A Dynamic Swarm Approach to Artificial Potential Field Collision Avoidance
Technical Report #CSSE12-02

Hosea Siu*, Miriam Figueroa†, James Holt‡, Saad Biaz‡

*Massachusetts Institute of Technology
hoseasiu@mit.edu

†University of Puerto Rico in Humacao
miriam.figueroa4@upr.edu

‡Auburn University
holtjma@tigermail.auburn.edu
biazsaa@auburn.email.edu

Abstract—Path planning and collision avoidance for unmanned aerial systems (UAS) has been a growing area of research because of their increasing importance in both the civilian and military aviation sectors. This study introduces a method for implementing dynamic swarm formation and separation within an artificial potential field (APF) framework for collision avoidance. Bivariate normal APF functions were combined with field limit functions to form the basis of a collision avoidance system. Swarm rules were then implemented to allow for flexible close formation flight of clustered UASs that were headed towards clustered waypoints. Vehicles in close formation flight patterns ignore each other's APFs and effectively combine their fields in order to decrease unnecessary APF interference with other UASs in the airspace. When combined with a genetic algorithm to optimize APF and swarm parameters, this method offered significant improvements to UAS collision avoidance in crowded airspace compared to flight without a collision avoidance algorithm, and has the potential to improve UAS flight performance in many areas outside of collision avoidance.

I. INTRODUCTION

Recent developments in unmanned aerial system (UAS) technology have spurred an expansion of their use in both the military and civilian sectors. Path planning and collision avoidance for UASs are essential parts of safety and mission success when UASs are used [1]. In cases of increasingly dense UAS flight paths, increased supervision and in many cases, direct human control is required to ensure that vehicles do not collide. In these scenarios, UASs reduce the risk to human pilots, but would do little to reduce the workload required of pilots. Thus, significant research has been conducted on various collision avoidance and path planning algorithms. The goal is to have a closed-loop control system which would direct UAS motion to take into account static and dynamic obstacles, as well as signal errors and environmental disturbances while allowing the vehicles to carry out mission objectives.

A. Problem Statement

The general problem presented in this paper is the development of an algorithm of the kind mentioned above which would minimize the likelihood of near-misses for UASs flying in a constrained airspace, while allowing them to reach the maximum number of pre-generated waypoints as possible. The specific constraints of this study were designed to present scenarios that are unlikely in the real world but would make the problem of collision avoidance more difficult in order to see what kinds of algorithms could best handle extreme stress tests of planning and avoidance. Only dynamic obstacles in the form of other UASs controlled by the same algorithm were presented, with static obstacles (such as mountains, buildings, etc) being ignored. This is because independent dynamic obstacles drastically reduce the allowable time of calculation for a UAS to avoid a near-miss such that if an algorithm for this were to be successful, it would be able to handle additional static obstacles with ease. This problem constrains the UASs to be Dubins vehicles, which are fixed-speed constrained-turn-angle vehicles, to further simplify the problem. This also makes the solution more general, since an algorithm for Dubins vehicles also applies to non-Dubins vehicles, but the reverse is not true. All UASs in this study have independent, randomly generated sets of waypoints to follow.

This paper’s contribution is the introduction of a dynamic, flexible, flock-based approach to the artificial potential field (APF) framework of collision avoidance. By itself, APF has been shown to be an effective means of collision avoidance for autonomous robotic vehicles
[2]. However, in its application to UASs in vehicle-rich airspace, its efficiency decreases due to the confined space that the forces are exerted over. Thus, swarming rules that allow for flock formation and a suspension of normal APF rules for some special cases is introduced in this study as a way to increase both the effectiveness of APF in collision avoidance and the efficiency of UAS flight paths in this kind of environment.

II. LITERATURE REVIEW

A very general overview of the families of collision avoidance and path planning algorithms may be seen in Figure 1. It should be noted that although the groupings are done on the diagram in a hierarchical manner for simplicity, in reality, the algorithms borrow heavily from each other and a true visualization of their relationships would involve a complex web. In a very rough sense, algorithms that plan further ahead may be called predictive or path planning as they consider a greater section of the airspace and produce more optimal solutions, whereas algorithms that only solve subsets of the airspace that are local to specific UASs are known as reactive or collision avoidance algorithms. Reactive algorithms require much less computational power, and thus can return many more solutions in the time that it takes a predictive algorithm to return a general solution, but because they only consider a small portion of space around objects, the returned paths tend to be less efficient. A selection of the different families of path planning and collision avoidance algorithms used in the past is presented in the remainder of this section.

