UAV-based path planning for efficient localization of non-uniformly distributed weeds using prior knowledge: A reinforcement-learning approach

Rick van Essen[a,*], Eldert van Henten[a], Gert Kootstra[a]

[a]*Agricultural Biosystems Engineering, Department of Plant Sciences, Wageningen University and Research, 6700 AA, Wageningen, The Netherlands*

Abstract

UAVs are becoming popular in agriculture, however, they usually use time-consuming row-by-row flight paths. This paper presents a deep-reinforcement-learning-based approach for path planning to efficiently localize weeds in agricultural fields using UAVs with minimal flight-path length. The method combines prior knowledge about the field containing uncertain, low-resolution weed locations with in-flight weed detections. The search policy was learned using deep Q-learning. We trained the agent in simulation, allowing a thorough evaluation of the weed distribution, typical errors in the perception system, prior knowledge, and different stopping criteria on the planner's performance. When weeds were non-uniformly distributed over the field, the agent found them faster than a row-by-row path, showing its capability to learn and exploit the weed distribution. Detection errors and prior knowledge quality had a minor effect on the performance, indicating that the learned search policy was robust to detection errors and did not need detailed prior knowledge. The agent also learned to terminate the search. To test the transferability of the learned policy to a real-world scenario, the planner was tested on real-world image data without further training, which showed a 66% shorter path compared to a row-by-row path at the cost of a 10% lower percentage of found weeds. Strengths and weaknesses of the planner for practical application are comprehensively discussed, and directions for further development are provided. Overall, it is concluded that the learned search policy can improve the efficiency of finding non-uniformly distributed weeds using a UAV and shows potential for use in agricultural practice.

Keywords: Deep Reinforcement Learning, Path Planning, Drones

*Corresponding author.
Email address: rick.vanessen@wur.nl (Rick van Essen)

1. Introduction

In recent years, Unmanned Aerial Vehicles (UAVs) have become increasingly popular for various applications in agriculture (Rejeb et al., 2022). Examples in agriculture are finding weeds or diseased plants in large arable fields (Albani et al., 2019; Chin et al., 2023), detecting cattle in pastures (Rivas et al., 2018; Liu et al., 2021), and blossom detection in orchards (Zhang et al., 2023). Generally, these applications have a main task: to find objects of interest in an area larger than the field-of-view (FoV) of the UAV, which requires the UAV to fly over the field. A limitation of using UAVs for these applications is their limited battery capacity (Rejeb et al., 2022; Gugan and Haque, 2023). Therefore, it is important to find a path between all the objects of interest that minimizes the UAV's flight time in order to increase the area that can be inspected in a single flight.

When these objects are uniformly distributed over the area, a coverage path planner covering the entire search area is suitable and efficient. However, in applications where the objects are non-uniformly distributed, it may be more efficient to use a search policy that searches for objects instead of covering the whole area. A task such as the detection of weeds is an example of an application with non-uniformly distributed objects, since some weed species occur in distinct patches in a field (Dessaint et al., 1991; Cardina et al., 1997; Colbach et al., 2000). A lot of work is done on weed detection using drone images (Xu et al., 2023; Anul Haq, 2022; Pei et al., 2022); however, they all follow predefined, row-by-row, flight paths.

Over the last years, Reinforcement Learning (RL) has gained more attention in path planning for both mobile robots (Yu et al., 2020; Gao et al., 2020; Niroui et al., 2019) as well as UAVs (Azar et al., 2021; Tu and Juang, 2023). These RL-based methods can learn search strategies through interaction with the environment by maximizing the information gathered in an environment (Lodel et al., 2022). RL offers several advantages for path planning: it does not need a detailed map of obstacles in the environment, can deal with noisy sensor information (Gao et al., 2020), and can learn spatial relations between objects of interest in the environment. Once trained, an RL learned policy makes sequential decisions about the direction of the robot or drone with relatively short calculation times, as compared to traditional methods like Dijkstra's algorithm, A*, and D* (Tu and Juang, 2023). This allows for online path planning, eliminating the need to calculate the complete path in advance. Using RL, it is possible to plan paths with less or uncertain prior knowledge about the environment (Yu et al., 2020; Gugan and Haque, 2023), thereby making path planning more flexible and reactive to a changing environment (Popović et al., 2024).

