Spring 4-2020

DRONE PROXIMITY DETECTION VIA AIR DISTURBANCE ANALYSIS

Qian Zhao

Jason Hughes

DRONE PROXIMITY DETECTION VIA AIR DISTURBANCE ANALYSIS

Q. Zhao, J. Hughes, D. Lyons,
Robotics and Computer Vision Laboratory
Fordham University NY 10458

ABSTRACT

The use of unmanned aerial vehicles (drones) is expanding to commercial, scientific, and agriculture applications, including surveillance, product deliveries and aerial photography. One challenge for applications of drones is detecting obstacles and avoiding collisions. A typical solution to this issue is the use of camera sensors or ultrasonic sensors for obstacle detection or sometimes just manual control (teleoperation). However, these solutions have costs in battery lifetime, payload, operator skill. We note that there will be an air disturbance in the vicinity of the drone when it's moving close to obstacles or other drones. Our objective is to detect obstacles from monitoring the aforementioned air disturbance, by analyzing the data from the drone's gyroscope and accelerometer. Results from three experiments using the Crazyflie 2 micro drone are reported here. We show that it is possible to reliably detect when a drone is passing under another by using data mining algorithms to recognize the air disturbance caused by the other drone.
Keywords: Drone, obstacle detection, inexpensive, data mining.

1. INTRODUCTION

The use of unmanned aerial vehicles (drones) is expanding to commercial, scientific, and agriculture applications, such as surveillance, product deliveries and aerial photograph. One challenge for any drone application is detecting obstacles and avoiding collisions. Typical solutions to this issue have been the use of camera sensors or ultrasonic sensors for obstacle detection along with obstacle avoidance software, or sometimes just manual control (teleoperation). However, these solutions have costs in battery lifetime, payload, and operator skill. Fortunately, most drones are equipped with an inertial measurement unit (IMU). The IMU reports on the drone's attitude and accelerations from the gyroscope and accelerometer. When a drone operates close to the ground there is a backwash of air from the ground – *the ground effect* [5][13]. This backwash causes extra perturbation of the drone and its flight control software has a more difficult time maintaining it in its commanded pose or trajectory. Thus, any pose or motion in this scenario will show higher IMU recorded perturbations than a pose or motion away from the ground. In fact, this perturbation situation has been studied in some other situations as well, close to a ceiling for example [15]. We theorize that when a drone is close to any surface this effect occurs to some degree. If one drone flies underneath another drone, the downdraft from the upper drone will also cause give rise to this effect. Our long-term objective is to be able detect the proximity of the drone to obstacles by analyzing the data from the gyroscope and accelerometer looking for this perturbation effect. As a first step, in this paper we investigate the air disturbance produced when one drones flies close to or underneath an overhead drone. This would give us an inexpensive 'drone proximity' sensor. We will employ a data-mining approach with no aerodynamic modeling.

We choose a small drone, ***the Crazyflie 2.0***, as the experiment tool. The Crazyflie 2.0 is a lightweight, open source flying development platform based on a micro quadcopter. It has several built-in sensors including gyroscope, and accelerometer. A python-based ROS (Robot Operating System) [8] driver was written to control the Crazyflie drone. Three experiments were designed to determine if it is possible to reliably detect the downdraft (air disturbance produced by the Crazyflie's rotors) from the sensor data from the gyroscope and accelerometer as one drone flies underneath a second.

The first experiment is a long-interval disturbance detection: the lower drone flies on a straight-line path that either passes beneath an upper drone or doesn't, and we attempt to detect which from the entire interval. The second experiment is a short interval disturbance detection, the goal of this experiment is to determine if it is possible to detect air disturbance from the gyro and accelerometer data in a small time slice like 1 second rather than the entire interval. The third experiment evaluates how well the classifier learned on one drone transfers to another. The experimental results show that we can

reliably detect air disturbance from the sensor data of the lower drone when it's flying beneath another upper drone; however, limited success is shown for transferring the classifier to a new drone.

