Review of Reinforcement Learning and Error Models for Drone Precise Landing

Côme PÉRIN, Antoine FUSIER

Index Terms—Neural Networks, Reinforcement Learning, Applied Computing, Engineering, Algebraic Topology

1 Introduction

As the use of autonomous drones increases across sectors ranging from agriculture to package delivery, the challenges of ensuring safety for individuals and efficiency in services make autonomous landing a critical issue. This challenge has specifically been addressed in [1], "Using Reinforcement Learning and Error Models for Drone Precise Landing," published in July 2024 in the *ACM Digital Library*.

The safety of autonomous landings is hindered by two types of errors: intrinsic errors arising from factors such as sensor inaccuracies [2], control system biases, and the effects of payload [3]; and extrinsic errors resulting from external forces such as wind or environmental disruptions [4].

The method proposed in [1] builds on and overcomes the specific limits of these popular limited methods (*i.e.* vision-based methods [5]) by a novel two-module framework. The first module corrects intrinsic errors with the spherical error model coupled to a real-time position correction algorithm, which can adapt to drone-specific variations without needing any extra real-time sensors like cameras [1]. The second module couples a complicated aerodynamic model with training via reinforcement learning (RL) [6] to handle the problem of extrinsic errors. This way, the dynamic adjustment to external wind forces can minimize positional drift and improve precision in the landings, even in difficult conditions [4].

By enabling the coexistence of two independent modules, the framework proposed by [1] attains considerable improvements in the accuracy of landings while maintaining negligible overhead in terms of computation, hence showing the promise for real-life application both in simulated and performance environments.

2 Context and Related Work

Although many vision-based correction methods that use cameras and markers have been widely explored [5], they suffer limitations from environmental conditions such as fog, shadows, or insufficient lighting, and therefore may render them unreliable under various real-world scenarios [7]. Due to the mentioned limitations and drawbacks, there arises a need for stronger solutions that can work efficiently under different environmental conditions.

Machine-learning approaches have also been put forth in this direction to further improve landing accuracy, training models on large datasets to predict and correct positional errors. However, because drones are put together using different materials, it is often common even among the same models from the same manufacturer that sensors, actuators, and control systems have some intrinsic differences, thus upping the positional error. This leads to the need for a more scalable and generic method that can adapt to the variability in configurations for accurate landings [8], [9].

Techniques attempting to address intrinsic errors include pose estimation with markers or noise filtering by algorithms such as Kalman filters. The overall performance can be improved. However, these works often do not generalize to accommodate the broad range of biased and time-varying effects found in a real-world drone system [10]. They also do not counter disturbances or external forces that most frequently corrupt the motion of drones.

To mitigate extrinsic errors, for example due to wind, stabilization algorithms based on IMU data are widely used. While these tackle gusts, they do not directly mitigate drift due to sustained external forces [11]. Using anemometers gives better wind correction, although it may be impractical because of increased cost, drag, and increased system complexity [12].

Reinforcement learning (RL) approaches have shown promise in addressing internal and external challenges by training drones to adapt to their environment. Techniques such as sequential deep Q-networks and adaptive multi-level quantization models have displayed higher degrees of precision in landing within simulation scenarios. On the contrary, they usually require extensive training datasets and generalizing to diverse settings often remains a challenge [6], [13].

Due to regulatory constraints on drone testing, simulation validated these methods. Platforms like PyBullet afford realistic environments for training and evaluation, simulating aerodynamic effects, collisions, and environmental impacts. However, the absence of real-world testing puts a limit on their reliability and generalizability [14].

Finally, aerodynamic modeling plays a vital role in handling external forces acting on landing. Models accounting for drag, ground effects, and wind vortex simulations have been integrated into RL frameworks allowing drones to adapt to dynamic wind conditions. Such models also enable the formulation of realistic training environments closely resembling physical environments [15].

The dual-module methodology proposed by Saryazdi *et al.* fuses the two internal error corrections and adaptation of RL to external factors, thus providing new benchmarks for an accurate landing of drones in diverse real-world conditions.

3 Contribution

A novel framework for achieving precision landing in drones is introduced in [1]. This framework addresses the limitations of existing methods by decoupling the treatment of intrinsic and extrinsic errors into two independent modules. The key contributions of this work are outlined below.

3.1 Intrinsic Error Correction

A path is modeled as a function $p : [0,1] \to \mathbb{R}^3$ where $[0,1]$ stands for the flight time and $\mathbb{R}$ for the 3D-space. The internal path of the drone (its belief in its own position) is noted p_{internal}.

