THE PROBLEM OF COLLISION AVOIDANCE IN UNMANNED AERIAL VEHICLES

TYLER YOUNG, THOMAS CRESCENZI, AND ANDREW KAIZER

ABSTRACT. In order for unmanned aerial vehicles (UAVs) to be widely adopted in civilian airspace, they must be capable of safe, autonomous flight. The problem of collision avoidance in UAVs is discussed in its theoretical foundations, and a formulation of the problem is given which clarifies what authors in the literature are concerned with when designing their algorithms. An overview is given of the methods of collision avoidance and path planning most widely represented in the literature, including A* ("A-star") search, geometric methods, mixed-integer linear programming (MILP), and artificial potential fields (APFs). Discussion of the strengths and weaknesses of each approach accompanies its description, as well as steps which may be taken to contend with any weaknesses.

1. INTRODUCTION

In recent years, interest in unmanned aerial vehicles (UAVs) has grown steadily in the private sector. Applications as varied as fertilizing crops, surveying land, and patrolling borders could employ UAVs to increase efficiency, lower costs, and keep humans from potentially dangerous situations [1].

As civil and commercial interest in UAVs grows, however, a number of obstacles remain which prevent their widespread adoption in nonmilitary situations (i.e., in civilian airspace). Primary among these obstacles, at least in the United States, is the requirement set forth by the Federal Aviation Administration (FAA) that for UAVs to be integrated into the national airspace, they must be at least as competent (capable of operating safely) as an equivalent human pilot *without* cooperative communication (such as commands from a human controller or information from neighboring aircraft) [3]. Many UAVs today are designed to operate primarily with the remote guidance of a human pilot, but to be widely certified for flight in the national airspace, regulatory agencies require the aircraft to be capable of safe autonomous operations in the event of a contingency. There are approximately 0.5 midair collisions per million flight hours in the United States [9]; to be cleared for flight in civil airspace, then, a UAV would have to demonstrate an ability to fly at least this safely autonomously.

Safe air operations are often discussed in terms of an operator's ability to "sense and avoid" potential conflicts. For an unmanned aircraft, this requires that the

Date: June 8, 2011.

onboard "pilot" possess some means of detecting nearby aircraft, as well as a means of altering its course in advance to avoid endangering itself or other aircraft.

A UAV needs not only to be able to alter its course, but it needs to do so in a way which is intelligible to human pilots observing it. By following standard flight procedures (such as guidelines for determining right of way), a UAV can take action to resolve conflicts while simultaneously communicating its intent to pilots around it (i.e., by the way in which it alters its trajectory). Additionally, the planned path must be flyable; it cannot call for abrupt 180° turns, or maneuvers which put too much stress on the airframe. The final primary consideration for a UAV's "sense and avoid" system, of course, is the aircraft's efficiency in negotiating the airspace, whether in terms of time required, fuel used, or danger avoided.

The most desirable path for an autonomous aircraft, then, is a flyable path of lowest possible cost which always maintains a safe distance from other aircraft. Having found this optimal path, the planning computer must also be able to re-evaluate its plans often in order to account for new craft in the airspace, changes in environmental conditions, and so on.

Collision avoidance, then, may be summarily divided into two parts: conflict detection (maintaining awareness of other aircraft and potential obstacles) and conflict resolution (maneuvering to avoid hazards in light of the system's knowledge from the conflict detection system). These two objectives are, respectively, to "sense" and "avoid" in as efficient a manner as possible.

1.1. **Computation of a Best Path.**

Thus, there is understandably a great deal of interest in creating UAVs able to autonomously plan paths which are optimized for both efficiency and safety to take the aircraft from an arbitrary location to its target. Creating such a system is clearly a hard problem, as there are a large number of constraints to satisfy, but it is in fact an even greater challenge than it appears at first glance: finding a best path is NP-complete [10, p. 869], meaning that the computation is among the most difficult problems which can be solved algorithmically. More precisely, NP-complete problems are computational problems for which no polynomial-time solution (a solution which can be computed in a reasonable amount of time on a large set of input) is known. These problems are widely believed to be intractable [8, p. 9]—that is, they are theoretically solvable, but as far as we know, finding a solution requires too much time to be useful in most cases.

