Decentralized Collision Avoidance Framework for Unmanned Aerial Vehicles

Andrew Cunningham
Computer Science Dept.
Rensselaer Polytechnic Institute
Troy, New York
Email: cunnia3@rpi.edu

Victoria Wu
Computer Science Dept.
University of Missouri - Kansas City
Kansas City, Missouri
Email: vcw7c5@mail.umkc.edu

***Abstract*—Collision avoidance for unmanned aerial vehicles (UAVs) is a vital issue when using multiple drones. Decentralized collision avoidance algorithms allow for greater range and independence when compared to a centralized approach; however, it also shifts the workload of detecting and resolving conflicts from a more powerful ground station onto individual agents, requiring both a computationally efficient algorithm and supporting framework. This paper presents a lightweight, decentralized collision avoidance framework for implementation on a low cost processor. An existing distributed artificial potential field algorithm is modified for use with this framework. Finally, the effectiveness of this framework is verified in simulation and tested using real UAVs.**

I. Introduction

As Unmanned Aerial Vehicles (UAV) become more widely utilized in both civilian and military endeavours, the need for an effective collision avoidance resolution algorithm becomes more pressing. Existing collision avoidance algorithms can be classified into two different types - centralized and decentralized. In a centralized collision avoidance algorithm, a central control unit will handle all collision avoidance computations. While centralized algorithms can find globally optimal solutions, they are often impractical because they require the presence of a ground station, which may limit the range that UAVs can travel. Decentralized algorithms provide an alternative in which all collision avoidance computations are run locally on each UAV.

The primary disadvantage associated with decentralized collision avoidance algorithms is that they usually produce sub-optimal solutions. This results from the fact that decentralized algorithms typically operate on incomplete information and only avoid collisions locally. However, this situation reflects reality because UAVs will rarely have perfect information about their environment and neighbors. Decentralized collision avoidance algorithms also offer several advantages. One key advantage that decentralized algorithms offer is that they are robust, so no single failure will disable the entire system. In addition, many decentralized algorithms are scalable due to the fact that the local interactions do not change much as more UAVs are added.

Decentralized collision avoidance algorithms are well suited for the inherent mobility and range of UAVs, but require a computationally efficient algorithm and surrounding framework. In this paper, we present a lightweight, distributed collision avoidance framework for use in UAVs. In section II, we first briefly define our problem statement, as well as the constraints imposed. In section III, we give a general overview of existing distributed collision avoidance algorithms that have similar constraints to our problem statement.

II. Problem Statement

Our main intent is to ultimately develop a distributed, lightweight collision resolution framework that is 1. ready for use with a low cost processor, and 2. able to modularly support various distributed collision resolution algorithms.

Our framework operates in conjunction with an existing ground control station and simulator developed in [1]. The ground control station is responsible for sending desired goal waypoints to simulated and real UAVs while receiving telemetry updates to track progress. The simulator is responsible for generating telemetry messages for simulated planes, and maintaining the state of all simulated planes.

Because our framework is meant to be implemented on hardware and not just in simulation, certain requirements are set to account for real world conditions. One of the main requirements is that the framework must be lightweight. The size of the drone used limits the weight of the payload, which in turn limits the type and power of processor that can be used. Because our aim is to implement a distributed framework, another limitation is that all processing for collision avoidance must be done locally. These two limitations combined require that any framework developed, as well as any algorithm used, must be as lightweight as possible to ensure effective real time performance.

In order to simplify the problem, we assume cooperative agents, as defined in [2]. Cooperative agents share flight information either passively by broadcasting, or actively by responding to a query. Non cooperative agents, on the other hand, do not actively share flight information and are considered obstacles. We constrain our problem to detecting and resolving conflicts with solely cooperative agents; in our case, UAVs passively broadcast their telemetry information consisting of speed, bearing, current and destination coordinates to other UAVs. To further simplify our problem, certain constraints are placed on the UAVs performance. The UAVs are constrained to a fixed speed and altitude, limiting the UAVs movement to turning right or left.

In order to test this framework, we modify and decentralize an existing APF algorithm implemented by an REU team from the previous year [3].

