Original Article

Institution of MECHANICAL ENGINEERS
Journal of AEROSPACE ENGINEERING

Proc IMechE Part G:
J Aerospace Engineering
0(0) 1–21

DOI: 10.1177/0954410016647074
uk.sagepub.com/jaero
SAGE

Distributed reactive collision avoidance for a swarm of quadrotors

J Leonard, A Savvaris and A Tsourdos

Abstract

The large-scale of unmanned aerial vehicle applications has escalated significantly within the last few years, and the current research is slowly hinting at a move from single vehicle applications to multivehicle systems. As the number of agents operating in the same environment grows, conflict detection and resolution becomes one of the most important factors of the autonomous system to ensure the vehicles' safety throughout the completion of their missions. The work presented in this paper describes the implementation of the novel distributed reactive collision avoidance algorithm proposed in the literature, improved to fit a swarm of quadrotor helicopters. The original method has been extended to function in dense and crowded environments with relevant spatial obstacle constraints and deconfliction manoeuvres for high number of vehicles. Additionally, the collision avoidance is modified to work in conjunction with a dynamic close formation flight scheme. The solution presented to the conflict detection and Resolution problem is reactive and distributed, making it well suited for real-time applications. The final avoidance algorithm is tested on a series of crowded scenarios to test its performances in close quarters.

Keywords

Unmanned aerial vehicle, conflict detection and resolution, distributed reactive collision avoidance

Date received: 9 March 2015; accepted: 22 March 2016

Introduction

One of the primary problems in multi-vehicle systems is guaranteeing collision-free operation. Unlike most control problems, collision avoidance lacks a defined goal state. Instead it requires that a number of conditions, generally inequalities regarding minimum distances, are met at all times. A great number of different solutions have been proposed to the collision avoidance problem, with different approaches depending on the specific application (manned aircraft, cooperative swarms, obstacles, non-cooperative intruder vehicles, etc.). A review of the most widely used ones was published by Massachusetts Institute of Technology researchers Kuchar and Yang.[1]

Most conflict detection approaches consider either the horizontal plane or three-dimensional space, ground proximity warning system[2] being the only relevant example of a purely vertical approach. This, however, does not imply that the conflict resolution provided will make use of the same space. A well-known example of this limitation is the traffic alert and collision avoidance system (TCAS)[3] used by manned civil aircraft, which considers conflicts in both the horizontal and vertical directions, but produces strictly vertical resolution advisories (RAs). Other approaches such as those of Shin et al.[4] and Waller and Scanlon[5] try to improve on these limitations for an optimal use of the technology over non-segregated airspace. NASA's Airborne Information for Lateral Spacing system for parallel landing,[5] make a conservative worst-case assumption (triggering an alarm whenever any possible set of future trajectories will lead to a conflict) or consider the whole probability space, determining the probability of a conflict based on the likelihood of possible future trajectories. The path predictive avoidance used in Melega et al.[6,7] manages a three-dimensional avoidance by defining step variations in the heading and altitude of the vehicles. The method requires, however, a previous knowledge of the paths taken by all the conflicting agents, which is unavailable in the present case.

Once a conflict has been detected or predicted according to the appropriate criteria, a fundamental aspect of conflict detection and resolution (CD&R) methods is how the conflict is resolved. These systems are meant for sporadic use and are more concerned

Center of Cyber-Physical Systems, Cranfield University, UK

Corresponding author:
J Leonard, Center of Cyber-Physical Systems, Cranfield University, College Road, Cranfield MK43 0AL, UK.
Email: j.leonard@cranfield.ac.uk

with safety than with optimality. However, a collision avoidance algorithm designed for frequent or permanent interactions cannot afford to overlook optimality. Policy-based methods for autonomous vehicles, like the one presented in 2007 by Pallottino et al.,[8] offer the advantage of scalability, as their implementation is decentralised and computation time is often independent from the number of vehicles in the system.

Optimisation approaches define a cost function (which may be based on controlling action, energy consumption, time delay or spatial deviation among other criteria) and compute the set of trajectories with the lowest cost resulting in a safe deconfliction. TCAS, for example, finds the least-aggressive manoeuvre within a limited set of climb or descent actions while achieving an escape from the conflict. Protocol approaches, like the Hwang–Tomlin algorithm,[9] can be considered as a combination of prescribed and optimised methods, as they run an optimisation search through a set of prescribed manoeuvres. Other methods, like the dynamic window approach,[10] use a short lookahead and only consider those potential trajectories which are reachable within a short time interval given the vehicle's dynamic constraints. In general, however, the search becomes computationally intensive as the space of possible trajectories increases making it unsuitable for real-time implementation. Integrated guidance and control approaches have therefore been proposed in the missile guidance and UAV control literature to minimise the overall reaction times for reactive collision avoidance.[11] However, the technique is mostly used to avoid static obstacles rather than coordinate multi-vehicle collision scenarios.

The distributed reactive collision avoidance (DRCA) algorithm, developed by Lalish[12] and further discussed in Lalish and Morgansen,[13] is a distributed method providing mathematically provable avoidance guarantees between any number n of vehicles with limited control authority. The block diagram on Figure 1 illustrates the basic functioning of this CD&R approach.

Originally providing steering controls for the simple constant speed unicycle model,[14] the solution was later extended to variable speed vehicles on the horizontal plane with the ability to stop and reverse[15] and generalised to three dimensions.[16] Finally, the detection scope of the DRCA was increased to include the avoidance of fixed obstacle. The improved deconfliction was implemented on quadrotors[17] and used to mimic biological systems.[18] It is important to note that the algorithm models vehicle dynamics as second order, in other words, it assumes that acceleration commands have an immediate effect. While vertical acceleration commands have a practically immediate effect (as propeller inertia is negligible), horizontal acceleration commands will only cause body moments, which will then lead to body angular rates and eventually to the required attitude, producing the desired acceleration. As a result, propeller speed controls vertical acceleration but only horizontal snap (the second derivative of acceleration). In order to be nontrivial, the problem of collision avoidance needs to consider vehicle dynamics that are inherently fourth-order on the horizontal plane. It will be important to bear this in mind later on, when discussing the accuracy of the collision avoidance, as these higher order dynamics act in practice like a time delay in horizontal acceleration commands. Furthermore, in many cases, it must deal with additional issues such as velocity and acceleration saturations, control delays and non-holonomic motion constraints, all of which need to be accounted for if a reliably safe solution is to be provided. Although the final version of Lalish's DRCA[17] and its re-implementation by Anderson[19] both account for a three-dimensional environment, their conflict-breaking phase is still planar making it sub-optimal as the number of conflicting vehicles increases. Additionally, these algorithms were tested in a clear environment or with very sparse fixed obstacles. Just like Boardman et al.[18] investigates the performances of the DRCA with large number of conflicting vehicles, it is important to test its validity when the obstacle density increases. Automata for critical safety measures will therefore have to be added on top of the DRCA to guarantee that the vehicles keep a minimum clearance distance with its environment as well.

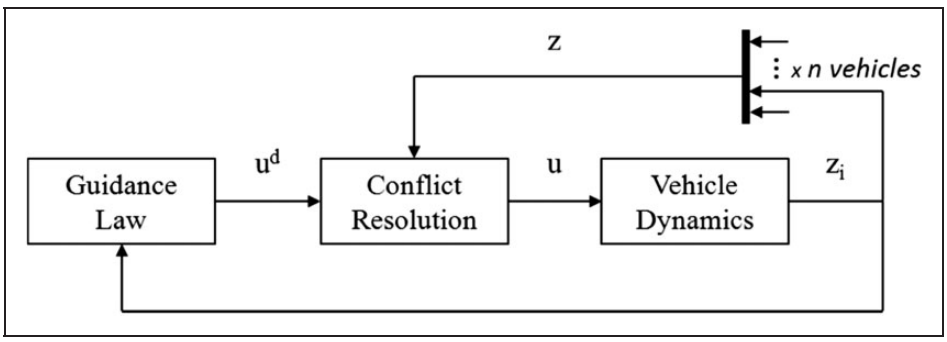

Figure 1. Block diagram of an n-vehicles system using a conflict resolution controller.

Distributed reactive collision avoidance

The DRCA algorithm developed in Lalish[12] is distributed, rather than decentralised because each vehicle requires position and velocity information from every other vehicle in the system, and not only from its nearest neighbours. The DRCA does not make any assumptions about the size, speed or actuation limitations of surrounding vehicles, which can be completely different from one another, as can be their respective tasks. Unlike other reactive force field approaches, this method is not merely based on relative position, but also on relative velocity, through the concept of *collision cone* introduced in Figure 2. The algorithm presents the advantage of not only reacting to an imminent conflict, but can also predict potential conflicts awaiting to happen and prevent vehicles from approaching them.

Collision and conflict

The collision cone concept was formally introduced by Fiorini and Shiller[20] in 1993. The main insight is that two vehicles that are moving towards each other at a constant velocity will eventually collide if nothing is done to alter this situation. By defining minimum separation distance, a safety circle or sphere can be imagined around one of the vehicles j, as shown in Figure 2, such that a collision is said to occur whenever the other vehicle i, considered as a point object, enters this region. Usually, the minimum separation distance is defined as the sum of the vehicles' radii (R_i and R_j) plus a safety margin to account for communication lag, sensor noises or model imprecisions d_{margin} so that

$$d_{sep} = R_i + R_j + d_{margin} \tag{1}$$

By considering vehicle i's relative velocity with respect to vehicle j, it is clear that if this velocity vector points to the inside of the safety region a collision is bound to occur eventually. Hence the collision cone (or triangle in two dimensions), also referred to in the literature as velocity obstacle, is defined as the set of velocities that point into the safety sphere (or circle) potentially causing a collision. According to this criterion, vehicles i and j are said to be in a conflict if and only if their relative velocity lies within their associated collision cone. In Figure 2, this would correspond to the relative velocity vector v being inside the grey collision cone. Note that the reciprocal geometric construction can be made for vehicle j towards vehicle i with an identical result. Although Chakravarthy and Ghose[21] extend the cone concept to irregularly shaped objects, the DRCA algorithm uses the simpler, reasonably conservative spherical approach.

Working principle

The DRCA algorithm works in two phases, a deconfliction manoeuvre and a deconfliction maintenance phase, and its avoidance guarantee rests upon three results illustrated by the flow chart in Figure 3. The first result is that as long as the vehicles are separated by a certain length, a deconfliction manoeuvre will prevent them from colliding. The second is that the vehicles will reach a conflict-free state at the end of the manoeuvre. The deconfliction maintenance controller then ensures that system remains that way as a final result. The purpose of the first phase is to quickly reach a state where all relative velocity vectors are outside their associated collision cones, so that the strong guarantees in the second phase will apply. It can in principle use any deconfliction criterion such as the prescribed 'all turn left/right' rule used in

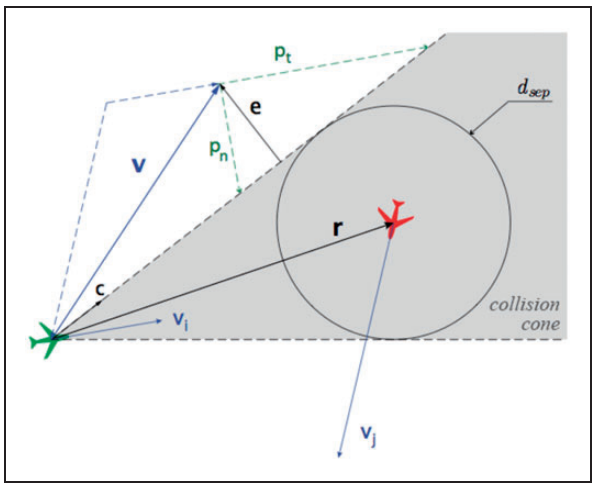

Figure 2. DRCA geometry to illustrate the collision cone concept. v_i and v_j are the vehicles' speeds, r is their relative position and v their relative velocity. The conflict measures p_t and p_n are shown, as well as the cone vectors c and e.

