Rigid-Body Simulation-Based Probabilistic Analysis of Drone Crash Impact Points and Its Embedded System Application

Lee, Mu-Won[1]

[1] *Chungbuk Science High School, Cheongju, Chungbuk, 28189, Republic of Korea*
lmw.hpc@gmail.com

Abstract

This study utilized the C#-based **BepuPhysics2 rigid-body physics engine** and applied **isotropic drag** to the drag model to construct a **simulation environment** for creating a safety device that **predicts the impact point** in the event of a drone crash. Through this, the **probability distribution** was analyzed and developed into a **C library** applicable to embedded systems.

To ensure the versatility of the data, **altitude, horizontal speed, and vertical speed** were set as the main independent variables. An exhaustive search was conducted using a **Grid Sweep** method to generate foundational data for constructing a **LUT (Lookup Table)**.

Simultaneously, to reflect the uncertainty of the actual atmospheric environment, **wind speed, wind direction, and turbulence** were assigned as **Monte Carlo**-based random values. By performing repeated simulations, the **probability distribution** of the impact point under specific flight conditions was obtained. Through this, a **fall prediction C library** capable of accommodating various aircraft specifications was established.

1. Introduction

When a crash occurs, existing drones provide operators with **GPS data right up to the moment of signal loss**, and operators use this data to locate the drone. However, drones do not always **fall vertically downward**. Drones can crash in various locations due to the influence of **wind, previous speed, and direction**. Even if an operator attempts a search considering these factors, the lack of knowledge regarding where and with what probability the drone might be located can cause difficulties in a rapid search. If this environment is constructed through simulation and the drone's impact point is expressed in the form of a **probability distribution** to be provided to the operator upon a crash, it would be of great assistance in locating the drone. For this reason, this study was initiated with the idea that dropping a drone in a simulated environment, expressing the impact points as a probability distribution, and developing it into a **C library** would enhance convenience when applied to **drone firmware development**.

2. Related Work

As the operation of Unmanned Aerial Vehicles (UAVs) rapidly expands into commercial and civilian sectors, **ground impact** caused by system failures during flight is emerging as a major factor threatening third-party safety. Dalamagkidis (2008) systematically outlined the necessity of safety assessments for integrating UAVs into the national airspace, which became the starting point for subsequent research on UAV crash risk assessment.

Research aiming to quantitatively derive the **probability distribution** of crash impact points was initiated in earnest by La Cour-Harbo (2020). This study analytically derived the probability density function of the ground impact point in a ballistic descent situation by combining a **second-order drag model** with probabilistic assumptions of flight parameters. It is characterized by considering both the influence of wind and the uncertainty of the drag coefficient. In a subsequent study by the same author (La Cour-Harbo, 2019), this was expanded into a model that quantifies the **fatality rate** per unit of flight time based on the derived probability distribution.

Liu et al. (2021) proposed an approach that explicitly deals with **multi-uncertainty**. Fluctuations in the drag coefficient, velocity at the time of the crash, and local wind effects were all treated as random variables, and by combining them, the distribution of the ground impact point was estimated.

As for applied research in urban environments, the **ground risk map** study by Primatesta et al. (2020) is representative. This study proposed a method to generate a 2D risk map by combining the impact distribution model proposed by La Cour-Harbo (2020) with population density, sheltering factors, and no-fly zones in urban areas, and utilizing this for **path planning**.

Standard methodologies for **6-Degrees of Freedom (6DOF) rigid-body dynamics** simulations of UAVs are systematically organized in the book by Beard and McLain (2012), and the simulator implementation in this study is based on the equations of motion and coordinate system definitions presented in this literature. As an atmospheric turbulence model, the **Dryden model** is widely used in the field of aviation simulation and is standardized in the U.S. Military Handbook MIL-HDBK-1797 (U.S. Department of Defense, 1997). Hakim and Arifianto (2018) presented a case of implementing the Dryden continuous turbulence model in a simulation environment.

