Formation Flight of Swarms of Autonomous Vehicles In Obstructed Environments Using Vector Field Navigation

Christopher Lum, Juris Vagners, Matthew Vavrina, and John Vian

Abstract— **Maneuvering a single vehicle from one location to another is a common problem encountered by many autonomous systems. This problem becomes more complicated when an operator desires to maneuver multiple vehicles simultaneously in a coordinated fashion through a complex and potentially dynamic environment. This work investigates a modular control strategy which allows a group of vehicles, or swarm, to move in formation from a starting location to an ending location in space while navigating various obstacles and obstructions in a cluttered environment. The algorithm structure allows different aspects of the controller to be tuned and modified as desired for a specific task. High fidelity simulation is used to verify the algorithm before transitioning to flight test using multiple autonomous vehicles in real time.**

I. Introduction

A large portion of missions tasked to autonomous systems include intelligence, surveillance, and reconnaissance (ISR) scenarios. In recent years, there has been an increased interest in using micro air vehicles (MAVs) and other small aerial platforms which have the capability to operate indoors and in cluttered and complex environments to perform this type of ISR work. While their small form factor allows unique mission capabilities, these systems tend to have somewhat low reliability in comparison to other, larger unmanned aerial systems (UAS). To combat this problem, redundancy can be added by using multiple vehicles simultaneously in a cooperative fashion to perform the mission [1, 2]. An interesting problem in this context is the issue of allowing a single operator to move a swarm of vehicles from a starting location to a final destination while avoiding both inter-vehicle collision and environmental obstructions. A modular, autonomous algorithm which increases the autonomy of this type of system is the focus of this paper.

Multi-vehicle control has been studied by several groups in the past. Reynolds et al. performed some of the earliest work concerning formation and multi-vehicle control in the context of modeling groups of biological agents in schools and swarms [3]. This work was extended to computer graphics and modeling in later work [4–6]. Reynolds' work focused mostly on uncontrolled formation and swarm modeling. Other groups such as Popovic [7] and Kumar [8] have also looked at modeling movement of autonomous formations from a computer graphics and vision perspective. More traditional formation control work was performed by Balch et al. [9, 10] where they looked at formation control of multi-vehicle systems. These works investigated control algorithms to allow a group of vehicles to perform formation based navigation.

The strategy in this paper uses a different, less tightly coupled control algorithm to maneuver a large group of vehicles in a cluttered environment. The control algorithm follows ideas similar to work presented by Lipinski [11], Kim [12], and Murray [13] where vehicle navigation is achieved using stream functions and fluid models.

This paper presents a modular algorithm and control architecture that can be used to navigate a large number of heterogeneous agents in formation in a potentially cluttered environment. The main motivation for this work is to allow a single operator to efficiently and safely move a swarm of vehicles in formation from one location to another. The swarm should be able to autonomously navigate environmental obstructions such as obstacles, choke-point, and canyons. Vector fields representing individual desired vehicle velocities are generated by modeling how incompressible, inviscid fluid would naturally flow around these obstructions. These basis vector fields can be used to construct more complicated vector fields for other obstruction classes such as choke-point and canyons. These desired velocity vectors are then filtered by a tactical collision avoidance algorithm which takes into account vehicle volume and scales the magnitude of the desired velocity vector as a function of the conflict distance between agents. This system allows vehicles to deconflict and maintain spacing during various navigation maneuvers. These algorithms are then tested in a high fidelity simulation environment developed by Boeing Research and Technology. Once proved in simulation, the algorithms are seamlessly transitioned to the Vehicle Swarm Technology Laboratory (VSTL), an indoor test facility equipped to test multiple agent scenarios. Previous work at the University of Washington [14, 15] explored the use of various techniques to address the search problem with a group of heterogeneous agents.

Section II describes the algorithm development and its function. In addition to the main formation flight algorithm, a tactical collision avoidance filter is described in Section III. Simulation results for various scenarios are shown in Section IV. This section also describes the flight test environment and presents flight test data. Finally, Section V presents conclusions and some future directions of research.

C. Lum and J. Vagners are with the Department of Aeronautics and Astronautics, University of Washington, Seattle, WA, USA [lum, vagners]@u.washington.edu

M. Vavrina and J. Vian are with the Boeing Company, Seattle, WA, USA [matthew.a.vavrina, john.vian]@boeing.com

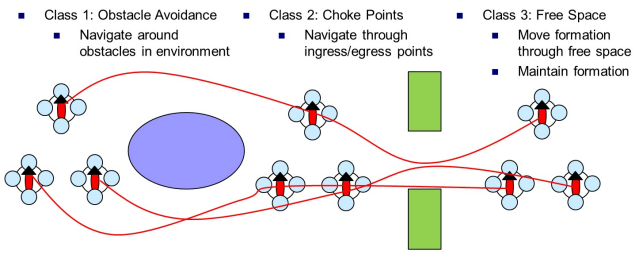


Fig. 1. 3 main sub-algorithms which allow swarm to navigate obstacles, choke-points, and maintain formation.

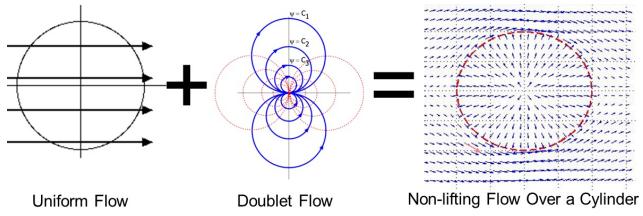


Fig. 2. Generating obstacle avoidance vector field using inviscid, incompressible flow theory.