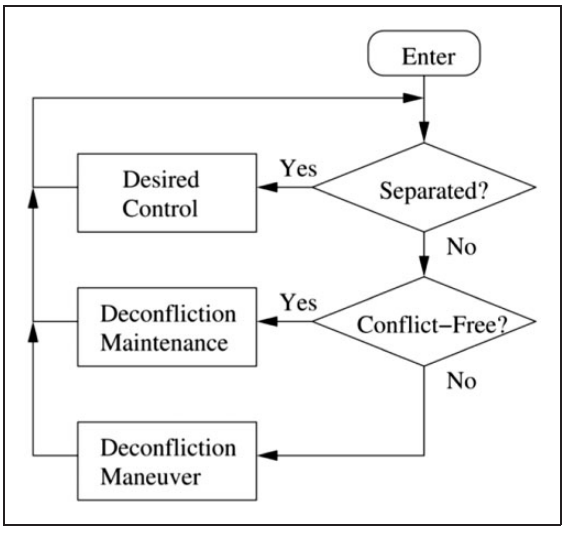

Figure 3. Distributed reactive collision avoidance high-level flow chart.[12]

Morgansen and Tsukamaki[14] and Lalish and Morgansen[15] or the optimisation algorithm later proposed in Lalish and Morgansen.[13]

In fact, it is reasonable to adapt this deconfliction manoeuvre to the specific dynamic capabilities of each vehicle involved, provided it can be shown to work safely given reasonable initial conditions. A different deconfliction law is implemented for the system considered in the present work, which will be detailed in the following section.

It should be noted that reaching a conflict-free state is not always possible: one can always come up with a set of pathological initial conditions such that, given control authority limitations, no combination of control commands will possibly avoid a collision (e.g. two vehicles initially very close and moving towards each other at a very high speed).

Once a conflict-free state is reached the main phase of DRCA, called deconfliction maintenance, comes into play. In this phase, collision cones effectively repel the relative velocity vector in such a way that it guarantees the vehicle will never accelerate into a conflict again. For this, each vehicle needs a way to measure how close its velocity vector is to the surrounding collision cones. For every interaction considered, the algorithm finds the collision cone boundary that is closest to the velocity vector $\mathbf{v}$. In three-dimensional space this corresponds to finding one of the cone generatrices, but by projecting the problem on the plane defined by the relative position and relative velocity vectors, as pictured on Figure 2, the cone is restricted to a triangle and the number of boundary vectors is reduced to 2. Let c be the unit vector defining the side of the collision cone closest to $\mathbf{v}$ and I the identity matrix, the distance from $\mathbf{v}$ to the cone boundary can be obtained by equation (2). The full mathematical derivation is outside the scope of this paper, but can be found in Lalish.[12]

$$e = \begin{cases} \mathbf{v} & if\ c^T\mathbf{v} \leqslant 0 \\ (I - cc^T)v & if\ c^T\mathbf{v} > 0 \end{cases} \tag{2}$$

Note that when $\mathbf{v}$ points away from the cone ($c^T\mathbf{v} \leqslant 0$), the closest point on the cone is its tip, hence $e = \mathbf{v}$. With this geometry in place, the next step is to determine how much margin exists in the orthonormal frame before a conflict forms. The orientation of the coordinate system is arbitrary but choosing the tangent t, normal n and binormal b notation fixes the coordinates to the vehicle's body and helps to combine the effects of multiple collision cones. The signed distances p_k with $k \in \{t,\ n,\ b\}$ are each defined as the allowable independent avoidance control (variation in velocity) along direction k before a conflict is reached so that $p_{k,ij} = \|e_{ij}\|^2 / e_{ij}^T\, k$. Figure 2 shows the p_t and p_n distances along the tangential and normal directions.

Let us define three thresholds $\varepsilon_t,\ \varepsilon_n,\ \varepsilon_b > 0$ so that when $|p_k| > \varepsilon_k$, the conflict can be considered far enough in the direction k to be ignored. Vehicle i stores these distances for each vehicle j within the algorithm's interaction range and subsequently obtains the positive and negative values that are closest to zero (i.e. closest to a conflict) along each direction, respectively $p_{k,i}^+ = min_j\{p_{k,ij} > 0, \quad \varepsilon_{k,i}\}$ and $p_{k,i}^- = -max_j\{p_{k,ij} < 0, \quad -\varepsilon_{k,i}\}$.

The avoidance control command u_k created by the DRCA along direction k can then be constructed as a bilinear interpolation of the triples $(p_k^+,\ p_k^-,\ u_k)$ so that

$$u_k = \frac{p_k^+}{\varepsilon_k} u_k^{min} + \frac{p_k^-}{\varepsilon_k} u_k^{max} + \frac{p_k^+ p_k^-}{\varepsilon_k^2}\left(u_k^d - u_k^{min} - u_k^{max}\right) \tag{3}$$

where u_k^d is the desired control input produced by the vehicle's guidance law, and $(u_k^{min},\ u_k^{max})$ are the current actuation limits of the vehicle in this direction. Consequently, the extremes of the control function are

$$\begin{cases} u_k(p_k^+ = 0,\ p_k^- = 0) = 0 \\ u_k(p_k^+ = \varepsilon_k,\ p_k^- = 0) = u_k^{min} \\ u_k(p_k^+ = 0,\ p_k^- = \varepsilon_k) = u_k^{min} \\ u_k(p_k^+ = \varepsilon_k,\ p_k^- = \varepsilon_k) = u_k^d \end{cases} \tag{4}$$

This illustrates that a vehicle will never accelerate into a conflict, follow the desired control action if it is conflict-free, and remain within the actuation limits during a deconfliction manoeuvre to ensure that the commands passed back to the controller do not lead to a forbidden velocity. Although the total amount of computation for the overall system is $O(n^2)$, each vehicle accounts for its own interaction and therefore runs in linear time $O(n)$. This lets the algorithm scale efficiently for a large number of vehicles. Not being computationally expensive, the algorithm can be run at the same frequency as the main controller loop hence the use of the nominal state propagation in Figure 1 is acceptable.

Quadrotor-specific implementation

The DRCA algorithm described before constitutes the core for the collision avoidance scheme developed in this paper. Although the algorithm is universally applicable to any vehicle at a high level, there are several particularities that need to be addressed when applying the algorithm to a specific vehicle type. Additionally, some significant modifications have been introduced in the original algorithm to extend its functionality and improve the desired performance in concrete scenarios.

Guarantees and limitations

The DRCA algorithm guarantees that once a conflict-free state is reached between all vehicles in the group,

the system will remain conflict-free thanks to the fact that the collision maintenance phase will never allow a vehicle to accelerate into a conflict. This is a uniquely strong guarantee that makes DRCA one of the more reliable collision avoidance methods in the current state of the art and the full mathematical proof of it can be found in the original DRCA implementation.[12] Yet, it is important to note some of the assumptions it makes: firstly, vehicle dynamics are modelled as second order, which implies that acceleration commands are instantly followed by the vehicle. Secondly, it requires that all vehicles be running some form of the algorithm. Although later implementations of the algorithm included the handling of fixed obstacles[17,18] their use was limited so that the space available to the conflicting vehicles for avoidance was mostly unrestricted.

The following sections will detail how the DRCA algorithm can be extended to account for the quadrotor dynamics, taking full advantage of its capabilities, and giving it more versatility in crowded environments.

Modelling

The quadrotor is very simply modelled as four rotors equipped at each end of a cross. All the propellers axes are fixed and parallel. They are mounted directly on the DC motors shaft so there is no need for reduction gears. The blades have a fixed pitch and push the air downwards. The front and rear propellers rotate counter-clockwise while the left and right ones turn clockwise. This configuration of counter rotating blades removes the need of a tail rotor for classic helicopters to control their heading. The dynamics and full 6DoF equations of motion for this vehicle can be found in Bresciani[22] and Bouabdallah.[23]

Due to the quadrotor's rotational symmetry about its z_b body axis, it is appropriate to consider the dynamic limitations to be cylindrical, essentially distinguishing between vertical motion along z_b and horizontal motion perpendicular to it. As the quadrotor controller is designed to operate in near-hover conditions, it can be considered that the quadrotor's main axis remains vertical at all times so that $z_b = z$. One of the desired frame direction vectors is hence chosen to be the unit upward vertical vector z. On the other hand, with four controllable degrees of freedom, the quadrotor can move in any direction regardless of its yawing attitude. There are therefore no strong reasons to use the x_b and y_b body axes. Instead, the horizontal projection of the vehicle's velocity is used to define the horizontal tangent vector t, and the horizontal normal vector n. This frame is convenient in practice as it simplifies the implementation of the dynamic and kinematic constraints for quadrotor operations.

With most common aerial vehicles, a maximum speed is defined as a spherical constraint, where the norm of the velocity vector cannot exceed a certain value. However, quadrotor dynamics markedly differ in the vertical and horizontal planes because the quadrotor presents second-order along the z-axis but fourth-order along the other two axes. This distinction between vertical and horizontal motion laws imposes a cylindrical operational velocity space as pictured on Figure 4 resulting in the following two rules:

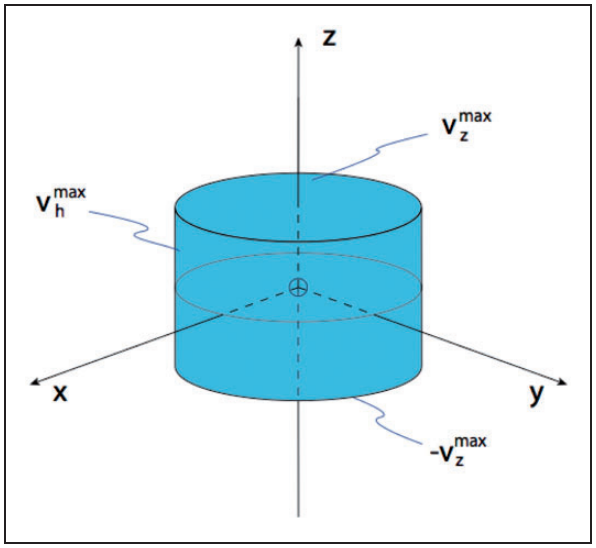

Figure 4. Velocity boundaries of the quadrotor for the DRCA as a cylindrical surface to distinguish between vertical and horizontal motion.

- If the norm of the vertical velocity reaches a maximum value v_z^{max} then no horizontal acceleration is allowed.
- If the norm of the horizontal velocity reaches a maximum value v_h^{max} then no vertical positive acceleration is allowed.

In the present implementation of the algorithm, v_z^{max} and v_h^{max} are chosen to be equal and relatively small ($\leqslant$ 5m/s) to keep the units safe while flying through a dense environment. This value could safely be made greater in an open environment where the obstacles are more sparse.

Limited horizon

The algorithm initially considers all other agents in the system but in a large environment with a significant number of dispatched units, it becomes computationally inefficient to maintain deconfliction with units far away. A filtering process to determine whether or not they should be of concern regarding collision avoidance should therefore be added based mainly on separation. A limited horizon is defined before the conflict detection phase for the DRCA algorithm to become active. This horizon can be set to a fixed value or somehow be correlated to the risk

of collision or to the degree of uncertainty. In this sense it might be desirable to make the algorithm's horizon grow with the vehicle's speed, in such a way that a vehicle moving slowly would require a smaller lookahead than a fast-moving one. This also makes vehicles that are moving fast begin avoidance sooner than those moving at a slower speed, which is generally desirable, as the latter are likely to be conducting operations requiring greater precision, such as tracking a target on the ground or approaching a landing point. In the implementation considered here, the algorithm's horizon radius γ defined in equation (5) varies linearly with speed from a minimum radius γ_{min} at zero velocity to a maximum value γ_{max} at the maximum possible velocity set by the previous velocity space.

$$\gamma_i = \gamma_{min} + \frac{\|\mathbf{v}\|}{v_{max}}(\gamma_{max} - \gamma_{min}) \tag{5}$$

The difference in quadrotor dynamics along the vertical and horizontal axes would call for an anisotropic lookahead horizon, such as an ellipsoid, by assigning different weights to the distance along each direction. For the sake of simplicity, a spherical horizon is used here instead, so that the filter is based on the Euclidean distance between agents. Other factors can affect the degree of uncertainty and the risk of collision, such as the number of vehicles in the system and the environment's obstacle density, but they are not taken into account in the present implementation.

