Universiteit Leiden
The Netherlands

Opleiding Informatica

Assessing flight and sensor accuracy of the Crazyflie 2.1

Aleksandr Petrov

Supervisors:
Dr. M.F.T. Müller-Brockhausen & Dr. M. Preuss

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl 15/12/2025

Abstract

Drones are becoming increasingly important in various areas due to their mobility, compact size, and ability of autonomous task execution, making reliable distance sensing and navigation essential. This work presents a quantitative analysis on hyperparameters to determine the key configuration for optimal accuracy of the nano-quadcopter Crazyflie 2.1, emphasizing the Crazyflie Python API and Multi-ranger deck. This is achieved through identifying and tuning hyperparameters that affect accuracy.

The quantitative approach consists of controlled flights, during which the sensor readings from each individual drone flight were collected, along with manually measured reference distances, and stored into multiple .csv datasets.

From the results, a 95.16% and a 95.95% route precision accuracy were achieved for complex and simple routes, respectively. Meanwhile, minimal sensor deviation of 2.43% and 1.32% is achieved in distance sensing.

Contents

1 Introduction **1**
1.1 Thesis overview 1

2 Related Work **1**

3 Methodology **2**
3.1 Tools and materials 2
3.2 Software setup 3
3.3 Variable selection 3
3.4 Constants 3
3.5 Experiment design 4
3.6 Metrics 4
3.6.1 Sensor deviation 4
3.6.2 Accuracy 4
3.7 Mitigation of measurement bias 5

4 Experiments **5**
4.1 Data collection 5
4.1.1 Tested variables and conditions 5
4.1.2 Procedure 6
4.2 Metric extraction 6

5 Results **7**
5.1 Accuracy 7
5.2 Sensor deviation 7

6 Conclusions **9**

7 Discussion 10

8 Further research 10

References 11

1 Introduction

In the last decade, multidisciplinary fields, such as the construction industry, have experienced an increase in the dependence on drone technology. As stated by Choi et al. [CKKN23], it is a crucial aspect to optimize the workflow, improve the coordination, and mitigate the risks regarding dangerous aspects of the construction. Utilization of drones is relevant in 3D modeling of the buildings, and it contributes to precise mapping [CKKN23]. In these fields, particularly the route and sensor accuracy play a crucial role. Whether the drone can steadily fly around a building and log precise distances has an immense impact on the result of the building's 3D scan. Besides the importance of the drone utilization in various fields, it is important to consider the category of drone used in the mentioned tasks. The Crazyflie 2.1 is a nano-quadcopter that provides low cost and reduced size, introducing flexibility both in maneuvering and replacibility [ZZS+24].

This work explores the limitations of such a drone and its modules. Although some studies have examined the accuracy [CS20][VW24], few have brought up the quantitative analysis of the Crazyflie API, where accuracy was broadly examined by having the drone perform precise maneuvering and sensor reading. In order to address this gap, this work investigates the control reliability of the translation from API control commands to real-world flight behavior.

- **How accurate is the Crazyflie API in controlling the drone's behavior during the execution of predefined maneuvers?**

To that end, an investigation into the drone's hyperparameters will be conducted, resulting in the following sub-question:

- **Which hyperparameters most significantly affect the Crazyflie's accuracy when following a predefined route?**

By looking into various hyperparameter configurations, it will become clear what really affects and even bottlenecks the Crazyflie's accuracy in the execution of predefined routes, and therefore the limitations of the Crazyflie Platform will become clear.

1.1 Thesis overview

This work, under the supervision of Dr. M.F.T. Müller-Brockhausen & Dr. M. Preuss at The Leiden Institute of Advanced Computer Science (LIACS), will elaborate on the refined process of this work in the next chapters. Section 2 addresses the related work in the field of drones. Section 3 discusses the system of methods used for the experiment. Section 3.6 explains the used formulas involved in metric computation. Section 4 elaborates on the process of data collection and metric computation. Section 5 discusses the findings, and section 6 clarifies why the found results exhibit a particular structure. Section 7 talks about the encountered limitations, and section 8 looks into the future for possible further work.