3.1.1 Error Model Basis

An *error model* $\mathcal{P}$ is a function that takes a path and a metdata vector m ($m : [0,1] \to \mathbb{R}^k$) then returns a modified path. The trivial error model that operates no modification over the given path is noted $\mathcal{P}_{\mathbb{I}}$.

$(\mathcal{P}_i - \mathcal{P}_{\mathbb{I}})$ is then a positional error. Thus, an approximation $\tilde{p}_{\text{true}}$ of the drone's true path can be given by (1).

$$\tilde{p}_{\text{true}} = (\mathcal{P}_{\mathbb{I}} + \Sigma_{i=1}^{N} c_i(\mathcal{P}_i - \mathcal{P}_{\mathbb{I}}))(p_{\text{internal}}, m) \tag{1}$$

Where $\forall i \in \{1, \dots, N\}\ c_i \in \mathbb{R}$

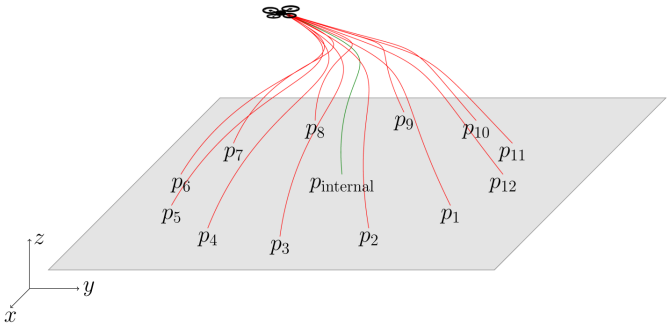

Fig. 1. Example extracted from [1] of a set of error model paths $p_i := \mathcal{P}(p_{\text{internal}}, m)$ for the drone's internal position frame while landing.

3.1.2 Spherical Error Basis

Observing that accumulated errors are *orientation dependent*, it is possible to define a generic spherical error model basis to compute $\tilde{p}_{\text{true}}$. [1].

Some bias ($(b_i)_i \in \mathcal{S}^2$) are chosen on the unit sphere according to *Fibonacci Lattice* [16] ensuring that these points are approximately equidistant. The basis defined by [1] is presented in equation (2).

$$(\mathcal{P}_i(p,m))(t) := p(t) + \alpha \int_0^t \|v_m(t')\|\, \mathrm{R}_m(t')[\mathbf{b}_i]\, dt' \tag{2}$$

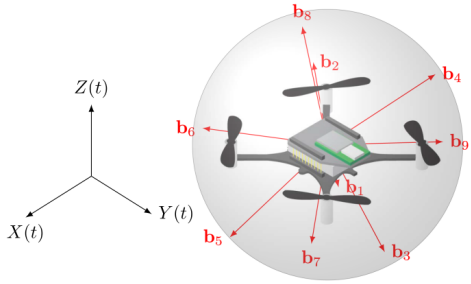

Fig. 2. Example extracted from [1] of a set of bias $\mathbf{b}_i \in \mathrm{S}^2$. The stationary frame basis vectors $(X(t), Y(t), Z(t))$ are found with respect to the inertial position frame (x, y, z) via a transformation by Euler angles at each time $t \in [0,1]$

In Equation (2), $v_m(t')$ represents the drone's velocity at time t', α is a scaling factor empirically chosen based on the expected correction size, and $\mathrm{R}_m(t')$ is the rotation matrix transforming the bias vector $\mathbf{b}_i$ from the drone's stationary frame to its inertial position frame.

3.1.3 Calibration Method

The calibration method proposed by [1] consist in sampled paths p_k where $(p_k)_{\text{true}}$ and $(p_k)_{\text{internal}}$ are measured. Errors are then defined in (3). Note that $t = 1$ is used for simplicity and no loss of generality due to $(c_i)_i$ being constant.

$$\begin{aligned} e_k &:= (p_k)_{\text{true}}(1) - (p_k)_{\text{internal}}(1), \\ (\tilde{\mathbf{e}}_k)^i &:= ((\mathcal{P}_i - \mathcal{P}_{\mathbb{I}})((p_k)_{\text{internal}}, m))(1). \end{aligned} \tag{3}$$

Moreover, (4) holds if $\tilde{(p)}_{\text{true}} = p_{\text{true}}$ [1].

$$\forall k \in \{1, \dots, M\} \quad \Sigma_{i=1}^{N} c_i(\tilde{\mathbf{e}}_k)^i = \mathbf{e}_k \tag{4}$$

The preceding system represents a $(3M)N$ least squares linear regression problem with the unknowns $(c_i)_i$, which can be estimated in closed form using the Moore–Penrose pseudoinverse operation [1] [17].