Section 2 reviews prior literature in this area. Section 3 presents our experimental methodology and experiments. Section 4 presents the results of the three experiments. Section 5 presents conclusions and future directions.

2. LITERATURE REVIEW

Research on robotic quadcopter flight has been an active area in recent years. While some researchers prefer to design and build their own autonomous quadcopter e.g., [1], we have chosen to use an existing small microdrone, the Crazyflie-2 [10] in our work on obstacle detection via air disturbance monitoring. With its small payload, this drone has limited ability to carry addition obstacle avoidance sensors. For drones with a larger payload however, several methods have been developed for obstacle avoidance. In [2], for example, a methodology is presented for autonomous control of the AR.Drone 2.0 in partly unknown indoor flight using only on-board visual and internal sensing. Low-cost ultrasonic sensors were proposed in [3] for obstacle detection and collision avoidance for an autonomous quadcopter. That evaluation shows that the system is capable to detect a wall through smoke, which is not possible with optical sensors of course. But the system lacks the capacity to detect soft surfaces like foamed material and people wearing clothes. In contrast, our proposed approach, which evaluates the air disturbance produced by the quadcopter drones, can, at least in theory, handle that situation where the ultrasonic approach works and where it fails. In [14], the authors study collision detection for the case where a drone with rotor guards impacts a wall, to determine where and on which rotor guard the collision occurred based on IMU data. This approach is in fact quite similar in inspiration to our approach; however, we aim to avoid the collision with a surface rather than detect it. And in the short term, in the work reported here, we just look for potential collisions with a second drone, not a surface.

There is research in detecting the so-called ground effect for quadrotors [5]: the added buoyancy an air vehicle experiences close to the ground due to the trapped cushion of air beneath it. Ground effect is an aerodynamic phenomenon which reduces the induced drag of aircraft and thereby increases its lift-to-drag ratio. It is the lift increase generated by rotors when an aircraft is close to a surface. However, the approach to quadcopter control including detecting ground-effect described by Danjun et al. [5] doesn't only get the altitude data from the inertial measurement unit (IMU), but it also relies on the aerodynamic model of the vehicle which considers the mass of drone and the thrust produced by the four rotors. In the work reported here, we will not leverage any vehicle modeling in our data analysis. Ground effect detection is also used by [6] to control the motion of an ornithopter (a 'hummingbird' robot). While our concept could include ground-effect detection (it does not in this work), we focus instead on detection of the proximity of another quadcopter above the height at which ground effect is present.

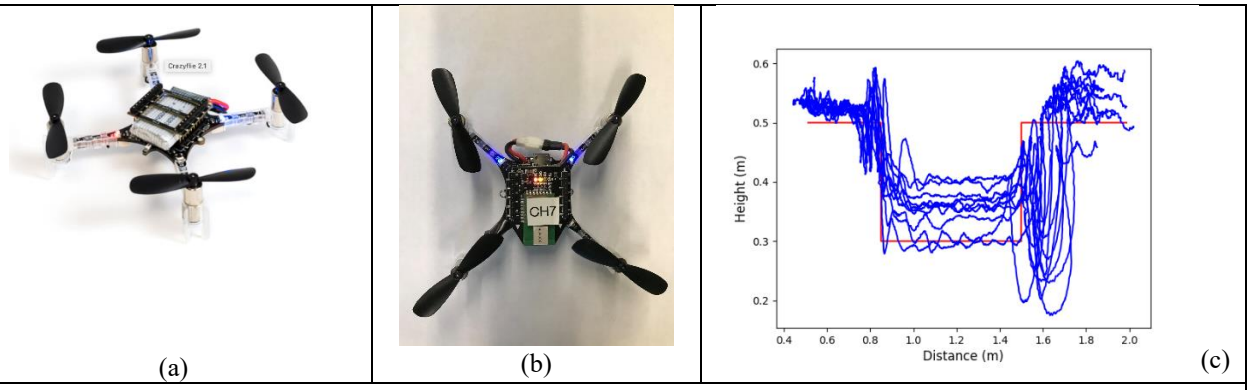
Gu et al. [4] report empirical results with real-life data, collected on the Crazyflie micro drone in a physical room with a consumer-grade fan, showed that (1) wind speed can be detected with high accuracy after training, not only on the same drone, but also across different drones of the same class (i.e., Crazyflie drones) , while (2) wind direction can be detected with high accuracy after training on the same drone, but with limited generalizability to other drones. The results show that using the gyroscope and accelerometer to detect the air disturbance is a feasible approach. But in this reported research, wind gusts from the side of the drone, i.e., perpendicular to the direction of motion, is produced by a consumer fan. This is not suitable for our experiment because we want to know how a drone will behave when it experiences a wind gust from an upper drone. However, we note that this will also be perpendicular to the motion of the drone, supporting evidence that this approach is worth evaluating.