2 Related Work

Recent studies have explored and examined various aspects of the Crazyflie 2.1 platform, including obstacle detection as well as pattern and gesture recognition. Chadehumbe et al. (2020) [CS20] have investigated drone obstacle detection and avoidance in combination with the shortest-path navigation. Their work resulted in Flow deck[1] showing the best performance. However, the study solely focused on flight control improvement rather than accuracy assessment. The authors used the Multi-ranger deck[2] for measurement correction, a component also employed in this work.

In another study by vanHorn et al. (2024) [VW24], the field of machine learning for wind pattern recognition is examined, where IMU (Inertial Measurement Unit) is utilized. The study resulted in a successful wind pattern recognition. Similarly, Zauli et al. [ZZS+24] offer an insight into the limitation of the IMU sensor by looking at its capabilities of frequency detection. The vibration measurements were used in addressing a pseudo-aging infrastructure. This resulted in less than one percent deviation from the reference sensors. However, these studies focused primarily on the IMU sensor readings, whereas this work replaced it with the distance readings from the Multi-ranger deck.

In the most recent study, Tsie-A-Foeng (2025) [TAF25] introduced hand gesture and voice recognition by integrating the MediaPipe & Hand Landmarks library with the KNN classifier and the VOSK speech recognition toolkit coupled with techniques such as cosine similarity, respectively. The results showed successful recognition in both approaches but primarily focused on the accuracy in the context of hand gesture recognition rather than quantitative accuracy.

Although these studies provided a valuable insight into how the Crazyflie performs with various positioning systems, showing the possible applications in the field of machine learning, few studies examine the drone's movement precision, and no concrete data on the accuracy is available solely under the influence of the Multi-ranger deck. In all other instances, assistance from modules such as the loco position deck is incorporated, with the example of Multi Agent RL by Felten F. [Fel24], additionally utilizing the lower-level commander, whereas this study employs high-level commander instructions. In case accuracy is mentioned, it is either treated as a secondary result or examined from a different application-specific perspective. In contrast to these works, explicit quantification of the drone's positional and sensor accuracy is applied, varying the hyperparameters. The assessment is carried out for defining the key configuration with the most influence on the measurement and movement precision of the CrazyFlie 2.1.

3 Methodology

3.1 Tools and materials

For the dataset collection, the experiment has been provided with a Crazyflie 2.1 drone [Bita]. This choice was on account of the drone itself being small and therefore portable. In the matter of a breakdown, numerous replacements are available. The attachments used for it are the Multi-ranger deck [Bitd], 250 mAh LiPo battery, Crazyradio PA (2.0), micro-USB cable, and Flow deck v2 [Bitc]. For the collection of the reference measurements, a smartphone camera was sufficient (HD, 30 FPS).

All software operated on a Linux system, Ubuntu 22.04.5 LTS. The Crazyflie API ran on Visual Studio Code. The custom scripts made for data processing were `accuracy.py`, `sensor_deviation.py` and `minmax.py`.

[1]Attachable VL53L1x ToF sensor enables movement measurement in relation to the ground [Bitc]

[2]Enables the Crazyflie to detect object distances relative to itself [Bitd]

3.2 Software setup

For initiating Crazyflie's takeoff from the ground, it is essential to access all the necessary functionalities, available in the Crazyflie API, cflib [Bitb]. Consequently, the establishment of communication between Crazyflie and the cflib must first be configured. The hardware setup followed instructions from the manufacturer.

The software is implemented in Python. With cflib, the CrazyRadio PA is connected to the computer via a Uniform Resource Identifier (`URI`), which defines the channel and bitrate. Consequently, Crazyflie's identification is performed. The configuration and initialization of the communication system is carried out via the `SyncCrazyflie` interface. Upon the establishment of connection, the drone is able to track the obstacle distance measurements for each individual direction (`range.left`, `range.right`, `range.front`, and `range.back`).

3.3 Variable selection

During the orientation phase, the identification of variables with the greatest influence on performance was required. The choice comprised the following variables: `STOP_DIST`, `ASCEND`, `DUR`, `BRAKE_DUR`, `STEP`, `LOG_PERIOD`, `LOG_MARGIN`. The first selected variable was the `STEP_DUR`. It was reasonable to assume that movement speed has a direct impact on accuracy, with the increase likely reducing stabilization and movement precision. Longer durations introduce stabilization time after each step, while shorter durations introduce motion deviation on account of increased speed. Therefore, it was selected as the first factor of influence.