III. Literature Review

A framework for categorizing collision avoidance systems is presented in [4]. In this model, a collision avoidance system is conceptually broken down into different modules, see Figure 1 for an illustration. A collision avoidance algorithm can be designed by including the conflict detection and escape trajectory selection, or conflict resolution, modules. In this framework, a collision is defined as an instance in which two or more UAVs violate a minimum separation distance between each other and a conflict occurs when two or more UAVs will enter a collision sometime in the future. In order to avoid collisions, it is necessary to first detect the corresponding conflict. Therefore, conflict detection is a prerequisite of conflict resolution.

A. *Conflict detection and awareness*

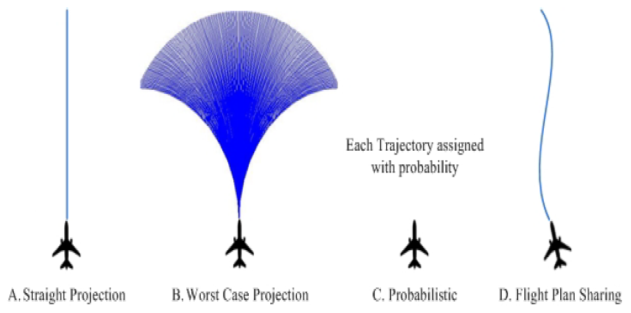

Fig. 2. The four main types of state projection [4]

1) State Projection: State projection refers to the method used to predict the future position of a UAV to detect conflicts. Figure 2 provides a visual representation of each type of state projection and [4] provides a description of each:

Nominal or straight projection traces the UAVs current trajectory and does not account for changing flight plans. A conflict is detected if a UAV will collide with another UAV given that they both continue on their current headings. However, straight projection is not reliable in cases where flight plans change frequently or for long projection times due to the fact that changed headings render previous predictions incorrect.

Worst-case projection traces all possible routes that a UAV could take and detects a conflict if any of these flight paths intersects with the flight path of another UAV. Projection distance must be limited in this approach to prevent oversensitivity in conflict detection and to minimize computational time devoted to prediction.

Probabilistic projection is similar to worst-case projection in that it considers multiple variations on current trajectory, but it adds the additional layer of considering the likelihood that the specific variation will occur.

Flight plan sharing allows for a more direct detection of conflicts as it involves having multiple UAVs sharing portions of their planned trajectories with other nearby UAVs. This method allows for accurate conflict detection, but presents the difficulty of actually transmitting these plans.

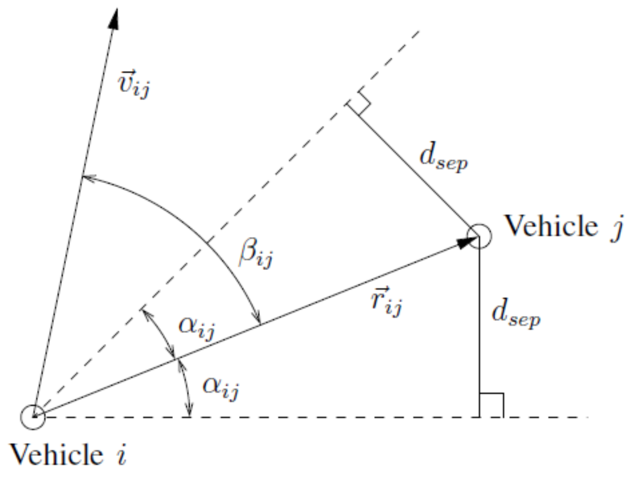

Fig. 3. The collision cone. A conflict is detected when velocity vector $\vec{v_{ij}}$ lies between the dotted lines [5]

2) Collision Cone: The collision cone is a pairwise conflict detection system in which a system of two moving agents is converted into a system with a single agent and a static obstacle [6]. Collision cones can be used to determine if a pair of agents are in conflict, assuming that the agents will continue on their current trajectories.

A conflict is detected if the relative velocity vector lies within the collision cone. Mathematically, a minimum seperation distance, d_{sep}, will be violated sometime in the future if $\beta < \alpha$ where:

$$\beta = \angle\vec{v} - \angle\vec{r},$$
$$\alpha = \arcsin(\frac{d_{sep}}{||\vec{r}||})$$
$\vec{r}$ = relative position vector
$\vec{v}$ = relative velocity vector

B. *Conflict Resolution*

1) Predefined: In a predefined or rule based collision avoidance algorithm, certain rules define how a plane should maneuver given a specific situation. One such implementation is detailed in [7]. Potential collisions are categorized according to the angle of the collision as seen in Figure 4; depending on the type of collision, each plane will execute a predefined escape maneuver. This decision is executed independently, with each plane detecting the collision from its point of view.

