Challenges in Using Drone Swarms as Video Game Platforms

Minhyuk Park[1] and Tsz-Chiu Au[2]

***Abstract*— One of the most appealing applications of drone swarms is drone light shows, in which a group of drones displays a sequence of light patterns in the sky by changing from one drone pattern to another. The surge of drone light shows around the globe have proved its popularity as an entertainment platform for the mass. In this paper, we go one step further to consider using drone swarms as video game platforms and discuss the technical challenges in building such platforms. We discuss the pros and cons of drone-swarm-based video game platforms and present the limitations of using drone swarms as 3D displays for gaming. After that, we formulate the planning problems for achieving a high frame rate and the detection of virtual collisions between the objects represented by different drone swarms. Finally, we discuss the outlook of drone-swarm-based video games in the near future.**

I. INTRODUCTION

Drone light shows have emerged as popular entertainment worldwide in the past decade. Drone light shows are a vital and environment-friendly substitution for fireworks in celebrations and festivals. Moreover, drone light shows are also effective broadcast media for disseminating information to people in a local region. Given the success of drone light shows, we should consider other types of entertainment that utilize the capability of drone swarms. Among all entertainment in the present day, video games are arguably the most popular entertainment enjoyed by people in all age groups. How spectacular drone-swarm-based video gameplay would be if the game used a large portion of the sky as a computer monitor? Some existing drone light shows have already demonstrated the potential of playing video games with drones. Fig. 1 shows a drone light show promoting the new Super Mario Bros. movie. Fig. 2 shows a scene in an Ultraman drone light show at Kobe Meriken Park in Japan. These drone light shows have caught many audiences' eyes.

By exploiting the physical capability of drones, we intend to develop new genres of video games called *drone-swarm-based games*. Both Fig. 1 and Fig. 2 have illustrated the possibility of using drone swarms as a video game platform. Still, they also highlight some technical challenges such as slow frame rate, slow response time, etc. In this paper, we discuss the technical challenges of using drone swarms for gaming. We define the pixel formation planning problem for drone swarms to maintain the illusion of animations, which is crucial for video games. We also discuss some approaches for detecting the collision of two virtual objects represented by two independent groups of drones.

[1][2]Department of Computer Science and Engineering, Ulsan National Institute of Science and Technology, South Korea. {sqmemory13,chiu}@unist.ac.kr

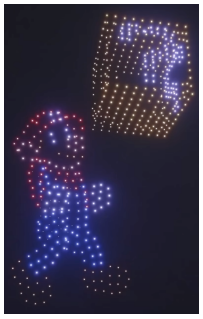

Fig. 1: Mario jumps to hit a bonus brick.[3]

Fig. 2: Ultraman throws a laser knife at a monster.[4]

We shall consider the drone-swarm systems that are the same ones for drone light shows. These are outdoor drone swarm systems equipped with an LED light bulb that can shine a wide range of color light in all directions. The drones in the drone swarms are small-sized drones remotely controlled by wireless communication networks. Global Positioning System (GPS) achieves localization of the drones, often augmented with Real Time Kinematic Positioning (RTK). We do not consider indoor drone swarm systems such as Crazyflie [23], though nano quadcopters can be used for drone-swarm-based video games for indoor entertainment.

This paper is organized as follows. In Sec. II, we survey the works pertinent to drone light shows and the use of drones for entertainment, sports, and gaming. Sec. III presents the technical challenges of using drone swarms as video game platforms. Sec. IV and Sec. V briefly describe our approaches to address some technical challenges. Finally, Sec. VI discusses the outlook of drone-swarm-based gaming.

