Fly-Crash-Recover: A Sensor-based Reactive Framework for Online Collision Recovery

A Technical Report submitted to the Department of Systems and Information Engineering

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Matthew Trivett
Spring, 2020

Technical Project Team Members
Nicholas Anselmo
Anders Christoffersen
Miller Garret
Ryan Remias
Shirley Wang

On my honor as a University Student, I have neither given nor received
unauthorized aid on this assignment as defined by the Honor Guidelines
for Thesis-Related Assignments

Signature:_____ Date:_____
Matthew Trivett

Signature:_____ Date:_____
Assistant Professor Nicola Bezzo, Department of Engineering Systems and Environment,
Electrical Engineering and Computer Engineering

Fly-Crash-Recover: A Sensor-based Reactive Framework for Online Collision Recovery of UAVs

Shirley Wang, Nicholas Anselmo, Miller Garrett, Ryan Remias, Matt Trivett,
Anders Christoffersen, and Nicola Bezzo

***Abstract*—Unmanned Aerial Vehicles (UAVs) are becoming increasingly popular thanks to the multiplicity of operations in which they can be deployed such as surveillance, search and rescue, mapping, transportation, hobby and recreational activities. Although sensors like LIDARs and cameras are often present on such systems for motion planning to avoid obstacles, collisions can still occur in very dense and unstructured environments, especially if disturbances are present. In this work, we research techniques to recover UAVs after a collision has occurred. We note that the on-board sensors, especially the inertial sensor used to stabilize the UAV, run at a high frequencies obtaining hundreds of data points every second. At run-time, this can be leveraged at the moment of a collision to quickly detect and recover the system. Our approach considers knowledge of UAV system dynamics to predict the expected behavior of the vehicle under safe flight conditions and leverage such expectations together with inertial data to detect collisions rapidly (on the order of milliseconds). We also propose a potential field-based approach to map the collision and create the correct reactive maneuver to avoid the collided object and bring the system back to a stable and safe configuration. Experiments are executed using ROS on two micro-quadrotor UAV platforms having different dynamics and performances, while colliding with poles and walls positioned in different configurations. In our results, we are able to show that the UAVs are successfully able to detect and avoid a collision, while also providing a rigorous analysis of the conditions in which the system can recover from imminent collisions.**

***Index Terms*—UAV navigation, Collsion recovery, Motion planning**

I. Introduction

Recent advancements in drone technology have expanded their usability into diverse fields such as delivery services, reconnaissance, and search and rescue. In many applications, using an unmanned aerial vehicle (UAV) instead of an actual human being to complete a mission is safer and more cost efficient. However, as noted in [1], drones are beginning to be employed in unknown environments where levels of interaction and contact are high. Deploying drones in these situations can be advantageous, but in the case of UAV collision as depicted in Fig. 1, valuable technologies can be lost, sensitive information can be stolen, and safety of those in the surrounding environment can be threatened. Current research in this area is divided in two categories: 1) improving resiliency of the outer physical frame and 2) detecting collisions using sensor data and provide recovery operations. In this work, we focus on the second approach. By using an external frame to absorb most of the contact, a UAV may be more robust and likely to survive a crash than one without a protective frame. But this may not be enough to guarantee that the system will recover after a collision, as depicted in Fig.1. The second approach is more beneficial as it leverages sensor data to detect and aid collisions and quickly provide stabilization and recovery adaptations, thus reducing the chance of failure. Specifically in this work we build a framework that leverages the internal inertial data compared with the commanded inputs to detect and estimate the point of collision (POC) and use such information and artificial potential field theory to recover the UAV into a stable flight.

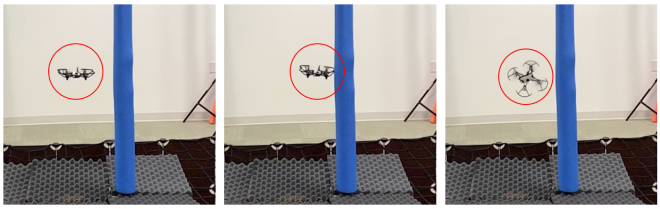


Fig. 1. Example of UAV colliding with an obstacle and ultimately failing.

The rest of the paper is organized as follows: in Section II, we discuss the state of the art on drone collision resiliency and recovery. We then continue with Section III which details our problem statement and Section IV which covers our overall approach for detection, estimation of collision and recovery. This is followed by Section V which provides results from testing with two UAVs colliding with different objects. Conclusions and discussion on future work are finally drawn in Section VI.

