Formation Control of a Large Group of UAVs with Safe Path Planning and Obstacle Avoidance

Gergely Regula
MTA – BME Control Engineering Research Group
H-1111 Budapest, Kende u. 13 – 17., Hungary
Email: gergely.regula@sztaki.mta.hu

Béla Lantos
Research Centre of Vehicle Industry, Széchenyi István University
H-9026 Győr, Egyetem tér 1., Hungary
Email: lantos@iit.bme.hu

***Abstract*—In this article we propose a hierarchical control structure for multi-agent systems. The main objective is to perform formation change manoeuvres, with guaranteed safe distance between each two vehicles throughout the whole mission, even if motion is restricted to a limited space and static obstacles are present. The key components that ensure safety are a robust control algorithm that is capable of stabilising the group of vehicles in a desired formation and a higher level path generation method that provides all the vehicles with safe paths, based on graph theoretic considerations. The method can efficiently handle a large group of any type of vehicles. As an illustration, the results are applied to a group of 18 quadrotor UAVs, where 2 of the UAVs cannot receive information from the others.**

***Index Terms*—multi-agent system, formation control, distributed control, robust control, obstacle avoidance, UAVs, quadrotor helicopters**

I. Introduction

Increasing attention has been focused on the problem of controlling large scale systems that are built up from several smaller subsystems, e.g. a group of UAVs. The problem is fundamentally different from single vehicle control since the stability of single vehicles does not guarantee the stability at the group level. Moreover, controlling a group of vehicles together can result in better overall performance and certain tasks can also be performed more effectively. Examples to such cases are surveillance missions, fuel consumption reduction by travelling in formation.

Advances in communication technology, miniaturisation and increased computation power open the way to implement not only local, but also formation level control algorithms on board of a single vehicle. Performing all the required calculations in a centralised manner is often not viable. In such cases, distributed solutions are required, even though additional problems arise, e.g. communication errors or delays.

Several methods have been elaborated that solve certain problems related to multi-vehicle systems. Each of them have their strengths and weaknesses, thus they have evolved in parallel. Two of the most frequently applied control methods are the model predictive control (MPC) and robust control techniques.

Obstacle and collision avoidance are most often solved by applying MPC methods [1]–[4]. MPC involves numerical optimisation (occasionally mixed integer programming) at every single time instant and it is a flexible framework, various objectives can be included into the problem formulation. The cost is the increased computational complexity that may require more computational power than what currently exists.

Virtual potential field (VPF) methods can also be applied to problems mentioned above [5], [6]. These methods are mostly applied in single vehicle problems. A VPF is included in the system dynamics such that obstacles and target locations exert repulsive and attractive forces on the vehicles, respectively, thus guiding them towards their targets. Most commonly, these forces are inverse proportional to the squred distances between the corresponding objects or points.

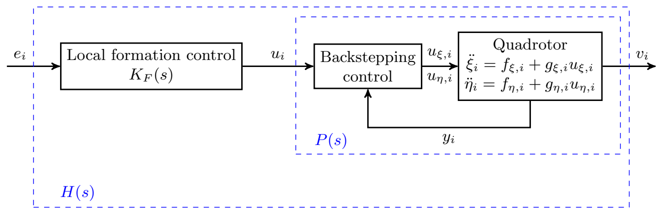
Other approaches include robust control methods [7]–[11] that can guarantee certain types of robustness and performance but cannot handle hard constraints the way MPC can. This is the motivation of the method we propose in the following. A promising formation stabilising algorithm is presented in [11], which ensures that vehicles reach a desired formation, even if the communication topology changes almost arbitrarily and arbitrarily quickly. It utilises the graph theoretical results of [12]. However, it neither guarantees collision-free motion for the vehicles, nor is capable of avoiding obstacles. We extend this approach by a higher level method effectively which tackles both problems mentioned above, even for a relatively large group of vehicles.

The article is structured as follows. Preliminary results are briefly summarised in Section II, which include the previous results of the authors and present the method, the capabilities of which is extended by our new method. The main contribution of the article, i.e. the safe path generating algorithm is presented in Section III, which is followed by two practical examples in Section IV. The article ends with a short conclusion and summary of the results.

II. Preliminary Results

The relation between formation stability of connected linear systems and graph-theory was discussed in the pioneering work of Fax and Murray [12]. They revealed that the stability of a formation of a group of identical systems is closely related to the eigenvalues of the normalised Laplacian associated to the communication topology graph of the group.

Based on their work, Popov and Werner presented a control design method in [11] that extends this analysis framework. They incorporate communication topology and its change



e_i
Local formation control $K_F(s)$
u_i
Backstepping control
$u_{\xi,i}$
$u_{\eta,i}$
Quadrotor
$\ddot{\xi}_i = f_{\xi,i} + g_{\xi,i}u_{\xi,i}$
$\ddot{\eta}_i = f_{\eta,i} + g_{\eta,i}u_{\eta,i}$
v_i
y_i
$P(s)$
$H(s)$

Figure 1. Single quadrotor with local controllers.