A. Potential Fields

Artificial potential fields (APFs) have been one of the most widely researched reactive mechanisms for ground-based robot movement control. Under this kind of algorithm, objects are modeled as attractive and repulsive forces, with the goal point being attractive and obstacles being repulsive. This creates an implicit path planning system based on the robot’s attraction to minimum potential valleys formed by the artificial forces it feels from the obstacles and goals around it. In this way, APF acts to bring a robot through the path of greatest attraction, which would ideally allow it to avoid obstacles [3]. APF has been a framework for a significant body of work that sought to improve on it while taking advantage of its simplicity and short calculation time [4]. The improvement has been necessary because at a very basic level, most implementations of APF suffer from the problem of local force minima being entered by ground robots, after which the robot may be stuck in the same location indefinitely. It should be noted though, that UASs do not often suffer from this problem with APF, since they have inherent forward momentum and would not stop in a very confined area.

UASs that are modeled as Dubins vehicles are particularly unlikely to enter local minima since the very definition of the Dubins vehicle, requires them to be at a constant speed. This, in addition to the fact that the UASs’ only obstacles are other, dynamically moving UASs, means that the overall potential fields are changing from step to step, thereby greatly reducing the danger of local minima [5]. However, Dubins vehicles may
enter infinite loops under APF when their turning radius does not allow them to successfully reach a waypoint because the attractive force of the waypoint causes what essentially becomes a central force requiring the UAS to continuously use its maximum turning motion. This is a problem that must be overcome by changes to the pure APF method, and is among a handful of special cases where APF must be altered slightly from its general behavior in order to avoid looping behavior, highly inefficient detours, and head-on collisions.

B. Skeleton Graph

Skeleton graph planning methods such as visibility graphs, Dijkstra’s Algorithm and Voronoi Diagrams have also been used in collision avoidance as explicit path planners.

Visibility graphs are constructed by connecting the vertexes of polygonal obstacles and then having robots travel along those lines. For dynamic cases, these paths may vary over time. A full path plan for all time can also be constructed from visibility graphs if the robot velocities can be varied. In these cases, robot velocities are changed so that they are planning a collision avoidance path through both space and time, such that they may occupy the same space as another robot (cross visibility paths) as long as they do not do so at the same time. Such a system would split path planning into path planning and velocity planning components. However, with robots of fixed speeds, other mechanisms such as additional steering may be required for near-misses to be avoided [7].

When applied to path planning, Dijkstra’s Algorithm turns the movement space into a series of nodes, for which it then proceeds to solve for the least-cost path. Nodes are assigned costs based on the factors that matter in the specific scenario, such as length, fuel cost, and environmental factors. The edges of these nodes then form a graph for which the minimum cost may be solved for [8]. Dijkstra’s Algorithm first assigns a tentative value of zero to the initial node being examined, and infinity for all others, and calls all nodes unvisited. Then, setting the initial node as the current node, it considers all neighbors of the current node and finds the cost of traversing to those nodes by adding the cost of the current node to specific one being traveled to. If a node cost is lower than what was previously found, the old cost is overwritten in favor of the lower cost. Once all neighbors of a node are visited, the current node is listed as visited and not traversed again. The neighbor with the lowest distance is then considered the current node, and the process is repeated. If there are still nodes in the unvisited set at the end of the algorithm, their costs remain infinite and are considered unreachable [9]. It should be noted that Dijkstra’s Algorithm is considered a form of optimized graph searching using skeleton graphs, which is why it appears under the optimization branch in Figure 1. This is an example of an algorithm that blends two different “families” of path planning, and shows why hierarchical representations of such algorithms is insufficient.

In a related way, Voronoi Diagrams plot out paths of greatest distance away from neighboring obstacles and requires that robots attempt to move along these lines, thus putting them at the safest locations possible [10]. These and other similar methods can be adapted to a dynamic movement problem by recalculating the associated skeleton graphs at each time step and updating the waypoints given to the robots as the environment changes around them. However, dynamic replanning is limited by the calculation time required for these
algorithms, and is made significantly more difficult with the addition of more robots.

C. Cell Decomposition

Cell decomposition methods also split the movement space into parts, but rather than graph nodes, these methods use cells that have movement costs associated with them. The cost that each cell has is primarily dependent on factors such as its distance from the goal and its distance from obstacles, though other factors could also be added to make the method more broadly applicable to different classes of problems. The two main components of a cell decomposition algorithm are therefore the way in which to split the movement space into cells and how to assign values to them. Cells may be of different sizes and shapes, and the construction of the cell diagram is dependent on the specific implementation of the algorithm. Examples of shapes that are commonly used include squares, trapezoids, and triangles, all of which may have uniform or nonuniform sizes depending on the specific implementation. The weighting of cell costs when they are near obstacles complicates these algorithms, but they remain relatively straightforward for static obstacles. The problem is made more complex for dynamic obstacles, since the calculation of cell costs must consider not only the present position of an obstacle, but also the likelihood of it being in another position in the future. This complexity is thus magnified by the number of UASs present. In the most basic case, cells that are in a direction that is closer to the goal are assigned highest values, and cells that are occupied by another robot are assigned a negative value. The likelihood of other UASs movements, especially when they are not broadcast, or if they are affected by some error makes the proper generation of a good cost heuristic an important part of ensuring an effective cell decomposition algorithm.

