IMPLEMENTATION OF DECONFLICTION IN MULTIVEHICLE AUTONOMOUS SYSTEMS

A. P. Melander*, N. D. Powel*, E. Lalish*, K. A. Morgansen*, J. S. Jang and J. Vian****
***Department of Aeronautics and Astronautics, University of Washington, Seattle, WA, 98195-2400; ****The Boeing Company, Seattle, WA, 98124**

Keywords: *Deconfliction, multivehicle control, autonomous systems*

Abstract

The focus of the work in this paper is the implementation of a distributed deconfliction algorithm for vehicles with constraints on velocity and acceleration. The algorithm was tested on two testbeds: a quadrotor system and an underwater vehicle system. The quadrotor system has vehicles that can hover in place and move in arbitrary directions, while the vehicles in the underwater system move with a constant forward velocity and can only change direction by changing heading. Results are demonstrated for a variety of numbers of vehicles as well as types of collision directions.

1 Introduction

As multi-vehicle autonomous systems are studied and implemented, the issue of conflict resolution becomes increasingly important. From mobile robots performing a cooperative search to air traffic control for unmanned aerial vehicles, collision avoidance is of utmost importance for safety. Much of the work so far on collision avoidance has been sponsored by the FAA to support a potential move to free-flight air traffic control [13], [7], whereby aircraft can avoid each other in a decentralized manner rather than relying on a land-based controller. Similar concepts have been discussed regarding autonomous harbor control for ships [11], [6]. These scenarios will become more important as unmanned vehicles are introduced, because safety will need to be guaranteed for them to be accepted by their manned counterparts.

Many conflict resolution strategies have been proposed using varying degrees of automation. Some of these strategies are designed specifically for Air Traffic Control (ATC) applications, while others are more general in nature. An overview and classification of many collision avoidance algorithms can be found in [8]. The Distributed Reactive Collision Avoidance (DRCA) algorithm developed in [9] is a conflict resolution scheme guaranteeing collision avoidance, either in the plane or in 3D, for an arbitrary number, n, of nonholonomic vehicles. The DRCA algorithm is designed to work with vehicles that have limited control authority and complex dynamics (such as aircraft, which have low acceleration compared to speed and must bank to turn). According to the collision avoidance classification in [8], DRCA is a nominal (projects current states into the future along a single trajectory), horizontal plane, global algorithm that combines turning and speed changes and detects conflict. It is most closely associated with a force field approach, although it does not strictly meet this definition because DRCA does not treat vehicles as charged particles.

An imperative feature for any avoidance algorithm used for automating air or shipping traffic must be a guarantee of collision avoidance. For a collision avoidance guarantee to be valid for real vehicles, it must also restrict the maximum acceleration. Any collision avoidance scheme that does not meet this criteria can be ruled out for

such applications. Many of the approaches previously proposed guarantee avoidance, but only for a limited number of vehicles [15], [2]. Because traffic in the air and on the sea is rapidly increasing, a higher likelihood exists for a collision (or at least a conflict) involving multiple vehicles. The DRCA algorithm is ideal for such applications because it guarantees collision avoidance (including a restriction on maximum acceleration) for an arbitrary number of vehicles simultaneously.

Furthermore, a centralized avoidance scheme should be avoided because of the high computational load required by the central node, lack of robustness (what happens if it breaks?), and inability to respond quickly to emergency situations. The DRCA algorithm distributes computation among the entire group because each vehicle accounts only for its own interactions. This distribution makes for *O(n)* calculations on each vehicle, which should be reasonable in most applications. It is not a centralized algorithm, but it is not strictly decentralized since states of all other vehicles are required (not just the nearest neighbors). Since it is not centralized or decentralized, the term "distributed" is used in naming the algorithm.

The focus of the work in this paper is to proceed beyond the theoretical and simulation work developed in [9] by implementing the DRCA algorithm on actual hardware. The remainder of the paper is organized as follows to present the implementation. In Section 2, system modeling is discussed, and the deconfliction algorithm is presented. The two testbeds being utilized for implementation are discussed in Section 3. Results of the deconfliction implementation are given in Section 4, and conclusions are given in Section 5.

2 Decentralized Reactive Collision Avoidance

2.1 System Model

The work here presents implementation of a method for deconflicting n vehicles. The notation throughout this paper will use bold face for vectors, hats over unit-vectors, script capital letters for sets, standard capital letters for matrices and functions, and everything else is assumed scalar. Quantities subscripted with t, n, or b refer to the tangent, normal, or binormal direction, respectively.

Each vehicle has a nominal desired control input, $\mathbf{u}_d(t)$, which comes from an arbitrary outer-loop controller. This controller is designed for the vehicle to perform a desired task, which could range from target tracking to waypoint navigation, area searching, etc. The goal of the DRCA algorithm is to adjust the control input on each vehicle to guarantee collision avoidance while simultaneously staying close to the desired control input (keeping in mind that this desired control can change with time).

For this approach to collision avoidance, the only vehicle states that matter are position and velocity. Orientations affect performance, as they often have bearing on the magnitude of acceleration available in a particular direction, but they do not directly affect the underlying features of conflict and collision. In this way, many different vehicle models work equivalently with this approach. To simplify the math, a simple vehicle model will be used for most of the following analysis: a 3D double integrator, which for the i^{th} vehicle is given by

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} \mathbf{r}_i \\ \mathbf{v}_i \end{bmatrix} &= \begin{bmatrix} \mathbf{v}_i \\ \mathbf{u}_i \end{bmatrix} \\ \frac{d}{dt}\Theta_i &= \Omega_i \Theta_i, \end{aligned} \tag{1}$$

where $\mathbf{r}, \mathbf{v} \in \mathbb{R}^3$ are the position and velocity of the vehicle center of mass, and $\mathbf{u} \in \mathbb{R}^3$ is the control input. The matrix $\Theta = [\hat{\mathbf{t}}, \hat{\mathbf{n}}, \hat{\mathbf{b}}]$ defines the orientation of a body-fixed coordinate frame located at the vehicle center of mass relative to an inertial coordinate frame, and Ω is the cross product matrix of the body rotation vector $\boldsymbol{\omega} = [\omega_t, \omega_n, \omega_b]^{\mathsf{T}}$. Note that the orientation of the vehicle (defined by the $\hat{\mathbf{t}}$, $\hat{\mathbf{n}}$, and $\hat{\mathbf{b}}$ vectors) is only used as a local coordinate frame for the DRCA algorithm. The orientation does not directly affect the dynamics ($\mathbf{r}$ and $\mathbf{v}$), and as such can be arbitrary. However, many vehicle's input constraints are related

to their orientation, and so it can be useful to tie this local coordinate frame to the actual body coordinates of the vehicle.

We constrain the input by use of an arbitrarily varying constraint set, $\mathbf{u}_i \in \mathcal{C}_i$. The only requirement is that $\mathcal{C}_i$ always contain the origin. A simple example of an input constraint set that limits maximum acceleration and velocity is

$$\mathcal{C}_i = \Big\{ \mathbf{u}_i \in \mathbb{R}^3 \Big| \, \|\mathbf{u}_i\| \le u_{max}, \\ \|\mathbf{v}_i\| \ge v_{max} \implies \mathbf{u}_i^\mathsf{T} \mathbf{v}_i \le 0 \Big\}. \quad (2)$$