II. Related Work

As mentioned in the previous section, current research on collision recovery focuses either on building external mechanical protections or using sensing and planning/control methods. Authors in [2] deal with the former method and propose a flexible frame that absorbs the impact experienced during crashes and lifts the UAV if it falls on the ground. Similarly, authors in [3] structure their outer frame in a spherical shape creating a gimbal-like system to maintain the UAV at level flight.

Standing from a sensing and control point of view, authors in [4] formally characterize the equations of motion and design controllers to maintain stable flights on a quadrotor despite losing multiple propellers. Authors in [5] focus on an encoded collision strategy that takes form in three stages: detection, logic and recovery, and characterization with respect to the UAV behavior. Authors in [1] expand on this idea by developing a strategy where the drone switches from a normal flight state to a recovery state at the moment of collision by leveraging an admittance and impedance controller. Authors in [6] also present a similar system architecture, however they consider solely utilize IMU data for estimating the UAV attitude and velocity. Finally, standing more from a machine learning point of view, authors in [7] solve the problem of detecting and responding to failures on actuators and sensors

by training a reinforcement learning policy for fault-tolerant control of UAVs.

In our work, we consider similar challenges from the aforementioned work and build a sensor-based reactive framework to detect and estimate the point of collision at run-time to quickly recover the UAV into stable flight.

III. Problem Statement

In this project, we aim to study methods to recover and reconfigure a UAV into a safe mode of operation and if possible, continue the mission, even with degraded performance. Formally:

Problem - *UAV Collision Recovery*: Consider a quadrotor UAV tasked to perform a go-to-goal operation while avoiding N_o obstacles positioned at $\boldsymbol{o}_i = [x_{o,i}, y_{o,i}, z_{o,i}]'$, with $i = 1, \ldots, N_o$, along its followed trajectory τ. The quadrotor operation is considered safe and successful iff $||\boldsymbol{x}(t) - \boldsymbol{o}_i|| > \epsilon$, $\forall t > 0$ and $\forall i = 1, \ldots, N_o$, where ϵ is a minimum distance to maintain that considers the dimension of the quadrotor and $||\cdot||$ is the euclidean distance norm. If a collision with an obstacle i occurs at a time $\hat{t}$, then $||\boldsymbol{x}(\hat{t}) - \boldsymbol{o}_i|| \leq \epsilon$. The objective is to find a policy to recover the system after a collision such that $\forall t > \hat{t}$:

$$\begin{aligned} ||\boldsymbol{x}(t) - \boldsymbol{o}_i|| &> \epsilon \\ ||z(t) - z_{ground}(t)|| &> \epsilon \\ \lim_{t \to T} ||\boldsymbol{x}(t) - \boldsymbol{x}_{goal}|| &= 0 \end{aligned} \tag{1}$$

where $\boldsymbol{x}(t) = [x(t), y(t), z(t)]'$ is the 3D position of the quadrotor at time t and $z_{ground}(t)$ is the height of the ground at time t. The first two conditions in (1) provide safety margins to maintain a safe distance from obstacles and the ground, while last liveness condition is to guarantee that after a recovery is performed, if possible, the system continue the planned operation toward the goal. Note that if continuing the operation is not feasible, we would like to obtain a policy to at least stabilize and safely land the UAV before the goal.

IV. Approach

The goal of this research project is to create a generalized approach for collision detection and recovery. Fig. 2 summarizes our framework. We assume a quadrotor has a designated goal to visit and encounters one or multiple stationary obstacles during its operation. When a collision with an obstacle happens, the quadrotor enters a three-fold process: 1) detection 2) estimation and 3) recovery. Detection involves monitoring quadrotor inertial data during flight which is ideal because such data is available in any UAV. Estimation is the stage used to identify and record the point of collision (POC) with respect to the quadrotor. The POC is determined by further analyzing quadrotor IMU data at the moment of a crash. Finally, the recovery stage involves stabilizing the vehicle after the crash and then creating an adjusted path toward its desired goal. The estimated POC is used to remap the quadrotor's flight path. This three-fold approach is further explained in detail in the next sections.

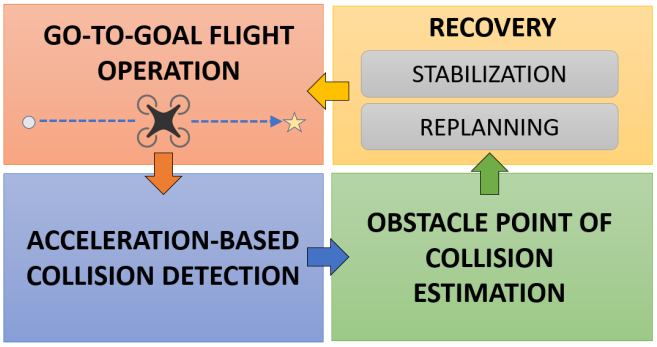


Fig. 2. Diagram of the proposed framework consisting of detection, estimation and recovery.

