Learning to Seek: Autonomous Source Seeking with Deep Reinforcement Learning Onboard a Nano Drone Microcontroller

Bardienus P. Duisterhof[1,3] Srivatsan Krishnan[1] Jonathan J. Cruz[1] Colby R. Banbury[1] William Fu[1]

Aleksandra Faust[2] Guido C. H. E. de Croon[3] Vijay Janapa Reddi[1]

***Abstract*—We present fully autonomous source seeking onboard a highly constrained nano quadcopter, by contributing application-specific system and observation feature design to enable inference of a deep-RL policy onboard a nano quadcopter. Our deep-RL algorithm finds a high-performance solution to a challenging problem, even in presence of high noise levels and generalizes across real and simulation environments with different obstacle configurations. We verify our approach with simulation and in-field testing on a Bitcraze CrazyFlie using only the cheap and ubiquitous Cortex-M4 microcontroller unit. The results show that by end-to-end application-specific system design, our contribution consumes almost three times less additional power, as compared to competing learning-based navigation approach onboard a nano quadcopter. Thanks to our observation space, which we carefully design within the resource constraints, our solution achieves a 94% success rate in cluttered and randomized test environments, as compared to the previously achieved 80%. We also compare our strategy to a simple finite state machine (FSM), geared towards efficient exploration, and demonstrate that our policy is more robust and resilient at obstacle avoidance as well as up to 70% more efficient in source seeking. To this end, we contribute a cheap and lightweight end-to-end tiny robot learning (tinyRL) solution, running onboard a nano quadcopter, that proves to be robust and efficient in a challenging task using limited sensory input.**

***Index Terms*—Motion and Path Planning, Aerial Systems: Applications, Reinforcement Learning**

I. Introduction

Source seeking is an important application for search and rescue, inspection, and other jobs that are too dangerous for humans. Imagine cheap and disposable aerial robots inspecting ship hauls for leaks, aiding search for survivors in mines, or seeking a source of radiation in nuclear plants. For that reality, we need small, agile, and inexpensive robots capable of fully-autonomous navigation in GPS denied environments that can be deployed quickly, without additional set-up or training.

Nano quadrotors are a lightweight, cheap, and agile hardware platform, and an ideal candidate for source seeking in GPS denied environments. To make them work, we need to add appropriate sensors and navigation software. However, they are severely resource constrained. Their available memory, battery, and compute power is limited. Those constraints pose challenges to the existing autonomous navigation methods, and the sensor and software selection needs to be carefully designed. The memory constraints means the system cannot store large maps used in traditional planning, the battery constraints means that we need to consider energy consumption of the system [1], and the limited compute power means that large neural networks cannot run.

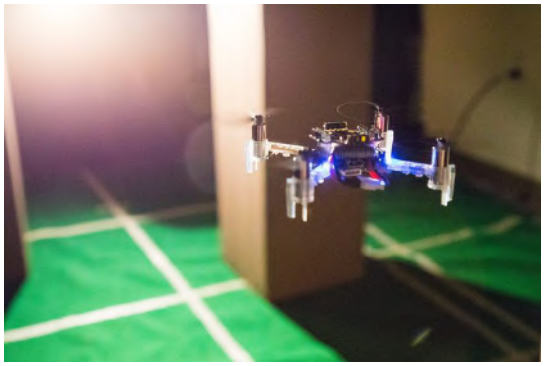


Fig. 1. CrazyFlie nano quadcopter running a deep reinforcement learning policy fully *onboard* with robust obstacle avoidance and source seeking.

Source seeking applications needs motion planning capable of obstacle avoidance that can be deployed quickly, without apriori knowledge of the obstacle placement, without the need to configure or re-train for a specific condition or environment. Instead, the algorithms we deploy need to generalize to a wide range of unknown environments for effective deployment. The currently available finite state machines on nano quadcopters [2] are designed for certain conditions and environments, and may not generalize well. Traditional navigation solutions like SLAM cannot run, as they require more memory and compute than our nano quadcopter can offer. Instead, we need a mapless navigation solution that uses little sensory input, memory and compute, while also generalizing to different environments.

To address these challenges, we present a fully autonomous source-seeking nano quadcopter (Figure 1), by equipping nano UAVs with sparse light-weight sensors, and using reinforcement learning (RL) to solve the problem. We contribute POMDP formulation including source observation features, and sensor modelling. Our method operates under memory, compute, and energy constraints with zero-shot transfer from

[1]Harvard University, [2] Robotics at Google, [3]Delft University of Technology - bduisterhof@g.harvard.edu. The work was done while Bart was a visiting student at Harvard.