For the DRCA algorithm, one must choose a set of rectangular constraints $\mathcal{R}$ (which can also vary with time, state, etc.) for each vehicle that encloses its $\mathcal{C}$, as well as a corresponding saturation function, $S : \mathcal{R} \to \mathcal{C}$. The function S must be continuous, must become the identity map for any $\mathbf{u} \in \mathcal{C}$, and must preserve the sign of each component of $\mathbf{u}$ when decomposed in the $\hat{\mathbf{t}}$, $\hat{\mathbf{n}}$, and $\hat{\mathbf{b}}$ directions. In this example, one can choose

$$\mathcal{R}_i = \left\{ \mathbf{u}_i \in \mathbb{R}^3 \Big| \, -u_{max_i} \le u_{t_i} \le u_{max_i}, \dots \right\}, \quad (3)$$

and

$$S_i = \begin{cases} \mathbf{u}_i \frac{u_{max}}{\|\mathbf{u}_i\|}, & \|\mathbf{u}_i\| > u_{max} \\ \mathbf{u}_i - \frac{\mathbf{v}_i \mathbf{u}_i^\mathsf{T} \mathbf{v}_i}{v_{max}}, & \|\mathbf{v}_i\| \ge v_{max}, \mathbf{u}_i^\mathsf{T} \mathbf{v}_i \ge 0 \\ \mathbf{u}_i, & \text{otherwise.} \end{cases} \quad (4)$$

An example of how more complex vehicle dynamics can be represented by this simple model with an appropriate choice of input constraint set follows. Let us model a vehicle which can move forward with variable speed and turn in two axes (a 3D unicycle model) with limits on its turn rate, forward acceleration, and maximum speed. One way to describe the model mathematically is by

$$\frac{d}{dt} \begin{bmatrix} \mathbf{r} \\ s \end{bmatrix} = \begin{bmatrix} s\hat{\mathbf{t}} \\ u_a \end{bmatrix}, \\ \frac{d}{dt}\Theta = \Omega\Theta,$$

where $|u_a| \le u_{a_{max}}$, $|\omega_n| \le \omega_{n_{max}}$, $|\omega_b| \le \omega_{b_{max}}$, and $|s| \ge s_{max} \implies u_a s \le 0$. Alternatively, an equivalent representation of the system is (1) with $\mathbf{u} = u_a\hat{\mathbf{t}} + \|\mathbf{v}\|\,\omega_b\hat{\mathbf{n}} - \|\mathbf{v}\|\,\omega_n\hat{\mathbf{b}}$. The tangent vector must be initialized to the same direction as the velocity vector, but the dynamics will keep the two vectors aligned from then on. In this case, $\mathcal{R}$ can be defined by

$$u_{t_{max}} = -u_{t_{min}} = u_{a_{max}} \\ u_{n_{max}} = -u_{n_{min}} = \|\mathbf{v}\|\,\omega_{b_{max}} \\ u_{b_{max}} = -u_{b_{min}} = \|\mathbf{v}\|\,\omega_{n_{max}},$$

and the accompanying saturation function is

$$S = \begin{cases} \mathbf{u} - \frac{\mathbf{v}\mathbf{u}^\mathsf{T}\mathbf{v}}{s_{max}}, & \|\mathbf{v}\| \ge s_{max}, \mathbf{u}^\mathsf{T}\mathbf{v} \ge 0 \\ \mathbf{u}, & \text{otherwise.} \end{cases}$$

Normally one would not equate a holonomic model to a nonholonomic one, largely because of differences in controllability. However, full controllability is not essential to the DRCA algorithm since only position and velocity are essential. The DRCA algorithm is designed to use any control authority available, assuming controllability in the position and velocity states. Note that full controllability is generally required for the nominal control, $\mathbf{u}_d(t)$.

The relative position vector from vehicle i to vehicle j is denoted $\tilde{\mathbf{r}}_{ij} \equiv \mathbf{r}_j - \mathbf{r}_i$, while the relative velocity vector is defined in the opposite sense: $\tilde{\mathbf{v}}_{ij} \equiv \mathbf{v}_i - \mathbf{v}_j$. Note that these definitions imply that $\dot{\tilde{\mathbf{r}}}_{ij} = -\tilde{\mathbf{v}}_{ij}$, and $\dot{\tilde{\mathbf{v}}}_{ij} = \mathbf{u}_i - \mathbf{u}_j$.

A useful quantity to define is the dimensionless Deconfliction Difficulty Factor, η, to compare different systems in which collision avoidance is to be implemented. This factor is defined as

$$\eta = \frac{v_{max}^2}{u_{max} d_{sep}}. \quad (5)$$

Conceptually, this factor is the ratio of the worst case turning radius to the required separation distance. It can also be thought of in terms of stopping distance.

The DRCA algorithm was designed primarily for systems with large η (greater than unity) such as aircraft and ships, where the collision avoidance task is difficult because of the dominance

of vehicle inertia. Vehicles with small η (significantly less than unity) such as mobile robots can often be modeled as having a direct velocity command, since the inertia becomes insignificant. In such cases, potential function methods for collision avoidance may be preferable to the DRCA algorithm because of their simplicity and ability to closely pack vehicles. The downside of potential function methods is their general lack of provable guarantees, especially when inertia is considered.

2.2 Deconfliction Algorithm

The vehicles considered here are modeled as point masses, however physical vehicles have finite size. Therefore, to account for physical constraints in the theoretical model, the condition for conflict is not to attain the same position in space at the same time, but rather to come within a minimum allowed distance of each other at some point in time. This minimum distance could be, for example, the five nautical mile separation between aircraft required by the FAA or the sum of the radii of two vehicles.

Definition 1 (Collision) *A collision occurs between vehicles i and j when*

$$\left\|\tilde{\mathbf{r}}_{ij}\right\| < d_{sep,ij},$$

where $d_{sep,ij}$ is the minimum allowed separation distance between the vehicles' geometric centers.

For two vehicles not actively in a collision, the next question is whether they will collide if they remain on their present headings. This situation will be called a conflict.

Definition 2 (Conflict) *A conflict occurs between vehicles i and j if they are not currently in a collision, but with zero control input (i.e. constant velocity), at some future point in time they will enter a collision:*

$$d_{min,ij} \equiv \min_{t>0} \left\|\tilde{\mathbf{r}}_{ij}\right\| < d_{sep,ij}. \tag{6}$$

The following lemma provides a useful way to check for conflicts. To simplify the notation in the rest of this paper, the ij subscripts will generally be suppressed (for example, $\tilde{\mathbf{r}}_{ij}$ will be written as $\tilde{\mathbf{r}}$).

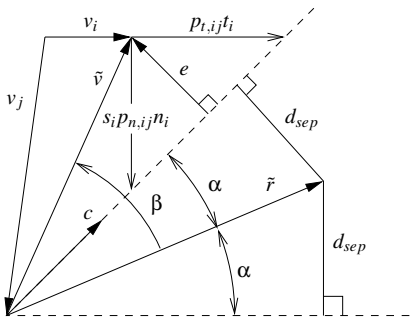


Fig. 1 An augmented depiction of the collision cone used in DRCA.

Lemma 2.1 *Let* $\beta = \angle\tilde{\mathbf{v}} - \angle\tilde{\mathbf{r}}_0$, $\alpha = \arcsin\left(\frac{d_{sep}}{\|\tilde{\mathbf{r}}_0\|}\right)$, *and* $\tilde{\mathbf{r}}_0$ *be the relative position vector at the time conflict is being checked. A necessary and sufficient condition for no conflict to occur is*

$$|\beta| \geq \alpha.$$

The angle α *represents the half-width of the collision cone ([3], [4], [2]), which is depicted in Fig. 1.*

A proof using this notation is given in [10], but is conceptually the same as the original collision cone proofs from [3].

If the system initially contains conflicts, a deconfliction maneuver must be performed to bring the system into a conflict-free state. One such maneuver is for all vehicles to turn left until the system is conflict-free. Once the deconfliction maneuver has been performed and the system is in a conflict-free state, then the deconfliction maintenance controller can be used to keep the system conflict-free. This controller allows each vehicle to use its desired control input unless that input would cause the vehicle to come into conflict with another vehicle. A basic block diagram of this process is shown in Fig. 2.