II. RELATED WORK

Some research and commercial endeavors have already capitalized on utilizing drone swarms for entertainment. Based on technologies such as RTK GPS [31] and sophisticated in-flight planning algorithms [9], drone light shows have attracted significant attention in both research and commercial sectors [12], [19]. Weng et al. described a multi-view approach for drone light show [30]. Mao et al. presented a drone swarm light show design platform for K-12 children [18]. Drones can enhance existing entertainment as well. For example, drones can participate in music performances, during which multi-copters physically hit levers to play a musical instrument during flight [14]. In drone-augmented dance, a performer can dance with flying drones [13]. In virtual tourism, drones were used along with

[3]https://www.youtube.com/watch?v=shyKVbrbtnA
[4]https://www.youtube.com/watch?v=a-fOhLjMI0U

virtual reality goggles for virtual tourists visiting remote environments [14]. Kim et al. presented a survey of drone use for entertainment and augmented/virtual reality [15]. Drones have also found applications in competitive sports. World Air Sports Federation officially recognizes drone racing and drone soccer and hosts rules for competitions [11]. Drone fishing utilizes drones to assist in bait casting [5]. Drone surfing uses drones as a tug for surfers [7].

Drone-based games have started to gain popularity in recent years. A notable example is a game developed by Parrot, which enables two-player multiplayer interactions and allows players to virtually engage in aerial dogfights via their mobile phone screens [22]. Similarly, Drone Prix is a virtual drone racing game where pilots compete for time in an augmented reality race course [8]. Like robot combat, drone combat is a new kind of robot competition based on drone dogfight matches [1]. Kljun et al. introduced StreetGamesz, a gaming platform based on a moving projector platform (MPP) in which drones carry projectors to project game-playing elements at different locations [17]. Through motion tracking and projection of a playing area that can move and follow players, StreetGamesz could facilitate multiplayer street games like a jump rope. These works clearly show that the era of drone-based games has begun.

III. TECHNICAL CHALLENGES

The drone-swarm-based video game systems violate some assumptions of the existing computer graphics techniques for video games. This section presents some major technical challenges pertaining to drone-swarm-based video games.

A. Maintaining the Illusion of Animations

Maintaining the illusion of animations is very important to video games. On computer screens, animations are produced by showing a sequence of still images very quickly so that the characters in the images appear to be moving. We can partially replicate the effect of drone swarms by rapidly changing the color of LED lights. However, due to the distance between drones, the same pixel on a character looks like jumping from one location to another. Therefore, we need a new way to maintain the illusion of animations, given the physical constraints of drone swarms.

B. Collision Detection of Virtual Objects

Collision detection is a commonly used operation in video games. For example, a fighting game has to check whether two characters have touched each other in a fight. A shooting game has to detect whether a bullet hits a character. Suppose an object is represented by a set of drones that depict the outline of the object. Then, the *virtual* collision detection is the detection of the intersection of the virtual objects presented by the drone swarms.

C. Human-Drone Physical Interaction

If we can make drones safe enough for humans to touch them, drone touching can be a type of user input that a game can utilize so that audiences can provide some degree of control over the game. For example, two drones can fly toward the audience for them to cast votes for two different options in a game. Some existing works on human-drone physical interaction include GridDrones, which has individual drones representing a voxel to render a graphical user interface [4]. Abtahi et al. presented a touch-based interaction system with quadcopters to accomplish basic control tasks such as simple translation movements and landing [2].

D. Other Challenges

Reliable and real-time communication between a drone swarm and a controlling system of the drone swarm is critical for smooth gameplay. Network latency or unreliable connections can result in delays or interruptions, affecting the player's experience. Innovative techniques are required to ensure network stability and low-latency communication.

Sounds and background music are important elements in video games. If the audience and the players are put in some designated locations, we can use loudspeakers to broadcast the audio. Audiences outside these locations would have to steam the audio over the Internet.