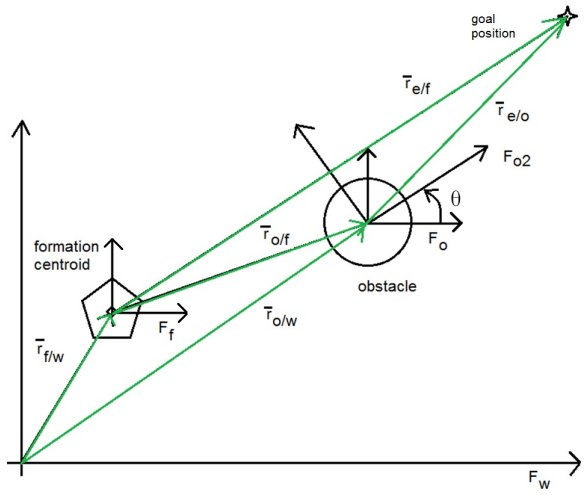


Fig. 3. Geometry showing various frames and vectors used for algorithm.

II. Algorithm Description

The motivating scenario for this work involves a single operator controlling a large group of vehicles during an ISR mission. The algorithm developed is used to move the swarm from one location to another. While in transit, the swarm should be able to autonomously navigate environmental obstacles such as buildings or other obstructions. It should also be able to maneuver through constricted areas such as windows or narrow canyons (these are referred to as choke-points). Finally, the swarm should be able to maintain a formation during transit in unobstructed airspace. The control algorithm is comprised of three sub-algorithms which handle each of these three cases as illustrated in Figure 1.

The control architecture is modular in the sense that different control laws may be used at different times, depending on the scenario [16]. For example, in Figure 1, a class 1 controller would be first used to navigate the obstacle. Once the swarm has cleared the obstacle, a class 2 controller would be activated to bring the swarm through the choke-point. Once the swarm passed the choke-point, a class 3 controller would be used to regroup the formation and move the group together as a single, cohesive unit. We investigate each of these three classes on controllers in the following subsections.

A. Class 1: Navigating Obstacles

When the swarm encounters an obstacle in its path, vehicles in the swarm should autonomously navigate around the obstacle. If the obstacle is directly in the path of the swarm, the swarm should bifurcate around the obstacles. If the obstacle does not significantly encumber the swarm (most vehicles have an unobstructed line of sign to the goal position), vehicles in the swarm should not significantly deviate from straight line paths from their current location to the goal position. This behavior is similar to how an incompressible fluid might flow around a solid obstacle (similar to a rock in the middle of a flowing stream). The class 1 control algorithm uses these ideas by creating a vector field which represents a virtual set of streamlines around an obstacle. This is done by using inviscid, incompressible flow theory and combining a uniform flow with doublet flow to generate a vector field of non-lifting flow over a cylinder [17–19] as shown in Figure 2.

The stream function, ψ, corresponding to Figure 2 with a cylinder of radius r in a field of velocity V_∞ is given by

$$\psi = V_\infty r sin(\theta)(1 - \frac{R^2}{r^2}) \tag{1}$$

The stream function can be differentiated to obtain flow field velocities in polar coordinates

$$V_r = \frac{1}{r}\frac{\partial \psi}{\partial \theta} \qquad V_\theta = -\frac{\partial \psi}{\partial r} \tag{2}$$

The vector field defined by Eq. 2 only produces flow from left to right. This vector field needs to be rotated depending on the location of the centroid of the formation, $\overline{r}_{f/w}$, the location of the centroid of the obstacle, $\overline{r}_{o/w}$, and the location of the goal position, $\overline{r}_{e/w}$. The global, world frame, F_w can be translated to the centroid of the obstacle to become the obstacle frame, F_o. This frame is then rotated by angle $\theta = \text{atan2}(\overline{r}_{e/f}(2), \overline{r}_{e/f}(1))$ to obtain frame F_{o2}. The geometry of the situation is shown in Figure 3.

Eq. 2 can then be converted to cartesian coordinates in order to obtain a vector of flow field velocity at any given point, expressed in F_{o2}. This is effectively the desired velocity for the vehicles to follow and is denoted $\overline{V}_d^{o2}$. For implementation purposes, this desired velocity vector is then expressed in the world frame

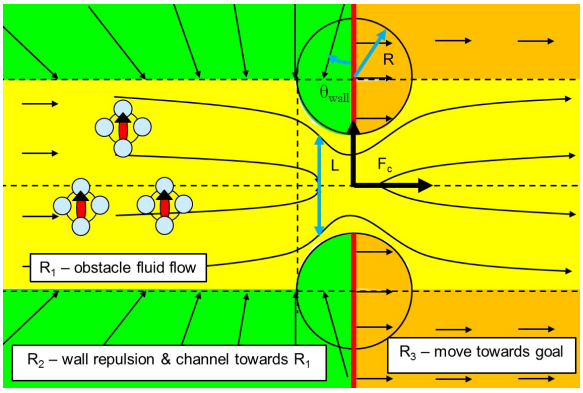


Fig. 4. Composite vector field and regions associated with a choke-point. Choke point of width, L, shown in red.

$$\overline{V}_d^w = C_{w/o2}(\theta)\frac{\overline{V}_d^{o2}}{||\overline{V}_d^{o2}||} \tag{3}$$

where $C_{w,o2}(\theta)$ is standard coordinate rotation matrix [20]. The class 1 controller consists of agents following the vector field defined by Eq. 3. As long as the vehicle is able to follow the generated streamlines, this method generates collision free trajectories which emulate how fluid would flow around the given obstacle as it travels from the centroid of the formation towards the goal location. These trajectories are smooth in the sense that there are no discontinuities and therefore are suitable for fixed wing aircraft or agents with non-holonomic constraints such as unicycle models. Furthermore, if the agents are not in conflict to start with and they have negligible volume, this method ensures that flight paths will not cross and generate collisions. Modifications to allow for finite volume agents are shown in Sections III. Another benefit of this approach is that it allows obstacles of a known radius to be directly modeled in the control algorithm (R in Eq. 1 corresponds to the actual obstacle radius).