In order to smoothly transition from the desired control to the avoidance control, each vehicle needs a way to measure how close its velocity vector is to causing a conflict. The first step is to

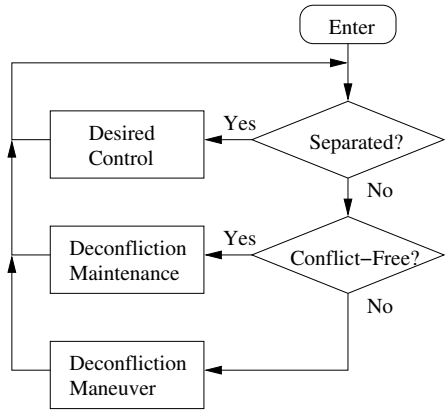


Fig. 2 DRCA algorithm flow chart.

construct a unit-vector, $\hat{\mathbf{c}}$, representing the side of the collision cone nearest $\tilde{\mathbf{v}}$. The vector $\hat{\mathbf{c}}$ is found by rotating $\tilde{\mathbf{r}}$ by α around a vector $\mathbf{q} = \tilde{\mathbf{r}} \times \tilde{\mathbf{v}}$ and normalizing:

$$\hat{\mathbf{c}} = \frac{\tilde{\mathbf{r}}}{\|\mathbf{r}\|}\cos\alpha + \left(\frac{\mathbf{q}\times\tilde{\mathbf{r}}}{\|\mathbf{q}\|\,\|\mathbf{r}\|}\right)\sin\alpha. \tag{7}$$

Next, construct a normal vector, $\mathbf{e}$, from the collision cone to the relative velocity vector, $\tilde{\mathbf{v}}$ (see Fig. 1). If $\hat{\mathbf{c}}^\mathsf{T}\tilde{\mathbf{v}} > 0$, then $\mathbf{e} = (I - \hat{\mathbf{c}}\hat{\mathbf{c}}^\mathsf{T})\tilde{\mathbf{v}}$, but if $\hat{\mathbf{c}}^\mathsf{T}\tilde{\mathbf{v}} \leq 0$ (the vehicles are headed away from each other), then no normal exists, and the nearest point on the collision cone is the tip, so $\mathbf{e} = \tilde{\mathbf{v}}$. Therefore:

$$\mathbf{e} = \begin{cases} \tilde{\mathbf{v}}, & \hat{\mathbf{c}}^\mathsf{T}\tilde{\mathbf{v}} \leq 0 \\ (I - \hat{\mathbf{c}}\hat{\mathbf{c}}^\mathsf{T})\tilde{\mathbf{v}}, & \hat{\mathbf{c}}^\mathsf{T}\tilde{\mathbf{v}} > 0. \end{cases} \tag{8}$$

In order to combine the effects of multiple collision cones, a useful approach is to decompose the system into three component directions and analyze those directions separately. Let the coordinate system be defined by the orthonormal vectors $\hat{\mathbf{t}}$, $\hat{\mathbf{n}}$, and $\hat{\mathbf{b}}$. The orientation of this coordinate system is arbitrary, but the convention of using tangent, normal, and binormal notation is chosen since fixing the coordinates to the body of the vehicle often simplifies analysis.

The next step is to determine how much control (change in velocity) can be applied in each of these directions before a conflict forms. For simplicity, a conservative approach is taken whereby the signed distance is found from $\tilde{v}$ to the tangent plane enclosing the collision cone (defined by the normal vector $\mathbf{e}$) in each of the $\hat{\mathbf{t}}$, $\hat{\mathbf{n}}$, and $\hat{\mathbf{b}}$ directions. These signed distances are

$$p_{t,ij} = \frac{\|\mathbf{e}_{ij}\|^2}{\mathbf{e}_{ij}^\mathsf{T}\hat{\mathbf{t}}_i},\ p_{n,ij} = \frac{\|\mathbf{e}_{ij}\|^2}{\mathbf{e}_{ij}^\mathsf{T}\hat{\mathbf{n}}_i},\ \text{and}\ p_{b,ij} = \frac{\|\mathbf{e}_{ij}\|^2}{\mathbf{e}_{ij}^\mathsf{T}\hat{\mathbf{b}}_i},$$

which are described graphically in Fig. 1.

Define $\varepsilon_t, \varepsilon_n, \varepsilon_b > 0$ as thresholds such that when $|p_t| > \varepsilon_t$, the conflict is far enough away that it can be ignored (and likewise for p_n and p_b). The n-vehicle deconfliction maintenance controller running on vehicle i computes p_t, p_n, and p_b to each of the other vehicles and then finds the closest conflict in each direction, i.e.

$$\begin{aligned} p_{t_i}^+ &= \min_j \left\{ p_{t,ij} > 0, \varepsilon_{t_i} \right\} \\ p_{t_i}^- &= -\max_j \left\{ p_{t,ij} < 0, -\varepsilon_{t_i} \right\}, \end{aligned} \tag{9}$$

and likewise for p_n and p_b. Note that by definition $0 < p^\pm \leq \varepsilon$. To simplify notation, in any case where a relation holds in all of the tangent, normal, and binormal directions, the subscript will be suppressed.

The input is constructed using a function, F, such that in each direction $u = F(p^+, p^-, u_d)$ (meaning the control choice is does not produce a conflict-generating velocity). The control function chosen for the implementation of the DRCA algorithm here is

$$\begin{aligned} F(p^+, p^-, u_d) = &\frac{u_{min}}{\varepsilon}p^+ + \frac{u_{max}}{\varepsilon}p^- \\ &+ \frac{u_d - u_{max} - u_{min}}{\varepsilon^2}p^+p^-, \end{aligned} \tag{10}$$

because it is a bilinear interpolation of the following ordered triples of the form (p^+, p^-, u):

$$\begin{aligned} P_1 &= (0,0,0) & P_2 &= (\varepsilon, 0, u_{min}) \\ P_3 &= (0,\varepsilon,u_{max}) & P_4 &= (\varepsilon,\varepsilon,u_d). \end{aligned}$$

An example of this control function is shown in Fig. 3. Because F depends on the desired control, $\mathbf{u}_d$ must be saturated such that

$$u_{min} \leq u_d \leq u_{max}. \tag{11}$$

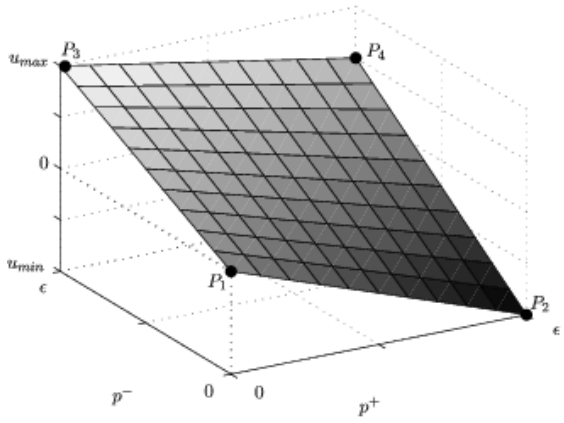


Fig. 3 Example of the control function, F. Note that P_4 moves increases and decreases with changing $\mathbf{u}_d$.

This choice of control function means that once $\mathbf{u}$ is constructed from its three components, then $\mathbf{u} \in \mathcal{R}$. Then the saturation function S will give the final resultant control vector, which will be in $\mathcal{C}$. The octant-preserving nature of S will ensure this final saturation step does not violate the requirements for the proof of collision avoidance.

A more intuitive way to choose a value for ε is to relate it to a gain-like parameter,

$$k = \frac{u_{max} - u_{min}}{\varepsilon}.$$

Note that k has units of inverse seconds. Also, one can see that the magnitude of the gradient of the control function will always be less than or equal to k, regardless of the desired control.

Assuming the deconfliction group, $\mathcal{D}$, has n vehicles, this algorithm's computation time on each vehicle scales as $O(n)$, because it only requires the computation of each other vehicle's collision cone and then substitutes these results into the control function. Technically $O(n^2)$ computations happen in the entire group, but since these computations are independent of each other (only linked by the sensed or communicated states), they can happen in parallel in a distributed fashion, so only the per-vehicle scaling affects computation time.