This approach is relatively simple and easy to implement, taking up only as much processing power is needed to determine the type of collision. However, this approach is inherently static, and thus may not guarantee collision avoidance for special cases. In addition, because the maneuvers are arbitrarily

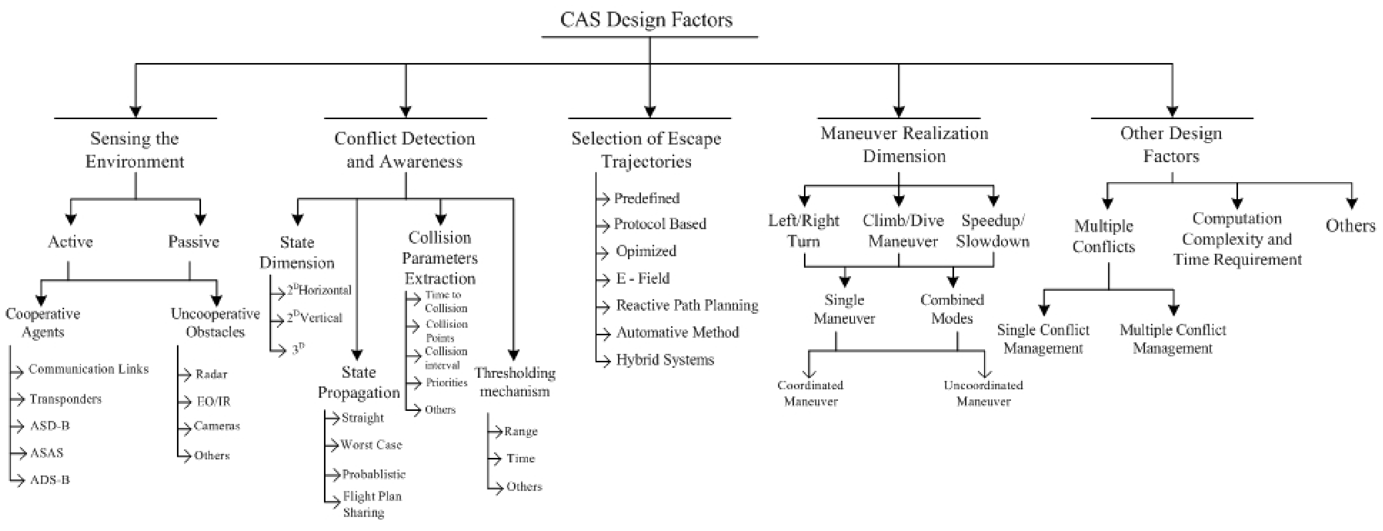

Fig. 1. Illustrates the main CAS design factors divisions with its subdivisions [4]

predefined with no room for modifications, this approach may result in an inefficient escape maneuver when a more efficient one is available.

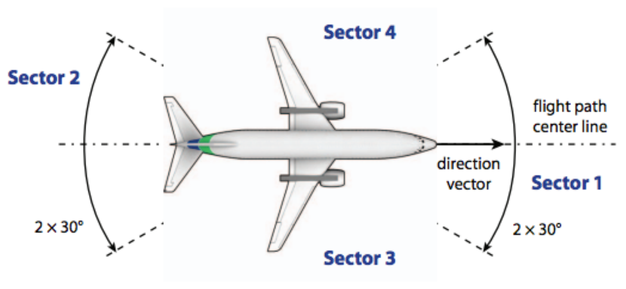

Fig. 4. Categorizing collisions based on angle. [7]

2) Protocol based: A negotiation based approach uses a predefined protocol to compromise and negotiate to achieve a mutually beneficial collision avoidance path. This approach assumes cooperative agents. The goal of such an approach is to find conflict-free flight paths for agents that are mutually beneficial.

One implementation of a protocol based collision avoidance algorithm is discussed in [8]. The algorithm consists of four main steps - detection, negotiation set generation, path optimization, and execution. For the purposes of explaining the steps, a pair of conflicting agents is assumed.

