Do A Barrel Roll, But In Real Life

Joshua Pulido
jpulido@oxy.edu
Occidental College

1 Problem Context

With the rapid development of technology in both the virtual reality and drone fields, what were once niche experiences have now become household brands, especially thanks to companies like Meta and DJI. That begs the question: why has there not been software that brings the two spaces together in a entertaining way? While there are certainly methods for using VR headsets to view drone footage, very little effort has been put into pushing the space further and innovating in the augmented reality field using drones. My goal is to be among the first to look into the space; more specifically, I plan on creating a video game that has the player fly a real world drone, while interacting with a virtual environment using the other buttons of a game controller.

Before talking about the game itself, it is notable to mention that such a game would bend the definition of augmented reality (AR) as commonly known today. The most popular AR games, including the popular game Pokemon GO, often focus on overlaying digital objects on top of the real world, thereby "augmenting reality". Virtual reality (VR) loses the "reality", instead focusing on creating an immersive virtual world that feels as real as possible. While augmented reality is not limited to smartphones, other devices such as Microsoft's HoloLens have not yet reached the commercial viability of their VR counterparts. This is where the use of a drone, with it's included camera, comes into play. By combining streaming the video feed from a drone to a VR headset, an AR game becomes possible where it otherwise would not be; being able to physically fly the drone around comes as a bonus that adds to the entertainment value and allows for interaction with the real world.

It is important to note that we are still relatively early in what will likely be a long history for both the VR/AR and drone spaces of technology, and as such the two are not necessarily designed to function together. Nevertheless, I have multiple ideas for how to make them cooperate, which I will go more in depth on in the Technical Background section. The obstacles may be difficult to overcome, but through a series of compromises, I believe that a prototype game can be developed, with the goal of creating a more integrated system in the future. There are also ethical considerations that must be made, such as whether or not inexperienced pilots should be allowed to fly drones, especially while trying to play a game at the same time. Within the scope of this project, it may not be possible to address such issues; however, in the creation of a full game that would be released to the general population, they would more than likely be dealt with through a required training program before being allowed to pilot the actual drone.

2 Technical Background

In an effort to capture the feel of the first person view (FPV) headsets that are traditionally used to fly drones, it makes the most sense to use a VR headset. Notably, recent advances in mobile technology have led to a split in the VR headset field, with some headsets requiring a connection to a computer and others having the ability to be completely independent. The ideal way of making this game would be the development of an application that runs natively on an "all-in-one" headset, most likely the Oculus Quest 2. However, depending on other factors that could come into play, it might not be possible to run this type of game on Android based systems. In this case, it would make the most sense to use a PC to run the game alongside a headset such as the Valve Index. Regardless of which platform ends up being preferred, the most obvious solution for developing a 3-dimensional (3D) video game would be Unity.

Although there are many popular game engines, the two that come to mind for 3D video game development are Unity and the Unreal Engine. I have chosen Unity because of its integration with the C# language, for which there are many APIs that allow for computers to interact with drones. Unity also provides many built in tools that help simplify the game design process. Regardless of what the final gameplay ends up being, one important feature I want to include is the procedural generation of levels. There are algorithms out there that can create some very detailed levels, but since my game will have to be relatively simple, I imagine that I will end up making my own. Currently, my idea is to create an algorithm that takes the dimensions of the environment as input, up to a max value equivalent to the drone's range. Then, using a random number generator and pre-established rules for game features, I hope to create a graph, where each node is a feature, such as an enemy or a power-up, and the weights are the distance between each item. The final part of the algorithm would just perform a clean up, again based on rules that it is fed. With that being said, there is still a lot more to determine before writing this algorithm, such as where the inputs are collected. Ideally, this would be done using sensors on the drone itself, but not every drone has access to distance-sensors. There is also the challenge of mapping the real world to the virtual environment by defining what 1 meter in the real world means in game. It is common for games to exaggerate in game physical values to make them more fun and engaging, but in the case of this game, doing so could potentially detract from the immersiveness. Therefore, there will have to be a balance between the entertainment value and the realness that the game offers. There is also the potentially difficult and non-trivial issue of getting a drone to interact with a computer.

