The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

A Discrete Grid Abstraction for Formation Control in the Presence of Obstacles

Damjan Miklic, Stjepan Bogdan, Rafael Fierro, and Sanjin Nestic

Abstract— **In this paper we present a formation reconfiguration methodology designed for controlling groups of autonomous agents in environments densely populated with obstacles. Our approach is based on abstracting the group of agents by a discrete rectangular grid. Agent and obstacle positions are mapped onto the formation grid. Then, collision free formation transition trajectories are computed using discrete event scheduling techniques that have been well-established in the manufacturing systems domain. The main contribution of this paper is a unified formation control framework that explicitly takes obstacles into account. Using discrete event system analysis tools we show that our approach guarantees convergence to the desired formation while avoiding obstacles and inter-agent collisions.**

I. INTRODUCTION

The productivity of a group of units working together in a coordinated manner surpasses the sum of individual productivities when working on the same task alone. Employing a group of autonomous robots to perform certain tasks offers advantages that reach beyond greater speed of execution and robustness to unit failures. A coordinated team of robots can accomplish tasks that could not be handled by a single unit, such as large object manipulation, distributed exploration and sensing or even coordinated negotiation of obstacles.

The enhanced capabilities of multi-agent systems do not come without a price and the control problems related to such systems exhibit a new dimension of complexity. The fundamental issues of system stability, convergence to the desired state and robustness must be considered. High system dimensionality, complex interactions, inherent parallelism and uncertainties make analysis and control synthesis a challenging task that has been attracting significant attention in the research community for over a decade, [1] - [2]. A lot of research efforts have been focused on defining and analyzing system stability in the context of multi-agent systems [3], [4], [5]. Authors in [5] present a method of ensuring provable stability of decentralized switching systems that can be applied to a wide variety of simple control laws, such as nearest neighbor and target tracking rules. Of the above mentioned research results, only [4] explicitly takes into account obstacles in the environment.

In addition to the general problem of stability, several problems specific to multi agent systems have been formulated and analyzed, such as flocking, consensus, coverage and pattern formation (also referred to as formation control). In this work we consider formation control with obstacle and inter-agent collision avoidance. The variety of different approaches that have been proposed for addressing formation control can roughly be divided into two groups: navigation functions [6], [7] and graph-theoretical approaches such as [2], [8]. Modern navigation function approaches have the advantage of being decentralized and having provable stability properties. However, they tend to have a higher number of parameters that need to be hand-tuned for a specific application and can suffer from the local minima problem. The navigation function approach generally offers a framework for dealing with obstacles, but there are few results that take them into account explicitly. Graph-theoretical approaches offer another natural way of modeling interactions in multi-agent systems. For example, authors in [8] have developed a method for automatic generation of control programs for accurate achievement and maintaining of geometric formations in a deadlock-free manner using so-called "embedded graph grammars". However, they do not consider obstacles in the environment, nor do they deal with group motion planning.

In this paper, we propose a formation controller that guarantees collision-free convergence to the desired formation in the presence of obstacles in the environment and takes group navigation into account. We make use of the concept of abstraction [9] by introducing a discrete rectangular grid as a formation abstraction. This enables us to decouple the formation control problem and consider group motion and formation reconfiguration separately. Obstacles are mapped onto the formation grid and taken into account explicitly when planning transition trajectories. A big advantage of the proposed approach is that it provides us with a framework that explicitly considers the effect of obstacles on the formation transition system. This method guarantees collision-free transitions between formations, while ensuring that deviations from the desired formation are not greater

The work of D. Miklic at the University of New Mexico was supported by the U.S. Bureau of Educational and Cultural Affairs through the Fulbright Program.

The work of S. Bogdan is supported by the Ministry of Science, Education and Sports of the Republic of Croatia, grant #036-0363078-3016 "Task Planning & Scheduling in Robotic and Autonomous Systems".

The work of R. Fierro is supported by NSF grants ECCS CAREER #0811347, IIS #0812338, CNS #0709329, and by DOE University Research Program in Robotics (URPR).

When this work was done, D. Miklic was a visiting researcher at the Electrical and Computer Engineering Department, University of New Mexico, Albuquerque, NM 87131-0001, USA. (e-mail: damjan.miklic@fer.hr)

S. Bogdan and S.Nestic are with the Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia. (e-mail: stjepan.bogdan,sanjin.nestic@fer.hr)

R. Fierro is with the MARHES Lab, Electrical & Computer Engineering Department, University of New Mexico, Albuquerque, NM 87131-0001, USA (e-mail: rfierro@ece.unm.edu)

than absolutely necessary to avoid obstacles.

