Circular Flight Patterns for Dronevision

Shuqin Zhu
University of Southern California
Los Angeles, USA
shuqinzh@usc.edu

Shahram Ghandeharizadeh
University of Southern California
Los Angeles, USA
shahram@usc.edu

ABSTRACT

This paper presents the design and implementation of a circular flight pattern for use by a 3D multimedia display, a Dronevision (DV). A DV uses drones configured with light sources, Flying Light Specks (FLSs), that are battery powered. The flight pattern enables a swarm of FLSs to enter an opening, granting them access to the charging coils to charge their batteries. We present two algorithms for an FLS to travel from its current coordinate to rendezvous with its assigned slot on the flight pattern, Shortest Distance (SD) and Fastest Rendezvous Time (FRT). In addition to quantifying the trade-off associated with these algorithms, we present an implementation using a swarm of Crazyflie drones with Vicon localization.

Holodecks Reference Format:
Shuqin Zhu and Shahram Ghandeharizadeh. Circular Flight Patterns for Dronevision. Holodecks, 2(1): 1-11, 2024.
doi:10.61981/ZFSH2404

Holodecks Artifact Availability:
See https://github.com/flyinglightspeck/CircularFlightPattern for our open source software implementation and its data set. A video demonstration of our implementation using a swarm of Crazyflie drones with Vicon is available at https://youtu.be/H60r2oTPB4k.

1 INTRODUCTION

A Dronevision (DV) is a non-immersive 3D multimedia display detailed in [4]. A swarm of cooperating miniature drones configured with RGB light sources, Flying Light Specks (FLSs), to illuminate 3D point clouds and provide haptic interactions [30]. Figure 1 shows a DV illuminating a rose with a falling petal captured using a depth camera. The ceiling of the DV consists of wireless charging coils used to charge the battery of FLSs with a fixed flight time.

STAG [31] is an algorithm that continuously charges FLSs by staggering their battery flight time. It minimizes the number of charging stations. In addition, it minimizes the number of FLSs that are in transit from an illumination to the charging coils. This number may range from 55 to 218 FLSs with today's batteries and the Rose point cloud requiring 65K FLSs [31].

A challenge is how a swarm of tens of FLSs may fly through an opening of the DV to access the charging coils. This is non-trivial for several reasons. First, the system must consider downwash [8, 14, 26, 66, 87, 94], a region of instability caused by the flight of one FLS that adversely impacts other FLSs entering this region, e.g., loss of control or unpredictable behavior. Second, the flight pattern may be at an arbitrary angle θ relative to the z-axis. Third, FLSs should occupy a moving slot in the flight pattern while minimizing either the traveled distance or the amount of time required to occupy the slot for arbitrary θ angles. This paper presents the design, simulation, and implementation of a circular flight pattern. Figures 2a and 2b show a horizontal and a vertical orientation of a circular flight pattern accessing an opening. With 2a (2b), the opening at the top (back) provides FLSs with access to the charging coils on top (back) of the Dronevision.

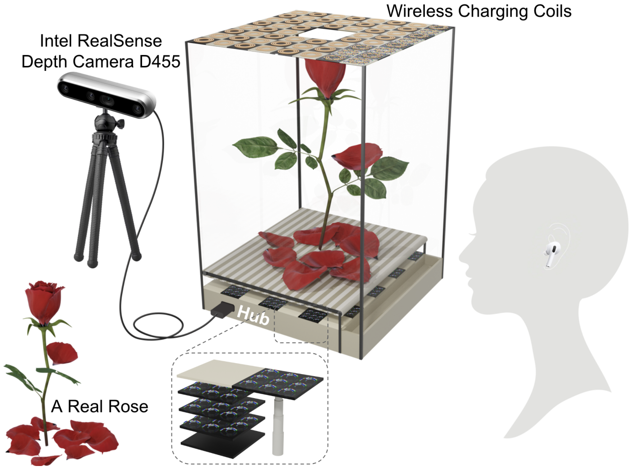


Figure 1: A Dronevision, DV [4].

Definition 1.1. A *flight pattern* is a formation consisting of a fixed number of slots where all slots maintain a general pattern or shape with a fixed distance between two consecutive slots. This is commonly termed a rigid formation [13, 81]. Slots travel at a fixed speed and in the same direction. Once an FLS occupied slot is below the opening, the occupying FLS flies through the opening and relinquishes its slot.

This paper focuses on circular flight patterns with a fixed radius R. The distance between the slots is dictated by downwash. For example, with quadrotor representing an FLS, its downwash is represented as a sphere with a fixed radius r [37, 50]. The sphere must be inclusive of the drone. We set the distance between two consecutive slots to be 2r since two consecutive slots may be occupied by an FLS. Slots move either clockwise or counter clockwise.

A centralized scheduler hosted on the Hub [31], see Figure 1, of the DV may maintain the coordinates of the slots on the flight pattern and assign a vacant slot to an FLS. This raises the following research questions: First, what algorithms enable an FLS to compute a path from its current coordinate to rendezvous with its assigned

---

Proceedings of the Holodecks Foundation, Vol. 2, No. 1.
doi:10.61981/ZFSH2404

slot? Second, what are their tradeoff? This paper provides an answer to these two questions.

We present two algorithms, Shortest Distance (SD) and Fastest Rendezvous Time (FRT), for an FLS to rendezvous with its assigned slot. As implied by their names, SD computes the shortest distance while FRT computes the fastest time. We provide analytical models, simulation studies, and an implementation of the circular flight pattern. We use the simulation model to quantify the tradeoff associated with SD and FRT. The implementation consists of a swarm of 5 Crazyflie drones using the Vicon localization technique. Figures 3a and 3b show the opening of this implementation from a corner and the bottom, respectively. It verifies the correctness of the analytical models and the simulation model that embodies them. Its output is almost identical to that of the simulation model, see Figure 4 and video demonstration.

Contributions of this study include:

- Design and implementation of circular flight patterns at any angle θ relative to the z-axis. It realizes horizontal (θ=0°), vertical (θ=90°), and in-between θ values. (Section 2.)
- θ neutral algorithms to compute the Shortest Distance (SD) and the Fastest Rendezvous Time (FRT) for an FLS to rendezvous with its assigned slot. (Section 2).
- Analytical models, a simulation study, and an implementation of the flight pattern and the SD and FRT algorithms. The implementation uses a swarm of Crazyflie drones with the Vicon localization system. Click demonstration for a video.
- An evaluation of SD and FRT, highlighting their tradeoffs. (Section 4).
- We open source our software implementation and its data set at https://github.com/flyinglightspeck/CircularFlightPattern.

The rest of this paper is organized as follows. Section 2 details the design of a circular flight pattern, and SD and FRT algorithms. Section 3 presents an implementation using Crazyflie drones. Section 4 evaluates SD and FRT, quantifying their tradeoffs. Section 5 presents related work. Brief conclusions are presented in Section 6.

2 A CIRCULAR FLIGHT PATTERN SLANTED θ DEGREES

A single layer circular flight pattern locates on a plane. It has a fixed center P_C, a radius R, and N slots. The slots are rotating either clockwise or counter-clockwise at the linear speed S_{slot}. A normal vector $\overrightarrow{V_{Norm}}$ defines the angle θ between the flight pattern and the z-axis, see Figure 5. The normal vector may be perpendicular to the ground (θ=0, horizontal, see Figure 2a), parallel to the ground (θ=90, vertical, see Figure 2b), or slanted at θ degrees relative to the z-axis. See Figure 5.

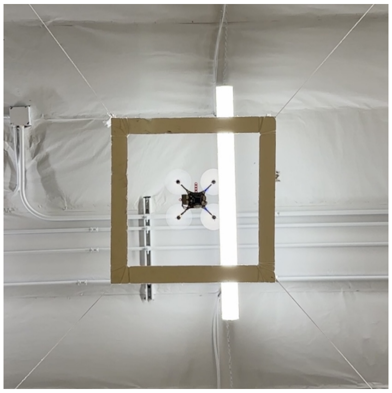
To accommodate downwash, the distance between two consecutive slots d_s is required to be greater than or equal to twice the radius r of the sphere that models a drone and its downwash, $d_s \geq 2r$. The maximum number of slots is $N = \frac{2\pi R}{2r}$, $d_s = 2r$. The numerator is the circumference of the flight pattern. The denominator $2r$ is the minimum allowed distance between two slots. Obviously, a flight pattern may consist of fewer slots n, $n < N$. In this case the distance between slots may be larger than the required minimum, $d_s = \frac{2\pi R}{n}$, $d_s \geq 2r$.

(a) **A horizontal flight pattern,** $\theta = 0$.

(b) **A vertical flight pattern,** $\theta = 90$.

Figure 2: Dronevision with a circular Flight Pattern: the horizontal (θ=0°) and vertical (θ=90°) alignment makes the charging coils accessible to FLSs.

(a) Corner view of the opening.



(b) Bottom view of the opening.

Figure 3: the opening of an implementation with a swarm of Crazyflie drones

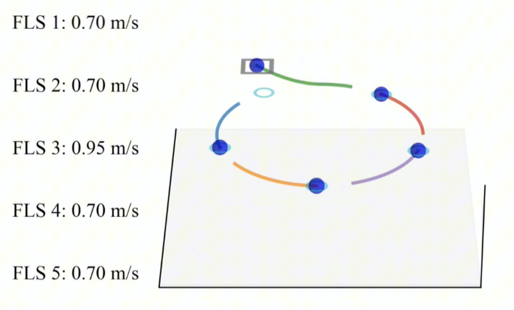


Figure 4: Visualization of simulation.

