User safety: Stabilizing an Aggressively Thrown Drone

Jun Xian Chen
University of Southern California
jxchen@usc.edu

1 Introduction/Purpose

A holodeck is an immersive 3D representation of real and virtual objects. Drones have been gaining popularity in the field of Holodecks. Drone sizes range from millimeters to meters. Given drones' compact size, a soft throw can easily damage them. When a drone suddenly becomes uncontrollable, the drone can injure a human. The project focuses on how to recalibrate a drone when it is aggressively thrown without injuring the people around it and the drone itself.

2 Related Work

With the development of drones, there have been issues with drones losing control. Drones have been shown to demonstrate amazing aerobatics in maneuvering challenging environments (Abbeel et al., 2010). The paper used apprenticeship learning algorithms: An inverse reinforcement learning approach by generating a reward function based on the observed. Then the drone performs the aerobatics; the drone is penalized if deviating from the trajectory. The results show high performance for autonomous helicopter aerobatics.

There has been research on drones stabilizing themselves after losing control (Faessler et al., 2015). Given the high-performance acrobatics, stabilizing a drone using acrobatics has become feasible. The paper discusses a method for drones to stabilize themselves with a monocular vision-based quadrotor after being aggressively thrown. The paper showed that drones could recover and re-initialize their vision-based state estimation pipeline from any attitude.

An application of drone for holodeck is in the Flying LEGO Bricks experience hosted by LEGO (Rubens et al., 2020). Due to safety requirements, the drones were in a closed-off area. First, the children designed a butterfly controller by placing five LEGO bricks on the left and right wings. Then the controller is scanned to animate the drones in the controllers' positions. Lastly, the children could maneuver the drones by rotating and moving the controller.

3 Setup

The drone bought is a DJI Tello shown in Figure 1. The interface with the drone is by using the DJI Tello APP on the iPhone to configure the IMU and PyCharm to configure the movements through WiFi. To connect to the iPhone, the mobile data must be turned off; to connect to the laptop, the public firewall must be turned off. Then remove the propeller and propeller shield to configure the IMU. Open the iPhone app, click on the gear, go to more, click the triple period, and click the calibrate on the IMU status. Follow the instructions to place the drone in the proper orientation for configuration. Lastly, connect the drone to the laptop for the proper application. For Python, there is a DJITelloPy api to interact with the DJI Tello directly. The library includes movements to tell how the drone moves and sensors to detect the speed and acceleration of the drone. The environment for the drone is an outdoor field. An indoor field was used before; however, the hover can sometimes cause it to keep increasing height and hit the ceiling.

4 Milestones

1: 10/3 - 10/4: Download Airsim. Get a programmable drone and set up the programming on my desktop.

2: 10/4 - 10/14: Start the drone simulation to emulate being thrown. Figure out the appropriate deceleration.

3: 10/14 - 10/17: Test the speed and rotation

Figure 1: Tello

level of the drone. The data will be used for the recalibration after being thrown.

4: 10/17 - 10/24: Assume the drone is grabbed and figure out the thrown detection for the drone. Find the minimum velocity change threshold.

5: 10/24 - 11-7: Program the PID controller to respond appropriately (slow down) when the drone is aggressively thrown.

6: 10/31 - 11/7: Test the drone reactions with motors on against a basketball and baseball throw.

7: 11/7 - 11/16: Get the drone to fly away from a human while simultaneously readjusting itself.

8: 11-16 - 11/30: Prepare the presentation and generate the report.

5 Design Rationale

For a full holodeck experience, objects need to be thrown. When an object is thrown near people, people could get hurt. In addition, the object should not be damaged. Suddenly stopping a drone at high speed could be potentially damaging to the drone. The idea would be to slow the drone down at an appropriate deceleration while avoiding the objects.

6 Problems

One of the milestones was to simulate an aggressively thrown drone; however, due to my laptop's limitations, Airsum could not properly run on it. Whenever it renders a location, constant static overwhelms the screen.

Given the versatility and a Python library, the DJI Tello seemed like a promising drone to work on this project. However, I could not afford to get another drone at this point. Speed detection only works on the z-axis. The acceleration is dependent on the position as well as movements. If the drone were held still at two different positions, the acceleration could be drastically different. To combat the former, human avoidance would be focused only on the z-axis; to combat the latter, the current acceleration is subtracted by the weighted average of the previous acceleration is used. Once moved, it would be best to wait a few seconds to let the drone acceleration settle down before throwing.

Another problem is the drone can only be flown when the roll and pitch are zero. This affects the basketball throw as the roll and pitch are changed as it is thrown at 45 degrees instead of 90 degrees. In addition, the orientation of the roll and pitch are determined by the startup. If I were to rotate the roll by 90 degrees, start it up, which would be the default position. Another problem would be there is no set speed in a direction, and the position movement has a minimum requirement making the PID control difficult.