The paper is organized as follows. In Section II we introduce the specific formation problem that we are dealing with and state it in a mathematically formal way. In Section III we describe the grid-based formation abstraction and the discrete-event reconfiguration algorithm. A way to deal with obstacles within the formation grid is presented in Section IV. In Section V we present a formal analysis of the proposed algorithms. Some illustrative simulation results are presented in Section VI. Concluding remarks are given in Section VII.

II. PROBLEM STATEMENT

The general problem we are interested in solving can be summarized as follows. We would like to design a controller capable of driving a group of agents $\{a_n\}$ through an environment populated with obstacles, while maintaining a desired formation as closely as possible. For a group of N agents we define a formation by their positions on a $(2N+1) \times (2N+1)$ grid $\mathcal{G}$. By maintaining formation "as closely as possible", we mean that the only situation in which agents are allowed to leave their designated position within the formation is to avoid a collision with an obstacle or another agent. As soon as the designated position is not under threat of collision, the agent should reclaim it. To tackle the problem described above, we have split it in two subproblems. On the higher level, we deal with driving the center cell of $\mathcal{G}$ (x_g, y_g) through the environment, towards a goal point (x_q, y_q), without colliding with obstacles. On the formation control level, our goal is to maintain the desired agent positions on $\mathcal{G}$, switching to other formations only to avoid collisions with obstacles that appear on the grid as it is moving through the environment. In this paper, we focus on the second part of the problem which we write down more formally below.

From the point of view of the lower level formation controller, a formation of N agents is described by a binary $(2N+1)\times(2N+1)$ *formation matrix* $\mathbf{F}$. The entry $\mathbf{F}(i,j) = 1$ denotes that cell (i,j) is occupied by an agent. The state of each agent is described by its coordinates on the formation grid, $\mathbf{a}_n = [i\ j]^T$. Grid coordinates correspond to standard matrix notation i.e., relative to the upper-left corner of the grid. Similarly, we define a binary *obstacle matrix* $\mathbf{O}$ where $\mathbf{O}(i,j) = 1$ denotes an occupied cell.

Problem 1: Given initial and desired formation matrices $\mathbf{F}^{(s)}$ and $\mathbf{F}^{(t)}$ and an obstacle matrix $\mathbf{O}$, find a sequence of states $(\mathbf{a}_n(k))$ for each agent a_n, that will ensure a collision-free and deadlock-free transition between formations.

In the above problem statement, by *collision-free* we mean that at any time step k, a cell can be occupied only by one agent or $\mathbf{a}_m(k) \neq \mathbf{a}_n(k)\ \forall m,n = 1,\ldots,N$. Furthermore, agents may only occupy cells that are free of obstacles, $\mathbf{O}\left(\mathbf{a}_n(k)\right) = 0\ \forall n,k$. The deadlock-free requirement implies that the agents will achieve the desired formation in a finite number of steps.

When considering a potential solution to Problem 1, we assume that all agent actions are coordinated by a central controller which has full information of agent states and obstacle configuration. This assumption is in line with several potential application scenarios, like the UGV group coordination by an UAV [10]. On the other hand, the proposed coordination methodology can be decentralized by using inter-agent communication and a consensus algorithm like the one described in [11].

III. THE DISCRETE GRID FORMATION ABSTRACTION

As stated in the previous section, a formation of agents is described by their positions on a virtual discrete grid $\mathcal{G}$. From the point of view of the formation coordination algorithm, the multi-agent system is discretized in time and in space. Space discretization means that each agent can only occupy one cell at a particular time. Time discretization implies that state changes occur only at discrete time steps. Furthermore, agent motion within one time step is restricted to the four neighboring cells. We first consider the transition from an initial formation $\mathcal{F}^{(s)}$ to a desired formation $\mathcal{F}^{(t)}$ on an obstacle-free grid. In the next section, we show how this approach can be extended to take obstacles into account.

The formation transition algorithm can be broken down into the following steps:

1) Target assignment
2) Path allocation
3) Trajectory scheduling

In the first step, we assign a target position in the final formation to each individual agent. We then assign a path to each agent, consisting of the cells connecting its initial and final positions on the grid. Finally, we implement a trajectory scheduling policy to ensure a collision-free transition. In the remainder of the section, we explain each of these steps in more detail.

A. Target assignment and path allocation

The target assignment step consists of creating initial/final position pairs. It is performed in an optimal way, by minimizing the total distance between initial and final positions using the Kuhn-Munkres [12] algorithm. On an obstacle-free grid distances can be computed simply by using the $\mathcal{L}_1$ norm,

$$d(\mathbf{a},\mathbf{t}) = |i-k| + |j-l|\,, \tag{1}$$

where $\mathbf{a} = [i\ j]^T$ and $\mathbf{t} = [k\ l]^T$ are initial and target positions respectively. In the presence of obstacles, feasible paths from all initial positions to all targets need to be determined first and their lengths are used as distances.