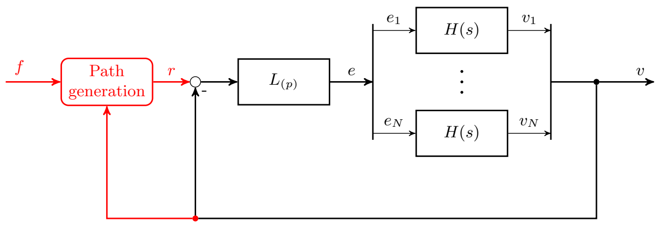


Figure 2. Formation change logic.

as a disturbance into the control design. Thus, by the aid of well-known robust control design techniques, formation controllers can be obtained locally. The design is robust against communication topology changes and is independent of the number of the vehicles forming the group.

This robust formation control method is suited to our control system applied to quadrotor helicopters, which is presented in detail in [13]. It can also be extended to include model uncertainties in the future. Our method is a backstepping control algorithm that stabilises the nonlinear dynamics of the quadrotor in a specified 3D position and yaw angle. Thanks to the backstepping control's linearising and decoupling effect, the closed loop system can be treated as four separate linear systems.

A quadrotor with its local controllers is depicted in Fig. 1. The notations follow the conventions of the previous works. The input and output of the formation-level controller $K_F(s)$ are the weighted formation error and the reference path of the helicopter, respectively, while y_i contains the measurements required by the backstepping control. The signals in v_i consist of the position coordinates and the yaw angle of the i-th helicopter.

III. Safe Formation Change

The most crucial strengths of the algorithms in the preliminaries are that they are capable of stabilising a group of any number of vehicles with almost any kind of communication topology that holds certain connectivity properties. However, there is a major drawback that is not explicitly tackled by the algorithm, i.e. it is not guaranteed that the vehicles keep safe distance from each other during the transients. Linear robust control methods cannot satisfy such constraints. Therefore, either different control algorithms are required for such problems, such as model predictive control (MPC), or collision avoidance must be implemented on a higher level.

The proposed method follows the latter approach and is the main contribution of the paper. Given a number of identical vehicles in an initial formation (defined by spatial points $S_i \in \mathbb{R}^3$), the task is to occupy the specified target positions T_j within finite time and keeping a predefined minimum distance d_s between each other during the transition. Vehicles are assigned a target position dynamically during the path generation. For simplicity reasons, the vehicles track straight paths between the start and target positions and may not necessarily move all at the same time since one might act as an obstacle to the other, depending on the structure of the initial and target formation. The algorithm also takes into account that the vehicles have a maximum travelling speed. There is only one restriction, which is related to the formation and the predefined safety distance. The ratio between the minimum distance between each pair of vehicles in their initial and target positions and the safety distance should exceed a constant value specified later:

$$\min_{\substack{i,j\\i\neq j}} \frac{d_{S,ij}}{d_s} > c \qquad \min_{\substack{i,j\\i\neq j}} \frac{d_{T,ij}}{d_s} > c, \tag{1}$$

where $d_{\bullet,ij} = \|\bullet_i - \bullet_j\|$ and d_s is the safety distance. The crucial aim is to find the smallest possible c. As it will be revealed later, the above constraint is not overly restrictive in real applications since the safety distance is related to the physical dimensions of the vehicles.

In the following, the safe path generating method will be presented, then as an illustration, a formation changing scenario will be shown.

A. Path Generating Algorithm

The basic idea of the proposed algorithm is to avoid online path planning and optimisation at every sample time instant. Instead, trajectories will be generated in a simple but efficient way only if the formation of the vehicle group has to be changed. The generated paths will be safe at the same time. Throughout the paper, safety region of a point or a route have the following meaning.

Definition 1 (Safety region): The safety region of a spatial point P is the set points for which the following condition holds:

$$R_{P,d_s} = \left\{ Q \in \mathbb{R}^3 \;\middle|\; \|P - Q\| \le d_s \right\}, \tag{2}$$

where d_s is the safety distance. Safety region can be defined for a line segment $\overline{ST}$ similarly:

$$R_{\overline{ST},d_s} = \left\{ Q \in \mathbb{R}^3 \;\middle|\; d(\overline{ST}, Q) \le d_s \right\}. \tag{3}$$

The formation change logic is integrated into the control logic as shown in Fig. 2, while the steps of the method are described in Tab. I. In Fig. 2, $L_p = L \otimes I_p$ is a time varying matrix is a time varying matrix describing the communication interconnection and $H(s)$ represents a vehicle with all its on-board control (see Fig. 1). Here, $\otimes$ stands for the Kronecker product, $L = [L_{ik}]$ is the $N \times N$ normalized Laplacian matrix of the communication graph and p is the dimension of v_i [12].

Table I
ALGORITM OVERVIEW.

Phase 1 **Direct transition**

Yes
- 1.1) Selecting candidate paths
- 1.2) Conflict search in "dual" graph
- 1.3) Maximum clique or single route search
- 1.4) Found new route?

No

Phase 2 **Correction routes**

Yes
- 2.1) Selecting candidate correction routes
- 2.2) Conflict search in "dual" graph
- 2.3) Maximum clique or single correction route search
- 2.4) Found new correction route?

