]),$$

where $[\cdot]$ is the round function that returns the closest integer $([0.5] = 1)$ and $d$ is the safety distance introduced earlier.

Given that $p_i(t) = (x_i(t), y_i(t))$ at time $t > 0$, the grid position of agent $i$ is given by $g_i(t) \equiv (g_{i1}(t), g_{i1}(t)) = (\text{GRID}(x_i(t), y_i(t)))$. Now consider two agents $i$ and $j$ whose grid positions are given by $g_i(t)$ and $g_j(t)$.

Definition 5: The collision function $\text{COLLIDE} : \mathbb{Z}^2 \times \mathbb{Z}^2 \rightarrow B$ satisfies: $\text{COLLIDE}(g_i(t), g_j(t)) \equiv \bot$ if $|g_{i1} - g_{j1}| \geq 2 \text{ or } |g_{i2} - g_{j2}| \geq 2$; otherwise, $\text{COLLIDE}(g_i(t), g_j(t)) \equiv \top$. The neighboring function $\text{NEIGHBOR} : \mathbb{Z}^2 \times \mathbb{Z}^2 \rightarrow B$ satisfies: $\text{NEIGHBOR}(g_i(t), g_j(t)) \equiv \top$ if it holds that $||g_{i1} - g_{j1}+1|| + 1, |g_{i2} - g_{j2}+1|| \leq \lambda_d$, where $\lambda_d \equiv \sqrt{7}/d > 1$; otherwise, $\text{NEIGHBOR}(g_i(t), g_j(t)) \equiv \bot$. 


Lemma 1: By Definition 1, agents i and j are collision-free at time $t > 0$ if $\text{COLLIDE}(g_i(t), g_j(t)) = \bot$; they are connected if $\text{NEIGHBOR}(g_i(t), g_j(t)) = \top$.

Proof: Simple algebra based on Definitions 4 and 5. ■

2) Building Blocks: We introduce five building blocks in this part that are essential for the construction of EGGs.

(I) Labels on vertices and edges. The first building block is the modified embedded graph $\gamma(t) = (G(t), p(t))$ where $G(t) = (N, E(t), l, e)$, where $l$ and $e$ are the vertex and edge labeling functions. Each vertex has a label with three named fields $\{\text{id}, \text{mode}, \text{data}\}$, where $\text{id}$ is the agent ID; $\text{mode}$ is the agent status, including $\{\text{check, static, move}\}$, and $\text{data}$ stores data for the execution, which has three sub-fields $\{\text{nb, pt, gl}\}$, where $\text{nb}$ saves the set of other agents’ IDs; $\text{pt}$ saves a tentative path; and $\text{gl}$ saves a positive gain parameter. Moreover, the edge between neighbors has the named field id., i.e., the edge from agent $i$ to $j$ has the id as $(i, j)$. We use dot notation to indicate the value of label fields. An agent is static if its mode is static or active if its mode is move.

To start with, we need the notion of a local sub-graph for agent $i \in N$, denoted by $G_i(t) = (V_i(t), E_i(t))$, where $V_i(t) = \{i\} \cup N_i(t)$, where $N_i(t) = \{j \in N | (i, j) \in E_i(t)\}$; (ii) $(j, k) \in E_i(t)$ if $(j, k) \in E(t)$, $\forall j, k \in V_i(t)$. Clearly, $G_i(t)$ is a sub-graph of $G(t)$ and it can be constructed locally by agent $i$. Clearly if $G(t)$ is connected, then $G_i(t)$ is connected, $\forall i \in N$.

(II) Neighbor marking scheme. The second building block is the mechanism to maintain graph connectivity while the agents are moving. The main idea is to choose locally some agents to be static and the others to be active. Assume that agent $i \in N$ satisfies $s_{\text{mode}} = \text{move}$. We denote by $N_i^s(t) = \{j \in N_i(t) | j.\text{mode} = \text{static}\}$ the set of static neighbors; $N_i^a(t)$ the set of active neighbors; and the others are in the check mode. A marking scheme of agent $i$ at time $t > 0$ marks a subset of its neighbors, denoted by $N_i^m(t) \subseteq N_i(t)$, as the potential agents to become static.