Path allocation consists of allocating the shortest feasible path between each of the assigned initial/final position pairs. Because the assignment procedure does not necessarily have a unique solution, at this point we can perform *path balancing*. All of the assigned paths are compared and if agent a_n crosses any cell assigned to agent a_m and has a longer total path, their goal points are swapped. This operation does not change the total distance traveled by all agents. It is performed to ensure that agents never cross a goal cell that has already been occupied, as this would lead to deadlock. Once occupied, goal cells cannot be released by an agent, due

to the *no preemption* requirement of the trajectory scheduling algorithm. Balancing the paths also tends to reduce the total time required for a formation transition.

B. Trajectory scheduling

Once each agent has been allocated a path, a dispatching strategy must be implemented in order to ensure inter agent collision avoidance. To deal with this problem in a structured manner, we model the multi-agent system on the formation grid as a discrete event system and introduce a formal description using the matrix model described in [13]. Adopting the terminology from manufacturing control systems, we model grid cells as system *resources* and cell occupations as *jobs*. According to this formalism, the path $\mathcal{P}_n = (p_n(k)) = ((i(k), j(k)) \,|k = 1, \ldots, |\mathcal{P}_n|)$ assigned to agent a_n requires the resource set $\mathcal{R}^n = \left(r_{i(k)j(k)}\right)$ and represents the job sequence $\mathcal{J}^n = \left(v^n_{i(k)j(k)}\right)$. The complete job set for a formation transition consists of all the job sets assigned to individual agents, $\mathcal{J} = \left(\mathcal{J}^1, \ldots, \mathcal{J}^N\right)$. The complete resource set is a union of all the resources required by all the agents, $\mathcal{R} = \bigcup_{n=1}^{N} \mathcal{R}^n \subset \mathcal{G}$. As more than one agent might require a particular resource (cell) during formation change, $|\mathcal{R}| \leq |\mathcal{J}|$. Rules describing formation transition behavior can be written as follows:

$$\begin{aligned} &\text{IF cell } (i(k-1), j(k-1)) \text{ is occupied} \\ &\text{AND cell } (i(k), j(k)) \text{ is free AND } u^d_k = 1 \\ &\text{THEN move to } (i(k), j(k)) \\ &\quad \text{AND release } (i(k-1), j(k-1)), \end{aligned} \tag{2}$$

where $k = 2, \ldots, |\mathcal{P}_n|$. Each agent's path contributes $|\mathcal{P}_n|-1$ transition rules.

Looking at the rules (2) we can see that the preconditions part of the IF-clause (between `IF` and `THEN`) consists of one job that needs to be completed and one resource that needs to be available in order for the next job to be started. We assign a boolean variable $x_i(k)$ to the *precondition* part of every formation transition rule, $x_i(k) = 1$ *iff* the precondition is satisfied at time step k. Then the *logical state vector* is defined as the column vector $\mathbf{x}(k) = [x_i(k)]\,,\ i = 1, \ldots, \sum_{n=1}^{N} (|\mathcal{P}_n| - 1)$.

The *resource idle vector* $\mathbf{r}^c(k) = [r^c_i(k)]\,,\ i = 1, \ldots, |\mathcal{R}|$ has one boolean element for every resource in the resource set $\mathcal{R}$, and $r^c_i(k) = 1$ *iff* the i-th resource r_i is idle (the corresponding cell is not occupied) at time step k and $r^c_i(k) = 0$ otherwise. Similarly, the *job completed vector* $\mathbf{v}^c(k) = [v^c_i(k)]\,,\ i = 1, \ldots, |\mathcal{J}|$ has one boolean element for every job in the job set $\mathcal{J}$, and $v^c_i(k) = 1$ *iff* the i-th job v_i has just been completed (the corresponding cell is occupied) at time step k and $v^c_i(k) = 0$ otherwise. These two vectors constitute the *system state vector* $\mathbf{m}(k) = \left[\left(\mathbf{v}^c(k)\right)^T \left(\mathbf{r}^c(k)\right)^T\right]^T$. Any given value of the system state vector uniquely determines an agent formation on the formation grid. Occupied grid cells correspond to $v^c_i = 1$ entries of the job completed vector.