Closest-escape iterations

The fundamental guarantees of DRCA are provided in the deconfliction maintenance phase, once all interacting vehicles are out of conflict. However, a drawback of the limited horizon is that two vehicles can find themselves in conflict by the time their horizons account for one another. From this initial conflict, the affected vehicles must first perform a deconfliction manoeuvre. There is no unique approach to define such manoeuvre and so far the DRCA has been implemented with methods like the 'all-turn-left' (or 'right') command or a velocity space optimisation technique. However, the solutions generated by the former method consist in a non-optimal constant-speed avoidance and though the velocity space optimisation approach fixes these issues, it does not provide a three-dimensional deconfliction and can even fail to provide a suitable solution. In view of these limitations, an alternate approach is proposed for this implementation, specifically intended for quadrotor platforms with high-order dynamics. The procedure is built around two different principles. The first is based on finding the closest escape from the collision cone within the complete three dimensional velocity space. The second is much like other traditional force-field conflict resolution methods, where elements constituting a potential threat exert a repulsive force on the vehicle. The accelerations resulting from both of these methods are superimposed to define the desired deconfliction manoeuvre.

The first principle, called closest-escape iterations, is based on a similar insight as the deconfliction maintenance phase. The algorithm looks for the closest point on the collision cone boundary, the only difference being that in this case the velocity vector is inside said cone. This point is determined by identical computations as from the outside of the cone, and can therefore be easily integrated into the general DRCA algorithm: if during the execution of DRCA, the relative velocity vector is found to be inside the collision cone, then vector e, defined in equation (2) as the vector pointing from the tip of the relative velocity vector to the closest point in the collision cone boundary, coincides with the smallest necessary velocity increment to exit the collision cone.

$$\Delta\mathbf{v}_{esc} = e \tag{6}$$

The algorithm in vehicle i then does two things. It firstly assumes that the velocity instantly jumps to the desired escape value

$$\mathbf{v}_i = \mathbf{v}_i + \Delta\mathbf{v}_{esc} \tag{7}$$

meaning that the current vehicle pair would no longer be in conflict. It subsequently restarts the conflict detection process for all relevant interactions using the new fictitious velocity, as new conflicts might have arisen due to the jump (i.e. the velocity vector might have jumped out of one collision cone and into another). If a new conflict is indeed found, the process is repeated and the velocity jumps again to the closest boundary, hence restarting the conflict-checking process. On the other hand, an interaction which does not cause a conflict is immediately discarded as it is not necessary to compute its deconfliction maintenance values. Once all interactions have been checked, the algorithm takes the resulting escape velocity and produces a maximum rate acceleration command towards it. This principle is illustrated in Figure 5 for a simple two-vehicle conflict.

In most conflict scenarios involving a reasonable number of vehicles, the closest-escape iterations will terminate after the first jump, meaning that the optimal three-dimensional solution has been found. However, a conflict scenario involving more vehicles in a dense environment might lead the current deconfliction process to enter an infinite loop where the velocity vector only jumps from one collision cone to the next without finding an escape solution. To prevent this loop from happening, the algorithm keeps track

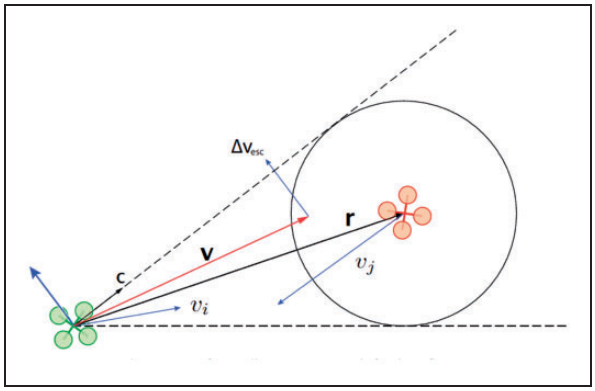

Figure 5. Geometry for the closest-escape deconfliction manoeuvre.

of the number of jumps n_Δ and augments the magnitude of the jump by a small amount δ at each iteration

$$\Delta \mathbf{v}_{esc} = (1 + \delta n_\Delta)e \tag{8}$$

Additionally a maximum number of iterations n_Δ^{max} is defined such that if reached, the algorithm no longer restarts, and instead moves straight to computing the acceleration required for the manoeuvre, but taking a desired escape velocity

$$\mathbf{v}_i = 0 \tag{9}$$

In other words, the algorithm attempts to stop the vehicle's motion if no solution has been found. This decision is purely heuristic, and is based on the precautionary insight that slowing the vehicle could reduce the chances of colliding or would at least provide more time for the situation to evolve and a feasible solution to appear. This could theoretically be applied to more than one vehicle in the conflict (until the conflict can be resolved in the defined number of jumps) but never appeared more than once in the most challenging scenarios. The stop was implemented as a safety measure to mitigate risks in group proximity flights and ensure the integrity of the units in all situations. The closest-escape iterative method was originally introduced and simulated as a standalone deconfliction criterion but it exhibited one significant limitation. Although a safe velocity is always found, the algorithm assumes that it can be reached instantly. But this is not true in practice. The quadrotor takes some time to reach it due to its limited accelerations and its limited horizontal snap. Moreover, the analytical approach described so far dealt with a continuous-time system, which is in practice 'discretised' by the controller update frequency.

In the original implementation of the DRCA,[12] gain-like parameters (K_t, K_n, K_b) were introduced to more intuitively choose values for (ε_t, ε_n, ε_b) in the tangent, normal and binormal directions so that $K_t = (u_{max} - u_{min})/\varepsilon_t$ (similar for normal and binormal directions). In a conflict between vehicles i and j, let $K = K_{t_i} + K_{n_i} + K_{b_i} + K_{t_j} + K_{n_j} + K_{b_j}$, it was demonstrated that the system remains stable in the presence of a complex multiplicative uncertainty bounded by $w(j\omega)$ if k satisfies

$$\frac{K}{\sqrt{K^2 + \omega^2}} < \frac{1}{|w(j\omega)|} \tag{10}$$

Simplifying the previous unmodeled dynamics to a pure time delay τ, the paper showed that the system is stable in the presence of such a delay if

$$K < \frac{1.47}{\tau} \tag{11}$$

In the original implementation, the operating environment was large and mostly free, allowing the values of the threshold ε to be fixed reasonably high. This would lead to smaller values of K and larger allowed delay τ. In the present case, however, the DRCA horizon was limited down to γ_i (from equation (5)) in order to avoid far away threats from being considered in the collision-avoidance. With a shorter horizon distance, the gain K becomes larger and the allowed delay τ becomes smaller.

In a crowded scenario, the delay between the computation of the optimal escape control and the moment the quadrotor manages to reach it can therefore surpass the allowed value and lead the vehicles to a collision. A second method is therefore introduced that is meant to delay the potential collision and give the vehicle enough time to reach its escape velocity and leave the collision cone.

Repulsive force field

The second principle is introduced as an enhancement to the basic closest-escape deconfliction manoeuvre described above, and is designed to work in conjunction with it. Similar to classical force field approaches, this method applies a fictitious repulsive force F_{rep} to vehicle i, directed away from the conflicting vehicle j, and with a magnitude inversely proportional to the square of the distance

$$F_{rep} = -a_{rep} \frac{1}{\|r_{ij}\|^2} \frac{r_{ij}}{\|r_{ij}\|} \tag{12}$$

where a_{rep} decides on the strength of the repulsive action. However, the force is not computed for all surrounding vehicles and obstacles, but only for those found to be in a conflict in order to help the iterative closest escape. The value of the a_{rep} is also made dynamic in an effort to avoid the infinite jump loops mentioned above. Every time the closest-escape process falls back into a collision cone it previously tried to escape, the magnitude of the repulsive force applied to the quadrotor in conflict is incremented.

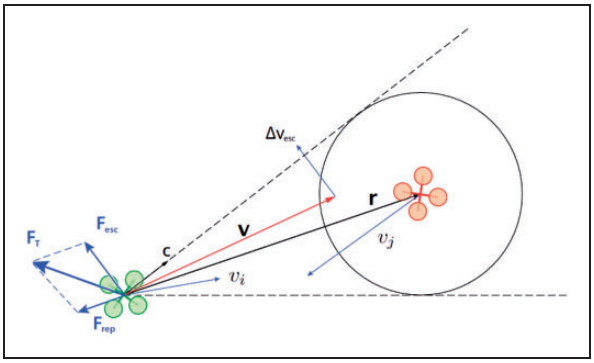

Figure 6. F_{esc} and force-field F_{rep} deconfliction manoeuvre.

This has the effect of giving each conflict a greater repulsive effect the more difficult it is to resolve by the closest-escape method. At the end of the conflict checking process, both avoidance laws are superimposed by adding their respective acceleration commands into a resulting command F_T as illustrated in Figure 6.

The avoidance control command u_k created by the DRCA along direction k, and previously defined in equation (3), can now be expressed as

$$u_k = \frac{p_k^+}{\varepsilon_k} u_k^{min} + \frac{p_k^-}{\varepsilon_k} u_k^{max} + \frac{p_k^+ p_k^-}{\varepsilon_k^2}\left(u_k^d - u_k^{min} - u_k^{max}\right) + u_{k_{rep}} \tag{13}$$

$$u_{k_{rep}} = -a_{rep} \frac{1}{\|r_{ij}\|_k^2} . \delta t . sgn\left(\|r_{ij}\|_k\right) \tag{14}$$

where $\|r_{ij}\|_k$ is the projection of the relative position between the two vehicles onto direction k, and δt is the time-step resulting from the discretisation of the continuous-time system. In the crowded scenario, the limited horizon and unmodeled dynamics can create a delay τ_{ud} greater than the allowed pure delay τ, so that the inequality of equation (11) does not hold anymore. The addition of the repulsive force F_{rep} tends to lower the value of K but it needs to be properly tuned for the system's stability to be restored. Let us define K_{DRCA} as the value of K before the repulsive action was added. Equation (11) can then be written as

$$K_{DRCA} + \frac{u_{rep}^{max} - u_{rep}^{min}}{\gamma_i} < \frac{1.47}{\tau_{ud}} \tag{15}$$

By increasing the repulsive factor a_{rep} until equation (15) holds true, the system will be stable with the added dynamics. The combined manoeuvre therefore presents better performance than the iterative method alone, as it usually helps reduce the approach velocity to the closest or most threatening conflict, giving the quadrotors enough time to escape the collision cone. The difference with respect to the standalone closest-escape iteration is especially noticeable in scenarios with a dense obstacle topology which will be introduced in the next section.

Degenerate coplanar case resolution

DRCA deals with potential conflicts by finding the closest point in the collision cone boundary and accelerating away from it when appropriate. Consider a three-dimensional case where three or more vehicles are lying on a common plane and are moving in such a way that their relative velocities are also contained in this plane. Now, if a conflict is approached, the DRCA algorithm will only modify the vehicles' trajectories within this plane, as the closest point in the collision cone is always in the plane defined by relative position and relative velocity (r–v plane). This means that throughout the whole avoidance process and subsequent motion, the vehicles will remain in the same plane creating a two-dimensional solution in a three-dimensional environment. This might be acceptable in some simple contexts, but clearly becomes non-optimal in complex scenarios, as it fails to make use of one whole dimension of space. Two-dimensional solutions in complex problems with multiple vehicles can result in more intricate trajectories and a greater amount of control action, let alone a longer execution time, essentially because the problem is being made unnecessarily hard by forcing a coplanarity constraint on the resolution process.

This situation can be avoided by deliberately making one or several vehicles accelerate out of the common plane, hence forcing a three-dimensional solution. As soon as the relative positions and velocities are no longer coplanar, the non-generic case is broken and a three dimensional resolution naturally follows. Instead of randomly choosing the units to force out of the conflict plane, the quadrotors are given priority levels. In practice, the DRCA presented here is used by a UAV swarm intelligence for energy management[24] and the priority values are based on health monitoring decisions which will not be covered in the scope of this paper. However, the effects of this prioritisation on the avoidance process are of interest.