(a) Air Learning training env. (b) 0 obstacles. (c) 3 obstacles. (d) 7 obstacles.

Fig. 2. Air Learning training environment (2a), and three distinct trajectories in flight tests (2b-2d). Blue dot is the start and green dot is the destination.

simulation to reality, and generalizes to new environments without prior training. We fit an ultra-tiny neural network with two hidden layers, which runs at 100 Hz on a commercially available off-the-shelf ARM Cortex-M4 microcontroller unit (MCU). Our source-seeking nano quadcopter, shown in Figure 1, seeks a light source as a proxy for any point source, such as a radiation source. We use a light source as it allows for better understanding of behavior in real flight tests. Our nano quadcopter locates the light source in challenging conditions, with varying obstacle counts and positions, even beyond the environments presented in simulation (Figure 2a). The robot first explores, when it is far away from the source and it detects no gradient in light but just noise. It then seeks reactively for the light source when it detects gradients (Figure 2).

We compare our work with baselines in three areas: 1) nano quadcopter system design, 2) source seeking performance and 3) obstacle avoidance robustness. From a systems perspective, the state of the art solution for learning-based navigation onboard a nano quadcopter [3] consumes almost three times more additional power. From a source seeking perspective, the state of the art RL light seeking approach [4] reaches an 80% success rate in 'simple' simulated environments, and 30% in 'complex' simulated environments, which we outperform by reaching a 94% success rate in real flight tests in cluttered and randomized test environments (Figure 2b-2d). Finally, we compare against a finite state machine (FSM) geared towards exploration, which we outperform by more robust obstacle avoidance and efficient source seeking. Hereby, we show that our simulator-trained policy provides a high-performance and robust solution in a challenging task in on-robot experiments, even in environments beyond those presented in simulation.

II. Related Work

Deep reinforcement learning has proven to be a promising set of algorithms for robotics applications. The fast-moving deep reinforcement learning field is enabling more robust and accurate but also more effortless application [5]. Lower-level control has been demonstrated to be suitable to replace a rate controller [6]–[8] and was recently used to perform model-based reinforcement learning (MBRL) with a CrazyFlie [9]. High-level control using deep reinforcement learning for obstacle avoidance has been shown with different sets of sensory input [10], [11], but not yet on a nano quadcopter. Although light seeking has been demonstrated before using Q-learning [4], it used multiple light sensors and reached a success rate of 80% in simple environments and 30% in complex environments. Thanks to our observation space design and larger network, *we present a deep reinforcement learning-based model for robust and efficient end-to-end navigation of a source seeking nano quadcopter, including obstacle avoidance, that beats the current state of the art.*

Traditional source seeking algorithms can be divided into four categories [12]: 1) gradient-based algorithms, 2) bio-inspired algorithms, 3) multi-robot algorithms, and 4) probabilistic and map-based algorithms. Even though gradient-based algorithms are easy to implement, their success in source seeking has been limited due to their unstable behavior with noisy sensors. Previous algorithms have yielded promising results, but rarely considered obstacles [13]. Obstacles are important as they make the problem harder, not just from an avoidance perspective, but also considering phenomena like shadows. *We contribute a deep RL approach, capable of robust source seeking and obstacle avoidance on a nano drone.*

In a multi-agent setup, particle swarming [14], [15] has shown to be successful in simulation. However, PSO swarms require a positioning system, and lack laser-based obstacle avoidance. Finally, probabilistic and map-based algorithms are more flexible but require high computational cost and accurate sensory information. In contrast to traditional methods, deep reinforcement learning can learn to deal with noisy inputs [16] and effectively learn (optimal) behavior for a combination of tasks (i.e., generalize). Hence, source seeking on a nano quadcopter is a suitable task for deep reinforcement learning, as it can combine obstacle avoidance with source seeking and deal with extraordinary noise levels in all sensors. *We leverage these advantages of deep reinforcement learning to produce a robust and efficient algorithm for mapless navigation for source seeking.*

III. Method

We present our application-specific system design (Section III-A), using lightweight and cheap commodity hardware. Next, we describe our simulation design (Section III-B), used to train deep-RL policies by randomizing the training environment and deploying a source model. Finally, in Section III-C, we show the POMDP formulation.

