Drones autonomously following AprilTags

Giol Vinyals-i-Roca

Under the direction of

Professor Chuchu Fan
Charles Dawson and Kunal Garg
MIT AeroAstro, REALM

Research Science Institute
August 1, 2023

Abstract

Autonomous drones have become a significant area of interest, with numerous potential applications yet to be fully explored. This paper presents a software solution developed for the Tello drone, a versatile and affordable device, to enable it to autonomously follow AprilTags. AprilTags are used as visual markers to provide a reference for the drone's position and orientation. The software utilizes computer vision techniques to detect and calculate the relative positions of AprilTags, allowing the drone to adjust its velocities to move towards the detected tags. Moreover, the paper addresses the problem of moving the drone through gates autonomously using a similar approach. The software developed for the Tello drone can potentially be extended to more complex drones, making them suitable for diverse real-world applications, including package delivery, search and rescue missions, and disaster management. We include the prototyping of a more capable drone. The project emphasizes affordability, safety, and educational applications, providing a stepping stone for students and enthusiasts to explore the world of autonomous drones.

Summary

This paper introduces an exciting development in the world of autonomous drones – those flying vehicles that can navigate on their own without a pilot. Drones are becoming increasingly popular and have a wide range of potential uses, like delivering packages or helping in rescue missions. The project focuses on the Tello drone, an affordable and versatile drone. We have developed special software using computer vision to make the drone follow markers called "AprilTags", which are printed, pre-positioned visual markers, allowing the drone to navigate autonomously through gates, like an explorer in an obstacle course. The project is educational and safe, encouraging experimentation. In the future, these ideas could be applied to larger drones for exciting applications like package delivery and emergency assistance. We have also started investigating the possibility of implementing all the Tello drone software to a bigger drone that we have assembled.

1 Introduction

Autonomous drones and other autonomous vehicles are a burgeoning field, with many sub-fields yet to be explored [1]. They are a popular and complex challenge, not only because of the required software, but also the hardware to be able to perform the actions fast and accurately.

Autonomous drones normally have the same parts as a normal drone, plus an autopilot: a flight controller that doesn't rely on a pilot. These autopilots can be programmed in order to perform tasks or missions [2].

Safety and perceived safety are two important components that need to be taken into account when designing drones, especially if their possible applications are human related, such as search and rescue missions, package delivery, or disaster management and humanitarian aid. We believe that autonomous drone can be used for the good, and if people perceive them as a dangerous thing, the difficulty of the application process to real missions would increase. Previous work on autonomous drones has been mainly focused on artificial intelligence, machine learning, and neural networks, but much is still underdeveloped. The sensors and computers have gotten small, and being able to reduce the size of the drones opens more possibilities [3].

Though, the economical cost of the components and software development is increasingly becoming more accessible, it can still improve. Toy drones like the Tello drone, from DJI and Ryze [4], are much more accessible, both economically and technically, since it is already prepared to be programmed using one of the most popular coding languages: Python. How to use Tello drones for semi-autonomous missions is discussed in [5]. An image of the Tello drone can be seen in 1

In our paper, we will show the software that we have developed using a Tello drone, which could be applied to real world applications, to bigger and more complex drones, and for education purposes, so that students can start getting into autonomous drones in an approachable way.

Figure 1: Image of the Tello drone

A very clear and direct example is package delivery. Currently, the last mile delivery is done by trucks, cars, or people, but there is research about using drones instead [3]. A common idea is that the main carrier of the packages is a truck (or similar) and in addition to the packages it also transports drones that will perform the last mile delivery. One challenge is the return of the drone to the truck. Once the drone has delivered the package(s) it returns to the truck which is a simple mission taking into account that the drone has GPS connection, but once it is close to the truck it has to land, on a moving target, without good GPS connection due to tall buildings or other interferences which are more common at ground level.

A possible solution would be to install AprilTags in the truck so that the drone can easily detect, follow and land. An AprilTag is a type of visual marker designed to be easily recognized by computer vision systems. It is a 2D barcode-like pattern that consists of a black border with a unique binary pattern inside. Every tag has a different ID. An example of an AprilTag can be seen in 4

This study aims to develop a software that enables the drones to autonomously follow AprilTags, thus enhancing the capabilities of Tello drones and creating new possibilities for the application of these drones in diverse domains.