Several works have studied path planning using RL. Panov et al. (2018) and Yu et al. (2020) showed that a trained RL network is able to generate a policy to move an agent in a grid-based world to a target

location while avoiding obstacles. Tang et al. (2024) and Chronis et al. (2023) showed that an RL-learned path planner yielded shorter paths and required less computational time than traditional methods like A* for an environment with static obstacles. Castro et al. (2023) used RL to avoid obstacles while executing a flight path for inspecting fly traps in orchards. Westheider et al. (2023) showed an RL-based path planner for cooperative multi-UAV monitoring of terrains, and showed the applicability of the method by discovering temperature hotspots on a real-world thermal dataset of a crop field.

A novel neural network architecture for coverage path-planning and object search was introduced by Theile et al. (2020, 2021), combining detailed high-resolution local information with low-resolution global information. They show that this representation is efficient for UAV path planning to gather data from Internet-of-Things devices within a simulated city. Such representation can also be useful for a weed detection application using UAVs, where low-resolution global information about the field may be available on forehand, and local high-resolution information about the UAV's field-of-view is available during flight. However, their approach assumes full prior knowledge about the location of the target devices and the obstacles, which is, in many agricultural applications, uncertain or even absent. For example, the location of weeds in a field may not be exactly known beforehand, however, some prior knowledge can be derived from the locations of the weeds during the previous year (Colbach et al., 2000; Dessaint et al., 1991). Uncertainty in prior knowledge also makes it more difficult to determine when to terminate the search task. When having full prior knowledge, there is enough information to stop the search task when all objects are found. However, when the prior knowledge becomes more sparse and unreliable, the number of objects might be unknown. This is especially true for weed detection, as the exact number of weeds in a field is unknown. This makes the end of the search task undefined. Yang et al. (2018) and Druon et al. (2020) used a learned stopping action, where the agent could decide to stop searching when it was not profitable anymore. Building on the idea of combining high-resolution local information and low-resolution global information in a single RL network architecture from Theile et al. (2021), we developed and evaluated a reinforcement-learning-based UAV path planner that can deal with this uncertain prior knowledge, to localize non-uniformly distributed weeds in a field and to decide when to terminate the search.

We propose an RL-based path planner that works on a higher abstraction level, getting a local weed map extracted from a perception module based on a camera image, and providing actions in terms of flight directions that would be executed by a drone's flight controller. Figure 1 illustrates how the resulting simulation-trained policy can be used in the real world to control the drone. The drone's camera image, containing variations in the weeds' appearance due to, e.g., lightning conditions and natural variation, is

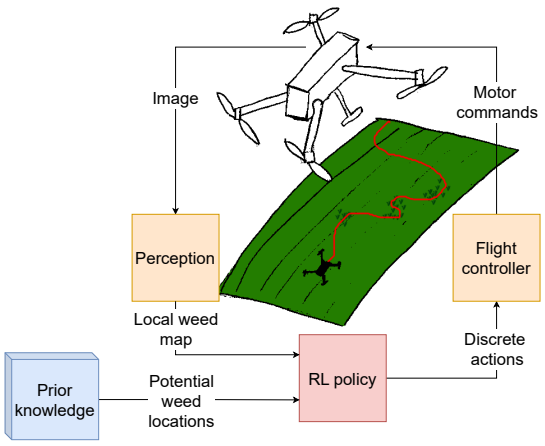


Figure 1: High-level drone control using Reinforcement Learning (RL) in combination with a local weed map from a perception system and some potential weed locations from the prior knowledge. The output of the detection network is used together with prior knowledge as input for the RL policy, the resulting discrete actions are then translated by the drone's flight controller to motor commands.