3. EXPERIMENTAL METHODS AND DESIGN

The drone hardware used in this study is described in the first subsection below. Three experiments are proposed to study whether it is possible to reliably detect the situation where one drone flies under a second just from inspection of the air disturbance perturbations as measured by gyroscope and accelerometer. The remaining subsections present a description of these experiments.

3.1 Crazyflie 2.0 and Loco Positioning System

The Crazyflie 2.0 (Fig. 1(a, b)) is a versatile open-source quadcopter development platform that only weighs 27g. The drone is equipped with low-latency/long-range radio as well as Bluetooth LE. It also features four 7mm coreless DC-motors that give the Crazyflie a maximum takeoff weight of 42g [10]. This capacity enables it to carry multiple (small) expansion decks. For the work described in this thesis, the Crazyflie is equipped with the Loco Positioning System (LPS) deck [12], shown in Fig. 1(b). The Loco Positioning System is a system that tells where the quadcopter is in 3D Cartesian coordinates (x, y, z) with respect to a fixed coordinate frame (established when the LPS is installed).



(a) (b) (c)

Figure 1: The Crazyflie quadcopter (a); with LPS deck (b); flow-deck adjusted step (c).

The Crazyflie drone has several built-in sensors including gyroscope, accelerometer and magnetometer sensors. In our model, the done is equipped with an MPU-9250 inertial measurement unit (IMU) [11] that includes all these sensors. The gyroscope sensor has digital-output X-, Y-, and Z-Axis angular rate 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 accelerometer with a programmable full-scale range of ±2g, ±4g, ±8g and ±16g and integrated 16-bit ADCs.

The basis of the LP system is a set of Anchors that are positioned around the room (compare to the satellites in GPS); they are the reference points from which position/distance data is gathered. The other part of the system is one or more Tags (compare to a GPS receiver) that are fixed to the object(s) that are to be tracked. By sending short high frequency radio messages between the Anchors and Tags, the system measures the distance from each Anchor to the Tags and calculates the position of the Tags from that information [12].

The Crazyflie library (cflib[1]) supports the control of drones for experiments; however, it doesn't support the sensing of quadcopter position well. Besides, the library has some difficulty in meeting the challenge of being a platform for long term research with single drones or a swarm of drones, to control the drone(s) to move as expected. This is because the drone itself sometimes doesn't have very stable performance due to a variety of factors such as battery power, battery location, blade friction and so forth. This could raise an issue in trying to program a drone to fly in a straight line! We address this issue by working with ROS (the open-source Robot Operating System) and the Loco Position System (described above). A lightweight ROS driver was written in Python which worked as follows: One component of the driver listened to velocity messages in ROS, sending them to the drone using cflib, and published pose messages generated from the LPS system. A second component accepted setpoints from the user and using the pose information generated velocity messages to achieve that setpoint.

A further complication is that the Crazyflie drones fly most stably when used with a combination of the LPS deck and the *flow-deck*, and additional add on board that uses a time of flight sensor to accurately measure height and an optical flow sensor to control motion over the ground precisely. However, if a drone with a flow-deck flies over another drone (or indeed any rise or depression), the flow deck adjusts the height to be constant, causing a up/down motion of the drone. The ROS driver was written to detect the motion and compensate for it to maintain level flight, albeit with the knowledge of the change in measured height. Fig. 1(c) shows an average of 10 flights over a step in height using this software.