The system is guaranteed to maintain its conflict-free state despite arbitrary control authority restrictions with the following algorithm (see [9] for full theorem statement and proof). Each time a new vehicle is added to the deconfliction group, $\mathcal{D}$, it performs its deconfliction maneuver, broadcasting a conflict-free velocity to the group. Since this broadcast velocity is the quantity used by the deconfliction maintenance algorithm until the new vehicle is truly conflict-free, the deconfliction group never sees any new conflicts.

3 Testbeds

Two testbeds were utilized to evaluate the DRCA algorithm for different vehicle and environment types. The first testbed is composed of autonomous air vehicles which can hover in place and move in arbitrary directions. The second testbed is composed of autonomous underwater vehicles which are constrained to move at a constant velocity and can only change direction via heading changes.

3.1 Boeing Quadrotor System

The Boeing Vehicle Swarm Technology Lab (VSTL) was developed to provide rapid prototyping capabilities for mission algorithms, vehicle hardware, and health management ([14],[1],[5]). The indoor testbed is a $30.5 \times 15.2 \times 6.1\text{m}^3$ volume that allows for testing a heterogeneous mixture of autonomous vehicles. The testbed has full-state feedback capability through a VICON position reference system, pulsing visible light that bounces off reflective markers attached to the vehicles and using a system of cameras to triangulate the vehicles' position, velocity and attitude. Position accuracy is sub-millimeter and angular accuracy is sub-degree. The overall laboratory is shown in Fig. 4. Data for multiple vehicles is provided to the central data processing hub at 100 frames per second with approximately 10 milliseconds of latency [14]. The vehicles used for testing the DRCA algorithm are heavily-modified Draganflyer quadrotors as shown in Fig. 5.

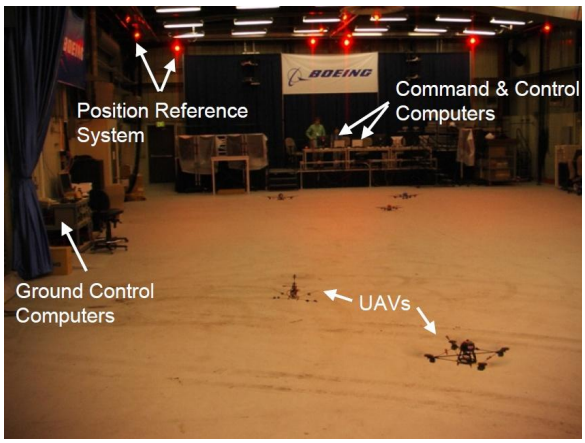


Fig. 4 Vehicle Swarm Technology Laboratory (VSTL) developed by the Boeing Research and Technology group.

3.2 University of Washington Fin-Actuated Autonomous Underwater Vehicles

The UW testbed is composed of a set of three fin-actuated autonomous underwater vehicles (Fig. 6) operating in a freshwater tank $6.1 \times 2.4 \times 2.4\text{m}^3$ in size. Four grey-scale underwater cameras in the upper corners of the tank are connected to a computer with a quad-core processor and 4GB RAM to perform real-time 3D tracking of objects in the tank. Position and velocity information is broadcast from this tracking computer to the underwater vehicles at 5 Hz. All control calculations are done onboard the vehicles themselves. Additional details on this system can be found in [12].

Fig. 5 Quadrotor vehicle equipped with reference markers.

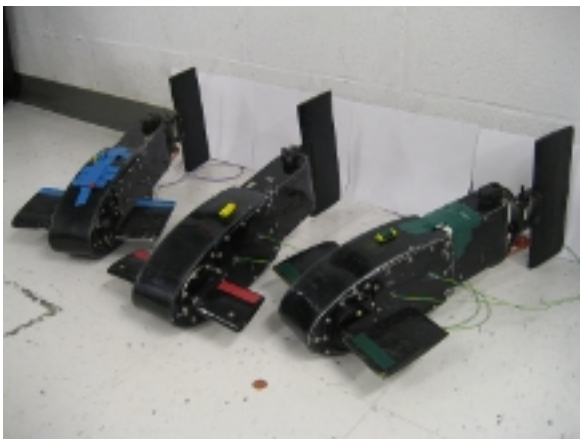


Fig. 6 University of Washington Fin-Actuated Autonomous Underwater Vehicle system.

3.3 Testbed Constraints and DRCA

Although these testbeds are adequate for testing the DRCA algorithm, they do not entirely match the assumed operational criteria of DRCA. The DRCA algorithm was designed using "velocity space." Because rotorcraft are continually changing velocity while hovering, the quadrotors are constantly going in and out of conflict even if they are commanded to be stationary a safe distance apart. Additionally, the algorithm is designed with the assumption of existence of space that is free of collision cones. This assumption guarantees collision avoidance in free operational space, but it can present problems in an enclosed environment. In certain scenarios, vehicles may be presented with the choice of a collision or going out of bounds (assuming that the boundary is a choice rather than a hard physical constraint), so that collision avoidance can no longer be guaranteed.

4 Experimental Results

Four experiments were run with the quadrotor system: one moving vehicle with a static obstacle, one moving vehicle and one vehicle with a fixed waypoint, two vehicles flying directly at one another and four vehicles flying at one another. Experimental results were compared to predicted results from numerical simulation using Matlab. Four experiments were run with

the UW system: two vehicles without collision avoidance performing a collision, two vehicles with DRCA avoiding the collision from the first experiment, two vehicles colliding obliquely without deconfliction, and two vehicles with an oblique interaction using DRCA.

4.1 Boeing Quadrotor System

The DRCA algorithm was implemented in C and C++ and is called at 50 Hz. For the purposes of this demonstration, the prescribed deconfliction maneuver for the vehicles was "all-turn-left". Several parameters can be adjusted within the DRCA algorithm to produce desired behavior: d_{sep}, s_{max}, u_{max}, ε, dt, and *bound*. The first three parameters were defined in the discussion of the DRCA. The term *bound* is the distance between vehicles at which deconfliction begins. DRCA expects to operate at the control (acceleration) level, modifying the desired control input to produce a new control that guarantees collision avoidance. However, the Boeing programming architecture outputs desired velocities instead of controls, so dt was a gain used to convert desired control to velocity.

The algorithm was tested both in experiment and in a Boeing simulation that uses Matlab Simulink to interact with the simulation environment SwarmView. The parameters above were adjusted to produce desirable collision avoidance performance. Since d_{sep} is defined for safety and s_{max} and u_{max} are properties of the vehicle, the three parameters actually determined were ε, *bound*, and dt. The parameters were determined by an initial set from simulation results before flight test and then refinement during flight test. The final values used to produce the results below are given in Table 1. The algorithm was tested in four different scenarios described below. The flight path and vehicle separation data came from the VICON position reference system. The conflict and control information used data from internal logging within the DRCA algorithm itself and is shown for vehicle 1 only.

Table 1 Parameters used in DRCA simulation

Parameter	d_{sep}	s_{max}	u_{max}
Value	1.0	1.25	5.6638
Parameter	ε	dt	*bound*
Value	1.0	$0.4 \times \frac{s_{max}}{u_{max}}$	6.0

4.1.1 *Static obstacle with one flying vehicle*

Flight testing for the static obstacle case (with the static obstacle being a quadrotor with no collision avoidance running) gives results similar to those found in simulation (Figs. 7(a)-7(d)). A bit of delay is present in the flight test that shows up in the flight path (Fig. 7(a)) and in the controls (Fig. 7(b)). Other than this delay, the flight path and controls looked similar in flight test and simulation, as does the algorithm performance. Observing Fig. 7(c) shows that the vehicles in flight test and in simulation had similar separation distances. Upon closer inspection, the vehicles came about 0.2 m closer in simulation than in flight test. Fig. 7(d) shows the speed of the moving vehicle and verifies the delay in the hardware environment. It also shows that the maximum speed in simulation was greater than the 1.25 m/s specified. This difference in speed probably resulted in the vehicles moving closer together in simulation (notice that the time of maximum speed corresponded to the time of minimum separation distance).