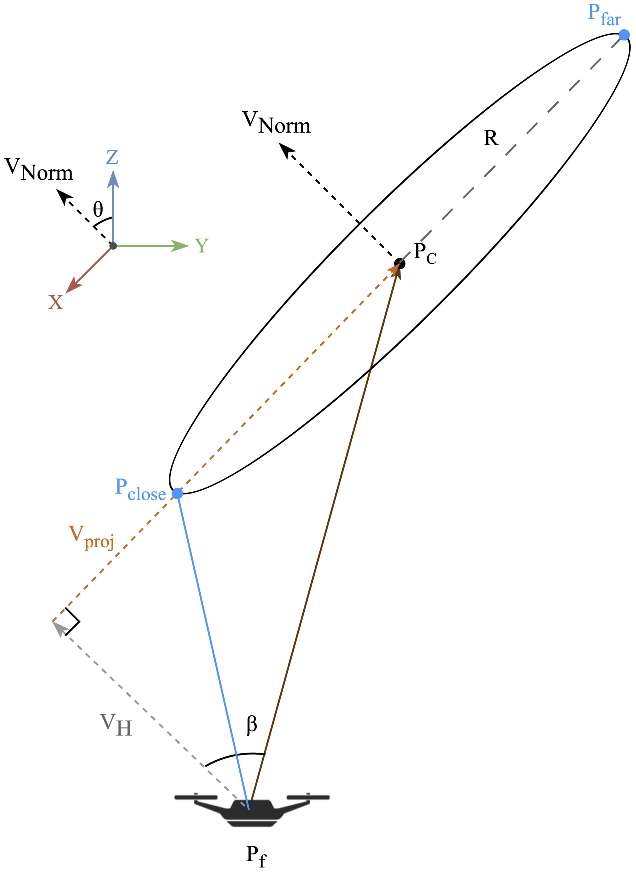


Figure 5: An FLS relative to a flight pattern.

The slots on the flight pattern are assigned to FLSs. An FLS travels from its current coordinate P_f to rendezvous with its assigned slot. It matches the speed of the slot (flight pattern) to occupy it. Meeting these requirements is non-trivial because both the slot and the FLS are moving. A design requires an answer to the following: What is the coordinate of an assigned slot once the FLS arrives at the flight pattern? What is the shortest distance for the FLS to rendezvous with its slot? What is the fastest time for the FLS to rendezvous with it slot? An answer to these questions is a tradeoff between distance and time. The following sections quantify this tradeoff. Section 2.1 provides an algorithm that computes the location of a slot after some time interval for any value of θ. Subsequently, Sections 2.2.2 and 2.2.3 present SD and FRT algorithms that trade distance for time. These algorithms depend on a velocity model. One is presented in Section 2.2.4.

2.1 Slot Coordinate as a Function of Time

The slots of a flight pattern are logical. Each is identified by a unique id. At its initialization time, the scheduler determines the radius R of the flight pattern, the number of slots N, the coordinates of each slots and assigns them a unique id, and the speed of the slots. Algorithm 1 computes the coordinates of the Slot sID after t time units. It assumes each slot of the flight pattern is indexed from 1 to N, $1 \leq sID \leq N$. Its input is a Flight Pattern object FP, the identity of a slot sID, and t time units from now. The object FP has a value for the following variables: $\overrightarrow{V_{Norm}}$ that defines the angle θ, radius R, speed S_{slot}. The output of Algorithm 1 is the coordinate of the slot sID after t time units. The first step of this algorithm computes a unit vector by dividing $\overrightarrow{V_{Norm}}$ with its length. Step 2 computes the rotation angle ϕ of a slot after t time units. Steps 3-6 calculate the rotation matrix for the slot. Step 7 identifies the current location of slot sID. Step 8 applies the rotation matrix to the vector from the slot to the center of the flight pattern. The resulting vector $\overrightarrow{V_R}$ is from the center of the flight pattern to the location of the slot after t time units. Step 9 converts the vector $\overrightarrow{V_R}$ to a 3D coordinate of the Slot sID.

```
Algorithm 1: GetSlotPosition($FP, sID, t$)
1 $[x_N, y_N, z_N] \leftarrow \frac{FP.V_{Norm}}{\|FP.V_{Norm}\|}$
2 $\phi \leftarrow \frac{FP.S_{slot}}{FP.R} \times t$
3 $a \leftarrow \cos(\phi)$
4 $b \leftarrow \sin(\phi)$
5 $c \leftarrow 1 - \cos(\phi)$
6 $M_{Rotation} \leftarrow$
    $\begin{bmatrix} a + x_N^2 c & x_N y_N c - z_N b & x_N z_N c + y_N b \\ y_N x_N c + z_N b & a + y_N^2 c & y_N z_N c - x_N b \\ z_N x_N c - y_N b & z_N y_N c + x_N b & a + z_N^2 c \end{bmatrix}$
7 $P_{slot} \leftarrow FP.slots[sID]$
8 $\overrightarrow{V_R} \leftarrow M_{Rotation} \overrightarrow{FP.P_C P_{slot}}^{\mathrm{T}}$
9 return $P_C + \overrightarrow{V_R}$
```

2.2 FLS Rendezvous with a Slot: SD and FRT

An FLS f in a Dronevison must rendezvous with its assigned slot, i.e., arrive at the same coordinates and at the same time as its slot. The Dronevision display space is well defined with no obstacles. It is for use in an indoor setting with no environmental factors such as wind. Here, we focus on a single layer circular flight pattern, consisting of a fixed number of slots N that are rotating counter-clockwise at a fixed speed. An FLS is provided with a slot, the current coordinates of the slot, and the speed at which the slot is moving. We present two rendezvous algorithms for use by the FLSs. The first, Shortest Distance (SD), computes the path with the shortest travel distance. The second, Fastest Rendezvous Time (FRT), computes the path with the fastest travel time. Both assume the current location of an FLS, P_f, is a known coordinate. SD uses analytical models with the point on the flight pattern closest to P_f. FRT uses dynamic programming by searching the range between the closest and the farthest point on the flight pattern to P_f. Both techniques assume (a) the velocity model of Section 2.2.4 and (b) an FLS is able to travel at either the same speed or a faster speed than the speed of a slot, $S_{max} \geq S_{slot}$.

The next section describes how to compute the closest and farthest rendezvous point to the coordinate P_f of an FLS. These are denoted as P_{close} and P_{far}, respectively. While SD of Section 2.2.2 uses P_{close}, FRT of Section 2.2.3 uses both P_{close} and P_{far}.

2.2.1 *Closest & Farthest Point.*

Flight patterns may have different orientation in the global coordinate system. A general counter-clockwise rotating flight pattern, FP, lies on a plane Π. The included angle between Π and the positive direction of the z-axis (vector $[0, 0, 1]$) of the global coordinate system is θ degree. Let the normal vector of the flight pattern be $\overrightarrow{V_{Norm}} = [x_N, y_N, z_N]$, where $|\overrightarrow{V_{Norm}}| = 1$. Similar to previous, let $P_f = [x_f, y_f, z_f]$, and $P_C = [x_c, y_c, z_c]$. See Figure 5. Then, $\overrightarrow{P_f P_C} = [x_C - x_f, y_C - y_f, z_C - z_f]$. The distance from P_f to Π can be described as the length of the vector $\overrightarrow{V_H}$, where $\overrightarrow{V_H}$ start from P_f and end at a point on Π, and $\overrightarrow{V_H}$ is perpendicular to Π, meaning $\overrightarrow{V_H} \parallel \overrightarrow{V_{Norm}}$. Hence, $V_H = |\overrightarrow{P_f P_C}| \times \cos(\beta) \times \overrightarrow{V_{Norm}}$, where cosine value of the included angle between $\overrightarrow{P_f P_C}$ and $\overrightarrow{V_{Norm}}$, $\cos(\beta) = \frac{\overrightarrow{P_f P_C} \cdot \overrightarrow{V_{Norm}}}{|\overrightarrow{P_f P_C}| \times |\overrightarrow{V_{Norm}}|}$. The projection of vector $\overrightarrow{P_f P_C}$ on the plane Π: $\overrightarrow{V_{proj}} = \overrightarrow{P_f P_C} - \overrightarrow{V_H}$, and the closest point position $P_{close} = P_C - R \times \frac{\overrightarrow{V_{proj}}}{|\overrightarrow{V_{proj}}|}$, the farthest point position $P_{far} = P_C + R \times \frac{\overrightarrow{V_{proj}}}{|\overrightarrow{V_{proj}}|}$

The simple horizontal and vertical flight patters of Figures 2a and 2b are a special case with θ set to 0° and 90°, respectively.

2.2.2 *Shortest Distance (SD) Path.*

This technique minimizes the distance d traveled by an FLS to its assigned slot. It computes a straight line with a starting point set to the FLS's coordinates, an end point set to the closest point P_{close} on the flight pattern, some wait time δ at the starting point, and the FLS flight duration. The duration is dictated by the FLS velocity model to travel d with the time required to decelerate to match the speed of a slot in the flight pattern.

When d is such that an FLS may reach its maximum speed, we consider 3 variants of SD. Their key difference is the duration of wait time δ and the speed used to travel. All variants consider the time to either accelerate or decelerate to match the speed of a slot in a flight pattern.

Variant 1 requires an FLS to travel at its fastest speed S_{max} by adjusting the duration of δ. When compared with the other alternatives, it maximizes the wait time δ_{max}. Its highest speed must be as fast as the speed of the slots (S_{slot}) in the flight pattern or faster, $S_{max} \geq S_{slot}$. When $S_{max} > S_{slot}$, Variant 1 considers the time to decelerate to match the speed of its assigned slot S_{slot}.

Variant 2 requires an FLS to leave its current coordinates as soon as possible. This means it may travel at its slowest speed $S_{slow} > 0$ to rendezvous with its slot by adjusting the duration of δ. It minimizes the wait time δ_{min} when compared with the other variants. If S_{slow} is faster or slower than S_{slot}, Variant 2 considers the time to decelerate or accelerate to match the speed of its slot.

Variant 3 is a hybrid that uses a speed in between the minimum S_{slow} and the maximum S_{max} speed. It is motivated by the observation that the flight time of an FLS on its remaining battery lifetime may be maximized if it travels at a pre-specified speed $S_{Battery}$. Assuming this speed is somewhere between the minimum and maximum speed ($S_{slow} \leq S_{Battery} \leq S_{max}$) then, this technique's wait time δ_{hybrid} is somewhere between the minimum and maximum wait times, $\delta_{min} \leq \delta_{hybrid} \leq \delta max$. Note that Variant 3 is the same as Variants 1 and 2 when S_{Bat} equals S_{max} and S_{slow}, respectively.