Adopting the above conventions, we can derive the matrix model of the multi-agent formation transition system. The matrix model consists of four system matrices. The *job-sequencing matrix* $\mathbf{F}_v$ and *job-start matrix* $\mathbf{S}_v$ relate the logical state vector $\mathbf{x}$ to the job set $\mathcal{J}$. The *resource requirements matrix* $\mathbf{F}_r$ and the *resource release matrix* $\mathbf{S}_r$ relate the logical state vector to the resource set $\mathcal{R}$. The details of the procedure for deriving system matrices given $\mathcal{J}$, $\mathcal{R}$ and $\mathbf{x}$ are described in [13]. The only difference specific to our formation control problem is the fact that the *input* and *output matrix* are empty, as we are not considering the scenario of agents entering or leaving the formation at this point. We can write down the system matrices in a more compact form as the *activity completion matrix* $\mathbf{F} = [\mathbf{F}_v\ \mathbf{F}_r]$ and the *activity start matrix* $\mathbf{S} = \left[\mathbf{S}^T_v\ \mathbf{S}^T_r\right]^T$.

Matrices $\mathbf{F}$ and $\mathbf{S}$ provide a model of the formation transition system. In this model a *conflict* situation can occur when two agents are requiring the same cell for their next move. In order to prevent inter-agent collisions, we need to implement a conflict resolution strategy. We associate a boolean control variable u^d_k with every potentially conflicting rule, as shown in (2). The vector $\mathbf{u}^d = \left[u^d_k\right]$ containing all control variables is called the *dispatching vector*. Our conflict resolution strategy can be completely described by two binary matrices. The *dispatching matrix* $\mathbf{F}_d$ defines how the dispatching vector affects the transition rules. The *dispatching vector release matrix* $\mathbf{S}_d$ defines how the dispatching vector is updated based on the logical system state.

For our formation transition problem, we are using the so-called *p-invariant* control structure with a *last buffer first served* (LBFS) strategy. This is an approach that has been well-established in the area of manufacturing systems control [14]. In our case, this means we are ensuring that an agent is allowed to enter a shared grid cell only after all agents with longer remaining paths have already passed through that cell. In order to be able to determine path distances from a shared cell to all target cells, we introduce the *rule distance matrix* $\mathbf{D}_x$ where $\mathbf{D}_x(i, j)$ represents the "distance" from rule x_j to rule x_i. If the rules are part of the same path and $x_i \geq x_j$, then their distance is equal to the number of rules between them. Otherwise, their distance is defined as -1. Algorithm 1 outlines how matrices $\mathbf{S}_d$, $\mathbf{F}_d$ and the initial value of the dispatching vector $\mathbf{u}_d(0)$ can be computed from $\mathbf{F}_r$ and path length information.

Now we can write down the rules for updating the logical and system state vectors at each time step k as follows:

$$\overline{\mathbf{x}}(k) = \mathbf{F} \vartriangle \overline{\mathbf{m}}(k-1) \nabla \mathbf{F}_d \vartriangle \overline{\mathbf{u}}_d(k-1) \tag{3}$$

$$\mathbf{m}(k) = \mathbf{m}(k-1) + \left[\mathbf{S} - \mathbf{F}^T\right] \mathbf{x}(k) \tag{4}$$

$$\mathbf{u}_d(k) = \mathbf{u}_d(k-1) + \left[\mathbf{S}_d - \mathbf{F}^T_d\right] \mathbf{x}(k). \tag{5}$$

The symbols ∇ and $\vartriangle$ denote matrix addition and multiplication respectively, in *and/or* algebra. These operations are performed like standard matrix operations, but replacing element-wise multiplication by logical `AND` and addition by `OR`. The overline symbol denotes element-wise logical negation. We can determine the agent formation at time step k by reading cell occupations directly from the system state

Algorithm 1: ControlMatrices

Input: $\mathbf{F}_r$, plen $=\{|\mathbf{P}_n| : n = 1, \ldots, N\}$
Output: $\mathbf{F}_d$, $\mathbf{S}_d$, $\mathbf{u}_d(0)$

Compute the rule distance matrix $\mathbf{D}_x$ from path length vector plen
foreach *column* $\mathbf{f}_j$ *of* $\mathbf{F}_r$ **where** $\sum_i f_{ij} > 1$ **do**
    Create dispatching matrix component $\mathbf{F}_{d,j}$ from $\mathbf{f}_j$
    foreach *column* $\mathbf{d}_k$ *of* $\mathbf{D}_x \cdot \mathbf{F}_{d,j}$ **do**
        Find distance from $u_d(k)$ to last rule on the path, $d_u^x(k)$=max$(\mathbf{d}_k)$
    end
    Set $\mathbf{u}_{d0,j} = 1$ where max$(\mathbf{d}_u^x)$ and 0 otherwise
    Sort $\mathbf{d}_u^x$ descending and permute columns of $\mathbf{u}_{d0,j}$ and $\mathbf{F}_{d,j}$ accordingly
    Compute $\mathbf{S}_{d,j}$ by shifting the last row of $\mathbf{F}_{d,j}^T$ to the top
end
Concatenate $\mathbf{F}_d$:=$[\mathbf{F}_{d,j}]$, $\mathbf{S}_d$:=$\left[\mathbf{S}_{d,j}^T\right]^T$, $\mathbf{u}_{d0}$:=$\left[\mathbf{u}_{d0,j}^T\right]^T$

vector $\mathbf{m}$.