Currently, the industry standard for piloting drones is through the use radio frequencies to transmit the required data to control a drone. This includes the instructions given to the drone by a controller, as well as the video feed given to an FPV headset by an onboard camera. Therefore, interacting with most drones in the market through a computer would require radio-capable hardware that is uncommon on PCs. With that said, drone company DJI has set a precedent of using the WiFi-standard 2.4 and 5 Ghz frequencies for their drones, allowing computers to easily connect through their WiFi chips, which potentially eliminates an entire step in the process. It is important to note, however, that latency plays a huge issue when working with any device over wireless communication, and in this case it is especially important to evaluate the latency of the video stream. One study on latency[5], performed by researchers at the National Institute of Information and Communications Technology in Yokosuka, Japan, found that the use of the 2.4 GHz radio band caused major latency issues due to the large amounts of interference. Although their research focused more on beyond-line-of-sight flight, the latency still poses an issue at closer distances, and worsens with the amount of wireless devices in the area. Theoretically, even with the interference and latency issues, the video output should be sufficiently fast enough to provide for a playable experience, but in practice this could end up not being the case. While it would be preferred to have a perfect product, compromises will clearly have to be made to make this project possible. For this reason, I have decided to use DJI and Ryze's Tello drone, which has a public SDK and a large developer community. While the video quality is not the highest resolution, at 720p, the latency should be reasonable within approximately 100 meters from the player. This might restrict what is possible with gameplay somewhat, but it should allow for a working end product, and support for other drones could be added after the fact, although that is likely outside the scope of this prototype.

3 Prior Work

In the department of video games that interact with real drones, very little has been accomplished. Only a single commercial game can be found using search engines that matches the description of this type of game. Named Drone Prix AR[7], this game by developer Edgybee was the first known attempt at creating an augmented reality game using a drone, in this case developed for the Android and iOS platforms. Despite lacking the immersiveness that a VR headset would bring, the game largely accomplishes a similar feel to my idea. However, it is important to note that the game no longer exists on both Google's Play Store and Apple's App Store, and there is very little evidence that the game ever existed. Perhaps this is not by coincidence - creating a game like this is something that few have done before, and given the current size of the drone market, such a product is likely not commercially viable. Nevertheless, the game and its idea, while novel, is one that many would gladly play. In fact, although there are very few video games that use AR technology with drones, there have been many other projects that run parallel to this idea. For example, one group of researchers[10] sought to create an augmented reality spectating application for drone FPV racing, and was written several years ago when the technology was not as capable as it is today. Their work mostly involved overlaying a visual UI and smart assistant that explained the race as it progressed, so it shares some similarities with my concept. However, they also used external motion detectors and other devices that will not be used in my project.

When it comes to working with Tello drones, the specific drone that I will be using in this project, there are many interesting projects that developers have made publicly available on Github. While most of these have nothing to do with video games, there are a couple that stand out as ones I can take some ideas from. One such project, created by user carter-james89[4], allows for the Tello to follow a "virtual drone" in Unity, with real time video output in the Unity game scene. Although this is not necessarily a video game, it proves that interaction with the drone in Unity using the C# library[6] created by user Kragrathea is possible. Another really interesting library is Joy[3], a simple tool developed by GiovanniGrieco that allows for the Tello drone to be flown using a traditional game controller, and written in Python. Unlike most of the other projects that use DJI's own API, Joy sends packets directly to the drone, which allows for faster responsiveness, an absolute necessity for my game to be playable. I believe that these libraries will provide me with sufficient information to make the Tello work with the game, but there is an active community of developers, including an entire website, dedicated to the Tello that

I can go to for any help if need be.

Aside from the technology side to this game, equally important is the actual gameplay. While I am still unsure exactly what genre of game I want to play, the core of this game will be an immersive flight experience, a necessity when a real drone is being flown. The biggest inspiration for my game is No Man's Sky, a space exploration game, particularly the space flight mechanics that exist. In the game, the player can fly around to collect resources, and can also optionally engage in combat with other starships. Despite a rocky release, and after significant updates, has received praise for being a "promising experience for those looking for more battles among the stars", and provides one of the most immersive gameplay experiences for both normal systems as well as VR[1]. In particular, the combat involves a lot of defensive maneuvers to avoid incoming attacks, while also providing "lock on" systems for attacking the enemy spaceships. The game's physics engine also tries to be as realistic as possible, while still maintaining an arcade-like feel. Even if the final product does not end up involving space-fighting, my hope is to capture a similar feel to the flight that No Man's Sky has. Alternatively, I could take it a different direction, and ditch the realism for a game like Star Wars: Squadrons. While the gameplay is very similar to that of No Man's Sky, the entire feel is different, ditching realism entirely for an experience that is focused purely on having fun. The game also provides a very fun multiplayer game which adds to the experience, something I would like to implement in my project although realistically do not think is possible.

