Obstacle Detection Via IMU Data Classification In Autonomous Quadcopters

J. Hughes, D. Lyons
Department of Computer and Info. Science
Fordham University
Bronx, New York, USA
jhughes50@fordham.edu

***Abstract*—An autonomous drone flying near obstacles needs to be able to detect and avoid the obstacles or it will collide with them. In prior work, drones can detect and avoid walls using data from camera, ultrasonic or laser sensors mounted either on the drone or in the environment. It is not always possible to instrument the environment, and sensors added to the drone consume payload and power - both of which are constrained for drones.**

This paper studies how data mining classification techniques can be used to predict where an obstacle is in relation to the drone based only on monitoring air-disturbance. We modeled the airflow of the rotors both physically and mathematically to deduce higher level features for classification. Data was collected from the drone's IMU while it was flying with a wall to its direct left, front and right, as well as with no walls present. In total 18 higher level features were produced from the raw data. We used an 80%, 20% train-test scheme with the RandomForest (RF), K-Nearest Neighbor (KNN) and GradientBoosting (GB) classifiers. Our results show that with the RF classifier and with 90% accuracy it can predict which direction a will is in relation to the drone.

***Index Terms*—Collision Avoidance, Autonomous Vehicle Navigation, Classification, Air-Disturbance**

I. Introduction

State of the art autonomous flying drones can fly to a preprogrammed destination while avoiding objects with no input from a user [13]. Such approaches leverage drone or environment-mounted camera, laser, or ultrasonic sensors to detect potential collisions. However, one critical barrier to more widespread drone usage in industry is battery life [13]. Because the drones must be light to fly they must have small batteries, and thus they have a short flight time. The previously mentioned sensors for object avoidance cut down on the drone's battery life substantially. In our prior work we showed wind currents can be detected using a classifier [1,2]. The wind induces greater pose disturbance, and this can be detected from the data gathered by the drone's internal sensors. Additionally, the drone experiences greater pose disturbances when it is close to large objects like walls. This is because backwash from the wall interferes with the airflow created by the rotors of the quadcopter. This interference causes a similar effect to that of a wind current.

The idea behind this paper is that the backwash wind current created by a quadcopter being near an object interferes with the pose of the drone which can be used to detect which side of the drone the interfering object is located. This eliminates the need for the camera, laser or ultrasonic sensors which thus increases the battery life. In a typical RC teleoperated drone scenario, the onboard computer would have to work no harder, since the data points could be gathered and sent to the remote computer. The base computer would then do the calculation and prediction which can then be sent to the onboard driver program to trigger the drone to avoid an object.

As a step towards more general collision detection, this research address the interaction of the drone with perpendicular walls because they provide a good surface for the air from the backwash from the quadcopter's rotors. Our initial focus on walls also reflects the fact that it is likely that drones will encounter these when they fly autonomously in buildings. The quadcopter was flown perpendicular to the walls to collect clear data.

The drone used in this research is the crazyflie 2.0, a lightweight drone ideal for indoor use and with a straightforward programming interface [7]. The quadcopter has an inertial measurement unit sensor (IMU). This measures the gyroscope in the (x, y, z) planes, acceleration with an accelerometer again in the (x, y, z) planes. From the gyroscope and accelerometer roll, pitch and yaw angles can be calculated for recording. There is also a barometer to measure air pressure. On the bottom of the quadcopter is a Flow Deck that measures the drone's z position. The drone also has an Loco Positioning System (LPS) Deck that measures its (x, y, z) cartesian position. While the Flow Deck and LPS are crucial for the data collection and data mining experiments, they are not used at all in the proposed collision detection.

We first modeled the drone's airflow both physically and mathematically to help us generate more meaningful features for classification. To gather the data the drone was flown along a wall to its left, right and front as well as with no wall present. After collecting the data we generated features based on the modeling conducted earlier. Multiple tests were then run with the data. First, the presence of a wall was tested, meaning could the classifier detect a difference between the left and right data and the no wall data. Next, the classifier was used to detect left versus right wall and left versus right versus front wall and finally left versus right versus front versus no

wall. These tests were repeated on three different classifiers: RandomForest, GradientBoosting and NearestNeighbor.

II. Literature Review

There have been prior studies done on the ground effect in quadcopters. A ground effect occurs when the drone is flying near the ground and the air that is being pushed down from the rotors hits the ground and backwashes up on the drone, causing instability [17]. This has been studied extensively to make quadcopter landing and takeoff more stable. Drone to drone wind detection was studied in [2]. This project had one drone fly underneath the other and could successfully detect that there was a drone over top using only the internal sensors of the bottom quadcopter and a classifier. Additionally, the researchers in [1] were able to detect wind gusts using only the internal sensors of the drone. They had the drone fly along a path that encountered an air current from a fan and built a classifier to determine if the drone was in a wind gust or not.