Definition 6: The marked set of neighbors $N_i^m(t)$ is allowed if: (i) for any neighbor $j \in N_i(t)$, it holds that either $j \in N_i^m(t)$ or there exists $g \in N_i^a(t)$ that $(j, g) \in E_i(t)$; (ii) $N_i^s(t) \subseteq N_i^m(t)$ and $N_i^m(t) \cap N_i^a(t) = \emptyset$. ■

Definition 7: The marked sub-graph $G_i^m(t)$ is $\triangleq (V_i^m(t), E_i^m(t))$ has $V_i^m(t) = \{i\} \cup N_i^m(t)$ and $(j, k) \in E_i^m(t)$ if $(j, k) \in E_i(t)$, $\forall j, k \in V_i^m(t)$.

(III) Potential path synthesis. The third building block is the synthesis algorithm to derive a local path for an active agent $i \in N$ to move towards its point of interest $z_{id}$ of $z_{id} = (z_{id}^x, z_{id}^y) \in Z_i$ while keeping connected and collision-free to all its marked neighbors in $N_i^m$.

Denote by $p_i$ the tentative discrete path of agent $i$ with length $L_i \geq 1$ that obeys the following structure:

$$p_i^t = q_i^t q_i^{t+1} \cdots q_i^{t+L_i}$$

where $q_i^t = (s_i^t, l_i^t, v_i^t)$ is a 3-tuple with the desired state $s_i^t = (x_i^t, y_i^t, \theta_i^t) \in \mathbb{R}^3$, the approximated time $t_i^t$ when $s_i$ will be reached, and the linear velocity $v_i^t$ at $q_i^t$ when heading towards $q_i^{t+1}$, $\forall l = 0, 1, \cdots, L_i$. Notice that $q_i^0 \triangleq (s_i(t), 0, V_i)$ initially, where $V_i$ is the reference linear velocity. Moreover, the position $p_i^t = (x_i^t, y_i^t)$ of $s_i^t$ should correspond to the center of a grid $g_i^t = \text{GRID}(p_i^t)$ and two consecutive positions $p_i^t$, $p_i^{t+1}$ correspond to two adjacent grids, $\forall l = 0, 1, \cdots, L_i - 1$. Given the current state $s_i(t)$ of agent $i$, the potential cost of $p_i$ is given by $\text{Cost}(p_i) \triangleq \sum_{l=0}^{L_i-1} \left(\|p_i^l - p_i^{l+1}\| + \alpha \cdot |\theta_i^l - \theta_i^{l+1}|\right)$, where the first term is the total traveled distance and the second term is the total turned angles; $\alpha > 0$ is the chosen weight on turning cost. To synthesize $p_i$, we consider the following problem:

$$\min_{p_i} \|p_i^{l+1} - z_{id}^x, p_i^{l+1} - z_{id}^y\| + \beta \cdot \text{Cost}(p_i),$$

such that $G_i^m(t)$ remains connected if $p_i = p_i^{l+1}$ and $\text{COLLIDE}(g_i(t), g_{j}(t)) = \bot$, $\forall l = 0, 1, \cdots, L_i$ and $\forall j \in N_i^m(t)$, where the first term is the tentative progress; $\beta > 0$ is a tuning parameter; and the conditions say that along $p_i$, $G_i^m$ should remain connected and collision-free.