The second parameter of interest initially considered was `STEP`, as a larger STEP might decrease the accuracy due to increased flight distance and greater difficulty in fetching accurate packages with the Multi-ranger deck. However, the choice was made to observe the influence of `LOG_PERIOD` instead, as the selected factor directly impacts the frequency of package logging and therefore the amount of present disturbance in distance measurement. Considering `LOG_PERIOD`, the performance of the Multi-ranger deck is taken into account, rather than only API-controlled movement via cflib-calls.

3.4 Constants

Several parameters were kept constant throughout the whole process of data collection. `STEP` is the single step length in meters along a specific trajectory, fixed at 0.4. The particular value was chosen, allowing the drone to fly a sufficient distance, with the Multi-ranger deck having a constant possibility of obstacle detection within its effective range. `ASCEND` is defined as the altitude in meters the drone should ascend to, preparatory to executing the passed route. The specific height was chosen due to a significant requirement in battery power for reaching a higher altitude, while a lower altitude increases the risk of ground collision. `STOP_DIST` is a threshold within which the drone should terminate its trajectory, kept at 0.3 meters. `BRAKE_DUR` is the braking time in seconds upon encountering an obstacle within `STOP_DIST` threshold, kept at 0.5 for a smooth speed reduction. After step completion, `LOG_MARGIN` of 1 second is used as extra time for package logging after moving up to the next trajectory. This compensates for forward drift during trajectory switch, which can cause inaccurate distance logging.

3.5 Experiment design

The implementation was set up out of a set of scripts, later merging into a single program, with the purpose of logging the relevant flight data of the drone during route execution. The predefined route is decomposed into individual components, executing each individual direction sequentially.

When following a trajectory, the drone collects the traveled distance from the Multi-ranger deck sensors. The readings are collected for evaluation of Crazyflie's internal component accuracy in a controlled indoor environment when depending on obstacle distances.

The results are written into a .csv file in the form of `Step,Direction,StartDist,TaperStart,FlownDist,TaperStop,Obstacle`. `Step` represents the index of the current trajectory, `TaperStart` and `TaperStop` imply manually measured distances from the video, and `Obstacle` represents the presence of an obstacle within the STOP_DIST threshold in the form of a boolean flag. The data of each individual flight is labeled with a number, referring to a recording of a particular route execution.

The systematic approach ensures the collection of all the necessary sensor data for each individual run.

3.6 Metrics

After the dataset completion, Python scripting is applied for automation of data processing. The main calculated metrics are the **movement accuracy** and **sensor deviation** of the Multi-ranger deck, derived from the collected data. The majority of the code of each individual script follows an identical structure, differing solely on the computation logic for the selected metric, which is summed up in several formulas.

3.6.1 Sensor deviation

For quantification of **sensor deviation**, first the `real_accuracy` (η) is computed, derived from division of `taper_est` (β) and `expected` (θ), represented by:

$$\eta = \frac{\min(\beta, \theta)}{\max(\beta, \theta)} \times 100 \tag{1}$$

where the `expected` implies the distance expected to be flown and `taper_est` is defined as the physically measured flown distance. Afterwards, the difference (δ) between the sensor precision (γ) and the `real_accuracy` is taken, specified as:

$$\delta = \texttt{abs}(\eta - \gamma). \tag{2}$$

All the differences are summed up and divided by the corresponding amount of entries, for single, double, and triple direction, being 8, 16, and 24, respectively.

3.6.2 Accuracy

The **accuracy** is resembled in a similar way, where we first compute the accuracy (formula 1) which subsequently is added to the average `avg`. Likewise, the avg is divided by the corresponding amount of entries for obtaining the result.

3.7 Mitigation of measurement bias

Several precautions were taken for measurement bias minimization during the data collection process. First of all, the obstacle material is kept constant throughout the whole process of data collection. Preliminary results exhibited a slight variety in sensor readings by the Multi-ranger deck when partly changing the reflective properties of the material. This might result in non-representative data, as the sensor deviation might be the result of reflection from inconsistency in material pick.