[1] https://github.com/bitcraze/crazyflie-lib-python

3.2 Long Interval Disturbance Detection

The goal of the first experiment is to determine whether it is possible for a lower drone flying on a straight line path that may or may not pass beneath an upper drone, to determine from inspection of data from the entire path whether there was an upper drone or not. A single drone was positioned to hover at a height of 0.5m above the ground approximately 1.3 meters along a horizontal track. A second drone was positioned at the start of the horizontal track, flown to 0.2m above the ground and then forward along the track under the first drone for approximately 2.4m. This experiment was repeated 10 times with an upper drone present, and 10 times with no upper drone present. The time of each trip was 24 seconds., The position, gyro, accelerometer, stabilizer data from the sensors mounted on the lower drone are stored to a CSV file. The data log is recorded at intervals of 10 ms, so the whole data file for one path has 2400 rows of data. Each entire path is labelled either as drone present, or as no drone present. Fig. 2 shows an example graph of the height of the moving drone against distance covered with upper drone present. The effect of the upper drone is pronounced at a distance of approx. 2.75m. Similar effects can be seen in the gyroscope and accelerometer data. Fig. 3 shows a selection of 4 video frames from one run of the lower drone flying under the upper drone.

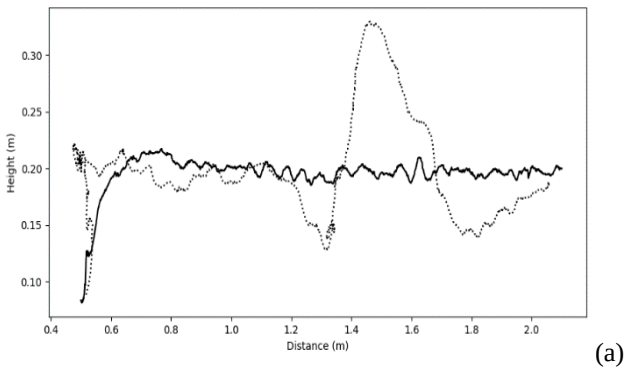


Figure 2. Graph of lower drone height against forward distance with upper drone present at x=1.3m (dotted line) and not present (solid line) (a); Gyroscope data for a case with drone present at x=1.3m (b).

Some of this data, the training data, will be used to train a classifier and another portion of this data will be used to evaluate the classifier in a standard *k*-fold cross validation.

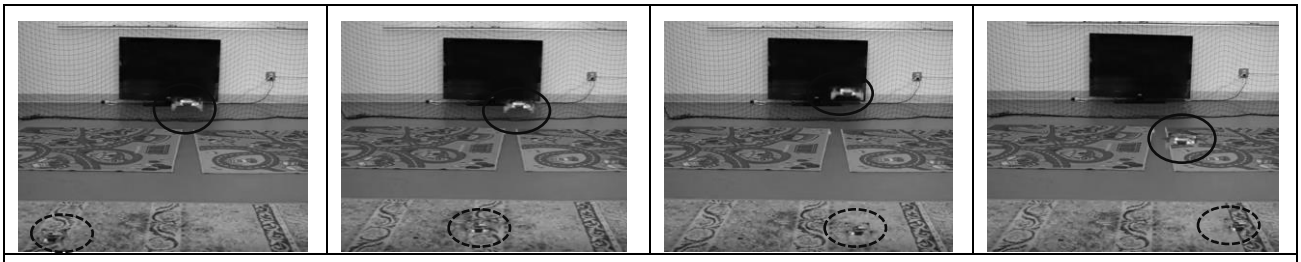


Figure 3. Example frames from one run of a lower drone (dashed ellipse) flying under an upper drone (solid ellipse)