A. System Design

We configure a BitCraze CrazyFlie nano quadcopter for source seeking, taking into account its physical and computational limitations, as visible in Table I. While adding a camera may be useful for navigation, its added weight and cost make effective deployment more difficult. Instead, we

Developer	Bitcraze	Parrot	Delta
Vehicle	CrazyFlie 2.1	Bebop 2	
Takeoff weight	27 g	500 g	18.5 x
Max payload	15 g	70 g	4.6 x
Battery (LiPo)	250 mAh	2700 mAh	10.8 x
Flight time	7 min	25 min	3.6 x
Size (WxHxD)	9.2 cm x 9.2 cm	32.8 cm x 38.2 cm	12.7 x

TABLE I
CRAZYFLIE VS. BEBOP2. THE DELTA BETWEEN THEM IS SIGNIFICANT.

use only cheap, lightweight and robust commodity hardware. We configure our nano quadcopter with laser rangers for obstacle avoidance, an optic flow sensor for state estimation, and a custom light sensor to seek a light source (Figure 3). The nano quadcopter carries four laser rangers with a range of approximately 5 m, facing in the negative and positive direction of the x and y axis of the robot' body frame. We attach an ultra-tiny PCB to the laser ranger board, fitting an upward-facing TSL2591 light sensor.

B. Simulation Environment

Source modelling is key to the success of our nano quadcopter. We model source intensity as a function of the distance from the source. We generate this function by capturing data in the testing environment with the light source present. We capture the light intensity in a two-dimensional grid with our light sensor. Once captured, we use the data to fit a function with two requirements: 1) $\lim_{dist\to 0} < \infty$ and 2) $\lim_{dist\to\infty} = 0$. A Gaussian function meets both requirements and is shown in Figure 5. The function has the form: $f(x) = a \cdot e^{-\frac{(x-b)^2}{2c^2}}$ with $a = 399.0, b = -2.6, c = 5.1$. The R-squared error, measuring the goodness-of-fit, is 0.007, implying a high-quality fit. Additionally, we inject Gaussian noise with a standard deviation of 4. The noise observed in recordings had a standard deviation of 2; however, in flight with unstable attitude, we expect more noise, so we inject more noise than recorded. In flight tests (Section V-D), we present the robustness of this function when shadows and reflections are present.

C. POMDP Setup

To learn to seek a point source, we choose reinforcement learning with partial state observations, which we model as a Partially Observable Markov Decision Process (POMDP). The agents is modeled as a POMDP by the tuple (O, A, D, R, γ) with continuous observations and discrete actions. We train and deploy a DQN [17] algorithm, using a feedforward network with two hidden layers of 20 nodes each and activations.

The observations, $\mathbf{o} = (l_1, l_2, l_3, l_4, s_1, s_2) \in O$, consist of four laser ranger values in front/right/back/left directions , l_1-l_4,, and two custom 'source terms', s_1 and s_2. The source terms are inspired by [18], and provide an estimate for source gradient and strength. We first compute a normalized version of the light sensor readings c, as sensor readings are dependent on sensor settings (e.g., integration time, gain). We then add a low-pass filter and compute c_f (Equation 1). We then compute s_1 (Equation 2), which is effectively a normalized and low-pass version of the gradient of c (i.e., it is the light gradient over time). Finally, we compute s_2 (Equation 3), a transformation of c_f.

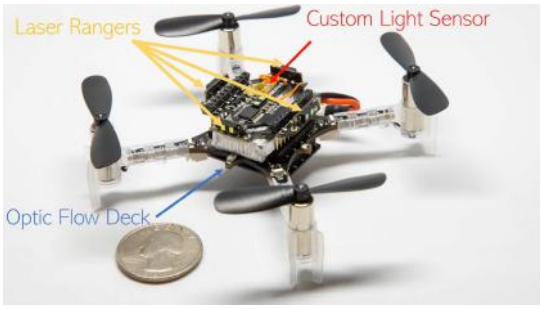


Fig. 3. Our BitCraze CrazyFlie with multiranger deck, custom TSL2591 light sensor board and an optic flow deck.

$$c_f \leftarrow 0.9 \cdot c_f + 0.1 \cdot c \quad (1) \qquad s_1 = \frac{c - c_f}{c_f} \quad (2) \qquad s_2 = 2 \cdot c_f - 1 \quad (3)$$

Figure 10 shows traces of all variables onboard a source-seeking nano quadcopter. Term s_1 is effectively a normalized and low-pass version of the gradient of c (i.e., it is the light gradient over time). The low-pass filter has a high cutoff frequency, but is useful in filtering outliers. Normalization is crucial when far away from the source, when the gradient is small and finding the source is hard. Term s_2 is a transformation of c_f. The signal provides the agent with a proxy of distance to the source, which it can use to alter behavior.