Moreover, a substantial number of flights should be excluded, not meeting the conditions of a successful performance. Though no track of total flight number was kept, a rough estimate of successful flights would be 10%. The tight route passage restricts the drone from deviating from the trajectory, where even a small divergence may result in a crash or biased readings. For example, a slight angle change from the pre-defined trajectory may be present, exceeding no more than 5 percent. If not meeting the condition, the sensor readings become unsuitable for the dataset, as they now introduce inflated distance readings. In case of a crash, any unsuccessful flight implies incomplete data collection.

Furthermore, for bias mitigation related to the flight performance, the battery was kept at a full charge across all flights. A lower battery percentage may reduce flight stability, increasing the likelihood of a crash or introducing additional swaying.

Lastly, the amount of lighting is strictly controlled. The Multi-ranger deck sensors perform optimally in an environment with diffused lighting. Therefore, direct sunlight is kept minimal, preventing floor reflection, as it can increase hover instability or invalidate sensor readings (32.77 m).

4 Experiments

4.1 Data collection

4.1.1 Tested variables and conditions

A quantitative type of research was conducted to evaluate hyperparameter influence. This design was chosen to obtain measurable data on comparison in flight accuracy and sensor deviation. The chosen independent variables used for the influence of the drone behavior are `LOG_PERIOD`, which represents the package logging frequency of the Multi-ranger deck, and `DUR` which represents the single step duration (see section 3.3). The reasoning behind the choice is to investigate whether varying them has any effect on the sensor logging behavior and the movement precision during flying. The dependent variables are **accuracy** and **sensor deviation**.

The `LOG_PERIOD` was interchanged between **50** and **70 ms**, and the DUR value was set to **2.5**, **1.5** and **0.5 s**. Each unique combination of `DUR` and `LOG_PERIOD` was used in 24 flights across 3 different route complexities, 8 flights for one complexity each. A few flights would cause outliers to have a larger influence on the whole dataset, while doing too many repetitions becomes impractical when taking the timeline into account. A total of 6 unique configurations were tested, resulting in a total of 144 flights. This way, sufficient repetition was ensured to mitigate the effect of any outlier while keeping the total workload manageable.

To ensure a controlled environment, natural lighting is used, and obstacle surfaces were static and indoor. This work does not capture any external factors like sun reflection or wind turbulence for the purpose of bias mitigation (see section 3.7) and to see how different combinations of `STEP_DUR` and `LOG_PERIOD` exclusively would influence the performance of Multi-ranger Deck and flight precision.

4.1.2 Procedure

The collection of the data consists of several steps, including three routes of increasing complexity. For each subsequent route, an additional direction is added to the previous one for the increase of complexity. This enables the systematical assessment of how route complexity influences movement precision and sensor deviation.

After every flight, the collected data is stored into a `.csv` file. For obtaining reference values, the data gathering implied recording every individual run with a camera while the drone collected the sensor readings along its trajectory, using a smartphone camera (see Section 3.1). The measurement of the reference values is done stepwise, only measuring the distance upon a full velocity stop, completing a full step. To ensure consistency, each measurement is taken exactly at the position of the corresponding sensor, which is 1.5 cm away from the center of the drone. (see Figure 1). Afterwards, the dataset is finalized by incorporating manually measured reference values from the corresponding recording.

Following completion of data collection, automated data processing is applied, exporting the results into a `.txt` file. The text files contain computed sensor deviation (see Section 3.6.1) and movement precision (see Section 3.6.2).

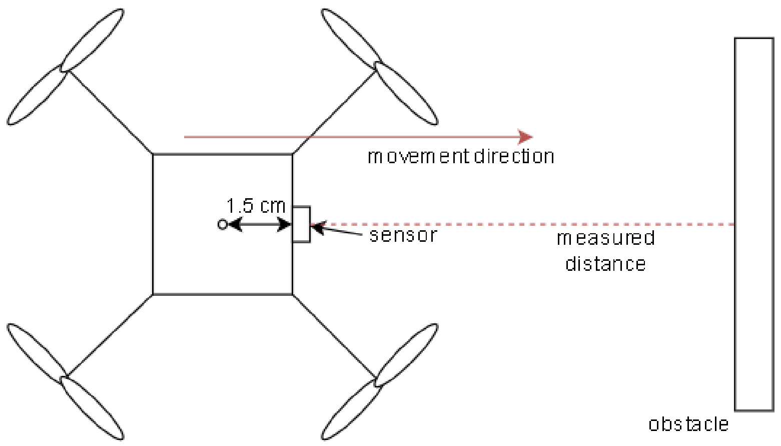

Figure 1: Measurement of reference distance

4.2 Metric extraction

The data processing involves calculation of **accuracy**, **average-** and **minimum-maximum sensor deviation**, previously explained in section 3.6. The data used for these calculations consists of the collected sensor readings and reference measurements stored across multiple `.csv` datasets.