No

Phase 3 **Resolving trapped targets**
- 3.1) Checking all routes for conflicts
- 3.2) Maximum clique search

Finished

The first two phases may consist of several steps. During phase 1, as many vehicles as possible move directly from their initial positions to certain target positions. The steps are repeated as long as new routes are found, otherwise we proceed to the next phase. In phase 2, certain vehicles that have already reached a target regroup so that empty targets (target points which are not occupied by any vehicle) are generated in the proximity of new vehicles. The condition for advancing to the final phase is similar to that in the previous case. In the last phase, vehicles that still remain in their initial positions can simultaneously move to a target.

The key in each phase is how to determine which vehicles are allowed to move at the same time. Graphs will be constructed that contain information about the risk of collision. The number of vehicles taking part in each step will correspond to the size of a clique in these graphs. For computational reasons, certain heuristics will also be included in the algorithm. The main theorems are stated first, while their proofs will be presented in the Appendix.

Theorem 1: Let N denote the number of vehicles in a group, S_i their initial positions and T_j the target points, for which (1) holds with $c = \frac{4}{\sqrt{7}}$. Applying phases 1 and 2 in Tab. I to the group will transfer every vehicle but the trapped ones to a target position in less than or equal to N steps.

Theorem 2: All the vehicles remaining in their start position after phases 1 and 2 in Tab. I can be transferred to the remaining target positions simultaneously in one final step, thus the algorithm always finds a suitable solution to the formation change problem.

1) Phase 1 – Direct Transition: During every step of this phase, the aim is to find as many routes as possible, along which vehicles can occupy empty targets in parallel. Routes are defined as follows.

Definition 2 (Route): A route connects an occupied start position and an unoccupied target point directly with a straight line.

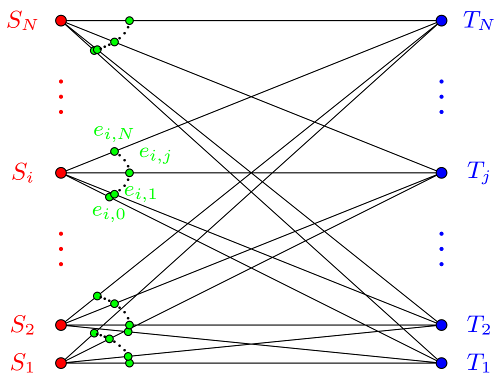


Figure 3. Path search graph.

First, a graph $\mathcal{G}$ describing the candidate routes has to be formed. The vertices of the graph correspond to the initial and target positions and the edges correspond to a route between an initial and a target point. Since in the simplest case every vehicle has the possibility of travelling towards any target point, this graph is a full bipartite graph (see Fig. 3).

Next, it should be checked whether vehicles stay within the safety region of a route or a route conflicts with another. Such routes have to be filtered out during the current step. In this context, conflict is defined as follows.

Let the start position of a vehicle be S_i and its target be T_i. Let us parametrise the path $\overrightarrow{S_iT_i}$ in the following well-known way:

$$\gamma s_i + (1-\gamma)t_i, \qquad \gamma \in [0;1], \tag{4}$$

where s_i and t_i are the vectors pointing to S_i and T_i, respectively, while γ can be considered as a time variable. This choice will simplify calculations since it means vehicles start and end their motion at the same time, no matter how long distances they need to travel. The distance between two moving vehicles is

$$d(\gamma) = \gamma(s_i - s_j) + (1-\gamma)(t_i - t_j). \tag{5}$$

Definition 3 (Conflicting routes): Two routes are in conflict with each other if the distance $d(\gamma)$ is less than the safety distance d_s for $\gamma \in [0;1]$.

These pieces of information can be collected into a "dual" graph $\mathcal{G}_d$ where each vertex corresponds to an edge in $\mathcal{G}$ (green circles in Fig. 3) and there is an edge between two vertices if the distance between the corresponding two routes is greater than d_s.

The task is then to find as many routes as possible among which there do not exist pairs that are in conflict with each other. In other words, a maximum clique has to be found within $\mathcal{A}(\mathcal{G}_d)$, which is the adjacency matrix of $\mathcal{G}_d$.

It is known that the maximum clique cannot contain more vertices than the number of vehicles. However, in most cases the size of the maximum clique is less than this value, due to the fact that vehicles can act as obstacles to each other, i.e. they are inside the safety region of a route. Therefore, the

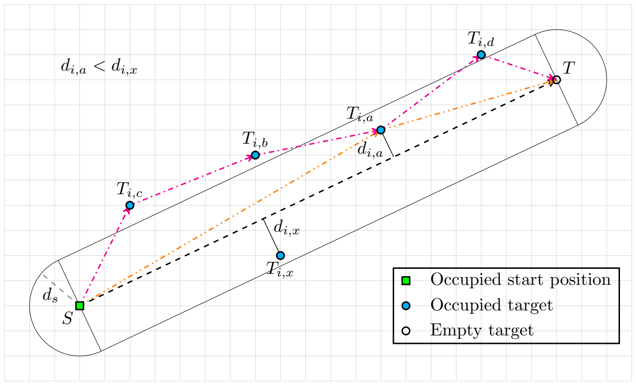


Figure 4. Correction route generation.