These two studies show that the internal sensors can be used to detect the presence of wind disturbances. Specifically, [2] shows that drones can detect wind currents created by rotors. These papers differ from this project in the respect that they do not predict from which direction the wind is coming. This is pertinent to building a truly autonomous drone that does not rely on outside sensors. Its important to note that in our initial research we tried a purely bottom-up data mining approach, testing on only the raw data from the IMU. We found this did not work: that it was necessary to generate higher level, more descriptive features, and giving rise to the research presented in this paper.

Yoon et. al. and Diaz et. al. from NASA's Ames Research Center used Computational Fluid Dynamics (CFD) to model the airflow of a drone's rotors in [5,6]. They show that the velocity of the air underneath the rotor takes the form of a cylinder. The air moves faster within the cylinder in the shape of an hourglass. This is important because it is our hypothesis that when a drone is near a wall the air underneath it's rotors is getting pushed outward and then interacting with the wall, causing pose disturbances. Their work was done using the DJI Phantom 3 which is large and powerful. Thai et. al. [4] also did CFD for the DJI Phantom 3. Unlike the others, their work shows one instance of the drone's propeller spinning and shows that the velocity has a helical flow pattern to it. They also show that the airflow from the rotors has the hourglass shape.

Other researchers at McGill University have worked on wall detection without using additional sensors in [3]. This study has drones with rotor guards fly directly at the wall. They are looking at what speed can the quadcopter travel at, hit the wall and still recover and not crash from hitting the wall. The idea behind this study is to have an autonomously flying drone that hits walls, recovers and moves in a different direction after the collision. This project wants to avoid the collision and just use the air disturbance from the wall to fly autonomously. Also, not all drones fly with rotors guards, and without them the drone would just crash.

III. Modeling

Physical and mathematical modeling of the airflow of the rotors was done to help better understand the effect the nearby wall had on the drone which would help to generate higher level features. The mathematical modeling specifically helped to create a feature that calculates the proposed between the wall and the drone based on the quadcopter's roll and pitch angles.

A. Mathematical Modeling

Mathematical modeling was done to simulate the airflow under the rotors of the drone because we are interested in seeing how to air from the rotor is effect by the wall and how that in turn effects the drone's pose stability. When watching the drone fly near the wall pose disturbances are clearly visible. We wanted to provide mathematical evidence for this effect.

Deters et. al. [11] used three dimensional Navier-Stokes equations to look at a rotor wake vortex. In forward flight they calculated a spiral vortical wake geometry. The three-dimensional Navier-Stokes equation for helical flow from Ershkov [9] was solved. We have the following variables:

- $\vec{u}$: air velocity vector containing the radial, tangential and downward velocities
- $\vec{w}$: a time-dependent field equaling $\alpha \cdot \vec{u}$
- ∇p: change in pressure from above the rotor to underneath the rotor
- ρ: air density coefficient
- ν: kinematic air viscosity
- $\vec{F}$: force vector in x, y, z directions
- $-\nabla\phi$: force potential from $\vec{F}$

The initial Navier-Stokes equation is as followed:

$$\begin{aligned} \nabla \cdot \vec{u} &= 0, \\ \frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} &= -\frac{\nabla p}{\rho} + \nu \cdot \nabla^2 \vec{u} + \vec{F} \end{aligned} \tag{1}$$

The equation was transformed to include the curl expression represented by $\vec{w}$ in the following equations.

$$\begin{aligned} \nabla \cdot \vec{u} &= 0, \\ \frac{\partial \vec{u}}{\partial t} &= \vec{u} \times \vec{w} + \nu \cdot \nabla^2 \vec{u} - (\frac{1}{2}\nabla(\vec{u}^2) + \frac{\nabla p}{\rho} + \nabla\phi) \end{aligned} \tag{2}$$

By expanding the curl expression a system of partial differential equations was created and then solved. Ershkov explains that the curl field arises from the source of vorticity in the fluid field, which in this case would be the rotor of the drone. $-\frac{\nabla p}{\rho} + \nabla\phi$ is the x, y, z force vector. The system of equations gives the solution as:

$$\vec{u} = \exp\left(-\nu \cdot \alpha^2 \cdot t\right) \cdot \vec{u}(t_0). \tag{3}$$

where $\vec{u}(t_0)$ refers the velocities at the rotor which can be shown as

$$\vec{u}(t_0) = \begin{bmatrix} u_r(t_0) \\ u_t(t_0) \\ u_z(t_0) \end{bmatrix} \tag{4}$$