3.1.4 Design Strategy

This section outlines the use of the precomputed error model for live position correction.

Input: (p_{internal}, m)
Output: $\mathbf{e}_{\text{controller}}$
1: $p_{\text{local true}} \leftarrow \mathcal{P}(p_{\text{internal}}, m)$
2: $s \leftarrow (p_{\text{internal}}(1), m(1))$
3: $s' \leftarrow (p_{\text{local true}}(1), m(1))$
4: $a \leftarrow C(s, s')$
5: **do action** a
6: $\mathbf{e}_{\text{controller}} \leftarrow ((\mathcal{P}_{\mathbb{I}} - \mathcal{P})(p_{\text{internal}}, m))(1)$ # Accumulated deviation in controller space.
7: **return** $\mathbf{e}_{\text{controller}}$

Fig. 3. Algorithm for Live Position Correction [1]

Figure 3 shows how the algorithm estimates the true path based on the internal position. Corrections are applied iteratively to align the drone's position with its true trajectory. The controller is adjusted accordingly to prevent reversion of the corrections.

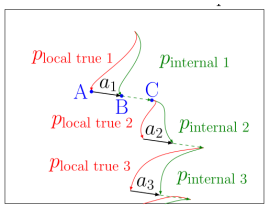

Fig. 4. Visual representation of the internal position correction

Figure 4 illustrates the correction results. Dashed arrows indicate the predicted deviation, while solid paths represent corrected trajectories. The algorithm reduces errors over time, ensuring alignment with the true path without requiring additional sensor inputs.

3.2 Extrinsic Error Correction

The main purpose of this section is to train a reinforcement learning (RL) agent to face external perturbation. Strategies scalable to real cases rely on simulations that include:

- **Aerodynamic modeling:** Simulations must capture forces like drag, ground effects, and wind vortices.
- **State space diversity:** Training must expose the agent to varied conditions and drone states.
- **Reward design:** The reward function should balance landing time, accuracy, and stability.

3.2.1 Aerodynamics Model

An aerodynamics model implementation of the Bitcraze Crazyflie 2.X drone in PyBullet Gym was used for training the RL model [1] [14]. However, the drag force function modelling the resistance experienced by the drone as it move through the air has been modified, partially using the work of [15].

$$D(x,t) = -\mathbf{K}_{\text{aero}}\dot{\Theta}_{\Sigma}\mathbf{R}^{-1}(\dot{x} - W(x,t)) \tag{5}$$

The drag force function from [15] is modified by adding $\mathbf{R}^{-1}$ to account for drone orientation and $W(x,t)$ to model wind effects, enhancing aerodynamic accuracy [1].

3.2.2 Wind Generation Model

A wind generation model is described in [1], enabling the scaling of RL strategies from simulation to real-world scenarios. The wind $W(\mathbf{x},t)$ (7) consists of several components, each is modeled using a Gaussian decaying formula $f(x,y,z,r)$ (6).

$$f(x,y,z,r) = \exp\left(-\frac{x^2+y^2+z^2}{2r^2}\right)(-y,x,0) \tag{6}$$

$$\begin{aligned} W_i(\mathbf{x},t) &= (-1)^{o_i}\omega_{\text{max}} f(\mathbf{x} - c_i(t), r_i) \\ W(\mathbf{x},t) &= \sum_{i=1}^{n} W_i(\mathbf{x},t) \end{aligned} \tag{7}$$

In this model, $c_i(t)$ defines the center of the i-th wind vortex, r_i its radius, and ω_{max} the maximum rotational velocity. These parameters are sampled probabilistically, with $c_i(t)$ drawn uniformly over the domain, and r_i and ω_{max} sampled from predefined ranges. This ensures dynamic and varied wind conditions for better RL generalization.

3.2.3 Reinforcement Model Methods

To execute the RL algorithm, [1] defines the state space $\mathcal{S}$, which includes the drone's position, velocity, Euler's angles, angular velocities, and motor speeds: $s = (\mathbf{x}, \dot{\mathbf{x}}, \mathbf{E}, \dot{\mathbf{E}}, \mathbf{P}) \in \mathcal{S}$, as well as the action space $\mathcal{A}$. The actions in $\mathcal{A}$ are position difference vectors $a = (dx, dy, dz)$, which the DSLPID controller [18] uses to compute the required thrust for drone orientation.