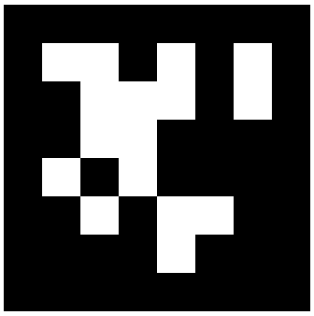

Figure 2: An Apriltag

2 Background

For this paper, we will use the Tello drone as the platform to develop the Apriltag follower software. The Tello drone was developed by the Ryze company, in partnership with DJI and Intel. It is a 80 grams, 25 centimeters (crossed motor distance) drone, with a camera capable of transmitting 720 p real time video. Equipped with propeller guards, ultrasound height sensor and a bottom-facing camera used for precise hovering, Tello is a very safe and versatile drone [4].

The Tello drone can also be programmed using different coding languages: Scratch and other languages through an SDK. We will use Python because it is open source, has an understandable and adaptable syntax, and is widely used.

3 Methods

In this paper, we aim to solve two main problems:

1. **Develop software for the Tello drone to follow an AprilTag autonomously**
2. **Develop software for the Tello drone to move through gates autonomously**

There are different key aspects that we want to this project to follow:

- Affordability: we want this project to be very affordable, so that it can be used for educational purposes.

- Safety: we want the algorithm to be reliable and safe.
- Autonomy: we want the Tello drone to be fully autonomous, with absolutely no need of human control.

3.1 Following AprilTags Autonomously with the Tello drone

One of the aims of this project is to develop a software that permits a Tello drone to follow an AprilTag in an autonomous way.

The first step is to detect the AprilTag, using the AprilTags Python library.

Once we have detected the AprilTag, we calculate the relative position with the drone following these steps:

1. Measure the AprilTag's apparent width (d) in pixels when placed at a known distance Z (e.g., 1 meter = 100 cm) from the camera.
2. Calculate the focal length (f) of the camera using the formula:
$$f = \frac{d \cdot Z}{D}$$
where D represents the physical size of the AprilTag in units (e.g., $D = 0.18$ cm).
3. When the AprilTag's apparent width in subsequent frames is d' pixels, estimate the distance Z' between the camera and the AprilTag using the formula:
$$Z' = \frac{D \cdot f}{d'}$$
where Z' is the current distance from the camera to the AprilTag.

For our Tello drone equipped with a camera, we set $D = 0.18$ cm and placed the AprilTag at a known distance $Z = 0.5$ meters (50 cm) from the camera. The measured apparent width of the AprilTag was $d = 333$ pixels.

Using the calculated focal length $f = 925$, we estimated the distance Z' for subsequent frames. For instance, when d' was measured as 250 pixels, the estimated distance Z' was:
$$Z' = \frac{0.18 \cdot 925}{250} = 2.12 \text{ meters}$$

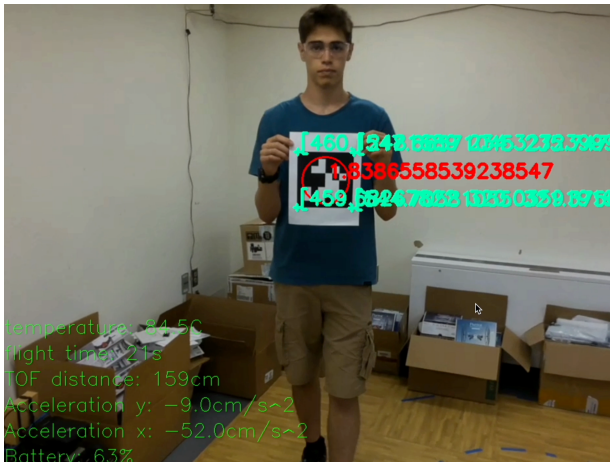

Figure 3: Drone image

This is an image of how the AprilTag is detected 3.

Once the Apriltag is detected and we know the relative position to the drone, the next step is to make the drone move towards it.

In order to do this, we have developed three different approaches. The last one is the one that we use.

The first approach is to send *move* commands to the Tello drone. We send a $move\_forward/backward$ (mainly effects pitch), $move\_right/left$ (mainly effects roll) and $move\_up/down$ (mainly effects throttle).

A problem with that method is that the *move* command is a complex command, that doesn't allow the drone to send or receive data while it is being performed.