Vehicle priority is first introduced in the DRCA by defining a danger horizon γ_D for the vehicle, on top of the velocity-based horizon γ described in equation (5). γ_D is always smaller than γ and is meant to replace the γ horizon of the higher priority vehicle when a conflict occurs. Therefore, when two vehicles with different priorities come within a certain distance from each other, the lower priority vehicle will react to a new interaction first, by conducting the appropriate action (deconfliction maintenance or manoeuvre) so as to maintain a conflict-free state with the conflicting agent. On the other hand, the higher priority vehicle will only need to react if separation becomes smaller than the danger radius γ_D, as shown in Figure 7.

This approach to vehicle priority is different than the one suggested in Lalish[12] for the original

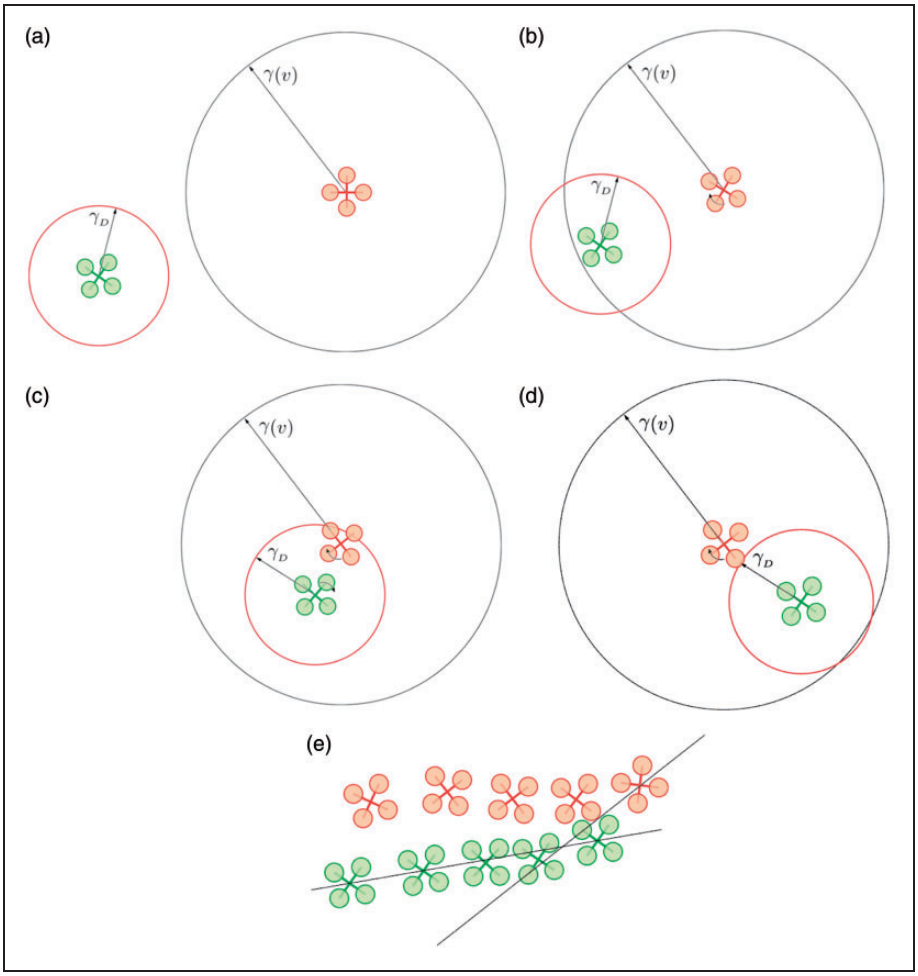

Figure 7. Example of double-horizon vehicle priority. The green vehicle has a higher priority.

implementation of the DRCA, where higher priority vehicles are assigned a smaller velocity threshold ε, equivalent to a thinner repulsion buffer around the collision cone. This way the higher priority vehicle would tolerate relative velocities closer to the collision cone until the conflict is effectively reached. The priorities are then ignored and all vehicles are treated equally. In most cases, the conflict resolution is initiated at a sufficiently large distance for avoidance to be feasible with only the lower priority vehicle manoeuvring since making the higher one manoeuvre can be undesirable. The approach proposed here will keep the higher priority vehicle on its desired path longer, hence giving more time for the conflict to be resolved by the lower priority quadrotor alone. It only makes the two vehicles manoeuvre when avoidance becomes close to unfeasible without cooperation from both agents.

Going back to the three-dimensional avoidance manoeuvre, a symmetry breaking criterion was required to determine which vehicles accelerate out of the plane and in which direction. In the present implementation, this criterion is based on the vehicles' priority level so that the vehicles with the most critical task are less likely to deviate from path. Ties in priority levels are arbitrarily broken using the vehicle ID number. Hence for a swarm of n vehicles, let us define this breaking criterion b as

$$b = n.\ priority + ID \tag{16}$$

If a vehicle i finds itself in a situation where it is interacting with two or more other vehicles on the same r–v plane, it computes its own value b_i as well as the value for each other vehicle b_j for $j \in \{conflict,\ j \neq i\}$. If it has the lowest of all values, it aims to accelerate perpendicularly out of the common plane in an upwards direction and at maximum rate. This counts to all effects as a velocity jump, which is checked for safety by re-running the DRCA with the new velocity.

To illustrate the relevance of the coplanarity resolution scheme presented here, consider the scenario depicted in Figure 8, where 25 point-mass vehicles (with quadrotor dynamics) are attempting to cross a circle diametrically at the same time, all intended trajectories converging through the centre. First, the system is run with a simple implementation of DRCA, with a pure closest-escape iterative approach for the deconfliction manoeuvre. The solution

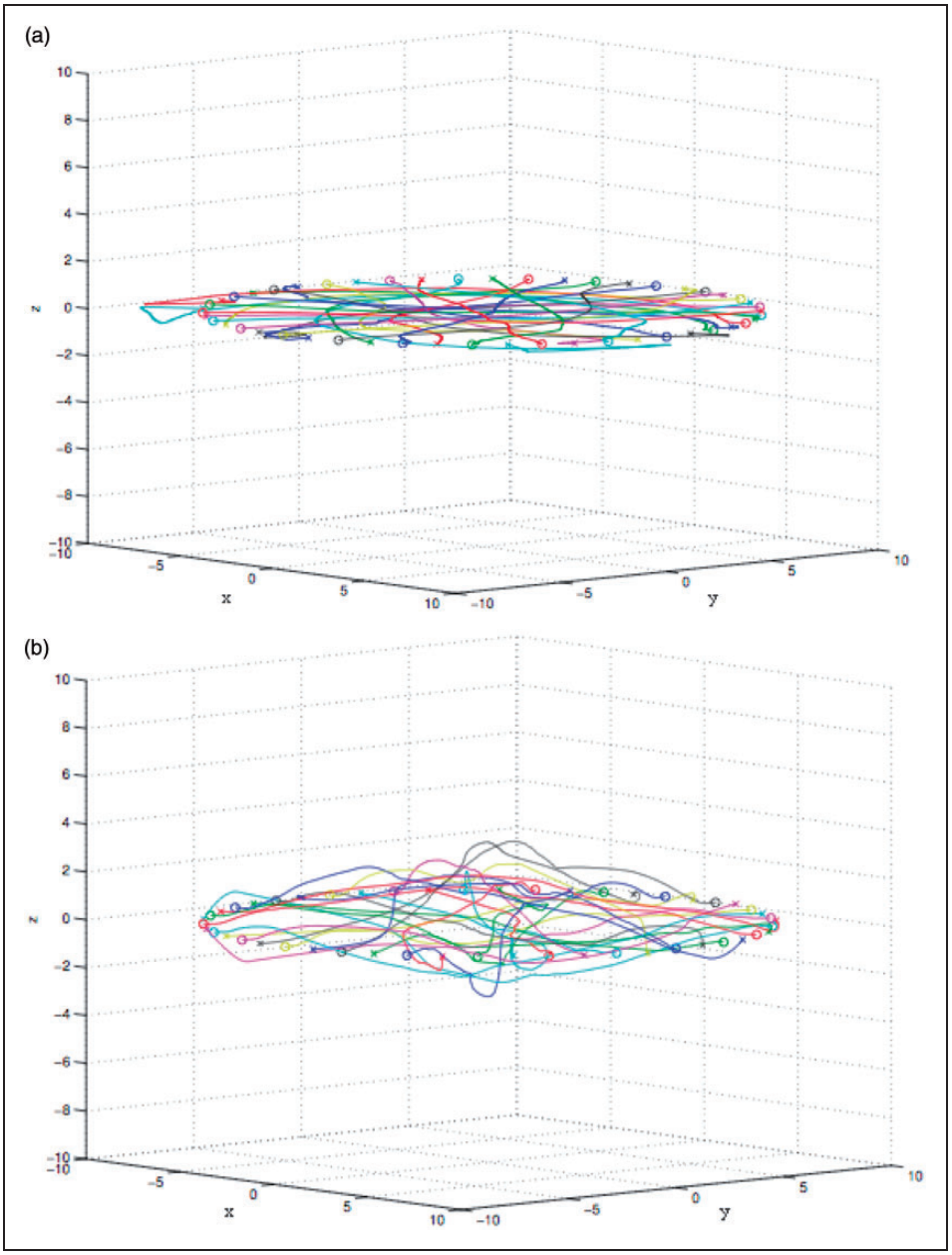

Figure 8. Vehicle trajectories over time: introduction of the third dimension in the deconfliction manoeuvre: (a) two-dimensional deconfliction; (b) three-dimensional deconfliction.

displayed in Figure 8(a) produces complex trajectories constrained within the horizontal plane. Some trajectories can even be seen reaching outside of the initial circle to escape the crowded space. On the other hand, the solution obtained after addition of the three-dimensional coplanar case resolution (Figure 8(b)) presents more direct trajectories for each vehicle making them easier to travel and reducing the time needed to reach their destination. If projected on the collision plane, all trajectories would fit in the initial circle.

In the scenario depicted in Figure 8, the time required to travel the diametrical path should be 20 s. The delay introduced by avoidance actions with respect to this planned completion time is 16.3 s with the standard two-dimensional resolution, which drops to 3.7 s with the forced three-dimensional solution, i.e. over four times shorter.

Obstacle avoidance

Until now, implementations of the DRCA algorithm have always considered an obstacle-free environment. However, the quadrotor's limited size and nonholonomic capabilities make it the perfect vehicle for flights in crowded spaces such as buildings or city streets.[25,26] The extension to the DRCA presented in this paper therefore tries to include a way to model fixed obstacles into the deconfliction process. Although the CD&R algorithm would already be complemented by a path planning routine which would account for the obstacle topology,

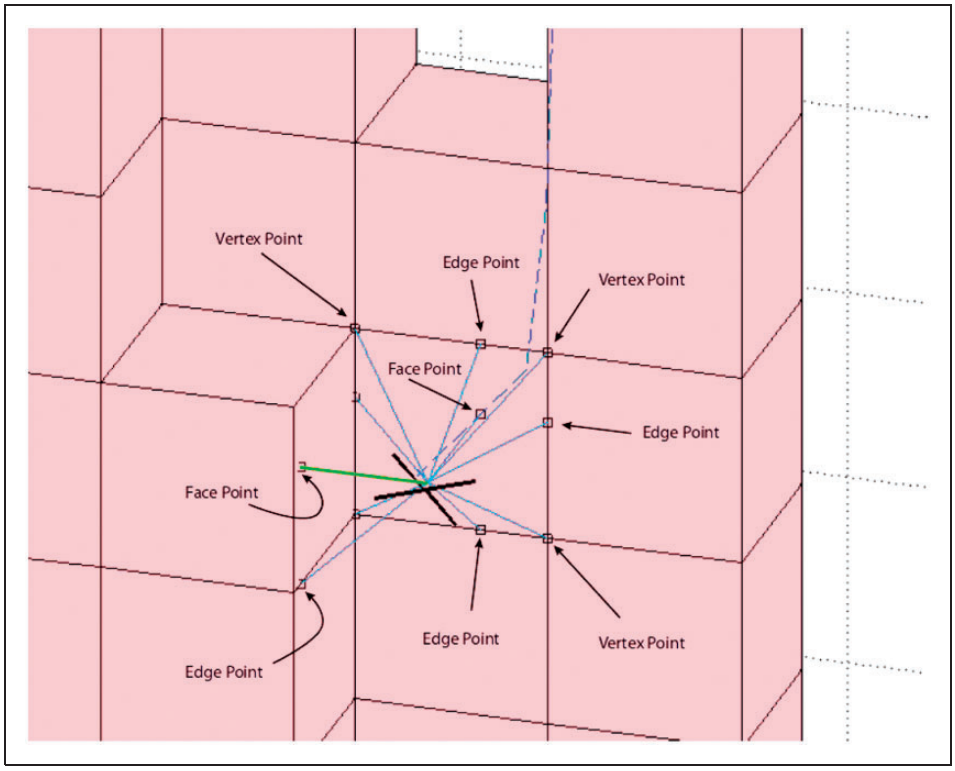

Figure 9. Simulation snapshot showing nearby obstacle cells transformed into obstacle points.

any deviations from the nominal path caused by a deconfliction manoeuvre could make a vehicle accelerate towards an obstacle if they are not included in the decision process.

