Reinforcement Learning for Outdoor Balloon Navigation

A Successful Controller for an Autonomous Balloon

By Simon L. Jeger, Nicholas Lawrance, Florian Achermann, Oscar Pang, Mirko Kovac, and Roland Y. Siegwart

Autonomous ballooning allows for energy-efficient long-range missions but introduces significant challenges for planning and control algorithms, due to their single degree of actuation: vertical rate control through either buoyancy or vertical thrust. Lateral motion is typically due to the wind; thus, balloon flight is both nonholonomic and often stochastic. Finally, wind is very challenging to sense remotely, and estimates are often available only via low-temporal-and-spatial-frequency predictions from large-scale weather models and direct in situ measurements. In this work, reinforcement learning (RL) is used to generate a control policy for an autonomous balloon navigating between 3D positions in a time- and spatially varying wind field. The agent uses its position and velocity, the relative position of the target, and an estimate of the surrounding wind field to command a target altitude. The wind information contains local measurements and an encoding of global wind predictions from a large-scale numerical weather prediction (NWP) model around the current balloon location. The RL algorithm used in this work, the soft actor–critic (SAC), is trained with a reward favoring paths that reach as close as possible to the target, with minimum time and actuation costs. We evaluate our approach first in simulation and then with a controlled indoor experiment, where we generate an artificial wind field and reach a median distance of 23.4 cm from the target within a volume of $3.5 \times 3.5 \times 3.5$ m over 30 trials. Finally, using a fully autonomous custom designed outdoor prototype capable of controlling altitude, long-range communication, redundant localization, and onboard computation, we validate our approach in a real-world setting. Over six flights, the agent navigates to predefined target positions, with an average target distance error of 360 m after traveling approximately 10 km within a volume of $22 \times 22 \times 3.2$ km.

Digital Object Identifier 10.1109/MRA.2023.3271203
Date of current version: 8 June 2023

BACKGROUND

Lighter-than-air vehicles, such as balloons and blimps, consume, in contrast to other flying vehicles, extremely low amounts of energy to stay in the air. They are capable of completing long-range missions over days, weeks, and sometimes even months [1]. This makes them ideal platforms for scientific aerial sampling applications, such as monitoring aerosols [2], volcanic plumes [3], and other atmospheric data [4].

Lighter-than-air vehicles have a lower overall density than the surrounding air, which causes buoyancy. The magnitude of the buoyancy force is influenced through either temperature or gas composition. Different balloon types exist: hot-air balloons are sometimes used as scientific platforms [5], but most of the time, their use case is focused on recreational activities. Compared to gas balloons [6], they are less efficient, as the density difference of hot air compared to helium and hydrogen is lower, and they require additional energy to change the air temperature inside the balloon. However, they offer simple buoyancy control via direct heating of the

1070-9932/23©2023IEEE

buoyant gas. For long-range missions, a hybrid concept combining lower-density gases and heated air, called "Rozière," is often used [7].

Balloons have only one degree of actuation. By changing their buoyancy, they accelerate upward or downward, which is used to change altitude. Lateral movement cannot be achieved internally and occurs only through drag forces induced by wind. Navigation in 3D space is, then, made possible by choosing the right altitude, where the system drifts with the wind in the desired direction. This results in a challenging path planning problem, as the time-optimal (or even feasible) path between two points might require a spatial detour (Figure 1).

The world of autonomous ballooning is quite small. Most prior work describes modeling the ascent of balloons [8] or focuses on heuristic station keeping algorithms in simulation [9].

Project Loon [1] tried solving a similar problem in the past. To provide Internet connectivity, it launched balloons tasked with station keeping to altitudes of 20 km, using an RL-based control method [quantile regression deep Q-network (DQN)]. The focus of our work lies in flying a balloon to a predefined target position at a low altitude (up to 3 km above mean sea level). Compared to high altitudes, lower-altitude wind inside the planetary boundary layer varies over smaller length scales and timescales, which allows for a higher number of reachable locations but complicates the path planning.

RL has also been used to control autonomous blimps [10], which are similar in physical appearance but offer limited lateral maneuverability. This results in a different control problem since balloons have no lateral propulsion and therefore are exposed to the wind. These previously mentioned RL-based methods focus on off-policy methods, mainly variants of the DQN family. In this work, we propose using the SAC, which is also an off-policy method but has the advantage of a continuous action space.

This work focuses on a gas-filled weather balloon with vertical speed control via a small electrically powered propeller, as the required components are widely available and the mechanical complexity is low. This simplification can be made without loss of generality, and a suitable policy could be found for similar balloon-like systems capable of altitude control. The contributions of this work include

- training an RL agent for low-altitude balloon navigation to reach a target position, with only a single degree of actuation
- design and hardware implementation of autonomous balloons for indoor and outdoor settings
- simulation-to-real transfer of the trained RL agent, leading to the demonstration of fully autonomous indoor and outdoor low-altitude balloon flight.

PROBLEM DESCRIPTION

This work addresses the problem of balloon path planning with the goal of reaching a 3D target region in a predicted, but possibly incorrect, wind field, using only one degree of actuation. The resulting path p should yield a residual distance $d_{\min}$ between the balloon and target that is less than or equal to an acceptance radius d_a while staying within spatial bounds B and minimizing the duration of flight T and energy consumed E due to actuation (1). The latter is weighted by the factor c such that

$$\begin{aligned} \underset{p}{\text{argmin}} \quad & T(p) + c \cdot E(p) \\ \text{subject to} \quad & p \in \mathcal{B} \\ & d_{\min}(p) \leq d_a. \end{aligned} \tag{1}$$

In our work, we treat this as a time-discretized problem with decision interval Δ_t between actions a.

Due to flight regulations in Switzerland, autonomous systems are restricted to stay under flight level 100. This pressure-based and, hence, weather-dependent description corresponds to an altitude of about 3 km above mean sea level, acting as a ceiling of the spatial boundary B.

APPROACH

This problem has high complexity, considering the nonisotropic costs, spatial and temporal variation, and uncertainty in the wind data. Solving this problem efficiently is difficult with classical planners, as they do not scale well with dimensionality and colored noise [11]. Once deviated from the planned path, replanning is required. This becomes challenging, as it is done based on an NWP estimate, which, in practice, can differ significantly from the ground truth. The nonholonomic constraints of the wind-dependent movement of the balloon lead to nonisotropic costs, which prevent short cuts for path search algorithms, making this approach computationally very expensive.

This article focuses on building a robust and adaptable RL-based system, allowing the planning of nonobvious connections and solutions and outputting a policy that can be evaluated anywhere in the state space, rather than just a plan.

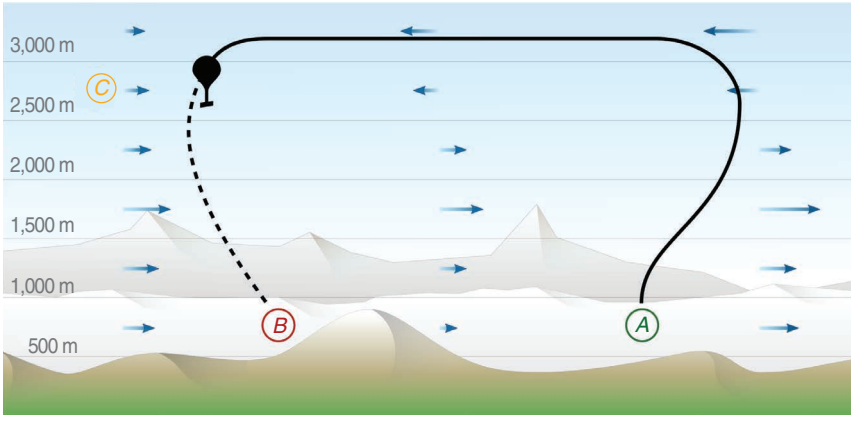

FIGURE 1. A balloon can control only its altitude and is dependent on wind for lateral movement. To fly from start *A* to target *B*, the shortest feasible path might contain a spatial detour, while some points, such as *C*, are not reachable at all.

We treat the problem as a Markov decision process with a tuple of states, actions, and transition probabilities. To allow the use of a data-driven method, such as RL, observations are generated in simulation and evaluated according to a reward function. This is used to train an agent that can then apply the learned policy in a real-life environment.

STATE SPACE

We define two primary frames, an inertial ground-fixed east–north–up frame, denoted by superscript $\mathcal{I}$, and a body-fixed frame attached to the balloon center of gravity b. The balloon position in the inertial frame is $\mathbf{r}_b^{\mathcal{I}} \in \mathbb{R}^3$, and the target position is $\mathbf{r}_t^{\mathcal{I}} \in \mathbb{R}^3$.