UAV crash analysis based on **Monte Carlo simulations** was first conducted by Rudnick-Cohen et al. (2016). This study calculated the statistical characteristics of the crash probability distribution by applying probabilistic inputs to a deterministic dynamics model and repeating multiple trials.

The **differentiating factors of this study** are as follows. While previous studies mainly adopted a method of deriving the prob-

ability density function of the impact point by **directly applying probabilistic assumptions to an analytical drag model**, this study **combines 6DOF rigid-body physics simulations with the Monte Carlo method** so that the **empirical distribution emerges from the data** solely through deterministic equations of motion and stochastic disturbance inputs, without prior assumptions about the distribution. This is based on the statistical convergence guaranteed by the **Law of Large Numbers**. Additionally, porting the derived distribution characteristics into a **lightweight C library** to **estimate crash risk zones in real-time on an embedded flight controller** is another point that differentiates this study from previous research.

3. System Design

This system consists of two main pillars: a **PC simulator** and an **embedded C library**. The PC simulator generates impact point data by repeatedly simulating drone crashes under various flight conditions, and the C library embeds this data in the form of a LUT to utilize it for real-time prediction.

3.1. Drone Model Abstraction

The drone is abstracted into three physical parameters: **mass**, **drag coefficient** (C_d), and **reference area** (A). By specifying only these three values, the same simulator can be applied regardless of the aircraft's shape or size, ensuring **versatility**.

The flight state at the time of the crash is represented as **FlightState**, with **altitude** (h), **horizontal speed** (v_h), and **vertical speed** (v_v), which can be directly acquired via GPS/IMU, set as the main independent variables. Considering that attitude and angular velocity right before a crash are difficult to specify, and their effects are averaged out during the Monte Carlo iterations, they were fixed to **Identity** and **Zero** vector, respectively.

3.2. Environment Model

The environment model consists of **atmospheric conditions** and a **wind field**. For air density ρ, the sea-level standard value of $1.225\ \mathrm{kg/m^3}$ was applied.

Wind is expressed as the sum of a steady wind component defined by **average wind speed** and **wind direction**, and a stochastic turbulence component generated by the **Dryden turbulence model**. In each Monte Carlo trial, wind speed is independently sampled from a uniform distribution of $[0,\ v_{max}]$, and wind direction from a uniform distribution of $[0°,\ 360°]$ to reflect the uncertainty of the atmospheric environment.

The coordinate system uses domain coordinates (X = East, Y = North, Z = Up), and the transformation to/from the BepuPhysics2 internal coordinates (X = East, Y = Up, Z = North) is handled in the Physics layer.

3.3. Crash Scenario

The crash scenario assumes **total thrust loss**. From the moment the crash is detected, the motor thrust is immediately set to 0, and thereafter the drone falls under the influence of only gravity and air resistance.

The flight state space is exhaustively searched using a **Grid Sweep** method. The ranges for altitude, horizontal speed, and vertical speed are each divided into a grid, and for each grid combination, **983** Monte Carlo simulations are performed to estimate the impact point distribution. The results of each trial are saved as a **CSV file**, which is later used to construct the LUT for the C library.

4. Physics Simulation

4.1. Rigid-Body Dynamics

The physics simulation utilizes the rigid-body integrator of **BepuPhysics2 v2.4.0**. The drone is modeled as a rigid body with mass and an inertia tensor, and external forces are summed at each time step to numerically integrate the **6-DOF** equations of motion.

The simulation terminates the moment the drone reaches the ground $(Z = 0)$, and the (X, Y) coordinates at this point are recorded as the **crash point**. The crash point is expressed in relative coordinates with the crash start location as the origin.