The DRCA considers the vehicles as point objects surrounded by a protection sphere of radius d_{sep} defined in equation (1). This representation fits the quadrotor geometry well. Although the vehicle is generally thinner in the z direction, the vertical backwash from the propellers prevents quadrotors from getting too close above one another and so a spherical separation space is acceptable. However, the geometry of a floorplan or of a city street relates more to adjacent polygonal obstacles than spherical ones. For the present work, the map was decomposed as a fixed sized grid where each cubical cell is either free of obstacle or occupied. As no further information is available, the collision avoidance algorithm has to assume that each occupied cell is in itself a cube-shaped obstacle. It would be easier, of course, to extend the sphere-based approach used for inter-vehicle collision avoidance to obstacle cells, surrounding them with a protection sphere. This would provide a conservative approach, as each cube of side L can be replaced by a sphere of diameter $L\sqrt{3}$. However, this change effectively augments the size of each obstacle and, in a dense obstacle environment this will often occlude narrow passageways such as doorways in indoor settings. The method presented here is designed to accommodate any kind of mobile robot navigation in a cell environment with obstacles. Instead of augmenting obstacle cells into spheres, this approach reduces them to points. Each obstacle cell is converted into its closest point to the vehicle (thus the most dangerous one). This obstacle point behaves similar to the vehicle's shadow would on the actual obstacle, moving along the obstacle's surface. Consequently, obstacle points can be assigned a position and a velocity, which are respectively the projection of the vehicle's position and velocity on the obstacle cell's boundary.

Figure 9 depicts three different types of obstacle points which arise from this projection. Face points are the exact orthogonal projection of the quadrotor's x_q, y_q, z_q position on the obstacle's face along the x, y or z direction. A face point will therefore share two of its grid coordinates with the quadrotor and follow its movements across the face of the cell it is currently projected on. Edge points are placed on the closest edges of the cells which share exactly one coordinate (x, y or z) with the vehicle's cell. In other words, edge points are the projections of a face point onto the edges of that face. They are also assigned a velocity but this time restricted to the single direction of the edge it is on. Finally, vertex points lie on the vertices of the closest obstacle cell. They do not share any grid coordinates with the vehicle and their velocity is fixed to zero. It is important to notice that the nature of the obstacle points will change as the quadrotor moves through the environment, and that the transition from one kind of point to the other is continuous. At the point when the vehicle enters a neighbouring

cell, a vertex point of the previous cell may become an edge point placed at the vertex of the new cell, and vice versa. Similarly, a surface point can transition to an edge point or a vertex point while the vehicle changes cell.

Once reduced to their point equivalents, the obstacles are passed to the DRCA algorithm as point objects with radius $R_j = 0$, a given velocity, and a maximum priority. By assigning obstacle points a position and a velocity, the DRCA process implicitly sees the obstacle points like any other agent and guarantees that the quadrotor will not steer towards them. Giving them a maximum priority value ensures that if a vehicle reaches a conflict with its obstacle point, it will not expect the point to help in the deconfliction manoeuvre. For instance, when travelling next to a wall, the face obstacle point replicates the vehicle's velocity parallel to it, which leads the DRCA never to attempt circumventing the wall point. Instead the vehicle will fly parallel to it effectively never approaching the wall in the deconfliction maintenance phase, as doing so would immediately result in a conflict. For example this can be seen in Figure 9. As the quadrotor is approaching the wall, the green vector pointing to the left face point is indicating a conflict with this obstacle cell.

This way of modelling the obstacles and inserting them directly in the DRCA architecture increases the amount of computations by a number m (for m obstacles). However, the limited horizon γ introduced in equation (5) also limits the number of obstacle cells considered by the algorithm to m_l the same way it limits the amount of quadrotors considered to n_l. This helps maintaining a linear-time performance of $O(m_l + n_l)$ (see Figure 10).

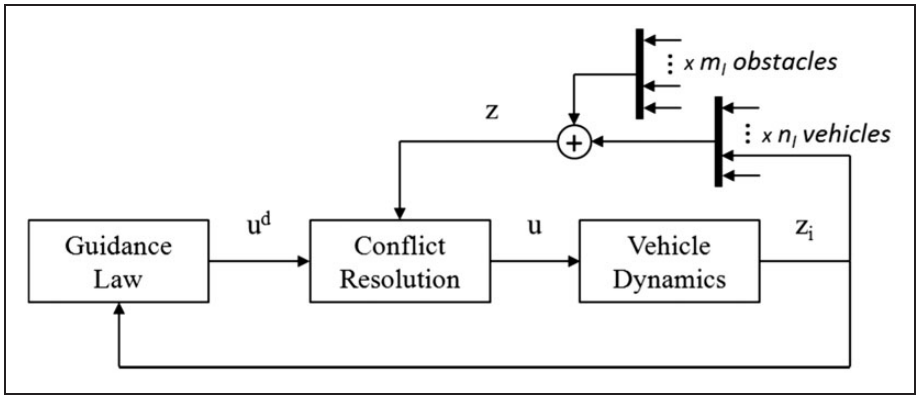

Figure 10. Updated inputs of the conflict resolution block.

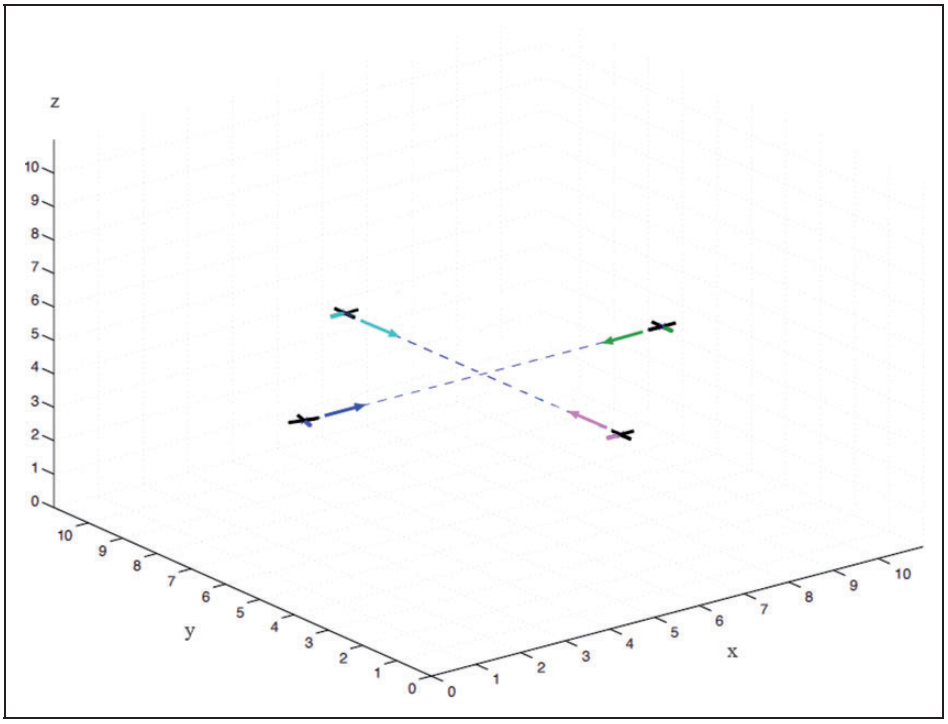

Figure 11. Starting conditions for the two crossing scenarios.

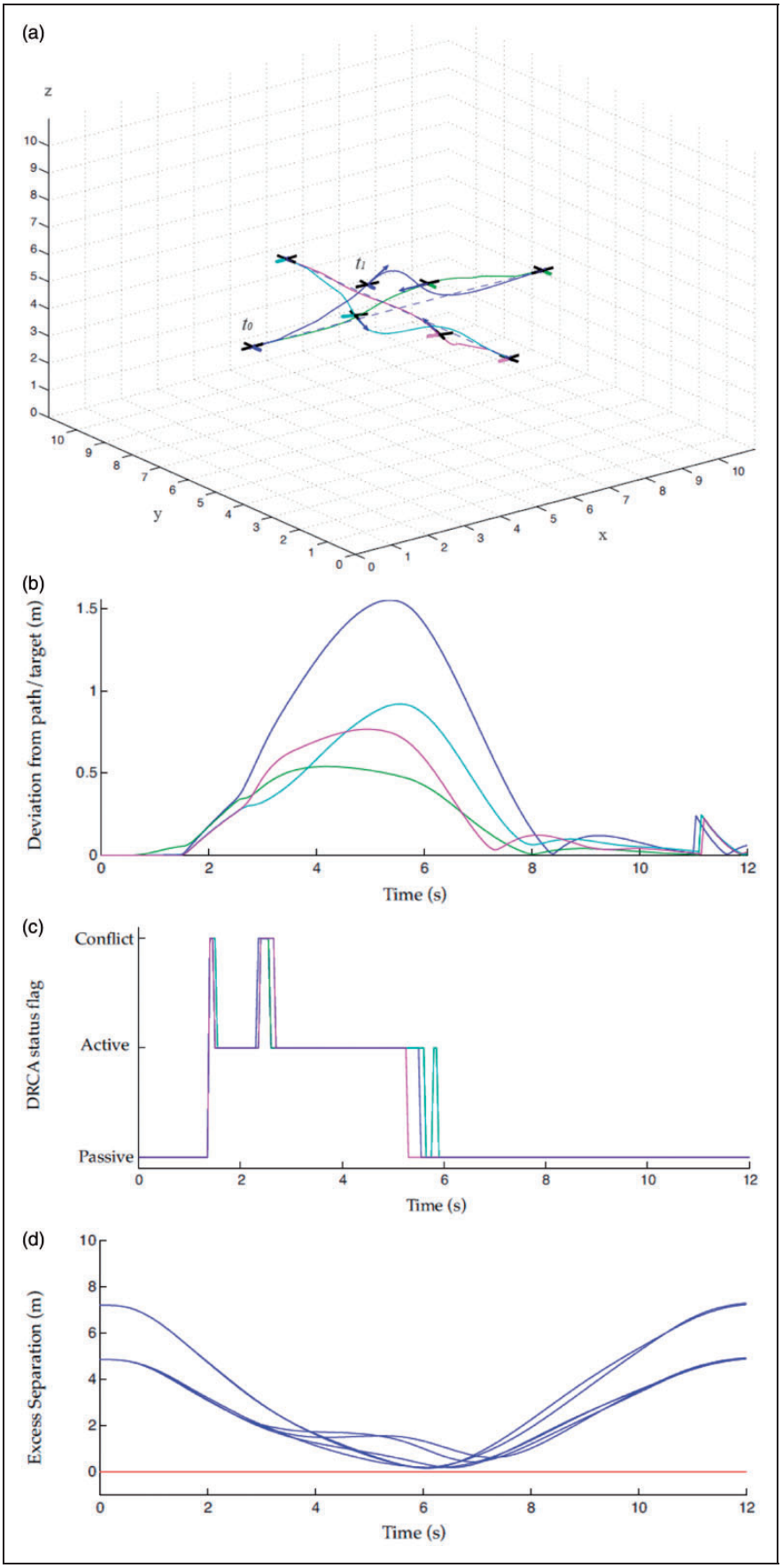

Figure 12. Four vehicles deconfliction manoeuvre with equal priority levels: (a) deconfliction paths (t_0=0 s, t_1=6 s); (b) deviation from nominal path; (c) DRCA status; (d) inter-agent excess separation $d_{exc} = r - d_{sep}$.