4.1.2 *Fixed waypoint with one flying vehicle*

The results for the case of one flying vehicle and one vehicle with a fixed waypoint (with DRCA activated on both vehicles) are shown in Figs. 8(a)-8(d). Fig. 8(a) shows that the flight paths of the vehicles were virtually identical with simulation. Likewise, Fig. 8(c) shows that the experimental controls and conflict resolution were extremely close to the simulated results. The speed (Fig. 8(d)) had some minor variations, but was quite similar overall between experiment and simulation. As specified, the speed of vehicle two had a maximum at 1.25 m/s. Fig. 8(b) shows that the vehicles maintained similar separation over time in simulation and in flight test. It also shows

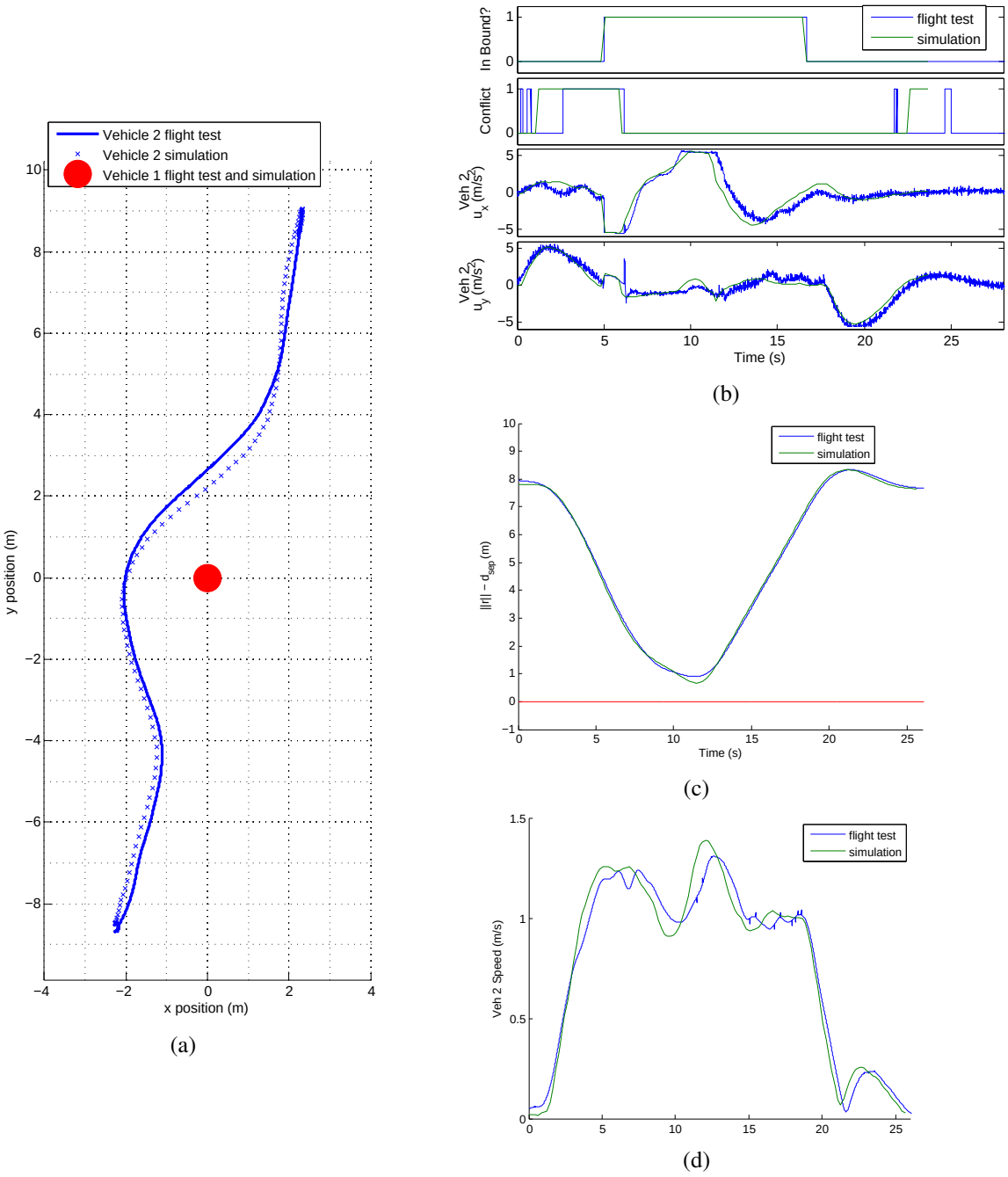


Fig. 7 Static obstacle with one vehicle. Fig. 7(a) shows the trajectories of the vehicles, and Fig. 7(b) gives the controls of the vehicles and the times that the vehicles were in bound and in conflict. Fig. 7(c) gives the difference between vehicle separation and minimum separation distance. Note that a collision occurs if the curve becomes negative. Fig. 7(d) shows the velocity of the moving vehicle.

that the vehicles did not collide. Overall, the DRCA algorithm performed as designed in this case.

4.1.3 Two vehicles flying at one another

In the case of two vehicles flying directly at one another, flight test results were quite similar to simulation (Figs. 9(a)-9(d)). Fig. 9(a) shows that the paths in simulation and in experiment were