IV. FORMATION TRANSITIONS IN THE PRESENCE OF OBSTACLES

We take obstacles into account in our grid-based formation abstraction by representing them as occupied grid cells. As the group of agents is moving through an obstacle populated environment, from the point of view of the grid abstraction obstacles are sweeping through the formation grid. Obstacle motion is discretized both in space with grid cell size w and in time with time step l. We are assuming that within one time step an obstacle-occupied cell cannot move further than to one of its eight neighboring cells. Furthermore, we are assuming that we can predict obstacle motion at least one time step in advance i.e., at time step l we know the obstacle configurations $\mathbf{O}(l)$ and $\mathbf{O}(l+1)$. Finally, in order to be able to avoid obstacles while maintaining the formation grid, we assume that each agent can traverse at least $N+1$ cells within one obstacle time step l.

Given $\mathbf{O}(l)$ and $\mathbf{O}(l+1)$, the current and next obstacle configuration matrix respectively, our evasion strategy consists of the following steps:

1) Determine the movement direction for each obstacle
2) Generate new formation target positions in the obstacle-free space
3) Find collision-free paths to the new target positions
4) Apply the formation transition strategy described in the previous section to avoid collisions

In the rest of the section we explain each of the above steps and give a simple illustrative example.

A. Obstacle segmentation and movement direction

In order to enable efficient obstacle avoidance, we need to determine the movement direction of the obstacles. As matrices $\mathbf{O}(l)$ and $\mathbf{O}(l+1)$ can contain several independent obstacles, we must first perform *obstacle segmentation* and consider each obstacle separately. The segmentation result is a set of P_l isolated regions $\mathcal{O}_{seg}(l) = \{\mathbf{O}_p(l) | p = 1, \ldots, P_l\}$ each represented by its own binary $(2N+1) \times (2N+1)$ matrix $\mathbf{O}_p(l) = [o_{ij,p}]$. For each obstacle, we compute its "center of mass" coordinates using the equations

$$i_{c,p} = \frac{\sum_{i,j} i \cdot o_{ij,p}}{\sum_{i,j} o_{ij,p}},\ j_{c,p} = \frac{\sum_{i,j} j \cdot o_{ij,p}}{\sum_{i,j} o_{ij,p}}, \tag{6}$$

where $i, j = 1, \ldots, (2N+1)$. Having computed the centers of mass for each obstacle in $\mathbf{O}(l)$and $\mathbf{O}(l+1)$ respectively, we can perform obstacle matching and compute movement gradients for each obstacle. The number of obstacles P_l and P_{l+1} does not necessarily have to be the same, as obstacles can appear and disappear from cells on grid edges within one time step. To match obstacles, we use the Euclidean distance between their centers of mass (in integer grid coordinates)

$$d_{p,q} = \sqrt{(i_{c,p}(l) - i_{c,q}(l+1))^2 + (j_{c,p}(l) - j_{c,q}(l+1))^2}. \tag{7}$$

If the distance is smaller than a predefined threshold ε, $d_{p,q} < \varepsilon$, we consider $\mathbf{O}_q(l+1)$ to be a match for $\mathbf{O}_p(l)$. We are using a threshold value of $\varepsilon = 1.5$, because the maximal expected obstacle movement is one diagonal grid cell.

To obtain information on obstacle motion within the current time step we compute the quantized gradients of the mass center along the rows and columns,

$$\begin{aligned} \triangle\ i &= \mathrm{sgn}(i_{c,p}(l+1) - i_{c,p}(l)) \\ \triangle\ j &= \mathrm{sgn}(j_{c,p}(l+1) - j_{c,p}(l)). \end{aligned} \tag{8}$$

For new obstacles, $\mathcal{O}_{seg}(l+1) \setminus \mathcal{O}_{seg}(l)$ we assume diagonal motion i.e., $\triangle\ i = \pm 1$ and $\triangle\ j = \pm 1$. We are not interested in deducing the direction of motion for obstacles leaving the grid, $\mathcal{O}_{seg}(l) \setminus \mathcal{O}_{seg}(l+1)$, as it does not affect the avoidance strategy.