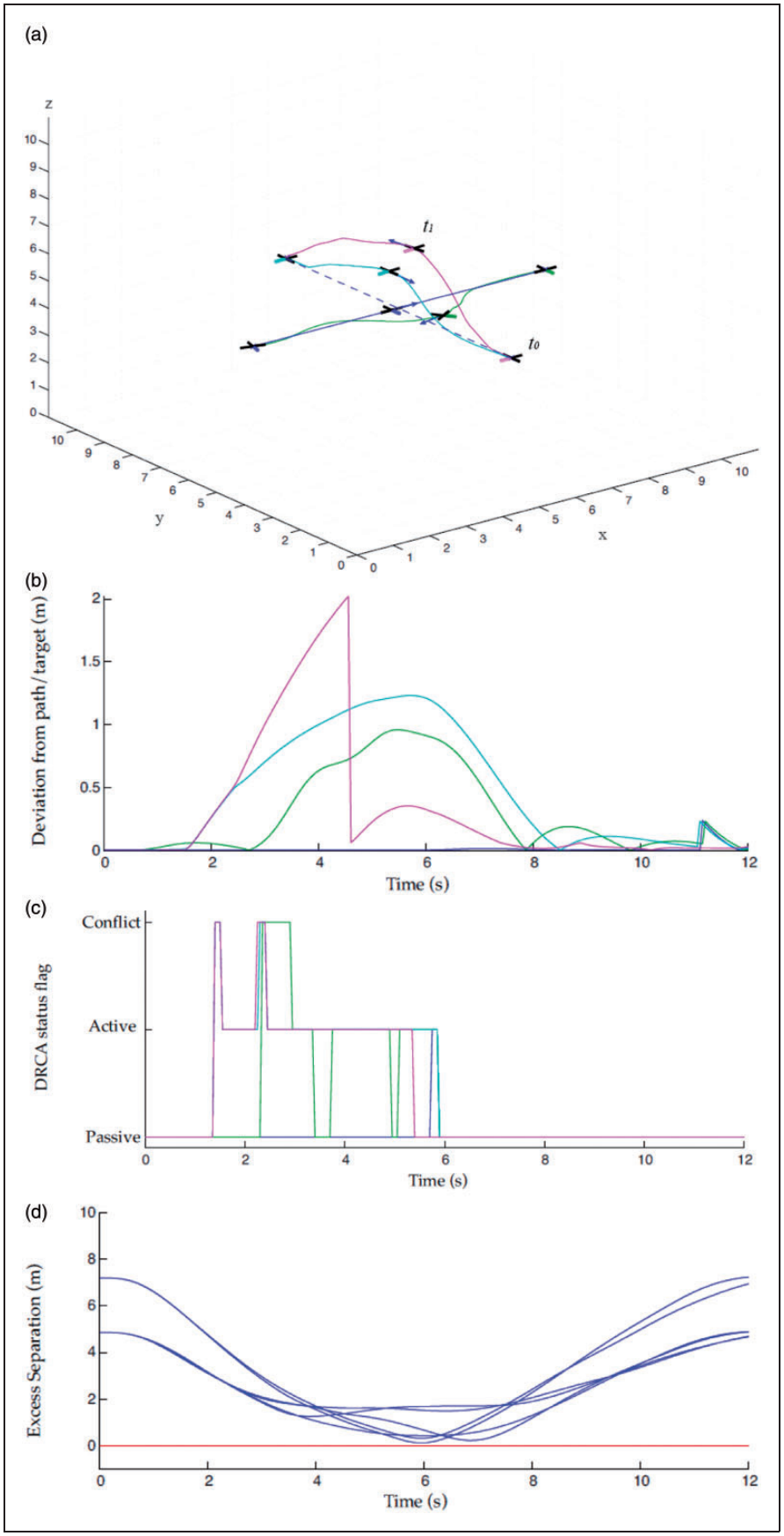

Figure 13. Four vehicles deconfliction manoeuvre with $P_{blue} > P_{green} > P_{cyan} > P_{magenta}$: (a) deconfliction paths (t_0=0 s, t_1=6 s); (b) deviation from nominal path; (c) DRCA status; (d) inter-agent excess separation $d_{exc} = r - d_{sep}$.

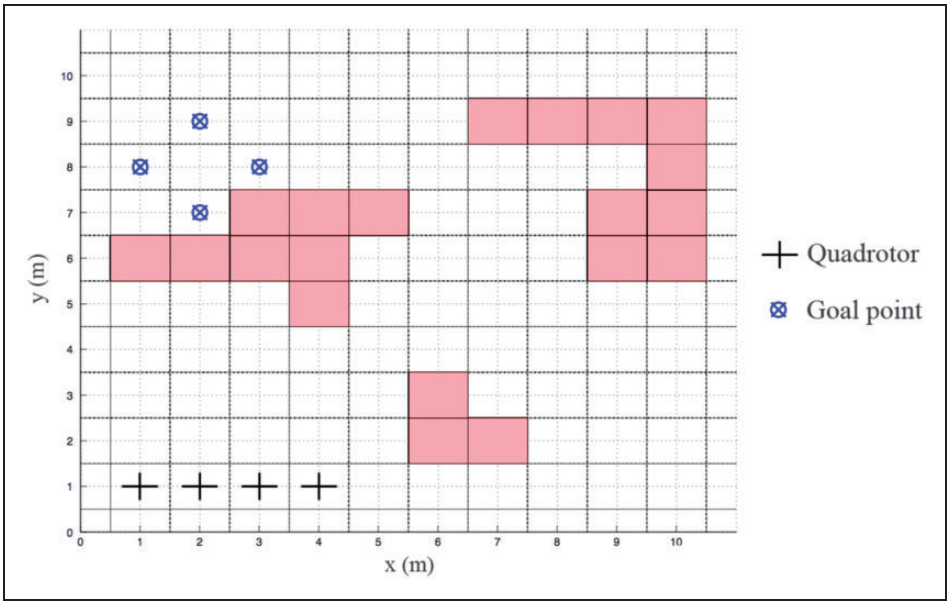

Figure 14. Top view of the initial conditions for the multi-vehicle mission.

Near hover

A significant side-effect of an obstacle filled environment is the limited space available to the units to move around. It may come a time where two or more quadrotors are therefore required to hover at positions relatively close to each other or to one or several obstacles, namely within the minimum horizon γ_{min} defined in equation (5). Although the desired velocity of a hovering vehicle is zero, sensor noises, actuator imperfections or external factors always create small uncertainties resulting in small relative velocities. Once injected into the DRCA, these movements are interpreted as potential conflicts and push the vehicles involved to continuously perform unnecessary avoidance actions. Turning the collision avoidance completely off is not desirable as the hovering agents might still need to avoid an incoming vehicle. It is therefore necessary to define a set of admissible conditions for the DRCA to discard these particular interactions, effectively introducing a safe proximity filter based on relative position and velocity. During the conflict detection step, an interaction will be discarded even when inside the DRCA horizon γ if all of the following rules hold

- The vehicle's own velocity is smaller than a certain threshold value v_{hov} – this allows small deviations from hover.
- The interacting vehicle or obstacle is at a distance greater than a minimum safety margin γ_F – this value represents the minimum allowable distance between agents for safe operation.
- The relative approach velocity is no greater than a threshold value v_F – this allows the filter to stay active when the deviations from the hover behaviour become significant.

Performance analysis

The scenarios investigated here highlight the improvements made to the original DRCA. All simulations are run using a three-dimensional environment and a full high-order dynamic model of the quadrotor vehicle.

Crossing scenario

Four quadrotors are placed at each corner of a square and attempt to simultaneously reach the diagonally opposite corner, forcing the initial layout to be coplanar (Figure 11). In the first instance, all four vehicles have the same priority level meaning that the deconfliction responsibilities are expected to be equally shared between all vehicles. In Figure 12(a), the vehicles travelling right and left (blue and green) fly upwards, whereas the ones travelling towards and away from the observer (cyan and magenta) descend slightly thus limiting the deviation from the path of each vehicle and generating avoidance trajectory as smooth as possible (Figure 12(b)). Due to the fact that all vehicles have the same speed and priority level, Figure 12(c) shows that all vehicles react to the conflicts at similar times and in similar fashion. It is important to note that the minimum separation distance is approached but never violated (Figure 12(d)). The algorithm therefore generated safe avoidance trajectories without making the clearance distances greater than they needed to be for strict safety. The result is a near-optimal solution to the three-dimensional collision problem.

The same initial scenario was run but this time with the introduction of different priority levels. The blue vehicle travelling from left to right, the green one from

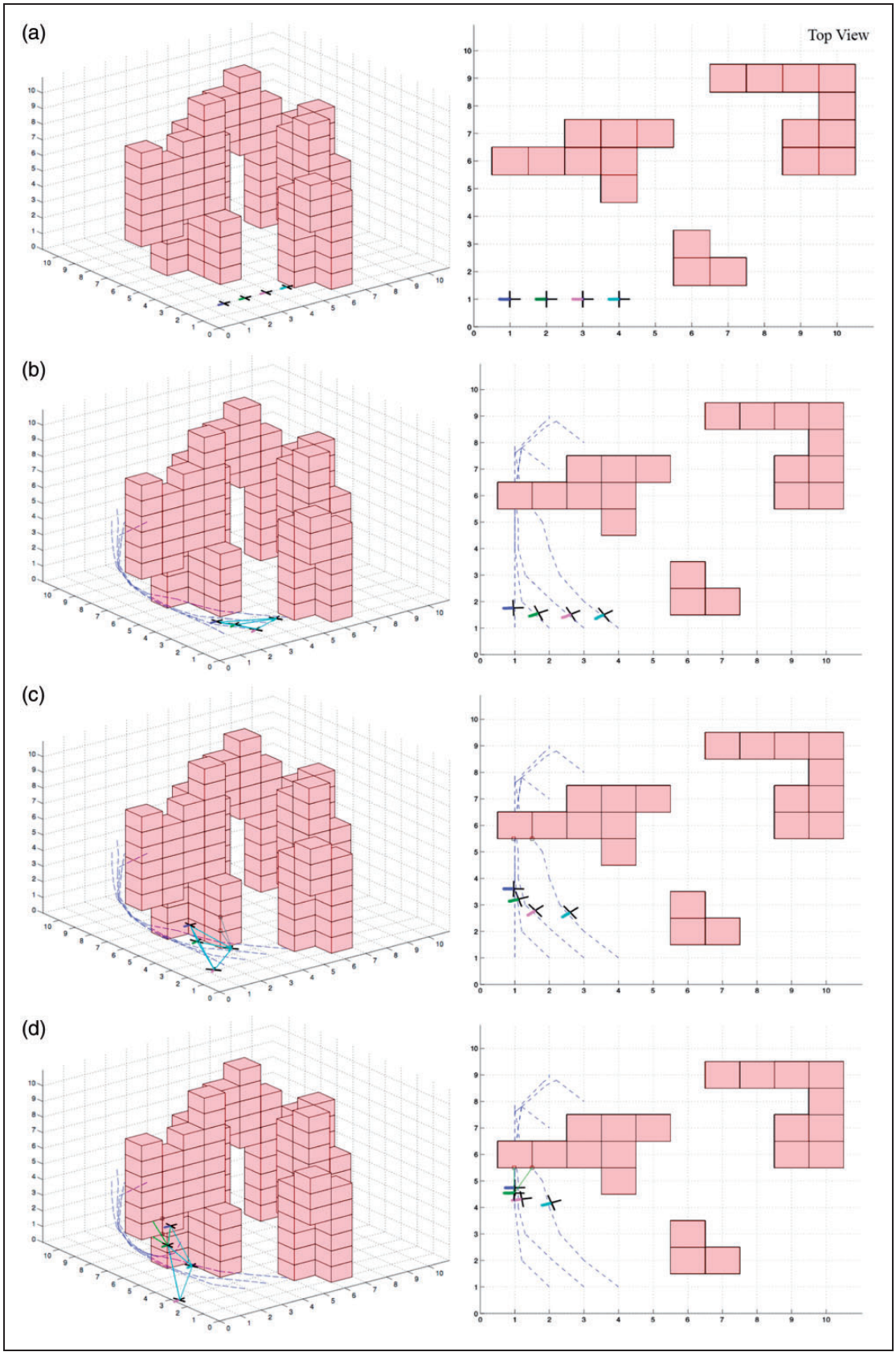

Figure 15. Sequential depiction of the quadrotor swarm's behaviour carrying out a mission in a densely populated obstacle environment: (a) time = 2 s: take off; (b) time = 4 s: separation and path planning; (c) time = 6 s: the DRCA begins acting; (d) time = 9 s: first conflicts are detected; (e) time = 12 s: successful deconfliction manoeuvre; (f) time = 15 s: begin the ascent; (g) time = 18 s: one vehicle is kept behind; (h) time = 20 s: two targets reached; (i) time = 22 s: two vehicles hovering; (j) time = 25 s: targets reached.