3.3 Short Interval Disturbance Detection

The goal of this experiment is to determine if it is possible to detect air disturbance from the gyro and accelerometer data in a small time slice of 1 second. The second experiment is a more realistic evaluation (than the first experiment) of how air disturbance could be deployed as a drone proximity sensor. The data from 20 runs with heights and distances as in experiment one (10 with upper drone and 10 without) are divided into 1 second samples. The samples are labelled as either drone present or as no drone present, depending on whether they were taken at a location where the physical extent

of the drones overlapped (at time *t=8* to *t=12* seconds inclusive). This resulted in 40 samples of the drone label, and 133 of the no-drone label. This division into labeled, 1-second data results in a grossly unbalanced dataset, since most of the samples are for no drone present. To rectify this, the drone present case was balanced by replication to yield equal numbers of both labels.

Cross validation was again used to train and test a classifier. Additionally, to evaluate the classifier, random 1 second slices were tested *in sequence* – to simulate how the classifier would be used in practice. This was done 30 times and the average graphed.

3.4 Learning Transfer

This experiment evaluates how the proximity detection performs if the model constructed from the data gathered on one drone is used on a second drone. This will help us understand how the classifier may need to be tuned or calibrated when installed on a new drone. The experimental method was broadly the same as for the previous experiment, except that the data samples on which the classifier is tested will be from another drone. Our original objective was to replicate the data collection procedure used in prior experiments on the new drone. That was not possible, because the heights (0.5m upper, 0.2m lower) did not result in reliable flights when a new drone was used! The heights had to be modified to 1m upper and 0.25m lower for the flight to be sufficiently stable for data collection. The distance and time remained the same, as did the data preparation and testing.

3.5 Data Preparation

We pre-processed the time-series data by segmenting it into overlapping periods using a slicing window of 1 second. Since the data are collected at 100Hz, one data window comprises of 100 raw data items. Next, each data window from each sensor, i.e., accelerometer, gyroscope and stabilizer, was used to extract 40 high-level features, including:

1. Mean (3 values): Mean sensor value (one for each axis).
2. Standard deviation (3): Standard deviation (each axis).
3. Average Absolute Difference (3): Average absolute difference between the 100 values and the mean of these values (each axis).
4. Average Resultant Acceleration (1): For each of the 100 sensor values in the window, take the square root of x, y, and z-axis value and then average them.
5. Binned Distribution (30): 10 equal-sized bins are formed using the range of (maximum – minimum) of the 100 values and record the fraction of the 100 values within each bin.

Because the drone is more unstable just as it takes off and just as it lands, the portions of the time sequence data just around take off and just around landing are omitted from processing for every experiment.

Learning the parameters of a prediction function and then testing it on the same data used for training is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set [18]. In our data samples, the test size is set as 0.2, which means 4/5 part of data are used for training, the rest is used for test. In a *k*-fold cross-validation, this step training/testing is repeated *k* (=5) times.

For this part, we use the Gradient Boosting Classifier (GB) module from the scikit-learn (Python machine learning) library [17]. GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage, regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Here the binomial deviance loss function is used, since we are classifying data into two groups. After training/testing, we get the confusion matrix from predictive result. Finally, we use the score function to returns the mean accuracy on the given test data and labels.

4. RESULTS

A data mining approach is used to analyze the data from the three experiments and to train a classifier [16]. The data from the air disturbance experiments are collected as time-series data logged from the accelerometer and gyroscope generated at a rate of 100Hz. Raw data such as these may be arbitrary, unstructured, or even in a format that is not immediately suitable for automated processing. Feature engineering and transformation are applied to generate the formatted data that will be used in data mining processing. We will employ the Gradient Boosting [17] algorithm to analyze the formatted data and determine if the presence of a drone can be automatically detected. Gradient boosting is an ensemble machine learning algorithm for classification problems. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. Several other algorithms were also tested for this work; however, the best results were obtained with Gradient Boosting, and that will be the only one reported here.