After cells and costs are assigned, search algorithms such as breadth-first, depth-first, best-first, and A* are then applied to the cells to determine the cost of movement, and return the minimal cost path [1]. Because the speed of these algorithms is limited primarily by the search time, changing the size of the cells and the time allotted for the search (or equivalently, the space that each robot will search) will change the nature of this algorithm from a complete planner where the entire system has been considered and the way each robot will move for the entire time has been set before they began, to a more reactive algorithm where paths are recalculated for each robot at each time step for their local subspace [11].

D. Mathematical Optimization

Mathematical programming algorithms are ones that transform the problem of path planning into a single mathematical problem, which is then solved using methods such as mixed integer linear programming. Linear constraints are put into the equations based on the physical constraints of the system—such as vehicle speed, altitude, turning limits, fuel, time, etc— which limit the possible solutions [12] [13]. The primary advantage to this kind of algorithm is that once linearized, the entire path planning and collision avoidance problem may be solved by commercially available linear system solvers such as CPLEX in an AMPL/MATLAB interface. However, it should be noted that a major disadvantage of this kind of algorithm is the fact that the problem grows exponentially with the number of independent dynamic objects in the system. This means that for larger problems that involve more UASs, MILP methods are prone to not being able to solve the system in time for the UASs to actually use the solutions. Additionally, since problems are solved in their entirety before any movement occurs, some implementations of MILP may not be very robust, and will not be able to take disturbances or errors into account.

Receding horizon is one way in which a large problem may be subdivided into smaller subsystems which MILP would be able to solve for much more quickly. Under this method, the problem is divided according to distances, with subsystems being clusters of objects that are close to each other, with objects that are further away temporarily considered separate systems. For example, a system of nine independent objects might be reduced down to three systems of three objects each, which
is much easier to solve. However, such subdivision is more difficult when greater speeds, limited airspace, and increased numbers of objects are introduced, since the effective time before obstacles are reached is dramatically reduced [14].

E. Genetic Algorithms

Genetic algorithm (GA) methods for path planning and collision avoidance utilize the principles of evolutionary computation in generating paths, comparing them, and combining them to create better solutions [15]. These algorithms can take in the constraints of previously known information about the airspace, such as static obstacles or known obstacle trajectories, and incorporate changes in the environment, such as air currents [16]. Additionally, changes in the objectives or the costs of various actions can be dynamically incorporated as the solutions adapt and are compared under newly introduced rules. Paths are initially randomly generated from the given constraints, and then compared against each other according to a cost system. Poorly-performing paths are removed from the candidate pool, and are replaced by combining parts of surviving paths, along with some additional induced randomness. The new paths are evaluated and then ranked against all the candidates. In this way, GA methods can usually generate some kind of solution in a relatively short period of time, though initial solution tend to be very sub-optimal. However, GA is designed to improve on initial solutions over time. As such, the quality of the solution would increase with the amount of time given. It should be noted though, that the number of generations of paths required to generate a near-optimal solution increases significantly with the number of UASs present, and like MILP, this kind of path planning may not be able to generate viable solutions for complex cases before the UASs must use it, if GA is being used directly to find paths.

While GA has primarily been used for path planning, it may also be adapted to path planning and collision avoidance in an indirect way. Evolutionary computation methods may be used to alter other parameters used by the UASs in planning, or to alter the way in which the system utilizes another collision avoidance algorithm entirely. An example of this is the use of GA to alter the parameters of potential fields- such as boundaries and strength- which were subsequently used as a reactive collision avoidance method [17]. However, layering GA on another algorithm increases the complexity of the system, which limits somewhat the possible combinations that it could be used with to algorithms that are fairly quick.

F. Dubins Paths

Dubins Paths represent a purely path-planning algorithm that does not take obstacles into consideration. It is worth noting here due to the constraints imposed on the vehicles used in this study, as mentioned above. Dubins Paths consist of either three consecutive curves, or a curve, a straight line, and a curve, which collectively have been proven to define the shortest path between two given vector configurations [19]. In the application of UASs, the bearing of the vehicle when it reaches a waypoint is generally not as important as reaching the waypoint. However, allowing Dubins Path to determine the final bearing is helpful, since it could orient the UAS in the best bearing for it to immediately proceed to its next location, if that is known beforehand. In this case, given an initial vector, the subsequent vectors are found by pointing the next waypoint towards the waypoint after that, repeating this until the last waypoint is reached. The last waypoint would then have a vector that points away from the previous waypoint to achieve the shortest path.