The accuracy presents how close the traveled drone distance is to the expected distance. Sensor deviation is expressed as the deviation from the manually measured distances, while minimum-maximum implies the same deviation but taking into account the boundary values.

No data cleaning was required, as the dataset already had a fixed structure. Only invalid or incomplete readings had to be excluded from the data, as they could not be utilized for the objectives of this work.

Data visualization was performed using bar graphs (e.g., Figure 2), proving to be the most efficient way of representing accuracy, sensor deviation, and min/max values. This approach fitted the research design, as it enabled direct comparison between individual hyperparameter configurations, making recognition of emerging patterns trivial.

5 Results

5.1 Accuracy

Figure 2 illustrates the Crazyflie's accuracy in traveled distance under varying `STEP_DUR` values for all three route complexities. A specific pattern can be noted. With a shorter `STEP_DUR` the deviation from the trajectory becomes greater, decreasing the precision in movement. This specifically stands out for more complex routes.

For `single` and `double direction`, the trajectory accuracy is at its peak for `STEP_DUR` equivalent to 1.5. When comparing `STEP_DUR` of 2.5 and 0.5, with a shorter duration, the drone deviates more from its trajectory. In contrast, a continuous decrease in movement precision is present for `triple direction`, with a decreasing `STEP_DUR`.

Overall, the results suggest an optimal `STEP_DUR` of 1.5 seconds for trivial routes, pointing out the balance between stability and precision. Meanwhile, complex routes exhibit a high sensitivity to `STEP_DUR` and show an optimal accuracy with the longest duration.

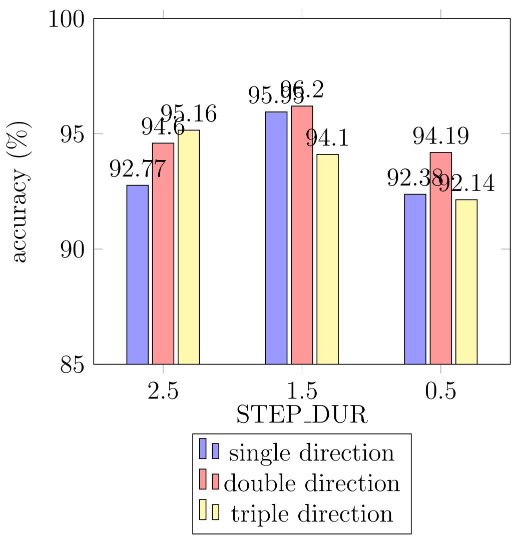

Figure 2: Accuracy based on travelled distance

5.2 Sensor deviation

Figure 3 introduces the sensor deviation from the manually measured reference distances, recorded at a `LOG_PERIOD` of 50 ms. Each color represents a specific route complexity, corresponding with the colors from figure 2. The results from the left graph (average deviation) indicate that with the decreasing `STEP_DUR` the sensor deviation increases for each individual route. While the increase for `single direction` remains minimal, the most intricate combination (`triple direction` with shortest `STEP_DUR`) introduces measurement error, exhibiting an amplification in the sensor deviation with the additional complexity.

For the minimum and maximum sensor deviation shown in the right graph, a similar pattern can be observed. With the increasing difficulty, represented by the complexity of the path and a decreasing `STEP_DUR`, the spread between minimum and maximum deviation increases with it. This indicates a decrease in stabilization for shorter `STEP_DUR`, especially with complicated routes.

In general, with a `LOG_PERIOD` of 50 ms, both `STEP_DUR` and the path difficulty increase have a strong influence on the Multi-ranger deck's ability to log accurate packages, providing considerably less reliable results for shorter durations.