4.1 Results: Experiment One

Table 1 shows the confusion matrix (CM) and accuracy score for Experiment one, the experiment in which we train a classifier to distinguish between examples of the 24 second flights in which there was a drone overhead and examples in which there was not, using *n=20* trials. Table 1 (a, top) shows the CM for the model learned from gyroscope data. Actual_0 and predicted_0 refer to the true and predicted no-drone case, and actual_1 and predicted_1 refers to the true and predicted drone case. The CM for Table 1(a) has a strong L-R diagonal indicating that the classifier is producing the correct prediction in general.

The accuracy score returns the mean accuracy on the given test data and labels. For Table 1(a) the accuracy is 88.4%. Table 1(b) shows the equivalent results for the classifier trained on accelerometer data. The CM again shows a strong L-R diagonal, and the accuracy is 94.1%. It's possible these can be combined for even better accuracy, but we leave this for future work.

Table 1: Confusion Matrix and Accuracy for Exp.One, Gyroscope (a), Accelerometer (b)

Confusion Matrix ¶

```
In [32]: get_confusion_matrix(y_test_gyro, y_pred_gyro)
```

Out[32]:

	actual_0	actual_1
predicted_0	3576	306
predicted_1	597	3285

```
In [33]: clf_gyro.score(X_test_gyro, y_test_gyro)
Out[33]: 0.883693972179289
```

(a)

Confusion Matrix

```
In [36]: get_confusion_matrix(y_test_acc, y_pred_acc)
```

Out[36]:

	actual_0	actual_1
predicted_0	3702	180
predicted_1	278	3604

```
In [37]: clf_acc.score(X_test_acc, y_test_acc)
Out[37]: 0.9410097887686759
```

(b)

Table 2: Confusion Matrix and Accuracy for Exp.2, Gyroscope (a), Accelerometer (b)

Confusion Matrix

```
In [32]: get_confusion_matrix(y_test_gyro, y_pred_gyro)
```

Out[32]:

	actual_0	actual_1
predicted_0	5152	308
predicted_1	792	4668

```
In [33]: clf_gyro.score(X_test_gyro, y_test_gyro)
Out[33]: 0.8992673992673993
```

(a)

Confusion Matrix

```
In [36]: get_confusion_matrix(y_test_acc, y_pred_acc)
```

Out[36]:

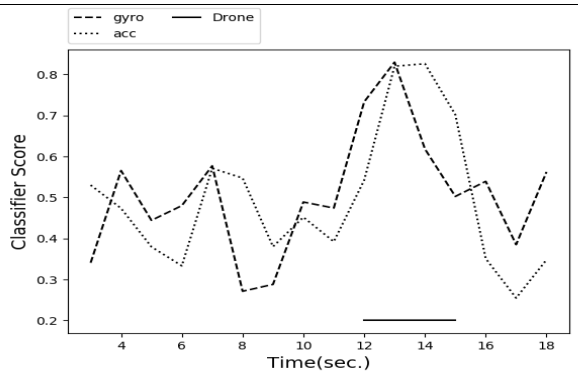
	actual_0	actual_1
predicted_0	5236	224
predicted_1	1992	3468

```
In [37]: clf_acc.score(X_test_acc, y_test_acc)
Out[37]: 0.7970695970695971
```

(b)

4.2 Results: Experiment Two

Recall that in this experiment, the classifier is trained on the data from all missions divided into one second slices, labelled and rebalanced, and is determining for each one 1 second of sensor data if there is another drone in the vicinity. Again



(c)

Figure 4. Gyroscope score against time (b); accelerometer score against time (c), experiment two.

n=20 trials were carried out. Table 2 shows the results for this. The CM for the gyroscope classifier is the better of the two, with an accuracy is 89.9% as opposed to 79.7%.

As an additional test for Experiment Two, we evaluated the classifier by picking time slices in sequence and evaluating the classifier score on them. Time slices were sampled in sequence and the classifier score generated. This was done 30 times, and the average classifier score per time slice is shown in Figure 4 as a graph of classifier score (for each of the two classifiers) against time. Both scores peak strongly in the vicinity of the drone, supporting the potential usefulness of this approach as a 'drone proximity sensor.'