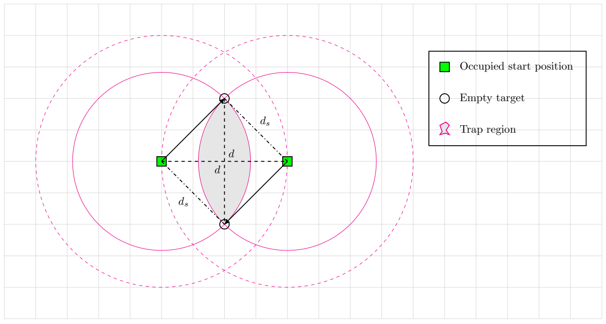


Figure 5. Trapped vehicles (extreme case, $c = \sqrt{2}$).

above method has to be repeated as long as there are new vehicles that can find their way to the targets.

Note that since stationary and moving vehicles constitute obstacles of different nature, certain vehicles that are unable to reach a target may be able to do so in later steps.

Clique search will be discussed in more detail in Section III-C.

2) Phase 2 – Correction Routes: Since the algorithm above cannot guarantee that all the vehicles reach a target position, a variant of this method has to be applied afterwards, which further reduces the number of vehicles remaining in their starting points. For this purpose, the notion of correction route has to be introduced.

Definition 4 (Correction route): A correction route connects an occupied initial position with an unoccupied target point via a chain of routes defined by intermediate occupied target points. No other vehicles stay within the safety regions of the constituting routes.

The purpose of correction routes is that along the segments of each such route the vehicles can regroup creating an unoccupied target point in the vicinity of an occupied starting point, which can be reached by a new vehicle. It will be shown that c_{min} in Theorem 1 guarantees that all the vehicles but the one in a start position may move in parallel without entering the safety region of another, which reduces the total time and energy required for the change of formation.

The construction of a correction route is an iterative process and consists of the following steps. The first task is to check if an occupied intermediate point T_i with minimum distance from the line section between the current start and target position (initially $\overrightarrow{ST}$) exists within the safety distance. The reason for selecting such point is to minimise the total length of the correction route. If no such point is found, the route is generated. Otherwise, correction route generation is split into two parts and thus the safety region changes (this is the reason for the iterative nature of the process). Finally, when the process is finished, the intermediate points are collected in the right order. Correction route generation is illustrated in Fig. 4. The first intermediate target point found during the process is $T_{i,a}$ since the other candidate $T_{i,x}$ is farther from $\overrightarrow{ST}$. The subscript indices correspond to the order in which the algorithm finds the intermediate points.

When searching for correction routes, it has to be ensured that each intermediate point is closer to the target point than the previous one (including the starting point) and that the subsequent routes are not in conflict with each other. Otherwise, correction routes could possibly be infinite.

If correction routes that satisfy the above requirements exist, another search, similar to the direct transition phase can be performed. The only difference lies behind the meaning of conflict between a pair of correction routes.

Definition 5 (Conflicting correction routes): Correction routes are in conflict with each other if the distance between any pairs of the constituting routes are in conflict with each other.

It can be proved geometrically that if $\sqrt{2} \leq c$, then the routes connecting the target points are never in conflict with each other. Moreover, all the intermediate points in the correction route are closer to the target than the previous one including the initial point (which already holds for $2/\sqrt{3} \leq c$). The extreme case corresponds to an isosceles right triangle whose sides are of length $\sqrt{2}d_s$, $\sqrt{2}d_s$ and $2d_s$. If any of the sides of this triangle increases, so does the minimum distance between the vehicles.

Due to the lack of distance constraints between start and target positions, the case involving the first two vehicles in a correction route is different. The simplest method for ensuring safe motion along the first line segment of a correction route is that the first vehicle waits until its path becomes safe. This can be improved considerably, however, for space reasons, discussion is omitted.

3) Phase 3 – Trapped Targets: In occasional cases, certain target points are left empty even after phases 1 and 2. We will call these targets trapped.

Definition 6 (Trap/trapped vehicle): A target point is said to be trapped if it lies within the safety region of two or more vehicles remaining in their initial positions after the correction route generation phase.

Such configuration is shown in Fig. 5. The most straightforward way to resolve these situations is to ensure that all the vehicles remaining in their start positions are involved in trapping target points and within every trapped region there is only one empty target point. The following lemma will provide the minimum c value for which safe paths can be generated in one step and show that the arrangement depicted in Fig. 5 corresponds to the extreme case.

Lemma 1: Let S_i and S_j be two occupied initial points that trap target T_i and let T_j be a target trapped by S_j and another initial point S_x. The trajectories $\overrightarrow{S_iT_i}$ and $\overrightarrow{S_jT_j}$ are not in conflict with each other if and only if $\sqrt{2} \le c$ holds.