The goal of the RL algorithm is to provide a strategy that maximizes a reward function $R : \mathcal{A} \to \mathcal{S}$ (8):

$$\begin{aligned} R_1(s) &= -\frac{1}{c^2}\left\|\mathbf{x} - \mathbf{T}\right\|^2 - 1, \\ R_2(s) &= 1 - \frac{1}{2}\left(\frac{\max(|\alpha|,|\beta|)}{\pi} + \frac{\max(\mathbf{P})}{\mathbf{P}_{\text{max}}}\right) \\ R(s) &= \begin{cases} R_1(s) & \|\mathbf{x} - \mathbf{T}\| \geq c, \\ R_1(s) + R_2(s) & \|\mathbf{x} - \mathbf{T}\| < c. \end{cases} \end{aligned} \tag{8}$$

In the reward function (8), c defines the threshold distance from the target $\mathbf{T}$ below which additional rewards are given. $\mathbf{P}$ represents the motor speeds, normalized by $\mathbf{P}_{\text{max}}$, the maximum motor speed, to penalize excessive power usage. α and β denote the drone's pitch and roll angles, ensuring stability by penalizing extreme inclinations. The function encourages the drone to minimize its distance to $\mathbf{T}$ while maintaining stable angles and optimizing energy efficiency, balancing precision and control.

After initial experiments with models such as LSPI and OLSPI, the study continued with SAC (Soft Actor-Critic), which demonstrated the best results after training for 1,372 episodes. SAC effectively balances precision and robustness under varying wind intensities, ranging from $\omega_{\text{max}} = 0$ to 2 m/s. [1]

3.2.4 Actions

Two approaches for drone landing actions are compared: RL-based and deterministic DSLPID.

RL Actions RL optimizes both accuracy and time but tends to favor speed. A rescaling trick forces precision near the target, improving accuracy in dynamic conditions like wind.

DSLPID Actions The DSLPID method is simple and stable, moving directly toward the target by halving the remaining distance. However, it struggles with external disturbances like wind.

Comparison RL adapts better to dynamic environments, offering higher precision, while DSLPID is easier to implement and performs well in controlled settings.

3.3 Experimental Validation

The framework's efficacy was validated through a combination of simulation-based and real-world experiments on both modules, providing a robust demonstration of its applicability.

3.3.1 Internal Position Correction

Using the live correction, Figure 5 shows that a model created with $N = 500$ spherical basis model and $M = 30$ data points shows in real conditions a *29% reduction in average radial differences* between the target $\mathbf{T}$ and landing point, compared to no correction [1].

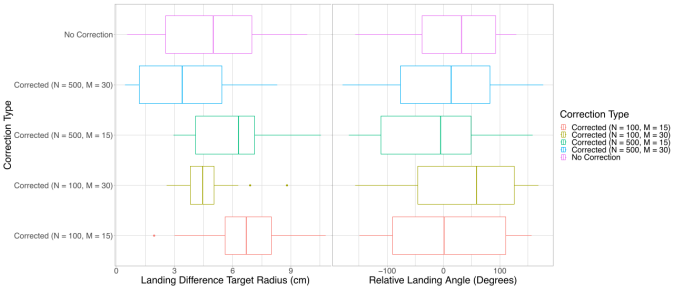

Fig. 5. Result given by [1] shows distance between target **T** and the landing point for each trial by live position correction type (left) and landing angle of each trial relative to the target **T** (right).

3.3.2 *External Position Correction*

Figure 6 shows that in real conditions, the RL strategy outperforms the straight-to-target approach only when there is wind. In calm atmosphere, the average radial distance increases by about 350% in simulation, whereas in real experiments it decreases by 63% under wind conditions.

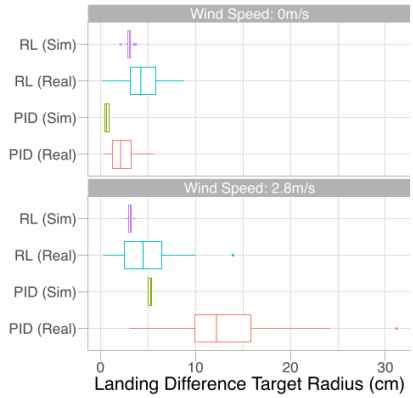

Fig. 6. Comparison of strategies by landing difference Target Radius, given by [1]

4 Discussion

Some subjects of discussion are exposed below.