4.3 Results: Experiment Three

The third experiment evaluates how well the classifier learned on one drone transfers to a second. There were *n=5* trials of this new configuration carried out and the data collected. The 4/5 train/test split was used again, with the training data coming from the prior experiment and the test data from the additional 5 trials. Figure 5 shows a graph of classifier performance, computed as in the second part of Experiment Two above. Recall that not only was a new drone used for the 5 trials, the heights in the new trials differed from those used in the training data. The gyroscope classifier score peaks very early, before the lower drone encounters the upper drone. The accelerometer score apparently shows no useful pattern. We discuss the interpretation of this and the other graphs in the conclusion.

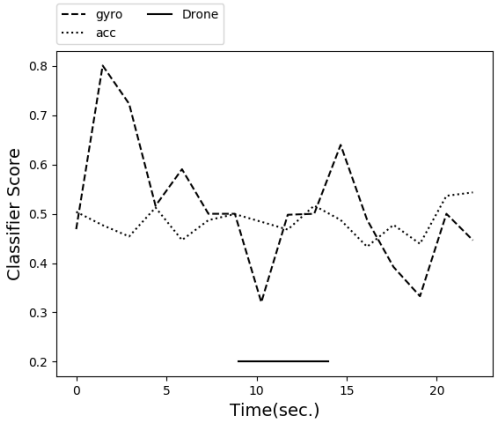


Figure 5. Gyroscope score against time slice (b); accelerometer score (c), experiment three (transfer).

5. CONCLUSION

The main contribution of the paper is the three experiments performed on drones to see whether it's possible to detect the air disturbance caused by passing close to or underneath an upper drone from analyzing sensor data. The experiments consist of a drone flying along a path that takes it momentarily under a second drone, to see if this situation can be detected by data-mining just from the air disturbance caused by the rotors of the upper drone. We discuss these results below.

Long interval disturbance detection tests the lower drone flying on a straight-line path that passes beneath an upper drone. The experiment has $n=20$ different trials, 10 with and 10 without upper drone, and the time of each flight trip is 24 seconds. This means we have 20 instances of 24-sec time series data. Training a gradient boost classifier and conducting a k-fold cross-validation produces the following performance: with accelerometer data, we get over 94% accuracy and with gyroscope data, over 88% accuracy. From the confusion matrix data, we can calculate that precision is 95% and 91% for each respectively, and recall is 93% and 85% respectively. The strong accelerometer performance could be due to the classifier relying on the height changes experienced around the upper drone.

Short interval disturbance detection attempts to see if it is possible to detect wind interference from the gyroscope and accelerometer data in a small time slice as the lower drone approaches the upper. For this experiment, the classifier trained on accelerometer data achieved 80% accuracy and that on gyroscope data, 90%. From the confusion matrix data, we can calculate that precision is 94% for both, and recall is 64% and 85% respectively. Although precision is good for both sources, both recall and accuracy are low for the accelerometer. The result shows that the approach to detect the proximity of the drone by analyzing the data in short time slice window is possible using gyroscope data. The height changes from experiment one are most likely not of a long enough duration to be visible in the data slices for this experiment; the gyroscope classifier is detecting the rotation instability which apparently has a shorter time scale effect.

The final experiment explores how well the classifier generalizes to a second drone. It was not possible to replicate the exact same experimental conditions from experiment two on a second lower drone. The effect of the upper drone was sufficient to completely destabilize the second lower drone using the initial heights (0.2m lower, 0.5m upper). This is already an indication that individual drones differ greatly, and transfer of learning between drones might be difficult. The experiment was replicated at different heights (0.25 lower, 1.0m upper) for the second, lower drone. The results in Fig. 5 look poor on first view. Looking just at the gyroscope trained classifier, there is an extremely strong detection, but at a much greater distance from the upper drone than before. We argue that this is a real detection. It was already evident that the upper drone had a much greater effect on this second lower drone, then it did on the first – that is why the heights had to be changed. We argue that the greater effect is what is leading to detection at a much earlier time. This would lead us to conclude that the classier either needs to be trained on the drone it's used on, or that a classifier trained on one drone would need significant tuning (perhaps transfer learning) to work on another drone.