Proof: We will prove the lemma by translating the conditions into an optimisation problem and show that $c_{tr} = \sqrt{2}$ is necessary and sufficient for the safe trajectories. The minimum distance between the two vehicles is the solution to the problem below.

$$f_{min} = \min_{s_j, t_j, \gamma} d^T d \tag{6a}$$

subject to

$$0 \le \gamma \le 1 \tag{6b}$$

$$\left.\begin{aligned} (s_j - s_i)^T (s_j - s_i) \le d_s^2 \\ (s_j - t_i)^T (s_j - t_i) \le d_s^2 \\ (t_j - t_i)^T (t_j - t_i) \le d_s^2 \end{aligned}\right\} \tag{6c}$$

$$(s_j - t_j)^T (s_j - t_j) \ge (c_{tr}\, d_s)^2. \tag{6d}$$

Here the objective function is the squared distance of two moving vehicles. The constraints in (6b) are related to traps and (6d) controls the relative position of the two targets. The position of the two starting points is fixed. This problem can be transformed into a problem which only contains equation constraints.

$$f_{min} = \min_{p_1, q_1, \gamma, \varepsilon_i, \lambda} d^T d + \lambda^T \phi, \tag{7}$$

where the vector ϕ contains the equation constraints above and λ is the vector containing the Lagrange multipliers. The constraints are defined as follows.

$$\begin{aligned} 0 &= \gamma(1-\gamma) - \varepsilon_0^2 \\ 0 &= (s_j - s_i)^T (s_j - s_i) - d_s^2 + \varepsilon_1^2 \\ 0 &= (s_j - t_i)^T (s_j - t_i) - d_s^2 + \varepsilon_2^2 \\ 0 &= (t_j - t_i)^T (t_j - t_i) - d_s^2 + \varepsilon_3^2 \\ 0 &= (s_j - t_j)^T (s_j - t_j) - (c_{tr}\, d_s)^2 - \varepsilon_4^2. \end{aligned} \tag{8}$$

The problem above can be solved by the well-known Lagrange multipliers method. Careful examination of the gradient reveals that the location of the optimum has the following properties:

- The four spatial points lie in the same plane,
- $\|s_j - s_i\| = \|s_j - t_i\| = \|t_j - t_i\| = d_s$,
- $\|s_i - t_i\| = \|s_j - t_j\| = c_{tr}\, d_s$.

The four points thus form a trapezium whose fourth side's length is $(c_{tr}^2 - 1) d_s$. The minimum distance is

$$d_{min} = \sqrt{f_{min}} = \frac{c_{tr}^2}{2} d_s, \tag{9}$$

which is not less than d_s if and only if $\sqrt{2} \le c_{tr}$. ■

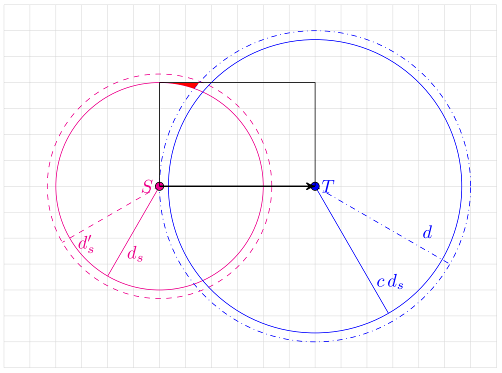


Figure 6. Ensuring convergence to the target.

B. Generating Suitable Correction Routes

The problem mentioned in III-A2 is illustrated in Fig. 6. Suppose a correction route has to be generated from start position S and target T. When generating the correction route, vehicles may have already occupied target positions in the red area, which lies within the safety region of route $\overrightarrow{ST}$. The distance between a vehicle in the red area and the target is greater than $\|\overrightarrow{ST}\|$. Since these points cause divergence from the target, it should be avoided that correction routes include them as intermediate points. Moreover, for a target T_i within the red zone, $S \in R_{\overline{T_iT}, d_s}$ holds, since $\|\overline{ST}\| < \|\overline{T_iT}\|$.

A solution to this problem is as follows. If all the routes and correction routes that end in a target point which has at least one occupied initial point within an increased safety distance d'_s are filtered out, then it is ensured that suitable correction routes can be found in each step. The ratio between d'_s and d_s can be read from the figure when $d = c \cdot d_s$:

$$d'_s = d_s \cdot c \sqrt{2\left(1 - \sqrt{1 - \frac{1}{c^2}}\right)}. \tag{10}$$

The downside, however, is that c_{min} has to be increased by the same ratio, as it is revealed by the configuration depicted in Fig. 7. A vehicle in the red region in Fig. 6 may block vehicles from reaching targets. If these points are kept empty, they may act as if they were trapped, thus they are treated as trapped. Therefore, the ratio between d and d'_s should be kept

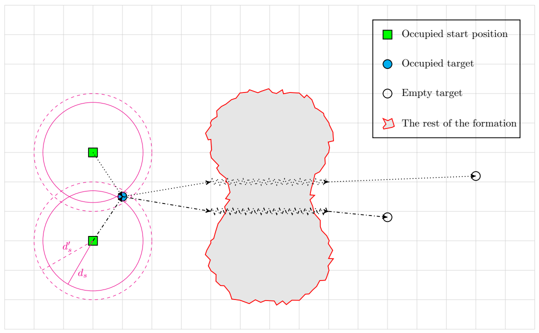


Figure 7. Difficulty caused by vehicles in the red zone in Fig. 6.

at $\sqrt{2}$, which yields $c_{min} = \frac{4}{\sqrt{7}}$. It has to be mentioned that the change is less than 7%, therfore it is not an overly strict constraint, especially if we consider that the densest achievable arrangement for $c = \sqrt{2}$ corresponds to a cubic grid.