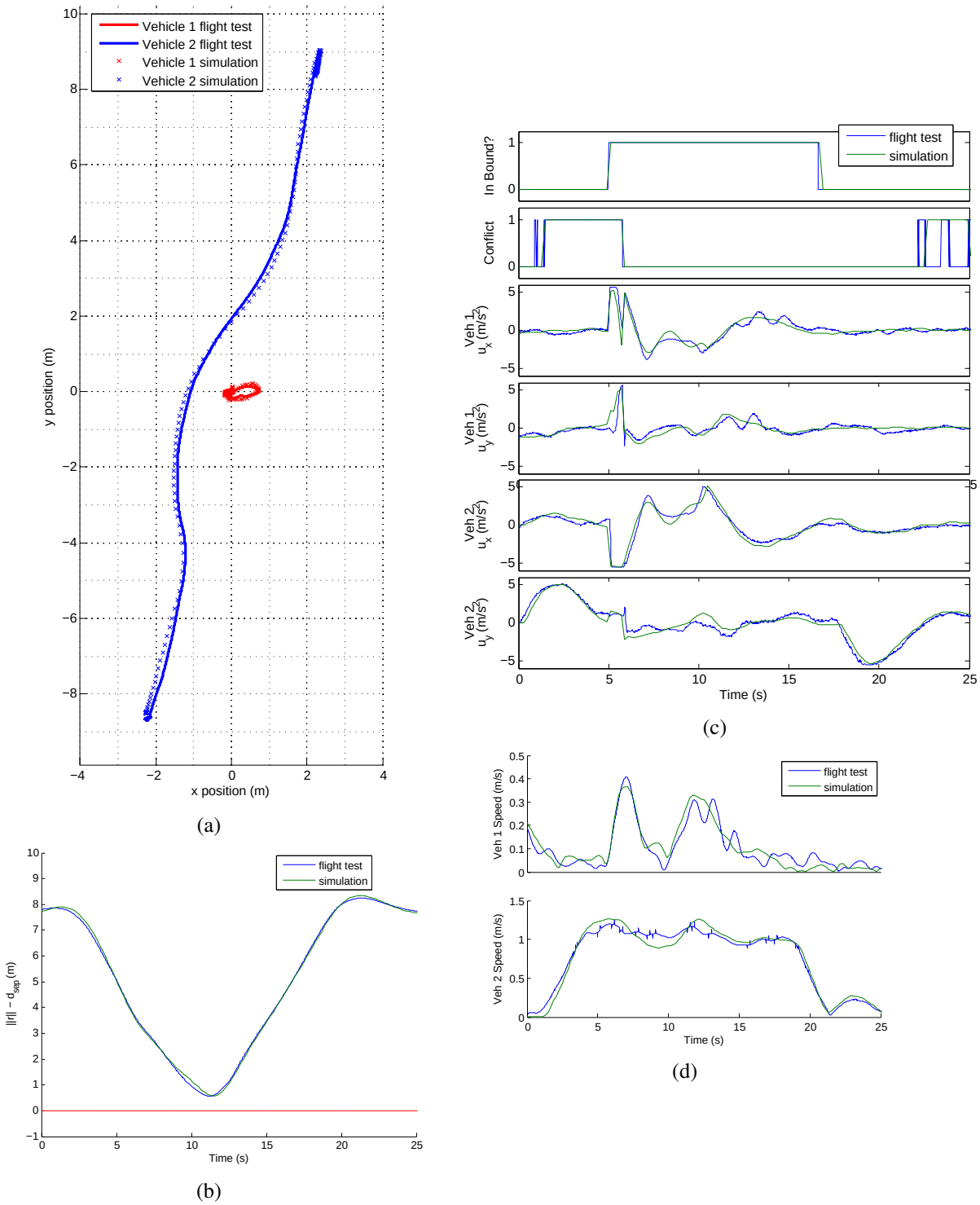


Fig. 8 Fixed waypoint with one flying vehicle. Fig. 8(a) shows the trajectories of the vehicles, and Fig. 8(c) gives the controls for the vehicles and the times that the vehicles were in bound and in conflict. Fig. 8(b) gives the difference between vehicle separation and minimum separation distance. Note that a collision occurs if the curve becomes negative. Fig. 8(d) shows the vehicle velocities.

similar, but that they did not line up perfectly. In flight test, the vehicles began their deconfliction maneuver earlier than in simulation and seemed to have a little bit of a lag in the controls when

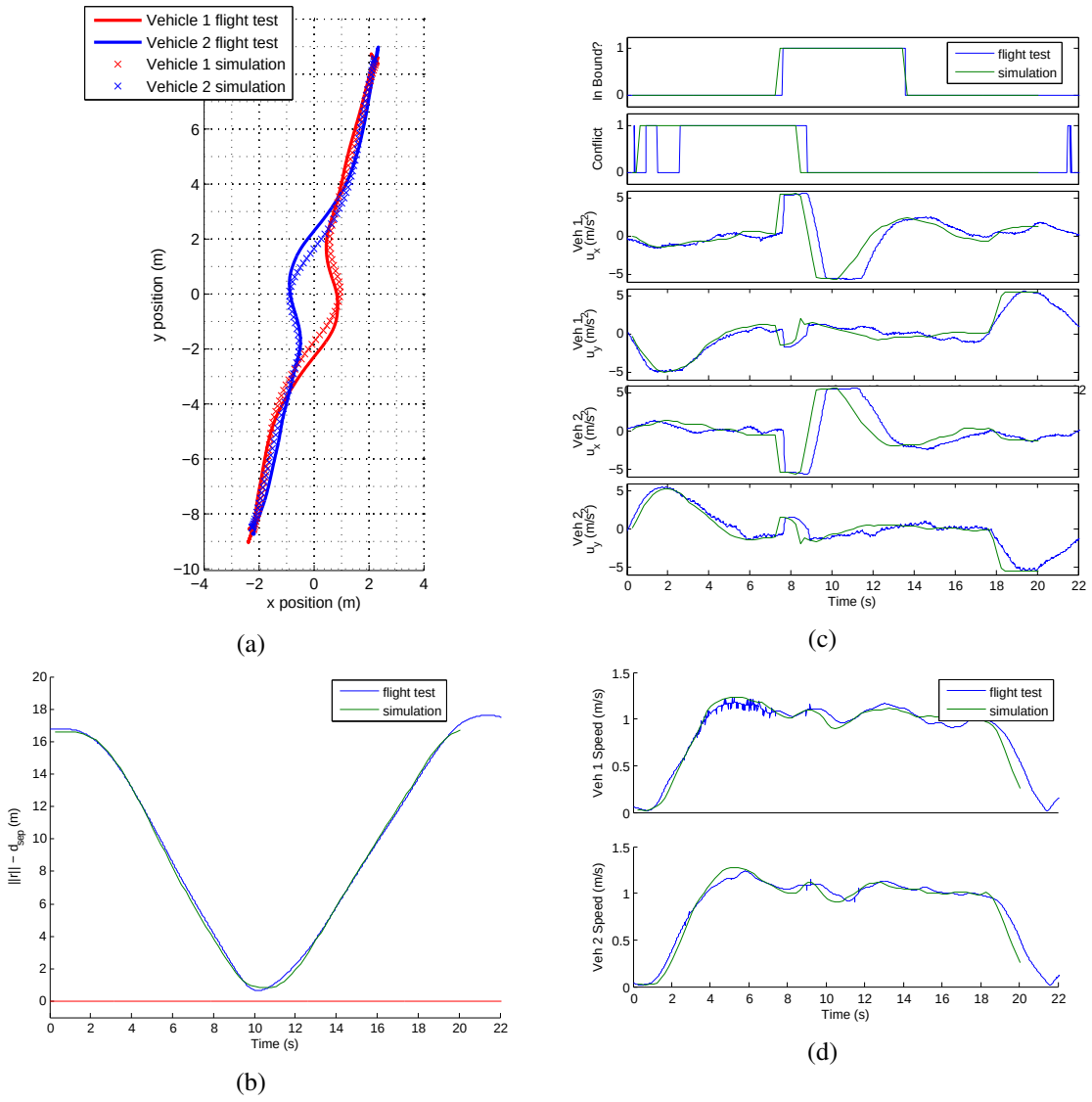


Fig. 9 Two vehicles flying at one another. Fig. 9(a) shows the trajectories of the vehicles, and Fig. 9(c) gives the controls for the vehicles and the times that the vehicles were in bound and in conflict. Fig. 9(b) gives the difference between vehicle separation and minimum separation distance. Note that a collision occurs if the curve becomes negative. Fig. 9(d) shows the vehicle velocities.

compared to simulation. Fig. 9(c) shows that controls in flight test were delayed compared to simulation, which likely was the cause of the different trajectories. Other than this delay, the controls were quite similar. Fig. 9(c) also shows that the vehicles deconflicted a bit slower in flight test compared to simulation. Fig. 9(d) shows the speed of the vehicles, which further confirms the delay in the system. Fig. 9(b) shows that the vehicles had similar separation with respect to time in simulation and flight test. The fact that the curve did not become negative indicates that the vehicles succeeded in avoiding collision.

4.1.4 Four vehicles flying at one another

The results from the four-vehicle head-on flight test are shown in Figs. 10(a)-10(b). Fig. 10(a) shows the trajectories of all four vehicles in flight test and compares them with simulation. All flight paths correspond well between simulation and experiment. Fig. 10(b) portrays the similar separation distances in flight test and in sim-