IV. MAINTAINING THE ILLUSION OF ANIMATIONS

Since the era of Atari games, video games typically consist of the following graphical elements: 1) a character or an object that is under a player's control, 2) non-player characters (NPCs) that interact with the player's character, 3) a background which sets the situation for the interaction among characters or objects, 4) visual effects such as explosions and bullets, 5) information display such as game scores and health bars, and 6) transition scenes between two consecutive sections of a game. To port these elements to drone-swarm-based video games, we need to manipulate drones to display the pixels of these elements. In this section, we present the pixel formation planning problem, which is the key problem for providing an illusion of animations with a performance guarantee in drone-swarm-based video games.

A. Pixel Trajectory Tracking vs. Pixel Hopping

Suppose we use drones to represent a 3D character. Most of these drones are used as *pixels* on the edges of the 3D shape of the character, while the remaining drones are used as pixels to represent various features of the character. The question is how we can move the pixels for animation subject to the physical constraints of the drone swarm.

There are two ways to make the pixels move: *pixel trajectory tracking* and *pixel hopping*. In pixel trajectory tracking, a drone moves along a trajectory while the LED light is on. In pixel hopping, a drone switches off its LED light, and an adjacent drone turns on its LED light as if the pixel jumps from one drone to another. We prefer pixel trajectory tracking to pixel hopping for smooth animation. Still, pixel hopping is beneficial when a character has to move quickly, such that some pixels have to move faster than the drone's maximum speed. However, pixel hopping creates some discontinuity in the movement of pixels and

requires another drone to move to the target location ahead of time.

In multirobot systems, there are many works on controlling a team of robots to form and maintain a formation (e.g., [27]). These works are suitable for pixel trajectory tracking but not pixel hopping. Wang and Rubenstein considered task swapping during shape formation [28], which can be used for pixel hopping if the task is to show the pixels. However, task swapping is not pixel hopping—there should be dedicated algorithms for pixel hopping that provide a certain performance guarantee for animation.

B. The Pixel Formation Planning Problem

Let us define the drone-swarm-based pixel formation planning task for animation with some performance guarantee. Let $\mathcal{V}$ be a set of n drones. Let m be the number of pixels on a character, where $m \leq n$. We assume the number of pixels on a character remains the same during animation. The duration of the animation is T. We divide the duration evenly into h time steps. Let the start time of the time steps be $t_1, t_2, \ldots, t_h$, and the duration of each time step be $l = T/h$. A *frame* at time t_i is the configuration of all pixels: $f_i = \{(x_j, y_j, z_j, c_j)\}_{1 \leq j \leq m}$, where (x_j, y_j, z_j) is the *target location* of the pixel j with respect to the world frame, and c_j is the color of the pixel j. We say (x_j, y_j, z_j, c_j) is the *pixel configuration* of pixel j in f_i. Given a sequence of frames $F = \langle f_1, f_2, \ldots, f_h \rangle$, one for each time step, our task is to control the drones in $\mathcal{V}$ such that the following *hard requirement* is satisfied: for every $f_i \in F$ there exists a mapping g_i from $[1, m]$ to $[1, n]$ such that for each $(x_j, y_j, z_j, c_j) \in f_i$, the drone $\upsilon_k \in \mathcal{V}$ is at most distance D_{limit} from (x_j, y_j, z_j) at time t_i and the color of the LED light of υ_k is c_j, where $k = g_i(j)$. We say υ_k is the *selected* drone for the pixel j according to g_i, and D_{limit} is the *mislocation limit*, which is the maximum deviation from the target location of a pixel in a frame. In addition, there is a *soft requirement*: between two adjacent frames f_i and f_{i+1}, the selected drone υ_k for the pixel j should move from (x_j^i, y_j^i, z_j^i) to $(x_j^{i+1}, y_j^{i+1}, z_j^{i+1})$ at a constant speed with the LED color c_j^i, where $(x_j^i, y_j^i, z_j^i, c_j^i) \in f_i$ and $(x_j^{i+1}, y_j^{i+1}, z_j^{i+1}, c_j^{i+1}) \in f_{i+1}$ are the pixel configurations of the pixel j in f_i and f_{i+1}, respectively. We do not require all selected drones in all frames to satisfy the soft requirement; instead, we maximize the number of selected drones in all frames that satisfy the soft requirement.