The information given to the agent has the following form (2): the global position $\mathbf{r}_b^{\mathcal{I}} \in \mathbb{R}^3$, the inertial velocity $\dot{\mathbf{r}}_b^{\mathcal{I}} \in \mathbb{R}^3$ of the balloon, the position of the target relative to the balloon $\mathbf{e} = \mathbf{r}_t^{\mathcal{I}} - \mathbf{r}_b^{\mathcal{I}}$, the distance between the terrain and ceiling $\mathbf{c} \in \mathbb{R}$ at the current position of the balloon, a local lateral (x, y) wind measurement $\mathbf{w}_{xy} = (w_x^{\mathcal{I}}, w_y^{\mathcal{I}}) \in \mathbb{R}^2$, and, finally, the compressed wind information $\mathbf{M} \in \mathbb{R}^{\beta}$, depending on the encoding method used:

$$x = \begin{bmatrix} \mathbf{r}_b^{\mathcal{I}} \\ \dot{\mathbf{r}}_b^{\mathcal{I}} \\ \mathbf{e} \\ \mathbf{c} \\ \mathbf{w}_{xy} \\ \mathbf{M} \end{bmatrix} \in \mathbb{R}^{(12+\beta)}. \tag{2}$$

The local wind measurement $\mathbf{w}_{xy}$ is determined through first-principle calculations [finite differences, (6)] based on position data $\mathbf{r}_b^{\mathcal{I}}$ filtered through an extended Kalman filter (EKF), the actuating force F_u responsible for ascent and descent, and physical properties, such as cross-section area S, air density ρ_a, drag coefficient c_d, and mass m:

$$\dot{\mathbf{r}}_{b,t}^{\mathcal{I}} = \frac{\mathbf{e}_t - \mathbf{e}_{t-1}}{\Delta_t} \quad \dot{\mathbf{r}}_{b,t-1}^{\mathcal{I}} = \frac{\mathbf{e}_{t-1} - \mathbf{e}_{t-2}}{\Delta_t} \tag{3}$$

$$C = \frac{S \cdot \rho_a \cdot c_d}{m} \tag{4}$$

$$\mathbf{k} = \frac{\mathbf{F_u} \cdot \Delta_t + \dot{\mathbf{r}}_{b,t}^{\mathcal{I}} - \dot{\mathbf{r}}_{b,t-1}^{\mathcal{I}}}{\Delta_t \cdot C} \tag{5}$$

$$w_{xy} = \dot{\mathbf{r}}_{b,t-1}^{\mathcal{I}} - \operatorname{sign}(\mathbf{k}) \cdot \sqrt{|\mathbf{k}|}. \tag{6}$$

Before passing the state vector to the RL algorithm, all entries are normalized to an interval $\in [-1, 1]$, using (7), to improve training. The normalization constant $\bar{x}$ is the mean value for each state-space entry across all training data:

$$x_{\text{norm}} = \frac{x}{|x| + |\bar{x}|}. \tag{7}$$

WIND ENCODING

The encoding of the surrounding wind field is crucial to this problem, as it is increasingly difficult for the RL agent to train with a large state space. For this reason, only a compressed representation of the surrounding wind data is passed to the agent, determined by the predefined window size (x and y dimensions) and resolution (z dimension). Wind in the x and y directions usually has a much larger magnitude than in z. Since the system is directly controllable in z (i.e., closed-loop control for a target inertial altitude), it is sufficient to pass wind information only about the x and y dimensions. This assumption may hold only in relatively flat terrain and would need to be revisited for flights in mountainous regions, where the vertical wind may exceed the balloon's vertical control authority.

To further reduce the state space, the lateral xy wind is averaged across a local window, as wind is usually quite spatially continuous in those dimensions. This is then further averaged in the z dimension according to the resolution parameter to reduce the state space even more. This compression down to β entries still represents major trends within a small state space. There is a tradeoff to be met when choosing the window size: too small and only local wind information is passed to the RL-agent, too large and important details are averaged out. The same holds for the resolution: too high and the large state space makes training difficult, too low and important details are averaged out. Multiple approaches were considered for this compression; most notably, a variational autoencoder was evaluated in [15]. The benefit of this computationally rather expensive step was minimal, and therefore, use of the autoencoder was not continued.

ACTION SPACE

Instead of controlling thrust directly, a scalar action $a \in (0, 1)$ is produced, representing the relative altitude between the terrain and maximum flight level at the current xy position. A low-level closed-loop controller is then used to control the balloon and reach the desired altitude. This allows for a better transition into real-world experiments, as this second low-level controller counteracts some of the noise and imperfections in the modeling.

REWARD FUNCTION

The reward function is of critical importance in RL, affecting the agent's choice of policy when maximizing the expected return. During an episode, a reward is given after every action. An episode ends when the agent either runs out of bounds or runs out of time. The reward r is built up in the following way (Algorithm 1): for every time step Δ_t within bounds (meaning not leaving the environment $\mathcal{B}$ through the borders, touching the terrain, or reaching the maximum altitude), a small negative reward $c_{\text{step}} = -10^{-5}$ is given to encourage the agent to find the target as quickly as possible. Additionally, a small negative reward $c_{\text{action}} = -15 \cdot 10^{-5}$ proportional to the difference of the action to the current altitude is added to encourage smooth and energy-efficient solutions. If the agent gets within the acceptance radius of the target, the episode is terminated with a large positive reward $c_{\text{target}} = 1$. If the agent runs out of bounds or out of time, the episode is terminated with a large negative reward $c_{\text{out}} = -1$, $c_{\text{time}} = -1$. Since this results in quite a sparse problem, a bonus $c_{\text{bonus}} \in [0, 1)$ is rewarded at the end of an

ALGORITHM 1. The reward function.

if $\mathbf{r}_b^I \in \mathcal{B}$: **then**
  if residual $\leq d_a$: **then**
    $r = c_{\text{target}}$
  else if $t > t_{\text{max}}$: **then**
    $r = c_{\text{time}} + c_{\text{bonus}}$
  else
    $r = c_{\text{step}} \cdot \Delta_t + c_{\text{action}} \cdot |u|$
  end if
else
    $r = c_{\text{out}} + c_{\text{bonus}}$
end if

unsuccessful episode to promote paths that came close to the target but did not quite reach it, compared to paths that did not get close to the target at all. This bonus increases linearly based on the closest distance to the target during the episode, makes learning converge faster, and increases the success rate. The magnitude of those rewards matters only relative to one another, and all together, the rewards dictate the optimization problem: e.g., a c_{action} close to zero results in a waste of energy for unnecessary ascent and descent. On the other hand, a big negative c_{action} leads to static solutions, where the agent picks an altitude and stays there because the cost of change is too high. Similar effects can be observed with a c_{step} close to zero, where the agent favors very safe but slow routes. With a big negative c_{time}, the agent is willing to try high-risk routes just to reach the target faster. In our case, all those values were balanced heuristically in simulation in favor of smooth energy- and time-efficient trajectories.

CONTROL SCHEME: SAC METHODS

The SAC is a method that works with two different neural networks: an actor and a critic. To train both networks, random observation samples are drawn and passed through the critic, which calculates a temporal difference error and passes it to the actor, which then predicts an action. This off-policy algorithm aims to simultaneously maximize the expected return and entropy to successfully finish the task while acting as randomly as possible [12]. The SAC can deal with a continuous action space, which is crucial for the high-level control scheme used here. It is implemented using the PFRL library [13].

Other RL methods were considered, mainly, the DQN [14], as used in [1], but it did not perform as well as the SAC in our case [15]. The network we use is dense, 512 nodes wide, and four layers deep, with rectified linear unit activations [16].

WIND AND TERRAIN DATA

We use large-scale wind predictions from an NWP simulation provided by an online service [Meteomatics, on 25 April 2022 (https://meteomatics.com)]. The model contains the terrain information and wind in the x and y directions in a 3D grid. This NWP has a nonlinear altitude mapping, which depends on the height above the ground and features a higher resolution in the z dimension (≈30 m) than horizontally (1,100 m). The terrain data match the horizontal resolution of the wind data. The available dataset contains a large part of Switzerland over the time span of one day, in hourly resolution, for 12 evenly spaced days throughout the year. A mirrored version along the x- and y-axes is generated and added to the initial dataset to avoid directional biases in training.