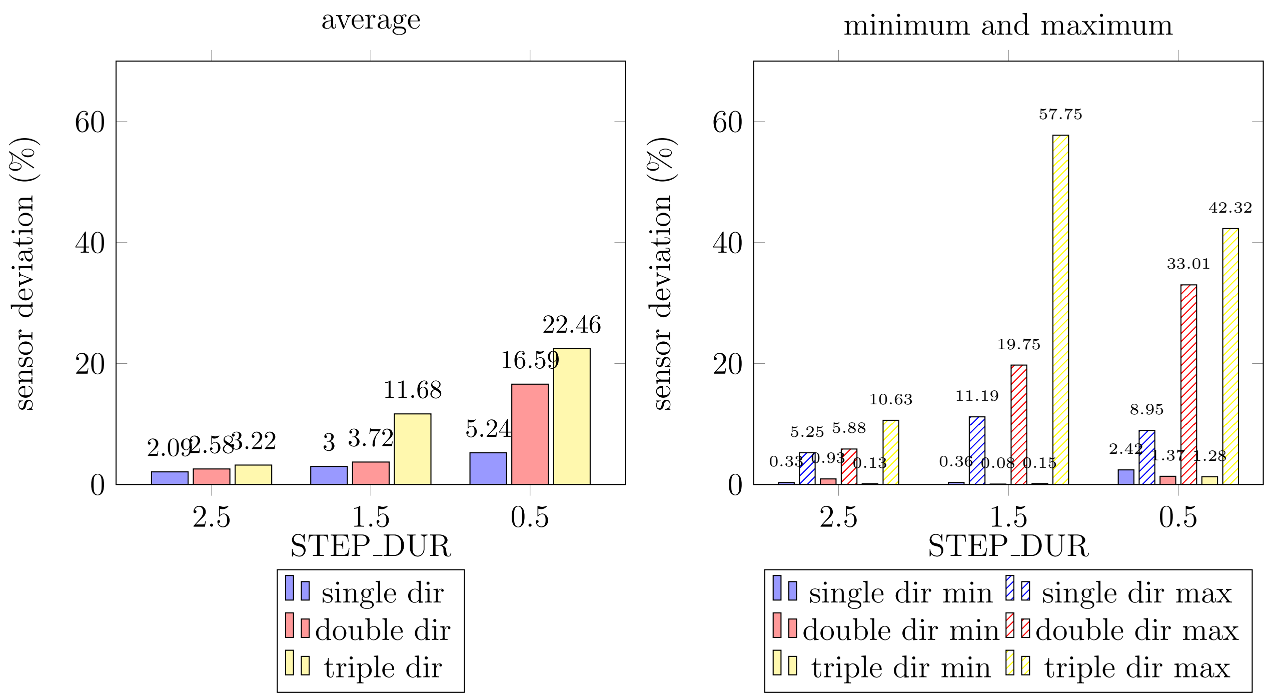

Figure 3: Sensor deviation from real perception (50 ms)

Figure 4 illustrates similar graphs, using the `LOG_PERIOD` of 70 ms. In contrast to the `LOG_PERIOD` of 50 ms, sensor deviation remains at its minimum for all durations and routes combined. Remarkably, the deviation becomes independent of route complexity, with `triple direction` no longer having an amplifying effect on the sensor deviation.

The same trend now also holds for the minimum and maximum deviation, which remains approximately the same for every combination of route and STEP_DUR. This indicates the increase of `LOG_PERIOD` to 70 ms having a stabilizing effect on the sensor accuracy of the Crazyflie.

Altogether, the results show that an alteration in `LOG_PERIOD` has a major influence on the sensor readings. The Multi-ranger deck operates most stable at a `LOG_PERIOD` of 70 ms, introducing minimal sensor deviation.