B. Obstacle-free reference positions and path planning

Once we have determined directions of motion for all obstacles, we can generate new reference positions for agents whose current positions are in danger of colliding with oncoming obstacles. Algorithm 2 outlines the procedure for generating the evasion formation $\mathbf{F}_e$. The algorithm finds evasion positions by searching in all possible directions of motion, preferring those lying perpendicular to obstacle motion and towards grid center. The function Match performs obstacle matching and gradient computation as described previously. The function FindFree searches for the first obstacle-free cell lying perpendicular to obstacle motion. The search is performed first towards the center of the grid, then towards the nearest edge, and finally towards the far edge of the grid.

After computing the evasion formation $\mathbf{F}_e$, we must find feasible paths between agent positions in $\mathbf{F}$ and $\mathbf{F}_e$, taking into account the obstacle set $\mathbf{O}(l)$. To do this, we construct a connectedness graph of the formation grid, where nodes represent obstacle-free cells and edges connect neighboring cells. We then use BFS (Breadth-first search) to find shortest

Algorithm 2: EvasionFormation

Input: $\mathbf{F}$,$\mathbf{O}(l)$,$\mathbf{O}(l+1)$
Output: $\mathbf{F}_e$

$\mathbf{F}_e$:= $\mathbf{F}$
$\mathbf{O}_{all}$:= $\mathbf{O}(l)$ & $\mathbf{O}(l+1)$
$\mathcal{O}_{seg}(l)$:= `Segmentation(`$\mathbf{O}(l)$`)`
$\mathcal{O}_{seg}(l+1)$:= `Segmentation(`$\mathbf{O}(l+1)$`)`
$\{\mathcal{O}_{seg}(l), \mathcal{O}_{seg}(l+1), \triangle\ \mathbf{i}, \triangle\ \mathbf{j}\}$:= `Match(`$\mathcal{O}_{seg}(l)$, $\mathcal{O}_{seg}(l+1)$`)`
foreach $\mathbf{O}_p$ **in** $\mathcal{O}_{seg}(l+1)$ **do**
    $\mathbf{F}_c$:= $\mathbf{F}$ & $(\mathbf{O}_p(l+1) \mid \mathbf{O}_p(l))$
    foreach $f_{c,ij} = 1$ **do**
        $\mathbf{F}_e(i,j)$:= 0
        (i,j) := `FindFree(`i,j,$\mathbf{O}_{all}$,$\triangle\ i_p$, $\triangle\ j_p$, $\mathbf{F}_e$`)`
        $\mathbf{F}_e(i,j)$:= 1
    end
end

feasible paths from every starting position in $\mathbf{F}$ to every target position in $\mathbf{F}_e$. Path lengths are used as inputs to the target assignment procedure described in the previous section. From this point onward, the formation reconfiguration procedure is the same as described in Section III except that BFS shortest feasible paths are used instead of $\mathcal{L}_1$ shortest paths.

V. CONVERGENCE ANALYSIS

We now turn our attention to the convergence properties of the described formation control algorithm. We want to show that for arbitrary initial and target formations, the transition will always be feasible i.e., the agents will reach the target formation in a finite number of steps.

A. Path allocation properties

We first establish the properties of the allocated transition paths, as these are fundamental to the convergence of the scheduling algorithm.

Lemma 1: Any path $\mathcal{P}_n$ generated by the algorithm described in Section III has the property

$$p_n(k) \neq p_n(j) \ if\ k \neq j. \tag{9}$$

In other words, each cell is assigned at most once to an agent's path, there is no backtracking or loops.

Proof: Up to the path balancing step, property (9) is obvious for both the free-space and the obstacle case. In the free-space case, we assign to each agent the direct shortest path to its target, which surely does not visit any cell twice. Similarly, in the case with obstacles, we find the shortest feasible path using the BFS algorithm which again does not contain duplicate cells. We can show by contradiction that this property is preserved after path balancing. We first note that due to the Kuhn-Munkres assignment the total path length $|\mathcal{P}| = \sum_{n=1}^{N} |\mathcal{P}_n|$ is minimal for the given formation matrices. We also note that path balancing does not change the total path length. Let us now assume that after the balancing step (9) is not true i.e., $\exists\ k \neq j$ such that $p_n(k) = p_n(j)$. In this case, we can write the path down as

$$\begin{aligned}\mathcal{P}_n = &(p_n(1), \ldots, p_n(k-1), p_n(k), p_n(k+1), \ldots,) \\ &\cup (p_n(j-1), p_n(k), p_n(j+1), \ldots, p_n(|\mathcal{P}_n|)).\end{aligned}$$