NOISE

A crucial part of a realistic simulation is an appropriate noise model to minimize the simulation-to-real gap to ensure that the learned policy can be directly applied. It forces the algorithm to learn robust policies for two sources of wind prediction error: time variations and inaccurate wind estimates as compared to real-world conditions. In the case of wind, noise needs to be random but also correlated. We use an industry standard turbulence model [17] with turbulence intensity $W_{20} = 15$ kn. The resulting disturbances are scaled to the average velocity in the wind field and added during the state transition. The resolution of the noise model is four times the one of the wind model in both the horizontal and vertical directions. This a tradeoff between realistic continuous wind gusts and computational complexity.

Normally distributed noise is added to the position estimate to simulate sensory inaccuracies. For lateral estimation through GPS, a standard deviation of 11 m is used. The pressure sensor-based altitude estimation is simulated with a standard deviation of 0 m.

RL FRAMEWORK SETUP

A simulated environment with a volume of $\mathcal{B} = 22 \times 22 \times 3.2\text{km}$ is used to collect observations for training (Figure 2). At the beginning of each episode, random patches of wind and terrain data are selected from the dataset as the training environment. During the episodes, the wind data are linearly interpolated in the temporal dimension.

The transition function takes the interpolated wind at the current position, adds noise to it, and calculates the next state, modeling the balloon as a sphere with mass and drag. To account for changes in volume due to the decreasing pressure and temperature with altitude, the size of the balloon is adapted accordingly based on a standard atmosphere model [18]. Modeling complex effects, such as apparent mass, can be circumvented: first, weight and thrust are measured. Then, the drag coefficient of the sphere in simulation is adapted to fit the terminal velocity when applying thrust in real-world experiments.

LABELING DATA: TARGET GENERATION

Balloons can travel only with the wind, never against it. This limits their set of reachable targets. Training requires observations from successful episodes; hence, the targets during training need to be reachable but also exhibit enough variation to prevent overfitting.

One way to ensure that a target is reachable is to perform a forward simulation of random actions until the balloon runs out

of bounds or time and then place the target on said trajectory. To achieve uniform coverage of the action space through this random rollout strategy, actions are chosen sequentially at random, with a fixed probability of 0.005 every second if the previously set altitude is reached. This prohibits choosing altitudes far above and below the balloon without it reaching them, causing an oscillatory behavior. Once a trajectory has been generated, target locations on the path are randomly selected until all the following conditions are met or a time-out is reached:

1) The target, with its acceptance radius, is fully within bounds.
2) The target is more than a minimum distance away from the starting point.
3) The trajectory leading to the target is nontrivial (not reachable with only action "thrust up").

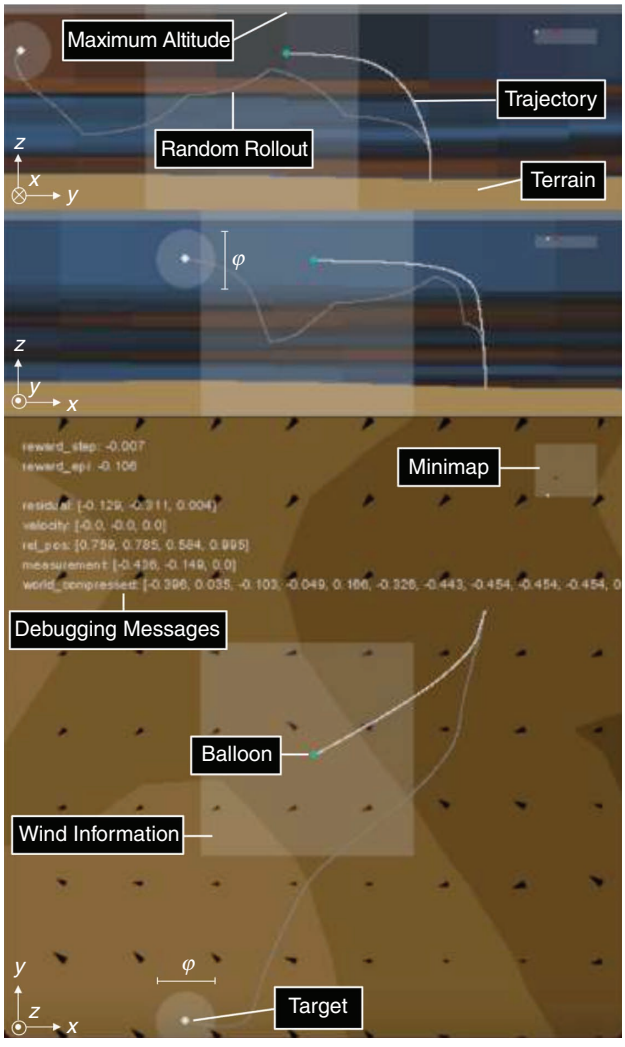

FIGURE 2. The 3D environment with terrain, wind, the randomly generated maximum altitude, the balloon, and the target with an acceptance radius. The brown areas in the lower half display the contour of the terrain from a bird's-eye perspective. Wind blowing toward the right is displayed as orange, to the left as blue, with brightness representing magnitude. In the bottom half, wind at the current altitude is displayed through black arrows. The gray line represents the random rollout used to generate the target, and the white line indicates the trajectory of the balloon.

If not all those conditions can be met, the target is placed in the middle of the trajectory. An example can be seen in Figure 2.

BENCHMARK

Because many algorithms in autonomous ballooning focus on station keeping and/or are not open source [1], [8], [9], a heuristic-based benchmark policy was developed. The goal here is to compare the performance of the proposed RL algorithm with an intuitive easy-to-grasp but effective algorithm. The benchmark policy reduces the problem to two important altitudes. First, the altitude where the projection of the wind vector $\mathbf{w}_{xy}(a)$ onto the residual vector to the target $\mathbf{e}$ is maximized, meaning that at this height, the balloon approaches the target at the highest speed. The other important altitude is the height of the target t_z itself; otherwise, the balloon misses, flying above or below the target. To switch between those two heights, a simple bang–bang controller is used that activates depending on the remaining distance in the x and y dimensions to the target (Algorithm 2). This distance parameter d_{switch} requires tuning. To identify those two altitudes, the benchmark uses only the precomputed NWP, without considering the local wind measurement.

TRAINING

Since an off-policy RL method is used, the training process allows the use of parallelization. Seventy independent agents trained for 60 h and over 12,000 episodes, writing their observations to a shared buffer that was then used to train their respective neural networks.

The initial position of the balloon is drawn from a uniform distribution around the map center, with a variation of 1 km in each direction to diversify the training. Then, the algorithm has 120 min to get within range of the target ($d_a = 300\ m$) that was generated through the random rollout method. For training, a rather large time interval between decisions ($\Delta_t = 3$ min) is used to make the problem less sparse.

TESTING

In simulation, training and testing are very similar. A previously unseen set of wind fields together with a randomly sampled turbulence field are used to build an environment where the balloon can move around to reach the target. In testing, a shorter time interval ($\Delta_t = 1$ min) is used to allow for more decisions without making the training any sparser. This increases the success rate by about 10% compared to

ALGORITHM 2. The benchmark.

if $\sqrt{e_x^2 + e_y^2} > d_{\text{switch}}$: **then**

$$a = \text{argmax}\left(\begin{bmatrix} w_x(a) \\ w_y(a) \end{bmatrix}^T \cdot \begin{bmatrix} e_x \\ e_y \end{bmatrix}\right)$$

else

$a = t_z$

end if

TABLE 1. The performance of the two algorithms compared in simulation within a 22 × 22 × 3.2-km environment, using an acceptance radius of 300 m.

METHOD	MEDIAN MINIMUM RESIDUAL	SUCCESS RATE
Benchmark	258 m	51%
RL	134 m	73%

the training time interval but also leads to slightly less smooth trajectories.

Tests are done in the same 3D environment as the training: a $22 \times 22 \times 3.2$-km volume with a target acceptance radius of 300 m. The simulation shows the superior performance of the RL over the benchmark (Table 1).

INDOOR EXPERIMENTS

A major challenge in robotics and especially in RL is the bridge between simulation and real-world environments. To test the performance of the algorithm in a real, but still controlled, environment, a small prototype was built and tested with an artificial wind generator. These experiments took place in the motion capture arena of the Aerial Robotics Lab at Imperial College London.

INDOOR PROTOTYPE

Testing the algorithm requires a prototype capable of closed-loop altitude control, position sensing, and communication. In short, the prototype should to be able to mimic the movement of any balloon-like system (hot-air balloon, gas balloon, and so on) to validate simulated scenarios.