B. Class 2: Navigating Choke-Points

The second type of environmental obstacle that the swarm may encounter is a choke-point. These are obstructions such as windows or narrow canyons where the swarm must somehow "become smaller" in order to navigate the obstruction. From a modeling perspective, a standard choke-point is parameterized by an origin, width, and orientation. A reference frame, F_c, is centered at the choke point origin with the positive x axis pointing through the choke-point.

Navigating a choke-point could be modeled using a similar incompressible fluid flow model by developing a vector field for uniform flow in a constricted pipe [19, 21]. However, superior performance was obtained by using a piecewise constructed vector field as shown in Figure 4.

This composite vector field is generated by adding additional parameters to the choke-point (effectively a derived class of the base choke-point described at the start of this section). We first consider only regions with a positive y^c component (above the dashed horizontal line in Figure 4), a virtual obstacle is superimposed over the choke-point. The space is then divided into three separate regions, R_1 (yellow), R_2 (green), and R_3 (orange). In R_1 regions, the vector field for standard obstacle avoidance is used (Eq. 3).

In R_2 regions, a vector field is created which is initially aligned with the x^c axis (at large negative values of x^c) and then transitions to align with the $-y^c$ axis when $x^c = -R$ and actually repels the agents from the wall when $x^c \in (-R, 0]$.

$$\overline{V}_d^c = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & x^c \leq -2R \\ \begin{bmatrix} -sin(\frac{\pi}{2R}x^c + \frac{\pi}{2}) \\ -cos(\frac{\pi}{2R}x^c + \frac{\pi}{2}) \end{bmatrix} & x^c \in (-2R, -R] \\ \begin{bmatrix} -sin(\frac{\theta_{wall}}{R}x^c + \theta_{wall}) \\ -cos(\frac{\theta_{wall}}{R}x^c + \theta_{wall}) \end{bmatrix} & x^c \in (-R, 0] \end{cases} \tag{4}$$

Finally, the vector field in R_3 regions simply serves to move the agents away from the choke-point and towards the goal-position $\overline{V}_d^c = [1 \;\; 0]^T$.

These vector fields are then simply reflected over the x^c axis, therefore defining a composite vector field for the entire space surrounding the choke-point.

By having the vehicles follow the prescribed velocity vector field, the emergent behavior is that the vehicles initially start far from the choke point and are not affected by the choke-point presence (they simply proceed to the right). As the agents approach the choke-point (x^c increases), agents which are not aligned with the choke-point (they are located in region R_2) begin to be funneled towards region R_1. Eventually, all vehicles enter the vector field defined in region R_1 which allow them to navigate through-the choke point. As they pass the choke point, the agents spread out again by following the stream lines.

C. Class 3: Free Space Formation Maintenance

Several algorithms exist for maintaining formation spacing and definition in free space [8, 9]. The free space formation maintenance control algorithm simply directs the vehicles towards desired formation positions. This has the potential to cause collision and should be improved in future development.

D. Class 4: Canyon Navigation

The previously described algorithms can be combined to achieve new goals and perform new tasks. For example, the class 2 algorithms for navigating choke-points can be used in series to create a vector field suitable for navigating narrow canyons. Figure 6 illustrates an example canyon scenario. The grey regions show the fixed geometry of the canyon walls. In order to generate a vector field for this scenario, several choke-points are superimposed over the canyon geometry (red lines in Figure 6). The vector field for each individual choke-point can be computed using

1 $\eta = \text{sort}(\{C_1, C_2, ..., C_n\})$
2 for each choke-point $C_i \in \eta$
3 if $\overline{r}^{C_i}(1) \leq 0$
4 $\overline{r}$ dominated by C_i
5 break
6 end if
7 end for

Fig. 5. Pseudo code for determining which choke-point is dominant for a given position.

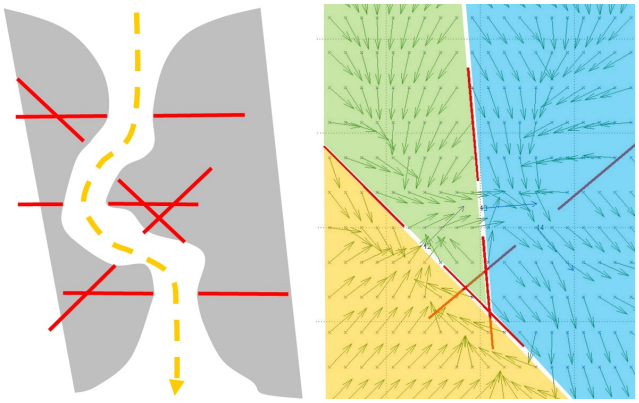


Fig. 6. Combining multiple choke-point vector fields to create a controller for navigating narrow canyon environments. Note that the left and right images are not correlated to one another.

methods described in Section II-B. The vector field for each choke-point is effective for the region directly in front of it, with precedence going to the choke-point which is encountered first when navigating the canyon. An algorithm for determining which choke-point is dominant for a given position in space is shown in Figure 5.

In line 1, the sort function is defined to sort the choke-points in order that they would be successfully navigated. This composite vector field and regions of dominance are shown in the right side of Figure 6.