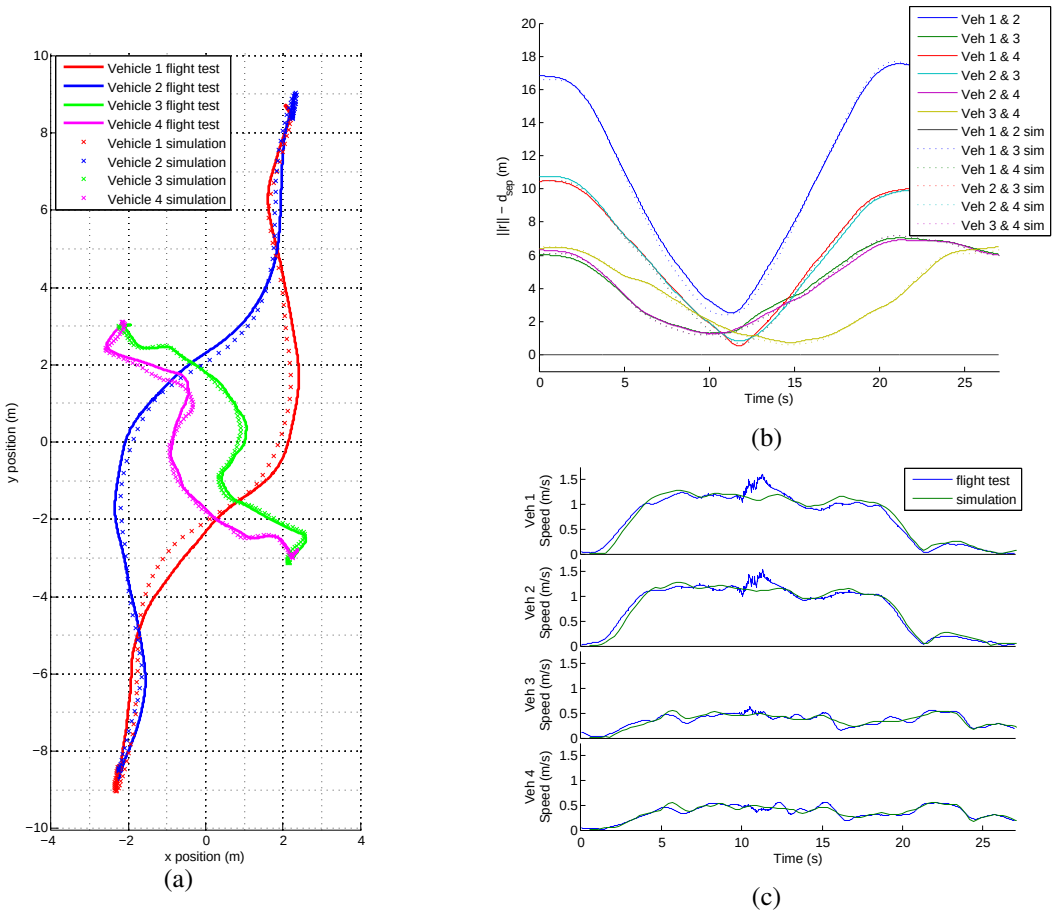


Fig. 10 Four vehicles flying at one another. Fig. 10(a) shows the trajectories of the vehicles, and Fig. 10(b) gives the difference between vehicle separation and minimum separation distance. Note that a collision occurs if the curve becomes negative. Fig. 10(c) shows the vehicle velocities.

ulation, and confirms that no collisions occurred between any vehicles. Fig. 10(c) shows similar speeds in simulation and in flight test, although speeds in the flight test spike a bit higher at the time when the vehicles are closest together.

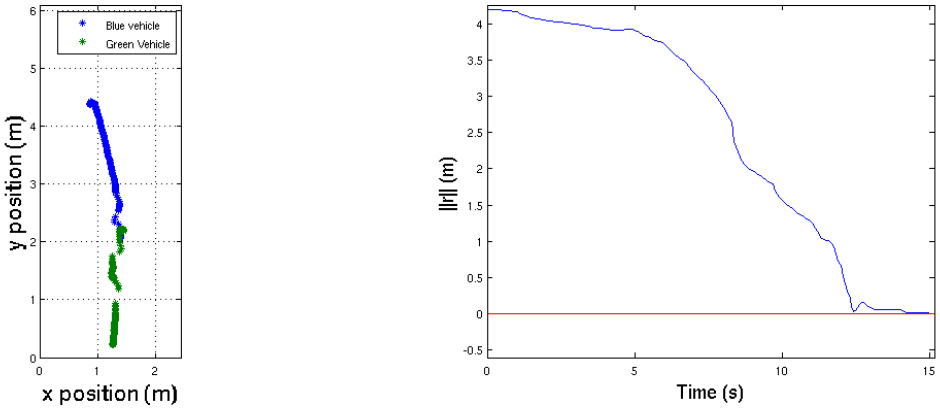
4.2 UW Underwater Vehicle System

The DRCA algorithm on the UW underwater vehicles used the "all-turn-left" deconfliction maneuver. The parameter *bound* used in the quadrotors was not used on the underwater vehicles because the DRCA algorithm was run at all times, regardless of vehicle separation. The vehicles were treated as having a constant commanded speed of 1.3 ft/s. The commanded separation distance d_{sep} was 1.5 ft.

4.2.1 Two vehicles with intentional collision

To demonstrate that the UW underwater vehicles will actually collide without use of a deconfliction algorithm when commanded to move directly at one another, two vehicles were set at opposite ends of the tank and commanded to seek a waypoint behind the opposite vehicle. DRCA was not used in this experiment.

Fig. 11(a) shows the trajectories followed by the vehicles during the experiment. Because the waypoint sought by each vehicle was directly behind the opposing vehicle, the waypoint algorithm caused the vehicles to engage in a head-on collision. Fig. 11(b) shows the distance between the vehicles. The distance between the vehicles decreases zero as the experiment progresses, indicating that the vehicles collided.



(a) (b)

Fig. 11 Two vehicles with intentional collision. Fig. 11(a) shows the trajectory of the underwater vehicles, and Fig. 11(b) shows the vehicle separation distance. Note that the vehicles collided while seeking their waypoints. The vehicles were *not* running the DRCA algorithm.

4.2.2 Two vehicles with DRCA

To demonstrate the effectiveness of the DRCA algorithm, the same waypoint controller and initial conditions were used as in §4.2.1, but with the DRCA algorithm running as an additional layer of control. The DRCA algorithm allowed the vehicles to reach their waypoints while avoiding collision.

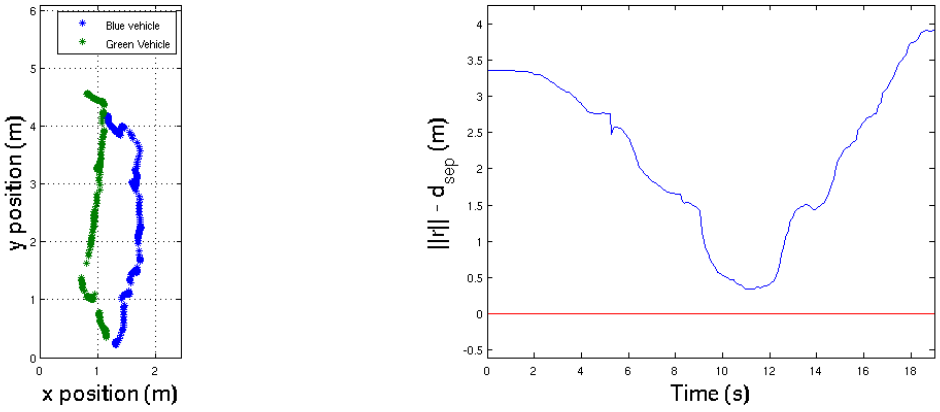
Fig. 12(a) shows the trajectories of the two vehicles running the DRCA algorithm. The vehicles started out in conflict and made an immediate turn to the left to reach a conflict-free state. They then passed each other outside the minimum separation distance and turned back to the right toward their waypoints. After they were completely past each other, they were able to go to their commanded waypoints, completing their mission. Fig. 12(b) shows the separation between the two vehicles during the run. Note that although the vehicles passed close to the minimum separation distance, they always remained in a collision-free state.

The only difference between the runs in this experiment and the runs in the experiment that intentionally caused a collision was the addition of DRCA control on top of the desired controller. The DRCA algorithm successfully prevented collision while permitting the vehicles to efficiently reach their targets.