Dubins Path may also be used for cooperative path planning. First, a Dubins Path is calculated for each UAS going from waypoint to waypoint. Next, near-misses are identified and recalculated under given constraints. In some cases, if simultaneous arrival is desired, Dubins Path may also be used to calculate that. In simultaneous-arrival cases, all Dubins Paths for a group of vehicles are calculated. The longest Dubins Path is then identified, and the curvature values for the other paths are altered iteratively so the path lengths converge on the longest one. This would make it so that only the original longest path is a true Dubins Path for that particular type of vehicle, while the others would be using non-maximum turning angles or clothoid arcs. This kind of adjustment method can also be applied to time-dependent path planning in general, to ensure vehicle arrival at given times [20]. It should be noted, however, that Dubins Paths do not supply any kind of collision avoidance, as they do not take obstacles into account at all. As such, they are generally used either in situations where there is only one moving vehicle, or where they are used in conjunction with a collision avoidance algorithm, which would provide collision avoidance waypoints for the Dubins Path to use as a template in path planning.

G. Swarming

Swarm robotics utilizes simple rules applied to individuals in a multi-robot system that allow them to act cooperatively to accomplish a task. Swarming is not a specific kind of algorithm, but is rather an approach to collision avoidance and path planning that uses low-level control to achieve emergent high-level behavior. Because individuals are not burdened with complex algorithms, the computation time is relatively low, which is an advantage if the rules are formulated well enough to accomplish the necessary collective behavior [21]. This also allows for this kind of system to be decentralized
Fig. 5: Two Dubins Paths demonstrating a curve-straight-curve sequence and a curve-curve-curve sequence [18].

Swarm systems have been inspired by biological systems such as ants, bees, bird, and fish. In the same way, these algorithms encounter many of the same issues that their biological counterparts encounter, such as communication, leadership, environment mapping, and learning. Formations are generally maintained by the rules of separation, alignment, and cohesion that each member follows [22]. Many implementations of swarm robotics build on potential fields, but must shape these fields in a specific way so that separation is not always the dominant rule.

An example of using specifically shaped potential fields to maintain a swarm cluster rather than just repel neighbors is the use of limiting functions that separate different potential fields at various distances around a central agent to maintain robots in motion around the center point [23]. Specifically, bivariate normal and sigmoid functions have previously been implemented inside concentric elliptical limiting functions to maintain a swarm of protective ground robots around a convoy of "escorted" vehicles between them [5] [24].

Other swarm mechanisms draw more heavily on biological inspiration. Pheromone swarming involves the use of trails that are left behind when robots move over a location. In a retrieval-type or search mission, the robots are sent out randomly and the robot that locates the goal returns to the base on the same path that it went out on, thus doubling the pheromone strength on that path. Robots are further constrained to try to follow pheromone trails that are above a certain strength, which means once a goal is located once, other robots will swarm towards it. Pheromone deposits can also be made to decay over time, or have other changes that affect the behavior of the swarm with different environmental or time factors. Pheromones are considered a type of implicit or environment-based communication between robots [25].

Communication is also a key factor in any kind of swarm implementation. Implicit communication like the kind mentioned above mean that the individual members of the swarm alter their environment in some way that would notify other members of pertinent information. Explicit communication is another kind of communication in which members specifically broadcast information about their locations, environments, or routes to other members of the swarm either together or just to targeted individuals. This kind of communication is especially useful in cases of localization and robotic exploration, which would allow the individual members to act on the knowledge of the entire group to best utilize its resources and avoid potential dangers.

III. IMPLEMENTATION

A collision avoidance algorithm that uses elements of artificial potential fields, swarm intelligence, and Dubins Path planning is introduced. The main method used here is implicit path planning via artificial potential field collision avoidance. This was supplemented with dynamic swarm formation and separation methods to allow clusters of UASs to move together in flocks and increase the efficiency of their movement. Dubins Path lengths are used to help determine whether or not it would be advantageous for clusters of UASs to join into a flock.

A. Artificial Potential Field Construction

The construction of the potential fields used in this study were determined by two functions: one which
defined the boundary of the field, and another which defined the force within the boundary. In determining the type of functions to use, emphasis was given to choosing forces and boundaries that would work well together. Specifically, sudden jumps in force caused by either the potential function or the boundary were avoided when possible, as to avoid erratic UAS flight patterns that could decrease efficiency.

1) Limit Function: While natural potential fields such as gravitational or electromagnetic fields are unbounded, there is little reason to leave this artificial field the same way. This is because past a certain point, most viable potential fields will be so weak that there would be negligible effects on surrounding UASs, and only take up unnecessary calculation time. For holonomic vehicles, a circular limit function with the vehicle at the center would be natural since it could potentially move in any direction and change directions at any time. However, for Dubins vehicles, it would be of little use to have an extended potential field behind the vehicle, since it is impossible for the UAS to be in a position right behind its current trajectory within the next few time steps. It is however, likely that for any given UAS, it will continue to move forward along its current trajectory, and slightly less likely that it will turn. Thus, given this information, and appropriate APF boundary may be constructed.