As can be seen, this generates a vector field which will guide all vehicles through complicated geometry such as a canyon.

III. Collision Avoidance

The control algorithms described in Section II generate collision free trajectories by assuming that agents are able to follow the prescribed velocity vectors exactly and that the vehicles are point masses. However, for realistic scenarios with actual vehicle dynamics, imprecise inner loop controllers, and finite volume agents, these assumptions are not realistic. For example, during a class 1 maneuver (avoiding an obstacle), vehicles which are initially conflict free may encounter a conflict as the streamlines compress near the obstacle as shown in Figure 7. This section investigates a collision avoidance filter that may be applied to the desired velocity vectors to allow vehicles to deconflict and remain in conflict free trajectories [22].

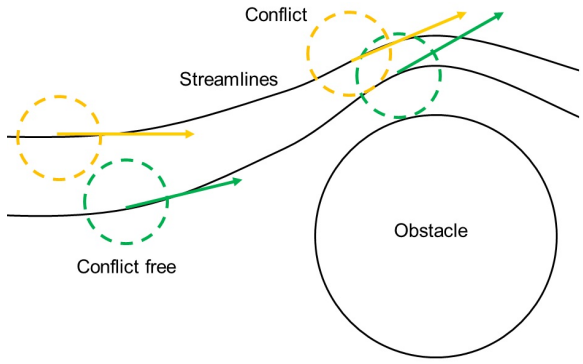


Fig. 7. Agents which are initially conflict free come into conflict during obstacle avoidance maneuver.

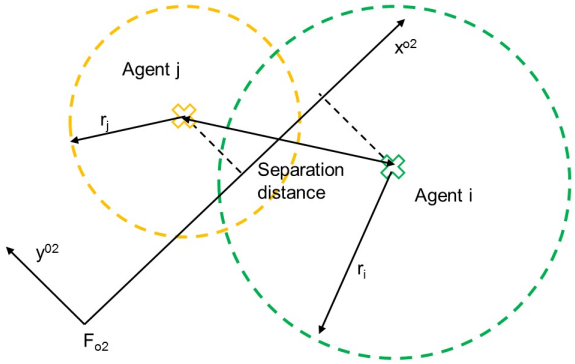


Fig. 8. Geometry showing conflict distance and geometry for two agents in conflict.

A. Tactical Filtering

The collision avoidance filter used in this application takes inspiration from how cars merge when entering a highway. In this scenario, the driver in front takes precedence and maintains their desired velocity. It becomes the job of the merging driver to modify their velocity vector in order to avoid a conflict. This is typically done by slowing down and allowing the separation distance between the two vehicles to grow to an acceptable distance before the merging driver resumes their desired velocity. In the scenario presented here, the concept of "in front" is achieved by expressing the agents velocity vectors in the F_{o2} frame as shown in Figure 8.

The separation distance between agents i and j is simply given as $s_d = ||\overline{r}_{i/w} - \overline{r}_{j/w}||$ where $\overline{r}_{k/w}$ denotes the position vector of agent k. Agents i and j are said to be in conflict if their separation distance is less than the sum of their radii. The magnitude of the conflict, d_c, is simply the difference between the sum of their radii and the separation distance

$$d_c = r_i + r_j - ||\overline{r}_{i/w} - \overline{r}_{j/w}|| \tag{5}$$

If the conflict distance is non-zero, then the trailing vehicle should slow down. Algorithmically, this translates to lowering the magnitude of the desired velocity vector of following agent as a function of the conflict distance, d_c. The

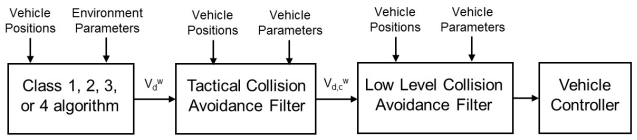


Fig. 9. Signal flow diagram for formation controller and various collision avoidance filters.

collision avoidance filter used in simulation and flight test is given by

$$\overline{V}_{d,c}^{w} = (||\overline{V}_{d}^{w}|| - \frac{\alpha||\overline{V}_{d}^{w}||d_c}{d_{c,max}})\frac{\overline{V}_{d}^{w}}{||\overline{V}_{d}^{w}||} \tag{6}$$

Eq. 6 describes the desired velocity vector of the trailing vehicle in the event of a conflict, $\overline{V}_{d,c}^{w}$. The term in parenthesis is the amount of velocity magnitude modification. The term α is a scalar which can be tuned to increase or decrease the aggressiveness of the collision avoidance filter. For example, with $\alpha = 0$, the collision avoidance filter is effectively turned off since $\overline{V}_{d,c}^{w} = \overline{V}_{d}^{w} \forall d_c$. The aggressiveness of the filter can be increased by increasing α. For $\alpha = 1$ if the agents are not in conflict, then $d_c = 0$ and the trailing vehicle simply maintains the original velocity vector $\overline{V}_{d,c}^{w} = \overline{V}_{d}^{w}$. However, if the vehicles are in maximal conflict (they are on top of one another), then $d_c = d_{c,max}$ and the trailing vehicle actually stops completely. The filter aggressiveness can be increased to actually make vehicles reverse direction by further increasing α past 1. In simulation and flight test with quadrotor aircraft, it was determined that $\alpha = 4$ yields desirable performance.

This filter is implemented in series with the original formation flight algorithms of Section II. The benefit of this is that the presence or absence of the collision avoidance filter does not affect algorithm behavior for formation control and environmental navigation.