A. Collision Detection

The first component of our recovery scheme consists on the design of a detector based upon the linear acceleration data coming from the on-board IMU. This data source was selected because it is universal to all UAVs, does not require significant filtering, and can rapidly detect changes in the vehicle's motion. For the horizontal axes of the vehicle (X and Y), the detector is triggered when the magnitude of the linear acceleration $\boldsymbol{a}$ exceeds a threshold value Δ determined through experimentation:

$$||\boldsymbol{a}_{xy}|| = \sqrt{a_x^2 + a_y^2} > \Delta \tag{2}$$

This method alone is not sufficient, as it does not account for collisions that can occur in the vertical dimension of the vehicle. Also, noise and the influence of the vehicle's attitude on the IMU readings during movement make such detection suboptimal. To overcome these challenges, the detector algorithm also analyzes less noisy data from the Z-axis. Additionally, most collisions in the horizontal plane will cause a shock to travel through the body of the UAV and will register a spike in linear acceleration in the Z-dimension of the vehicle. This change in acceleration can be used to quickly detect collisions not along the Z-axis as shown in Fig. 3.

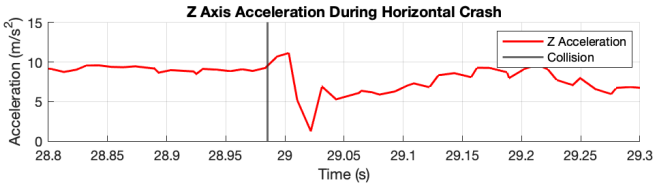


Fig. 3. Vertical (Z) acceleration during a horizontal collision.

Because the accelerations experienced by the IMU during vertical maneuvers easily exceed an otherwise reasonable trigger threshold, instead of a simple threshold, the Z-axis detector compares the response of the system to the commanded thrust in order to detect deviations that indicate a collision. As can be seen in Fig. 4, the commanded thrust and resulting acceleration of the UAV track each other well during maneuvers, with a slight delay between the command and response of the vehicle.

The Z-axis detector first accounts for the phase shift between the commanded thrust (T_c) and acceleration (a_z) by shifting the commanded thrust with an offset, t_σ, earlier than the current time (t_c), obtaining a shifted thrust $T_s = T_c(t_c - t_\sigma)$. Then, the detector subtracts a_z from T_s

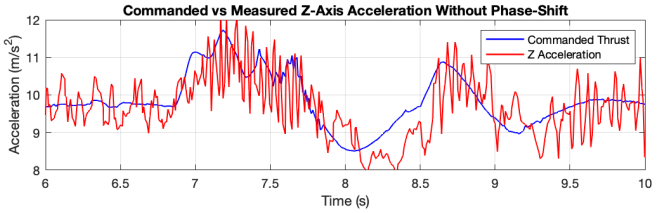


Fig. 4. Raw data comparing the commanded thrust to the IMU Z-acceleration. A slight phase shift is necessary to account for the delay between signals.

and evaluates against the trigger threshold as shown in the following equation:

$$|T_s - a_z| < \Delta_z \tag{3}$$

This operation filters the effects of the commanded thrust out of the UAV's Z-acceleration data. The resulting signal then captures only accelerations that are caused by an outside disturbance, such as a collision. By filtering out the effects of the commanded thrust, the resulting signal is centered and held steady around the point of zero acceleration, with only minimal noise remaining. As a result, any significant spike in acceleration signifies an outside disturbance and likely a collision. Fig. 5 shows this cancelling effect applied to a UAV's maneuvers. Plot (a) shows that after the command-correction is applied, the resulting signal in (c) stays between nominal values for no collision cases, while plots (b) and (d) show the detection operation during a collision.

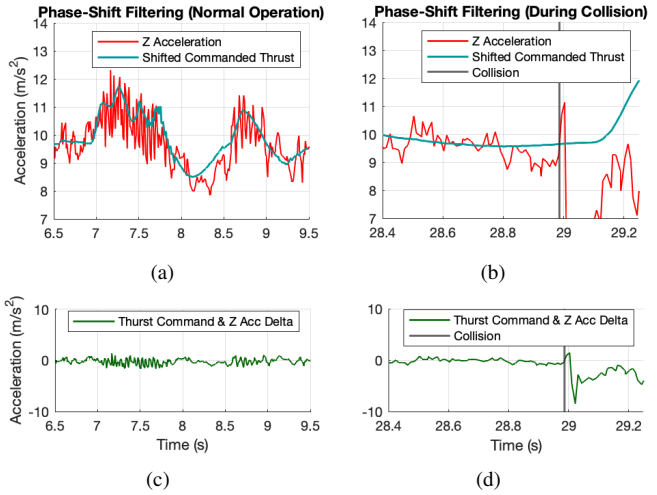


Fig. 5. Comparison between filtered command thrust data and z-acceleration values for a no-collision case (a),(c) and a collision case (b),(d).