Algorithm 2 implements SD by calculating the position and time for an FLS f to rendezvous with its assigned slot and its wait time at its starting coordinate denoted δ. The three variants can be implemented by using different velocity models for an FLS. These velocity models are presented in Section 2.2.4. Based on the coordinate of the closest point to f on the flight pattern, this algorithm computes the time for f to arrive at this position and the time for its assigned slot to arrive at the same coordinate. Algorithm 2 is a sequence of analytical expressions with $O(1)$ complexity. It outputs the rendezvous time, and the amount of time δ that the FLS must wait.

Algorithm 2: SD(FP, f, P_{close})

```
$P_{slot} \leftarrow FP.slots[f.sID]$
$\overrightarrow{V_{(slot,C)}} = \frac{\overrightarrow{P_{slot} FP.P_C}}{|\overrightarrow{P_{slot} FP.P_C}|}$
$\overrightarrow{V_{(close,C)}} = \frac{\overrightarrow{P_{close} FP.P_C}}{|\overrightarrow{P_{close} FP.P_C}|}$
$angle_{rotation} = \arccos(\overrightarrow{V_{(slot,C)}} \cdot \overrightarrow{V_{(close,C)}})$
if $(\overrightarrow{V_{(slot,C)}} \times \overrightarrow{V_{(close,C)}}) \cdot FP.\overrightarrow{V_{Norm}} < 0$ then
    $angle_{rotation} = 2\pi - angle_{rotation}$
end
$t_{slot} = angle_{rotation} \div \frac{FP.S_{slot}}{FP.R}$
$t_f = \textsc{Min-TimeVelocityModel}(|\overrightarrow{f.P_f P_{close}}|)$
if $t_{slot} < t_f$ then
    $t_{round} \leftarrow \frac{2\pi \times FP.R}{FP.S_{slot}}$
    $t_{slot} \leftarrow t_{slot} + \left\lceil \frac{t_f - t_{slot}}{t_{round}} \right\rceil \times t_{round}$
end
if Variant 1 then
    $\delta = t_{slot} - t_f$
end
else if Variant 2 then
    $\delta = \textsc{Fix-TimeVelocityModel}(|\overrightarrow{f.P_f P_{close}}|, t_{slot})$
end
else if Variant 3 then
    $\delta_{hybrid} \leftarrow$ The specified waiting time for Variant 3
    $\delta = \delta_{hybrid}$
end
return $[t_{slot}, \delta]$
```

2.2.3 *Fastest Rendezvous Time (FRT) Path.*

Algorithm 3 computes the path with the Fastest Travel Time (FRT) for an FLS to rendezvous with its assigned slot. The algorithm computes both the rendezvous time and its coordinate on the flight pattern by using the closest P_{close} and farthest P_{far} points on the flight pattern as a guide. Since the distance traveled by the FLS will not be longer than the distance from P_f to P_{far}, and will not be shorter than the distance from P_f to P_{close}, then the upper and lower bound can be calculated with the velocity model of the FLS using the fastest speed S_{max}. This defines an interval of time $[T_{min}, T_{max}]$. We break this interval into μ slices, each with ϵ duration, $\mu = \frac{T_{max} - T_{min}}{\epsilon}$. FRT uses binary search to determine when the assigned slot will rendezvous with the FLS in a time slice. It uses this slice as input to Algorithm 1 to determine a point on the flight path. This is the coordinates of the rendezvous location. The complexity of the algorithm is O(log μ).

Algorithm 3: FRT($FP, f, P_{close}, P_{far}, \epsilon$)

```
1  $lo \leftarrow$ VelocityModel($|\overrightarrow{f.P_f P_{close}}|$)
2  $hi \leftarrow$ VelocityModel($|\overrightarrow{f.P_f P_{far}}|$)
3  while $lo \leq hi + \epsilon$ do
4      $mid \leftarrow (lo + hi) \div 2$
5      $P_{slot} \leftarrow$ GetSlotPosition($FP, f.sID, mid$)
6      $t_{travel} \leftarrow$ VelocityModel($|\overrightarrow{f.P_f P_{slot}}|$)
7      $t_{diff} \leftarrow t_{travel} - mid$
8      if $t_{diff} > 0$ then
9          $lo \leftarrow mid + \epsilon$
10     end
11     else
12         $hi \leftarrow mid$
13     end
14 end
15 return[GetSlotPosition($FP, f.SID, lo$), $lo$]
```

Algorithm 3 shows the pseudo-code for FRT. Its input includes a flight pattern object FP, an FLS f, the closest P_{close} and farthest P_{far} points on the flight pattern, and ϵ which is the duration of a time slice. Its output is the coordinate of the rendezvous point and its time. Steps 1 and 2 use the velocity model of Section 2.2.4 to compute the time to travel to P_{close} and P_{far}, respectively. Steps 3 to 14 implement the binary search technique to compute the FLS rendezvous time with its assigned time slot. The complexity of Algorithm 3 is O(ln μ).

2.2.4 *Velocity Model.*

Assumption: An FLS has a maximum acceleration $a^{\uparrow}$, a maximum deceleration $a^{\downarrow}$, and a maximum[1] speed S_{max}. At any time, the FLS may not travel faster than S_{max}, and the rate of which the FLS's speed changes cannot exceed $a^{\uparrow}$ when accelerating or $a^{\downarrow}$ when decelerating. Note that all S_{max}, $a^{\uparrow}$ and $a^{\downarrow}$ are scalar values larger than 0, independent of the heading of an FLS.

Min-Time Velocity Model: SD (Variant 1) and FRT use the Min-Time Velocity Model. The velocity model describes the acceleration, deceleration, and a maximum speed S_{max} that an FLS may use to travel distance d. This is the distance from the FLS's current coordinate P_f to the coordinates of its assigned slot P_s, $d = |P_f P_s|$. We assume the starting velocity of the FLS is zero and the maximum FLS speed is higher than the speed of a slot, $S_{max} \geq S_{slot}$. Factors such as gravity may cause $a^{\uparrow}$ to not equal $a^{\downarrow}$.

Problem 1. *Minimize the time to travel distance d from P_f to P_s without exceeding the maximum speed S_{max}, or exceeding either the maximum acceleration $a^{\uparrow}$ or maximum deceleration $a^{\downarrow}$.*

Three scenarios constitute the solution to this problem:

(1) d requires the FLS to accelerate at rate $a^{\uparrow}$ to reach S_{max}, travel at speed S_{max} until a well defined point, and decelerate at rate $a^{\downarrow}$ to match the speed of its assigned slot at P_s.
(2) d requires the FLS to accelerate at rate $a^{\uparrow}$ to reach S_{max}. However, the FLS must decelerate immediately at the rate $a^{\downarrow}$ to match the speed of its assigned slot at P_s.
(3) d requires the FLS to accelerate to arrive at a well defined point. Prior to reaching S_{max}, the FLS must decelerate to S_{slot}, the speed of its assigned slot at P_s.
(4) d is too small, i.e., the FLS is too close to the slot, preventing the FLS from accelerating to match the speed of its slot at P_s.

An FLS detects the different scenarios using distance d to its destination, distance Δ_a (Δ_p) required to accelerate to S_{max} (S_{slot}), and distance Δ_d required to decelerate to match S_{slot} from S_{max}. The FLS is in Scenario 1 when Δ_a+Δ_d is less than d, Scenario 2 when $\Delta_a + \Delta_d$ equals d, Scenario 3 when $\Delta_a + \Delta_d$ is greater than d, Scenario 4 when Δ_p is greater than d. Note that the ideal scenario (not listed) is when Δ_p equals to d as it enables an FLS to accelerate from rest to occupy its slot at speed S_{slot}.

Below, we describe how an FLS computes Δ_a and Δ_d. Subsequently, we detail each scenario in turn.

Detection of alternative scenarios: The velocity of an FLS at time $T + t$ is a function of its speed at time T and acceleration or deceleration the afterwards t: $V_{i+1} = V_i + a\delta$. Where δ is the duration of the step. V_i is a value between 0 and S_{max}, $V_i \in [0, S_{max}]$. The distance an FLS travels while accelerating or decelerating in time t is:

$$\Delta = V_i t + \frac{1}{2} a(t)^2 \tag{1}$$

Where V_i is the starting speed and a is set to either $a^{\uparrow}$ or -$a^{\downarrow}$ depending on whether the FLS is accelerating or decelerating, respectively.

Consider the scenario when an FLS accelerates from a starting velocity of zero to reach the maximum speed. The amount of time required to reach the maximum speed S_{max} is $\frac{S_{max}}{a^{\uparrow}}$. During this time, the FLS travels distance Δ_a to reach S_{max}, Δ_a=$\frac{1}{2}a^{\uparrow}t^2$. When decelerating from the maximum speed with the objective to match S_{slot} then V_i is S_{max} and the required time is $t = \frac{S_{max} - S_{slot}}{a^{\downarrow}}$. The traveled distance Δ_d=$S_{max}t - \frac{1}{2}a^{\downarrow}(\frac{S_{max} - S_{slot}}{a^{\downarrow}})^2$, see Equation 1.

Alternative scenarios: In Scenario 1, the FLS cruises at speed S_{max} for a distance equivalent to Δ_c=d-(Δ_a + Δ_d). Its duration is $\frac{\Delta_c}{S_{max}}$. In Scenario 2, once the FLS speed is S_{max}, it starts to decelerate at the rate $a^{\downarrow}$ to arrive at its slot with speed S_{slot}. With Scenario 3, the FLS uses the following equation to compute the amount of time to arrive at its slot: $\frac{\sqrt{2a^{\uparrow}\frac{a^{\downarrow}}{a^{\uparrow}+a^{\downarrow}}d}}{a^{\uparrow}} + \frac{\sqrt{2a^{\uparrow}\frac{a^{\downarrow}}{a^{\uparrow}+a^{\downarrow}}d}}{a^{\downarrow}}$. With the last scenario, the FLS must move $\Delta_p - d$ away from its slot. Now, it is in the ideal scenario to accelerate to match S_{slot}.

Example 1.