To induce vertical movement in the prototype, different mechanisms were evaluated (Figure 3). Gas compression solutions are efficient since they require energy only during buoyancy changes but are difficult to build at the small scale required for indoor testing. Using only one propeller leads to the lightest option but results in spinning of the system during ascent and descent, as the fins mounted below the propeller counteract torque only at a specific rotational speed. Using two counterrotating propellers still allows for a simple and relatively light mechanism without inducing any rotation in the system.

Hence, the indoor prototype (Figure 4) is equipped with two counterrotating propellers. An Arduino Nano IoT is used for communication over Wi-Fi, and all computations are made off board: based on the current state of the prototype, the algorithm predicts an action that is turned into a pulse-width modulation (PWM) signal by a proportional derivative controller. The PWM signal is sent from the Nano to the electronic speed controller (ESC), which, in turn, controls the motors. Localization is done through the motion capture system installed in the flight arena and filtered through an EKF at 30 Hz. The power supply consists of a one-cell lithium polymer battery and a voltage step-up to consistently power the Arduino with 5 V. The complete (unfilled) system weighs 30 g and is lifted by a helium-filled foil balloon,

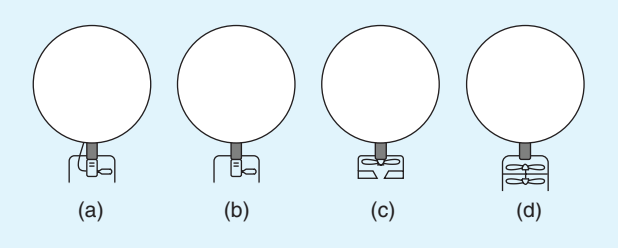

FIGURE 3. The evaluation of different mechanisms for altitude control: (a) releasing or pumping in lifting gas from a fixed-volume container into the balloon; (b) releasing air ballast from, or pumping it into, a fixed-volume container [19]; (c) actuation through one propeller, with fins mounted below to counteract the resulting torque; and (d) actuation through two counterrotating propellers.

chosen for its superior diffusion resistance. This is crucial for balloons of this small size, due to their high surface-to-volume ratio. The prototype is filled to a level where it stays slightly negatively buoyant. This allows for easy recovery, even when the motors stop spinning.

ENVIRONMENT

To control environmental factors as much as possible, the indoor experiments took place in the flight arena, equipped with a millimeter-precision motion capture system within a volume of $3.5 \times 3.5 \times 3.5$ m.

Lateral movement of the prototype required the development of wind generators. They consist of four motors equipped with large 17-in propellers mounted on a metal structure in a horizontal manner and were built with the goal of producing pure lateral air motion. The motors next to each other turn in opposite directions, which helps counteract some of the vertical flow induced by the propellers. The ESCs that control the motors receive their PWM signal through an Arduino, which, in turn, is connected to the local Wi-Fi. The generated wind of a single propeller was measured on a grid of 0.5×0.1 m at 99 locations, using an anemometer, and a continuous model of the wind flow was generated through bilinear interpolation between measurements (Figure 5). The resulting model was used in place of the simulated wind for training.

Due to size and time constraints, only a 2D wind case is considered in the indoor experiments. Wind is generated along the x direction, varying through the altitude z but staying constant in the y direction. For each test flight, the prototype starts in the middle of the arena and tries to reach a specified target point. Controlled movement is possible only within the xz-plane, due to the 2D nature of this setup, and therefore, the reward function evaluates distance only on the xz-plane (Figure 6).

TRAINING

For this case, the physical properties (diameter, mass, drag coefficient, and actuation force) in the simulated dynamics are adapted to the indoor prototype, the acceptance radius is lowered to 10 cm, and the decision interval Δ_t during training is changed to 1 s. The wind data used to generate the training set are built using the model from Figure 5, with randomly sampled height, direction, and power of the wind generator.

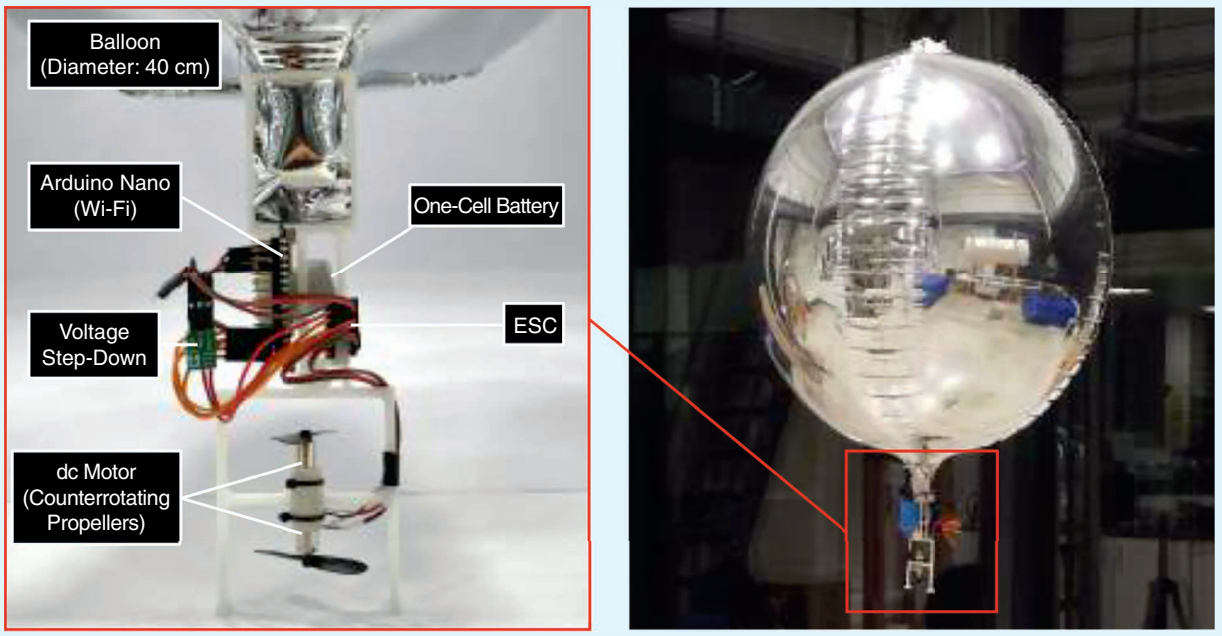

FIGURE 4. (a) The indoor prototype consists of a 3D-printed structure, an Arduino Nano IoT for Wi-Fi communication, a one-cell battery, and an electronic speed controller (ESC) that controls the two motors with their counterrotating propellers. (b) This is lifted by a foil balloon filled with helium, 40 cm in diameter.

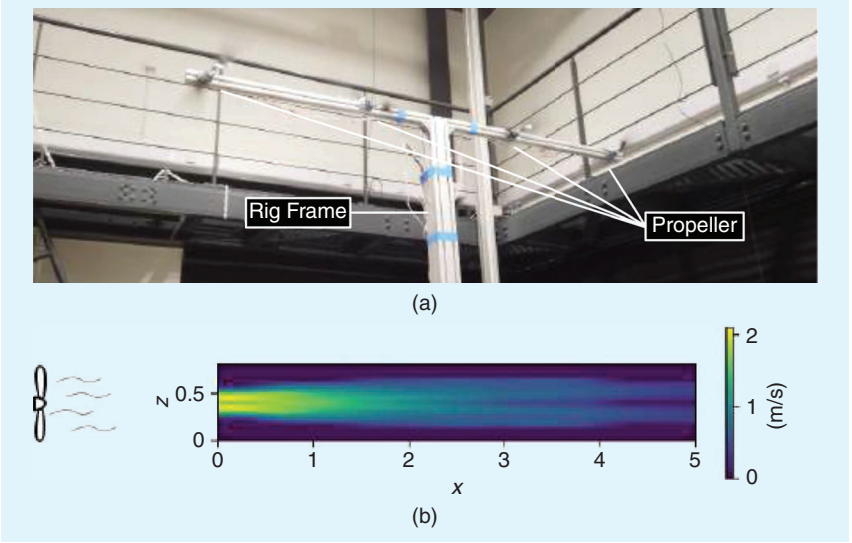

FIGURE 5. (a) A wind generator capable of generating wind speeds between 0.4 m/s and 5 m/s. (b) The resulting wind model of a single propeller of the wind generator was obtained through 99 measurements taken with an anemometer and bilinear interpolation.

EXPERIMENT

We compared the RL algorithm against the benchmark approach. The RL algorithm was trained entirely in simulation and directly applied to the real world without any changes to the policy. We tested three scenarios, each with the same wind field and start location but with distinct target locations. The experiments were repeated 10 times (for each method and target) to get consistent results. The decision interval Δt during testing was adapted to 0.5 s. Tests were terminated after a maximum duration of 25 s and if the balloon ran out of bounds. The battery was charged before every set of (10) flights and the balloon balanced with small weights to make the conditions reproducible.