On the AR side of things, my biggest source of inspiration is Pokemon GO. This game focuses mostly on collecting Pokemon which "appear" in the real world through augmented reality, which is admittedly completely different gameplay wise from the drone AR game I will be making. However, despite these differences, the game really accomplishes the same goal that I have, which is captivating players through interaction with the real world. Their use of GPS alongside the rendering of Pokemon in the real world with AR technology really makes the player feel as though they are in the Pokemon universe, and my hope is to do something similar. Although I will not be able to recreate the visual aesthetic of space, I want to make it feel as though the entire game is taking place in the real world, and not just the drone.

4 Ethical Considerations

When working with a drone, there are a large number of ethical considerations that should be taken into account. This section identifies just a few of them, and offers potential solutions to address the different ethical concerns that may arise. When it comes the use of a drone, both safety and privcacy concerns must be taken into account. Further, the environmental impact that such a game may have should be considered before actually entering production.

4.1 Inexperienced Pilots

One part of the game's goal is to introduce drone flight, at the most basic level, to new individuals who may be interested (although the game is not a learning tool, as mentioned prior). Therefore, it is reasonable to expect that many new players, if not the majority of them, will be flying a drone for the first time, and may not be familiar with the rules that have been put in place by the Federal Aviation Administration (FAA) concerning unmanned aerial systems (UAS), also known as drones. Many restrictions exist concerning where flight is and is not allowed; for example, flight in restricted airspace and near airports is explicitly prohibited[8]. However, it is not necessarily common knowledge for the average user, and thus must somehow be communicated to them through the game.

One of the largest fears with inexperienced pilots flying drones is their ability to hurt bystanders. Notably, the FAA does not prohibit the flight of drones around other people, but whether or not this game should be played around people who are unaware could be of concern. Even in the situation where gameplay is isolated, there remains the issue of the player crashing the drone into themselves or damaging property. This is an unfortunate byproduct of the novelty that including a real drone as part of the game, as the same could be said of racing drones in first person view (FPV) or even flying them for leisure. To solve this problem, it could be possible to implement a system which stops the drone from flying if it ever gets too close; however, this could prove to be difficult, if not impossible, depending on the number of sensors present on the drone. The Tello drone, which is being used in the development of the prototype, contains no sensors that could be useful for this, so another solution must be found. It could be possible to use the camera alongside image detection in place of a sensor, but this would require significantly more processing power than may be available while also running a VR game. Moreover, it also presents certain issues around security, and privacy.

4.2 Drone Camera + Data Collection

One of the leading arguments against the flight of drones is the privacy concerns that arise due to their aerial mobility and ability to navigate in ways that were previously impractical, if not impossible, for the average person. Many private citizens have expressed their fears over drones being used by individuals, such as their neighbors, to enter their properties and take photographs are stream live footage of their own homes. While this is a very important issue that should be addressed in the field as a whole, for this specific game the limited range and presence of the HUD make it sub-optimal for such use. Nevertheless, it is a real issue, and warnings should be placed that notify the player of the illegality of using the drone in such ways.

Another issue with privacy surrounds the use of data collected by the game, which would be used to enhance the gameplay experience in some ways. One feature that could be implemented in the final product is using sensors to scan the surrounding environment of the room prior to starting the game, and using this data to construct a game level that fits within the spatial constraints at any given place. The data would only be used immediately as a parameter to the algorithm; however, there would rightfully be privacy concerns about a 3rd party collecting information of the layout of every player's room. These concerns are amplified by other possible features, such as using the camera to detect people and other obstacles to prevent collisions. All of this image data is being processed by the computer, and it could be argued that bystanders are having their appearance processed by an algorithm without them having given explicit consent. Rodrigues, Kormann, and Al-Dulaimi[9] discuss the possibility of image recognition software complying to laws concerning privacy, and notes that it is possible to develop system that is fully compliant with EU law (and most likely US law as well). Regardless, such a system is impractical to develop for a video game. After all, the image recognition used by the game would be simple, only used for detecting that an object is nearby. Instead, a better solution might be to promise the users that none of the collected data is being stored, and to follow through on this. In order for the algorithms used by the game to function, they only need the data for a brief moment of time, after which it can promptly be deleted. Hopefully, providing a terms of use which explicitly states the way that data will be used would be sufficient to solve the privacy concerns, but it might be worth looking into alternative ways of object avoidance to avoid the issue altogether.