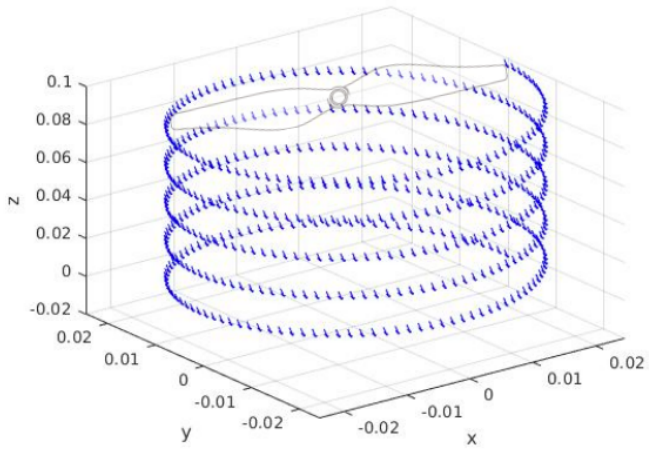


Figure 1: Velocity Vector Field

u_r, u_t, u_z refers to the radial, tangential, and downward velocities at the rotor of the quadcopter. The velocities at time 0 can be calculated with the equations from [11,12]:

$$u_r(t_0) = b \cdot \pi n \cdot 2r, \quad u_t(t_0) = r \cdot \omega, \quad u_z(t_0) = \sqrt{\frac{T/A}{2\rho}} \tag{5}$$

where b is a blade swirling factor, n is the rpm, ω is the induced velocity which is equal to u_z, T is the force of thrust, and A is the area of the rotor disk. The answers in (5) are then used in (3) to solve the equation as time continues, the resulting vectors for u_r, u_t and u_z were plotted as a vector field. From the vector field we can see that the air has some outward velocity which means the air will move towards the wall and create a backwash, causing a disturbance in the drone's pose. We expect the air to follow a similar shape and to bounce off the wall when we do physical modeling of the airflow.

B. Airflow Testing

In order to physically examine the airflow of the rotors, a drone was fixed to a spindle and placed against a black background. The rotors were engaged at various thrust levels. Talcum powder was dropped from above into the drone's rotors. Images were taken of the resulting powder motion, and this was used to reveal airflow interaction with a vertical surface brought into proximity of the drone. The powder was dropped into the rotors from 0.5 meters above. The rotors were tested at varied thrust starting at 25% and working up to 80%. Figure 2 shows the experimental setup of the drone. From the photos we can see that particles of talcum powder move as predicted in a helical pattern. We also see that there is a higher concentration of powder following an hourglass shape underneath the rotor shown best in figure 3c. This indicates an area of higher pressure. The mathematical model only shows the velocity of the air before it hits the wall. Once the air hits the wall the area of high pressure is created around the contact site.

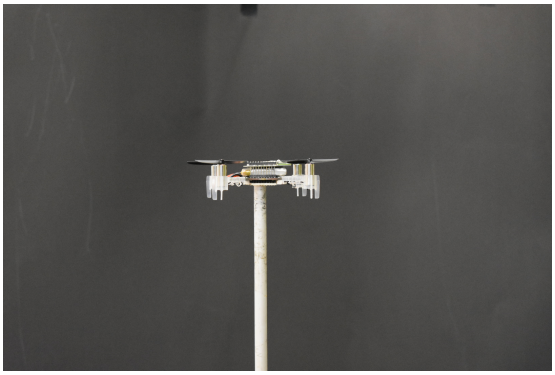


Figure 2: Experimental setup

The process was repeated but with a wall near the drone to examine how the airflow from the quadcopter interacts with the wall. The drone was placed two rotor diameter-lengths away from the wall, which is about 0.1m.

In Figure 4 we can see the hypothesized backwash effect from the wall indicated by the more dispersed powder further underneath the wall. The larger cloud of powder can interrupt the air flowing downward, and we propose that this causes an 'artificial' ground effect. The lift is compromised on one side of the drone, resulting in larger than nominal pose disturbances. We hypothesize that it should be possible to tell which side of the drone this occurs on by monitoring some higher-level features of the platform pose angles.

C. Feature Generation From Modeling

From the modeling it was deduced that the gyroscope would be the most effected IMU sensor on the drone. When we look at the data we can see that the pose disturbance is shown more in the gyroscope than in the accelerometer.