In light of this, any method for computing a safe path for a UAV must compromise either optimality or time; one must settle either for a path which is good without being the best, or one must begin the computation long before its solution is needed. Because conditions in the air may change rapidly (due to changes in weather, the arrival of unexpected aircraft, systems malfunctions, and more), computing an optimal path ahead of time is often impossible.

1.2. **Formulation of the Problem.** Bearing in mind the requirements for autonomy, safety, efficiency of path, and speed of computation, we may formulate the problem of collision avoidance in unmanned aerial vehicles thus:

> *In a short amount of time, find a flyable path of minimum or near-minimum cost which maintains appropriate distance from all other aircraft.*

This formulation is shared with much of the extant literature on the subject; variations between different authors' formulations lie in their method of defining the cost, with some authors focusing on terrain costs (risks associated with flying over mountains, traveling too far without fuel, etc.), others on dangers associated with military operations in hostile airspace, others on the economic cost of fuel, and so on. In reality, the parameters used in calculating cost are of almost no importance to the search algorithms themselves.

Worthwhile to note is the fact that path planning for collision avoidance may focus either on static or dynamic obstacles. Although it is trivial to add static obstacles to a system designed for dynamic ones, it is not clear that the systems described in the literature designed for static obstacles might so easily be adapted to dynamic ones. The choice one makes in this area will determine just how short a "short amount of time" really is. When planning for static obstacles, a 30-second calculation of an optimal path may be sufficiently fast, whereas when dealing with dynamic obstacles, anything over 1 or 2 seconds may be unacceptable.

For our purposes, we are concerned exclusively with dynamic obstacles, in the form of other, independent aircraft.

2. Literature Review

A large body of work exists addressing how best to confront the apparently inescapable trade-off between computing time and optimality in the path planning problem. The approaches which are best represented in the literature may broadly be divided into the following categories:

- A* search, a method of computing the lowest-cost path from one node to another in a mathematical graph (i.e., a set of nodes connected by edges)
- Artificial potential fields (APF), which simulate potential fields (*à la* magnetic fields) wherein the agent and its goal are of opposite "polarity" (meaning that the agent will be "attracted" to its goal), while the agent and its obstacles are of the same polarity (meaning the agent will be "repelled" from them)
- Geometric approaches, which rely on vectors to calculate a point of closest approach between one point mass (the simplified model of an aircraft) and another, and to suggest a modification to those vectors which avoids conflict or collision

- Mixed-Integer Linear Programming (MILP), a method of solving problems which involve "both discrete decisions and continuous variables" [2], used in particular for creating a path optimized for a number of different constraints

2.1. **A* Search Methods.** A* (spoken as "A-star") search is a method of finding best paths in a mathematical graph or tree. For the purpose of path planning in aerial vehicles, the airspace must be divided into a grid, where each square in the grid represents some area of the airspace. These squares are, in actuality, nodes in the graph used by A* to represent the airspace. The algorithm, then, begins at the start node (corresponding to the aircraft's initial position) and considers the possible nodes to which the aircraft could travel. It rates each of these possible nodes, estimating the cost of the best possible final path which incorporates that node. This rating of the nodes which are open to travel from some other node is referred to as "branching." Each node rated during the branching process is added to a heap (a partially ordered data structure whose "top" or "front" element is the element of lowest cost), which stores the list of nodes open to consideration for inclusion in the optimal path.

Having finished the branching process from the start node, and beginning with the top node in the heap (i.e., the node estimated to yield the lowest-cost path to the goal), the algorithm repeats the process: it considers all nodes open to travel from the new node, selects the one it estimates to be best, and continues the search until it reaches the destination. If, at any point, it estimates that the path it is following will be of higher cost than a path leading from some previous node (i.e., a node in the open heap), it will return to that previous node and search from there.

As we have said, the algorithm's consideration of the nodes to which it could travel from its current position is referred to as "branching." The algorithm's estimation of the best possible path from a current position is referred to as "bounding" the future search. A* search, then, is a branch-and-bound method of coping with NP-completeness. It is also referred to as a best-first search, as it follows the "most promising" paths first [13, p. 6].