4.2.3 Two vehicles with intentional oblique collision

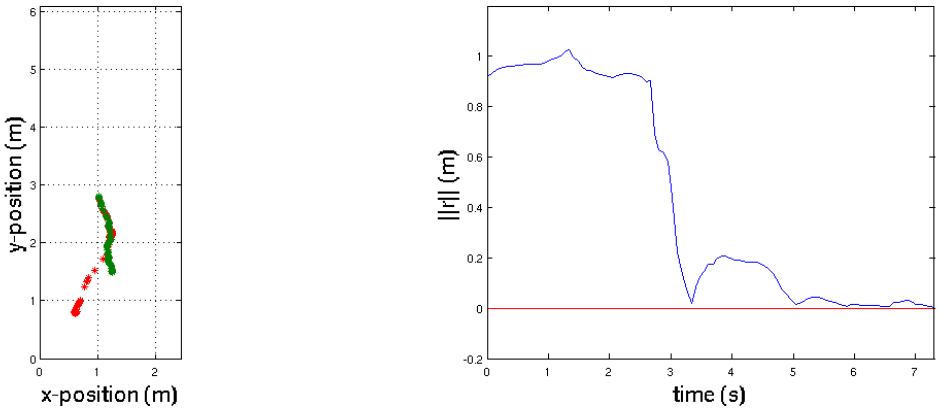
To demonstrate the deconfliction algorithm in a different situation, two vehicles were placed in adjacent corners of the tank and given waypoints along the long edge of the tank opposite their starting positions. Without the use of a deconfliction algorithm these conditions caused the vehicles to collide obliquely. For this experiment, one of the two fin-actuated vehicles was replaced with a remote control toy shark controlled by a human operator. The human operator drove the toy shark directly to its waypoint at a constant velocity.

Fig. 13(a) shows the trajectories of the two vehicles operating without deconfliction. Both vehicles move diagonally towards the opposing wall and meet in the center of the tank. Fig. 13(b) shows the distance between the two vehicles, which goes to zero, indicating a collision. Note that this collision is intentional and demonstrates that a collision will take place with these initial conditions if deconfliction is not employed.



(a) (b)

Fig. 12 Two vehicles with DRCA. Fig. 11(a) shows the trajectory of the underwater vehicles, and Fig. 11(b) shows the difference between vehicle separation and minimum separation distance. Note that the vehicles reached their waypoints without colliding.



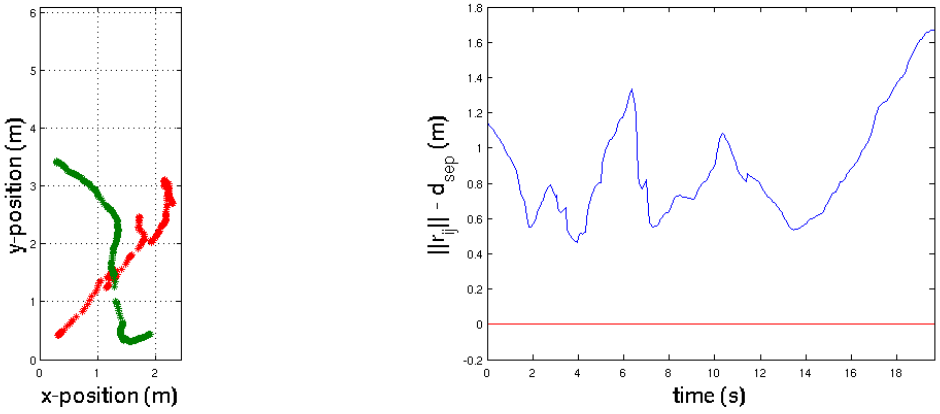
(a) (b)

Fig. 13 Two vehicles with intentional oblique collision. Fig. 13(a) shows the trajectory of the underwater vehicles, and Fig. 13(b) shows the vehicle separation distance. The vehicles were *not* running the DRCA algorithm.

4.2.4 Two vehicles with oblique interaction and DRCA

In this experiment, the conditions of §4.2.3 were repeated, but DRCA was added as an additional layer of control on the green vehicle. The red vehicle (the human controlled toy shark) was controlled directly to its waypoint and did not use the DRCA algorithm. Fig. 14(a) shows the trajectories of the two vehicles during the run. Because the vehicles begin in conflict, the green vehicle makes an immediate left turn to a safe heading. It then passes behind the red vehicle until it can follow its desired controller to the waypoint.

Fig. 14(b) shows the distance between the two vehicles in excess of the commanded separation distance. Note that the DRCA algorithm permits the vehicles to avoid collision even when one of the vehicles is not participating in deconfliction. At no point in the run did the vehicles come within the commanded separation distance of each other. The red vehicle did not intentionally behave antagonistically, but ignored the green vehicle during its run.



(a) (b)

Fig. 14 Two vehicles with oblique interaction and DRCA. Fig. 14(a) shows the trajectory of the underwater vehicles, and Fig. 14(b) shows the difference between vehicle separation and minimum separation distance. Note that the vehicles reached their waypoints without colliding.

5 Conclusion

The work in this paper has focused on the implementation of a deconfliction algorithm on two autonomous multivehicle platforms, one composed of air vehicles and one composed of underwater vehicles. For both testbeds, the DRCA algorithm was used to prevent collisions in a number of typical scenarios. Some particular points to note in these scenarios are that the DRCA algorithm was designed for systems with open boundaries with an assumption that conflict-free control choices exist and in which the only conflicts being considered are between vehicles using the algorithm. In fact, the results in the work here show that these assumptions can be relaxed, at least to a certain extent, and deconfliction can still be maintained. Results between experiment and simulation compared well indicating that the simulator for the air vehicles is accurately reflecting experimental performance. Current work is directed at exploring the effects of limited sensing (range and directionality) on the deconfliction algorithm.

Acknowledgment

The work in this paper was funded in part by the Boeing Company and in part by AFOSR grant FA-9550-07-1-0528.

Copyright Statement

References

[1] Bieniawski S, Pigg P, Vian J, Bethke B, and How J. Exploring health-enabled mission concepts in the vehicle swarm technology lab. *Proc Proceedings of 2009 Infotech@Aerospace Conference*, 2009.

[2] Carbone C, Ciniglio U, Corraro F, and Luongo S. A novel 3d geometric algorithm for aircraft autonomous collision avoidance. *Proc Proceedings of the 45th IEEE Conference on Decision & Control*, December 2006.

[3] Chakravarthy A and Ghose D. Obstacle avoidance in a dynamic environment: A collision

cone approach. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 28, No 5, pp 562–574, September 1998.

[4] Frazzoli E, Mao Z, Oh J, and Feron E. Resolution of conflicts involving many aircraft via semidefinite programming. *AIAA J. Guidance, Contr. and Dyn.*, Vol. 24, No 1, pp 79–86, 2001.

[5] Halaas D, Bieniawski S, Pigg P, and Vian J. Control and management of an indoor, health enabled, heterogeneous fleet. *Proc Proceedings of 2009 Infotech@Aerospace Conference*, 2009.

[6] Hwang C. N. The integrated design of fuzzy collision-avoidance and h-infinity autopilots on ships. *Journal of Navigation*, Vol. 55, pp 117–136, 2002.

[7] Kirk D. B, Heagy W. S, and Yablonski M. J. Problem resolution support for free flight operations. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 2, pp 72–80, 2001.

[8] Kuchar J and Yang L. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, pp 179–189, 2000.

[9] Lalish E. *Distributed Reactive Collision Avoidance for Multivehicle Systems*. PhD thesis, University of Washington, 2009.

[10] Lalish E and Morgansen K. Distributed reactive collision avoidance for multivehicle systems. *Proc Proc. IEEE Conf. Dec. Control*, 2008.

[11] Le M. D. An automatic control system for ship harbour manoeuvers using decoupling control. *Proc Proceedings of the IEEE Conference on Robotics & Automation*, 2001.

[12] Morgansen K, Triplett B, and Klein D. Geometric methods for modeling and control of free-swimming fin-actuated underwater vehicles. *IEEE Transactions on Robotics*, Vol. 23, No 6, pp 1184–1199, December 2007.

[13] Paielli R. A and Erzberger H. Conflict probability estimation for free flight. *Journal of Guidance, Control, and Dynamics*, Vol. 20, pp 558–596, 1997.

[14] Saad E, Vian J, Clark G, and Bieniawski S. Vehicle swarm rapid prototyphing testbed. *Proc AIAA Infotech@Aerospace Conference and AIAA Unmanned...Unlimited Conference*, April 2009.

[15] Tomlin C, Pappas G, and Sastry S. Conflict resolution for air traffic management: a study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, Vol. 43, pp 509–521, 1998.