It needs to be mentioned that for each wind field and target combination, the distance parameter d_{switch} of the benchmark method had to be tuned in the flight arena before testing. Tuning in simulation was insufficient, often resulting in the system moving out of bounds before getting close to the target. This makes the benchmark approach an infeasible option for real-world performance and highlights the robustness of the RL algorithm, which was taken out of simulation without any adaptations.

The targets for the three scenarios were placed in the following way (Figure 7):

1) at the bottom left (0.75 m, 0.5 m), requiring ascending to move to the left and then descending to reach the target
2) in the center (1.5 m, 1.5 m), requiring sequential ascent and descent to move in a circular path to reach the target
3) at the top right (2.1 m, 3.5 m), requiring staying at a lower altitude for a short period before ascending to reach the target.

The benchmark proved to be successful at reaching close to the target in this simple two-phase 2D wind case but was clearly outperformed by the RL in two of the three target scenarios (Figure 7). Full flight statistics with hyper parameter d_{switch} for each flight are available on the data page.

These experiments show that policies learned in a simulated environment with an appropriate noise model can be directly used in a controlled environment, performing consistently over 30 trials.

OUTDOOR EXPERIMENTS

The success of the RL approach in the indoor tests allowed us to continue to large-scale outdoor tests.

OUTDOOR PROTOTYPE

Testing in an open-world environment required the development of an outdoor prototype (Figure 8) capable of closed-loop altitude control, position sensing, communication, and onboard computation.

The propulsion system follows the same principle as the indoor prototype and consists of two counterrotating propellers. All computations are made on a Raspberry Pi Zero equipped with a Waveshare SIM7600G-H 4G module for global navigation satellite system (GNSS) reception and telecommunication (the antenna is mounted parallel to the ground to improve reception). An Adafruit BMP388 pressure sensor module is used for height estimation. All sensor inputs are filtered through an EKF. The local wind $\mathbf{w}_{xy}$ is estimated based on first-principle dynamics [finite differencing of the filtered position and velocity estimates, (6)]. The power supply consists of one three-cell 3,200-mAh battery and a voltage step-down to consistently power the Raspberry Pi with 5 V, allowing flight times of roughly 1.5 h. The required lifting force to compensate the total weight of 1.2 kg is generated through a helium-filled weather balloon with a filled diameter of 1 m. The diffusion of helium at this diameter and mission duration is not significant due to the low surface-to-volume ratio. To ensure safe recovery in the case of failure, the prototype is filled to a level where it stays slightly negatively buoyant.

The system itself is fully autonomous and performs all computations onboard via a PyTorch model running on the Raspberry Pi, requiring no further input to reach the target. For safety reasons, the RL algorithm can be overridden at any time through a command from the pilot on the ground via short message service (SMS). The pilot also receives regular status updates via SMS once per minute. This means of communication has no direct range limit and allows the balloon to travel anywhere within network coverage. As an additional redundant safety measure, a second independent GNSS tracker with its own means of communication and power source is attached to the prototype. Once the battery is connected, the Raspberry Pi boots and automatically launches the RL and balloon control scripts using system daemons. The control script is also responsible for GPS readout, communication, and motor actuation. In case of a malfunction, the script automatically restarts, allowing for recovery in the unlikely event of a short power supply interruption.

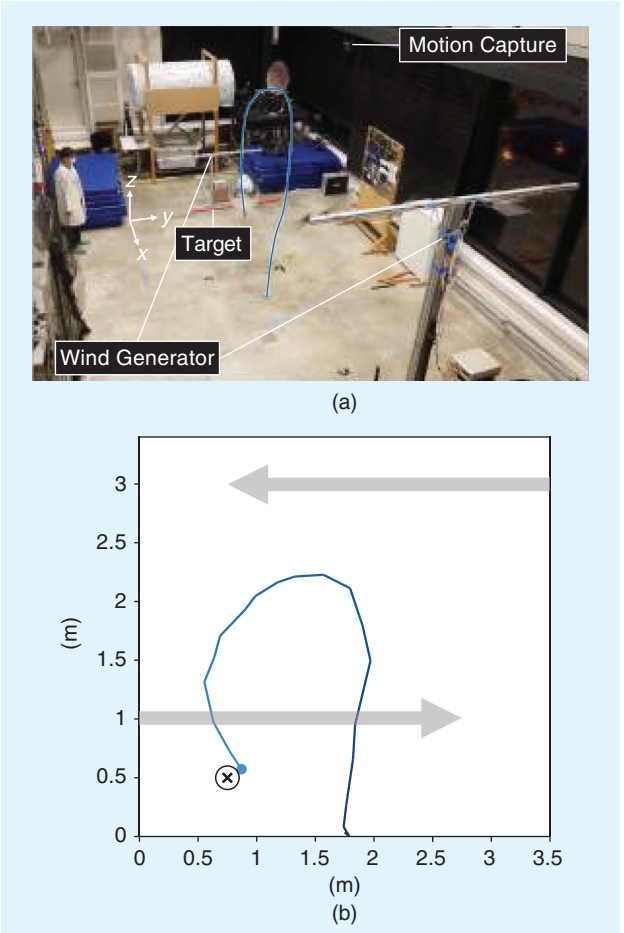

FIGURE 6. (a) The indoor test case, with two wind generators facing each other at different heights, resulting in a 2D wind field (varying in the *x* and *z* dimensions). (b) The prototype (blue trajectory) starts in the middle of the arena and tries to get as close as possible to the target, which corresponds to a target line in 3D [the red line in (a)].

To reach and remain at the RL-predicted altitude, a simple low-level switching controller is used (8) to keep the altitude tracking error e below a certain threshold $l = 15\,\text{m}$. This design was mainly chosen due to the underpowered nature of this system, leading to saturation in most scenarios, and to imitate how balloon pilots manually control their balloons:

$$u = \begin{Bmatrix} 1, & \text{for } e \le -l \\ -1, & \text{for } e \ge l \\ 0, & \text{else} \end{Bmatrix}. \tag{8}$$

ENVIRONMENT

All outdoor experiments took place in uncontrolled Swiss airspace, within a volume of $22 \times 22 \times 3.2$ km. To minimize risk, they were conducted in the sparsely populated,

primarily agricultural Seeland region in the northwest of Switzerland.

RL POLICY TRAINING

The simulation physical properties (diameter, mass, drag coefficient, and actuation force) were adapted to the outdoor prototype, and the decision interval Δ_t during training was changed to 180 s. This 3D environment was confined within a volume of $22 \times 22 \times 3.2$ km.

EXPERIMENT

Testing in the real world illustrates a few challenges in the setup. First of all, the target cannot be placed via a random rollout but has to be set manually. This was done by rolling out a flight path in simulation based on a real-time NWP that does not violate any of the no-fly zones and allows landing in a nonpopulated empty region. Then, the balloon was filled and balanced (with small weights), and the pressure sensor was calibrated against the terrain model and local base pressure. Once communication and localization checks were complete, the balloon was released. A car was necessary to follow the prototype, which can reach velocities of 60 km/h^{-1}. Tests were terminated by manually overriding the RL algorithm via SMS to land the balloon once it passed the target. The decision interval during testing was set to 60 s. To increase reproducibility, the battery was fully charged before every flight.

When the prototype receives a landing command, negative thrust is applied until, roughly 150 m above the ground, the motors are shut off, and the system slowly descends. This avoids spinning the propellers on the ground, which could lead to damage and injury.

Multiple fail-safes are implemented to avoid losing the craft: if the prototype runs out of bounds, loses communication for a certain time, or reaches an overall time limit, the landing procedure is initiated. Fail-safe time parameters were set in inverse proportion to the maximum wind speed on the testing day.