This algorithm's insight lies in the way in which it branches. Since the search problem is NP-complete, a simple in-order search of each possible path would take a prohibitively long amount of time. Instead, A* prioritizes open nodes based on its estimate of the best possible path from the node to the goal. This estimate is known as a heuristic. The estimated lowest cost of a path from the initial node to the goal which passes through a node n is denoted $f(n)$. The calculation of $f(n)$ is:

$$f(n) = g(n) + h(n),$$

where $g(n)$ is the known (previously calculated) best cost to go from the start node to node n, and $h(n)$ is the estimated cost of the best path from n to the goal node [8, p. 97].

The path calculated by the A* algorithm is guaranteed to be a global optimum whenever the heuristic function fulfills two requirements:

(1) $h(n)$ must never *overestimate* the cost from n to the goal, and
(2) $f(n)$ must be "consistent" in its method of estimation, meaning that a successor n' of a node n should never be estimated to have a higher cost than the step cost from n to n' plus the estimate $h(n')$ [8].

In this regard, A* has a clear advantage over other path planning algorithms. No algorithm can *possibly* generate better solutions, given the same search space and resolution; at best, other algorithms can generate equally good solutions. The disadvantage, of course, is that this optimality comes at the cost of computing time: in the worst case (i.e., using a poor or very general heuristic function), the computing time required grows *exponentially* with the size of the input, so large search spaces are completely impractical to work with.

Thankfully, the use of a good heuristic function (one which produces estimates very close to the actual cost without exceeding it) can reduce the time complexity of the problem from exponential to polynomial [13, p. 7]. This means that with the right heuristic, one could conceivably perform a search of a reasonably large problem space in seconds using no more than a small, lightweight computer onboard a UAV.

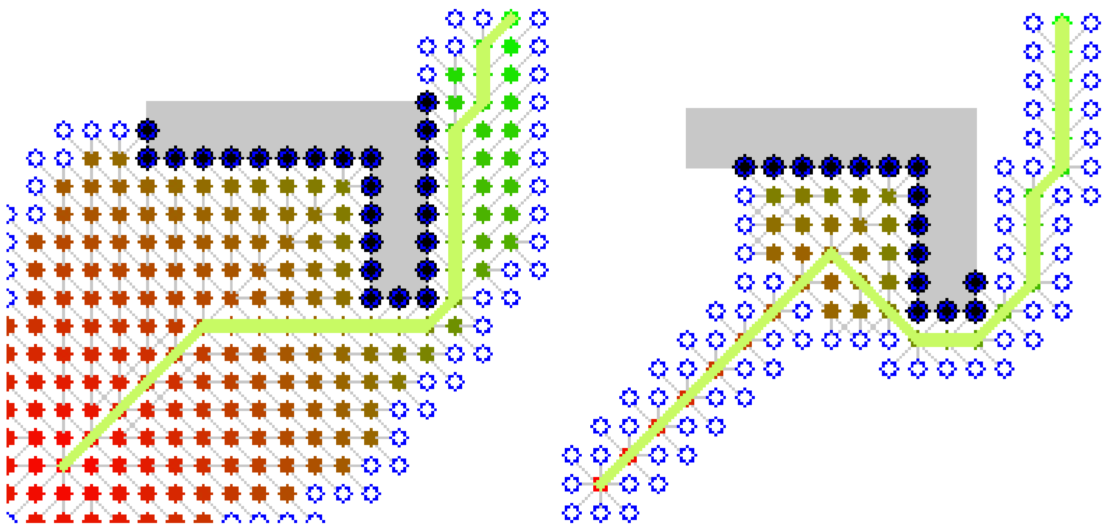

FIGURE 1. *"Classic" A* versus weighted A^*_ϵ search*
Image credit: Subhrajit Bhattacharya, released under the CC BY 3.0 license

2.1.1. *Reduced Complexity Varieties of A*.* In order to further reduce the time complexity of the search, a number of simplifications may be made to the algorithm. For instance, the A^*_ϵ (spoken "A-epsilon-star") algorithm searches several nodes in a row (all nodes within a fixed, positive ϵ value of the lowest-cost node) before each branching. This simplification is made on the (justified) assumption that the selection of the lowest-cost node is among the most time-consuming parts of the computation [11, p. 18].