Detection - Based on the flight plan data received from other agents, an agent can determine if a conflict is about to occur. Once a conflict is detected, the process of generating a negotiation set begins.

Negotiation Set Generation - Each agent generates a set of possible maneuvers it could take. In the implementation described in [8], the author suggests six possible alternate trajectories - left/right turn, climb/descend, and speed up/slow down. Each possible trajectory is then marked with a utility value representing the desirability of the trajectory to the agent. The utility of a trajectory is determined by a utility function, defined as the difference between the cost of the trajectory and the cost of a fallback trajectory. Figure 5 lists a number of possible cost functions [9]. The cost function encodes the preferences of each agent, and is taken into account during negotiations. The agent marks each possible trajectory with its calculated utility value and sends them over to the other agent.

$$\begin{aligned}
D &= \text{total distance traveled} \\
\Delta A &= \text{total altitude changes during flight plan} \\
\Delta H &= \text{total heading changes during flight plan} \\
Cost &= D \\
Cost &= D + \Delta A \\
Cost &= D + \Delta H \\
Cost &= D + \Delta A + \Delta H
\end{aligned}$$

Fig. 5. Cost functions for generating utility values. [9]

Once an agent receives a set of possible paths from a conflicting agent, it will pair them up with its own set to make a new set of all possible pairs of paths both agents could take. The resulting set is the preliminary negotiation set, and is made of up of deals, or pairs of possible paths. All deals in the preliminary negotiation set are then validated, with those generating further conflicts removed; the remaining pairs of non conflict generating paths make up the negotiation set.

After generating the negotiation set, the next step is optimization. The best pair of paths must be found, taking into account the utility values associated with each deal. One possible way to find the best deal is to use the monotonic concession protocol (MCP), a incremental bargaining process described in [9] and [10]. If no deal can be found, then a fallback deal must be executed. One option for the fallback deal is assuming the worst case scenario and finding a guaranteed safe trajectory no matter what the other agent does.

The number of agents involved in the negotiation affects how and when these four steps are executed. For pairs of conflicting agents, each agent can either execute all four steps independently, or arbitrarily assign a master and slave to prevent repetition of work (see Figure 6). For conflicts involving more than two agents, the same pair-wise algorithm can be applied iteratively, or the master slave approach can be expanded to replace the master with a coordinator responsible for finding collision free paths for a party of planes.

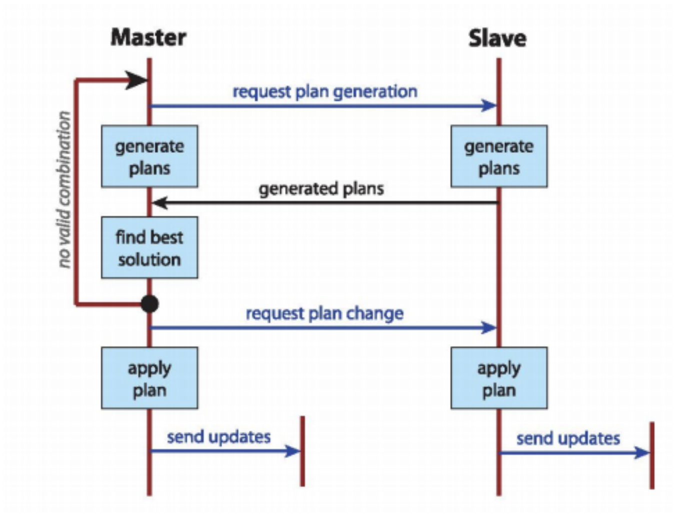

Fig. 6. Protocol based negotiations. [8]

This approachs main benefit is that it negotiates a mutually beneficial path, and the measure of the benefit can be fit to each agent. Each agent is able to inject its own preferences into the negotiating process. As a result, this approach can produce more efficient results and is more adaptable than a static, rule based method or a worst case method. However, the chattiness of the protocol requires more bandwidth for communications, and the algorithm itself requires more processing power depending on the number of agents involved.

3) Optimized: The optimization approach to selecting escape trajectories involves formulating the problem in terms of constraints and cost functions. Typically, the constraints are coupled with kinematic models of the planes and the cost functions are derived from a variety of metrics. An escape trajectory can then be found by minimizing the cost function subject to the aforementioned constraints utilizing any optimization technique. Commonly used optimization techniques for this purpose include genetic algorithms, expert systems and fuzzy control.