We have trained policies with non-filtered gradient inputs, leading to poor results (30% succes rate). While the direct gradient only contains information about the last time-step, c_f is the weighed sum of the entire history of measurements. It is a computationally cheap way of using history information, which is necessary as the fluctuations between two time-steps (0.01 s) are noise-dominated.

To teach the nano quadcopter to seek the source, the reward is computed at each step (instantaneous reward):

$$r = 1000 \cdot \alpha - 100 \cdot \beta - 20 \cdot \Delta D_s - 1 \qquad (4)$$

Here, α is a binary variable whose value is '1' if the agent reaches within 1.0 m from the goal else its value is '0'. β is a binary variable which is set to '1' if the nano quadcopter collides with any obstacle or runs out of the 300 steps.[1] Otherwise, β is '0', penalizing the agent for hitting an obstacle or not finding the source in time. ΔD_s is the change in distance, compared to the previous step. A negative ΔD_s means the agent got closer to the source, rewarding the agent to move to the source.

The agent's discrete actions space, $\mathbf{a} = \left(v_x, \dot{\psi}\right)^3 \in A$, consists of three pairs of target states, composed of target yaw rate and the target forward velocity (v_x). The target states are then realized by the low-level PID controllers. The three

[1]We set the maximum allowed steps in an episode as 300. This is to make sure the agent finds the source within some finite amount of steps and penalize a large number of steps.

actions are: move `forward`, rotating `left` or `right`. The forward-moving speed is $0.5\,\mathrm{m/s}$, and the yaw rate is 54°/s in either direction. Finally, the dynamics D of the environment are a simple drone model developed by AirSim [19] and γ was set to be 0.99.

IV. Implementation Details

We discuss the implementation details of our simulation environment (Section IV-A), and inference of the deep-RL policy onboard the nano quadcopter (Section IV-B)

A. Simulation

We simulate an arena to train an agent to seek a point source. The agent (i.e., drone) is initialized in the middle of the room, and the point source is spawned at a random location. By randomizing the source position and obstacle positions, we arrive at a policy that generalizes to different environments with different obstacle configurations. We use the Air Learning platform [11], which couples with Microsoft AirSim [19] to provide a deep reinforcement learning back end. It generates a variety of environments with randomly varying obstacle density, materials, and textures.

B. Inference

The CrazyFlie [20] is heavily constrained, carrying an STM32F405 MCU. Though our MCU is constrained in memory and compute, it is widely deployed. In fact, more than 30 billion general-purpose MCUs are shipped every year.[2] Their ubiquity makes them cheap, easy to use and expendable, and so suitable for cheap and disposable search and rescue robots.

Our implementation consists of a custom lightweight C library, capable of performing the necessary inference operations. The advantage of this approach is its small memory footprint and overhead, compared to a general inference framework like TensorFlow Lite for Microcontrollers [21].

The Crazyflie has 1 MB of flash storage. The stock software stack occupies 192 kB of the available storage, while the custom source seeking stack takes up an additional 6 kB. So the total flash storage used is 198 kB, which leaves an ample amount of free storage space (over 75%).

However, the memory constraints are much more severe. RAM availability during execution is shown in Figure 4. Of the 196 kB of RAM available on the Cortex-M4 microcontroller, only 131 kB is available for static allocation at compile time. The rest is reserved for dynamic variables (i.e., heap). During normal operation, the Bitcraze software stack uses 98 kB of RAM, leaving only 33 kB available for our purposes. The entire source seeking stack takes up 20.5 kB, leaving 12.5 kB of free static memory. Our policy runs at 100 Hz in flight.

V. Results

We evaluate simulation and flight results of our models. We evaluate training results (Section V-A), introduce our baselines (Section V-B) and evaluate the models in simulation (Section V-C) and flight tests (Section V-D). Finally, we analyze robot behavior (Section V-E) and endurance and power consumption (Section V-F).

[2]From IC Insights, Research Bulletin.