This method trades the guarantee of optimality for reduced search time. As illustrated in Figure 1, A^*_ϵ considers far fewer nodes, which may result in a sub-optimal path. However, if strict optimality is not necessary, considering fewer nodes in this way is a viable way of significantly reducing the time complexity of the search.

The "Sparse A* Search" (SAS) method, designed for military flight planning by Robert Szczerba, Peggy Galkowski, Ira Glickstein, and Noah Ternullo [10], stands out among simplified versions of the A* algorithm. SAS significantly reduces the complexity of the search through a few clever assumptions, nearly all of which preserve the optimality of the path generated.

SAS improves the time complexity of the A* algorithm first by discarding from consideration large portions of the searchable area based on the limitations of the aircraft and pilot. For instance, in keeping pilot fatigue down, a minimum length is imposed on each "leg" of the path; paths with straight-line portions shorter than the minimum leg length are not considered. Likewise, in keeping the route flyable, a maximum turning angle is imposed on the search; paths which require turns

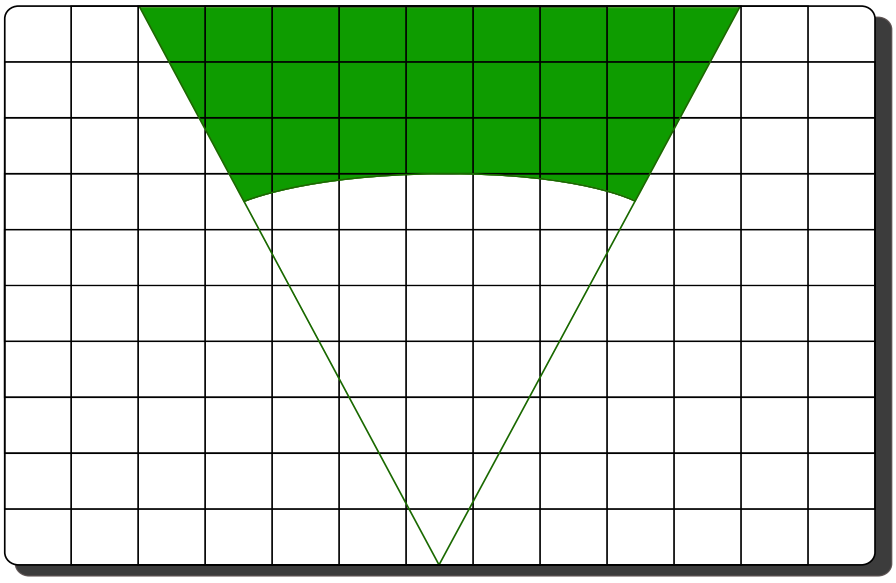

FIGURE 2. *Minimum leg length and maximum turning angle illustrated on a simple grid; only the shaded region is open to consideration.*

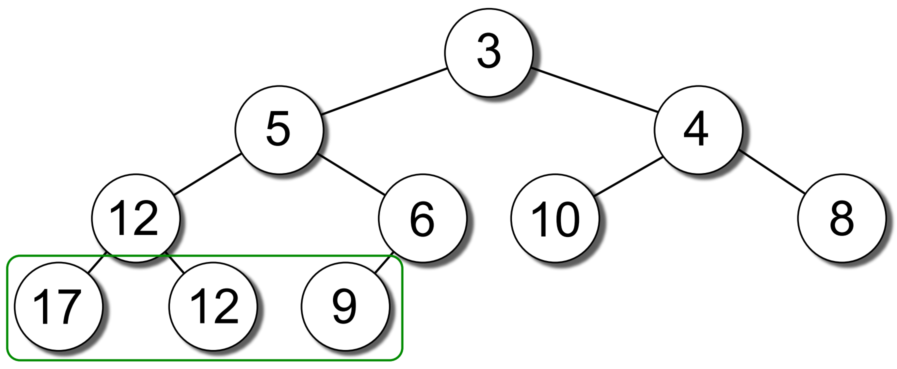

FIGURE 3. *A binary min-heap, whose lowest level will be pruned by SAS once full*
Image adapted from Wojciech Muła under the CC BY-SA 1.0 license

beyond the capability of the craft or the comfort of the pilot are not considered. Figure 2 illustrates these constraints on a small, simplified search area.