The results for each of the six trials are shown in Table 2. These tests demonstrated the advantages of RL in such a

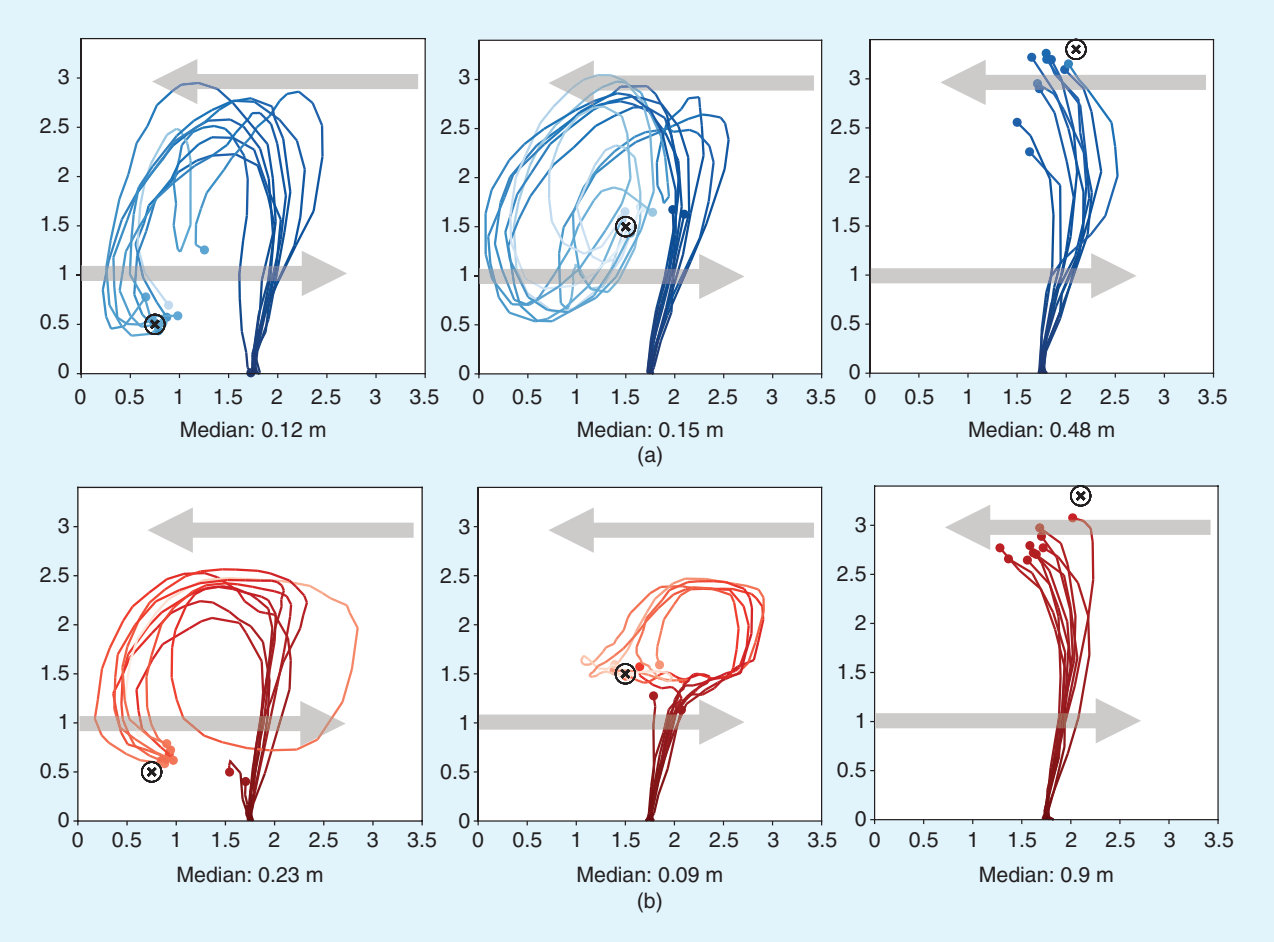

FIGURE 7. The 2D projection of the trajectory of the indoor testing series in a fixed-wind scenario (gray arrows). The color change in the trajectory represents time, and the endpoint shows the closest point of the trajectory to the target, marked by the black cross. The circle around the black cross represents the acceptance radius that was used during training. (a) RL has the overall smaller mean (30 cm) and median distance (23 cm), outperforming (b) the baseline (mean: 44 cm; median: 32.5 cm).

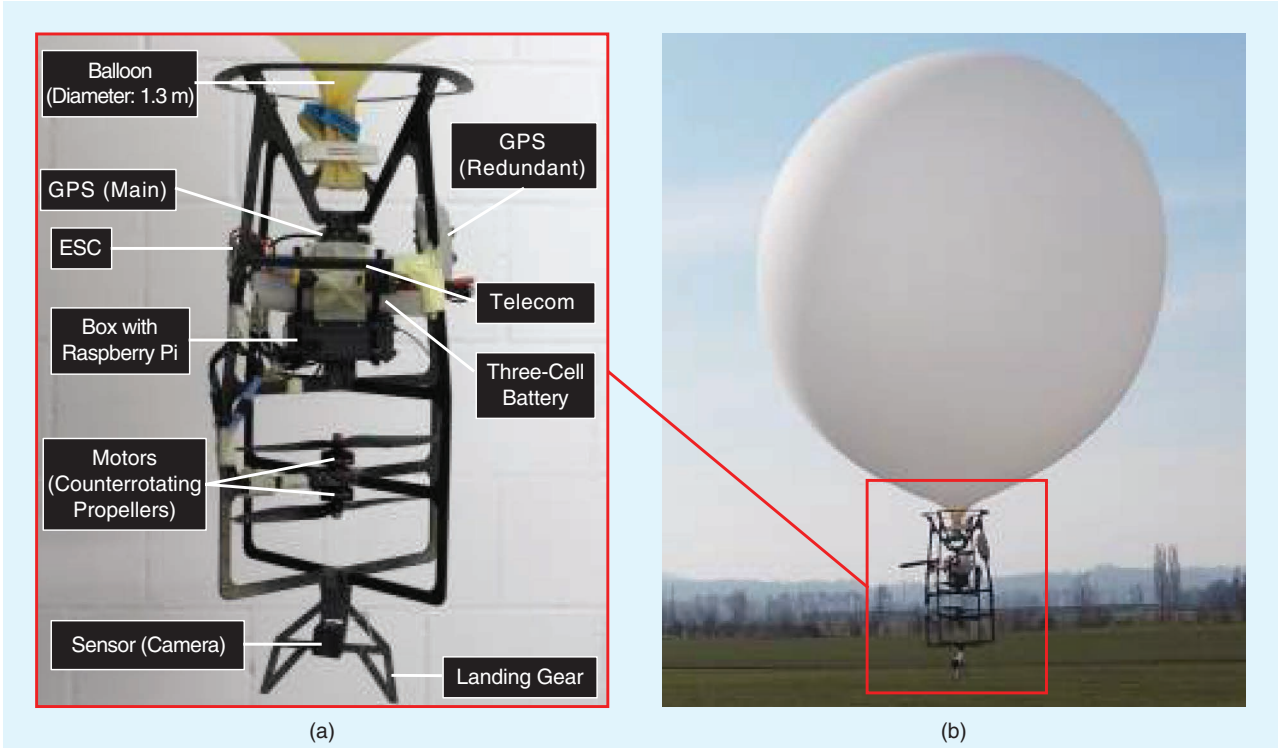

FIGURE 8. (a) The outdoor prototype consists of a carbon sheet structure held together by 3D-printed parts. A Raspberry Pi ensures communication via short message service; positioning is done via a global navigation satellite system and pressure sensor. The prototype is powered by a three-cell battery, isolated and heated by bubble wrap and chemical heating pads. (b) All this weighs a total of 1.2 kg and is lifted by a latex weather balloon filled with helium, 1.3 m in diameter. A camera is added as an exemplary sensor payload.

context. Even though the precomputed wind model was different from the actual wind conditions, the algorithm adapted the prediction based on local wind estimates and still succeeded at getting very close to the target in four out of six test cases. The remaining two targets were not reachable, based on the wind measurements collected during the flight. The predicted winds used in simulation to set the target positions differed significantly from the winds measured during flight, rendering the simulated trajectory infeasible.

A detailed analysis of the second and fourth flights appears in Figures 9 and 10. Despite the NWP being inaccurate in the second flight, the algorithm corrected its altitude accordingly and still managed to reach a point close to the target. However, during the fourth flight, the discrepancy of the NWP to the measurement was so large that there was almost no wind blowing into the negative y direction, making the target unreachable. In cases like this, the algorithm tends to stay at a constant height to avoid wasting energy.

These experiments show that the learned policy can be applied without adaptation on a real-world environment with a significant difference between the NWP and measured wind. The prototype proved to be reliable and allowed for consistent testing. Although six flights are not enough to statistically analyze the performance in real-world environments, it nevertheless demonstrates a proof of concept.

SENSITIVITY ANALYSIS

In another simulated experiment, we provided the agent with noisy input data to test the sensitivity with regard to input noise and analyze the information value of the different inputs to the agent. We added noise to the input data by rotating the measurements with a random uniformly sampled rotation. In the case of the wind predictions, this was done once per experiment, while the wind measurements and agent velocity were disturbed every

TABLE 2. The performance of RL with the precomputed wind model tested in the outdoor environment.