It was originally thought that an oval with a wider end in the airspace in front of the UAS would be beneficial, because it would protect the areas that the UAS theoretically could fly into within the next few time steps effectively. However, it was found that the more airspace APFs covered, the less effective the method became. UASs that were traveling along paths that would be very unlikely to intersect anyway would give each other unnecessarily wide berths, as they were affected by far-reaching force functions. This tended to cause more near-misses with UASs when approach vectors coincided with directions where APF forces felt were decreased slightly (see next section). At an extreme, APF forces that extended too far made the overall flight behavior of the UASs almost similar to gas particle motion, with UASs always flying to avoid others, resulting in deterministic, but erratic behavior that was ineffective at either collision avoidance or reaching waypoints.

As such, the field boundary was reconfigured to situate the UAS at the wider end of the oval, with the overall area reduced significantly. This allowed for a good buffer area around the UAS, with the most likely trajectory of a straight forward path protected by a boundary that extended further out in front. The field limit was constructed in top and bottom parts as follows under a Cartesian plane with respect to the UAS, where $\gamma$, $\alpha_{top}$, $\beta_{top}$, $\alpha_{bottom}$, $\beta_{bottom}$ are constant parameters that determine the boundary shape:

$$y_{top} = \sqrt{\gamma - \frac{\alpha_{top}x^2}{\beta_{top}}}$$  \hspace{1cm} (1)

$$y_{bottom} = -\sqrt{\gamma - \frac{\alpha_{bottom}x^2}{\beta_{bottom}}}$$  \hspace{1cm} (2)

2) Force Function: Many different possible force functions exist for defining a potential field within the boundary. Essentially the only criteria it had to meet was that it had to allow some maximum where the UAS could be situated. These forces would all be functions of the particular location relative to the UAS emitting the force. The force within the boundary was defined by a bivariate normal function with the UAS located at the force maximum. This kind of function was chosen for its compatibility with the oval boundary limit and is defined by equation 3. $F_{max}$ defines the maximum force that is exerted by the field, and $\alpha$ and $\beta$ define the horizontal and vertical stretching of the field to fit the the boundary. The direction of this force is simply directly away from the UAS that is emitting it. The form of this force within its defined boundary is shown in Figure 6.

$$F_{repulse} = F_{max} e^{-\alpha x^2 - \beta y^2}$$  \hspace{1cm} (3)

3) Feel Function: In addition to the force projected by UASs, the force that is actually felt was modified by the heading of the UAS feeling the force. The full force is only felt if the UAS feeling the force was heading directly towards the source of said force. The percentage it feels is decreased as the bearing if the heading is increasingly away from the source, with only half the force felt if the source is behind the UAS. This function is defined by 4 where $f$ is the maximum force felt at the front (100%), $b$ is the maximum force felt at the back (50%), and $\theta$ is the angle of the source of the force. The form of this function is shown in Figure 7.

$$F_{felt} = f - \frac{f - b}{2} - \frac{f - b}{2} \cos \theta$$  \hspace{1cm} (4)

On the whole, each UAS would feel a constant attractive force from the next waypoint or the plane it is following, and a repulsive force from each plane for which it is within the APF emission boundary. As these forces are vectors, the summation gives a vector force according to 5. This vector is then corrected to be within the constraints of the UAS's speed and turning radius to give it the next waypoint it should be at.

$$\vec{F} = \vec{F}_{att} + \sum \vec{F}_{felt}$$  \hspace{1cm} (5)
B. Swarming

Swarm intelligence was implemented via the use of flocks, which underwent a cycle of identification, formation, movement, and separation.

Flocks were identified in three steps. First, the waypoints that each of the UASs was heading towards at the present time would be considered. If the waypoints of two UASs were within a certain cluster distance ($d_{\text{cluster}}$) of each other, the next step would be invoked. Dubins Paths would be created from the UASs to each other, and if the time required to traverse both Dubins Path ($t_{\text{Dubins}}$) was shorter than some maximum, the two would be called a flock. Time, rather than distance, was used here to give an intuitive sense of how long it would take for a flock to form, in order to allow judgment on what values might be reasonable. The follower and leader in the flock are determined by which UAS is closer to its waypoint, with the one that is closer being the leader. If only one UAS has a Dubins Path time that is shorter than the maximum, it would be considered the follower regardless of the waypoint distances because minimizing the amount of movement required to form a flock needs to be prioritized both for efficiency reasons and to ensure that the two UASs’ APFs overlap properly. It should be noted that Dubins Paths are used here strictly to decide whether or not flocks should be formed, and are not actually followed as a path, since they do not take obstacles into account.