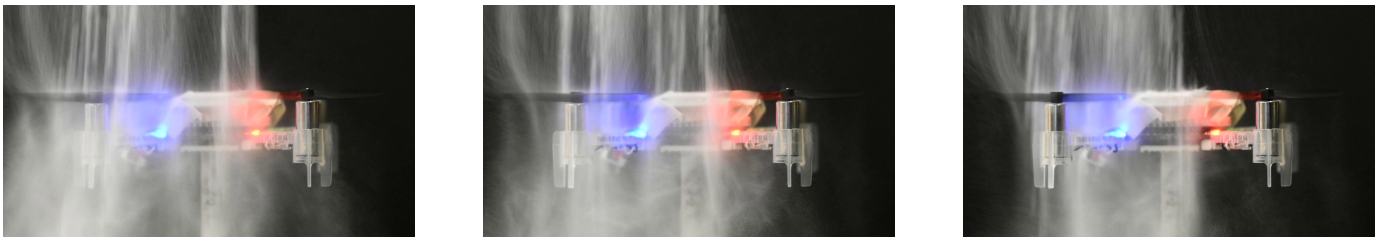
This in turn means the roll and pitch angle would also be effected since they are calculated from the gyroscope. It is predicted based on our modeling and experimental results that the drone's roll angle would be more affected from the wall being to the left or right and the drone's pitch angle would be more affected when the wall was to the front.

A feature was created that calculates the drone's angle in relation to the wall. This was done as follows:

- ϕ: Roll angle in the yz-plane.
- ψ: Pitch angle in the xz-plane.
- θ: Angle between drone and wall in the xy-plane.

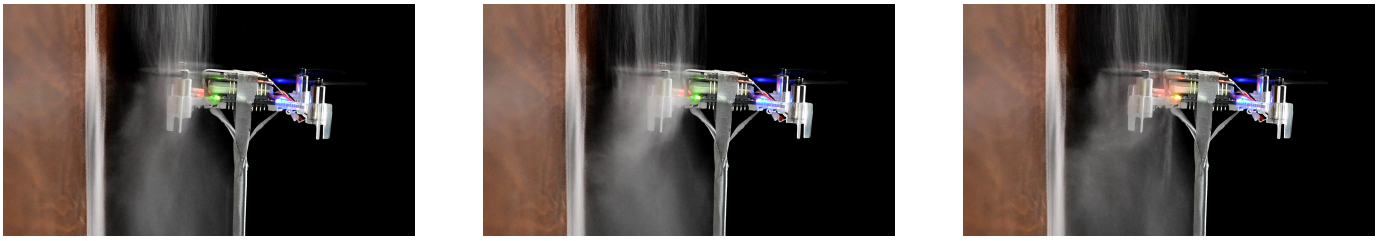
We averaged the roll and pitch angles in a sliding window and from this a unit vector was calculated for roll and pitch as followed:

$$\vec{r} = \begin{bmatrix} 0 \\ \sin\phi \\ \cos\phi \end{bmatrix} \quad \vec{p} = \begin{bmatrix} \sin\psi \\ 0 \\ \cos\psi \end{bmatrix} \tag{6}$$



(a) (b) (c)

Figure 3: Sequence of frames from video showing airflow at 80% thrust



(a) (b) (c)

Figure 4: Sequence of frames with wall at 80% thrust

Next, the vectors $\vec{r}$ and $\vec{p}$ were added together

$$\vec{Z} = \vec{r} + \vec{p} = \begin{bmatrix} \sin\psi \\ \sin\phi \\ \cos\phi + \cos\psi \end{bmatrix} \tag{7}$$

The vector is projected onto the xy-plane as that is the plane in which we are trying to measure the angle, the resulting 2D vector is

$$\vec{Z}_p = \begin{bmatrix} \sin\psi \\ \sin\phi \end{bmatrix} \tag{8}$$

The angle theta is obtained using $\arctan$.

$$\theta = \arctan\left(\frac{|\sin\psi|}{|\sin\phi|}\right) \tag{9}$$

As shown in Figure 5 the calculation is done with drone centered on the x-axis. A result of zero would mean that the wall is directly in front and a result of $\frac{\pi}{2}$ that the wall is perpendicular to the drone. It is expected that when the wall is to the front only the pitch angle is affected and when the wall is perpendicular only the roll is affected. When we have the same reading from pitch and roll we get $\frac{\pi}{4}$ which makes sense because we would expect the same effect on roll and pitch when the drone is at such an angle.