NUMBER	REACHABLE	DISTANCE	DURATION	MINIMUM RESIDUAL
1	Yes	9.8 km	76 min, 38 s	142 m
2	Yes	10.4 km	33 min, 31 s	536 m
3	Yes	13.6 km	48 min, 30 s	484 m
4	No	8.7 km	65 min, 57 s	8,154 m
5	No	13.1 km	36 min, 49 s	914 m
6	Yes	8.9 km	39 min, 52 s	275 m

Across all reachable cases, the prototype passed within an average minimum distance of 360 m to the target, with an average path length of 10.7 km.

time stamp to model disturbances and wind gusts. We tested different noise levels with rotations up to 30/90°, respectively. In the last experiment group, we disturbed the measurements with rotations larger than 90°, essentially rendering them completely uninformative, as they tend to be opposite to the true values. The results presented in Table 3 suggest that the agent heavily relies on the wind predictions, as the performance significantly drops with increased noise levels and fails when provided with invalid data. The agent seems to handle better the velocity noise and wind estimation noise but also relies less on these properties. The high success rate for the highly disturbed wind estimates suggests that the agent does not rely on this property.

CONCLUSION AND OUTLOOK

This work demonstrated a successful controller for an autonomous balloon flying from a start point to a predefined target, using RL. The agent tries to get within a certain range of the target as quickly as possible using a minimal amount of energy based on the relative target position, balloon local position, velocity, and global wind.

To show this, the SAC method was trained in parallel for 60 h on the outdoor case, reaching the target in 75.3% of the test cases in simulation. An indoor prototype along with two wind generators were built to evaluate the performance in a 2D indoor setting. Over three different scenarios of 10 episodes each, the prototype reached a median distance of 23.4 cm from the target, using the RL policy trained in simulation without adaptation. A fully autonomous outdoor prototype was developed to validate the algorithm in an open-world environment. Over all cases where the target was reachable, the system reached a mean minimum distance of 360 m to the target, after flying 10.7 km on average.

Both hardware tests showed that the trained policies are relatively robust to disturbances and poor initial estimates of the predicted global wind and measured local wind. This is mainly due to a realistic noise model that was used during training as well as using an EKF for accurate state estimation.

To allow the prototype to be used as a fully fledged sampling platform for collecting atmospheric data, the following improvements could be made. First, instead of handling NWP and measurements separately in the state space and leaving it up to the RL agent to mix them together, the NWP could be updated based on a weather model through the measurements. This would make the learning easier while also allowing past measurements to have an impact. One way to achieve this could be through predicting high-resolution wind fields based on terrain and real-time wind measurements [20].

Second, more real-life flight data could be collected to create a large database of observations. This database could then be used directly for training instead of generating observations in simulation. This would allow training to be very similar to the test case, characterizing the accuracy of the wind predictions.

Finally, currently, the time dependency of the wind prediction is not

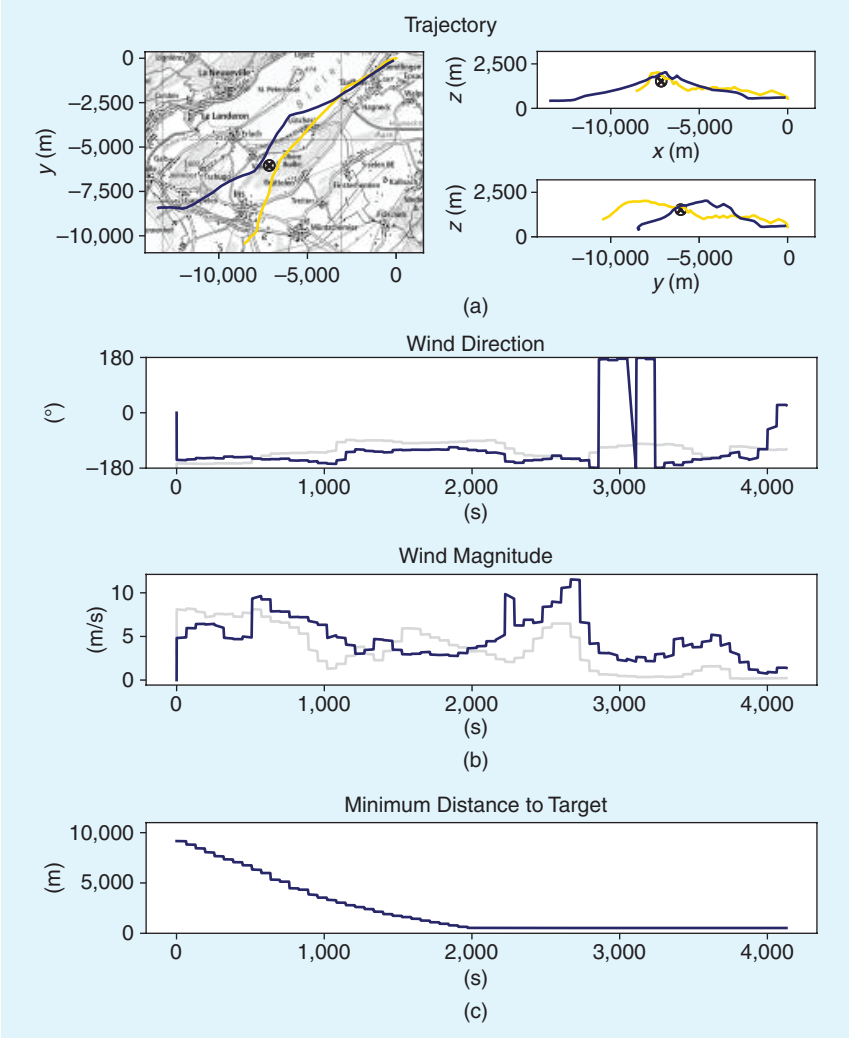

FIGURE 9. (a) The real-life trajectory (blue) and simulated trajectory (yellow) from the start at (0, 0) to the target (black cross), with the acceptance radius used during training (black circle). (b) The real-life wind measurement (blue) compared to the NWP (gray). (c) The minimum distance to the target during the experiment. The second flight (blue) travels 10.4 km to get as close as 536 m to the target. On this windy day, a clear change in direction and, therefore, altitude was necessary to reach the target. The simulated trajectory (light blue) is quite different from the executed trajectory, due to an offset in the NWP in both direction and magnitude. During this flight, the logging frequency was lower than in the other tests, due to a mistake in the setup.