Genetic algorithms search the solution space of possible paths using techniques inspired by evolution. Genetic algorithms generally follow a set process, described as follows: The first step in most genetic algorithms is to initialize a population, or a random set of initial solutions. The algorithm then enters a loop in which it will iteratively mutate the most

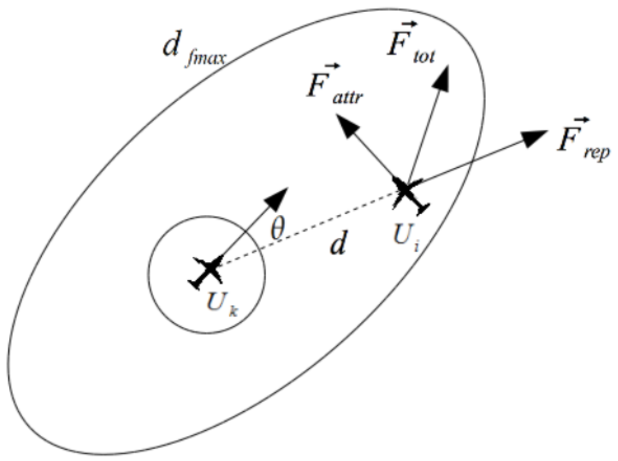

Fig. 7. Calculating virtual forces acting on a UAV [12]

successful members of the population and continue to do so until a suitably fit solution is found [3].

Expert Systems make use of rule bases to decide how to categorize and resolve conflicts. However, these systems require a large number of rules to recognize all type of conflicts and the resolutions to the types of conflict. Once the rule base is established, its use of rules makes it relatively easy to understand. [11]

Fuzzy control systems are modelled on fuzzy logic and follow a certain process. In the first step, fuzzy values are created from numeric or crisp input. The fuzzification of crisp input entails assigning descriptive tags to inputs that meet certain requirements. For example, an altitude of 500-1000 meters could be classified as high. A fuzzy control system can then apply a fuzzy rule base to this input to create a fuzzy output. Finally, this output can be defuzzified to produce a flight plan. [**?**]

4) E Field: The E-Field or APF (artificial potential field) approach to trajectory calculation involves modelling UAVs and their waypoints as charged particles. UAVs are assigned the same charge so that they repel each other and waypoints are assigned an opposite charge so that their corresponding UAVs are attracted to them. UAVs can then implicitly plan paths by summing attractive and repulsive forces to create the next desired heading.

Several factors need to be considered when designing an APF system. One consideration is the consideration of field shape. While in reality, charged particles can affect other charged particles over any distance, there is little reason to implement such a field for UAVs in the context of collision avoidance. If the APFs were to extend to a great distance, the contribution of force to the affected UAV would be negligible and would only use more computation time. Therefore, it is often the case that limit functions are created to define a finite field shape around a UAV. Another design consideration that should be taken into account is the field function. Once a UAV enters another UAVs APF, the field function determines

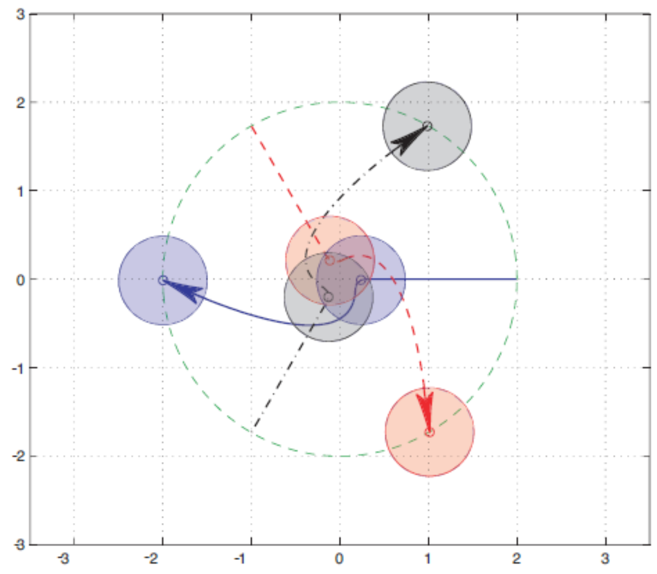

Fig. 8. Gyroscopic forces allow three agents to spin free of their initial configuration and reach their antipodal destinations [16]

the magnitude (and possibly direction) of the repulsive force [13].