Consider a horizontal flight pattern, FP.θ=0, rotating at a speed of 0.7 m/second, FP.S_{slot}=0.7 m/second. The radius of this circular flight pattern is 1 meter, FP.R=1 m, and the coordinate of its center is [0,0,0.8]. An FLS with a starting coordinate $[1, 1, 0]$ must rendezvous with its assigned slot. The maximum FLS speed is

---

[1] The speed of a stationary FLS is zero. The minimum speed that an FLS may travel is S_{slow}. It is dictated by Variant 2 of SD. The velocity model tries to realize S_{slow}.

S_{max}=1.5 m/second. Its maximum acceleration and deceleration are 1 m/second^2. Hence, T_{min} = 0.901 seconds and T_{max}= 2.543 seconds. Assume the duration of a time slice is $\frac{1}{30}$ seconds, ϵ=0.033 seconds. The number of time slots is 36, μ=36. Hence, FRT of Algorithm 3 requires $\lceil \log_2 36 \rceil = 6$ iterations. It computes a straight line path for the FLS to rendezvous with its assigned slot 0.83 seconds from the current time and travel 0.963 m with the fastest speed.

With the same settings, SD of Algorithm 2 (adjusted for Variant 2) requires a flight time of 2.574 seconds. This is more than 3x longer than FRT. However, its traveled distance (0.9 m) is 7% shorter than FRT. One reason for SD's long flight time is its wait time $\delta = 7.526$ second to rendezvous with its slot.

Fix-Time Velocity Model: Variants 2 and 3 use the Fix-Time Velocity Model. This velocity model describes how an FLS may achieve rendezvous speed S_{slot} after traveling distance d during time T. The value of T is determined by the time to rendezvous with t_{slot} (see Algorithm 2) and the waiting time δ, $T = t_{slot} - \delta$. Similar to the Min-Time Velocity Model, d=$|P_f P_s|$, and we assume the FLS starts from the velocity zero and $S_{max} > S_{slot}$.

Problem 2. *Travel distance d from P_f to P_s in a fixed time T without exceeding the maximum speed S_{max}, or exceeding either the maximum acceleration $a^{\uparrow}$ or maximum deceleration $a^{\downarrow}$.*

Four scenarios constitute the solution to this problem:

1. d requires the FLS to continuously accelerate for a duration T to reach S_{slot}, so to match the speed of its assigned slot P_s.
2. d requires the FLS to accelerate to reach S_{slot}, then travel at speed S_{slot} until it rendezvous with its slot at P_s.
3. d requires the FLS to accelerate to reach S'_{max}, travel at speed S'_{max} until a well defined point, and decelerate to match the speed of its assigned slot at P_s. Note that $S_{slot} < S'_{max} < S_{max}$.
4. d is too small, i.e., the FLS is too close to the slot, preventing the FLS from accelerating to match the speed of its slot at P_s.

The acceleration and deceleration of an FLS may not be fixed constant. Multiple optimization approaches can be adapted to calculate a smooth change in acceleration (deceleration) [41, 60]. Here, we describe the simplest version using a constant acceleration and deceleration. An FLS detects different scenarios using distance d to its destination, speed of its assigned slot S_{slot} and travel time T. *Detection of alternative scenarios:* If $\frac{S^2_{slot}}{2a^{\uparrow}}$ is greater than d, the FLS is in Scenario 4. Otherwise, the FLS is in Scenario 1 when $\frac{TS_{slot}}{2}$ is greater than or equal to d, Scenario 2 when $\frac{TS_{slot}}{2}$ is smaller than d and $\frac{S^2_{slot}}{2a^{\uparrow}} + (T - \frac{S_{slot}}{a^{\uparrow}})S_{slot}$ is greater than or equal to d, Scenario 3 when $\frac{S^2_{slot}}{2a^{\uparrow}} + (T - \frac{S_{slot}}{a^{\uparrow}})S_{slot}$ is smaller than d. Note that there is one special case in Scenario 3, where $S'_{max} = S_{max}$, the acceleration is $a^{\uparrow}$, and the deceleration is $a^{\downarrow}$. In this case, Fix-Time Velocity Model generates the same moving pattern as the Min-Time Velocity Model. This may happen when δ_{min} and δ_{hybrid} in Variant 2 and 3 of SD and δ_{max} of Variant 1 of SD are all limited to 0 by d and T

Below, we fill in the detailed acceleration and deceleration of an FLS in different scenarios based on a constant acceleration and deceleration model.

With Scenario 1, an FLS may accelerate with the rate of $a^{\uparrow\prime}$, $a^{\uparrow\prime} = \frac{S^2_{slot}}{2d}$. It will reach speed of S_{slot} by the time it rendezvous with its assigned slot.

With Scenario 2, the time that an FLS may accelerate is $t^{\uparrow}$, $t^{\uparrow} = 2(T - \frac{d}{S_{slot}})$. Hence the acceleration $a^{\uparrow\prime}$ can be calculated accordingly, $a^{\uparrow\prime} = \frac{S_{slot}}{t^{\uparrow}}$. Once it reach the speed of S_{slot}, it will move with this constant speed and

For Scenario 3, there are multiple ways an FLS can do to achieve the goal. An FLS maximize its accelerating time $t^{\uparrow}$ and later have a longer decelerating time $t^{\downarrow}$, or the opposite, or choose a balance between these two. There are three constraints on $t^{\uparrow}$ and $t^{\downarrow}$:

$$T \geq t^{\uparrow} + t^{\downarrow} \tag{2}$$

$$d = \frac{S'_{max} t^{\uparrow}}{2} + \frac{(S'_{max} + S_{slot}) t^{\downarrow}}{2} + (T - t^{\uparrow} - t^{\downarrow}) S'_{max} \tag{3}$$

$$\begin{cases} S'_{max} \leq a^{\uparrow} t^{\uparrow} \\ S'_{max} \leq a^{\downarrow} t^{\downarrow} + S_{Slot} \end{cases} \tag{4}$$

Here, we provide the equation for calculation with a focus on minimizing the accelerating time $t^{\uparrow}$ and the decelerating time $t^{\downarrow}$ by using $a^{\uparrow}$ and $a^{\downarrow}$ for acceleration and deceleration. Equation 3 can be re-formalized as:

$$d = \frac{S'^2_{max}}{2a^{\uparrow}} + \frac{(S'^2_{max} + S_{slot})}{2a^{\downarrow}} + (T - \frac{S'_{max}}{a^{\uparrow}} - \frac{S'_{max}}{a^{\downarrow}}) S'_{max} \tag{5}$$

and S'_{max} can be calculated accordingly:

$$S'_{max} = \frac{T - \sqrt{T^2 - (\frac{1}{a^{\uparrow}} + \frac{1}{a^{\downarrow}})(\frac{S_{slot}}{2a^{\downarrow}} - d)}}{\frac{1}{a^{\uparrow}} + \frac{1}{a^{\downarrow}}} \tag{6}$$

3 IMPLEMENTATION

We developed a simulation model of the techniques and implemented them using a swarm of Crazyflie drones. We specified a horizontal circular flight pattern with a radius of 1 meter, R=1, θ=0. It rotates counter-clockwise. The speed of its slots is $0.7m/second$. The slots are separated by a distance of 125.66 cm ($\frac{2\pi \times 1m}{5}$) and are 1.79 seconds apart. The diameter of a Crazyflie is 15 cm (including span of propellers). We set its maximum speed S_{max} at 1.5 m/second, and maximum acceleration and deceleration at $1m/second^2$. The dimensions of the space is $3m \times 3m \times 1.5m$, and the opening is $27cm \times 27cm$. Figures 3a and 3b show the side and bottom view of the opening of our implementation. A visualization of the simulation model is shown in Figure 4.

Our implementation starts with 5 Crazyflies at a stationary state on the ground. They fly to 5 random initial coordinate. Each drone is assigned with available slots using a Round Robin policy. We use the equations of Section 2.2.2 to compute the shortest path for each FLS to occupy its assigned slot, starting with the speed of zero.

The sudden turns when FLSs are rendezvousing with their assigned slots are handled by crazyswarm platform [66].

Once an FLS rendezvous with its assigned slot, it occupies the slot and flies at the speed of the slot until it is below the opening. Subsequently, it enters the opening an relinquishes its slot. The FLS flies to a corner of the display space, descends to the floor, and flies to its newly assigned random starting position to repeat the process. An experiment has a duration of one minute. It terminates by landing the 5 FLSs on the ground. See implementation.

4 EVALUATION

We used the experimental setup of Figure 6 to quantify the tradeoff associated with SD and FRT, Algorithms 2 and 3, respectively. The important configuration parameters of the flight pattern include its radius R=1 meter, the speed of slots 0.7 m/second, and the number of slots 5. We simulated starting point for one FLS along a line that is a fixed distance below the center of the flight pattern. This distance is a function of the radius of the flight pattern, $\omega \times R$. We vary the value of ω from 1 to 1000 including in between values. The distance between the points on the line is fixed at 1 meter, i.e., Point 10 is 9 meters away from Point 1 which is aligned below the center of the flight pattern. See Figure 6.

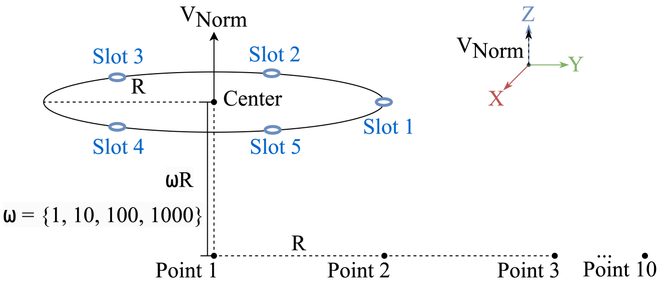


Figure 6: Experimental setup.

Summary of lessons: We conducted many experiments. A summary of lessons learned include: First, SD results in a shorter travel distance compared to FRT while FRT results in a faster rendezvous time when compared with SD. Second, the difference between SD and FRT becomes insignificant as we increase the distance between an FLS and its assigned slot, i.e., $\omega \geq 100$. Third, minimizing the time for a slot to make a rotation on the flight pattern expedites the FLS rendezvous time with an FLS. Similarly, increasing the FLS speed, its acceleration and deceleration expedites its rendezvous time with an FLS. Fourth, with SD, the shortest distance may be such that an FLS arrives at the rendezvous point and misses its assigned slot. In this case, the FLS must wait for one rotation of the slot, delaying the rendezvous time. See Figure 7b.