B. Collision Characterization and Estimation

The second component of our approach is to characterize where the vehicle collided with respect to the vehicle as well as its environment. The goal is to identify the POC, $\boldsymbol{o}_i$, which will be recorded as a known obstacle so that it can be avoided in the future. When a collision occurs, the resultant behavior can be divided into two categories: 1) the quadrotor bumps in the opposite direction of motion or 2) the quadrotor continues its motion tilting toward the obstacle. Both of these scenarios can be classified by checking the sign of the X and Y acceleration components in relation to the direction of motion of the UAV. For example in Fig. 6, we show the linear acceleration data

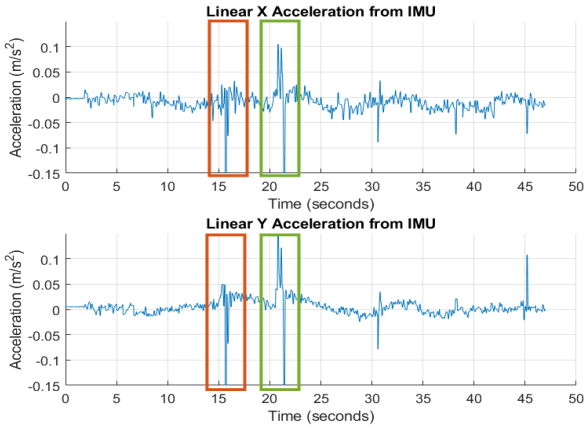


Fig. 6. The X and Y linear acceleration of a quadrotor colliding twice with the same obstacle oriented at 135°. Note the sign changes in the acceleration components.

for an experimental case study in which the quadrotor was traveling in the positive Y direction. Around $t = 15$s, the quadrotor has a collision recording accelerations in the -X and -Y directions. At $t = 22$s, the quadrotor collides again this time with accelerations in the +X and +Y directions. Both cases in this example are distinctive and associated with obstacles oriented with angles $90° < \theta < 180°$. For any collision case, we argue that a quadrotor should move in the direction opposite to its original flight direction along the normal of the POC. Based on these observations, we built the following estimator.

Consider the position of the quadrotor $\boldsymbol{x}(t)$ and ϵ its radius which is calculated as the distance from its center to the tip of one propeller. We define $\boldsymbol{v}(t)$ to be the quadrotor's command velocity vector, and $\boldsymbol{a}(t)$ as the quadrotor's linear acceleration vector. The angle λ between $\boldsymbol{v}(t)$ and $\boldsymbol{a}(t)$ at collision is defined as:

$$\lambda = \cos^{-1}\left(\frac{\boldsymbol{a}\cdot\boldsymbol{v}}{\|\boldsymbol{a}\|\|\boldsymbol{v}\|}\right) \tag{4}$$

The estimation vector $\boldsymbol{e}_i$ is then computed by:

$$\boldsymbol{e}_i = \begin{cases} \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}\epsilon, & \text{if } \lambda < 90° \\ -\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}\epsilon, & \text{if } \lambda \geq 90° \end{cases} = \langle x_e, y_e \rangle \tag{5}$$

and used to ultimately determine the POC $\boldsymbol{o}_i$ as follows:

$$\boldsymbol{o}_i = \{x_o, y_o, z_o\} = \{x(t) + x_e, y(t) + y_e, z(t)\} \tag{6}$$

Here we assume a level flight in the Z direction, therefore (4) and (5) assume calculation in only the X and Y dimensions.

C. Collision Recovery

The recovery approach has two main components: first stabilization, and then trajectory replanning.

1) Stabilization: The first phase of recovery begins by enforcing "aggressive" maneuvers to return to a horizontal configuration as quickly as possible. This allows the drone to achieve stability, and ensure the thrust is oriented correctly for further recovery maneuvers. The major challenge encountered during this phase is the non-linear nature of quadrotor dynamics at extreme pitch and roll angles. For these cases, a simple PID controller is not sufficient to achieve both quick response and low overshoot. To address this problem, we developed

a gain-scheduled PID controller to provide a robust solution that is easy to implement on many platforms. Previous research into this control scheme [8] in the context of quadrotors shows that a gain-scheduled PID controller can achieve stability as quickly as 0.2 seconds with little to no overshoot. Because this method only requires changing PID gains in response to the current pitch and roll, implementation typically requires minimal changes to the existing controllers. However, this method does require physical testing or accurate simulation of the dynamics of the vehicle to determine the appropriate values for the gains across the operating range.