B. Low-Level Collision Avoidance

The previously described collision avoidance filter is referred to as a tactical collision avoidance filter because it operates at a planning level by modifying the desired velocity commands which are sent to the vehicle controller. A second, low-level collision avoidance filter is implemented in the Boeing simulation and flight test environment to guarantee collision avoidance in the event of an emergency. Experience has shown that if the tactical collision avoidance filter is implemented properly, the low-level collision avoidance filter does not activate, all trajectory deconfliction is handled by the tactical collision avoidance filter.

The control signals originate from the class 1,2,3, or 4 algorithms and are then passed through the various collision avoidance filters before being applied to the actual vehicles. This signal flow is shown in Figure 9.

IV. Results

The algorithms are tested in both a rapid prototyping simulation environment developed at the University of Washington and a high fidelity simulation environment developed by Boeing Research and Technology. After passing software-in-the-loop verification and validation, the algorithms are transitioned to hardware flight test at the Boeing Vehicle Swarm Technology Laboratory (VSTL) in Seattle, WA.

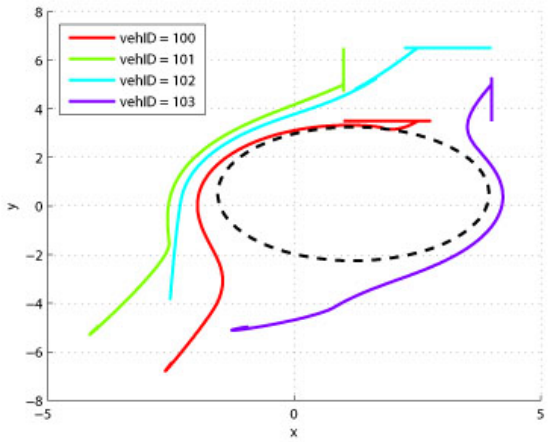


Fig. 10. Vehicle trajectories during obstacle avoidance maneuver.

A. Simulation Results

Several scenarios were simulated to exercise various aspects of the control algorithm and tactical collision avoidance filter. A simple mission is shown in Figure 10. In this scenario an operator assembles 4 vehicles at a rally point and then commands the formation to transition to the far side of an obstacle before regrouping in the same formation. As can be seen, the formation bifurcates around the obstacle with agents 100, 101, and 102 passing to one side of the obstacle while agent 103 navigates around the other side of the obstacle.

The agent velocities during this mission are shown in Figure 11. Each agent has a maximum velocity of 1 m/s. During the transition around the obstacle, the agents are constrained to a maximum velocity of 0.5 m/s. Between times 0 and 10 seconds, the formation is forming at the operator specified location using a class 3 algorithm. After the formation is assembled, the control is switched to a class 1 algorithm in order to move the formation to the far side of the obstacle. The effects of the tactical collision avoidance filter can be seen during this maneuver. Because agent 103 is not in conflict with other vehicles (it passes to the left of the obstacles whereas all other vehicles pass to the right), it is free to proceed at its maximal desired velocity $\overline{V}_{d,c}^{w} = \overline{V}_{d}^{w}$. However, the other three vehicles are initially in conflict at the start of the obstacle avoidance maneuver. After some initial transient effects resolve themselves, vehicle 100 is determined to be the vehicle in front. Therefore it proceeds along the stream lines at full velocity. Agents 101 and 102 actually reverse direction to deconflict with each other and vehicle 100 (Figure 11 is the magnitude of velocity, so the reversal is seen the sinusoidal motion of the vehicle 101 and 102 traces). Once vehicle 101 becomes deconflicted from

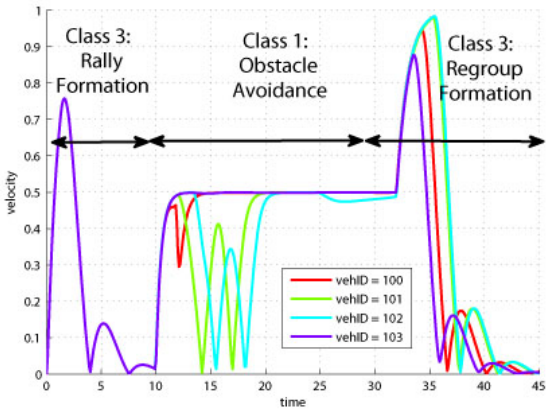


Fig. 11. Vehicle velocities during obstacle avoidance maneuver.

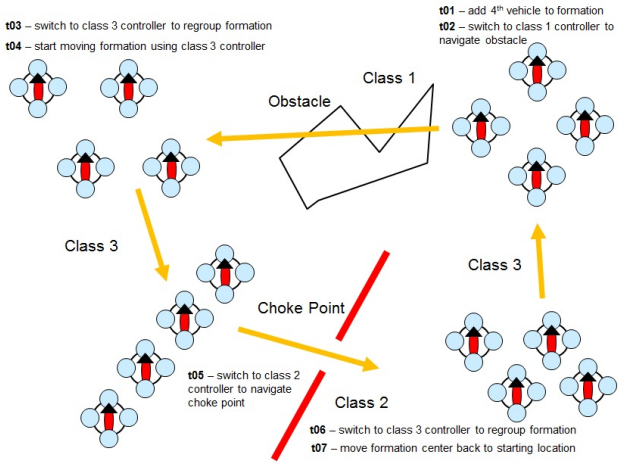


Fig. 12. Composite mission which exercises class 1, 2, and 3 algorithms.

vehicle 100, it is the next vehicle to proceed towards the goal along the streamlines. Finally, vehicle 102 is able to proceed forward once vehicle 101 has progressed forward sufficiently. Finally, at time 32 seconds, the swarm has successfully navigated the obstacle and the class 3 controller is used to regroup the formation.