However this implies that there exists a shorter path $\mathcal{P}'_n$ between $p_n(1)$ and $p_n(|\mathcal{P}_n|)$ obtained by removing all the cells between the first occurrence of $p_n(k)$ and $p_n(j+1)$ from $\mathcal{P}_n$. Then by simply replacing $\mathcal{P}_n$with $\mathcal{P}'_n$ in $\mathcal{P}$ we would obtain a set of paths with shorter total length, which is in contradiction with the minimal path requirement. ■

Lemma 2: Given two paths $\mathcal{P}_n$ and $\mathcal{P}_m$ obtained by the proposed algorithm, such that $p_n(k_m) = p_m(j_n)$ then $p_n(k > k_m) \neq p_m(j < j_n)$ and conversely $p_m(j > j_n) \neq p_n(k < k_m)$.
In other words, there are no paths that form a "head-on" situation.

Proof: Again, by contradiction, suppose there exist

$$\begin{aligned}\mathcal{P}_n &= (\ldots, p_n(k-1), p_n(k), p_n(k+1), \ldots), \\ \mathcal{P}_m &= (\ldots, p_m(j-1), p_m(j), p_m(j+1), \ldots)\end{aligned}$$

such that $p_n(k) = p_m(j)$ and $p_n(k-1) = p_m(j+1)$. Then, by swapping paths starting from $p_n(k)$ we would get the path

$$\mathcal{P}_n = (\ldots, p_m(j+1), p_n(k), p_n(j+1), \ldots)$$

passing through the same cell twice, which is in contradiction with Lemma 1. ■

B. Trajectory scheduling properties

We can now state our main convergence theorem which guarantees that every feasible formation transition will complete in finite time.

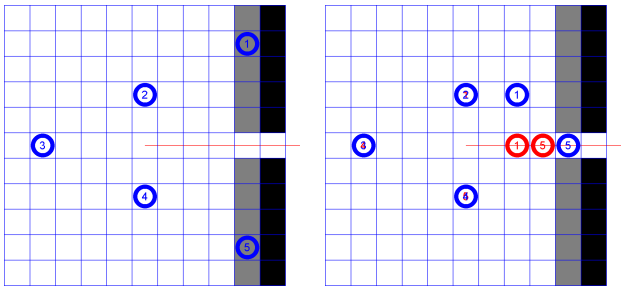
Theorem 1: Given the formation control strategy described in Sections III and IV, a transition between arbitrary formations will be completed in a finite number of steps, provided there exist at least one feasible (obstacle-free) path between all initial and target position pairs.

Proof: At this point, we just provide a sketch of the proof, as the full proof requires a significant amount of background in discrete-event system theory and would not fit within the format of the paper. Due to the no preemption requirement of the matrix-based transition controller, deadlock could potentially occur if $p_m(k) = p_n(|\mathcal{P}_n|)$ ($p_m(k)$ is a_n's goal point) and a_n reaches it before a_m has passed through it. It follows from Lemmas 1 and 2 that this is in fact the only situation when deadlock can occur. However, this is exactly the situation that our p-invariant LBFS control strategy prevents, because it ensures that an agent can occupy a shared cell only after all agents with longer remaining paths have already passed through it. ■

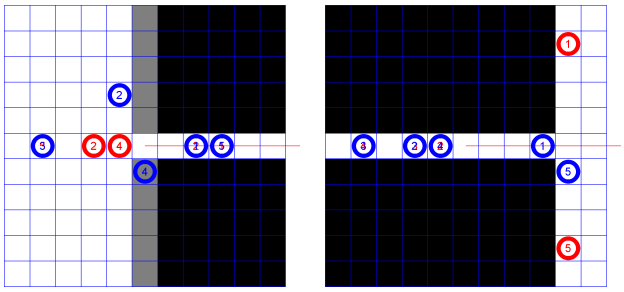
VI. SIMULATION RESULTS

In order to illustrate the described formation control algorithm, we present two examples that are encountered in group agent motion. In the first example, shown in Fig. 1, a group

of five agents is passing through a narrow corridor. In the second example shown in Fig. 2, an obstacle is approaching the agent group diagonally from the left. This corresponds to a situation where the agent group is rotating clockwise (group heading is denoted in the figures by the solid red line). In both examples, the agents are initially forming a V-shaped formation and required to restore this formation as soon as the obstacle configuration allows it. Current obstacle position $\mathbf{O}(l)$ is denoted by black squares while gray squares denote the predicted position $\mathbf{O}(l+1)$.