4.2. Aerodynamic Model

Drag is calculated as shown in Equation (1).

$$\mathbf{F}_d = -\frac{1}{2}\rho C_d A\,|\mathbf{v}_{rel}|\,\mathbf{v}_{rel} \tag{1}$$

Here, ρ is the air density, C_d is the drag coefficient, A is the reference area, and $\mathbf{v}_{rel}$ is the **relative velocity**, which is the difference between the drone's velocity and the wind velocity. The direction of the drag is opposite to the relative velocity, and its magnitude is proportional to the square of the relative velocity.

In this study, an **isotropic drag model** is applied. This assumes that the drag coefficient C_d and reference area A are constant regardless of the drone's attitude, based on the assumption that differences in drag by direction due to irregular attitude changes of the drone during a crash are averaged out during the Monte Carlo iterations.

4.3. Turbulence Model

Atmospheric turbulence is implemented based on the **Dryden turbulence model** (U.S. Department of Defense, 1997). The Dryden model generates a time-domain turbulence velocity signal by passing band-limited **white noise** through a first-order IIR **forming filter**.

The 3-axis components of turbulence $(u,\ v,\ w)$ are generated independently, and to eliminate correlations between trials, an **independent random seed** is used for each Monte Carlo trial. The generated turbulence velocity is added to the steady wind vector at each time step to constitute the **instantaneous wind velocity** acting on the drone.

5. Experiments and Results

5.1. Simulation Setup

The Grid Sweep consists of combinations of **8 altitude levels** (10.4 120.3 m) and **8 cruise speed levels** (0.0 20.0 m/s), and for each grid combination, **approximately 983 Monte Carlo trials** were performed. The entire dataset consists of **62,912**

trials. The drone parameters used in the simulation are shown in Table 1.

Table 1. Simulation Parameters.

Parameter	Value
Drone Mass	0.6 kg
Drag Coefficient C_d	1.1
Reference Area A	0.03 m^2
Altitude Range	10.4 120.3 m (8 levels)
Cruise Speed Range	0.0 20.0 m/s (8 levels)
Number of Trials	Approx. 983 trials/combination
Total Trials	62,912 trials

5.2. Impact Point Distribution

The main statistics for the **fall distance distribution** across all trials are as follows. The overall average fall distance is **20.33 m**, with a standard deviation of 15.0 m, and the **95th percentile (p95) is 49.02 m**.

The effect of **altitude** on the fall distance is summarized in Table 2. As altitude increases, both the fall distance and the p95 safety radius tend to monotonically increase. This is because the longer the fall time, the more the effects of wind and turbulence accumulate.

Table 2. Fall Distance Statistics by Altitude.

Altitude (m)	Mean (m)	p95 (m)
10.4	8.76	20.88
26.1	13.37	31.25
41.8	16.81	38.53
57.5	19.72	45.20
73.2	22.00	48.77
88.9	24.67	54.10
104.6	27.27	58.30
120.3	30.03	62.80

The effect of **cruise speed** is also evident. The average fall distance at a cruise speed of 0 m/s is only 9.73 m, but at 20 m/s, it increases to 33.93 m, an increase of approximately **3.5 times**. The p95 safety radius also expands from 30.38 m to 65.70 m. This indicates that the horizontal momentum at the time of the crash has a dominant effect on the impact point distribution.

5.3. Verification of the Law of Large Numbers

According to the **Law of Large Numbers**, the sample mean of independent trials converges to the population mean as the number of trials N increases. To verify this, the cumulative sample mean was observed while incrementally increasing N under the conditions of **altitude 57.5 m and cruise speed 8.57 m/s** (983 trials) (Table 3).

At $N = 10$, the sample mean is 17.727 m with high variance, but as N increases, it shows a tendency to **converge around 17.26 m**. For $N \geq 400$, the fluctuation range of the sample mean stabilizes within 0.2 m, confirming the **statistical convergence** of the Monte Carlo simulation. This implies that approximately 400 trials per grid combination are sufficient for reliable distribution estimation.

Table 3. Convergence of Sample Mean by Number of Trials N (Altitude 57.5 m, Cruise Speed 8.57 m/s).