The solution contains four steps: (i) determine the general search area. Given the positions of the marked agents, the general search area $S_i \subseteq \mathbb{Z}^2$ satisfies that $g_i = (g_{i}^x, g_{i}^y) \in S_i$ if $\text{NEIGHBOR}(g_i, g_j(t)) = \top$, for at least one neighbor $j \in N_i^m(t)$; (ii) remove any grid $g_i \in S_i$ that $G_i^m(t)$ is not connected if $g_i = g_{j}$ or $\text{COLLIDE}(g_i, g_{j}(t)) = \top$ for any neighbor $j \in N_i^m(t)$. Thus all elements of $p_i$ should belong to this general search area; (iii) the augmented-graph construction. We augment the states in $S_i$ with robot orientations and compute the transition cost based on function $\text{Cost}(\cdot)$; (iv) shortest path search. We search for the shortest paths from $n_0$ to every other node in $n_i$ by Djikstra’s algorithm. At last, find the destination $n_d^* \in n_i$ that minimizes the cost in (4). Denote the shortest path from $n_0$ to $n_d^*$ by $p_i^m = n_0 n_{i1} n_2 \cdots n_{Li-1} n_d^*$, where $n_i = (g_i, \theta_i) \in n_i$ and $L_i$ is the length of this path. Give the shortest path $p_i^m$ above, each element $q_i^t = (s_i^t, l_i^t, v_i^t) \in p_i$ can be derived by setting $s_i^t = (g_{i}^x - l_i^t \cdot d_i^x, g_{i}^y - l_i^t \cdot d_i^y, \theta_i^t)$ and $v_i^t = V_i$, $\forall l = 0, \cdots, L_i$, and $t_i^t$ is computed by $t_i^{l+1} = t_i^l + \frac{\|s_i^t - s_i^{l+1}\|}{W_i}$, $\forall l = 1, 2, \cdots, L_i$, which accumulates the time for agent $i$ to move from $s_i^t$ to $s_i^{l+1}$ with linear velocity $v_i^l$ and angular velocity $W_i$.

If a solution to (4) exists, the results are the tentative path $p_i$ and the associated $N_i^m$. Moreover, its tentative gain is given by $\chi_i = \|p_i^{l+1} - z_{id}\| - \|p_i(t) - z_{id}\|$. For the ease of notation, we denote this local path synthesis procedure by a single function: $(p_i, \chi_i) = \text{CHECK}(s_i(t), N_i(t), z_{id}, N_i^m)$. As a result, agent $i$ executes its tentative path $p_i$ by following and staying within the sequence of grids along $p_i$.

Lemma 2: Assume that $p_i$ has a solution at time $t_0 > 0$. If all marked neighbors in $N_i^m$ remain static and agent $i$ executes $p_i$, until $t_1 > t_0$, then $G_i^m(t)$ remains connected and all agents within $V_i^m(t)$ are collision-free, $\forall t \in [t_0, t_1]$.

Proof: Follows directly from the formulation of (4). ■

(IV) Path adaptation. The fourth building block is the path adaptation algorithm for any active agent while executing its tentative path. Assume that at time $t > 0$ an active agent $i$ may detect another agent $j \in N$ that does not belong to $N_i^m$, when its state $s_i(t)$ corresponds to $q_j^m \in p_j$ in (3), where $0 < w_0 < L_i$. We consider two cases below:
If \( j.\text{mode} = \text{static} \), then agent \( i \) only needs to check if its future path segment is in collision with this static agent \( j \). Its future path segment is given by \( p_i[t_0; L_i] = q_i^0, q_i^0 + 1, \ldots, q_i^L \), where \( t_i = (s_i^0, t_i^0, v_i^0) \) is defined in (3). Therefore if \( \text{COLLIDE}(q_i^0, g_j(t)) = \perp \), \( \forall w = w_0, w_0 + 1, \ldots, L_j \), it means they will not collide and \( p_i \) remains unchanged; otherwise, \( p_i \) is adapted by repeating the synthesis procedure, but with the new neighboring set \( N_i(t) \).

If \( j.\text{mode} = \text{move} \), then agent \( j \) is also moving and executing its path \( p_j \). In this case, it is more complicated to check whether they will be in collision. We assume that agent \( j \)’s position \( s_j(t) \) corresponds to \( q_i^w = p_j \), where \( 0 < v_i < L_j \). Its future path segment is given by \( p_j[t_0; L_j] = q_j^0, q_j^0 + 1, \ldots, q_j^L \), where \( q_j^i = (s_j^t, t_j, v_j^t) \) from (3). Given \( p_i[t_0; L_i] \) and \( p_j[t_0; L_j] \), a potential collision between agents \( i \) and \( j \) can be detected by the function:

\[ \text{COLLIDEPath}(p_i, p_j) = \perp, \]  