A further question that needs to be investigated is whether it is possible to design the trajectory of the first vehicle in the correction route without increasing c_{min} so that it remains safe. Recall that in case $c = \frac{4}{\sqrt{7}}$, d_s is the distance between S and $\overline{T_iT}$ but not the minimum of $d(\gamma)$.

C. Clique Finding in $\mathcal{A}(\mathcal{G}_d)$

A number of maximum clique search algorithms have already been developed by research groups, see e.g. [14]–[17]. The algorithm presented in [17] is considered as an efficient method in most cases, thus it is applied to our problem as well.

Since finding a maximum clique in a graph is known to be NP-complete [18], certain modifications and additional heuristics are necessary to be applied to the algorithm to make it tractable in case the number of vehicles reaches the order of 50. One way of accelerating the search is that during the graph construction step, only a subset of all possible routes are considered. Selection is made after sorting the target distances from each initial position. Based on the order, n routes are selected as evenly as possible (e.g. if $n = 3$ and there are 5 empty targets, we select the 1st, 3rd and 5th shortest routes). This method performed the best among the ones we tried. Note that the most important in the selection method is that the shortest route should always be selected, which – by construction – ensures that at least one vehicle will reach a target in every step. Also note that this step is important because of the considerable time required for creating $\mathcal{A}(\mathcal{G}_d)$ itself, the size of which is N^2-by-N^2!

Even though this modification greatly decreases the search space, finding the maximum clique in the reduced graph may still require a long time. In most practical cases a first candidate clique is found in a relatively short time, and its size is not much less than that of the maximum clique. Finding new candidates can be time consuming. Thus, a time limit is introduced that cancels further search if a new candidate is not found within this limit.

The above modifications are destructive in the sense that applying them most likely results in finding a clique whose size is less than that of the maximum clique of the original adjacency matrix. However, all the vehicles still reach a target point, though the number of the required steps may increase.

Time and energy consumption can also be taken into consideration. Since route lengths are already available when the clique search begins, these pieces of information can be utilised as a tie-breaker when sorting the vertices based on their degree (c.f. lines 9 – 13 of Fig. 4 in [17]). This way, the shortest routes are checked as early as possible.

D. Integrating Obstacle Avoidance into the Algorithm

Suppose that a vehicle stops receiving state information from the other vehicles or does not access the reference trajectory but is still capable of maintaining its position in the frame of reference. A similar case is the existence of static obstacles in the environment that constrain the motion of the vehicles. We will show that a slight modification to the algorithm enables us to treat such cases.

Malfunctioning vehicles can be treated as follows. The main problem with malfunctioning vehicles is that they act as obstacles. In phases 1 and 2 of the original algorithm the case was similar. However, in phase 2, vehicles could be involved in correction routes, which solved the problem. This is not true for malfunctioning ones, therefore, a modified concept is needed, so that the correction route method could still be applied.

The key idea is that we introduce virtual occupied target points around the malfunctioning vehicle. When correction routes are generated, it has to be ensured that along the segments of each correction route no vehicle enters the safety region of the malfunctioning vehicle. A suitable way of ensuring this is that virtual targets are distributed over the surface of a sphere with a radius $c_{min}d_s$ around the vehicle. The distance between the virtual targets should be as long as possible and at the same time short enough to guarantee that correction routes will remain on the "surface" of the sphere. A close to optimal choice of this distance is d_s.

The algorithm of phase 1 needs no modification since during this phase there is no difference between a malfunctioning vehicle and a vehicle in a target position. These vehicles only need to be included as targets to avoid. In phase 2, in contrast, distinction has to be made between real and virtual targets for two reasons.

1) Virtual targets are close to each other, which affects correction route generation.
2) A vehicle has to travel from a real target to another real one, occasionally along virtual target points.

The second reason is rather technical, the first one needs explanation. Correction route generation is effective if the virtual targets do not divert a vehicle to a wrong direction. This can be avoided if the closest neighbours of a virtual intermediate point are filtered out from the set of new candidate virtual points. This condition can be formalised as follows. If

$$\|t_v - t_i\|^2 < \frac{1}{2}\|s_i - t_i\|^2 \tag{11}$$

holds for a virtual target t_v, when s_i and t_i are the current start and end point of the correction route segment, then it is ignored. This has the additional benefit that the correction route leaves the surface of the sphere as soon as possible, not at the closest point to the end of the route.

The method presented here will be illustrated in the second example in the following section.

IV. Practical Examples

A. Simple Formation Change Scenario

As a first illustrative example, a formation change manoeuvre involving a group of 25 quadrotors is presented. The vehicles are placed randomly in the 3D space and the target

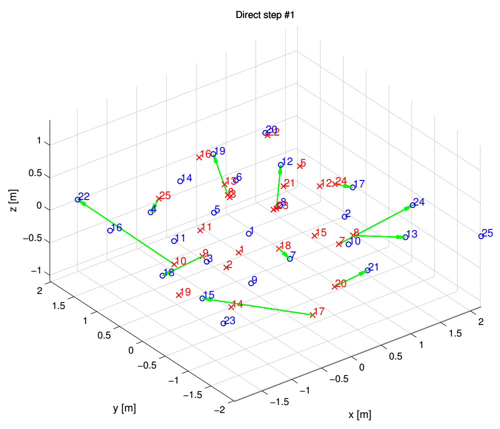


Figure 8. Example scenario 1, direct phase, step 1.