2) Replanning: The second component of recovery is to replan a flight path for the quadrotor after a collision occurs. Given a quadrotor with a desired goal, we aim to minimize the possibility of it colliding with the same obstacle multiple times. We propose to leverage the theory of artificial potential fields as an approach to obstacle avoidance.

The theory of potential fields combines attractive and repulsive fields to generate a vector field [9]. A potential function is a function $U : \mathbb{R}^m \to \mathbb{R}$. The idea is that any object placed at any point in this potential field desires to move to a position of lower potential. This can be calculated into a vector field based on the gradient (7) of the potential function for every point $\boldsymbol{x}$ in the field at a particular time t. A quadrotor's velocity at a point $\boldsymbol{x}$ is calculated in (8) as the negative gradient of the potential field or the sum of all forces acting on the field:

$$\nabla U(\boldsymbol{x}) = \left[\frac{\partial U}{\partial \boldsymbol{x}_1}(\boldsymbol{x}), ..., \frac{\partial U}{\partial \boldsymbol{x}_m}(\boldsymbol{x})\right]' \tag{7}$$

$$\boldsymbol{\tau}(\boldsymbol{x}) = -\nabla U(\boldsymbol{x}) = F_{att}(\boldsymbol{x}) + F_{rep}(\boldsymbol{x}) \tag{8}$$

For our approach, the potential field generated is the sum of one attractive field and multiple repulsive fields. The attractive field component and subsequent force vector calculation is given by:

$$\begin{aligned} U_{att}(\boldsymbol{x}) &= \frac{1}{2}\xi(\|\boldsymbol{x} - \boldsymbol{x}_{goal}\|^2) \\ F_{att}(\boldsymbol{x}) &= -\nabla U(\boldsymbol{x}) = -\xi(\boldsymbol{x} - \boldsymbol{x}_{goal}) \end{aligned} \tag{9}$$

where $\boldsymbol{x}$ is the quadrotor's position, $\boldsymbol{x}_{goal}$ is the goal position, and ξ is a constant gain.

In section IV-B, we proposed a method for calculating the POC. This position is then recorded and used to create a map of POCs. This map of POCs is then used to create repulsive potential fields to recover the UAV as follows:

$$\rho(\boldsymbol{x}) = \|\boldsymbol{x}(t) - \boldsymbol{o}_i\| \tag{10}$$

$$U_{rep}(\boldsymbol{x}) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho(\boldsymbol{x})} - \frac{1}{\rho_0}\right)^2, & \text{if } \rho(\boldsymbol{x}) \leq \rho_0 \\ 0, & \text{if } \rho(\boldsymbol{x}) > \rho_0 \end{cases} \tag{11}$$

$$F_{rep}(\boldsymbol{x}) = \eta\left(\frac{1}{\rho(\boldsymbol{x})} - \frac{1}{\rho_0}\right)\frac{1}{p^2(\boldsymbol{x})}\nabla\rho(\boldsymbol{x})$$

where the function $\rho(\boldsymbol{x})$ is used to calculate the distance between the quadrotor and the nearest POC in (10); $\boldsymbol{x}(t)$ is the quadrotor's position at time t, and $\boldsymbol{o}_i$ is the POC calculated from (5). Each obstacle in this environment is represented as a repulsive field expressed in (11), where η is an experimental constant, and ρ_0 is the distance from which the repulsive field begins acting on the quadrotor considering the dimension paramter of the quadrotor, ϵ.

V. Results

A. Experimental Setup

Our approach was validated through experimentation with two quadrotors in a controlled environment: 1) a DJI Tello quadrotor protected by propeller guards which is a robust platform but with limited access to the low-level controller and 2) an unprotected Crazyflie 2.0 quadrotor, which provides access to low-level control parameters. Orientation and position information for all experiments were acquired using on-board IMU sensors and a Vicon motion capture system (MOCAP), for ground truth. To test and analyze our framework on these two platforms we considered collision on a pole and on a flat panel positioned at different orientations.

In Fig. 7, we show the control architecture used during experiments built using the Robot Operating System (ROS) to allow communication between the MOCAP system, an external base station computer running the high-level detection estimation and replanning algorithms, and the drones which provided IMU data to the external computer and executed the low-level commands. This setup allowed us a rapid turnaround on control algorithms without the need to recompile code to the drones themselves, and with much more portability between our platforms and to other platforms in the future.