N	Sample Mean (m)	Sample Std. Dev. (m)
10	17.727	9.373
30	16.706	6.918
50	16.381	7.141
100	16.893	7.072
200	17.486	7.475
400	17.421	7.374
983	17.264	7.344

6. EMBEDDED SYSTEM APPLICATION

6.1. Empirical Regression Model

To utilize the impact point distribution derived from the simulation results in real-time on an embedded system, a **3D Lookup Table (LUT)** approach was adopted. The LUT consists of a 3D grid with axes for **altitude** (h), **cruise speed** (v_c), and **vertical speed** (v_v), and each grid point stores distribution statistics such as **mean fall distance**, **standard deviation**, and **p95 safety radius** for that specific flight state.

The LUT-based approach does not require real-time computation of complex numerical integration or probability models, making it suitable for **Microcontroller Units (MCUs)** with limited computational resources. Values between grid points are estimated using **trilinear interpolation**, providing continuous predictions even for input values finer than the grid resolution.

Trilinear interpolation sequentially applies linear interpolation in three axial directions and calculates a predicted value as a weighted average of eight adjacent grid points for any given input (h, v_c, v_v). Since the prediction function is activated simply by **loading the CSV file once** without any additional learning or computation, it can be immediately applied to various aircraft specifications just by replacing sweep.csv.

6.2. C Library Design

The embedded C library (`drone_drop.h` / `drone_drop.c`) complies with the **C99** standard and has no external dependencies other than linking `-lm`. The internal memory solely uses **static allocation** and does not use heap allocation at all, allowing it to operate safely in both RTOS environments and bare-metal firmware. The memory footprint for the LUT data is approximately **117 KB BSS**.

The public API of the library is as follows.

```
DroneDrop_Status drone_drop_init(const char
    *csv_path);

DroneDrop_Status drone_drop_predict(
   float altitude_m,
   float h_speed_mps,
   float v_speed_mps,
   DroneDrop_Prediction *out);

void drone_drop_free(void);
int drone_drop_lut_size(void);
```

`drone_drop_init()` is called once during boot to parse the CSV sequentially and build the LUT. Afterwards, passing the **altitude, horizontal speed, and vertical speed** acquired

via GPS/IMU to `drone_drop_predict()` returns the prediction results in the `DroneDrop_Prediction` struct. The main fields of the output struct are shown in Table 4.

Table 4. `DroneDrop_Prediction` Output Fields.

Field	Description
`mean_x_m`	Forward mean offset (m)
`mean_y_m`	Lateral mean offset (m)
`std_x_m`	Forward standard deviation (m)
`std_y_m`	Lateral standard deviation (m)
`mean_distance_m`	Mean fall distance (m)
`p95_distance_m`	95% safety radius (m)

The **95% safety radius** (`p95_distance_m`) is calculated as follows.

$$r_{95} = \mu_d + 1.6449 \cdot \sigma_d \tag{2}$$

Here, μ_d is the mean fall distance, and σ_d is the standard deviation of the fall distance. The coefficient 1.6449 is the z-score corresponding to the one-sided 95th percentile of a normal distribution.

6.3. MCU Application

Because this library complies with the **C99** standard and has no external dependencies, it operates on **any MCU** that supports a C99 compiler. As long as a means to read files (FatFS, SPIFFS, LittleFS, etc.) is secured, it can be integrated regardless of the platform.

The target hardware assumed in this study is the **STM32F405**. A **u-blox NEO-M9N** is used as the GPS receiver, and the 3-axis velocity (`velN`, `velE`, `velD`) is directly acquired through the **UBX-NAV-VELNED** message. Horizontal speed is calculated as the composite of the North and East velocities, and vertical speed uses the downward component. It is assumed that the LUT data file (`sweep.csv`) is stored on an **SPI Micro SD** card and accessed via **FatFS**.