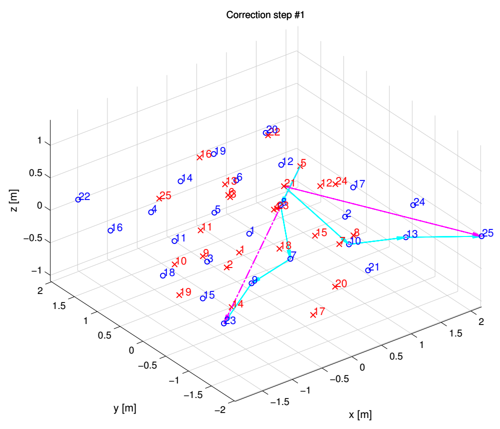


Figure 9. Example scenario 1, correction phase, step 1.

positions are chosen randomly in the xy-plane, satisfying the constraints of (1) with $c = \frac{4}{\sqrt{7}}$. The vehicles point to the same direction ($\Psi_{d,i} = 0$) throughout the mission.

Communication topology is chosen randomly, two vehicles are connected with a probability of 0.2, which means that each vehicle exchanges information with 5 others on average. For simplicity, the topology is fixed throughout the mission.

The coefficients of the backstepping controller and the robust formation controller are tuned so that the quadrotors track constant and ramp reference paths at a desired performance. Robust stability is achieved and all the designed controllers are stable. The full formation-level controller is obtained by placing the four controllers in the diagonal of a 4-by-4 matrix.

Reference paths are generated so that the speed of vehicles never exceeds 1 m/s. Specifying a suitable maximum speed is also necessary for guaranteeing the stability of the backstepping controller of each vehicle. Computation time statistics are shown in Tab. II, where columns $t_{\mathcal{A}(\mathcal{G}_d)}$, t_{MC} and $|MC|$ show the time required for adjacency matrix generation, finding a maximum clique and the clique's size, respectively. Tests were performed by the aid of MATLAB on an average P4 PC. All the algorithms were executed on a single core. It can be seen that the most time consuming step is the first, in particular the adjacency matrix generation, which is common in general situations.

The first steps of phases 1 and 2 in the example formation change are shown in Figs. 8 and 9 (the rest are omitted for space reasons). The graphs show the paths of vehicles involved in the transition steps. Start and target positions are marked by red crosses and blue circles, respectively. Only vehicles that change position are shown for transparency reasons. An additional dashed arrow connects the starting and end points of each correction route in Fig. 9. At each step, a maximum of 5 of all the possible routes are selected from each occupied start position. It is worth mentioning that trapped targets occur rarely in practice, since vehicles that might be involved in such situations usually find their way to different target points.

The safety distance is set to $d_s = 0.45$ m. Throughout the whole simulation, the minimum distance between two vehicles is 0.46 m.

Table II
PATH GENERATION STATISTICS.

Phase	Step #	$\mathbf{t}_{\mathcal{A}(\mathcal{G}_d)}$	$\mathbf{t}_{\mathbf{MC}}$	$\|\mathbf{MC}\|$
Direct	1	0.8356 s	0.0288 s	11
	2	0.3865 s	0.0241 s	8
	3	0.0650 s	0.0024 s	4
Correction	1	0.0261 s	0.0007 s	2

B. *Obstacle Avoidance Manoeuvre*

The second example involves 18 quadrotors, two of which malfunction. The task is that the other 16 vehicles move above these vehicles so that they can safely land without risking a collision. The vehicles are initially in random places in the 3D space and the targets lie within two parallel planes. The points satisfy the distance constraints. All the other settings are similar to the previous example's.

For space reasons, only the correction route generation step involving virtual target points is shown in Fig. 10. Green coloured crosses depict the virtual targets, while the black crosses correspond to the malfunctioning vehicles. To illustrate the algorithm better, the radii of the spheres around the malfunctioning vehicles is increased to $2d_s$, thus the number of virtual targets around a malfunctioning vehicle is 49.

Calculations are performed on the same machine as before and the total time required for the formation change manoeuvre design is 0.8 s.

CONCLUSION

The proposed path generation method together with a carefully tuned robust formation controller is capable of guaranteeing a safe formation change with a practically negligible constraint on the formation topology for any type of vehicles. Additionally, the framework is suitable for solving certain obstacle avoidance problems. The efficacy of the algorithm was shown in formation change missions involving a large group of quadrotor helicopters.

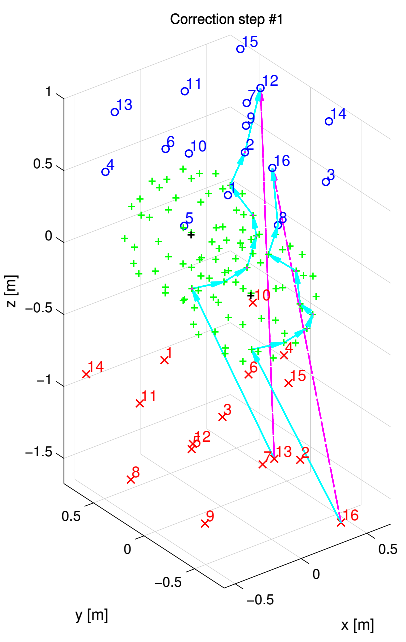


Figure 10. Example scenario 2, correction phase, step 1.