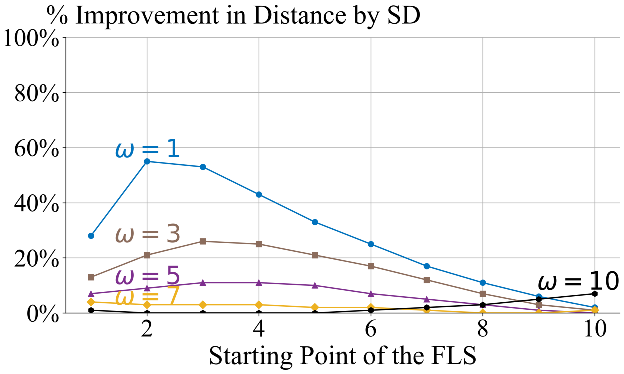
Table 1 shows the numerical results of distance traveled and time spent by FLS starting at point 1, assigned with Slot 2, different values of ω.

Detailed Results: The experimental results presented in this section use a flight pattern with the specification of Example 1 and our Crazyflie implementation of Section 3.

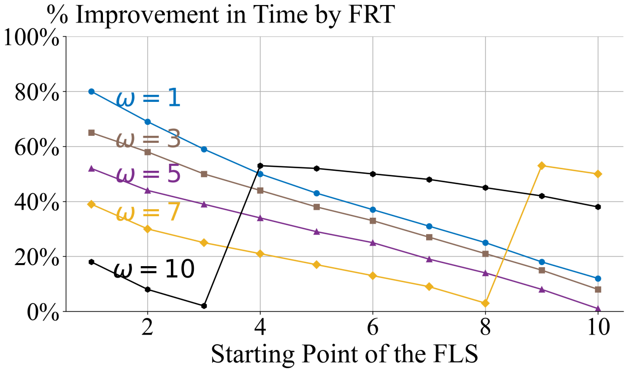
Figure 7a shows the percentage improvement in distance provided by SD when compared with FRT. The x-axis of this figure

Table 1: Traveled distance and rendezvous time, SD and FRT.

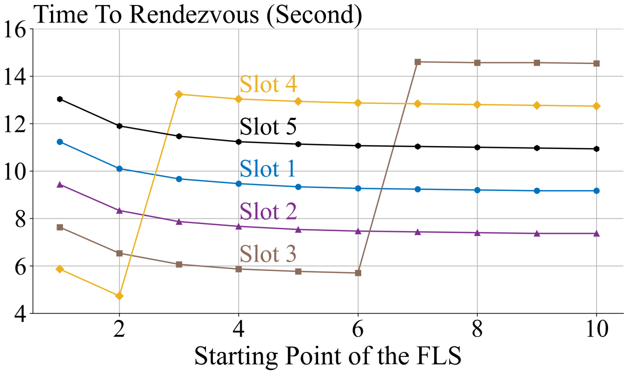
ω	**Dist Traveled (m)**		**Time (second)**	
	SD	**FRT**	**SD**	**FRT**
1R	1.00	1.40	9.43	1.87
5R	5.00	5.38	9.43	4.57
10R	10.00	10.06	9.43	7.70
100R	100.00	100.02	72.27	67.67
1000R	1,000.00	1,000.00	673.67	673.67



(a) SD's % improvement in distance when compared with FRT.



(b) FRT's % improvement in time when compared with SD.



(c) Rendezvous time with different slot assignments, SD, ω=5.

Figure 7: An evaluation of SD and FRT.

denotes the points on the line below the flight pattern. An FLS is assigned Slot 2. (We discuss other slots at the end of this section.) SD's highest percentage improvement is observed with low values of ω (e.g., ω=1), i.e., when the line is closest to the flight pattern. This improvement decreases as we increase ω. Beyond $\omega \geq 10$, the percentage improvement provided by SD is insignificant because the traveled distance is large. This distance dominates to make the difference between the closest P_{close} and farthest P_{far} point (fixed at 2R) on the flight pattern insignificant.

Figure 7b shows the percentage improvement in time by FRT when compared with SD as a function of the points on the line. The different lines correspond to the different values of ω.The percentage improvement provided by FRT decreases as a function of ω with a few exceptions. The percentage decrease is due to an increase in the distance traveled by the FLS. SD requires the FLS to fly at the fastest speed similar to FRT. With sufficiently large distances, the rendezvous time to the closest point versus to a point between the closest and farthest points on the flight pattern becomes insignificant. This explains the decrease in percentage improvement as a function of ω. This explanation also applies to the general decrease of a line, say $\omega = 1$, as a function of the points on the line. Point 10 is 10x farther away than Point 1, rendering the aforementioned different insignificant.

Figure 7b shows sudden jumps in the percentage improvement with ω values 7 and 10. This is due to an FLS arriving at its rendezvous point P_{close} only to miss its assigned slot. The FLS waits for the slot to make a full rotation. This results in a significant increase in rendezvous time with SD. Note that SD requires the FLS to travel at its fastest speed. Reducing its speed will only increase the percentage improvement provided by FRT.

The slot assigned to an FLS dictates whether it must wait for a rotation of the slot. Figure 7c highlights this by focusing on ω=5. It shows the time to rendezvous for different slots on the flight pattern as a function of the FLS location on the line, i.e., points shown in Figure 6. While with Slots 1, 2, and 5, the rendezvous time is a line that decreases smoothly and tends to flatten out, it increases with Slots 3 and 4 due to an FLS waiting for either a full or a partial rotation of its slot.

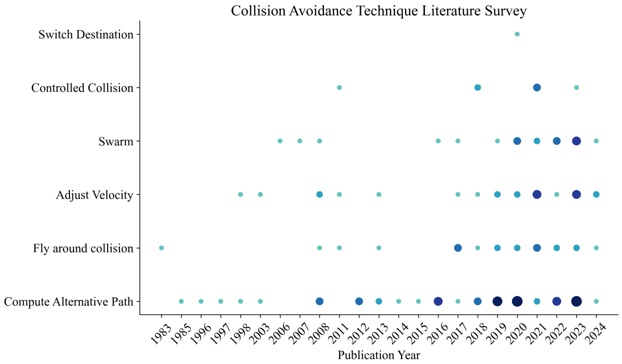


Figure 8: 97 collision avoidance/handling studies.

Table 2: A summary of FRT and variants of SD.

	Distance Traveled	**Flight Time**	**Battery Flight Time**
SD:V1	Optimizes	Optimizes	✗
SD:V2	Optimizes	✗	✗
SD:V3	Optimizes	✗	Optimizes
FRT	✗	Optimizes	✗

5 RELATED WORK

The concept of flight patterns for use by a Dronevision is first introduced in [107]. It presents alternative shapes for a flight pattern, e.g., square, circle, and ellipsoid. It also describes flight patterns that are either single layer or hierarchical. We focus on the circular pattern of [107] and extend it by presenting different angles θ for a flight pattern relative to the Z-axis, and algorithms SD and FRT that enable an FLS to rendezvous with its assigned slot on the flight pattern. We present analytical models, simulation studies, and an implementation using a swarm of Crazyflie drones. These novel extensions are absent from [107].

Our novel extensions are absent from prior studies in collision handling techniques. This is based on our survey of 97 studies that were published from 1983-2024 . Figure 8 shows the six forms of collisions as a function of the publication year. The size of a circle and its darkness denotes the number of studies, ranging from 1 to 6. 48 studies compute an alternative path [10, 11, 14, 18, 19, 22, 23, 28–32, 34–36, 38, 40, 42, 45, 47–49, 52, 57, 58, 62, 68, 72, 74, 78, 80, 82, 84, 85, 87, 90, 92, 95–97, 99, 100, 102–105, 108], 21 studies fly around the collision point [3, 12, 15, 17, 21, 23, 24, 27, 43, 44, 53, 66, 67, 71, 76, 77, 83, 86, 91, 93], 26 studies adjust velocity [17, 20, 21, 23, 27, 28, 30, 38, 43–45, 51, 52, 54, 64, 67, 72, 76, 79, 80, 83, 86, 91, 93, 97, 99], 27 studies employ a swarming technique [1, 2, 5–8, 12, 16, 20, 25, 30, 31, 33, 39, 55, 59, 63, 65, 66, 73, 88, 89, 98, 101, 105–107], 6 studies use controlled collision [46, 51, 54, 56, 61, 64], and 1 study switches destination of the colliding agents [76] [2].

We also analyzed more recent papers related to drone swarm managements [9, 69, 70, 75]. None have the concept of a flight pattern, SD and FRT algorithms.

6 CONCLUSION

This paper presents the design and implementation of a circular flight pattern that enables a swarm of FLSs to enter an opening of a 3D multimedia display, Dronevision. This opening provides FLSs with access to charging coils. We presented two novel algorithms, SD and FRT, that enable an FLS to occupy its assigned slot with the flight pattern at an arbitrary angle θ relative to the z-axis. The 3 variants of SD optimize for different metrics as shown in Table 2. One may view Variant 1 of SD as a hybrid of FRT that prioritizes optimizing traveled distance followed with travel time. Experimental results show SD minimizes the distance traveled by an FLS while FRT minimize the time traveled. Our implementation using a swarm of Crazyflie drones validates the correctness of our designs. Both our design and implementation scale to a large number of drones and slots. While we focused on a 3D multimedia display, the

---

[2] Total is 129 since a study overlaps multiple categories.

concept of a flight pattern and our algorithms are flexible for use by other applications that require a swarm of drones to fly through an opening.

ACKNOWLEDGMENTS

This research was supported in part by the NSF grants IIS-2232382 and CMMI-2425754.