The firmware integration flow is as follows. Upon boot, `drone_drop_init()` is called once to load the LUT. If a crash is detected during flight, the current altitude, horizontal speed, and vertical speed are read from the GPS to call `drone_drop_predict()`. The returned `p95_distance_m` is provided to the operator as the **safety search radius**, and `mean_x_m` and `mean_y_m` as the **expected impact direction offsets**.

According to the simulation results of this study, under the conditions of an altitude of 57.5 m and a cruise speed of 8.57 m/s, the **p95 safety radius is estimated to be 30.73 m**, and the probability of the drone being within this radius is 95%. Utilizing this, the operator can set a **30.73 m radius from the last received GPS point** as the priority search area immediately following a crash.

7. CONCLUSION

In this study, by combining the **BepuPhysics2 rigid-body physics engine** and the **Monte Carlo method**, the probability distribution of drone crash impact points was derived based on simulations, and this was implemented as a **C library** that can be utilized in real-time on an embedded system.

The main results are summarized as follows. As altitude and cruise speed increase, the fall distance and p95 safety radius monotonically increase, and under the 20 m/s cruise speed condition, the average fall distance increased by approximately 3.5 times compared to a hover. Through an analysis of sample mean convergence according to the number of trials N, it was confirmed that **approximately 983 trials** per grid combination are sufficient for reliable distribution estimation.

The implemented C library has the characteristics of following the **C99** standard, using **static memory allocation**, and having **no heap usage**, making it integrable into any MCU that supports a C99 compiler. By receiving altitude, horizontal speed, and vertical speed from the GPS, estimating the impact point distribution through **trilinear interpolation**, and providing the **95% safety radius** to the operator, the efficiency of drone search after a crash can be enhanced.

Limitations of this study include applying an **isotropic assumption** to the drag model, failing to reflect differences in drag by direction due to drone attitude changes, and the fact that **field verification** through actual crash experiments was not performed. Future studies will introduce an attitude-dependent drag model and quantitatively evaluate the accuracy of the simulation model through comparisons with measured GPS trajectory data.

REFERENCES

Beard, R. W., & McLain, T. W. (2012). *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press.

Dalamagkidis, K., Valavanis, K. P., & Piegl, L. A. (2008). *On Integrating Unmanned Aircraft Systems into the National Airspace System*. Springer.

Hakim, T. M. I., & Arifianto, O. (2018). Implementation of Dryden Continuous Turbulence Model into Simulink for LSA-02 Flight Test Simulation. *Journal of Physics: Conference Series*, 1005, 012017.

La Cour-Harbo, A. (2019). Quantifying Risk of Ground Impact Fatalities for Small Unmanned Aircraft. *Journal of Intelligent & Robotic Systems*, 93(1–2), 367–384.

La Cour-Harbo, A. (2020). Ground impact probability distribution for small unmanned aircraft in ballistic descent. In *Proc. ICUAS 2020* (pp. 1442–1451). IEEE.

Liu, X., Li, X., & Zhang, J. (2021). Ground Risk Assessment of UAV Operations Based on Horizontal Distance Estimation under Uncertain Conditions. *Mathematical Problems in Engineering*, 2021, 3384870.

Primatesta, S., Rizzo, A., & La Cour-Harbo, A. (2020). Ground Risk Map for Unmanned Aircraft in Urban Environments. *Journal of Intelligent & Robotic Systems*, 97, 489–509.

Rudnick-Cohen, E., Herrmann, J. W., & Azarm, S. (2016). Risk-based path planning optimization methods for unmanned aerial vehicles over inhabited areas. *Journal of Computing and Information Science in Engineering*, 16(2).

Stevens, B. L., Lewis, F. L., & Johnson, E. N. (2015). *Aircraft Control and Simulation* (3rd ed.). John Wiley & Sons.

U.S. Department of Defense. (1997). *Flying Qualities of Piloted Aircraft* (MIL-HDBK-1797). DoD.