(5)

if \( \text{COLLIDE}(p_i^w, p_j^t) = \perp \) and \( |t_i^w - t_j^t| < \Delta_t \), for any \( p_i^w \in p_i[t_0; L_i] \) and any \( p_j^t \in p_j[t_0; L_j] \), where \( \Delta_t > 0 \) is a design parameter as the allowed time difference. Then agents \( i \) and \( j \) keep their current paths unchanged; otherwise, \( \text{COLLIDEPath}(p_i, p_j) = \top \) means that the paths may collide. For now, we assume that agent \( i \) is chosen to change its path \( p_i \). Let \( w_i \in \{w_0, w_0 + 1, \ldots, L_i \} \) be the smallest index within \( p_i[t_0; L_i] \) that a potential collision could happen by (5) and the associated index within \( p_j[t_0; L_j] \) is \( v_i \in \{v_0, v_0 + 1, \ldots, L_j \} \). Then agent \( i \) would avoid this collision by reducing its speed within the segment \( p_i[t_0; w_i] \), while \( p_i[t_0; L_i] \) remains unchanged.

To find a suitable linear velocity \( \nu < v_{\text{max}} \) for elements in \( p_i[t_0; w_i] \), we consider the optimization problem:

\[ \min_{0 < \nu < v_{\text{max}}} |V_i - \nu| \text{ such that } v_i^t = \nu, \forall w = w_0, \ldots, w_i; \]  

\[ \text{COLLIDE}(p_i^w, p_j^t) = \perp \text{ and } |t_i^w - t_j^t| < \Delta_t, \forall p_i^w \in p_i[t_0; L_i] \]  

and \( \forall p_j^t \in p_j[t_0; L_j] \), where \( V_i \) is the reference velocity. The conditions above ensure that \( p_i \) and \( p_j \) will not collide after adjusting the velocity. The above problem can be solved as follows: firstly, choose \( \nu = \max_{0 < w < L_i}(v_i^t) \) and a proper step size \( \delta_v > 0 \). Then gradually decrease \( \nu \) by \( \delta_v \) and check if the conditions above are fulfilled. If not, repeat this procedure until \( \nu = \nu^* \) is small enough and all conditions above are fulfilled. As a result, \( \nu^* \) is the suitable linear velocity for \( p_i[t_0; w_i] \). Moreover, the time instants \( \{v_i^w \} \) within \( p_i[t_0; L_i] \) are updated accordingly. If \( \nu < 0 \), then the initial position of agent \( i \) is in collision with parts of agent \( j \)’s path. Thus it changes its mode according to the EIGs defined later. For the ease of notation, we denote this process by the function: \( p_i = \text{SLOWDOWN}(s_i(t), p_i, p_j) \), which is only applied to the agent that adapts its path.

(V) Continuous control for tracking. The fifth building block is the continuous controller for an active agent to track its tentative path. We rely on the nonlinear control scheme from [13] for unicycle models that handles bounded control inputs and ensures the tracking of a reference trajectory with a provable bounded tracking error. The reference trajectory is constructed by the simple turn-and-forward controller. The guarantees for convergence and bounded tracking error are shown in Theorem 1 of [13]. For brevity, we denote it by the function: \((v_i(t), u_i(t)) = \text{MOVE}(s_i(t), p_i)\).

3) Graph Grammars: With the above building blocks, we now present the complete graph grammars:

\[ \text{[R0]} \]  

At \( t = 0 \), each agent \( i \in \mathcal{N} \) initializes its label by setting \( i.\text{id} = i, i.\text{mode} = \text{check or i.\text{mode} = static randomly, and i.data.nb = 0, i.data.pt = [-], i.gi = 0, where [-] denotes an empty sequence. Moreover, for any agent } j \in N_i(0) \), it sets \((i, i).\text{id} = (i, j)\).