The second approach is to send the $go\_to\_relative\_position(x, y, z)$. But the base of the problem remains basically the same. It is not possible to read and analyze the data of the camera while the drone goes to the relative position.

The third approach is the one final solution. It consists of changing the velocities of the drone. The key aspect of this is that changing the velocities is not a blocking command, hence, we can detect and analyze the camera data while the drone is moving.

We do this by using the $drone.set\_rc\_velocities(pitch, throttle, roll, yaw)$. Note that we are using pitch, throttle, and yaw (not roll). A diagram of this third and final solution can

be seen in Figure 4.

The third approach is more reliable, accurate and fast. The first and second use blocking commands, which prevent the flow of information from proceeding and prevent the software from continuing to analyze the images from the camera. Since the AprilTags can be moving targets, the more continuous the data is the better.

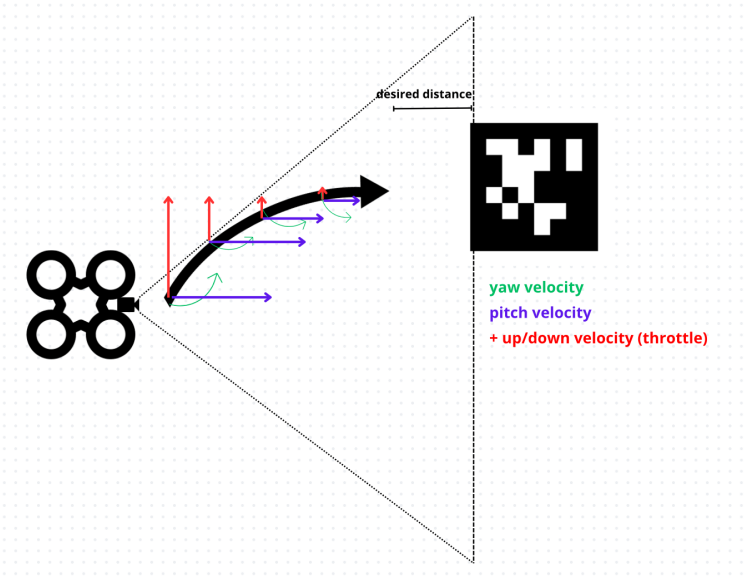

Figure 4: Diagram of the moving solution.

3.2 Moving through gates autonomously

The second objective of this project and paper is to develop a software for the Tello drone to go through gates autonomously.

The gates that we are going to use are squares with 150 cm long sides with a 100 cm side length hole in the middle. They have a total of 8 AprilTags, 4 on the right side and 4 on the left side, each with a different ID, as seen in Figure 5.

To develop the software to go through the gates, we use some algorithms and methods used previously to follow an Apriltag.

The most relevant (in a very high level view) are:

- Detect AprilTags, corner position in the camera field, center, and ID.

- Calculate the relative position from the AprilTags to the drone. The drone only needs to find 5 AprilTags to detect the gate.
- Go to a certain position (relative to the drone and the AprilTags), changing the velocities.

Figure 5: Image of one of the gates used.

Once we have collected the data from the camera, calculated the relative position from each AprilTag, we can calculate where the center of the gate is and approach it.

To use this, we are going to develop and test the code for a specific case, shown in Figure 6.