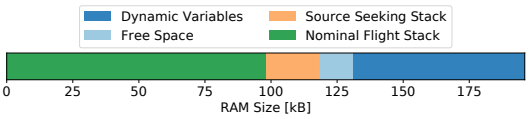


Fig. 4. RAM usage on the Bitcraze CrazyFlie, using a custom float inference stack. Total free space: 12.5 kB

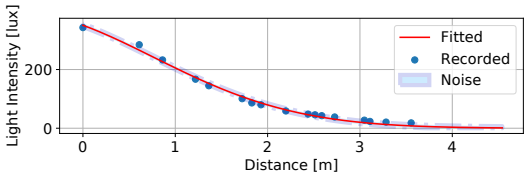


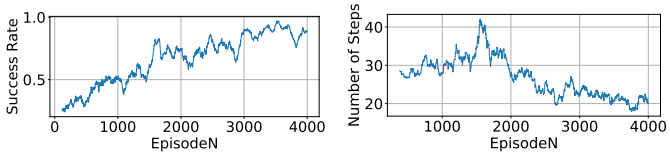
Fig. 5. Light intensity describing the function as used in training with 3σ (standard deviations) of the noise.

A. Training in Simulation

To evaluate the learning process, we present quality metrics (success rate and number of steps) during training. As shown in Figure 6, we train our policy up to convergence at around 3,600 episodes (or 100,000 steps). A consistent upward trend in success rate is visible, while number of steps shows a consistent decrease after an initial spike. The initial spike is caused by the agent becoming more successful, hence reaching more distant targets, instead of only finding close targets. After continued training success rate quickly drops, i.e., it over-trains after around 3600 episodes. We continued training to over 8,000 episodes and never saw an improvement in performance.

B. Baseline Comparison

We compare our approach with three baselines: 1) nano quadcopter system design, 2) source seeking performance and 3) obstacle avoidance robustness. From a systems perspective, the state of the art solution for learning-based navigation onboard a nano quadcopter [3] adds almost three times more power consumption (Section V-F), showing end to end application-specific system design can benefit mission metrics.



(a) Training success rates. (b) Steps to locate source.

Fig. 6. Quality metrics during training.

Model description	Success	# of Steps	Distance [m]	SPL
Our deep RL algorithm	96%	30.51	4.21	0.37
FSM baseline	84%	87.73	10.40	0.16
Random actions	30%	42.13	3.55	0.13

TABLE II
Models in simulation.

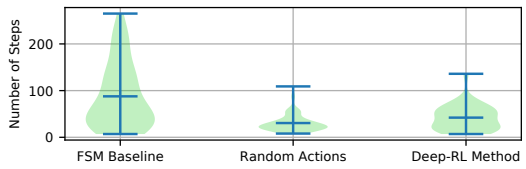


Fig. 7. Number of steps in simulation over 100 runs of each algorithm.

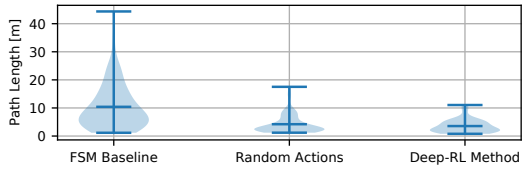


Fig. 8. Path length (i.e. traveled distance) in simulation over 100 runs.

Thanks to our low-dimensional sensory input, we perform inference at 100 Hz on the stock microcontroller processor.

From a source seeking perspective, the state of the art RL light seeking approach [4] reaches an 80% success rate in 'simple' simulated environments, and 30% in 'complex' simulated environments, which we beat by reaching a 94% success rate in real flight tests (Section V-D) in cluttered and randomized test environments. Our design of the observation space (i.e., the network inputs) has been critical in our success.

Finally, we compare against a finite state machine (FSM) geared towards exploration [22], which we outperform by more robust obstacle avoidance and more efficient source seeking, as shown in Table II. As demonstrated in [22], this approach is effective in exploration and often used in autonomous cleaning robots. This baseline serves to understand the difficulty of obstacle avoidance using limited sensory input, and to show our policy efficiently uses light information. We test it in the same simulation and real test environments, to put our approach into perspective. As a final baseline, we have also tested random actions, to show the effectiveness of the finite state machine (FSM). We cannot test the approach from [4], as, to the best of our knowledge, no public code is available.