The algorithm can be accelerated by performing computations in a distributed manner. Further methods with robust performance allowing constraints on the controller can also be taken into consideration, which are to be investigated in the near future.

Acknowledgement

This research was supported by the MTA – BME Control Engineering Research Group and MTA SZTAKI. The work of B. Lantos was supported by the European Union and Hungary in project TÁMOP-4.2.2.A-11/1/KONV-2012-0012: "Basic research for the development of hybrid and electric vehicles".

Appendix – Proofs of Theorems 1 and 2

Proof of Theorem 1: It is straightforward that routes found in phase 1 may be considered as correction routes. It is sufficient to show that omitting phase 1 and applying phase 2 from the beginning of trajectory generation leaves only trapped targets. Since after every step in phase 2, the number of occupied start positions decreases, it is obvious that the number of required steps is not greater than N.

The first part of the theorem follows from the fact that a target point T_j is possibly excluded from the search only if there exists a start position S_i for which $S_i \in R_{T_j,d'_s}$ holds. Otherwise, there exists an occupied start position S_i closest to T_j and there exists a correction route from S_i to T_j if the points satisfy (1) with $c = \frac{4}{\sqrt{7}}$. ■

Proof of Theorem 2: Let the distance ratio be $c = \frac{4}{\sqrt{7}} > \sqrt{2}$. The greatest distance between two points within the intersection of two start positions S_i and S_j is strictly less than $c\,d_s$ (see Fig. 5). Therefore, no intersection of R_{S_i,d_s} and R_{S_j,d_s} can contain more than 1 empty target. Since the number of vehicles trapping targets is equal to the number of trapped target points after phases 1 and 2, these intersections cannot be empty. As a consequence, trapped positions can only form closed chains or closed three-dimensional surfaces (they may form separate similar structures). In case vehicles travel at constant speeds along straight lines, they never enter the safety region of other vehicles, as it has been shown in Lemma 1. ■

References

[1] T. Keviczky, F. Borrelli, and G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105–2115, 2006.

[2] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[3] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, Dec. 2004, pp. 4286–4291 Vol.4.

[4] T. Schouwenaars, J. How, and E. Feron, "Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 2004, p. 14 pp.

[5] L. Huang, "Velocity planning for a mobile robot to track a moving target - a potential field approach," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 55 – 63, 2009.

[6] L. Tang, S. Dian, G. Gu, K. Zhou, S. Wang, and X. Feng, "A novel potential field method for obstacle avoidance and path planning of mobile robot," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 9, 2010, pp. 633–637.

[7] E. Shaw and J. Hedrick, "String stability analysis for heterogeneous vehicle strings," in *American Control Conference, 2007. ACC '07*, July 2007, pp. 3118–3125.

[8] J. K. Rice and M. Verhaegen, "Distributed computations and control in multi-agent systems," in *Proc. 4th Int. Conf. Autonomous Robots and Agents ICARA 2009*, 2009, pp. 44–49.

[9] P. Massioni, T. Keviczky, E. Gill, and M. Verhaegen, "A decomposition-based approach to linear time-periodic distributed control of satellite formations," *Control Systems Technology, IEEE Transactions on*, vol. 19, no. 3, pp. 481–492, may 2011.

[10] P. Massioni, "Decomposition methods for distributed control and identification," Ph.D. dissertation, Delft Center for Systems and Control, 2010.

[11] A. Popov and H. Werner, "A robust control approach to formation control," *Proc. European Control Conference, Budapest, Hungary*, pp. 4428–4433, 2009.

[12] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[13] G. Regula and B. Lantos, "Backstepping based control design with state estimation and path tracking to an indoor quadrotor helicopter," *Periodica Polytechnica Electrical Engineering*, vol. 53, no. 3–4, pp. 151–161, 2009.

[14] P. San Segundo, F. Matia, D. Rodriguez-Losada, and M. Hernando, "An improved bit parallel exact maximum clique algorithm," *Optimization Letters*, pp. 1–13, 2011.

[15] M. C. Schmidt, N. F. Samatova, K. Thomas, and B. H. Park, "A scalable, parallel algorithm for maximal clique enumeration," *J. Parallel Distrib. Comput.*, vol. 69, no. 4, pp. 417–428, Apr. 2009.

[16] E. Tomita, Y. Sutani, T. Higashi, S. Takahashi, and M. Wakatsuki, "A Simple and Faster Branch-and-Bound Algorithm for Finding a Maximum Clique," in *WALCOM: Algorithms and Computation*, M. Rahman and S. Fujita, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 5942, ch. 18, pp. 191–203.

[17] J. Konc and D. Janezic, "An improved branch and bound algorithm for the maximum clique problem," *MATCH Communications in Mathematical and in Computer Chemistry*, June 2007.

[18] R. V. Book, R. M. Karp, R. E. Miller, and J. W. Thatcher, "Reducibility Among Combinatorial Problems." *The Journal of Symbolic Logic*, vol. 40, no. 4, pp. 618+, Dec. 1975.