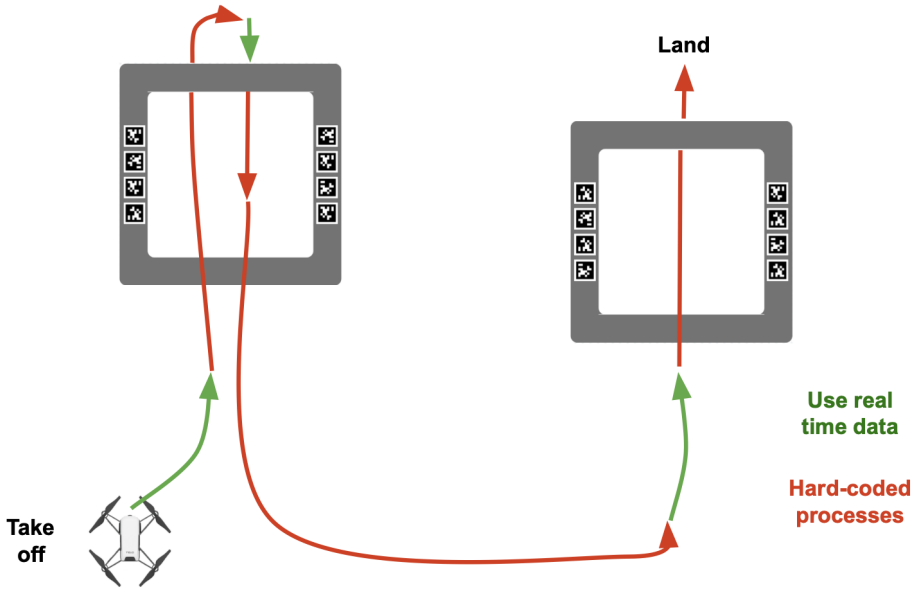

Figure 6: Diagram of the gates.

A very high level flowchart of how the gate's code works can be seen in 7.

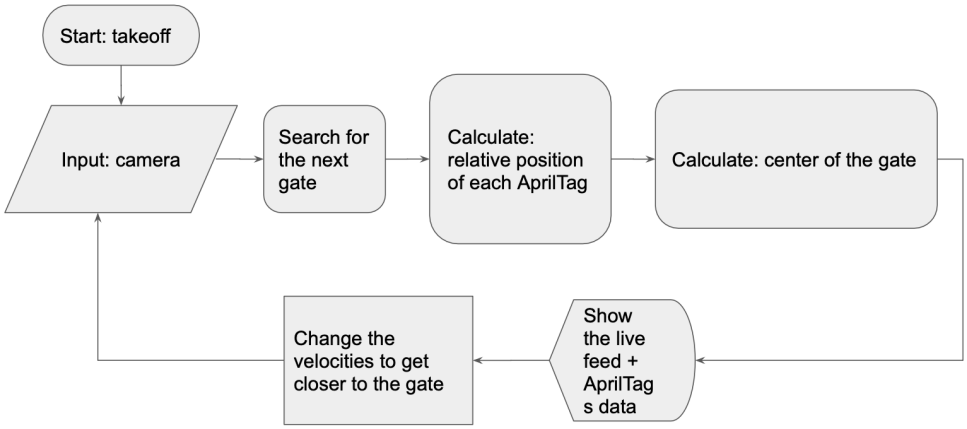

Figure 7: Diagram of the algorithm to move through gates.

3.3 Manual control

To ensure safety, we implemented a manual control system in the two solutions. The drone can be easily controlled from the computer, bypassing all autonomous commands. This feature is designed for emergency situations, but it can be used as a remote controller to learn, disseminate and teach about the project and drones, and have fun.

A detailed explanation about how the manual controller works can be seen in Appendix A.2.1

3.4 Detecting and approaching a person

We have also investigated about using the Tello drone to follow or approach a person. This, with a bigger drone capable of transporting a payload, could be used for search and rescue missions 5.

The software works similarly as the Apriltag follower A, but instead of searching and approaching an AprilTag it does it with a person. We have used MediaPipe [6] to detect a body.

This is an image of how a person is detected 8.

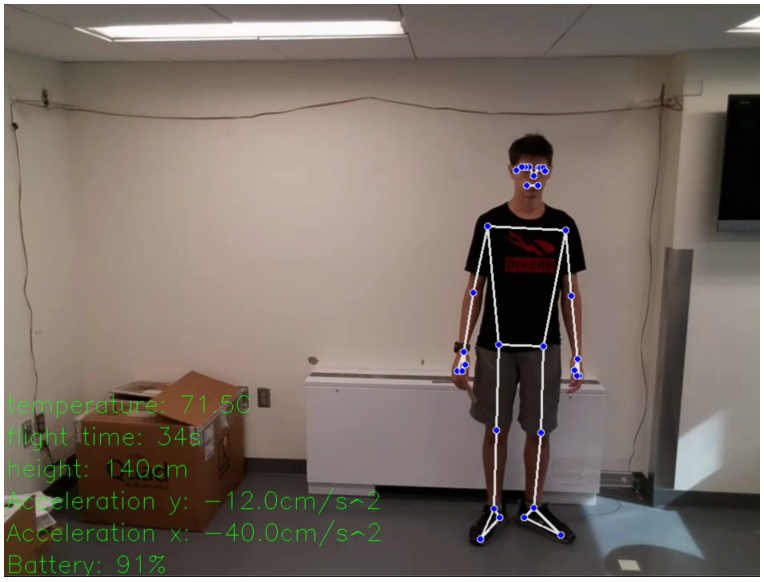

Figure 8: How a person is detected.