4.3 Environmental Concerns

With the exponential growth of the consumer electronics sector over the past few decades, one of the largest concerns that has arisen, especially in recent years, has been surrounding electronic waste (e-waste) and its environmental impact. Within the gaming industry alone, 34 terawatt-hours of energy are used annually across the globe, with this number expected to increase as gaming becomes a worldwide phenomenon [2]. Further, the quick improvements in technology contribute to constant releases, and even re-releases, of consoles, along with other hardware has led to lots of waste, including the 700,000 unsold Atari cartridges that were just buried in a New Mexico desert [2]. Thus, any video game must factor the environmental impact that they can have into the development and production cycles, especially if there are hardware components.

For this specific video game, there is a unique challenge in that it can not be assumed that the user will have the correct drone, or a drone at all. Thus, the most obvious solution is to provide a drone as part of a package with the game. While it could also be possible to use a commercial drone already in the market, and thereby decrease the environmental impact due to production, this would lessen the user experience as the user would be required to purchase a separate product in order to play the game. Thus, there are multiple tradeoffs, and unfortunately there is no real good solution to this issue. It might be possible to offset the environmental impact by promoting proper recycling of products within the game; however, the creation of this game would inevitably lead to a negative impact on the environment, much like every other game in the market.

5 Methods and Evaluation

In order to create a game in the time span of approximately 4 months with a team size of 1, a strict schedule will have to be followed to ensure that game systems and design are developed in time for the release of a prototype. There are distinct pieces of this game that can be taken as their own parts, which all come together to form the final product, notably the control scheme, game design, implementation of game systems in Unity. Given that this project aims to create a video game, the main criteria for evaluation is ”fun”. However, such ”fun” comes from different perspectives, and as such they must be considered in order to properly evaluate whether or not the game was a success. This is where user testing really comes into play throughout the whole process, to make sure that the final product is both playable and fun.

5.1 Control Scheme

Arguably one of the most important design elements for this game is the control scheme, which needs to be intuitive enough such that anyone can pick up a controller and learn how to fly the drone through trial and error. As a result, the initial focus of the project will be testing different configurations for flight, including multiple different controllers, in particular a traditional game controller and the Oculus Quest controllers.

Regardless of the actual controller used, the control scheme will involve mainly the use of the joysticks and buttons normally found on a game controller. With regards to the movement controls, the first control scheme will attempt to mimic a standard drone controller. However, this would most likely not be intuitive, as most modern games use the left joystick for translational movement and the right joystick for rotational movement, but these are inverted on a standard drone controller. As a result, the next logical step would be to swap these controls. However, another non-standard feature of drone controllers is the control of two separate features with a single joystick: vertical translation and rotation. To accommodate for this, the next control scheme that will be trialed will move vertical movement to the left and right triggers, for downward and forward movement, respectively, while the right joystick will now only control rotation using the horizontal input.

There will be additional controls that need to be mapped for different gameplay features, such as a button to use a weapon or use another ability, buttons to pause menu, and so forth. These can be anticipated in the control scheme, but will most likely be added after the game designed is finalized. Nevertheless, even during the early process, buttons can be selected based on mechanic types; for example, the ”A” and ”B” buttons can be reserved for primary features, while ”X” and ”Y” can be used for secondary mechanics, which is common across most games.

Once a few playable control schemes are designed, the final step would be to find the one that is most popular among players, through user testing. The main feature to be tested would be the feel of the movement, although other factors such as the ability to use other buttons during game play should also be considered. At the end of the testing period, the most popular control scheme will be selected as the default for all testing going forward; however, the other schemes can also be implemented as alternatives if they are popular among enough users.

5.2 Game Design

While the control scheme is being designed, it is also necessary to determine the exact design of the game. The first step in this will be deciding a concept, which currently is to make a game that emulates the feel of a Star Wars space fight, although the final concept might be limited slightly by what is actually possible in Unity. The next step would be to decide on the primary mechanics of the game; in this case, it is obvious that flight is one of the key mechanics, but others that are more gameplay-specific also must be decided. This could be a shield and weapon for a space-fighting game, or a speed boost for a racing game, and will depend on other decisions made about the game's key features.

Once these key elements of the game's design have been decided, a few different processes can be done in parallel. Much like in all other parts of the game's development process, user testing, or in this case surveying, should be done to see if the game design is something that a diverse group of players would actually want to experience. My main focus here would be to see if they would enjoy the idea, as well as the genre, and from there ask more specific questions about the planned features to get their opinions on how the game might play. The responses most likely we lead to some adjustments, or potentially even an entire redesign, but this would hopefully lead to an end product that is entertaining for a diverse group of people.