REFERENCES

[1] Hanif Zaini Abdul. 2020. *UAV Swarming with Collision Avoidance and Communication Constraints*. Ph.D. Dissertation. Nanyang Technological University. https://doi.org/10.32657/10356/137101

[2] Afzal Ahmad, Viktor Walter, Pavel Petracek, Matej Petrlik, Tomas Baca, David Zaitlik, and Martin Saska. 2021. Autonomous Aerial Swarming in GNSS-denied Environments with High Obstacle Density. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Xi'an, China, 570–576. https://doi.org/10.1109/ICRA48506.2021.9561284

[3] Shibbir Ahmed, Baijing Qiu, Chun-Wei Kong, Huang Xin, Fiaz Ahmad, and Jinlong Lin. 2022. A Data-Driven Dynamic Obstacle Avoidance Method for Liquid-Carrying Plant Protection UAVs. *Agronomy* 12, 4 (April 2022), 873. https://doi.org/10.3390/agronomy12040873

[4] Hamed Alimohammadzadeh, Rohit Bernard, Yang Chen, Trung Phan, Prashant Singh, Shuqin Zhu, Heather Culbertson, and Shahram Ghandeharizadeh. 2023. Dronevision: An Experimental 3D Testbed for Flying Light Specks. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 1–9. https://doi.org/10.61981/ZFSH2301

[5] Hamed Alimohammadzadeh and Shahram Ghandeharizadeh. 2023. SwarMer: A Decentralized Localization Framework for Flying Light Specks. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 10–22. https://doi.org/10.61981/ZFSH2302

[6] Hamed Alimohammadzadeh, Shuqin Zhu, Jiadong Bai, and Shahram Ghandeharizadeh. 2024. Reliability Groups with Standby Flying Light Specks. In *ACM Multimedia Systems* (Bari, Italy).

[7] Guillermo Angeris, Kunal Shah, and Mac Schwager. 2019. Fast reciprocal collision avoidance under measurement uncertainty. In *The International Symposium of Robotics Research*. Springer, 191–207.

[8] Senthil Hariharan Arul and D. Manocha. 2020. DCAD: Decentralized Collision Avoidance With Dynamics Constraints for Agile Quadrotor Swarms. *IEEE Robotics and Automation Letters* 5 (2020), 1191–1198. https://doi.org/10.1109/LRA.2020.2967281

[9] Godwin Asaamoning, Paulo Mendes, Denis Rosário, and Eduardo Cerqueira. 2021. Drone swarms as networked control systems by integration of networking and computing. *Sensors* 21, 8 (2021), 2642.

[10] Federico Augugliaro, Angela P. Schoellig, and Raffaello D'Andrea. 2012. Generation of Collision-free Trajectories for a Quadrocopter Fleet: A Sequential Convex Programming Approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Vilamoura-Algarve, Portugal, 1917–1922. https://doi.org/10.1109/IROS.2012.6385823

[11] Tomas Baca, Daniel Hert, Giuseppe Loianno, Martin Saska, and Vijay Kumar. 2018. Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Madrid, 6753–6760. https://doi.org/10.1109/IROS.2018.8594266

[12] Saptarshi Bandyopadhyay, Francesca Baldini, Rebecca Foust, Amir Rahmani, Jean-Pierre De La Croix, Soon-Jo Chung, and Fred Hadaegh. 2017. Distributed Spatiotemporal Motion Planning for Spacecraft Swarms in Cluttered Environments. In *AIAA SPACE and Astronautics Forum and Exposition*. American Institute of Aeronautics and Astronautics, Orlando, FL. https://doi.org/10.2514/6.2017-5323

[13] Jan Carlo Barca, A Sekercioglu, and Adam Ford. 2013. Controlling Formations of Robots with Graph Theory. In *Intelligent Autonomous Systems 12: Volume 2 Proceedings of the 12th International Conference IAS-12, held June 26-29, 2012, Jeju Island, Korea*. Springer, 563–574.

[14] Daman Bareiss and Joran van den Berg. 2013. Reciprocal Collision Avoidance for Robots with Linear Dynamics using LQR-Obstacles. In *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, USA, 3847–3853. https://doi.org/10.1109/ICRA.2013.6631118

[15] Rodney A. Brooks. 1983. Solving the Find-path Problem by Good Representation of Free Space. *IEEE Trans. Syst., Man, Cybern.* SMC-13, 2 (March 1983), 190–197. https://doi.org/10.1109/TSMC.1983.6313112

[16] Anthony J Calise and Daniel Preston. 2008. Swarming/Flocking and Collision Avoidance for Mass Airdrop of Autonomous Guided Parafoils. *Journal of guidance, control, and dynamics* 31, 4 (2008), 1123–1132.

[17] D. Cappello, S. Garcin, Z. Mao, M. Sassano, A. Paranjape, and T. Mylvaganam. 2021. A Hybrid Controller for Multi-Agent Collision Avoidance via a Differential Game Formulation. *IEEE Trans. Contr. Syst. Technol.* 29, 4 (July 2021), 1750–1757. https://doi.org/10.1109/TCST.2020.3005602

[18] Clement Chahbazian, Karim Dahia, Nicolas Merlinge, Benedicte Winter-Bonnet, Kevin Honore, and Christian Musso. 2022. Improved Kalman-Particle Kernel Filter on Lie Groups Applied to Angles-Only UAV Navigation. (2022).

[19] Han Chen and Peng Lu. 2020. Computationally Efficient Obstacle Avoidance Trajectory Planner for UAVs Based on Heuristic Angular Search Method. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Las Vegas, NV, USA, 5693–5699. https://doi.org/10.1109/IROS45743.2020.9340778

[20] Ji Chen, Hanlin Wang, Michael Rubenstein, and Hadas Kress-Gazit. 2020. Automatic Control Synthesis for Swarm Robots from Formation and Location-based High-level Specifications. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Las Vegas, NV, USA, 8027–8034. https://doi.org/10.1109/IROS45743.2020.9341466

[21] Yuda Chen, Haoze Dong, and Zhongkui Li. 2023. Asynchronous Spatial Allocation Protocol for Trajectory Planning of Heterogeneous Multi-Agent Systems. http://arxiv.org/abs/2309.07431 arXiv:2309.07431 [cs].

[22] Ziyang Chen, Fei Luo, and Chunjie Zhai. 2019. Obstacle Avoidance Strategy for Quadrotor UAV based on Improved Particle Swarm optimization Algorithm. In *2019 Chinese Control Conference (CCC)*. IEEE, Guangzhou, China, 8115–8120. https://doi.org/10.23919/ChiCC.2019.8865866

[23] Kai Cui, Mengguang Li, Christian Fabian, and Heinz Koeppl. 2023. Scalable Task-Driven Robotic Swarm Control via Collision Avoidance and Learning Mean-Field Control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, London, United Kingdom, 1192–1199. https://doi.org/10.1109/ICRA48891.2023.10161498

[24] Ema Falomir, Serge Chaumette, and Gilles Guerrini. 2018. A Mobility Model Based on Improved Artificial Potential Fields for Swarms of UAVs. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Madrid, 8499–8504. https://doi.org/10.1109/IROS.2018.8593738

[25] Malintha Fernando. 2021. Online Flocking Control of UAVs with Mean-Field Approximation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Xi'an, China, 8977–8983. https://doi.org/10.1109/ICRA48506.2021.9560899

[26] Eduardo Ferrera, Alfonso Alcántara, J. Capitán, Á. R. Castaño, P. Marrón, and A. Ollero. 2018. Decentralized 3D Collision Avoidance for Multiple UAVs in Outdoor Environments. *Sensors (Basel, Switzerland)* 18 (2018).

[27] Mael Feurgard, Gautier Hattenberger, and Simon Lacroix. 2024. Extending Guiding Vector Field to track unbounded UAV paths. (2024).

[28] Paolo Fiorini and Zvi Shiller. 1998. Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research* 17, 7 (July 1998), 760–772. https://doi.org/10.1177/027836499801700706

[29] D. Fox, W. Burgard, and S. Thrun. 1997. The Dynamic Window Approach to Collision Avoidance. *IEEE Robot. Automat. Mag.* 4, 1 (March 1997), 23–33. https://doi.org/10.1109/100.580977

[30] Shahram Ghandeharizadeh. 2021. Holodeck: Immersive 3D Displays Using Swarms of Flying Light Specks. In *ACM Multimedia Asia* (Gold Coast, Australia). ACM Press, New York, NY, 1–7. https://doi.org/10.1145/3469877.3493698

[31] Shahram Ghandeharizadeh. 2022. Display of 3D Illuminations using Flying Light Specks. In *ACM Multimedia*. ACM Press, New York, NY, 2996–3005. https://doi.org/10.1145/3503161.3548250

[32] Manaram Gnanasekera and Jay Katupitiya. 2020. A Time Optimal Reactive Collision Avoidance Method for UAVs Based on a Modified Collision Cone Approach. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Las Vegas, NV, USA, 5685–5692. https://doi.org/10.1109/IROS45743.2020.9341259

[33] Michael Hamer, Lino Widmer, and Raffaello D'andrea. 2019. Fast Generation of Collision-Free Trajectories for Robot Swarms Using GPU Acceleration. *IEEE Access* 7 (2019), 6679–6690. https://doi.org/10.1109/ACCESS.2018.2889533

[34] Lei He, Nabil Aouf, James F. Whidborne, and Bifeng Song. 2020. Integrated Moment-based LGMD and Deep Reinforcement Learning for UAV Obstacle Avoidance. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Paris, France, 7491–7497. https://doi.org/10.1109/ICRA40945.2020.9197152

[35] Jacob Higgins, Nicholas Mohammad, and Nicola Bezzo. 2023. A Model Predictive Path Integral Method for Fast, Proactive, and Uncertainty-Aware UAV Planning in Cluttered Environments. http://arxiv.org/abs/2308.00914 arXiv:2308.00914 [cs].

[36] Wolfgang Hoenig, T. K. Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. 2016. Multi-Agent Path Finding with Kinematic Constraints. *ICAPS* 26 (March 2016), 477–485. https://doi.org/10.1609/icaps.v26i1.13796

[37] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. 2018. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics* 34, 4 (2018), 856–869.