4 Applying the problems to a more complex drone

The Tello drone is a great platform, but if we want to apply these ideas and software to real world missions, we need a drone with more potential.

Since most of the applications of this project, as explained in 1, need to carry an additional payload (package delivery, rescue missions), the main aspect to improve the drone's power.

A secondary aspect is the ability to add sensors or functionalities without great difficulty, so that the project can evolve and gain both complexity and possible applications in the future.

In order to build a more powerful and extensible to future possibilities drone, we select the hardware components that we need. In Figure 9, the main generic components that drones have are shown.

The components that we are going to use to assemble the drone can be seen in 1. In addition to the typical components we are including a Raspberry Pi, that will be used as a companion computer with the flight controller, to be able to make the drone autonomous.

We propose to make the Raspberry Pi act as a receiver, and send simple commands (throttle, pitch, roll, yaw) to the flight controller.

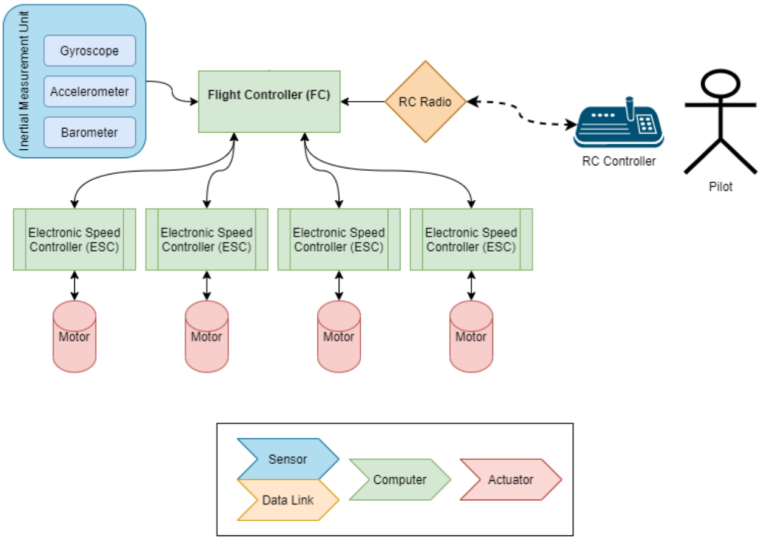

Figure 9: Block diagram of basic drone components [7].

This has not been done before, and it would eliminate the need for a very expensive and complex autopilot.

We propose to have a Raspberry Pi zero Wireless, connected to a camera, use a similar code to the one that we use for the Tello drone (see 3.1) to control the velocities of the drone using the input from the camera. With AprilTags, we would be able to perform autonomous missions just like the Tello drone, but with a greater capacity to carry packages and to add new functionalities or sensors, making the system even more robust and reliable.

This software has not been yet fully developed, but we will update it and implement it in the future. An image of the prototype can be seen in 10.

We have started developing the software. Currently, the drone is able to detect the AprilTags and calculate the relative position and the distance between itself and the AprilTag. The next step would be to send commands from the Raspberry Pi the flight controller. A possibility would be to use the PiSBUS library [8].

Component list	
Component name	**Characteristics**
Cinewhoop 3” Frame	Carbon Fiber frame and propeller guards, which take an important role in terms of safety.
T-Motor Mini F7 3-6S 20·20 Stack/Combo	Flight controller and ESC (Electronic Speed Controller): the main 2 components to control the drone.
1507 3100KV Motor	Brushless motors, connected to the ESC, they create the thrust using the propellers.
3 inch propellers	To create the thrust
DJI FPV System	RC control receiver, camera and video transmitter.
Raspberry Pi zero W	Used as a companion computer, to make the drone autonomous.
RPi Camera 3	Raspberry Pi Camera module 3, used to retrieve real time data.
4s 1300 mAh Li-Po battery	Battery that supplies energy to all the components of the drone.

Table 1: Components used for the drone prototype.