Further constraints on the path (such as a maximum route distance, maximum distance from a refueling station, or fixed heading when approaching the destination) may be considered to further reduce the area explored by the search. Introducing any of the constraints on SAS listed above will not result in a sub-optimal path; any *ideally* better path which is eliminated from consideration by the above parameters would *in fact* be undesirable anyway, due either to safety concerns or the physical limitations of the aircraft.

In their discussion of SAS, Szczerba et al. provide another method of reducing both time and space complexity (i.e., computing time and memory usage) [10]. Their method places a hard limit on the amount of memory used by the algorithm. As previously discussed, during the branching phase, A*'s heuristic function estimates the total cost of a path through a node and places that node on a heap (this heap is later used to select the next node from which to search). In order to limit memory usage, Szczerba et al. "prune" the heap after it reaches a predefined depth. When a new node is to be added to an already full heap (a heap of maximum depth), pruning occurs by:

(1) randomly selecting an element at the bottom of the heap (i.e., an element most likely to lead to a sub-optimal solution),
(2) replacing it with the node to be added, and
(3) restoring the heap structure (the ordering property), in an operation known as "up-heap" or "heapify-up," among other names.

For instance, in Figure 3, if the maximum queue depth were set to 4, the nodes highlighted in green would be open to pruning once the lowest level became full.

In addition to the obvious memory savings, pruning causes a smaller number of nodes to be searched overall, leading to a decrease in computing time.

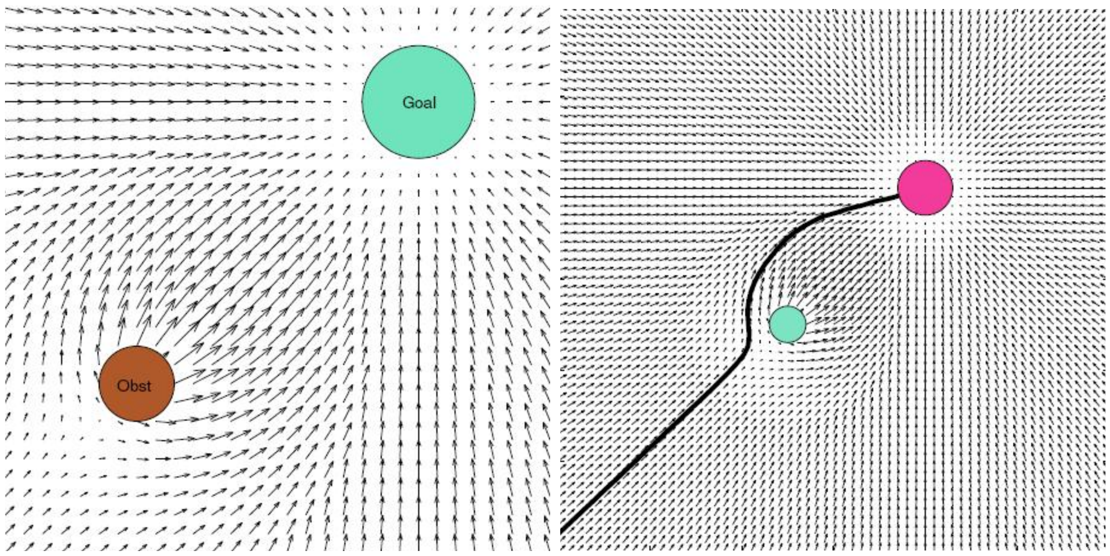

FIGURE 4. *Force vectors created by summing the potential fields, with the avoidance path taken by an aircraft around the obstacle*
Image credit: [12]

Unlike the other optimizations used by Szczerba et al., limiting memory usage in this way may result in a loss of optimality; nodes which are pruned from consideration may in fact lead to excellent solutions. With a relatively large maximum size, however, this will only occur through a failure of the heuristic, in an instance where it judges a possible path to be significantly less promising than it actually is. Nevertheless, in view of this possibility, Szczerba et al. recommend the maximum heap size be set as high as possible in order to prune as few nodes as possible from consideration.