After the system starts at \( t > 0 \), each agent \( i \in \mathcal{N} \) reconstructs its local graph \( G_i(t) \) and applies the rules below:

\[ \text{[R1]} \]  

If \( i.\text{mode} = \text{check} \), agent \( i \) first communicates with every neighbor \( j \in \mathcal{N}_i(t) \) and checks if \( j.\text{mode} = \text{move} \) and \( i.xdata.nb \neq 0 \). If so, it sets \( i.\text{mode} = \text{static} \) and adds agent \( j \) to \( i.xdata.nb \).

After that, if \( i.\text{mode} = \text{check} \) still holds, agent \( i \) chooses an allowed marked scheme \( N_i^m \) given \( G_i(t) \) and calls the function \( \text{CHECK}(s_i(t), N_i^m(t), z_i, N_i^m) \). If (4) has a solution as \( p_i \) and \( v_i \), \( \chi_i > 0 \), then it sets \( i.\text{mode} = \text{move} \) and \( i.xdata.nb = N_i^m(t), i.xdata.gi = \chi_i, i.xdata.pt = p_i \). Otherwise if \( \chi_i \leq 0 \), it sets \( i.\text{mode} = \text{static} \) and \( i.xdata.nb = 0 \). Otherwise if no solutions to (4) exist or \( \chi_i \leq 0 \), it sets \( i.\text{mode} = \text{static} \) and \( i.xdata.nb = 0 \).

\[ \text{[R2]} \]  

If \( i.\text{mode} = \text{static} \), agent \( i \) stays static. Then it communicates with \( j \in \mathcal{N}_i(t) \) and checks if \( j.\text{mode} = \text{move} \) or \( j \in i.xdata.nb \). If \( j.\text{mode} = \text{move} \) and \( j \notin i.xdata.nb \), it updates its \( i.xdata.nb \). Otherwise, for each \( j \in i.xdata.nb \), it checks whether \( i \in j.xdata.nb \). If not, it removes agent \( j \) from \( i.xdata.nb \). At last, it checks if \( i.xdata.nb = 0 \). If so, it sets \( i.\text{mode} = \text{check} \).

\[ \text{[R3]} \]  

If \( i.\text{mode} = \text{move} \), agent \( i \) first checks if \( j.\text{mode} = \text{static} \), \( \forall j \in i.xdata.nb \). If not, it stops moving by setting \( i.\text{mode} = \text{check} \) and \( i.xdata.nb = 0 \). Otherwise, it executes its tentative path \( p_i \) via the motion controller \((v_i, w_i) = \text{MOVE}(s_i(t), p_i)\). As discussed earlier, agent \( i \) may encounter other agents, e.g., \( j \in \mathcal{N}_i(t) \):

(i) if \( j.\text{mode} = \text{move} \), they exchange their gains and tentative paths. Then the agent with higher gain is given higher priority. Assume for now that \( i.xdata.gi < j.xdata.gi \). Then the agent with lower priority, i.e., agent \( i \), calls \( \text{COLLIDEPath}(p_i, p_j) \) by (5) to check if \( p_i \) and \( p_j \) will collide. If so, agent \( i \) calls \( \text{SLOWDOWN}(s_i(t), p_i, p_j) \). If it has a solution, agent \( i \) updates its path \( p_i \) by slowing down; otherwise, agent \( i \) stops moving by setting \( i.\text{mode} = \text{static} \) and \( i.xdata.nb = 0 \).

(ii) if \( j.\text{mode} = \text{static} \), agent \( i \) checks if it would collide with agent \( j \) given its current path \( p_i \). If so, it stops moving by setting \( i.\text{mode} = \text{check} \) and \( i.xdata.nb = 0 \).

\[ \text{[R4]} \]  

If \( i.\text{mode} = \text{move} \) and \( ||p_i(t) - z_i|| < c_i \), where \( c_i > 0 \) is the threshold from Definition 3, agent \( i \) has reached its goal point. Then agent \( i \) stops moving and resets \( i.\text{mode} = \text{static} \) and \( i.xdata.nb = 0 \).