One particularly interesting mission which exercises all aspects of the control strategy is shown in Figure 12. The mission timeline is as follows

- t00 Start simulation/mission with 3 vehicles. Vehicles assemble at rally point.
- t01 Operator adds a 4th vehicle to formation.
- t02 Operator commands newly formed formation to other side of an obstacle.
- t03 Swarm clears obstacles and regroups in original formation on far side of obstacle.
- t04 Operator changes formation configuration (from diamond to line) and moves centroid.
- t05 Operator commands formation to navigate choke-point.
- t06 Clears choke-point and regroups in original formation on far side of choke-point.
- t07 Operator commands swarm back to original position.

The trajectories of the agents during the composite mission are shown in Figure 13. In this scenario, the complicated geometry of the obstacle is enclosed with an equivalent sized circular obstacle (black dashed circle). The choke-point is shown in the lower left corner as a black dashed line. As can be seen, the vehicles initially move from their initial positions and form a diamond formation in the northern region of the area. The operator then commands the swarm to the far side of the obstacle and the class 1 algorithm maneuvers the agents successfully around the obstacle. During this maneuver, the collision avoidance filter is active and the agents deconflict as they travel around the obstacle. After the swarm clears the obstacle, the class 3 controller is used to regroup the formation. The user then decides to change the formation from a diamond to a line formation.

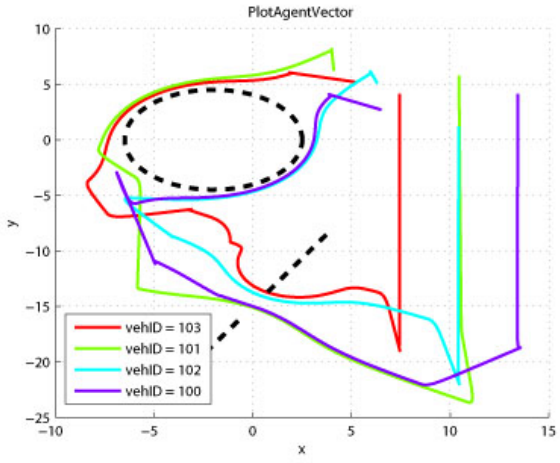


Fig. 13. Vehicle trajectories during composite mission.

This is perhaps the least optimal formation to achieve when attempting to navigate the upcoming choke-point (akin to 4 people walking abreast towards a single open door). This scenario was chosen since it would be the most taxing on the class 2 controller and tactical collision avoidance filter. As can be seen, all vehicles successfully enter and exit the choke-point in a collision free manner. Once the agents navigate the choke-point, the class 3 algorithm is used to regroup the formation and move it back towards the original location.

B. Boeing Hardware Environment

The algorithms were tested in the Vehicle Swarm Technology Laboratory (VSTL) developed by the Boeing Research and Technology group [23, 24]. This facility provides a large, indoor flight test arena where heterogenous teams may conduct various types of missions. The autonomous algorithms for each vehicle are executed on dedicated computers and the position information of all vehicles is captured with a system of cameras and coordinated pulses of light. The overall

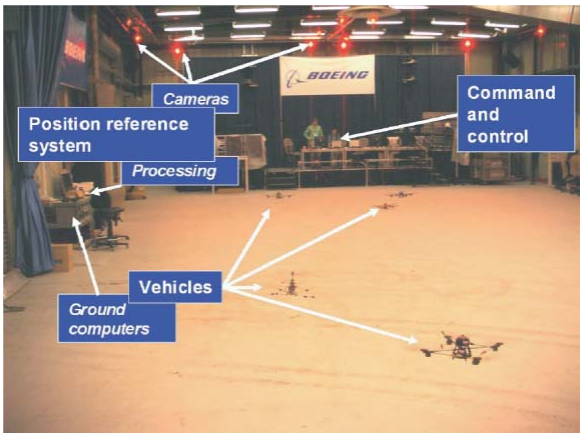


Fig. 14. Vehicle Swarm Technology Laboratory (VSTL) developed by the Boeing Research and Technology group.

Fig. 15. Quadrotor vehicle equipped with reference markers.

laboratory is shown in Figure 14. Data acquisition at 100Hz with sub 40ms latency is possible with this system for a large number of vehicles. The number of controlled vehicles is limited to 14 due to software and hardware architectures.

The flight test vehicles are heavily-modified, commercially available quad-rotor helicopters as shown in Figure 15.

The formation flight and tactical collision avoidance algorithm is developed in Python and integrated into the Boeing SwarmController software application. This is an application designed to interface with Boeing controllers and implement tactical or strategic control algorithms. Operators can interface with the vehicles and controllers via the SwarmView software application. A diagram of the software architecture and various applications in shown in Figure 16.

The algorithms can then be tested using a high fidelity numerical simulation environment developed in Matlab and Simulink. Once the algorithms are verified in simulation, they can be seamlessly integrated onto the actual hardware for real time operations. This facility has been used previously to validate other strategic, autonomous algorithms using this work flow [14, 15, 25].