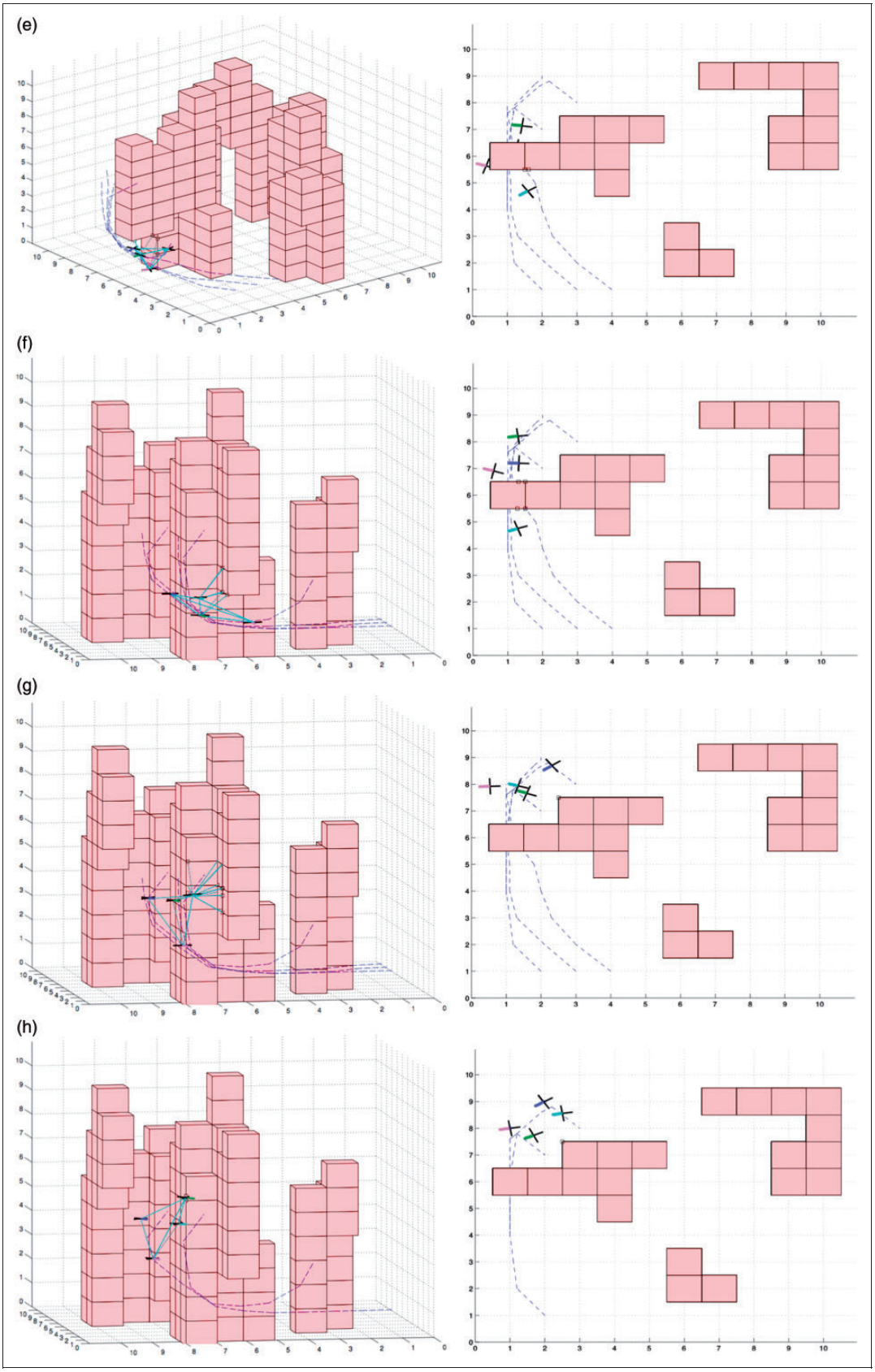

Figure 15. Continued.

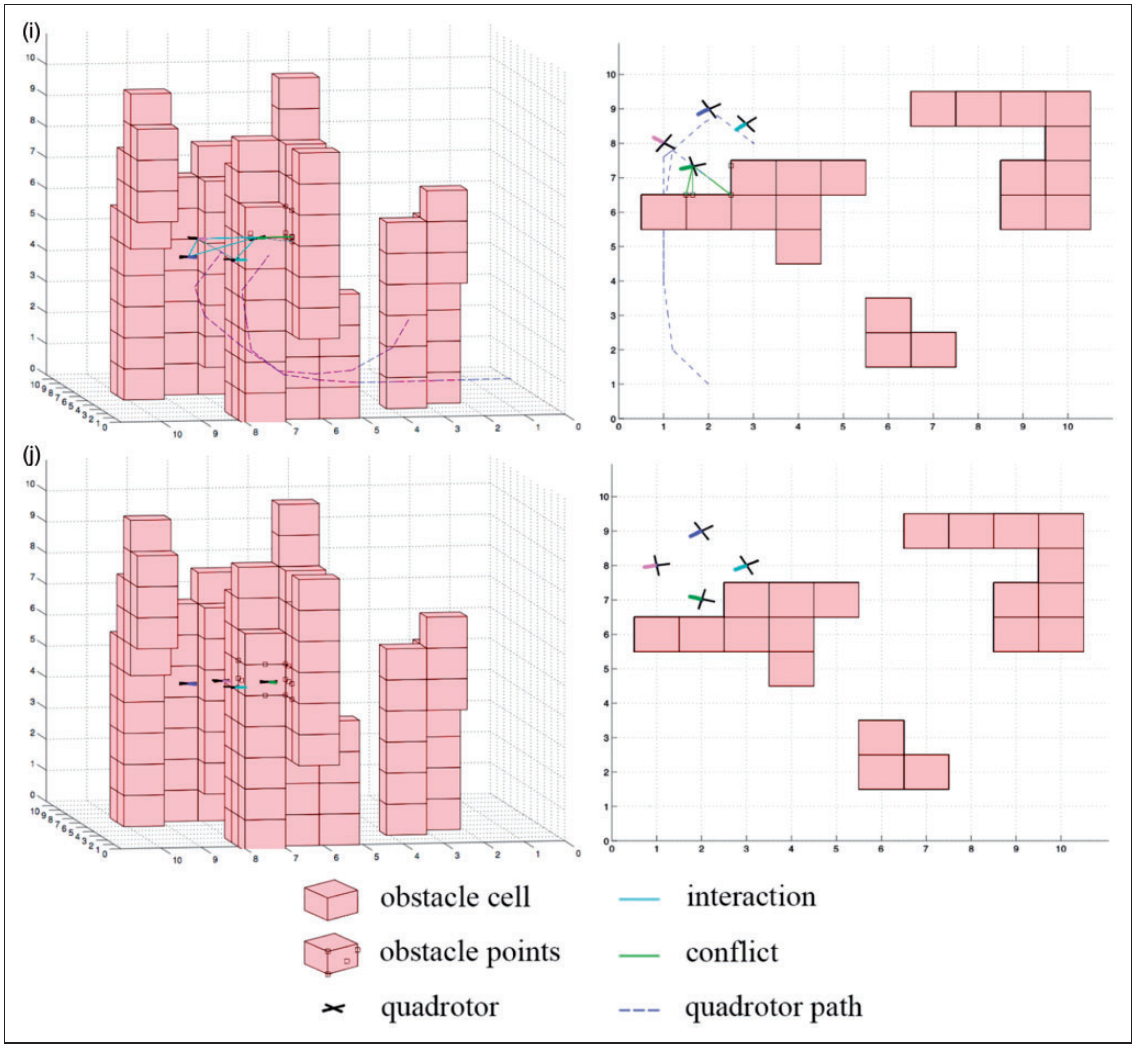

Figure 15. Continued.

right to left, the cyan one from top to bottom and the magenta one from bottom to top, the priority levels are fixed so that $P_{blue} > P_{green} > P_{cyan} > P_{magenta}$. Figure 13(a) shows that the two vehicles with the lowest priorities are forced out of the common plane so that the two higher priority vehicles keep their original altitude throughout the flight. On this plane, the green quadrotor then gives way to the blue one, effectively leaving it unaffected by the conflict. Figure 13(c) shows that the DRCA status of the blue vehicle quickly switches to active at $t = 5.9$ s without ever reaching a conflict status, keeping it on its nominal path at all times. The limited horizon implemented for computational lightness, restricts the detection and resolution scheme to a decentralised method rather than distributed, as only nearby vehicles are considered by the DRCA. The lower priority quadrotors (cyan and magenta) are therefore not aware of each other at the time $t = 1.5$ s when they detect the other two (Figure 13(c)) and both break off the conflict plane in an upward direction. At $t = 2.2$ s, they enter in a conflict with each other and the magenta vehicle, with the lowest priority level, climbs further up. When its cross-track error increases up to a given deviation threshold of $D_{tresh} = 2m$ at $t = 4.5$ s, the magenta vehicle's recalculates its path to the target from its current position. This drops the deviation down to zero (Figure 13(b)), which consequently lowers the controller action on the vehicle to push it back towards its nominal path. Once again, the minimum separation distance is closely approached but never crossed (Figure 13(d)).

Dense obstacle scenario

The scenario simulated here evaluates the collision avoidance system's ability to provide safe navigation in a highly constrained environment. The aim is to understand the impact of the algorithm's enhancements on the overall performances of the avoidance scheme so the scenario is purposely set-up beyond the scope of the DRCA's original implementation. Four quadrotors are dispatched in a $10 \times 10 \times 10$ m^3 environment and assigned to four different goal locations

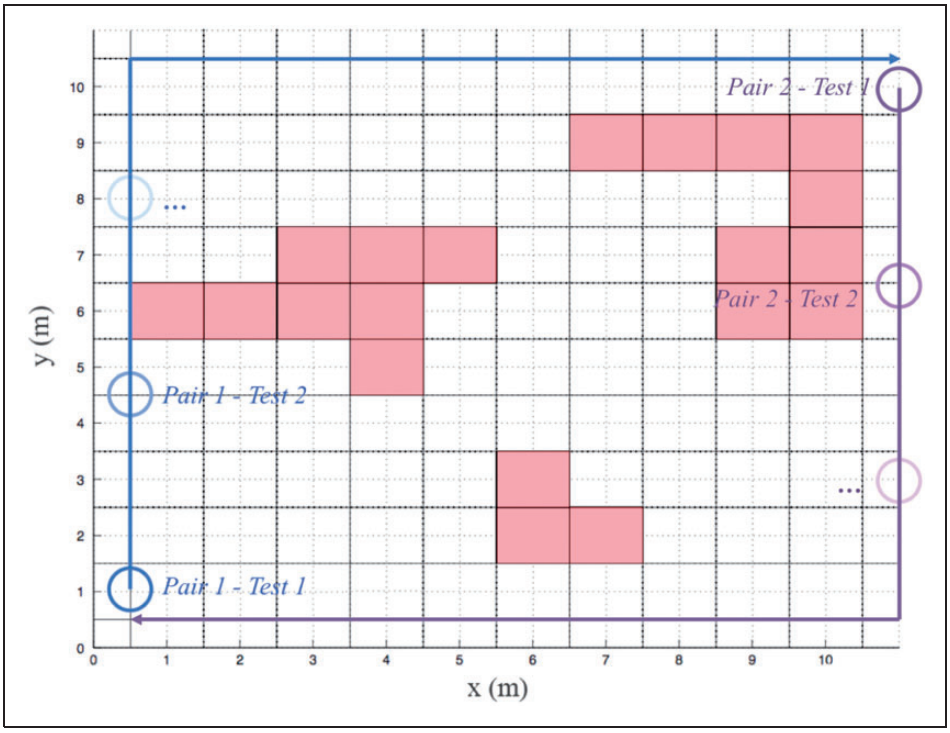

Figure 16. Initial positions of the quadrotors used in the performance study of the final DRCA.