- **Testing modules together:** The two modules were not tested simultaneously. Would their integration introduce conflicts or affect performance in real-world scenarios?
- **Onboard resource requirements:** Running both modules might increase demand on the drone's CPU. Could this impact real-time responsiveness on resource-constrained hardware?

5 Conclusion

This study presents a thorough synthesis of key findings that address one of the major challenges of landing a drone accurately and safely using a new double-module framework. This framework separates intrinsic and extrinsic error correction and embraces a spherical error model powered by reinforcement learning for significantly improved landing accuracy under a variety of conditions. Realistic modeling of aerodynamics and wind guarantees scalability to real-world application. These innovations are a big leap for autonomous drone operations, with promising applications in delivery, search and rescue, and anywhere autonomous systems need to be highly precise.

References

[1] S. Saryazdi, B. Alkouz, A. Bouguettaya, and A. Lakhdari, "Using reinforcement learning and error models for drone precise landing," *ACM Transactions on Internet Technology*, vol. 24, pp. 1–30, 8 2024.

[2] J. V.-V. Gerwen, K. Geebelen, J. Wan, W. Joseph, J. Hoebeke, and E. D. Poorter, "Indoor drone positioning: Accuracy and cost trade-off for sensor fusion," 2021.

[3] L. Qian, S. Graham, and H. H. Liu, "Guidance and control law design for a slung payload in autonomous landing: A drone delivery case study," *IEEE/ASME Transactions on Mechatronics*, vol. 25, pp. 1773–1782, 8 2020.

[4] X. Liu, K. Lam, B. Alkouz, B. Shahzaad, and A. Bouguettaya, "Constraint-based formation of drone swarms," 1 2022. [Online]. Available: http://arxiv.org/abs/2201.11916

[5] V. Sudevan, A. Shukla, and H. Karki, "Vision based autonomous landing of an unmanned aerial vehicle on a stationary target," 2017.

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 9 2013.

[7] F. Cocchioni, E. Frontoni, G. Ippoliti, S. Longhi, A. Mancini, and P. Zingaretti, "Visual based landing for an unmanned quadrotor," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 84, pp. 511–528, 12 2016.

[8] E. Petritoli, F. Leccese, and L. Ciani, "Reliability and maintenance analysis of unmanned aerial vehicles," *Sensors (Switzerland)*, vol. 18, 9 2018.

[9] U. Nepal and H. Eslamian, "Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs," *Sensors*, vol. 22, no. 2, p. 464, 2022.

[10] Y. Zhong, Z. Wang, A. V. Yalamanchili, A. Yadav, B. N. Srivatsa, S. Saripalli, and S. T. Bukkapatnam, "Image-based flight control of unmanned aerial vehicles (uavs) for material handling in custom manufacturing," *Journal of Manufacturing Systems*, vol. 56, pp. 615–621, 7 2020.

[11] G. P. Kumar, B. Praveen, U. T. Nisanth, and M. Hemanth, "Development of auto stabilization algorithm for uav using gyro sensor," *International Journal of Engineering Research & Technology*, vol. 5, no. 9, pp. 1–3, 2017.

[12] A. K. Ramanathan, L. M. Headings, and M. J. Dapino, "Airfoil anemometer with integrated flexible piezo-capacitive pressure sensor," *Frontiers in Materials*, vol. 9, 7 2022.

[13] N. A. Mosali, S. S. Shamsudin, S. A. Mostafa, O. Alfandi, R. Omar, N. Al-Fadhali, M. A. Mohammed, R. Q. Malik, M. M. Jaber, and A. Saif, "An adaptive multi-level quantization-..." ..., vol. ..., no. ..., p. ..., 2022.

[14] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 7512–7519.

[15] J. Förster, B. Thesis, and M. D. H. Raffaello, "System identification of the crazyflie 2.0 nano quadrocopter," 2015.

[16] D. P. Hardin, T. J. Michaels, and E. B. Saff, "A comparison of popular point configurations on $\{S\}^2$," 7 2016. [Online]. Available: http://arxiv.org/abs/1607.04590

[17] A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*, 2nd ed., C. B. in Mathematics, Ed. Springer, 2003.

[18] Z. Yuan, A. W. Hall, S. Zhou, L. Brunke, M. Greeff, J. Panerati, and A. P. Schoellig, "Safe-control-gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 142–11 149, 2022.