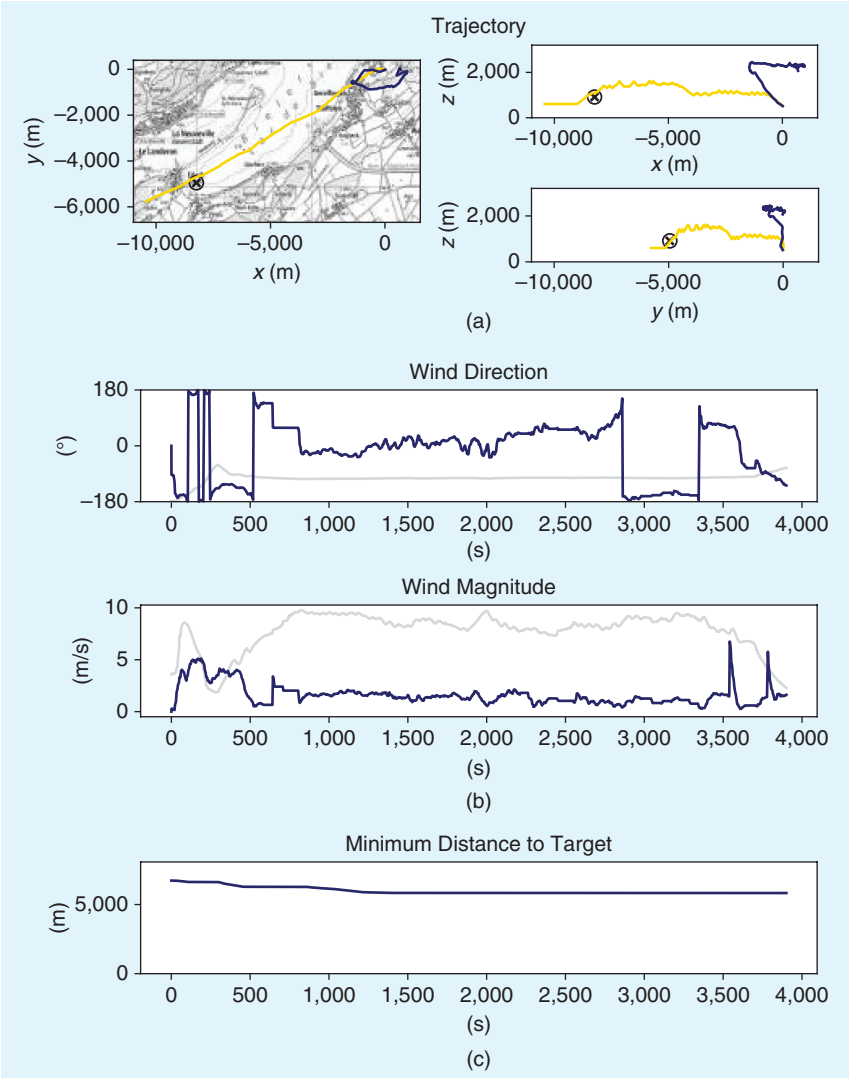

FIGURE 10. The fourth flight took well over an hour, but the prototype (blue) failed to get close to the target, regardless of passing through all altitudes, checking for wind blowing in the negative y direction. This is due to a significant inaccuracy in the directional NWP (gray) compared to the ground truth (blue). Hence, the target was not reachable. (a) The real-life trajectory (blue) and simulated trajectory (yellow) from the start at (0, 0) to the target (black cross), with the acceptance radius used during training (black circle). (b) The real-life wind measurement (blue) compared to the NWP (gray). (c) The minimum distance to the target during the experiment.

TABLE 3. The disturbance generated through uniformly random rotation of <30°, <90°, and >90°.

STATE ENTRY	NO NOISE	<30°	<90°	>90°
M	75.3%	48%	21.2%	2.6%
$\mathbf{w}_{xy}$	75.3%	65.8%	64.3%	59.1%
$\dot{\mathbf{r}}_b^{\mathcal{I}}$	75.3%	62.6%	51.3%	37.6%

The resulting success rate shows a clear dependency on the wind prediction and velocity. Wind measurement and velocity correlate very closely due to the large cross-section area and low inertia of the system. This sensitivity analysis shows that the algorithm favors velocity over wind measurement, likely due to there being less noise in the velocity measurement.

encoded, as we did not conduct test flights longer than 1.5 h. For longer tests, this time dependency could be encoded and passed into the state space. Additionally, the NWP could be updated on a regular basis in flight over the 4G network, instead of loading it only once before takeoff.

ACKNOWLEDGMENT

This work was supported, in part, by Engineering and Physical Sciences Research Council award EP/R009953/1 (CASCADE), the European Union Horizon 2020 AeroTwin project (grant ID 810321), a Royal Society Wolfson Fellowship (RSWF\R1\180003), and the Horizon 2020 research and innovation program, under grant agreement 730994.

The article has the following supplementary material:

- *video of indoor experiments:* https://youtu.be/DM-zIdogpts
- *video of outdoor experiments:* https://youtu.be/NxpNbGzozRk
- *code:* https://github.com/simonjeger/project_balloon
- *flight data and CAD files:* https://ieee-dataport.org/documents/experimental-data-autonomous-ballooning.

AUTHORS

Simon L. Jeger, Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland, and Laboratory of Intelligent Systems, EPF Lausanne, 1015 Lausanne, Switzerland. E-mail: simonjeger@gmail.com.

Nicholas Lawrance, Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland, and Robotics and Autonomous Systems Group, Brisbane, QLD 4069 Australia. E-mail: nicholas.lawrance@csiro.au.

Florian Achermann, Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland. E-mail: florian.achermann@mavt.ethz.ch.

Oscar Pang, Aerial Robotics Lab, Imperial College London, SW7 2AZ London, U.K. E-mail: k.pang@imperial.ac.uk.

Mirko Kovac, Aerial Robotics Lab, Imperial College London, SW7 2AZ London, U.K., and Laboratory of Sustainability Robotics, Swiss Federal Laboratories of Material Science and Technology, 8600 Dübendorf, Switzerland. E-mail: mirko.kovac@empa.ch.

Roland Y. Siegwart, Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland. E-mail: rsiegwart@ethz.ch.

REFERENCES

[1] M. G. Bellemare et al., "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, Dec. 2020, doi: 10.1038/s41586-020-2939-8.

[2] D. Vignelles et al., "Balloon-borne measurement of the aerosol size distribution from an Icelandic flood basalt eruption," *Earth Planet. Sci. Lett.*, vol. 453, pp. 252–259, Nov. 2016, doi: 10.1016/j.epsl.2016.08.027.

[3] N. Bryan, M. Stewart, D. Granger, T. Guzik, and B. Christner, "A method for sampling microbial aerosols using high altitude balloons," *J. Microbiological Methods*, vol. 107, pp. 161–168, Dec. 2014, doi: 10.1016/j.mimet.2014.10.007.

[4] A. Lee, D. Hanlon, R. Sakai, V. Morris, B. Demoz, and S. A. Gadsden, "Development of an autonomous unmanned aerial system for atmospheric data collection and research," in *Proc. Adv. Environ., Chem., Biol. Sens. Technol. XIII (SPIE),* 2016, vol. 9862, pp. 26–33, doi: 10.1117/12.2224547.

[5] L. Laakso et al., "Hot-air balloon as a platform for boundary layer profile measurements during particle formation," *Boreal Environ. Res.*, vol. 12, pp. 279–294, Jun. 2007.

[6] J.-L. Robyr, V. Bourquin, D. Goetschi, N. Schroeter, and R. Baltensperger, "Modeling the vertical motion of a zero pressure gas balloon," *J. Aircr.,* vol. 57, no. 5, pp. 991–994, Sep. 2020, doi: 10.2514/1.C035890.

[7] J. Blamont, "Planetary balloons," *Exp. Astron.,* vol. 22, no. 1–2, pp. 1–39, Oct. 2008, doi: 10.1007/s10686-008-9095-8.

[8] K. Garg, "Autonomous navigation system for high altitude balloons," Ph.D. thesis, Luleå Tekniska Univ., Luleå, Sweden, 2019.

[9] H. Du, M. Lv, J. Li, W. Zhu, L. Zhang, and Y. Wu, "Station-keeping performance analysis for high altitude balloon with altitude control system," *Aerosp. Sci. Technol., vol.* 92, pp. 644–652, Sep. 2019, doi: 10.1016/j.ast.2019.06.035.

[10] Y. T. Liu, E. Price, P. Goldschmid, M. J. Black, and A. Ahmad, "Autonomous blimp control using deep reinforcement learning," 2021, *arXiv:2109.10719.*

[11] A.-I. Toma, H.-Y. Hsueh, H. A. Jaafar, R. Murai, P. H. Kelly, and S. Saeedi, "Pathbench: A benchmarking platform for classical and learned path planning algorithms," in *Proc. 18th Conf. Robots Vision (CRV),* 2021, pp. 79–86, doi: 10.1109/CRV52889.2021.00019.

[12] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS),* 2016, vol. 29, pp. 3675–3683.

[13] Y. Fujita, P. Nagarajan, T. Kataoka, and T. Ishikawa, "ChainerRL: A deep reinforcement learning library," *J. Mach. Learn. Res., vol.* 22, no. 1, pp. 3557–3570, Jan. 2021.

[14] M. Roderick, J. MacGlashan, and S. Tellex, "Implementing the deep Q-network," 2017, *arXiv:1711.07478.*

[15] S. L. Jeger, "RL-based navigation for balloons in wind," Ph.D. thesis, ETH Zürich, Zürich, Switzerland, 2021.

[16] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2018, *arXiv:1803.08375.*

[17] M. Standard, "Flying qualities of piloted aircraft," U.S. Department of Defense, Washington, DC, USA, MIL-STD-1797A, 1990.

[18] M. Cavcar, "The international standard atmosphere (ISA)," Anadolu Univ., Eskişehir, Turkey, 2000. [Online]. Available: http://servidor.demec.ufpr.br/foguete/Festival2018/Minicursos/Trajet%C3%B3ria/The%20International%20Standard%20Atmosphere%20(ISA)%20-%20Documento%20n%C3%A3o%20oficial.pdf

[19] P. B. Voss, E. E. Riddle, and M. S. Smith, "Altitude control of long-duration balloons," *J. Aircr., vol*. 42, no. 2, pp. 478–482, Mar. 2005, doi: 10.2514/1.7481.

[20] F. Achermann, N. R. Lawrance, R. Ranftl, A. Dosovitskiy, J. J. Chung, and R. Siegwart, "Learning to predict the wind for safe aerial vehicle planning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA),* 2019, pp. 2311–2317, doi: 10.1109/ICRA.2019.8793547.