We define the control of the drones as follows. A pose ρ for a drone $\upsilon \in \mathcal{V}$ is the position and the orientation of υ in the world frame. More precisely, ρ is $(x, y, z, \theta^x, \theta^y, \theta^z)$, where (x, y, z) is the coordinate of the center of υ and θ^x, θ^y, and θ^z are the pitch, yaw and roll rotations of υ, respectively. A *formation* F for $\mathcal{V}$ is $\{\rho_k\}_{1 \leq k \leq n}$, where ρ_k is the pose of $\upsilon_k \in \mathcal{V}$. The *distance* between two poses $\rho_1 = (x_1, y_1, z_1, \theta_1^x, \theta_1^y, \theta_1^z)$ and $\rho_2 = (x_2, y_2, z_2, \theta_2^x, \theta_2^y, \theta_2^z)$ is $\mathsf{dist}(\rho_1, \rho_2)$ which is the Euclidean distance between (x_1, y_1, z_1) and (x_2, y_2, z_2). A *motion plan* π for a drone υ is a sequence of control commands for controlling υ. When υ acts according to a motion plan π starting with an initial pose ρ^1, a drone moves along a *trajectory* ζ, which is a sequence of poses of υ. The pose of υ at time t on a trajectory ζ is denoted by $\zeta(t)$. Thus, $\zeta(0) = \rho^1$. A *formation plan* for $\mathcal{V}$ is $\Pi = \{\pi_k\}_{1 \leq k \leq n}$, where π_k is a motion plan for $\upsilon_k \in \mathcal{V}$. Given a formation plan $\Pi = \{\pi_k\}_{1 \leq k \leq n}$ for $\mathcal{V}$ and an *initial formation* $F^1 = \{\rho_k^1\}_{1 \leq k \leq n}$ where ρ_k^1 is the initial pose of $\upsilon_k \in \mathcal{V}$, a *formation trajectory* Ξ is $\{\zeta_k\}_{1 \leq k \leq n}$, where ζ_k is the trajectory of υ_k according to π_k starting at ρ_k^1. Moreover, a *formation* F of $\mathcal{V}$ at time t is $\Xi(t) = \{\rho_k\}_{1 \leq k \leq n}$, where $\rho_k = \zeta_k(t)$ is the pose of υ_k at time t in $\zeta_k \in \Xi$. A formation F is *safe* if and only if $\mathsf{dist}(\rho_{k_1}, \rho_{k_2}) \leq D_{\mathsf{safe}}$ for all $\rho_{k_1}, \rho_{k_2} \in F$ where $k_1 \neq k_2$ and D_{safe} is the *safe distance* between two drones. A formation plan Π for an initial formation F_1 is *safe* for $\mathcal{V}$ with F^1 if and only if the formation F for $\mathcal{V}$ at time t is safe for all $0 \leq t \leq T$.

The LED lights on υ are controlled by a *light schedule* ψ, where $\psi(t)$ is the color of the LED light at time t. A *formation light schedule* is $\Psi = \{\psi_k\}_{1 \leq k \leq n}$, where ψ_k is a light schedule for υ_k. A *drone light plan* for $\mathcal{V}$ is a pair (Π, Ψ), where Π is a formation plan and Ψ is a formation light schedule. A drone light plan is safe for an initial formation F_1 if and only if Π is safe for $\mathcal{V}$ with F^1. Given a drone light plan (Π, Ψ) and an initial formation F^1, a *drone light configuration* of υ_k at time t is (ρ_k, c_k), where ρ_k is the pose of υ_k at time t and c_k is the color of the LED light on drone υ_k at time t.