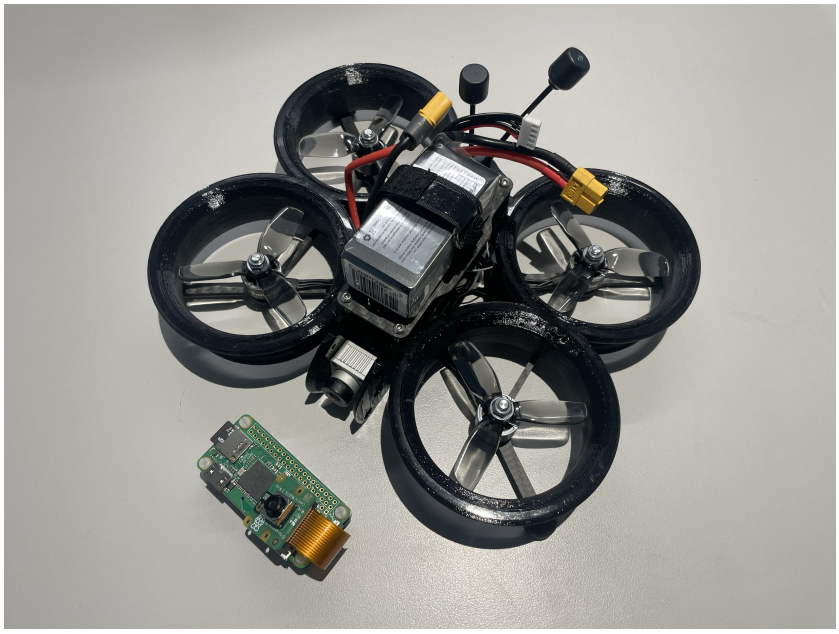

Figure 10: Image of the drone prototype

5 Discussion

Autonomous drones are a great tool, they can do amazing things. A (piloted) drone performed the first sea rescue in 2018 [9] and saved the lives of two teenagers. Drones have been used for firefighting purposes [10], and they can be used for search and rescue missions.

There are also drones for space exploration like the Mars Helicopter [11] or the future mission Dragonfly, an autonomous drone to investigate Titan [12]. Space exploration using autonomous drones can help us answer some of the biggest questions and mysteries of the universe, like: What the origins of life? Is there life somewhere else?

But we have to keep in mind that some people might want to use autonomous drones for other purposes, like military or surveillance. To kill people, to end lives.

The scope of this paper is to contribute to the development of science, technology, and engineering. We believe that these fields, especially autonomous drones, should be used to improve our lives, the lives of those who are in need, and our planet: our home.

6 Conclusion

In this paper, we presented a software solution for the Tello drone to autonomously follow AprilTags and navigate through gates. The implementation relies on computer vision techniques to detect the tags and calculate their relative positions to the drone. By adjusting the drone's velocities based on this information, we achieved successful autonomous navigation towards the tags and through the gates.

The project's focus on affordability, safety, and educational applications makes it suitable for students and enthusiasts interested in exploring autonomous drone technology. The ability to manually control the drone ensures safety and can be used for educational purposes and in emergencies.

With the potential to apply these concepts to more complex drones, such as those used in package delivery or search and rescue missions, the software developed in this project opens

up new possibilities for the application of autonomous drones in diverse domains.

Overall, this paper contributes to the advancement of autonomous drone technology and lays the foundation for further research and development in the field of autonomous aerial systems.

7 Acknowledgments

I would like to thank Professor Chuchu Fan, for mentoring me during this project, Charles Dawson and Kunal Garg, for helping and guiding me, and the REALM department from MIT AeroAstro. This has been a great experience thanks to the great team I've had the privilege to work with. I would also like to especially thank Agnes Robang, for tutoring me, Claire Wang for being the best counselor, and all my fellow rickoids.

I would also want to thank my family for supporting me always: especially Mònica, Lluís, Olau and Tell, without them this project could no have been possible.

I want to thank RSI, MIT and CEE, and Joves i Ciència for giving me this great opportunity.

References

[1] S. M. R. Islam. Drones on the rise: Exploring the current and future potential of uavs, 2023. 2304.13702.

[2] N. Wang, O. Catal, T. Verbelen, et al. Towards bio-inspired unsupervised representation learning for indoor aerial navigation, 2021. 2106.09326.

[3] J. Saunders, S. Saeedi, and W. Li. Autonomous aerial delivery vehicles, a survey of techniques on how aerial package delivery is achieved, 2022. 2110.02429.

[4] Ryze-Robotics. Feel the fun, 2020. URL https://www.ryzerobotics.com/tello.