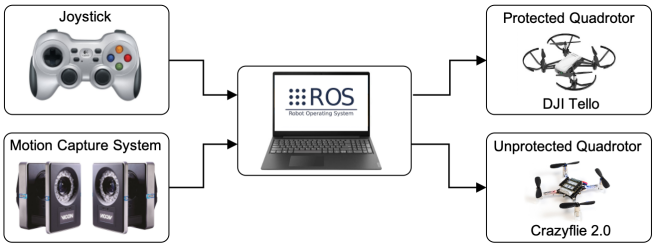


Fig. 7. Experimental setup utilizing ROS architecture.

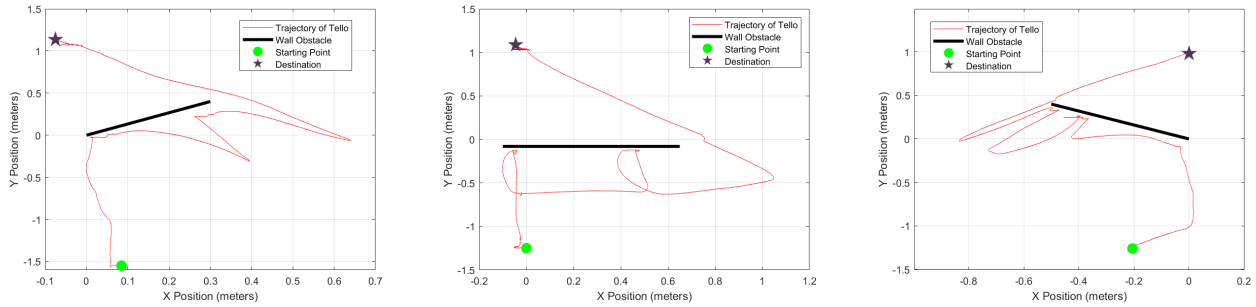
To help analyze the behavior of the UAV during a collision, we created a Matlab script that provides playback from different point of view angles as well as slow motion capabilities of MOCAP data in a fully simulated environment. The code of the simulator can be found at https://github.com/UVA-BezzoRobotics-AMRLab/Fly-Crash-Recover.

B. Protected Quadrotor Case Study

Our first set of experiments considered a protected DJI Tello quadrotor colliding with a wall obstacle. Fig. 8 shows a sequence of snapshots for the implementation of our approach in which the quadrotor successfully detected and maneuvered away from the wall.

Fig. 8. A quadrotor detecting a collision and recovering from the crash site.

In Fig. 9, we show the trajectory of a Tello in flight and encountering a wall obstacle in different orientations. In these



(a) 45° Wall Obstacle (b) Flat Wall Obstacle (c) 135° Wall Obstacle

Fig. 9. Three experimental cases with the DJI Tello

examples, the Tello begins at the green dot around (0,-1.5)m with a goal point at (0,1)m, marked by the black star. Upon colliding with the obstacle, detection is triggered determining the angle of collision, and recovering by moving in a correct direction away from the obstacle. Fig. 9(b) shows the full trajectory of the quadrotor associated to the snapshots in Fig. 8, where the vehicles encountered and detected the collision with a flat wall and maneuvered around it toward the goal. Fig. 10 details the acceleration and commanded velocities data occurring during the flight in Fig. 9(a). The quadrotor begins its flight at $t = 6.5$s and reaches its first waypoint at (0,-1)m at $t = 10$s. This is noted by the slow decrease of the X and Y command velocities. It then continues its flight towards point (0,1)m. At $t = 15$s, a POC is detected by the magnitude of the linear acceleration in X and Y. Since the value exceeded the set threshold of 1 m/s^2, the quadrotor entered a recovery mode in which it sent a command velocity in the opposite direction of the obstacle based on the detected collision acceleration vector. At $t = 19$s, the quadrotor resumed normal flight. Another POC was encountered at $t = 23$s, characterized by the sharp increase in linear acceleration. Once again it entered into the recovery mode flying away from the detected obstacle. At $t = 26$s, the quadrotor returns to normal flight reaching eventually its intended goal at $t = 32$s.

The other case studies presented in Fig. 9 show similar behavior and demonstrate the flexibility of our approach when the obstacle is positioned in different orientations.