2.2. Artificial Potential Field (APF) Methods.

The use of an artificial potential field (APF) in collision avoidance is adapted from particle physics and, in particular, the attractive and repulsive qualities of polarized charges. APF methods model the obstacles that an aircraft needs to avoid (such as other airplanes) as repelling point charges while modeling an aircraft's goal as an attractive point charge. When a plane comes into contact with the repelling APF (RAPF) of another plane, a force vector is calculated and the planes are given new waypoints which push them away from one another. Figure 4 illustrates the vectors calculated in the implementation of such fields.

The force function responsible for creating these fields must meet the following requirements:

(1) It must be continuous and differentiable.
(2) Its strength must increase inversely with the distance to obstacles.
(3) Its strength must decrease directly with the distance to goals [4].

Creating a force function suitable for a given situation is rather difficult. However, once the general mathematical function is developed, all calculation can be done by directly consulting the function's output. Thus, the time complexity of APF calculations is minimal (this is the significant advantage to using APFs over A*-based planning methods). While the functions themselves are complex, once they are programmed, one need only calculate the total force acting on a single plane in order to generate a force vector [4]. Once this vector is calculated, its value is input into a function that generates a new waypoint for the plane.

While the use of an artificial potential field is a very fast and computationally efficient technique for collision avoidance, the method is not without its drawbacks.

2.2.1. *Local Minima.* The first, and largest, of the problems associated with APFs is the existence of local minima. A local minimum is an area with a net force of zero [12]. This can happen when the aircraft is surrounded by obstacles, or when its goal is crossed by another aircraft (situations which are also problematic for A* search). However, local minima may be created in other ways as well. The most devastating of these conditions occurs when two aircraft are situated with respect to one another in a way that causes the force repelling the first plane from the second to be exactly the opposite of the force pulling the first plane to its goal [6]. In such a case, the first aircraft might "think" that it had already reached its goal, or worse, that it was safe to continue in a straight line, leading to a collision.

Numerous solutions to this problem have been proposed. The first is to cause the aircraft to default to an A*-determined path if it reaches a point in space with a net force of zero [12]. While using an A* path as a failsafe does overcome local minima, it also increases computation time significantly. Another solution to local minima is to use genetic algorithms to perfect the field calculations [6]. In the long run, this method reduces computation time. It does, however, incur the typical problems associated with genetic programming, including the many generations required to find an excellent function.

2.2.2. *Oscillatory Movements.* A second issue which must be addressed when using artificial potential fields is the problem of oscillatory movements. Such movements occur when the fields around an aircraft sum to forces moving in a wave or in small circles [12]. These circular movements cause the aircraft to take paths that are clearly suboptimal, or even to backtrack. While such movements may not cause collisions, they do greatly increase the time required for a plane to traverse a given path. This problem can be addressed by requiring a minimum distance between consecutive waypoints [12]. When the waypoints are spaced out by a set distance, most oscillatory movements can be eliminated. In all cases, however, the paths are not guaranteed to be optimal.

2.2.3. *Flyable Paths.* The fixes to the problems associated with potential fields discussed above work very well when working mainly with ground-based robots. When working with aircraft, however, it is quite difficult to ensure that the paths