B. Local Discrete Plan Synthesis

The previous section solves how each agent could move to its current goal point, while obeying the motion constraints. Here we tackle how each agent should choose and update
its goal point to fulfill its local task $\varphi_i$. The infinite discrete plan (denoted by $\tau_i$) that satisfies $\varphi_i$ is derived by the automaton-based model checking algorithm \cite{2,9}: $\tau_i = \pi_i \circ \pi_{i,1} \cdots \pi_{i,k_i-1} (\pi_{i,k_i} \pi_{i,k_i+1} \cdots \pi_{i,K_i})^\omega$, where $\pi_{i,k} = (z_{i,k}, a_{i,k}) \in \Pi_i$ where $z_{i,k} \in Z_i$ and $a_{i,k} \in \Sigma_i$, $\forall i = 0, 1, \cdots, K_i$ and $K_i > 0$ is the total length of the prefix and suffix. We refer the interested readers to \cite{9} for algorithms and implementation details. Given the local plans from all agents, we impose the assumption below:

**Assumption 1:** The plans $\{\tau_i, i \in N\}$ are feasible if $\gamma(t)$ is allowed by Definition 2 when $p(t)$ satisfies $p_i(t) = z_{i,k}$, $\forall i \in N$ and $\forall k = 0, 1, \cdots$.

**C. The Complete Solution**

When the system starts, each agent $i \in N$ derives its local plan $\tau_i$ and its current goal point $z_{i,0}$; then it follows the EGGs by [R-4] after it reaches $z_{i,0}$, it becomes static. Then it performs the action $a_{i,0}$ according to the plan $\tau_i$; after the action is accomplished, it remains static until all other agents have reached their respective goal points and finished the corresponding actions. It can be detected through the communication network that all agents are static. Then each agent updates its goal point by $z_{i,1} = z_{i,1}$ and sets $i.mode = \text{check}$, $\forall i \in N$. Then all agents follow the EGGs to make progress towards its new goal point. This procedure repeats indefinitely as the discrete plans have infinite length. Note that after agent $i \in N$ reaches $z_{i,K}$, it should set $z_{i,K} = z_{i,k}$, to repeat the plan suffix.

**Theorem 3:** All local tasks $\varphi_i$, $i \in N$ are satisfied while $\gamma(t) \in \Gamma_d$, $\forall t > 0$.

**Proof:** (Sketch) Since the workspace is to be unbounded and free of obstacles, at least one agent within $N$ can be active and make a progress towards its current goal point. The connectivity of $G(t)$ follows from Lemma 2 and the fact that two disconnected agents remain connected indirectly through the common marked agents. The collision avoidance is ensured by the formulation of (4) and the velocity adjusting algorithm. Moreover, Assumption 1 ensures that the intermediate configuration of all agents’ goals is feasible. At last, correctness of the discrete plan ensures that the local task $\varphi_i$ is satisfied, $\forall i \in N$.

V. SIMULATION AND EXPERIMENTAL STUDY

This section presents the simulation and experimental results of applying the proposed scheme to both simulated and physical multi-robot systems.

A. Workspace and Agent Description

The six robots are labeled $R_0, R_1, \cdots, R_5$ and each occupies a disk area of radius 0.05m. As shown in Figure 1, the communication range $d$ is uniformly set to 0.9m, while the safety distance $d$ is set to 0.15m. Moreover, their reference linear velocity is set to between 0.1m/s and 0.3m/s, under the maximal 0.4m/s. The angular velocity is set to between 0.4rad/s and 0.5rad/s, under the maximal 0.7rad/s.