C. Inference in Simulation

Training data provide limited information as the model is continuously changing. So, we evaluate our policy after training (as shown in Table II). We compare it in terms of success rate, the average number of steps, and average traveled distance. The number of steps and traveled distance are captured only when the agent succeeds. Additionally, we add the 'SPL' metric: Success weighted by (normalized inverse) Path Length [23]. We compute the SPL metric as:

$$SPL = \frac{1}{N} \sum S_i \frac{l_i - 1}{\max(p_i, l_i - 1)} \tag{5}$$

Here N is the number of runs considered, l_i the shortest direct path to the source, p_i the actual flown path and S_i a binary variable, 1 for success and 0 for failure. We subtract l_i by 1, as the simulation is terminated when the drone is within 1 meter of the source. We do not take into account obstacles in the path, making the SPL displayed a conservative estimate.

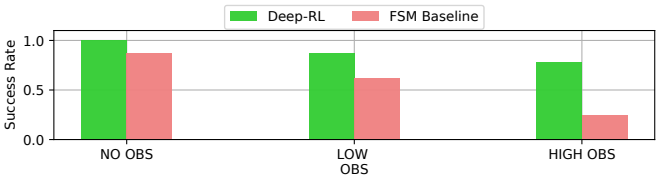


Fig. 9. Success rate over 104 flight experiments, comparing our deep reinforcement learning approach with the FSM baseline. Our solution consistently performs better, especially in high-density obstacle environments.

We evaluate our finite state machine (FSM) and fully random actions in simulation. Table II and Figures 7, 8 show the results of testing each method for 100 runs. Our deep reinforcement learning model outperforms the FSM baseline in every metric. It finds the source in 65% fewer steps, with a 14% higher success rate, with 60 % shorter paths and a 131% higher SPL. Random actions yield shorter successful paths, as shorter paths have a higher chance of survival with random actions. In fact, the average path length over all (successful and failed) attempts for the random approach is 5.7 m, while 4.19 m for our approach.

D. Flight Tests

We use a room that is approximately 5 m × 5 m in size (see Figure 2). We use a 50 W light source attached to the roof, radiating a 120° beam onto the ground, as the light source. We count a distance under 0.7 m as a successful run, while the drone is flying at 1 m/s and performs inference at 100 Hz. Figure 2 show four distinct trajectories during testing.

We conduct 114 flight tests in a variety of different scenarios, involving highly cluttered environments. Across a set of representative flight tests, we get an average success rate of 94%. This number is measured over a set of 32 experiments, 16 with no obstacles, and 16 with 3 obstacles in the room. This is representative to simulation as it alternates between no obstacles and sparse obstacles. All agents were initialized at different positions, at the same 4.6 m from the source on one axis. On the other axis, we varied drone position between 0 and 4.6 m from the source, resulting in an initial total source distance between 4.6 m and 6.5 m.

Obstacle configuration, density, and source position were randomized. We classify three distinct obstacle densities: 'NO OBS', 'LOW OBS', and 'HIGH OBS', featuring zero, three and seven obstacles respectively. To better understand the behavior of our algorithm, we decompose the results into two categories: 1) success rate and 2) mission time.

Over a total of 104 flight tests, we compared success rate of our model against the FSM baseline model. As can be seen in Figure 9, our model beats the baseline in all three obstacle density groups. The baseline reaches a 75% success rate in a set of representative flight tests ('NO OBS' and 'LOW OBS'), compared to a 84% success rate in simulation.

These results demonstrate that obstacle avoidance using solely a multiranger is challenging, as drift and limited vis-

ibility are the most prominent causes for crashing. In most crashes, the drone would either not see the obstacle or keep rotating close to an obstacle and eventually crash into the obstacle due to drift. The baseline serves to put our algorithm into perspective, i.e., it shows the robustness of our obstacle avoidance and source seeking strategy. A deteriorated success rate in the 'HIGH OBS' scenario is expected, as it has never seen such a scenario in simulation. *Despite some loss in success rate when adding more obstacles, our approach shows greater resilience to task complexity when compared to the baselines. It generalizes better than other baselines, even beyond the simulated environments, showing tinyRL's potential.*

Our objective is not only to perform successful obstacle avoidance, but also to find the source in as little time as possible. Nano drones are characterized by their limited battery life. Therefore, efficient flight is an important metric when evaluating the viability of an approach for real applications. Figure 11 shows the distribution of the mission time of successful runs, demonstrating an impressive advantage for our algorithm. Across the obstacle densities, from low to high, our policy was 70%, 55%, and 66% faster, respectively.