A second high-level feature as also developed. It uses the same sliding window average of the roll an pitch angles. The results from this algorithm followed more closely to the raw roll and pitch angles. The calculation is as follows:

$$r = \cos\phi \quad p = \cos\psi \tag{10}$$

$$\omega = \arctan 2(p, r) \tag{11}$$

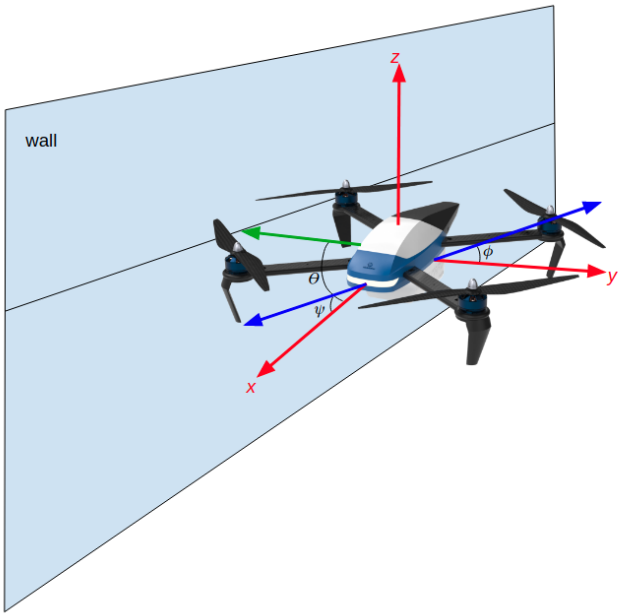
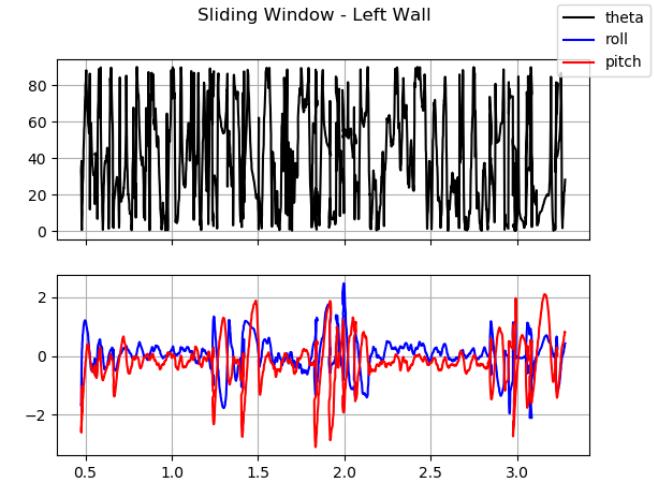
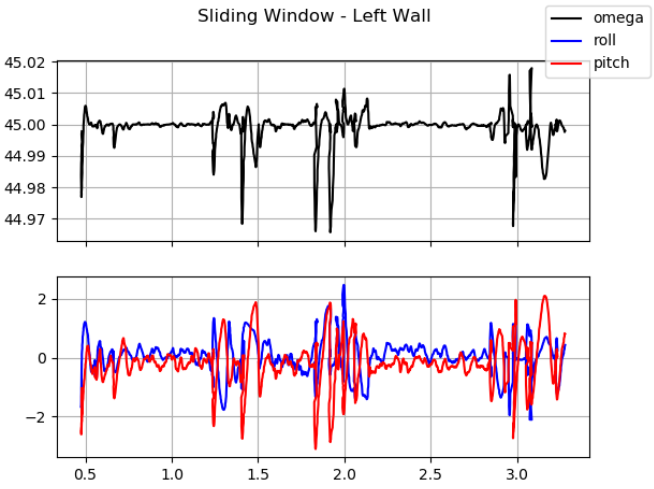


Figure 5: The drone's x, y, z axes in relation to the wall

We used this formulation because the cosine of the roll and pitch angles is $\vec{k}$ component using $\vec{i}, \vec{j}, \vec{k}$ form of the unit vectors mentioned in the previous algorithm. The $\vec{k}$ component refers the to Z direction of the vector. When we take $\arctan 2$ of both of these angles we get an angle that will be dependent on which angle was greater, and thus has greater effect on the drone.



(a) θ angle measurment



(b) ω angle measurment

Figure 6: Example time sequence data for each of the two high level features.

IV. Experimentation

A. Data Collection

Data needed to be gathered from the drone when it was flying near a wall to its left, right and front. Data was also collected from a drone flying while no walls or obstacles were near it. The drone used was a BitCraze Crazyflie 2.0 with a flow deck and loco position system (LPS) deck (fig. 2) [7]. The drone measures about 10 cm by 10 cm, and has total weight, including the decks, of 42g. The flow deck uses a laser sensor to determine the height of the quadcopter and the LPS deck gives off a signal to the LPS nodes in each corner of the flying area to determine its Cartesian position (x, y, z).