We assume drones are given enough time to move to their designated locations in the initial formation F^1 before executing a drone light plan (Π, Ψ). Given a sequence of frames $F = \langle f_1, f_2, \ldots, f_h \rangle$, the objective of *pixel formation planning* is to find an initial formation F^1 and a safe drone light plan (Π, Ψ) with F^1 such that 1) the hard requirement is satisfied (i.e., in every frame f_i there exists a mapping g_i such that the selected drone according to g_i is at most D_{limit} from the corresponding pixel's target location in f_i and the LED color is correct), and 2) the soft requirement is satisfied as much as possible (i.e., maximizing the total number of selected drones that satisfy the soft requirement in all frames). In this definition, the hard requirement provides a performance guarantee on the maintenance of the 3D shape of the character. By contrast, the soft requirement states the preference for pixel movement to pixel swapping.

Our pixel formation planning problem is similar to the *formation reconfiguration* problem, which aims to find a formation plan to transform an initial formation configuration into a final configuration [10]. But our focus is on the formation of pixels instead of drones. Sometimes, no solution satisfies the hard requirement, especially when n is smaller than m. One of our future works is to figure out the conditions on the sequence F of frames under which the pixel formation planning problem can be solved easily with the performance guarantee.

If the size of a character changes over time, the number m of pixels also changes. To expand a character, some additional drones with no LED light must move in position and then turn on their LED light. To shrink a character, some drones have to turn off their LED light and leave the swarm.

Fig. 3: A game of Space Invaders for Atari 2600.[6] The red box indicates the location of the character controlled by a player. The green box highlights an explosion visual effect. The cyan box contains the game score.

In the future, we will extend our model to facilitate smooth expansion or shrinkage.

C. Animation of Backgrounds and Visual Effects

In most cases, the animation of a background is mostly the same as the animation of a character and an NPC. The difference is that background objects must frequently appear and then disappear from a scene. The background could be a 3D outline that may gradually attenuate beyond a particular range. For example, explosion is a common visual effect in video games. The challenge of displaying an explosion is that pixels in an explosion spread widely in a scene. Therefore, we need to develop new algorithms for the animation of backgrounds and visual effects in drone-swarm-based games.

D. Information Display and Transition Scenes

A game typically needs to show some information, such as scores, timers, and health bars in specific locations in a scene. For example, Fig. 3 shows an Atari game called Space Invaders, in which the scores are shown at the top of the screen. Unlike movable characters or objects, this information can be presented by a group of drones in matrix formation. The drones act like a dot matrix LED screen that shows the information by turning on and off their LED lights or changing the LED colors. Both the drones for information display and transition scenes can be put on a different 2D plane separated from the main scene so that they would not interfere with the animation of other objects in the scene.

V. VIRTUAL COLLISION DETECTION

Suppose two drone swarms represent two different characters that interact with each other in a game. If a centralized controller controls the drones in both drone swarms, the virtual collision detection of the characters can be achieved by running a software simulation in the background that synchronizes with the drone swarms. Then, we can utilize a physics library such as PhysX from Nvidia for collision detection. PhysX implements several filtering stages in collision detection pipelines to reduce the computational cost and enable real-time collision detection. In robotics research, there are algorithms for collision detection between objects represented by point cloud data read by lidars or other sensors. Klein and Zachmann proposed to calculate implicit surfaces from point clouds to identify potential intersections of the point clouds [16]. Schauer and Nüchter explored the use of K-D tree search to check for possible collisions of a model represented by a 3D point cloud moving through a 3D point cloud of an environment [25]. Pan et al. used an axis-aligned bounding box tree for real-time collision detection and distance computation on point cloud sensor data [20]. For more information about collision detection in computer graphics, please refer to [29].