We consider this work to be a feasibility study, to investigate the usefulness of monitoring air disturbance as a way to do inexpensive proximity detection. We only evaluated transit of one drone under another, and we used a model free approach. Our conclusions are that air disturbance monitoring can be used to reliably detect the when the lower drone approaches under the upper drone; however, the approach does not easily generalize. There may be a limit to a purely data-mining approach, and aerodynamic modeling may be necessary if we want to apply this to more general obstacle detection. In that approach, the rotor air flow and its interaction with the wall, floor, or ceiling is modeled similar to the collision modeling in [14]. The model is used to predict features to measure in the feature generation step of data mining. In future work, we will apply this approach to wall detection from air disturbance monitoring.

6. REFERENCES

[1] J. Enslin, R. Nichols, G. Kitchell, C. Walsh and J. Welch, "Autonomous Quadcopter," Tech. Report P01722 Rochester Ins. of Technology, Rochester NY, 2007.

[2] T. Mac, C. Copot, R. D. Keyser and C. M. Ionescu, "The development of an autonomous navigation system with optimal control of a UAV in partly unknown indoor environment,," *Mechatronics,* vol. 49, pp. 187-196, 2018.

[3] N. Gageik, B. Paul and S. Montenegro, "Obstacle Detection and Collision Avoidance for a UAV with complementary low-cost sensors.," *IEEE Access 3,* pp. 599-609, 2015.

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

[5] L. Danjun, Z. Yan, S. Zongying and L. Geng, "Autonomous landing of quadrotor based on ground effect modelling," in *34th Chinese Control Conference (CCC)*, Hangzhou, China, 2015.

[6] J. Grauer and J. (. Hubbbard, "Inertial Measurements from Flight Data of a Flapping-Wing Ornithopter.," *Journal of Guidance Control and Dynamics.*, vol. 32, no. 1, p. 326–331., 2009.

[7] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf and Oskar von Stryk, "Comprehensive Simulation of Quadrotor UAVs using ROS and Gazebo" 3rd Int. Conf. on Simulation, Modeling and Programming for Autonomous Robot 2012.

[8] W. Honig and A. N., "Flying multiple UAVs using ROS," in *Robot Operating System (ROS)*, Springer , 2017, pp. 83-118.

[9] M. Fernando, "Geometric Tracking Controlling of a Crazyflie 2.0 Nano drone," DOI: 10.13140/RG.2.2.12876.92803/1 Indiana University, Bloomington, Indiana, 2018.

[10] "Crazyflie 2.0.," 5 2019. [Online]. Available: https://www.bitcraze.io/crazyflie-2/.

[11] "MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS MotionTracking™ Device.," April 2019. [Online]. Available: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/.

[12] "Loco Positioning system.," May 2019. [Online]. Available: https://www.bitcraze.io/loco-pos-system/.

[13] D. Bernard, F. Riccardi,, M. Giurato, M. Lovera, "A dynamic analysis of ground effect for a quadrotor platform," IFAC 2017.

[14] F. Chui, G. Dicker, I. Sharf, " Dynamics of a Quadrotor Undergoing Impact with a Wall" Int. Conf. Unmanned Aircraft Sys, Arlington VA, June 2016

[15] Y. Hsiao and P. Chirarattananon, " Ceiling Effects for Hybrid Aerial-Surface Locomotion of Small Rotorcraft" IEEE/ASME Trans. Mech. July 2019.

[16] C. C. Aggarwal, Data Mining: The Textbook., Springer, 2015.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. a. P. M. Brucher and Duchesne, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, pp. 2825--2830, 2011.

[18] R. E. Schapire and Y. Freund, Boosting: Foundations and Algorithms., Cambridge MA: MIT Press, 2012.