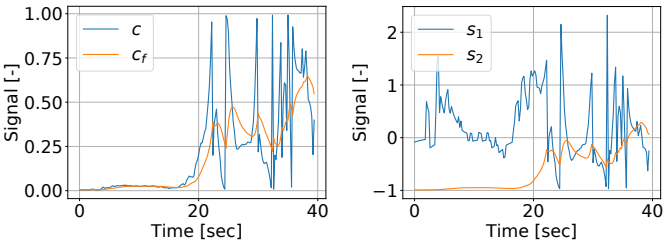
The baseline is again used to put our model into perspective. As demonstrated in [22], the FSM is effective in exploring an area without any source information. Because of its random character, the baseline shows a more even distribution of mission times. The deep reinforcement learning approach has a small number of outliers with high mission time, often caused by the agent getting stuck in a certain trajectory in the dark. As shown in Figure 10, the light gradient is limited far away from the source. The presence of noise makes it extremely hard for the agent to retrieve source information, at a great distance. We often observed a more direct path in the last $2.5\,\mathrm{m}$, compared to the initial exploration. *Our approach performs robust obstacle avoidance and efficient source seeking using only a tiny neural network.*

E. Behavior Analysis

To better understand the behavior of the nano quadcopter, we record source measurements and network inputs during flight. It can be seen that when far away from the source, extremely little information is present. In the first 20 seconds, almost no gradient is visible, forcing the agent to explore. The raw light readings are extremely noisy and unstable due to sensor noise, attitude changes and shadows. Once the agent 'sees' the source, it keeps traveling up the gradient and doesn't go back to the dark zone. Finally, the features work as imagined, s_1 is a normalized light gradient with a low-pass filter and s_2 is a transformation of c_f.

F. Endurance and Power comparison

We consider endurance as a performance metric for our solution. By performing source seeking on the CrazyFlie, we add weight and CPU cycles. We determine endurance in hover and compare a stock hovering CrazyFlie with our source seeking solution. We swap the multiranger deck with light sensor for the battery holder, lowering the weight from



(a) Normalized light measurements c and its low-pass version c_f. (b) Policy inputs s_1 and s_2, as described in Equations 2 and 3.

Fig. 10. Source measurements on the nano quadcopter while seeking the source. The first 20 seconds the sensor provides little information.

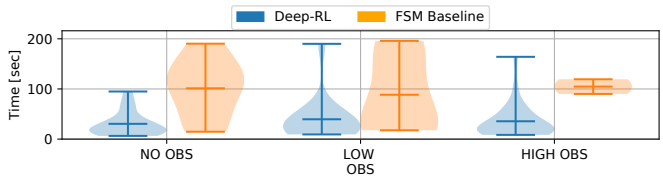


Fig. 11. Mission time in success over 104 flight experiments, comparing our deep RL approach with the FSM baseline.

$33.4\,\mathrm{g}$ to $31.7\,\mathrm{g}$ ($-1.7\,\mathrm{g}$). The endurance observed with the stock CrazyFlie is 7:06.3, while our solution hovers for 6:30.8, reducing endurance by $35.5\,\mathrm{s}$. With a battery capacity of 0.925 Wh, the average power consumption increased from $7.81\,\mathrm{W}$ to $8.52\,\mathrm{W}$(+$0.71\,\mathrm{W}$). It is expected that the vast majority of the extra power consumption comes from the extra weight, as the maximum consumption of the Cortex-M4 is $0.14\,\mathrm{W}$.

To put these numbers into perspective, we compare them to a CrazyFlie with a camera and additional compute [3]. As shown in [3], endurance is reduced by $100\,\mathrm{s}$ when adding the PULP-Shield, almost 3X more than in our experiments—*the state of the art methods for vision-based navigation on nano drones have large impact on endurance, and hence different sensors are worth investigating. TinyRL applications will likely make more use of sensors outside of the camera.*

VI. Discussion and Conclusion

We show that deep reinforcement learning can be used to enable autonomous source seeking applications on nano drones, using only general-purpose, cheap, commodity MCUs. We trained a deep reinforcement learning policy that is robust to the noise present in the real world and generalizes outside of the simulation environment. We believe our tinyRL methodology is useful in other real-world applications too, as robots learn to adapt to noise and other implicit information. As our policy was trained on a general point source model, we believe it will provide a high-performance solution for other sources, such as radiation. By simply swapping the sensor, the nano quadcopter may readily be deployed to seek other sources. Versatile software and hardware will be important when deploying these robots in the real world, while exploring unknown environments and seeking an unknown source.