In multi-player games in which players bring their own drone swarms that are controlled separately, we cannot use a centralized approach for virtual collision detection. Instead, we have to rely on some communication protocols to inform the controllers of different drone swarms about any virtual collision. Moreover, we need to prevent any physical collision of drones from different drone swarms. Some unmanned aircraft systems' (UAS) traffic management systems include provisions of multiple UAS operators handling their own swarms of UAS. To avoid potential collision between UAS of different operators, various communication protocols for inter-swarm communication for collision avoidance have been devised [6], [21], [24]. Some well-known decentralized collision avoidance algorithms are the buffered Voronoi cells [32], artificial potential field method [27], [28], [26], and the Optimal Reciprocal Collision Avoidance (ORCA) method [3]. These algorithms take relative state information of other neighboring agents and reactively change their trajectory if a collision is imminent.

VI. THE OUTLOOK

Drone-swarm-based gaming could become a niche entertainment alongside drone light shows. There could also be some exciting interplay between drone-swarm-based games and augmented-reality games. Compared with augmented reality games, drone-swarm-based games do not require audiences to carry augmented reality devices, making them more suitable for broadcasting the gameplay to the masses on special occasions. However, augmented reality games can enhance drone-swarm-based games with a higher level of realism. However, the cost of running a drone-swarm-based video game platform could remain high since we have yet to automate a drone swarm fully. Ultimately, the success of drone-swarm-based video games hinges on the innovation of game developers who come up with some hit game titles designed specifically for drone-swarm-based platforms.

ACKNOWLEDGMENTS

This work has been taken place at UNIST and was supported by NRF (2022R1A2C101216812).

[6] https://www.youtube.com/watch?v=_ftVrgJTl4w

REFERENCES

[1] Aerial sports league. https://www.aerialsports.tv/drone-combat, 2023.

[2] Parastoo Abtahi, David Y Zhao, Jane L E, and James A Landay. Drone near me: Exploring touch-based human-drone interaction. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–8, 2017.

[3] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed autonomous robotic systems: The 10th international symposium*, pages 203–216. Springer, 2013.

[4] Sean Braley, Calvin Rubens, Timothy Merritt, and Roel Vertegaal. Griddrones: A self-levitating physical voxel lattice for interactive 3D surface deformations. In *UIST*, volume 18, page D200, 2018.

[5] Nick Cast. What is drone fishing and how it works? https://www.remoteflyer.com/how-does-drone-fishing-work/, 2023.

[6] Anjan Chakrabarty, Corey A Ippolito, Joshua Baculi, Kalmanje S Krishnakumar, and Sebastian Hening. Vehicle to vehicle (V2V) communication for collision avoidance for multi-copters flying in UTM–TCL4. In *AIAA Scitech 2019 Forum*, page 690, 2019.

[7] Elizabeth Ciobanu. What's the buzz around drone surfing (and is it legal)? https://www.droneblog.com/whats-the-buzz-around-drone-surfing-and-is-it-legal/, 2023.

[8] DJI. Edgybees launches the first augmented reality game for dji drone users. https://www.dji.com/newsroom/news/edgybees-launches-the-first-augmented-reality-game-for-dji-drone-users, 5 2017.

[9] Xintong Du, Carlos E. Luis, Marijan Vukosavljev, and Angela P. Schoellig. Fast and in sync: Periodic swarm patterns for quadrotors. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9143–9149, 2019.

[10] Haibin Duan, Qinan Luo, Yuhui Shi, and Guanjun Ma. Hybrid particle swarm optimization and genetic algorithm for multi-uav formation reconfiguration. *IEEE Computational Intelligence Magazine*, 8(3):16–27, 2013.

[11] FAI. Fai sporting code aeromodelling volume F9 drone sport. https://www.fai.org/sites/default/files/ciam/wcup_drones/sc4_vol_f9_dronesport_22_2022-03-01_0.pdf, 3 2022.

[12] Jie Huang, Guoqing Tian, Jiancheng Zhang, and Yutao Chen. On unmanned aerial vehicles light show systems: Algorithms, software and hardware. *Applied Sciences*, 11(16):76–87, 2021.