7 Implementation

PID (proportional–integral–derivative is used to stabilize the drones. The proportional control controls the deceleration due to the current speed. The integration control controls the constant speed for the drone to hover. The derivative control controls the overshoot to prevent the drone from overreaching the equilibrium point.

Since acceleration detection is not accurate in the respective direction, it is accurate in a sense it detects acceleration. In the initial position, the drone is still; however, the initial position can change. The change in initial position changes the acceleration even though it is still. To combat the issue, the average weighted mean with 60% on the detected acceleration and 40% on the previous acceleration. The average weighted mean is used as the starting point for the acceleration, i.e., the detected acceleration is detected acceleration subtracted by the average weighted mean. If the sum in all directions is above 948 cm/s^2 with motors off and 200 cm/s^2 with motors off, then thrown is detected. This ensures the drone does not detect acceleration

while moving to a different initial position.

The baseball throw is mainly horizontal movement in the x and y direction. It is the only application with the motors on that works. In addition, to remove a layer of randomness, only the x-y acceleration is used in the detection. The starting position is shown in figure 2. The baseball throw is successful if my throw is mainly in the x and y direction; however, my throw is not perfect, and sometimes the drone tilts at an angle that turns off the motor. Before throwing the basketball throw, simply angling the motor in the position of a basketball throw turns off the engine. During a basketball throw, the roll and pitch are constantly changing. In addition, the takeoff command terminates the program if it is unsuccessful. When the drone detects being thrown, the roll and pitch are near zero at one point then the drone initiates the takeoff for the basketball throw. That is mainly unsuccessful; as a result, imitating a basketball throw with the drone in my hands instead of throwing it helps initiate the takeoff. The starting position is shown in figure 3.

For human detection, only the z-axis can be used. The drone can determine its altitude from the relative ground. Instead of throwing the drone at the human, the drone is slowly dropped onto the ground. It is similar if the drone were to be dropped on the human. The drone detects the relative ground to itself. If the altitude reaches below 40 cm, a human is detected. The starting position is shown in figure 3.

Figure 2: Baseball Starting Position

Figure 3: Basketball and Human Starting Position

8 Milestones Analysis

The first two milestones: could not be completed due to hardware requirements on my laptop.

The third milestone: the top speed is a shade over 8m/s; the rotation is almost instant when the command is given.

The fourth milestone: the acceleration detection part of the drone's kit. However, it has problems, as mentioned in the problem section. The solution is to use the average weighted mean to get a baseline for the acceleration.

The fifth milestone: the PID control, is programmed in the PyCharm file using a double array of size 3x3. The first row is the x, y, and z-axis, respectively. The second row is the position, integration, and derivative, respectively. Since the drone's speed detection does not work well and is already programmed in the code, The PID control could be used for another drone.

The sixth milestone: having the motors on while grabbing the drone would cause the drone to shut down; as a result, the initial position is the drone being held without motors on. The recovery of the baseball does not work well because the drone would not be in a position eligible to fly; however, the recovery of the basketball throw works most of the time.

The seventh milestone: completed when the PDF file is submitted online.

9 Discussion

The DJI Tello is not an appropriate drone for this project. There are many flaws, and the speed and acceleration detection are faulty. In addition, the takeoff command terminates the program after four unsuccessful tries instead of returning a value. The PID control is made for a more general setting; as a result, any drone could use the PID control with the appropriate command calls and constant values.
The drone successfully does a baseball; however, it has to stabilize itself before being able to receive additional commands. This project is for user safety, and I do not recommend using a DJI Tello on a throw-based trajectory. The drone does not react fast enough to prevent a collision.

10 Conclusion

Unfortunately, my laptop could not support AirSum. Given the problems and situation with the drone, there are many compromises and simplifications in the project. The PID configuration is in the Python code uploaded on Github, and the video demonstration is available on the link provided on GitHub.

Github link: https://github.com/Juneyyyy/CSCI599

References

Pieter Abbeel, Adam Coates, and Andrew Y. Ng. 2010. Autonomous helicopter aerobatics through apprenticeship learning. volume DOI: 10.1177/0278364910371999.

Matthias Faessler, Flavio Fontana, Christian Forster, and Davide Scaramuzza. 2015. Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor. volume ICRA 2015.

Calvin Rubens, Sean Braley, Julie Torpegaard, Nicklas Lind, Roel Vertegaal, and Timothy Merritt. 2020. Flying lego bricks observations of children constructing and playing with programmable matter.