Table 1. Time comparison of the vehicles crossing paths with and without distributed reactive collision avoidance (DRCA).

Time to completion (s)		
With DRCA	Without DRCA	Difference (%)
29.3	28.3	3.4
24.2	23.8	1.6
25.1	24.0	4.4
29.7	27.2	8.4
22.6	21.2	6.2
23.3	21.7	6.9

at which the vehicles are expected to hold a stable hover. These target positions are located close to each other and in the vicinity of numerous obstacle cells. To successfully complete the mission, the quadrotors must reach the target points as quickly and efficiently as possible without compromising their safety. It is assumed that the size and location of all obstacles in the area is known to the vehicles and represented as a cubic-occupied cell.

Figure 14 shows a top view of the obstacle layout of the area (red cells), the initial position of vehicles (black crosses) and the locations of their respective targets (in blue).

A sequential depiction of the system's behaviour throughout the completion of the task is presented in Figure 15(a) to (j). The four quadrotors initially take off simultaneously (a) and compute a conflict-free optimal path to their respective targets (b), represented by the dark blue dash lines on the figures. Due to their initial proximity, the inter-vehicle avoidance action activates early and forces the quadrotors to deviate from the nominal obstacle-free paths (c) and closer to the obstacles. When conflicts are detected with the obstacle wall (d), the DRCA initiates a three-dimensional deconfliction manoeuvre based on closest escape from the collision cone and the virtual conflict repulsive force. The new manoeuvre allows the vehicles to find a way around the obstacle wall which utilises the entire available space and avoids unnecessary detours (e). Once the obstacle is cleared, the quadrotors begin to ascend towards their targets while keeping track of their relative motion (f) and the surrounding obstacles. Being in a more confined space, the DRCA has to respond to a greater number of potential threats. Hence, it limits the climbing speed of one of the agents (g) to give more space to the others and let them reach their goal point more rapidly. Two of the targets being further away from the obstacle wall than the rest, the quadrotors assigned to them have less potential conflicts to consider and manage to reach their targets first (h). At instant (i), these two vehicles are now in a stable hover, which automatically gives them a higher priority level in deconfliction maintenance. Meanwhile, the other two quadrotors slowly converge towards their destination while trying to stay as parallel to the wall as possible to avoid a repulsive action from the DRCA. Only 25 s into the mission, all four quadrotors have reached their desired goal locations (j). The close proximity filter blocks any undesirable DRCA action and keeps the vehicles in a stable hover.

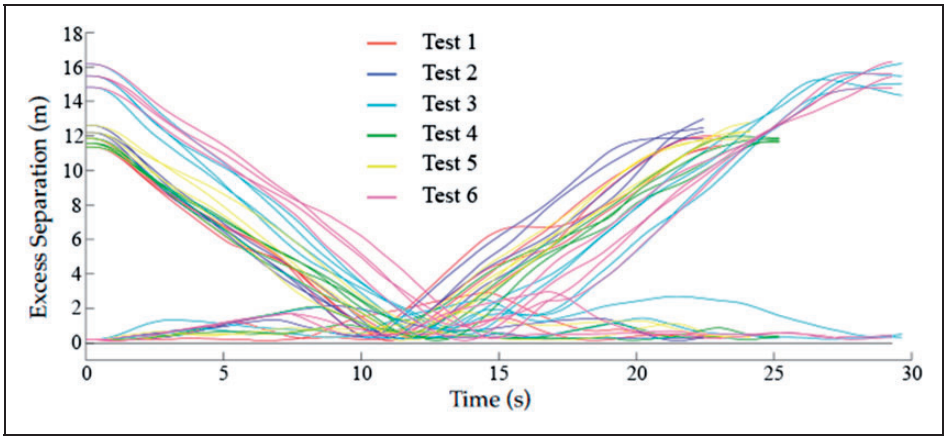

Figure 17. Comparative of the inter-agent excess separation $d_{exc} = r - d_{sep}$ for the six test runs.

Note that at instant (c) the deconfliction manoeuvres pushes one of the quadrotors particularly far from its optimal trajectory. Due to the conflicting actions between the vehicle's guidance law trying to bring it back on track and the DRCA forcing it away from the conflicts, the vehicle's flight behaviour becomes erratic. Similarly to the previous crossing scenario, it is therefore more efficient for the quadrotor to calculate a new shortest path to the target (d) from its current position.

The simulation results show that the DRCA can efficiently perform in a dense obstacle environment when implemented on platforms with sufficient manoeuvrability such as quadrotors. The degenerate coplanar conflict resolution provides quicker and more space-optimal deconfliction manoeuvres by using the three available dimensions. The simulation also underlines the importance of an accurate obstacle cell modelling technique to achieve an optimal use of the surrounding space as the targets assigned to the quadrotors in this example would be impossible to reach with a modelling approach based on spherical approximations.

Colliding pairs

The performances of the improved DRCA are confirmed through a series of crossing tests in the crowded environment presented in Figure 15. Four quadrotors are placed in two pairs on each side of the map and sent on a crossing pattern towards the initial positions of the opposing pair. The DRCA running on each unit therefore has to handle the close proximity with the quadrotor in the same pair, the head-on collision course with the other two vehicles, and the various fixed obstacles in the way. The simulation was run several times with different initial positions, as shown in Figure 16.

Six crossing scenarios were thus created and simulated, first with the DRCA active and then without any avoidance. In the second run, the units would actually collide but the test was performed to calculate the time needed to travel the shortest paths to target. These times are then compared in Table 1 with the times needed to run the same scenarios with the DRCA activated. The comparison shows that although the DRCA manoeuvres create deviations from the original path, the ensued time delay remains within 9% of the optimal flight time. The deviation is therefore minimal and the inter-agent excess separation distances shown in Figure 17 confirms it. The collision courses between opposing units crossing in the middle of the map are represented by the curves starting and finishing high, with a dip in the centre.

On the other hand the vehicles in each pair remain close to one another during the entire flight thus creating the flat curves near the zero excess separation line. Figure 17 shows that both types of curves get close to the zero line without ever crossing it as to ensure that the vehicles remain collision-free in all circumstances, while minimising the magnitude of their avoidance manoeuvres and keeping the units close to their optimal trajectories.

Discussion and conclusions

The appeal of the DRCA is its guarantee that as long as a conflict-free space can be reached, the multi-vehicle system will remain conflict-free at all times. Although this guarantee was made for second-order dynamics, it would be possible to compute a mathematical proof of safety accounting for the quadrotor's fourth-order dynamics. This proof would show that for every conflict with an initial relative velocity of magnitude v, where the agents involved are non-adversarial, there exists a threshold initial separation $d_0 = f(v, k, u_{max}, v_{max})$ such that if the initial distance between both agents is $r > d_0$ then a collision will always be averted by executing the combined deconfliction manoeuvre described in the paper. The computation of the exact value d_0 is beyond the scope of this work but by overestimating it in the generation of the limited horizon γ, it can be postulated that the

DRCA guarantee holds true. Therefore, the improved DRCA has proven itself to be a reliable and robust collision avoidance algorithm for a swarm of high-order dynamics vehicles in complex environments. The addition of priority levels and a three-dimensional avoidance manoeuvre improves on the performances of the original DRCA, generating quicker and less intricate trajectories. Furthermore, the obstacle avoidance and near-hover filter allow the algorithm to ensure collision-free cooperative flights, even in extreme conditions, with the potential to be used in vehicle systems of widely different nature and complexities.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

References

1. Kuchar JK and Yang LC. A review of conflict detection and resolution modeling methods. *IEEE Trans Intell Transport Syst* 2000; 1: 179–189.
2. Minimum performance standards-airborne ground proximity warning equipment. Document No. RTCA/DO-161A, May 1976.
3. Minimum performance specifications for TCAS airborne equipment. Document No. RTCA/DO-185, September 1983.
4. Shin H, Tsourdos A and White B. UAS conflict detection and resolution using differential geometry concepts. In: Sense and avoid in *UAS: Research and applications*. 2012. DOI: 10.1002/9781119964049.ch7.
5. Waller M and Scanlon C. NASA workshop on flight deck centered parallel runway approaches in instrument meteorological conditions. In: *NASA conference publication 10191*, December 1996.
6. Melega M, Lazarus S, Lone M, et al. Autonomous sense & avoid capabilities based on aircraft performances estimation. *Proc IMechE, Part G: J Aerospace Engineering* 2014; 228: 492–517.
7. Melega M, Lazarus S, Lone M, et al. Multiple threats sense and avoid algorithm for static and dynamic obstacles. *J Intell Robot Syst* 2014; 77: 215–228.
8. Pallottino L, Scordio VG, Bicchi A, et al. Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Trans Robot* 2007; 23: 1170–1183.
9. Hwang I and Tomlin C. Protocol-based conflict resolution for air traffic control (tech. rep. sudaar-762). Technical report, Stanford University, 2002.
10. Burgard W, Thrun S and Fox D. The dynamic window approach to collision avoidance. *IEEE Robot Autom Magaz* 1997; 4: 23–33.
11. Padhi R and Chawla C. Partially integrated guidance and control of unmanned aerial vehicles for reactive obstacle avoidance. *Mobile Intell Autonom Syst* 2012; 20: 357–383.
12. Lalish E. *Distributed reactive collision avoidance for multivehicle systems*. PhD Thesis, University of Washington, USA, 2009.
13. Lalish E and Morgansen KA. Distributed reactive collision avoidance. *Autonom Robots* 2012; 32: 207–226.
14. Morgansen KA and Tsukamaki T. Decentralized reactive collision avoidance for multiple unicycle-type vehicles. In: *Proceedings of the IEEE American control conference*, Seattle, WA, USA, 2008.
15. Lalish E and Morgansen KA. Decentralized reactive collision avoidance for multivehicle systems. In: *Proceedings of the 47th IEEE conference on decision and control*, Cancun, Mexico, 2008.
16. Lalish E and Morgansen KA. Distributed reactive collision avoidance for multivehicle systems. 2009.
17. Melander A, Powel N, Lalish E, et al. Implementation of deconfliction in multivehicle autonomous systems. In: *Proceedings of the 27th international congress of the aeronautical sciences*, 2010.
18. Boardman B, Hedrick T, Theriault D, et al. Distributed reactive collision avoidance for multivehicle systems. In: *Proceedings of the American control conference*, 2013.
19. Anderson E. *Quadrotor implementation of the three-dimensional distributed reactive collision avoidance algorithm*. Master's Thesis, University of Washington, UK, 2011.
20. Fiorini P and Shiller Z. Motion planning in dynamic environments using the relative velocity paradigm. In: *Proceedings of the IEEE international conference on robotics and automation*, 1993.
21. Chakravarthy A and Ghose D. Obstacle avoidance in a dynamic environment: A collision cone approach. *IEEE Trans Syst Man Cybernet A: Syst Humans* 1998; 28: 562–574.
22. Bresciani T. *Modelling, identification and control of a quadrotor helicopter*. Master's Thesis, Lund University, Sweden, 2008.
23. Bouabdallah S. *Design and control of quadrotors with application to autonomous flying*. PhD Thesis, Ecole Polytechnique Federale de Lausanne, Switzerland, 2007.
24. Leonard J, Savvaris A and Tsourdos A. Energy management in swarm of unmanned aerial vehicles. *J Intell Robot Syst* 2013; 74: 233–250.
25. Grisetti G, Burgard W and Grzonka S. A fully autonomous indoor quadrotor. *IEEE Trans Robot* 2012; 28: 90–100.
26. Richter C, Bry A and Roy N. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: *Proceedings of the international symposium on robotics research (ISRR)*, 2013.