The drone was flown by a base computer using a Python interface. A driver program was created to fly the drone in a straight line in the x-direction when the wall was to the left, right or not present and in the y-direction when the wall was to the front. The wall was 1.2m long and 0.6m tall. The drone flew at a speed of 0.1 m/s and a height of 0.25m. Data was collected every hundredth of a second. We would expect to have 1200 examples but instability in the drone caused by it's proximity to the wall, means it often flies slower than 0.1 m/s when it is along the wall, so we usually got around 1500 examples in each flight. The drone was flown along each wall side five times and flown with no wall five times for 30 seconds, producing about 3000 examples in each flight.

While the quadcopter was flying, data was collected from the onboard sensors which includes an MPU-9250 inertial measurement unit (IMU). The gyroscope sensor has digital-output based on the x, y, z axes angular rates sensors (gyroscopes) with a user-programmable full-scale range of ± 250, ± 500, ± 1000, and ± 2000°/sec and integrated 16-bit ADCs. The accelerometer is a digital-output triple-axis component with a programmable full-scale range of ± 2g, ± 4g, ± 8g, and ± 16g and integrated 16-bit ADCs. In addition to the gyroscope and accelerometer data, stabilizer (roll,pitch,yaw), cartesian position (x, y, z) and barometer data were collected every hundredth of a second [7].The data was stored in separate files based on the wall position or no wall present.

B. Classifier Building

The classifiers were built using the SKLearn and Pandas packages in Python [8,10]. The data was initially trimmed as follows. Based on the modeling we concluded that the gyroscope would be most affected by the drone's proximity to the wall. As a result we dropped all features that were not the gyroscope, or stabilizer. What remained was five features: x-gyroscope, y-gyroscope, z-gyroscope, roll, and pitch.

We generated 18 features to summarize the data for data mining. The data was segmented into a sliding window with one second of the time-series data. The sliding window had a 0.99 second overlap, meaning the sliding window increased by one one-hundredth of a second for each calculation of the higher level features. The 18 features were as followed:

- Mean (5): Mean values of sensor, x, y, z axis for gyroscope and roll and pitch angle.
- Standard Deviation (5): Standard deviation, x, y, z gyroscope, roll and pitch angle.
- Mean Absolute Deviation (5): Mean absolute deviation, x, y, z gyroscope, roll and pitch angle.
- Average Resultant (1): For each of the 100 values in the window, take the square root of the three gyroscope axis and average them together.
- Angle Relation to Wall (1): Average the angle for the window, calculate the unit vector of the roll and pitch angles, add them together and project the resulting vector onto the xy-plane. We take the $\arctan$ of the two vector components to get what angle the drone should be in relation to the wall in ideal conditions.
- Cosine Angle (1): Calculate the average roll and pitch angles for the window and then take the cosine of both averages and take the $\arctan 2$ of the resulting values.

These first 16 features were generated because of their descriptiveness of the data in a more holistic view. The final two features were generated based on the modeling presented previously in this paper.

C. Experiments

Four experiments were done using the data that was collected and generated and was split randomly with a 80%-train 20%-test scheme. The first experiment was simply testing wall versus no wall. The wall data used was taken from five flights of the drone flying with a wall to its left and five to its right and the no wall data was taken from five flights of the drone flying with no wall but for a longer time. The result was 16,900 examples of wall data and 17,883 examples of no wall data.

The second experiment was testing right wall data versus left wall data. The data was taken from the same five left and right flights mentioned previously, resulting in 8,700 examples of left wall data and 8,200 examples of right wall data.

The third experiment was done using three classes which were front, right and left wall. The front wall data was taken from five flight of the drone moving in the y-direction (sideways) with a wall directly in front of the drone. There were the same number of left and right examples shown above and an additional 9,100 examples of front wall data.

The final experiment used four classes which were front, right, left and no wall. Only three flights of the drone flying with no wall were used since those flights were a longer duration and the three flights were selected at random. This resulted in 8,888 examples of no wall data along with the same number of examples for the other three classes stated previously.

All four experiments were done on three classifiers, RandomForest (RF), GradientBoosting (GB) and K-NearestNeighbors (KNN) with $K = 5$. Each experiment was repeated 100 times on each of the three classifiers and an accuracy score was taken at each test and averaged at the end of the 100 tests.