[5] K. Hulek, M. Pawlicki, A. Ostrowski, et al. Implementation and analysis of ryze tello drone vision-based positioning using apriltags, 2023. 2305.05673.

[6] MediaPipe — Google for Developers — developers.google.com. https://developers.google.com/mediapipe, 2023. [Accessed 01-08-2023].

[7] B. Flight. Basic drone assembly, 2022. URL https://bellflight.github.io/AVR-2022/basic-drone-assembly/.

[8] GitHub - 1arthur1/PiSBUS: Raspberry Pi SBUS library — github.com. https://github.com/1arthur1/PiSBUS.git, 2018. [Accessed 01-08-2023].

[9] H. Brady. Drone saves drowning swimmers in australia for first time ever. *Natl. Geogr. Mag.*, 2018.

[10] T. Yang, S. Zhang, Y. Wang, et al. Optimized deployment of unmanned aerial vehicles for wildfire detection and monitoring, 2021. 2112.03010.

[11] mars.nasa.gov. Mars Helicopter - NASA — mars.nasa.gov. https://mars.nasa.gov/technology/helicopter/#Tech-Specs, 2021. [Accessed 01-08-2023].

[12] NASA's Dragonfly Mission to Titan Will Look for Origins, Signs of Life — nasa.gov. https://www.nasa.gov/press-release/nasas-dragonfly-will-fly-around-titan-looking-for-origins-signs-of-life, 2019. [Accessed 01-08-2023].

[13] V. i R. Giol. GitHub - Giol-Vinyals-i-Roca/RSI/2023 — github.com. https://github.com/Giol-Vinyals-i-Roca/RSI_2023.git, 2023. [Accessed 01-08-2023].

Appendix

A Tello autonomous AprilTag following Documentation

A.1 Introduction

This documentation explains the code for a Tello drone that can follow AprilTags using computer vision. The code is written in Python and uses the `cv2` (OpenCV), `apriltag`, `pygame`, `numpy`, and `djitellopy` libraries.

A.2 How it Works

The code allows you to manually control the Tello drone using the arrow keys and WASD on your keyboard. When the Tello drone detects an AprilTag in the camera feed, it will display the center, distance, and position of the detected AprilTag on the screen. The Tello drone will then move automatically to follow the AprilTags by changing its velocities.

A.2.1 Manual Control

To control the Tello drone manually, you can use the arrows and WASD on the keyboard of the computer the drone is connected with.

What does each key do:

- T: Take off
- L: Land
- esc: close the program and land
- Up arrow: makes the drone go forward (pitch)
- Down arrow: move backwards (pitch)
- Right arrow: move right (roll)

- Left arrow: move left (roll)
- W: move up (throttle)
- S: move down (throttle)
- D: rotate clockwise (yaw)
- A: rotate counterclockwise (yaw)

Note: Be aware that the take-off and land commands are not simple commands, hence the video feed might cut off during these operations. It is not recommended to send other commands while these are running, since it might cause lag and it could ultimately make the drone land or lose control.

A.3 Prerequisites

Before running the code, make sure you have the following libraries installed. The versions and other information can be found in [13].

- Djitellopy (`djitellopy`)
- OpenCV (`cv2`)
- AprilTag (`apriltag`)
- Pygame (`pygame`)
- Numpy (`np`)
- Time (`time`)
- AV (`av`)
- Math (`math`)

A.4 Code Structure

The code consists of the following sections:

1. Importing Libraries
2. Global Variables
3. Utility Functions
4. Main Class and Functions
5. The Main Function

A.5 Sections Explanation

A.5.1 Importing Libraries

The code starts by importing all the required libraries, including `cv2`, `apriltag`, `pygame`, `numpy`, and `djitellopy`.

A.5.2 Global Variables

Several global variables are defined to control the Tello drone's movement and various visual aspects. These include `S` (speed), `FPS` (frames per second), `LINE_LENGTH`, `CENTER_COLOR`, and `CORNER_COLOR`, among others.

A.5.3 Utility Functions

The code defines some utility functions to simplify drawing on the camera feed. These functions include `plotPoint`, `plotCircle`, and `plotText`.

A.5.4 Main Class and Functions

The main class is named `FrontEnd`, which handles the interaction with the Tello drone and the display using Pygame. The class contains the `__init__`, `run`, `keydown`, `keyup`, and `update` functions.