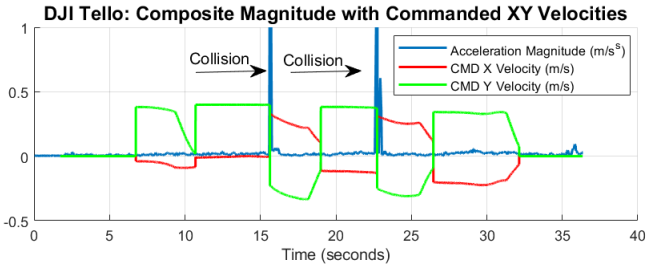


Fig. 10. Quadrotor flight data related to Fig. 9(a)

C. Unprotected Quadrotor Case Study

1) Detection: The second case study tests were performed on an unprotected Crazyflie quadrotor colliding with a pole. The detection algorithm was run on the high-level controller on the external computer at a rate of 100Hz. During the evaluation process of the recorded data, the response time of the algorithm was determined for each successful detection, along with the overall false positive rate over the total amount of flight time. For 16 experiments, the detection rate was 88%, and the average response time was $0.026s \pm 0.007s$ at a 95% confidence. The false positive rate during 8 minutes of flight time was 0.48 false positives per minute.

The response time of the algorithm was sufficiently quick and consistent to not significantly impact the time in which the vehicle was required to recover before crashing into the ground. In the two cases where the algorithm did not properly detect the collision, slight adjustments to the trigger threshold values for the axes would have adequately detected the event. One observation that impacted the results of these tests was the significant amount of noise that was present in the data from the X and Y axes of the IMU as compared to the Z dimension, which was far more stable. Upon further review of the observed false positives, each erroneous result was triggered by this noise present in the horizontal plane of the IMU. As a result, the false positive rate could likely be decreased by slightly raising the trigger thresholds for these axes.

Through further review of several of the collisions, the value of using the data from Z dimension along with X and Y dimensions of the IMU was proven. In multiple cases, the algorithm sensed a collision solely as a result of, or earlier because of, the acceleration experienced in the Z dimension. In the example collision shown in Fig. 11, the algorithm detected the collision 0.02 seconds earlier because of the additional dimension.

2) Recovery: Following the successful detection of a collision, the Crazyflie enters the stabilization phase of the recovery plan and attempts to achieve a stable hover as shown in Fig. 12. Fig. 13 provides an overhead view of a typical collision with an overlay of the phases of recovery.

Table I shows a number of key factors that contribute to the survivability of a collision using the recovery controller implemented on the Crazyflie platform. Factors with significance at the $p < 0.05$ level are highlighted in green, while those significant at $p < 0.2$ are highlighted in yellow. Of particular note is the strong correlation between low overshoot of the initial recovery maneuver and the increased survivability rate that is observed. This means better tuning of the non-linear initial recovery controller may help improve the success rate of the method, and also validates the theory that prioritizing stability is key to the overall recovery method. Another factor

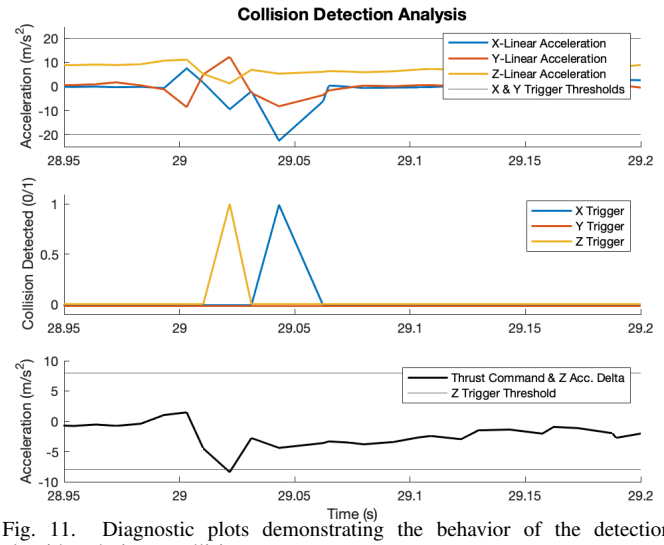


Fig. 11. Diagnostic plots demonstrating the behavior of the detection algorithm during a collision.

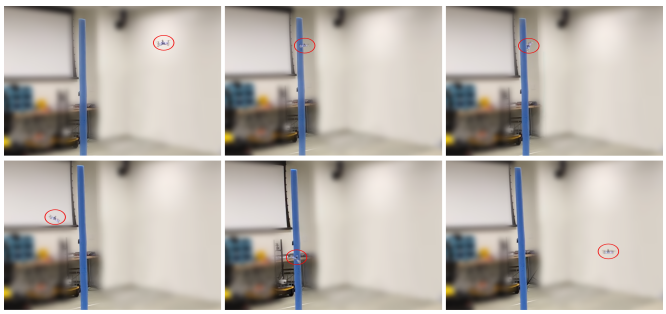


Fig. 12. Experiment for the recovery of the Crazyflie after colliding with a pole, showing the stabilization and replanning phases away from the POC.