Flocks were structured linearly, where each member of the flock had at most one leader and one follower. The first leader would be the flock leader and have no leaders of its own, and the last follower would be the flock tail and have no followers of its own. All members of a flock are aware of which other UASs are in their flock and would ignore their APF effects. Members would attempt to follow the UAS in front of them as closely as possible while maintaining a safe distance. The linear form of the flock was chosen so that the APF of the members of the flock would overlap as much as possible. The overlapping minimizes the amount of airspace covered by APF, so that the flight paths of other UASs would be minimally disrupted. Any UAS that was following another one also had its force boundaries reduced, following the same form as 1 and 2, but with different follower parameters being used.

UASs that are not in the flock feel at most the strongest two forces coming from a flock if it is in a region of APF overlap, as to ensure that the UAS does not avoid the leader at the expense of hitting a follower. This kind of flock structure resembles a column formation strategy, which has been shown to offer better performance in robotic collision avoidance than many other formations, including a no-formation strategy [26].

Due to the linear structure of flocks, other flocks and independent UASs may also join up by a process similar to the joining of two independent UASs, though only the flock leader may follow an independent UAS or flock tail, and only the flock tail can accept an independent UAS or flock leader into the flock.

Flocks separate when the flock leader gets close to its own waypoint. The leader will then split off, leaving the rest of the flock intact. The next UAS in the line then becomes the leader and goes towards its own waypoint, splitting appropriately. This way, members of the flock may go off to different waypoints in an effective manner that maintains the flock formation for the majority of the flight time.
C. Swarm Benefits

In addition to collision avoidance, swarming offers other important benefits to robotic systems and UASs in particular. These benefits mostly have to do with resource allocation and energy savings. For UASs that must perform sensing operations en-route to waypoints, formation flight allows them to focus their sensor resources, since they may now share at least part of their sensing workload with other members of the flock. Additionally, for fixed-wing aircraft, formation flight allows for members of a flock that are following another plane to take advantage of the induced upwash of the planes in front of them, which lowers the drag coefficient for the followers. Followers experiencing lower drag may see fuel efficiency benefits, as they may reduce their power and maintain the same speed [27].

It should be noted that the typical formations that allow for decreased drag are not the same as the column formation used here, but are rather a V-shaped formation. However, if the column formation strategy used here had each member offset slightly from a position directly behind its leader to form a staggered column, the UASs may be able to create a formation that balances the efficiency benefits of the combined APF with the fuel savings of formation flight.

D. Genetic Parameter Optimization

Since there are many constants that define the various parameters required for the force field, the force boundary, and the flock formation conditions, it was generally difficult to specify good constants for better performance beyond some general APF shapes that could be simulated by MATLAB plots. As such, a genetic algorithm was used to optimize the force and flock parameters.

For the genetic algorithm, candidate parameter sets were randomly generated and run through the simulation. Fitness was determined by the number of near-misses, conflicts, and waypoints returned by each candidate, with consideration given in that order. After evaluating the results, candidates were eliminated based on a probabilistic function of their fitness given by the following:

$$P(\text{survival}) = 1 - \left(\frac{R - 1}{N^2}\right)^2$$  \hspace{1cm} (6)

where \(N\) was the total number of individuals in that particular generation, and \(R\) was the individual’s fitness rank. Since evaluation of a candidate required running a simulation using those results, evaluation time limited the number of individuals that it would be feasible to consider. Thus, \(N\) values typically ranged from 8 to 12.

The eliminated candidates were then replaced by mixing two surviving candidates’ parameters and applying some mutation to allow for increased variability. Each parameter had a 1/3 chance of mutation, for which there was a defined limit to the mutation that would not allow the value to vary too far beyond its previous value. In addition to normal mutation, there was a 1/30 chance of an "extreme mutation," which would allow the parameter to randomly change to any allowable value for that parameter. In the case of only a single survivor, the pool was repopulated using mutants of the survivor following the same rules as mutations after mating.

Probabilistic survival rates were used to maintain candidate diversity, such that the candidate pool would not converge to similarity too quickly. Additionally, a history of evaluated candidates was kept, such that if any member that was about to be evaluated was found to have the same parameters as a previously considered candidate, it could be mutated before evaluation to ensure that the same parameters were not checked twice. In this way, successive generations of candidate solutions would be increasingly fit overall, and tend towards optimal parameters after many generations.

E. Test Platform

The algorithm tested in this study was implemented on a UAS simulation framework for Robot Operating System (ROS) Fuerte. Tests were run on Ubuntu Linux versions 11.1 and 12.04, which are the two most recent versions of Linux supported by ROS. UAS courses were created by randomly generating waypoints for UASs to follow. Courses differed by the number of UASs involved, and the space limit under which waypoints could be generated. For example, a 500 meter course simply constrains the locations that the waypoints could be generated to a 500m square. Courses were run for 10 minutes each, with enough waypoints for each UAS so that no UAS would make it to the end of its own
sequence within the time constraint. Additional constraints include the following: constant altitude flight, airspeed held constant at 25mph (approximately 11m/s), maximum turning radius of 22.5 degrees per second, telemetry update frequency of 1 Hz, and an assumed straight-line travel path between updates.