- `__init__`: Initializes the Pygame window, sets up the Tello drone connection, and creates an update timer.
- `run`: The main loop that continuously captures frames from the Tello drone's camera, processes them to detect AprilTags, and moves the drone accordingly to follow the detected tags.
- `keydown` and `keyup`: These functions handle the keyboard events and update the drone's velocities accordingly.
- `setvelocity`: Calculates the Tello drone's velocities based on the detected AprilTag's center and distance.
- `update`: Sends the Tello drone's velocities to the drone to control its movement.

A.5.5 The Main Function

The main function creates an instance of the `FrontEnd` class and calls the `run` method to start the drone's interaction and AprilTag following.

A.6 How to Use

1. Make sure you have installed all the required libraries.
2. Connect your computer to the Tello drone's Wi-Fi network.
3. Run the Python script.
4. The Pygame window will open, showing the Tello drone's camera feed.

5. Use the arrow keys to manually control the drone's movement.

6. When the drone detects an AprilTag, it will display its center, distance, and position on the screen.

7. The drone will then automatically follow the detected AprilTag.

A.7 Note

- Ensure that you have a clear and safe environment for the Tello drone to follow the AprilTags without any obstacles or hazards.

- Pay attention to the drone's battery level to prevent unexpected landings: when the battery level is below 15% the drone might automatically land.

- The drone will follow the AprilTags as long as it can detect them within the camera's field of view.

- Always operate the drone responsibly and follow local laws and regulations regarding drone usage and privacy.

B Tello autonomous gates Documentation

All the procedures are the same as with A. The manual control work in the same way too.

B.1 Code Overview

The code uses the Python programming language and several libraries to interact with the Tello drone, perform image processing, and detect AprilTag gates. Here's an overview of the main components:

1. Import Necessary Packages: Import the required libraries (the versions and other information can be found in [13]), including OpenCV (`cv2`), NumPy (`numpy`), AprilTag (`apriltag`), math, time, Pygame (`pygame`), and the Tello drone library `djitellopy`.

2. Declare Variables: Set global variables and constants used throughout the code.

3. Initialize Drone: Create a function `initialize_drone()` to establish a connection with the Tello drone and set initial settings.

4. Initialize Screen: Create a function `initialize_screen()` to initialize the Pygame window for visualization.

5. Initialize Detector: Create a function `initialize_detector()` to create an AprilTag detector object.

6. Get Drone Frame: Create a function `get_drone_frame(drone)` to retrieve the current frame from the drone's camera.

7. Get Tag Corners: Create a function `get_tag_corners(detector, gray_image)` to detect AprilTag corners and return the corners and detection results.

8. Draw Corners: Create a function `draw_corners(image, corner)` to draw lines at the corners of detected AprilTags.

9. Draw Center: Create a function `draw_center(image, radius, center)` to draw a circle at the center of detected AprilTags.

10. Plot Text: Create a function `plot_text(image, text, center)` to add text to the screen.

11. Calculate Centers and Distances: Create functions `calculate_centers_mean_gate(center, ID, id_max, id_min)` and `calculate_distances_mean_gate(distance, ID, id_max, id_min)` to calculate the mean centers and distances for specific gates. This center point is where the drone will go.

12. Set Velocity: Create a function `set_velocity(mean, d_mean)` to calculate the velocity based on the desired distance from the AprilTag center and the position of the gate.

13. Search (Not Used): Create a function `search()` that can be used to search for AprilTags if starting conditions are more flexible. This function is not used for our tests.

14. Keydown and Keyup Functions: Implement functions `keydown(key)` and `keyup(key)` to handle keyboard input for drone control.

15. Update Function: Create an `update()` function that sends the drone's velocity when `send_rc_control` is `True`.

16. Main Function: The main code starts execution in the `__main__` block.

17. AprilTag Detection and Drone Movement: The code continuously detects AprilTags, calculates velocity, and moves the drone accordingly to navigate through the gates.

18. Data Display: The code displays information about the drone's status, such as battery level, acceleration, distance, flight time, and temperature.

19. Gate Sequence: The code navigates the drone through three different gates.

20. Land: Finally, the code lands the drone after navigating through all gates.

B.2 Running the Code

To execute the code, ensure that the Tello drone is connected to your computer. Then, simply run the code, with the required libraries installed. The drone will take off, navigate through the specified gates, and eventually land. This code is specific to our setup of the gates, shown in 6.

C Code

All the final versions of code used for this project can be found in [13].