Level design can also begin at this stage, and will consist more of deciding what goes into a level and how to balance the game, rather than designing specific layouts. Since the game involves the real world, it might be difficult to create levels that work everywhere, so a procedurally generated system might work best, with the estimated size of the playfield being taken as an input and determined using sensors, or just as a setting in the game. As for what might appear in the level, this is largely determined by what features are actually implemented. If I stick to my idea for space-fighting game, this would include obstacles and power-ups that spawn in random locations, as well as the rules for when and how enemies are spawned into the game.

5.3 Unity Implementation

With regards to implementation in Unity, the first (and most important) part would be to get the virtual world in Unity to display in front of the camera output from the drone. There are a couple of different ideas here: either the video stream can somehow be placed as an actual background, which would be the ideal solution, or the video stream would be placed in world, but far away and very large, to simulate it being a background. Perhaps there is another solution, but this problem is the first that must be addressed, as it is the determining factor for what is and is not possible while using a real drone. As a part of this, resolution must also be taken into consideration; the video camera on the Tello drone streams in 720p, which is lower resolution than the current standard of 1080p, so some adjustments to the scaling of the game might have to be done to keep visual integrity.

Other game features, such as the primary mechanics, could also begin development immediately after the game design has been finalized, and would for the most part follow the standard procedure for making a game in Unity. This only gets complicated when the real drone is introduced, so movement and any other game mechanics that involve the drone would be require a more in depth development cycle. The first step in this would be getting the controller to interact with the drone through the various libraries that will be used, and making sure that the drone moves relatively smoothly. The end goal would be to tune the movement to where the game feels truly immersive, and user testing would assist in this endeavor. Alongside the mechanics, other core games systems, such as any AI for enemies, or the implementation of levels, can also take place. This would mostly involve writing algorithms and implementing them with Unity's built in systems.

For a simple AI, it would suffice to make enemies "lock on" to the player, using the transform.LookAt() method included in Unity, and move towards the player using the transform.MoveTowards() method. To keep the AI from being predictable, there will be animations which prevent the them from instantly locking on, allowing for the player to react to their actions and act accordingly. There will also be some decisions that the enemy is taking at every step. Currently, this involves deciding whether to shoot at the player, move closer, aim, or some mix of these actions, but can be expanded upon if more elements are added to the game. Note that these are specific to the dogfighting genre, and would be changed if a different direction was taken for the game design.

As mentioned above, the levels for this game will be procedurally generated, in the sense that each run through of the level will be a different experience, if not a completely new one. The first step in accomplishing this is to determine the different features that will be included, such as power-ups, enemies, obstacles, and other objects that might appear in the game world. Once this is decided, the next step would be to figure out where they might appear and implement rules for where they can and can not spawn. For example, a power-up should not spawn next to an enemy, because it would end up being useless in helping defeat the enemy. Once the rules are established, the final step would be to implement a balancing system, to ensure that the player is not overwhelmed by any one feature at a time. As a result, if there are many enemies, a player might be given the opportunity to obtain more power-ups than normal, to make sure the game is not only winnable, but also not frustrating.

5.4 User Testing

For the various stages of user testing, the goal is to conduct tests and surveys with a group of people who all are knowledgeable in one field of the project, such as flying drones or making games. This would be useful to get specific feedback on particular parts of the game. For example, someone knowledgeable in flying drones could properly evaluate whether or not the control scheme provides proper control over the drone, whereas a game developer could provide better feedback on how intuitive the control scheme is to learn and play with. However, I recognize that this provides for a consolidated base, so I would also like to include a few people who do not necessarily have any background knowledge, which will hopefully provide for different perspectives on the game. User testing will take course constantly during the development process, and each test will have specific features that are targeted alongside open ended questions where users can provide any additional feedback. Since this group will not be too large, it might make sense to keep all the questions open ended to get very specific feedback, all of which will be taken into consideration as changes are made after each test.