REFERENCES

[1] B. Boroujerdian, H. Genc, S. Krishnan, W. Cui, A. Faust, and V. J. Reddi, "Mavbench: Micro aerial vehicle benchmarking," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-51. IEEE Press, 2018, p. 894–907. [Online]. Available: https://doi.org/10.1109/MICRO.2018.00077

[2] K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon, "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment," *Science Robotics*, vol. 4, no. 35, 2019. [Online]. Available: https://robotics.sciencemag.org/content/4/35/eaaw9710

[3] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, Oct 2019.

[4] S. Dini and M. A. Serrano, "Combining q-learning with artificial neural networks in an adaptive light seeking robot," 2012.

[5] A. Faust, A. Francis, and D. Mehta, "Evolving rewards to automate reinforcement learning," *CoRR*, vol. abs/1905.07628, 2019. [Online]. Available: http://arxiv.org/abs/1905.07628

[6] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *CoRR*, vol. abs/1707.05110, 2017. [Online]. Available: http://arxiv.org/abs/1707.05110

[7] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 2, pp. 22:1–22:21, Feb. 2019. [Online]. Available: http://doi.acm.org/10.1145/3301273

[8] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," *CoRR*, vol. abs/1903.04628, 2019. [Online]. Available: http://arxiv.org/abs/1903.04628

[9] N. O. Lambert, D. S. Drew, J. Yaconelli, R. Calandra, S. Levine, and K. S. J. Pister, "Low level control of a quadrotor with deep model-based reinforcement learning," *CoRR*, vol. abs/1901.03737, 2019. [Online]. Available: http://arxiv.org/abs/1901.03737

[10] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," *CoRR*, vol. abs/1902.03701, 2019. [Online]. Available: http://arxiv.org/abs/1902.03701

[11] S. Krishnan, B. Boroujerdian, W. Fu, A. Faust, and V. J. Reddi, "Air learning: An AI research platform for algorithm-hardware benchmarking of autonomous aerial robots," *CoRR*, vol. abs/1906.00421, 2019. [Online]. Available: http://arxiv.org/abs/1906.00421

[12] X. xing Chen and J. Huang, "Odor source localization algorithms on mobile robots: A review and future outlook," *Robotics and Autonomous Systems*, vol. 112, pp. 123 – 136, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889018303014

[13] J. R. Bourne, E. R. Pardyjak, and K. K. Leang, "Coordinated Bayesian-Based Bioinspired Plume Source Term Estimation and Source Seeking for Mobile Robots," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 967–986, 2019.

[14] R. Zou, V. Kalivarapu, E. Winer, J. Oliver, and S. Bhattacharya, "Particle swarm optimization-based source seeking," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 865–875, July 2015.

[15] L. Parker, J. Butterworth, and S. Luo, "Fly safe: Aerial swarm robotics using force field particle swarm optimisation," *CoRR*, vol. abs/1907.07647, 2019. [Online]. Available: http://arxiv.org/abs/1907.07647

[16] A. Faust, O. Ramirez, M. Fiser, K. Oslund, A. Francis, J. Davidson, and L. Tapia, "PRM-RL: long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," *CoRR*, vol. abs/1710.03937, 2017. [Online]. Available: http://arxiv.org/abs/1710.03937

[17] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[18] G. de Croon, L. O'Connor, C. Nicol, and D. Izzo, "Evolutionary robotics approach to odor source localization," *Neurocomputing*, vol. 121, pp. 481 – 497, 2013, advances in Artificial Neural Networks and Machine Learning. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213005869

[19] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *CoRR*, vol. abs/1705.05065, 2017. [Online]. Available: http://arxiv.org/abs/1705.05065

[20] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, Aug 2017, pp. 37–42.

[21] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev *et al.*, "Tensorflow lite micro: Embedded machine learning on tinyml systems," *arXiv preprint arXiv:2010.08678*, 2020.

[22] J. Palacin, T. Palleja, I. Valganon, R. Pernia, and J. Roca, "Measuring coverage performances of a floor cleaning mobile robot using a vision system," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 4236–4241.

[23] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," *CoRR*, vol. abs/1807.06757, 2018. [Online]. Available: http://arxiv.org/abs/1807.06757