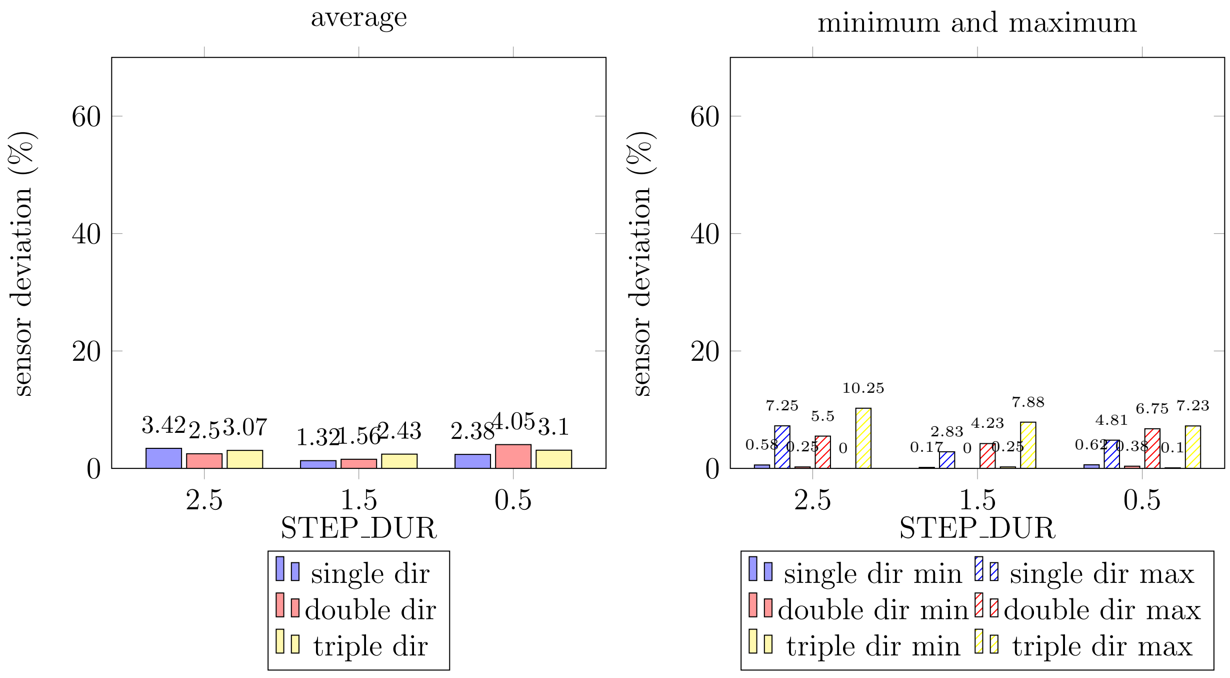

Figure 4: Sensor deviation from real perception (70 ms)

6 Conclusions

The goal of this study was the investigation of factors with the strongest influence on the accuracy of the Crazyflie. Using various configurations, the results demonstrated that `STEP_DUR` and `LOG_PERIOD` proved to be primary elements of influence on Crazyflie's movement precision and sensor deviation.

The results provide solid quantitative insight about accuracy and sensor deviation. For the API accuracy, trivial routes achieved the best performance of 95.95% at moderately short step durations of 1.5 s, due to the reduced time interval available for hover. For these routes stabilization time is redundant, solely causing an increased chance for route deviation and therefore an accuracy decrease. Meanwhile, for the complex flights, a 95.16% accuracy in movement was achieved under slow durations of 2.5 s due to requiring more stabilization during the alternation in trajectory to execute the route.

The sensor deviation primarily experienced an influence from `STEP_DUR` on the condition that the `LOG_PERIOD` was not optimal. It would only increase with the shorter durations and increasing route complexity due to the amount of shaking, reaching an average deviation of 22.46% and an outlier of 57.75%. This relates to the excessive data logging and lack of stabilization time, which results in a higher chance of incorrect distance interpretation. With longer logging intervals of 70 ms, the amount of excessive package overflowing is reduced, lowering the chance of fetching a stale package, leading to less sensor deviation. Hence, `STEP_DUR` exhibits a negligible influence on the sensor accuracy, which results in an average peak of 3.1% deviation for the highest complexity route, a reduction of 19.3%.

Overall, the work provides a sufficient insight into how various hyperparameter configurations affect the drone behavior with respect to accuracy in trajectory execution and sensor logging, demonstrating the best performance of 96.2%.

7 Discussion

During evaluation of multiple hyperparameter configurations, the drone was tested under various conditions to deduce determinants affecting the accuracy. Despite the significant findings, multiple factors ranging from hardware limitations to environmental obstacles had a considerable influence on data collection, posing challenges for mitigation.

Regarding the hardware, three Crazyflie units became unusable during the early stages of data collection. The breakdowns ranged from a broken motor mount to an unstable battery connection preventing the Crazyflie from taking off. Only with the fourth unit did the data collection proceed without major interruptions.