C. Flight Test Results

Various flight tests were conducted at the Boeing VSTL during December 2011. These include

- Obstacle avoidance using actual physical obstacles

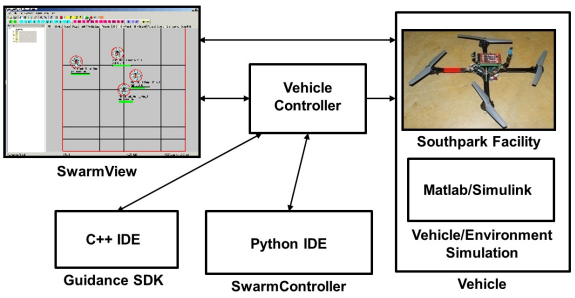
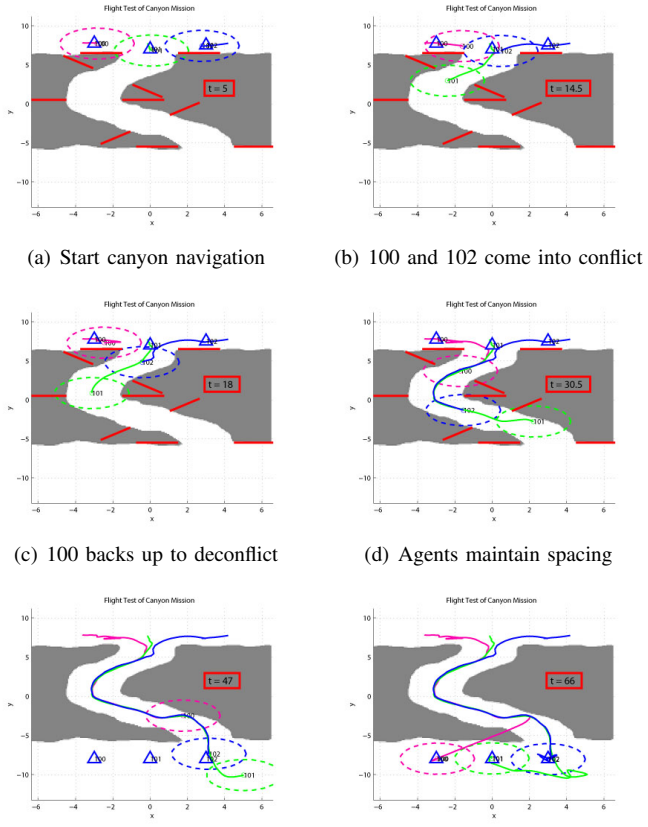


Fig. 16. Software architecture and applications in the Boeing VSTL. Formation flight control algorithms are implemented as part of the SwarmController application.

- Composite missions requiring both obstacle avoidance and choke-point navigation (Figure 12)
- Moving and dynamic obstacles and choke-points
- Complicated canyon missions

The most interesting flight test that was performed made use of existing choke-point algorithms to maneuver the swarm through a narrow canyon. The geometry of the canyon is shown in Figure 17. This example is unique because there is no direct line of sight from the entrance to exit. In other words, no straight line trajectories will allow a vehicle or swarm to navigate this obstruction. Based on this geometry, multiple choke-points are overlaid around the canyon and their vector fields are combined based on whichever choke-point is dominant in a given region. Finally, the collision avoidance filter is added on top of the controller. This allows a swarm to navigate the canyon without colliding with walls or other agents. The flight test telemetry of this canyon mission is shown in Figure 17.

During this mission, the operator first specifies that the three agents rally to an approximate line formation in front of the canyon mouth (Figure 17(a)). After the vehicles have assembled in formation, the operator commands the swarm to navigate the canyon by switching control from the class 3 controller to the class 4 controller. The agents begin to enter the canyon and almost immediately, agents 100 and 102 come into conflict (Figure 17(b)). The tactical collision avoidance filter activates and agent 100 actually reverses direction ($\alpha = 4$ for the filter) to deconflict with agent 102 which is in front (Figure 17(c). The agents then proceed through the canyon in single file while maintaining spacing due to the tactical collision avoidance filter (Figure 17(d)). During the flight test, the operator prematurely switched the control algorithm from the canyon navigation algorithm (class 4 algorithm) to the formation maintenance algorithm (class 3) before agent 100 clears the canyon (Figure 17(e)). This causes agent 100 to collide with the virtual canyon walls in its attempt to achieve the desired final formation (Figure 17(f)). Future iterations of this algorithm will check that agents have successfully cleared the canyon, obstacle, or choke-point before allowing the algorithm to switch to the formation maintenance controller. Despite this minor setback, this mission was completely successfully during the



(a) Start canyon navigation

(b) 100 and 102 come into conflict

(c) 100 backs up to deconflict

(d) Agents maintain spacing

(e) Switch to class 3 controller

(f) Agents achieve final formation

Fig. 17. Agent trajectories during canyon mission. Data is obtained from Vicon telemetry during flight test at Boeing VSTL.

flight test.

V. Conclusions

This paper presents a modular algorithm and control architecture that can be used to navigate a large number of heterogeneous agents in formation in a potentially cluttered environment. The main motivation for this work is to allow a single operator to efficiently and safely move a swarm of vehicles in formation from one location to another. The swarm should be able to autonomously navigate environmental obstructions such as obstacles, choke-point, and canyons. Vector fields representing individual desired vehicle velocities are generated by modeling how incompressible, inviscid fluid would naturally flow around the obstruction. These basis vector fields can be used to construct more complicated vector fields for other obstruction classes such as choke-point and canyons. These desired velocity vectors are then filtered by a tactical collision avoidance algorithm which takes into account vehicle radii and scales the magnitude of the desired velocity vector as a function of the conflict distance between agents. This system allows vehicles to deconflict and maintain spacing during various navigation maneuvers.

The versatility of this algorithm lies in its modular structure. Future improvements to various aspects of the controller can be contained to specific classes of algorithm and not influence the behavior of other classes. Future work is directed towards augmenting the class 3 formation maintenance controller. Currently, the controller is not guaranteed to provide collision free trajectories. As a consequence, during formation creation and maintenance, agents come into conflict and the tactical collision avoidance (or in extreme cases, the low level collision avoidance) must activate to deconflict the vehicles.