V. Results

The results for the four experiments are shown and described below. The bar graphs showing the results for each of the four experiments are shown in Figure 7.

A. Wall versus No Wall

For the results of the first experiment the RF classifier had the highest accuracy with a 99.85% accuracy for the average of 100 tests. The KNN classifier was second best at 87.74% accurate followed by GB classifier at 85.34% accurate.

B. Left versus Right Wall

This seemed to be the hardest task for RF to classify, but it still had the highest accuracy of the three classifiers with an accuracy of 80.68% followed by GB at 78.98% and finally KNN at 73.17%.

C. Left versus Right versus Front

Again the RF classifier preformed the best increasing in accuracy from the previous test to 87.45%. It was followed by GB at 72.94% accuracy and KNN at 66.47% accuracy.

D. Left versus Right versus Front versus No wall

RF continued to improve to 90.61% accuracy while KNN and GB continued to do poorer at 65.42% and 68.48% accuracy respectively.

VI. Discussion

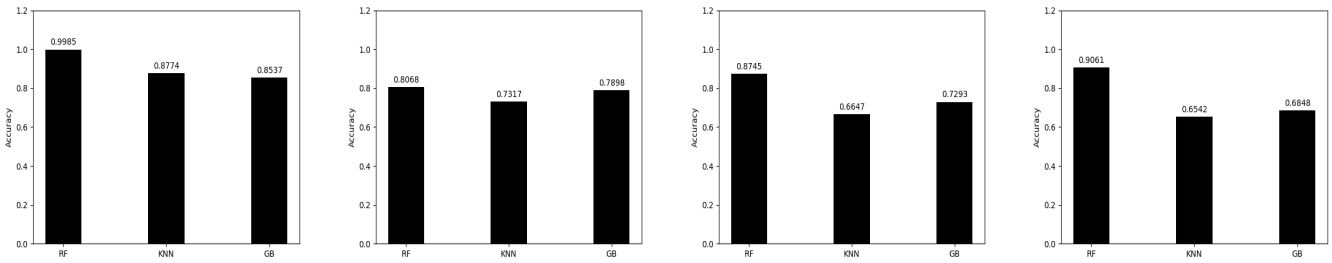
Here we discuss the accuracy of the three classifiers and which one would be best to implement into a driver. We also further discuss the 'artificial' ground effect that we have predicted and hypothesize is responsible for the success of our classifiers.

A. Classifier Accuracy

The RF classifier was most accurate in all 4 experiments, and the only one to preform better when testing on more than two classes when compared to left versus right testing. Contrary to this, KNN and GB both preformed worse an each consecutive experiment. All three classifiers preformed well on the first experiment. This is likely due to the clear effect the wall has on the quadcopter as shown in Figure 8. When the drone is flying near a wall there is clearly visible pattern of pose disturbance as compared to flying with no wall. The left versus right wall test was the worst performer for the RF classifier. This is likely due to the fact that a wall to the right has a similar effect to that of a wall to the left. The 80% accuracy was not bad considering the task the classifier was given. RF continued to do better in both the left, right, front experiment and the left, right, front, no wall experiment. The front wall has a different effect on the drone since the wall is on a different axis. The same applies to the final experiment with no wall because the no wall is distinctly different from when the drone is flying near a wall, whether it be to the front or side. RF's significantly better accuracy in the experiments makes is the best option to include in a driver program. While its has a long training time it can test very quickly when the drone is in flight.

B. Artificial Ground Effect

We hypothesize that the unsteadiness of the drone occurs because of an artificial ground-effect. It is well studied that when drones are near the ground they become unsteady because the air from their rotors hits the ground and backwashes towards the quadcopter. In our case the air is hitting the wall and then dispersing, as air does when it hits an object. This dispersion causes a mass of air underneath the drone acting as the "artificial ground". The proposition is shown in figure 9a and highlighted from the airflow imaging in 9b. Since the "ground" is only on one side of the drone there is likely a difference in gyroscope, accelerometer and stabilizer reading between a left wall, right wall and front wall.