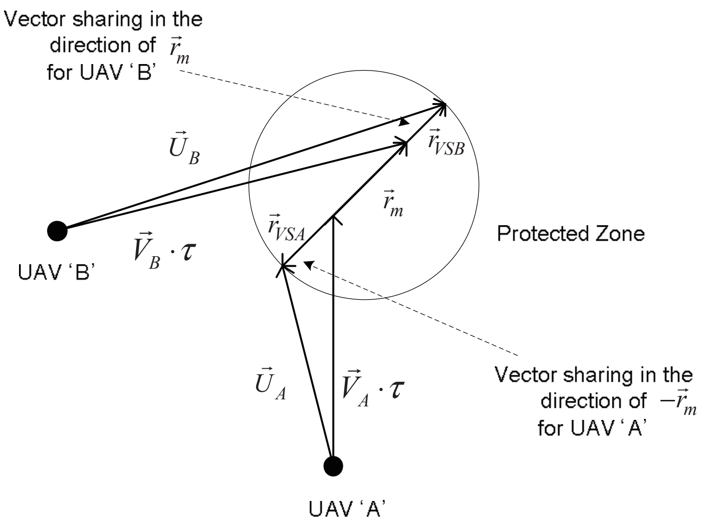

FIGURE 5. *Creation of an avoidance vector via the geometric approach*
Image credit: [5]

generated are flyable. Unflyable paths are those with too much oscillation or with waypoints outside the craft's maximum turning radius.

Under traditional methods of APF creation, it is difficult to take into consideration the aircraft's turning radius. Using a genetic programming approach to the creation of the force function, however, the potential fields may be tuned in such a way as to work within the aircraft's limitations after many generations [6].

Thus, while APFs are promising tools for collision avoidance in aircraft, they have many limitations that typically require a secondary programming method to be included on top of their simple force vector calculations.

2.3. **Geometric Approaches.** Geometric approaches to collision avoidance are built on 3-dimensional vectors which represent an aircraft as a point mass with a velocity (i.e., speed and heading) and location. An airplane's vector is calculated out to a specified time, and if it is predicted to come too close to another craft's vector, the aircraft are considered to be in conflict [5]. When a conflict is discovered, each airplane works to avoid the predicted conflict zone by calculating a new path known as their avoidance vector. The aircraft then adjust their speed, heading, and altitude as necessary to follow this avoidance vector.

Calculation of an avoidance vector is very fast; it requires even less computing time than the force vector calculation used in an artificial potential field. However, while this method works well with two aircraft given one waypoint each, it becomes more difficult to calculate a good avoidance vector as the airspace grows more crowded [5]. In addition to the problem of scaling, this method also hinges on the effective use of altitude control, which may not be desirable in all situations.

2.4. **Mixed-Integer Linear Programming (MILP) Methods.** Mixed-integer linear programming is a method of path planning which relies on a description of the problem in terms of a desired optimization of discrete decisions and continuous or integer variables. The mixed-integer linear program which results from this description of the problem may be solved using any of a number of extant software packages, both open-source and proprietary.

MILP has primarily been used in linear path planning around static obstacles. To adapt this strategy to collision avoidance with dynamic obstacles (e.g., other UAVs), one must first model the aircraft's flight paths in a linear, static way [7]. Performance characteristics of the aircraft such as their turning radii and acceleration capabilities must be rendered as linear, though they are not, of course, in reality. These limits on the aircraft's performance may effectively be represented as force magnitude limits [7]. For instance, the instantaneous turning rate ω may be limited in the program by adding the constraint:

$$\omega \leq \frac{f}{mV}.$$

Adding constraints in this way into the formulation of the problem ensures that the (optimal) path generated is a flyable one.

Once the flight characteristics of the aircraft are modeled in the input equations, the system of equations is solved in a way that creates paths which prevent collision

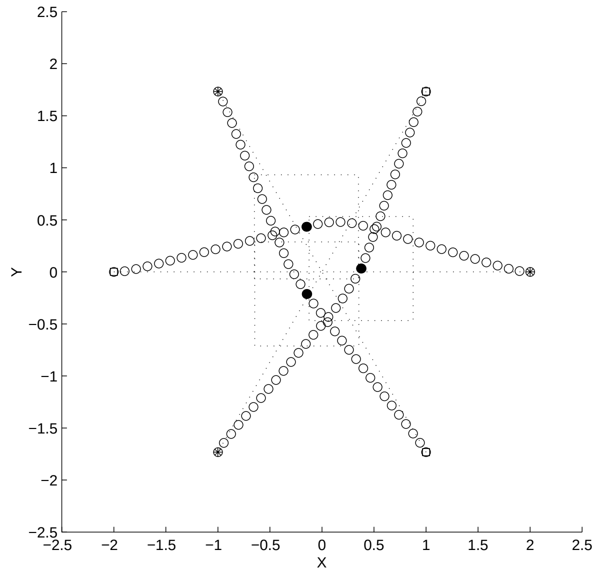

FIGURE 6. *Paths created for 3 planes. The black circles represent the three aircraft, the starred circles the goal waypoints, and the white circles the paths planned by the MILP algorithm*
Image credit: [7]

for all planes and produce optimal times to waypoints. These paths are calculated at once, from the aircraft's starting positions, and do not need to be calculated again except in the case of contingency.