[38] Huaxing Huang, Guijie Zhu, Zhun Fan, Hao Zhai, Yuwei Cai, Ze Shi, Zhaohui Dong, and Zhifeng Hao. 2022. Vision-based Distributed Multi-UAV Collision Avoidance via Deep Reinforcement Learning for Navigation. In *2022 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS). IEEE, Kyoto, Japan, 13745–13752. https://doi.org/10.1109/IROS47612.2022.9981803

[39] Jialei Huang, Fakui Wang, and Tianjiang Hu. 2023. CoFlyers: A Universal Platform for Collective Flying of Swarm Drones. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Detroit, MI, USA, 8808–8813. https://doi.org/10.1109/IROS55552.2023.10342485

[40] Ahmed Hussein, Abdulla Al-Kaff, Arturo De La Escalera, and Jose Maria Armingol. 2015. Autonomous indoor navigation of low-cost quadcopters. In *2015 IEEE International Conference on Service Operations And Logistics, And Informatics (SOLI)*. IEEE, Yassmine Hammamet, Tunisia, 133–138. https://doi.org/10.1109/SOLI.2015.7367607

[41] Wolfgang Hönig, James A. Preiss, T. K. Satish Kumar, Gaurav S. Sukhatme, and Nora Ayanian. 2018. Trajectory Planning for Quadrotor Swarms. *IEEE Transactions on Robotics* 34, 4 (2018), 856–869. https://doi.org/10.1109/TRO.2018.2853613

[42] Werner Alexander Isop and Friedrich Fraundorfer. 2019. Force Field-Based Indirect Manipulation Of UAV Flight Trajectories. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Macau, China, 2775–2782. https://doi.org/10.1109/IROS40897.2019.8967813

[43] Iswanto Iswanto, Alfian Ma'arif, Oyas Wahyunggoro, and Adha Imam. 2019. Artificial Potential Field Algorithm Implementation for Quadrotor Path Planning. *IJACSA* 10, 8 (2019). https://doi.org/10.14569/IJACSA.2019.0100876

[44] Ravinder Kumar Jyoti, Mohit Kumar Malhotra, and Debasish Ghose. 2021. Rogue Agent Identification and Collision Avoidance in Formation Flights using Potential Fields. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, Athens, Greece, 1080–1088. https://doi.org/10.1109/ICUAS51884.2021.9476866

[45] Yasuhiko Kamiyama. 2023. Perspective Chapter: On the Morse Property for the Distance Function of a Robot Arm. In *Motion Planning for Dynamic Agents*. IntechOpen.

[46] Sertac Karaman and Emilio Frazzoli. 2011. Sampling-based Algorithms for Optimal Motion Planning. *The International Journal of Robotics Research* 30, 7 (June 2011), 846–894. https://doi.org/10.1177/0278364911406761

[47] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. 1996. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE Trans. Robot. Automat.* 12, 4 (Aug. 1996), 566–580. https://doi.org/10.1109/70.508439

[48] Moslem Kazemi, Kamal K. Gupta, and Mehran Mehrandezh. 2013. Randomized Kinodynamic Planning for Robust Visual Servoing. *IEEE Trans. Robot.* 29, 5 (Oct. 2013), 1197–1211. https://doi.org/10.1109/TRO.2013.2264865

[49] O. Khatib. 1985. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, Vol. 2. Institute of Electrical and Electronics Engineers, St. Louis, MO, USA, 500–505. https://doi.org/10.1109/ROBOT.1985.1087247

[50] Anoop Kiran, Nora Ayanian, and Kenneth Breuer. 2023. Influence of quadrotor downwash on close proximity flight. *Bulletin of the American Physical Society* (2023).

[51] Takamasa Kominami, Zou Liang, Ricardo Rosales Martinez, Hannibal Paul, and Kazuhiro Shimonomura. 2023. Physical Contact with Wall using a Multirotor UAV Equipped with Add-On Thruster for Inspection Work. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Detroit, MI, USA, 6955–6961. https://doi.org/10.1109/IROS55552.2023.10341576

[52] Laura Krick, Mireille Broucke, and Bruce Francis. 2008. Getting Mobile Autonomous Robots to Form a Prescribed Geometric Arrangement. In *Recent Advances in Learning and Control*, Vincent D. Blondel, Stephen P. Boyd, and Hidenori Kimura (Eds.). Vol. 371. Springer London, London, 149–159. https://doi.org/10.1007/978-1-84800-155-8_11 ISSN: 0170-8643 Series Title: Lecture Notes in Control and Information Sciences.

[53] Bjorn Lindqvist, Pantelis Sopasakis, and George Nikolakopoulos. 2021. A Scalable Distributed Collision Avoidance Scheme for Multi-agent UAV systems. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Prague, Czech Republic, 9212–9218. https://doi.org/10.1109/IROS51168.2021.9636293

[54] Cheng Liu and Roberto Tron. 2021. Sensing via Collisions: a Smart Cage for Quadrotors with Applications to Self-Localization. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Xi'an, China, 4033–4039. https://doi.org/10.1109/ICRA48506.2021.9561896

[55] C. Lin Liu, Israel L. Donato Ridgley, Matthew L. Elwin, Michael Rubenstein, Randy A. Freeman, and Kevin M. Lynch. 2024. Self-Healing Distributed Swarm Formation Control Using Image Moments. *IEEE Robot. Autom. Lett.* 9, 7 (July 2024), 6216–6223. https://doi.org/10.1109/LRA.2024.3401171

[56] Zhichao Liu and Konstantinos Karydis. 2021. Toward Impact-resilient Quadrotor Design, Collision Characterization and Recovery Control to Sustain Flight after Collisions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Xi'an, China, 183–189. https://doi.org/10.1109/ICRA48506.2021.9561089

[57] Yixing Luo, Yijun Yu, Zhi Jin, Yao Li, Zuohua Ding, Yuan Zhou, and Yang Liu. 2020. Privacy-Aware UAV Flights through Self-Configuring Motion Planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Paris, France, 1169–1175. https://doi.org/10.1109/ICRA40945.2020.9197564

[58] Luís Martins, Carlos Cardeira, and Paulo Oliveira. 2019. Linear Quadratic Regulator for Trajectory Tracking of a Quadrotor. *IFAC-PapersOnLine* 52, 12 (2019), 176–181. https://doi.org/10.1016/j.ifacol.2019.11.195

[59] Silvia Mastellone, Dušan M. Stipanović, Christopher R. Graunke, Koji A. Intlekofer, and Mark W. Spong. 2008. Formation Control and Collision Avoidance for Multi-agent Non-holonomic Systems: Theory and Experiments. *The International Journal of Robotics Research* 27, 1 (Jan. 2008), 107–126. https://doi.org/10.1177/0278364907084441

[60] Daniel Mellinger and Vijay Kumar. 2011. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*. 2520–2525. https://doi.org/10.1109/ICRA.2011.5980409

[61] Yash Mulgaonkar, Anurag Makineni, Luis Guerrero-Bonilla, and Vijay Kumar. 2018. Robust Aerial Robot Swarms Without Collision Avoidance. *IEEE Robot. Autom. Lett.* 3, 1 (Jan. 2018), 596–603. https://doi.org/10.1109/LRA.2017.2775699

[62] Marcin Odelga, Paolo Stegagno, Nicholas Kochanek, and Heinrich H. Bulthoff. 2018. A Self-contained Teleoperated Quadrotor: On-Board State-Estimation and Indoor Obstacle Avoidance. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Brisbane, QLD, 7840–7847. https://doi.org/10.1109/ICRA.2018.8463185

[63] R. Olfati-Saber. 2006. Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory. *IEEE Trans. Automat. Contr.* 51, 3 (March 2006), 401–420. https://doi.org/10.1109/TAC.2005.864190

[64] Karishma Patnaik, Shatadal Mishra, Zachary Chase, and Wenlong Zhang. 2021. Collision Recovery Control of a Foldable Quadrotor. In *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 418–423. https://doi.org/10.1109/AIM46487.2021.9517341 arXiv:2105.12273 [cs, eess].

[65] Peng Peng, Wei Dong, Gang Chen, and Xiangyang Zhu. 2022. Obstacle Avoidance of Resilient UAV Swarm Formation with Active Sensing System in the Dense Environment. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Kyoto, Japan, 10529–10535. https://doi.org/10.1109/IROS47612.2022.9981858

[66] James Preiss, Wolfgang Honig, Gaurav Sukhatme, and Nora Ayanian. 2017. Crazyswarm: A Large Nano-Quadcopter Swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3299–3304. https://doi.org/10.1109/ICRA.2017.7989376

[67] Martin Rufli, Javier Alonso-Mora, and Roland Siegwart. 2013. Reciprocal Collision Avoidance With Motion Continuity Constraints. *IEEE Trans. Robot.* 29, 4 (Aug. 2013), 899–912. https://doi.org/10.1109/TRO.2013.2258733

[68] Markus Ryll, John Ware, John Carter, and Nick Roy. 2019. Efficient Trajectory Planning for High Speed Flight in Unknown Environments. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, Montreal, QC, Canada, 732–738. https://doi.org/10.1109/ICRA.2019.8793930

[69] Rashid A Saeed, Mohamed Omri, Sayed Abdel-Khalek, Elmustafa Sayed Ali, and Maged Faihan Alotaibi. 2022. Optimal path planning for drones based on swarm intelligence algorithm. *Neural Computing and Applications* 34, 12 (2022), 10133–10155.

[70] Fabrice Saffre, Hanno Hildmann, and Hannu Karvonen. 2021. The design challenges of drone swarm control. In *International conference on human-computer interaction*. Springer, 408–426.

[71] Daniel Schleich and Sven Behnke. 2022. Predictive Angular Potential Field-based Obstacle Avoidance for Dynamic UAV Flights. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Kyoto, Japan, 13618–13625. https://doi.org/10.1109/IROS47612.2022.9981677

[72] Brent Schlotfeldt, Dinesh Thakur, Nikolay Atanasov, Vijay Kumar, and George J. Pappas. 2018. Anytime Planning for Decentralized Multirobot Active Information Gathering. *IEEE Robot. Autom. Lett.* 3, 2 (April 2018), 1025–1032. https://doi.org/10.1109/LRA.2018.2794608

[73] Rajnikant Sharma and D Ghose. 2007. Swarm Intelligence based Collision Avoidance Between Realistically Modelled UAV Clusters. In *2007 American Control Conference*. IEEE, New York, NY, 3892–3897. https://doi.org/10.1109/ACC.2007.4282177

[74] Guni Sharon, Roni Stern, Ariel Felner, and Nathan Sturtevant. [n.d.]. Conflict-Based Search For Optimal Multi-Agent Path Finding. ([n. d.]).