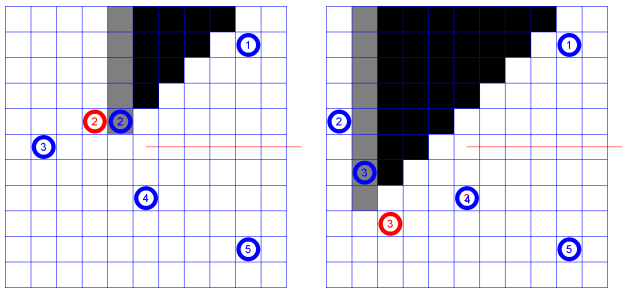


(a) $l = 1,\ k = 0$ (b) $l = 2,\ k = 5$

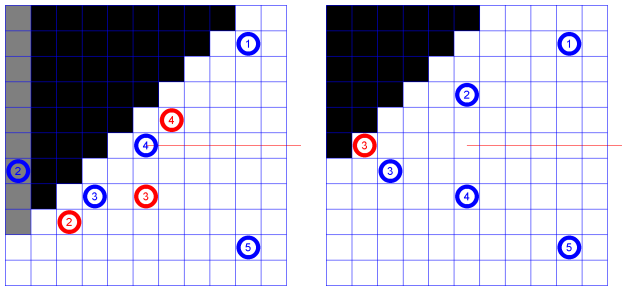


(c) $l = 6,\ k = 2$ (d) $l = 15,\ k = 4$

Fig. 1: Five agents in V-formation, squeezing through a narrow corridor.



(a) $l = 5,\ k = 1$ (b) $l = 8,\ k = 2$



(c) $l = 9,\ k = 3$ (d) $l = 13,\ k = 1$

Fig. 2: Five agents in V-formation, evading a diagonally moving obstacle.

VII. CONCLUSIONS AND FUTURE WORK

We have presented a formation control framework based on a discrete grid abstraction of the multi agent system. Using discrete event system scheduling techniques, our controller generates collision-free formation transition trajectories in the presence of obstacles. Analytical results and simulations show that transitions between arbitrary formations occur in a collision-free and deadlock-free manner. A significant advantage of our approach is the fact that it treats formation specifications and obstacles within an unified framework. Currently, we are implemented the grid-based formation algorithm described herein on a multi-vehicle experimental testbed.

REFERENCES

[1] T. Balch and R. Arkin, "Behavior-based formation control for multi-robot teams," *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 6, pp. 926–939, Dec 1998.

[2] J. Hendrickx, B. Fidan, C. Yu, B. Anderson, and V. Blondel, "Formation reorganization by primitive operations on directed graphs," *IEEE Transactions on Automatic Control*, vol. 53, no. 4, pp. 968–979, May 2008.

[3] H. Tanner, G. Pappas, and V. Kumar, "Leader-to-formation stability," *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 3, pp. 443–455, June 2004.

[4] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *Automatic Control, IEEE Transactions on*, vol. 51, no. 3, pp. 401–420, March 2006.

[5] B. Shucker, T. Murphey, and J. Bennett, "Switching rules for decentralized control with simple control laws," *American Control Conference, 2007. ACC '07*, pp. 1485–1492, July 2007.

[6] H. Tanner and A. Kumar, "Formation Stabilization of Multiple Agents Using Decentralized Navigation Functions," in *Robotics: Science And Systems I*. MIT Press, 2005, p. 49.

[7] D. Dimarogonas, S. Loizou, K. Kyriakopoulos, and M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-point agents," *Automatica*, vol. 42, no. 2, pp. 229–243, Feb. 2006.

[8] B. Smith, M. Egerstedt, and A. Howard, "Automatic deployment and formation control of decentralized multi-agent networks," *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.*, pp. 134–139, May 2008.

[9] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 5, pp. 865–875, Oct. 2004.

[10] N. Michael, J. Fink, and V. Kumar, "Controlling ensembles of robots via a supervisory aerial robot," *Advanced Robotics*, vol. 22, no. 12, pp. 1361–1377, 2008.

[11] N. Michael, M. Zavlanos, V. Kumar, and G. Pappas, "Distributed multi-robot task assignment and formation control," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 128–133, May 2008.

[12] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. [Online]. Available: http://link.aip.org/link/?SMM/5/32/1

[13] S. Bogdan, F. L. Lewis, Z. Kovacic, and J. José Mireles, *Manufacturing Systems Control Design: A Matrix-based Approach (Advances in Industrial Control)*. New York: Springer-Verlag, 2006.

[14] A. Gurel, S. Bogdan, and F. L. Lewis, "Matrix approach to deadlock-free dispatching in multi-class finite buffer flowlines," *IEEE Transactions on Automatic Control*, vol. 45, no. 11, pp. 2086–2090, 2000.