There are several issues that arise when APFs are used for collision avoidance on unicycle UAVs. If only APFs are utilized, then there is no method for resolving a head on collision between two UAVs with waypoints located behind each other. This results from the fact that there exists no force in this configuration that would cause the UAVs to turn away from each other. Another issue is looping. Looping occurs when a UAV circles its goal waypoint and is unable to reach it due to turning constraints. In a pure APF approach, the attractive force of a waypoint can provide the centripetal force needed to initiate looping. It is possible to resolve both of these issues by explicitly identifying the situation and then applying the appropriate maneuver(s) to resolve the issue [12].

APFs are a popular research topic and have been utilized in many different ways to implement collision avoidance. [12], [14] and [15] all make use of APFs to govern swarming behavior. Swarming, in this context, refers to the grouping of UAVs in a geometric fashion in response to a specific situation. Specifically, [12] made use of flocking behaviours to limit competition between UAVs that had similar destinations. [16] presents gyroscopic forces as another extension to APFs for collision avoidance. In [16], repulsive forces are replaced with a gyroscopic force that activates whenever two agents come within a certain distance of each other.

IV. Conclusion

The conclusion goes here.

Acknowledgment

References

[1] J. Holt, "Comparison of aerial collision avoidance algorithms in a simulated environment," Ph.D. dissertation, Auburn University, 2012.

[2] U. A. V. Roadmap, "Roadmap 2002-2027," *US DoD, December*, 2002.

[3] H. Siu, M. Figueroa, J. Holt, and S. Biaz, "A dynamic swarm approach to artificial potential field collision avoidance technical report# csse12-02."

[4] B. Albaker and N. Rahim, "A survey of collision avoidance approaches for unmanned aerial vehicles," in *Technical Postgraduates (TECHPOS), 2009 International Conference for*. IEEE, 2009, pp. 1–7.

[5] E. Lalish, K. A. Morgansen, and T. Tsukamaki, "Decentralized reactive collision avoidance for multiple unicycle-type vehicles," in *American Control Conference, 2008*. IEEE, 2008, pp. 5055–5061.

[6] Z. A. Daniels, L. A. Wright, J. M. Holt, and S. Biaz, "Collision avoidance of multiple uas using a collision cone-based cost function."

[7] D. Sislak, M. Rehak, M. Pechoucek, D. s. Pavlicek, and M. Uller, "Negotiation-based approach to unmanned aerial vehicles," in *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on*. IEEE, 2006, pp. 279–284.

[8] D. Šišlák, M. Pěchouček, P. Volf, D. Pavlíček, J. Samek, V. Mařík, and P. Losiewicz, "Agentfly: Towards multi-agent technology in free flight air traffic control," in *Defence Industry Applications of Autonomous Agents and Multi-Agent Systems*. Springer, 2008, pp. 73–96.

[9] S. Wollkind, J. Valasek, and T. R. Ioerger, "Automated conflict resolution for air traffic management using cooperative multiagent negotiation," in *AIAA guidance, navigation, and control conference*, 2004, pp. 16–19.

[10] G. Zlotkin and J. S. Rosenschein, "Negotiation and task sharing among autonomous agents in cooperative domains," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. Citeseer, 1989, pp. 912–917.

[11] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 1, no. 4, pp. 179–189, 2000.

[12] M. Hromatka, J. West, J. Holt, and S. Biaz, "A fuzzy logic approach to collision avoidance in smart uavs technical report csse12-05."

[13] J. Ruchti, R. Senkbeil, J. Carroll, J. Dickinson, J. Holt, and S. Biaz, "Uav collision avoidance using artificial potential fields," Technical Report# CSSE11-03. Auburn University, Tech. Rep., 2011.

[14] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 500–505.

[15] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 73–80.

[16] L. Barnes, W. Alvis, M. Fields, K. Valavanis, and W. Moreno, "Swarm formation control with potential fields formed by bivariate normal functions," in *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*. IEEE, 2006, pp. 1–7.

[17] D. E. Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber, "Collision avoidance for multiple agent systems," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 1. IEEE, 2003, pp. 539–543.