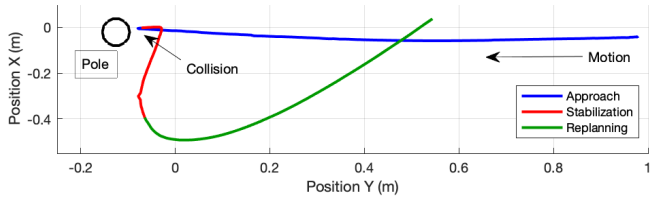


Fig. 13. Position of Crazyflie UAV and phases of the successful recovery presented in Fig.12

to note is the reliance on a low angular rate of the yaw dimension during the collision as well as the maximum yaw angle reached. One thing we noted during testing is that the low-level controller on-board the Crazyflie has trouble handling simultaneous maneuvers in both yaw and pitch or roll. This means that collisions that reduced the yaw angle allowed the controller to perform much better in stabilizing the roll and pitch to obtain a stable hover. This is a problem that may be overcome by re-tuning the low-level controller to ignore errors in yaw dimension during the initial recovery phase, however more testing is required to validate this assumption.

VI. Conclusion and Future Work

In this work, we have presented a framework for sensor-based reactive collision detection, estimation, and recovery of UAVs. Our approach focuses on portability and general UAV properties to create effective obstacle location recognition, impact recovery, and trajectory regeneration to allow completion of an assigned mission in a cluttered environment.

In future work we propose to investigate additional resiliency and recovery techniques to allow recovery in cases where the UAV sustains damages to the frame or motors. Additionally we plan to continue investigating potential field theory and trajectory generation techniques to better traverse and remap cluttered environments.

TABLE I
Analysis of recoverable vs non-recoverable collisions for the implemented controller on the Crazyflie.

	Average Value With Success	Average Value Without Success	P-Value
Linear Acc. During Crash- X (m/s^2)	21	25.5	0.338
Linear Acc. During Crash- Y (m/s^2)	34	32.5	0.446
Linear Acc. During Crash- Z (m/s^2)	11	11.5	0.456
Angular Rate During Crash- X (rad/s)	5.6	6	0.386
Angular Rate During Crash- Y (rad/s)	6.2	6.7	0.419
Angular Rate During Crash- Z (rad/s)	7	15.6	0.011
Angular Rate of Response- X (rad/s)	8.4	12.2	0.126
Angular Rate of Response- Y (rad/s)	5.6	8.2	0.230
Angular Rate of Response- Z (rad/s)	1.2	3	0.149
Max Angle During Crash- X (rad)	0.66	0.93	0.161
Max Angle During Crash- Y (rad)	0.46	0.57	0.263
Max Angle During Crash- Z (rad)	0.84	2.05	0.019
Max Angle of Response- X (rad)	0.37	2.7	0.003
Max Angle of Response- Y (rad)	0.41	0.88	0.032
Max Angle of Response- Z (rad)	0	2.7	0.001

VII. Acknowledgements

The authors thank the MITRE Corporation for sponsoring this research. We also thank Rahul Peddi and Shijie Gao for their technical assistance and guidance throughout this project.

References

[1] T. Tomić and S. Haddadin, "A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4197–4204.

[2] A. Briod, A. Klaptocz, J.-C. Zufferey, and D. Floreano, "The airburr: A flying robot that can exploit collisions," in *2012 ICME International Conference on Complex Medical Engineering (CME)*. IEEE, 2012, pp. 569–574.

[3] A. Briod, P. Kornatowski, J.-C. Zufferey, and D. Floreano, "A collision-resilient flying robot," *Journal of Field Robotics*, vol. 31, no. 4, pp. 496–509, 2014.

[4] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 45–52.

[5] G. Dicker, F. Chui, and I. Sharf, "Quadrotor collision characterization and recovery control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5830–5836.

[6] L. Beffa, A. Ledergerber, and R. D'Andrea, "State estimate recovery for autonomous quadcopters," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–7.

[7] F. Fei, Z. Tu, D. Xu, and X. Deng, "Learn-to-recover: Retrofitting uavs with reinforcement learning-assisted flight control under cyber-physical attacks," in *2020 IEEE International Conference on Robotics and Automation (ICRA) (submitted)*, 2020.

[8] J. Qiao, Z. Liu, and Y. Zhang, "Gain scheduling pid control of the quadrotor helicopter," in *2017 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE, 2017, pp. 1594–1601.

[9] N. Bezzo, Y. Yan, R. Fierro, and Y. Mostofi, "A decentralized connectivity strategy for mobile router swarms," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 4501–4506, 2011.