Once the game is considered "complete", there will be a final run of user testing, open to a larger group of people beyond those who were chosen for testing during the development process. This final test will focus more on how "fun" the game and user experience are as a whole. As a part of this, each player will be asked to first play the game, then rate the game based on how much they enjoyed it in a few factors. Perhaps the best way of doing this is to split it into a few scales based on the different parts of gameplay, including the feel of the flight, level design, and overall entertainment value. While these values are arbitrary, in general higher numbers would hint towards a more well designed game, whereas lower numbers would indicate significant room for improvement in the design and implementation of the game. One important piece of the user testing is also the playability of the game, otherwise known as how well the game performs. While a part of this is objective, there is also a subjective part to this, where every player might have a different experience in how they were able to play. Some might find it frustrating to fly a drone while playing, while others might find it too easy and desire a higher difficulty. While this is a question that should be asked throughout, it is most important at the end of the project to see whether or not the game has actually changed over time in response to the user feedback.

5.5 Playability

Alongside the subjective playability measures outlined above, there are also some objective measurements that can be taken. Given that this game is running on an Oculus Quest, the performance of the game on the VR headset could struggle if the game is not optimized, and so the average FPS should be measured and compared to the target of 30. In addition to the average FPS, the peaks and valleys should also be measured to identify any outliers, but they are unreliable and therefore should not necessarily be used to measure playability. Bugs and unintended gameplay should also be measured for playability; a buggy game is often not considered to be enjoyable. However, through optimization, user testing, and bug-fixing, both performance and bugs should be stable by the point of evaluation.

There are other measurements that should be taken which can not necessarily be remedied. The delay from the camera streaming from the drone to the computer, and the opposite direction, from the controller to the drone, are two of the most critical points of failure. If the perceived latency is too long, the game could end up being frustrating rather than entertaining. As a result, effort should be put in to minimize both these numbers as well as the effects they have. Since latency is in the nature of wireless technology, it might not be possible to eliminate this delay, so some should be expected. However, if it gets to a point where it is negatively impacting gameplay constantly, then that should be considered during the evaluation.

6 Proposed Timeline

After completing the tutorial report, and considering the methods and evaluation outlined in this paper, I have determined the following proposed timeline, of course subject to change.

- **End of August** - Design, Implement, and Test Control Scheme
- **Mid September** - Game Design + User Feedback on Control Scheme and Game Design
- **End of September** - Begin Implementing Game + User Testing
- **Mid October** - Continue Implementing Game + User Testing, (Pivot if necessary)
- **End of October** - Game Prototype Done + User Testing
- **Mid November** - Implement Feedback + User Testing
- **End of November** - Bug Fixing + Final User Testing & Evaluation
- **Mid December** - DONE!

A couple notes on the proposed timeline: designing and implementing the control scheme includes connecting the drone to Unity, which is one of the large parts of the project that I hope to get done over the summer. Implementing the game is more of rendering game objects over the camera stream, which is the part that may or may not be possible, I am not sure yet.

References

[1] Cryer, Hirun. *No Man's Sky: Outlaws update overhauls space combat with pirates and AI squadrons*. 2022. URL: `https://www.gamesradar.com/no-mans-sky-outlaws-announcement/`.

[2] D'Anasatasio, C. *Next-Gen Gaming Is an Environmental Nightmare*. 2020. URL: `https://www.wired.com/story/xbox-playstation-cloud-gaming-environment-nightmare/`.

[3] Grieco, Giovanni. *Joy*. 2021. URL: `https://github.com/GiovanniGrieco/Joy`.

[4] James, Carter. *UnityControllerForTello*. 2021. URL: `https://github.com/carter-james89/UnityControllerForTello`.

[5] Kagawa, T. et al. *A study on latency-guaranteed multi-hop wireless communication system for control of robots and drones*. Tech. rep. 2017. URL: `https://ieeexplore.ieee.org/abstract/document/8301849`.

[6] Kragrathea. *Tello Lib*. 2018. URL: `https://github.com/Kragrathea/TelloLib`.

[7] Murison, M. *Edgybees Launches Drone Prix Augmented Reality Game for DJI Pilots*. 2017. URL: `https://dronelife.com/2017/05/02/edgybees-augmented-dji-drone-prix/`.

[8] *Recreational Flyers And Modeler Community-Based Organizations*. 2022. URL: `https://www.faa.gov/uas/recreational_fliers/`.

[9] Rodrigues, M., Kormann, M., and Al-Dulaimi, M. *Data protection and privacy issues concerning facial image processing in public spaces*. Tech. rep. 2016. URL: `https://shura.shu.ac.uk/11080/1/root.pdf`.

[10] Sueda, K. et al. *Research and development of augmented FPV drone racing system*. Tech. rep. 2018. URL: `https://dl.acm.org/doi/10.1145/3283289.3283322`.