(a) Wall vs. No Wall (b) Left vs Right (c) Left vs Right vs Front (d) Left vs Right vs Front vs No Wall

Figure 7: Classifier Testing Accuracies

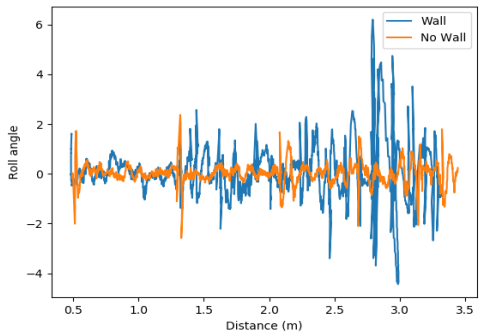
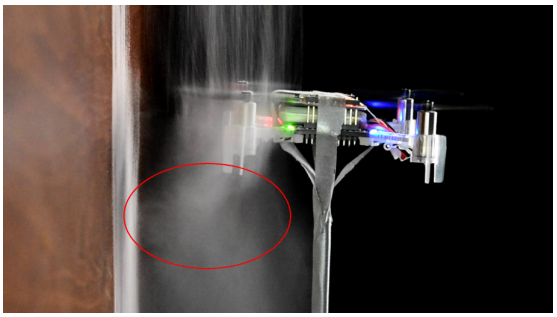


Figure 8: Wall vs. No Wall Roll Angle



(a)

Figure 9: Artificial Ground Effect

In summary, we can accurately predict if a wall is present or not. We can also predict whether the wall is to the left right or front with descent accuracy using the RF classifier. We determined that there is more difficulty determining left wall versus right wall. Lastly, we can also predict left vs. right. vs. front. vs no wall with 90% accuracy. We've determined that the RandomForest classifier is best for this classification.

C. Future Work

Future work includes adding features based on the mathematical modeling with the Navier-Stokes equations. Our expectation is that these will be more descriptive of the difference between a wall to the left or right or even at an non-perpendicular angle. Additionally, the classifier can be implemented into a driver program so that the drone can detect a wall using the classifier and then move around the obstacle accordingly.

REFERENCES

[1] S. Gu, M. Lin, T.-H. Nguyen and D. Lyons, "Wind gust detection using physical sensors in quadcopters," in arXiv:1906.09371 [cs.RO].

[2] Q. Zhou, J. Hughes, and D. Lyons, "Drone Proximity Detection Via Air Disturbance Analysis". SPIE Defense + Commercial Sensing Volume 11425, Unmanned Systems Technology XXII, May 2020; https://doi.org/10.1117/12.2556385

[3] F. Chui, G. Dicker, I. Sharf, "Dynamics of a Quadrotor Undergoing Impact with a Wall" 717-726. 10.1109/ICUAS.2016.7502535, 2016.

[4] A. Thai, R. Jain, S. Grace, "CFD Validation of Small Quadrotor Performance using CREATE-AV Helios" Vertical Flight Society 75th Annual Forum & Technology Display, 2019.

[5] S. Yoon, H. Lee, T. Pulliam, "Computational Analysis of Multi-Rotor Flows" 10.2514/6.2016-0812, 2016.

[6] P. Diaz, S. Yoon, "High-Fidelity Computational Aerodynamics of Multi-Rotor Unmanned Aerial Vehicles" 10.2514/6.2018-1266, 2018.

[7] Bitcraze Crazyflie 2.0, www.bitcraze.io/products/old-products/crazyflie-2-0/, Accessed May, 2020.

[8] Scikit-Learn, https://scikit-learn.org/stable/, Accessed April, 2020.

[9] S. Ershkov, "Non-Stationary Helical Flows for Incompressible 3D Navier-Stokes Equations", Applied Mathematics and Computation 274 (2016) 611-614.

[10] Pandas, https://pandas.pydata.org/, Accessed April, 2020.

[11] R. Deters, G. Ananda, M. Selig, "Slipstream Measurements of Small-Scale Propellers at Low Reynolds Numbers" 10.2514/6.2015-2265, 2015.

[12] M. Rwigema, "Propeller Blade Element Theory With Vortex Wake Deflection", 27th Internatonal Congress of the Aeronautical Sciences, 2010.

[13] D. Floreano, R. Wood, "Science, Technology and the Future of Small Autonomous Drones" **521**, 460–466 (2015). https://doi.org/10.1038/nature14542.

[14] M. Bangura, M. Melega, R. Naldi, R. Mahoney, "Aerodynamcis of Rotor Blades for Quadcopters" arXiv:1601.00733, 2016.

[15] T. Luukkonen, "Modelling and Control of Quadcopter" Aalto University, 2011.

[16] G. Hoffmann, H. Huang, S. Waslander, C. Tomlin, "Quadcopter Helicopter Flight Dynamics and Control: Theory and Experiment" AIAA Guidance and Control Conference and Exhibit, 20-23 August 2007.

[17] C. Hooi, F. Lagor, D. Paley, "Flow Sensing for Height Estimation and Control of a Rotor in Ground Effect: Modeling and Experimental Results" Annual Forum Proceedings - AHS International. 3. 1878-1889, 2015.