The robots’ motion and action model along with their local task specifications are defined as follows: robots $R_0, R_1$ have the local task as surveillance. Robot $R_0$ has four points of interest at $(1.5, 1.5), (0.2, 1.5), (0, 0), (1.6, 0)$ with labels $\{r_1\}, \{r_2\}, \{r_3\}, \{r_4\}$ and action $a_0$ as “take photos”. Its local task is to surveil $r_1, r_2, r_3, r_4$ in any order, which can be specified as the LTL formula $\varphi_0 = \bigwedge_{i=1}^{\cdot} A (\bigvee r_i \land a_0)$. Robot $R_1$ has points of interest close to $R_0$’s and its local tasks is similar to $\varphi_0$. Robots $R_2, R_3$ have the local tasks for providing services. Robot $R_2$ has three points of interest at $(1.2, 0.4), (0.6, 0.6), (0.6, 0.9)$ with labels $\{p_1\}, \{p_2\}, \{p_3\}$ and action $a_1$ as “provide services”. Its local task is to provide services to $p_1, p_2, p_3$ in sequence, namely $\varphi_2 = F (p_1 \land a_1) \land F ((p_2 \land a_1) \land F ((p_3 \land a_1)))$. Robot $R_3$ has points of interest close to $R_3$’s and its task is similar to $\varphi_2$. At last, robots $R_4, R_5$ are responsible for transporting goods between goal points. Robot $R_4$ has three points of interest $(1.1, 1.0), (1.5, 1.5), (1.0, 1.0)$ with labels $\{b\}, \{s_1\}, \{s_2\}$ and actions $\{a_2, a_3\}$ as “load and unload goods”. Its local task is to transport goods “\text{H}” from storage $s_1$ to base $b$ and “\text{B}” from storage $s_2$ to base $b$, i.e., $\varphi_4 = F (s_1 \land a_2) \Rightarrow (s_1 \cup (b \land a_3))$. Robot $R_5$ has three points of interest close to $R_4$’s and its local task is similar to $\varphi_4$. Initially, the agents start from a line graph.

B. Simulation Results

After the system starts, each robot first synthesizes its discrete plan $\tau_i$ as described in Section IV-B. For instances, robot $R_0$’s discrete plan is to visit $r_1, r_2, r_3, r_4$ in sequence and perform action $a_0$ at each point, which is then repeated, while robot $R_4$’s plan is to load goods “\text{H}” at $g_1$ and unload
it at $b$, then load goods “B” at $g_2$ and unload it at $b$, in sequence and repeat. Then they follow the EGGs as described in Section IV-A.3. Most of the time there are three to four robots moving. Figures 1 show some snapshots of how $G(t)$ changes with time. After each robot reaches its current goal point, it performs the planned action. It waits until all other robots become static and then updates its goal point. This procedure continues indefinitely and we simulate the system until $t = 72.5s$ when they have reached the forth goal point. Figure 2 verifies that all motion constraints are fulfilled by showing the evolution of the maximal length of the shortest path between any two vertices within $G(t)$ (i.e., its diameter) and the evolution of the minimal distance between any two robots. The complete simulation video can be found in [22].

C. Experimental Results

We implement the proposed scheme on a team of five Khepera II robots (diameter $0.12m$) at the GRITS Lab of Georgia Tech, as shown in Figure 3. They are differential-driven wheeled robots that communicates wirelessly with the base station computer. Their position and orientation are tracked in real-time by the OptiTrack system. The message exchange among the robots, between the robots and OptiTrack system are handled by Robot Operating System (ROS). The robots’ points of interest are scattered within the $3m \times 3m$ workspace and designed to be feasible by Assumption 1. The communication radius and safety distance are set to $0.9m$ and $0.15m$. The navigation controller is tuned properly to ensure that the robot tracks the synthesized path under a given precision. We omit the task description here due to limited space, which is similar to the simulation case but no robot actions are modeled in this experiment.

We run the system for 11 minutes and the robots have reached the fourth goal point in their respective plans. The whole experiment is recorded by an overhead camera and communication links among the robots are projected onto the ground. Some snapshots of the experiments are shown in Figure 3, where the robots are heading for different goal points. The complete experiment video can be found in [23], where we also show the diameter of $G(t)$ and the minimal distance between any two robots during the experiment to verify that both continuous constraints are satisfied.

VI. CONCLUSION AND FUTURE WORK

We have presented a hybrid control scheme for multi-robot systems with local tasks, under collision avoidance and connectivity maintenance constraints. Our solution relies on embedded graph control grammars and imposes only local communication and interactions. Future work includes the consideration of static obstacles and dependent local tasks.

REFERENCES