The constant airspeed and constrained turning angle makes each UAS a Dubins Vehicle, which is why Dubins Path is applicable to this problem. Constant altitude reduces the problem down to two dimensions, which makes collision avoidance much more difficult.

**F. Test Procedure**

Combinations of 4, 8, 16, and 32 UASs on 500m and 1000m square fields were used, with three different randomized courses used for each combination. A series of controls were run without collision avoidance, and each test that was conducted using collision avoidance used the best parameters that were obtained for that course obtained using genetic optimization. Additionally, a series of tests were run with 16 and 32 UASs that varied only the amount of swarming that was allowed, in order to test the effect of swarming on collision avoidance.

**IV. RESULTS**

Table I shows the optimal swarm and APF parameters obtained by genetic optimization. Note that $\alpha_{bottom} = \alpha_{top}$, $\alpha_{bottom,follower} = \alpha_{top,follower}$, and $\beta_{bottom,follower} = \beta_{bottom}$, as these are not independent variables, and are thus not included in the table. Additionally, genetic optimization sometimes returned $d_{cluster}$ and $t_{Dubins}$ values where one was zero and the other was non-zero. In these cases, the non-zero value was changed to zero, since swarming behavior will never occur when either value is zero.

Figures 9 to 12 show the changes in flight performance when the dynamic swarm APF using the optimized parameters was compared to the control case of no collision avoidance.

A near-miss was defined as occurring when two or more UASs were within one time step (11m) of each other, and a conflict was defined as occurring when two or more UASs were within two time steps (22m) of each other. A 100% reduction means that the number of near-misses or conflicts was brought down to zero by the algorithm. The percent change in waypoints reached was a simple ratio between the number of waypoints reached while using collision avoidance and the number
of waypoints reached without collision avoidance. This assumes that the number of waypoints reached without collision avoidance is the maximum possible number that could be reached in any particular case. The distance ratio is defined as the actual distance traveled to obtain waypoints over the straight-line distance between the waypoints obtained, and the change is the change from not using collision avoidance to using the dynamic swarm APF algorithm.

Figures 13 and 14 show the results of a series of simulations that demonstrate the effect of swarm behavior on scenarios involving 16 and 32 UASs. Scenarios with more UASs were used for these simulations because those are the scenarios where swarming behavior has the greatest chance of occurring and has the greatest potential effect. For these simulations, all optimal parameters were used except for $d_{\text{cluster}}$ and $t_{\text{Dubins}}$. The $d_{\text{cluster}}$ value was kept at 60m, and $t_{\text{Dubins}}$ varied from 0s (no swarm behavior) to 16s (Dubins Path travel times of up to 16s allowed for flock formation).

V. DISCUSSION

It may be seen from Figures 9 and 10 that dynamic swarming APF is an effective means of collision avoidance that can significantly decrease the number of near-misses and conflicts between UASs in crowded airspace. In many cases, this algorithm can drive the number of near-misses to zero, though it should be noted that this comes at an increasingly high cost in terms of waypoints reached and distance ratio as the density of UASs increases (Figures 11 and 12). This trade-off is due to the fact that with increasingly dense airspaces, UASs must move further and further away from their straight-line paths in order to avoid other aircraft. The effect is especially noticeable on the 500m fields, where increasing numbers of UASs increases the need to avoid other aircraft much more quickly.

The flocking simulations shown in Figures 13 and 14, as well as the $d_{\text{cluster}}$ and $t_{\text{Dubins}}$ values obtained by genetic optimization in Table I demonstrate that swarm behavior often has an adverse effect on collision avoidance in this type of simulation. The general downward trend in near-miss and conflict reduction shows that in very high density airspaces, the movement cost of positioning a UAS in order to follow a leader UAS becomes greater, as it is more likely to get too close to some other UAS in the process.

It should be noted though, that the high cost of flocking likely has to do with the random nature of these flight paths, and would likely be reduced significantly if flight paths were more orderly, or involved common start/waypoint locations, much like aircraft taking off from and landing at airfields. The dynamic nature of this algorithm’s flock forming and separation alleviates the problem of forming flocks too often, which makes it much more adaptable to different flight scenarios, as demonstrated by the different parameters produced by the optimization of the swarm and APF parameters. It is probable, however, that the optimal parameters are not simply a function of field size and number of UASs, but