These algorithms are verified using an advanced numerical simulator from the VSTL. They are then transitioned to the hardware test bed and validated using multiple vehicles in real time flight tests. Future research directions involve using the versatile Boeing test bed to investigate human/automaton interactions using the VSTL interface. Currently user interaction with the autonomous system is somewhat onesided where users dictate the behavior. Current research at the University of Washington is directed towards two way communication and information flow between human operators and autonomous systems with the goal of improving performance and mission success rates.

References

[1] Lum, C. W., Rysdyk, R. T., and Pongpunwattana, A., "Occupancy Based Map Searching Using Heterogeneous Teams of Autonomous Vehicles," *Proceedings of the 2006 Guidance, Navigation, and Control Conference*, Keystone, CO, August 2006.

[2] Lum, C. W., *Coordinated Searching and Target Identification Using Teams of Autonomous Agents*, Ph.D. thesis, University of Washington, Seattle, WA, March 2009.

[3] Reynolds, C. W., "Flocks, Herds, and Schools: A Distributed Behavioral Model," *ACM Computer Graphics*, Vol. 21, No. 4, 1987, pp. 25–34.

[4] Reynolds, C. W., "Steering Behaviors for Autonomous Characters," *Proceedings of the 1999 Game Developers Conference*, San Francisco, CA, 1999.

[5] Reynolds, C. W., "Interaction with Groups of Autonomous Characters," *Proceedings of the 2000 Game Developers Conference*, San Francisco, CA, 2000.

[6] Reynolds, C. W., "Big Fast Crowds on PS3," *Proceedings of the 2006 Sandbox a Video Games Symposium*, Boston, MA, 2006.

[7] Treuille, A., Cooper, S., and Popvic, Z., "Continuum Crowds," *ACM Transactions on Graphics*, Vol. 25, No. 3, 2006, pp. 1160–1168.

[8] Das, A. K., Fierro, R., Kumar, V., Ostrowski, J. P., Spletzer, J., and Taylor, C. J., "A Vision-Based Formation Control Framework," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, 2002, pp. 813–825.

[9] Balch, T. and Arkin, R. C., "Behavior-Based Formation Control for Multirobot Teams," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, 1998, pp. 926–939.

[10] Arkin, R. C. and Balch, "Cooperative Multiagent

Robotic Systems," *Artificial Intelligence and Mobile Robots*, Vol. 5, No. 1, 1998, pp. 227–295.

[11] Lipinski, D. and Mohseni, K., "Cooperative Control of a Team of Unmanned Vehicles Using Smoothed Particle Hydrodynamics," *Proceedings of the 2010 AIAA Guidance, Navigation, and Control Conference*, Toronto, 2010.

[12] Kim, J.-O. and Khosla, P. K., "Real-Time Obstacle Avoidance Using Harmonic Potential Functions," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 3, 1992, pp. 338–349.

[13] Waydo, S. and Murray, R. M., "Vehicle Motion Planning Using Stream Functions," *Proceedings of the 2003 International Conference on Robotics and Automation*, 2003.

[14] Lum, C. W., Vagners, J., Jang, J. S., and Vian, J., "Partitioned Searching and Deconfliction: Analysis and Flight Tests," *Proceedings of the 2010 American Control Conference*, Baltimore, MD, June 2010.

[15] Melander, A., *Quadrotor Implementation of the Distributed Reactive Collision Avoidance Algorithm*, Master's thesis, University of Washington, Seattle, WA, 2010.

[16] Lum, C. W. and Vagners, J., "A Modular Algorithm for Exhaustive Map Searching Using Occupancy Based Maps," *Proceedings of the 2009 Infotech@Aerospace Conference*, Seattle, WA, April 2009.

[17] Anderson, J. D. J., *Fundamentals of Aerodynamics*, McGraw Hill, 2010.

[18] Milne-Thomson, L. M., *Theoretical Aerodynamics*, Dover Books, 1966.

[19] Zovatto, L. and Pedrizzetti, G., "Flow About a Circular Cylinder Between Parallel Walls," *Journal of Fluid Mechanics*, Vol. 440, 2001, pp. 1–25.

[20] Stevens, B. L. and Lewis, F. L., *Aircraft Control and Simulation*, John Wiley and Sons, Hoboken, NJ, 2nd ed., 2003.

[21] Zheng, Z. C., "A Consistent Boundary Condition for VorticityStreamfunction Simulation of Wall-Bounded Vortex Flow," *Journal of Applied Mathematics and Computation*, Vol. 206, 2008, pp. 205–213.

[22] Pallottino, L., Scordio, V. G., Bicchi, A., and Frazzoli, E., "Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems," *IEEE Transactions on Robotics*, Vol. 23, No. 6, 2007, pp. 1170–1183.

[23] Bieniawski, S., Pigg, P., Vian, J., Bethke, B., and How, J., "Exploring Health-Enabled Mission Concepts in the Vehicle Swarm Technology Lab," *Proceedings of 2009 Infotech@Aerospace Conference*, Seattle, WA, 2009.

[24] Halaas, D., Bieniawski, S., Pigg, P., and Vian, J., "Control and Management of an Indoor, Health Enabled, Heterogeneous Fleet," *Proceedings of the 2009 Infotech@Aerospace Conference*, Seattle, WA, 2009.

[25] Nigam, N., Bieniawski, S., Kroo, I., and Vian, J., "Control of Multiple UAVs for Persistent Surveillance: Algorithm Description and Hardware Demonstration," *Proceedings of the AIAA Infotech@Aerospace Conference*, Seattle, WA, 2009.