The MILP approach benefits heavily from having only cooperative aircraft in the airspace. By giving each aircraft a path which is optimized for the "greater good," a globally optimal path may be used which is suboptimal (and thus not chosen otherwise) for an individual aircraft. For instance, Figure 6 shows three cooperative aircraft each being given a path modified slightly from a straight line in order to collectively reach their goals in minimal time.

MILP-planned paths are guaranteed to produce optimal paths. In practice, however, the time complexity renders problems of even moderate size intractable.

3. Discussion and Future Research

The path planning and collision avoidance algorithms reviewed above represent different means of dealing with the fact that the problem is NP-complete. Each algorithm sacrifices either speed of computation or optimality of path, and it is clear that each is suited to specific applications. Instances where computing power is very limited would benefit from APF or geometric approaches, whereas situations with greater computing power or less strict time requirements would wisely implement an A*- or MILP-based approach.

For our own future research, our preliminary testing indicates that a version of A* search drawing from the SAS heuristic of Szczerba et al. will be sufficiently fast while still giving an optimal solution. We will thus be implementing A* on a static, discretized model of our dynamic airspace, an idea which has been discussed in the literature but which, to our knowledge, has not been fully explored elsewhere.

References

[1] Federal Aviation Administration. Fact sheet – Unmanned aircraft systems (UAS), December 2010.

[2] Aerospace Controls Lab at Massachusetts Institute of Technology. Mixed-integer linear programming for control.

[3] Christopher Geyer, Snajiv Singh, and Lyle Chamberlain. Avoiding collisions between aircraft: State of the art and requirements for UAVs operating in civilian airspace. Technical report, Carnegie Mellon University Robotics Institute, 2008.

[4] Yun Seok Nam, Bum Hee Lee, and Nak Yong KO. An analytic approach to moving obstacle avoidance using an artificial potential field. In *1995 IEEE/RSJ International Conference on Intelligent Robots and Systems: Human Robot Interaction and Cooperative Robots*, volume 2, pages 482–487, August 1995.

[5] Jung-Woo Park, Hyon-Dong Oh, and Min-Jea Tahk. UAV collision avoidance based on geometric approach. *SICE Annunal Confrence*, pages 2122–2126, 2008.

[6] Yao-Hong Qu, Quan Pan, and Jian-Guo Yan. Flight path planning of UAV based on heuristically search and genetic algorithms. *31st Annual Conference of IEEE*, November 2005.

[7] Arthur Richards and Jonathan P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference*, volume 3, pages 1936–1944, 2002.

[8] S.J. Russell and P. Norvig. *Artificial intelligence: A modern approach.* Prentice Hall series in artificial intelligence. Prentice Hall, 3rd edition, 2010.

[9] Ryan J. Schaefer. A standards-based approach to sense-and-avoid technology. *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, pages 1–6, September 2004.

[10] R.J. Szczerba, P. Galkowski, I.S. Glicktein, and N. Ternullo. Robust algorithm for real-time route planning. *IEEE Transactions on Aerospace and Electronic Systems*, 36(3):869–878, July 2000.

[11] Karen Irene Trovato. *A* Planning in Discrete Configuration Spaces of Autonomous Systems.* PhD thesis, University of Amsterdam, September 1996.

[12] Kadir Firat Uyanik. Artificial potential fields, October 2009. RoboCup Small Size League project.

[13] J. van Tooren, M. Heni, A. Knoll, and J. Beck. Development of an autonomous avoidance algorithm for UAVs in general airspace. Technical report, EADS Defence and Security, Military Air Systems, 2007.