With respect to environmental limitations, lighting conditions could occasionally introduce reflection on the floor, influencing the Flow-deck's behavior. Consequently, a periodic drift away from the intended trajectory or rotation by 5-20 degrees during the takeoff would occur. The sensors of the Multi-ranger deck were also sensitive to the direct sunlight, reducing the maximum measurable distance, resulting in an invalid read (32.77 m). Despite having incorporated optimal circumstances—such as properly functioning parts, a battery level above 85%, and a stable environment—the unit obtained a flight completion of roughly 7-8 out of 10 flights. An earlier unit demonstrated a success rate of 0 out of 10 flights due to an unstable battery connection.

These limitations may have introduced minor instability during the data collection phase and, therefore, potentially have influenced the Crazyflie precision in movement and sensor accuracy.

8 Further research

To expand on this work, further research could investigate how hyperparameters influence drone behavior under various environmental conditions. In this work, the drone was tested under controlled conditions, such as lighting and obstacles held constant. These factors can serve as variables to assess the Crazyflie's Multi-ranger deck performance across various lighting and reflective surfaces.

Furthermore, this work can be extended by introducing additional layers of environmental complexity, such as simulated wind turbulence. The detection and avoidance of wind turbulence was previously implemented by van Horn et al. [VW24]. By integrating the approach of van Horn, the precision in path following under the influence of varying wind patterns can quantitatively be investigated, assessing the Crazyflie's true ability of environmental adaptation.

Finally, the Crazyflie's behavior during the braking phase upon entering `STOP_DIST` could be examined. Tadevosyan et al. have researched drone swarms racing in dynamic environments, with time-based obstacle positions [TSF+25]. The dynamic environment adaptation can be combined with our work, for looking further into the Crazyflie's response to static versus dynamic obstacles, enabling quantitative examination of braking accuracy during the `STOP_DIST` phase.

References

[Bita] Bitcraze. Crazyflie 2.1. https://www.bitcraze.io/products/old-products/crazyflie-2-1/. Accessed: 08-11-2025.

[Bitb] Bitcraze. The crazyflie python api explanation. https://www.bitcraze.io/documentation/repository/crazyflie-lib-python/master/user-guides/python_api/. Accessed: 10-11-2025.

[Bitc] Bitcraze. Flow deck v2. https://www.bitcraze.io/products/flow-deck-v2/. Accessed: 4-12-2025.

[Bitd] Bitcraze. Multi-ranger deck. https://www.bitcraze.io/products/multi-ranger-deck/. Accessed: 08-11-2025.

[CKKN23] Hee-Wook Choi, Hyung-Jin Kim, Sung-Keun Kim, and Wongi S Na. An overview of drone applications in the construction industry. *Drones*, 7(8):515, 2023.

[CS20] Chiedza Chadehumbe and Josefine Sjöberg. Autonomous flight of the micro drone crazyflie 2.1 through an obstacle course, 2020.

[Fel24] Florian Felten. Multi-objective reinforcement learning. 2024.

[TAF25] Ernie Tsie-A-Foeng. Transforming the crazyflie 2.1 nano-drone into an interactive pet. 2025.

[TSF$^+$25] Grik Tadevosyan, Valerii Serpiva, Aleksey Fedoseev, Roohan Ahmed Khan, Demetros Aschu, Faryal Batool, Nickolay Efanov, Artem Mikhaylov, and Dzmitry Tsetserukou. Attentionswarm: Reinforcement learning with attention control barier function for crazyflie drones in dynamic environments. *arXiv preprint arXiv:2503.07376*, 2025.

[VW24] Kyle VanHorn and Artur Wolek. Machine-learning-based wind detection and avoidance using a crazyflie micro drone. In *2024 Regional Student Conferences*, page 85491, 2024.

[ZZS$^+$24] Matteo Zauli, Federica Zonzini, Giulio Sciullo, Alessandro Marzani, Luca De Marchi, et al. Evaluating unmanned aerial vehicles for vibrational analysis: Exploiting integrated capabilities of nano-drones as nodes in a wsn. In *11th European Workshop on Structural Health Monitoring (EWSHM 2024), June 10-13, 2024 in Potsdam, Germany*, pages 1–8. NDT. net, 2024.