[75] Enrica Soria, Fabrizio Schiano, and Dario Floreano. 2020. SwarmLab: A MATLAB drone swarm simulator. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 8005–8011.

[76] Hang Sun, Juntong Qi, Chong Wu, and Mingming Wang. 2020. Path Planning for Dense Drone Formation Based on Modified Artificial Potential Fields. In *2020 39th Chinese Control Conference (CCC)*. IEEE, Shenyang, China, 4658–4664. https://doi.org/10.23919/CCC50068.2020.9189345

[77] Jiayi Sun, Jun Tang, and Songyang Lao. 2017. Collision Avoidance for Cooperative UAVs With Optimized Artificial Potential Field Algorithm. *IEEE Access* 5 (2017), 18382–18390. https://doi.org/10.1109/ACCESS.2017.2746752

[78] Wen Sun, Jur Van Den Berg, and Ron Alterovitz. 2016. Stochastic Extended LQR for Optimization-Based Motion Planning Under Uncertainty. *IEEE Trans. Automat. Sci. Eng.* 13, 2 (April 2016), 437–447. https://doi.org/10.1109/TASE.

2016.2517124

[79] Camilla Tabasso, Venanzio Cichella, Syed Bilal Mehdi, Thiago Marinho, and Naira Hovakimyan. 2021. Time Coordination and Collision Avoidance Using Leader-Follower Strategies in Multi-Vehicle Missions. *Robotics* 10, 1 (Feb. 2021), 34. https://doi.org/10.3390/robotics10010034

[80] Ardalan Tajbakhsh, Lorenz T. Biegler, and Aaron M. Johnson. 2024. Conflict-Based Model Predictive Control for Scalable Multi-Robot Motion Planning. http://arxiv.org/abs/2303.01619 arXiv:2303.01619 [cs].

[81] Wenbo Tan and Na Huang. 2022. Event-based Rigid Formation System with Cooperative Finite-time Control. *IMA Journal of Mathematical Control and Information* 39, 1 (01 2022), 235–253. https://doi.org/10.1093/imamci/dnab041 arXiv:https://academic.oup.com/imamci/article-pdf/39/1/235/42682698/dnab041.pdf

[82] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. 2016. Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. *IEEE Trans. Robot.* 32, 6 (Dec. 2016), 1498–1511. https://doi.org/10.1109/TRO.2016.2603528

[83] Jesus Tordesillas, Brett T. Lopez, John Carter, John Ware, and Jonathan P. How. 2019. Real-Time Planning with Multi-Fidelity Models for Agile Flights in Unknown Environments. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, Montreal, QC, Canada, 725–731. https://doi.org/10.1109/ICRA.2019.8794248

[84] Matthew Turpin, Nathan Michael, and Vijay Kumar. 2014. Concurrent Assignment and Planning of Trajectories for Multiple Robots. *The International Journal of Robotics Research* 33, 1 (Jan. 2014), 98–112. https://doi.org/10.1177/0278364913515307

[85] Jur Van Den Berg, Ming Lin, and Dinesh Manocha. 2008. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, Pasadena, CA, USA, 1928–1935. https://doi.org/10.1109/ROBOT.2008.4543489

[86] Jur Van Den Berg, Jamie Snape, Stephen J. Guy, and Dinesh Manocha. 2011. Reciprocal Collision Avoidance with Acceleration-Velocity Obstacles. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, Shanghai, China, 3475–3482. https://doi.org/10.1109/ICRA.2011.5980408

[87] Jur van den Berg, David Wilkie, Stephen Guy, Marc Niethammer, and Dinesh Manocha. 2012. LQG-obstacles: Feedback Control with Collision Avoidance for Mobile Robots with Motion and Sensing Uncertainty. In *International Conference on Robotics and Automation*. IEEE, 346–353. https://doi.org/10.1109/ICRA.2012.6224648

[88] Matthieu Verdoucq, Clément Sire, Ramón Escobedo, Guy Theraulaz, and Gautier Hattenberger. 2023. Bio-Inspired 3D Flocking Algorithm with Minimal Information Transfer for Drones Swarms. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Detroit, MI, USA, 8833–8838. https://doi.org/10.1109/IROS55552.2023.10341413

[89] Matthieu Verdoucq, Guy Theraulaz, Ramon Escobedo, Clement Sire, and Gautier Hattenberger. 2022. Bio-inspired Control for Collective Motion in Swarms of Drones. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, Dubrovnik, Croatia, 1626–1631. https://doi.org/10.1109/ICUAS54217.2022.9836112

[90] Glenn Wagner and Howie Choset. 2013. M*: A Complete Multirobot Path Planning Algorithm with Optimality Bounds. *Redundancy in Robot Manipulators and Multi-Robot Systems* (2013), 167–181.

[91] Li Wang, Aaron D. Ames, and Magnus Egerstedt. 2017. Safety Barrier Certificates for Collisions-Free Multirobot Systems. *IEEE Trans. Robot.* 33, 3 (June 2017), 661–674. https://doi.org/10.1109/TRO.2017.2659727

[92] Jonas Westheider, Julius Rückin, and Marija Popović. 2023. Multi-UAV Adaptive Path Planning Using Deep Reinforcement Learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Detroit, MI, USA, 649–656. https://doi.org/10.1109/IROS55552.2023.10342516

[93] Michael T. Wolf and Joel W. Burdick. 2008. Artificial Potential Functions for Highway Driving with Collision Avoidance. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, Pasadena, CA, USA, 3731–3736. https://doi.org/10.1109/ROBOT.2008.4543783

[94] Yang Xu, Shupeng Lai, Jiaxin Li, Delin Luo, and Yancheng You. 2019. Concurrent Optimal Trajectory Planning for Indoor Quadrotor Formation Switching. *Journal of Intelligent & Robotic Systems* 94 (05 2019). https://doi.org/10.1007/s10846-018-0813-9

[95] Zhefan Xu, Di Deng, Yiping Dong, and Kenji Shimada. 2022. DPMPC-Planner: A real-time UAV trajectory planning framework for complex static environments with dynamic obstacles. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, Philadelphia, PA, USA, 250–256. https://doi.org/10.1109/ICRA46639.2022.9811886

[96] Zhefan Xu, Yumeng Xiu, Xiaoyang Zhan, Baihan Chen, and Kenji Shimada. 2023. Vision-aided UAV Navigation and Dynamic Obstacle Avoidance using Gradient-based B-spline Trajectory Optimization. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, London, United Kingdom, 1214–1220. https://doi.org/10.1109/ICRA48891.2023.10160638

[97] A. Yamashita, T. Arai, Jun Ota, and H. Asama. 2003. Motion Planning of Multiple Mobile Robots for Cooperative Manipulation and Transportation. *IEEE Trans. Robot. Automat.* 19, 2 (April 2003), 223–237. https://doi.org/10.1109/TRA.2003.809592

[98] Ziwei Yan, Liang Han, Xiaoduo Li, Jinjie Li, and Zhang Ren. 2023. Event-Triggered Optimal Formation Tracking Control Using Reinforcement Learning for Large-Scale UAV Systems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, London, United Kingdom, 3233–3239. https://doi.org/10.1109/ICRA48891.2023.10160532

[99] Guang Yang, Mingyu Cai, Ahmad Ahmad, Amanda Prorok, Roberto Tron, and Calin Belta. 2023. LQR-CBF-RRT*: Safe and Optimal Motion Planning. http://arxiv.org/abs/2304.00790 arXiv:2304.00790 [cs, eess].

[100] Esen Yel and Nicola Bezzo. 2020. GP-based Runtime Planning, Learning, and Recovery for Safe UAV Operations under Unforeseen Disturbances. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Las Vegas, NV, USA, 2173–2180. https://doi.org/10.1109/IROS45743.2020.9341641

[101] Eiichi Yoshida and Ko Ayusawa. 2020. Towards Unified Framework for Trajectory Optimization Using General Differential Kinematics and Dynamics. In *Robotics Research: The 18th International Symposium ISRR*. Springer, 217–232.

[102] Jingjin Yu and Steven M. LaValle. 2016. Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics. *IEEE Trans. Robot.* 32, 5 (Oct. 2016), 1163–1177. https://doi.org/10.1109/TRO.2016.2593448

[103] Eyal Zehavi and Noa Agmon. 2021. Hybrid Path Planning for UAV Traffic Management. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Prague, Czech Republic, 6427–6433. https://doi.org/10.1109/IROS51168.2021.9636390

[104] Ji Zhang, Chen Hu, Rushat Gupta Chadha, and Sanjiv Singh. 2019. Maximum Likelihood Path Planning for Fast Aerial Maneuvers and Collision Avoidance. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Macau, China, 2805–2812. https://doi.org/10.1109/IROS40897.2019.8967828

[105] Boyu Zhou, Fei Gao, Jie Pan, and Shaojie Shen. 2020. Robust Real-time UAV Replanning Using Guided Gradient-based Optimization and Topological Paths. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Paris, France, 1208–1214. https://doi.org/10.1109/ICRA40945.2020.9196996

[106] Dingjiang Zhou and Mac Schwager. 2016. Assistive collision avoidance for quadrotor swarm teleoperation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Stockholm, 1249–1254. https://doi.org/10.1109/ICRA.2016.7487256

[107] Shuqin Zhu and Shahram Ghandeharizadeh. 2023. Flight Patterns for Swarms of Drones. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 29–33. https://doi.org/10.61981/ZFSH2303

[108] Baskın Şenbaşlar, Wolfgang Hönig, and Nora Ayanian. 2023. RLSS: Real-Time, Decentralized, Cooperative, Networkless Multi-robot Trajectory Planning using Linear Spatial Separations. *Auton Robot* 47, 7 (Oct. 2023), 921–946. https://doi.org/10.1007/s10514-023-10104-w