<table>
<thead>
<tr>
<th># UASs</th>
<th>Field Size (m)</th>
<th>$d_{\text{cluster}}$</th>
<th>$t_{\text{Dubins}}$</th>
<th>$\text{Time}$</th>
<th>$\mu$</th>
<th>$\sigma_{\text{top}}$</th>
<th>$\sigma_{\text{top,follower}}$</th>
<th>$\beta_{\text{top}}$</th>
<th>$\beta_{\text{top,follower}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>500</td>
<td>20</td>
<td>10</td>
<td>6000</td>
<td>0.001250</td>
<td>0.0008850</td>
<td>0.00030</td>
<td>1.2</td>
<td>0.27</td>
</tr>
<tr>
<td>8</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>9650</td>
<td>0.001162</td>
<td>0.000750</td>
<td>0.00030</td>
<td>1.2</td>
<td>0.27</td>
</tr>
<tr>
<td>16</td>
<td>500</td>
<td>60</td>
<td>8</td>
<td>6000</td>
<td>0.000088</td>
<td>0.000100</td>
<td>0.000010</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>32</td>
<td>500</td>
<td>0</td>
<td>0</td>
<td>6000</td>
<td>0.000088</td>
<td>0.000050</td>
<td>0.000005</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>60</td>
<td>8</td>
<td>6000</td>
<td>0.000088</td>
<td>0.000050</td>
<td>0.000050</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>60</td>
<td>8</td>
<td>6000</td>
<td>0.000088</td>
<td>0.000050</td>
<td>0.000050</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>16</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>6000</td>
<td>0.000088</td>
<td>0.000050</td>
<td>0.000050</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>32</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>6000</td>
<td>0.000088</td>
<td>0.000050</td>
<td>0.000050</td>
<td>1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

TABLE I: Optimized Parameters
also of the regularity of the UASs’ starting positions and waypoints. Increased regularity in starting positions and waypoints may favor increased swarming behavior and different modifications to the APF due to the flight path reduction benefits that are offered by having UASs flock together rather than independently avoid each other.

VI. Conclusions

For a random-start, random-waypoint test, dynamic swarm APF coupled with genetic parameter optimization proves to dramatically increase the flight performance of UASs compared to not using collision avoidance. Optimization and swarm performance analysis show that swarming is not favored in some cases, and especially not favored in extremely high-density cases where the maneuvering required to get into a swarm formation is more likely to cause near-misses and conflicts. There are, however, cases where swarming is favored, as the reduction in APF coverage of the airspace and close formations used by swarms allow for more efficient and less conflict-prone flight patterns.

VII. Future Work

A. APF Method

This study was limited to using bivariate normal APF fields contained in an oval shape. Further work may be able to find the limits of the APF method itself, determining more general solutions and trends for optimal force and boundary functions as a function of UAS density given random waypoints. Additionally, the limit of APF effectiveness could also be studied, looking into the upper bound of UAS density before APF fails entirely. For this kind of study, particle motion models may be useful for determining upper bounds of APF usefulness and for determining whether or not this kind of system eventually crosses into a chaotic regime.

B. Swarming Configurations

Only a linear swarm pattern was tested extensively in this study. While there is significant literature supporting column formations as the most successful shape for robotic formation navigation, in the case of UASs where other considerations may be important, it would be worthwhile to explore those tradeoffs. For example, if fuel considerations were to be taken into account, further work could attempt to balance the path efficiency benefits gained by overlapping APFs with fuel efficiency benefits of formation flight that follows a V-shaped flocking pattern to reduce drag. Sensor resource allocation may also be considered if a cost-benefit analysis were done on the sensing capabilities of swarms as compared to independent UASs, since this study only considered waypoints achieved as a measure of effectiveness.

C. Non-Random Flight

This study utilized random starting positions for UASs that were only constrained by having positions that were not initially in conflict with each other, and random waypoints. While this provides a difficult test of collision avoidance, it does not really capture the nature of most flight trajectories—namely, trajectories that begin and end at specific places such as airfields or other launch sites. As such, the effectiveness of swarm intelligence was limited by the random nature of the flight paths. However, if given common sets of start and end points, and common sets of waypoints—need not be traversed in the same order—the effectiveness of swarming may increase significantly due to the similarity of flight patterns.

D. Air Traffic Control Workload

Further investigation may involve the effect of swarming behavior on air traffic control (ATC) system workloads. As swarms form, it becomes unnecessary for some aircraft to be in contact with ATC, as they are simply following their lead aircraft. As such, ATC may effectively treat swarms as single entities, thereby reducing their workload in managing the airspace without suffering negative impacts to flight performance. This effect is likely most significant in manned or centrally controlled UASs, and unlikely to impact decentralized UAS control systems such as the APF framework used here.

Acknowledgements

The authors would like to thank Dr. Gilbert Crouse for his advice on and support of the project, the other members of the 2012 National Science Foundation Research Experience for Undergraduates on Smart UAVs for their help with figuring out metrics, code, and simulation software, and the National Science Foundation and the United States Department of Defense for their generous support of the REU program.

References