[13] Heesoon Kim and James A Landay. Aeroquake: Drone augmented dance. In *Proceedings of the 2018 Designing Interactive Systems Conference*, pages 691–701, 2018.

[14] Si Jung Kim, Yunhwan Jeong, Sujin Park, Kihyun Ryu, and Gyuhwan Oh. A survey of drone use for entertainment and avr (augmented and virtual reality). *Augmented Reality and Virtual Reality: Empowering Human, Place and Business*, pages 339–352, 2018.

[15] Si Jung Kim, Yunhwan Jeong, Sujin Park, Kihyun Ryu, and Gyuhwan Oh. *A Survey of Drone use for Entertainment and AVR (Augmented and Virtual Reality)*, pages 339–352. Springer International Publishing, Cham, 2018.

[16] Jan Klein and Gabriel Zachmann. Point cloud collision detection. In *Computer Graphics Forum*, volume 23, pages 567–576. Wiley Online Library, 2004.

[17] Matjaž Kljun, Klen Čopič Pucihar, Mark Lochrie, and Paul Egglestone. Streetgamez: A moving projector platform for projected street games. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, pages 589–594, 2015.

[18] Pengda Mao, Yan Gao, Bo Wang, An Yan, Xiaoyu Chi, and Quan Quan. Fast light show design platform for K-12 children. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9396–9402, 2021.

[19] Dharna Nar and Radhika Kotecha. Optimal waypoint assignment for designing drone light show formations. *Results in Control and Optimization*, 9:100–174, 2022.

[20] Jia Pan, Ioan A Şucan, Sachin Chitta, and Dinesh Manocha. Real-time collision detection and distance computation on point cloud sensor data. In *2013 IEEE International Conference on Robotics and Automation*, pages 3593–3599, 2013.

[21] Jong-Hong Park, Sung-Chan Choi, Jaeho Kim, and Kwang-Ho Won. Unmanned aerial system traffic management with wave protocol for collision avoidance. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 8–10, 2018.

[22] Parrot. AR.Drone. http://ardrone.parrot.com/parrot-ar-drone/usa/ar-games, 2015.

[23] James A. Preiss, Wolfgang Hönig, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304, 2017.

[24] Stefano Primatesta, Matteo Scanavino, Andrea Lorenzini, Francesco Polia, Enrico Stabile, Giorgio Guglieri, and Alessandro Rizzo. A cloud-based vehicle collision avoidance strategy for unmanned aircraft system traffic management (utm) in urban areas. In *2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 309–313, 2020.

[25] Johannes Schauer and Andreas Nüchter. Efficient point cloud collision detection and analysis in a tunnel environment using kinematic laser scanning and kd tree search. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40:289–295, 2014.

[26] Hang Sun, Juntong Qi, Chong Wu, and Mingming Wang. Path planning for dense drone formation based on modified artificial potential fields. In *2020 39th Chinese Control Conference (CCC)*, pages 4658–4664, 2020.

[27] Matthew Turpin, Nathan Michael, and Vijay Kumar. Trajectory design and control for aggressive formation flight with quadrotors. *Autonomous Robots*, 33:143–156, 2012.

[28] Hanlin Wang and Michael Rubenstein. Shape formation in homogeneous swarms using local task swapping. *IEEE Transactions on Robotics*, 36(3):597–612, 2020.

[29] René Weller. A brief overview of collision detection. *New Geometric Data Structures for Collision Detection and Haptics*, pages 9–46, 2013.

[30] Kai-Chun Weng, Shu-Ting Lin, Chen-Chi Hu, Ru-Tai Soong, and Ming-Te Chi. Multi-view approach for drone light show. *The Visual Computer*, pages 1–12, 2022.

[31] Anhua You, Jiahao Li, Zifang Xu, and Wencong Liu. Design of entertainment drone with rotating led display technology. In *Journal of Physics: Conference Series*, volume 2113, pages 012–058. IOP Publishing, 2021.

[32] Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017.