converted to a local weed map using an object detection network. The resulting weed map, combined with potential, uncertain weed locations derived from prior knowledge, is used as input to the RL policy. The policy determines the action with respect to the highest expected reward. The discrete output actions (fly north, east, west, south) are then translated to motor commands by the drone's flight controller. The advantage of this abstraction is that it allows us to train the RL agent in simulation by simulating the detection network's output and the prior knowledge, and to run the trained agent on real image data.

Simulation-based training is necessary because training an RL agent requires huge amounts of data, as it needs to explore an enormous state-action space (Gugan and Haque, 2023). For our use case, training an RL agent to localize weeds in a field would need the agent to collect a very large amount of training data containing a large variety of weed locations in the field to allow it to explore different policies. As this is infeasible to achieve in the real world, we trained the agent using simulations, as is commonly done in literature, e.g. Azar et al. (2021); Gao et al. (2020); Theile et al. (2020, 2021); Panov et al. (2018); Yu et al. (2020). An additional benefit of a simulation is that we can investigate the effect of different weed distributions, detection inaccuracies, and prior knowledge uncertainty in greater detail. It also allows for many more repetitions than would be feasible in real-world experiments and thereby shows the practical potential of such an RL-based path planner for localizing weeds in a field.

However, when training in simulation, it is important that the abstract simulation includes typical errors in a perception model and a flight controller. The typical errors in translating the image into weed coordinates are false positive and false negative detections, and inaccuracies in the location of the detections.

These are included in the simulation as explained in the materials section. Furthermore, the simulation makes some assumptions about the system: (a) it is assumed the drone is able to accurately execute discrete flight actions, such as fly one meter in a specific direction, (b) a maximum field size based on the number of grid-cells in the simulation and the area of each cell, (c) prior knowledge of the field shape and boundaries, (d) a trained detection network and (e) some potential, uncertain locations of weeds. The impact of all made assumptions is discussed at length in the discussion. To demonstrate the practical potential of learning RL policies in an abstract simulation, we show the performance of the simulation-learned RL policy on real-world image data.

The objective of this paper is to develop and evaluate an RL-based method for path planning to efficiently localize weeds using UAVs with a minimal path length. The novelty of our method is the abstract representation of the environment, which allows for a real-world weed detection application in combination with an object detection network. Despite being mainly a simulation study, we provide an important contribution to a real-world application by analyzing the effects of the spatial distribution of the weeds, typical errors in detection network output, and typical errors associated with prior knowledge. To show the transferability of the simulation-learned search policy to a real-world application, we also evaluate its performance on real-world image data without further training. Specifically, we study (1) the effect of the distribution of the weeds, (2) the effect of detection errors, (3) the effect of the quality of prior knowledge, (4) the effect of different stopping criteria, and finally, (5) the performance of the RL-based path planner on real-world data.

2. Materials and Methods

In this chapter, we describe the simulation environment, problem statement, the used RL method, and the experiments.

2.1. Simulation environment

The field is simulated as a square grid of $M \times M \in \mathbb{N}^2$ grid cells, where $\mathbb{N}$ is the set of natural numbers. The real-world size of each grid cell depends on the required spatial resolution, which is not part of the simulation environment. In this field, the number of weeds, n is drawn from a normal distribution $\mathcal{N}_{\text{obj}}(\mu, \sigma)$. These weeds are then distributed according to k different multivariate Gaussian distributions, where k is drawn from $\mathcal{N}_{\text{dist}}(\mu, \sigma)$. Each Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ has a random mean μ_i and a random covariance $\Sigma_i \in \{\Sigma_1, \Sigma_2\}$. By randomly distributing the weeds in each simulation, we ensure that the RL agent is

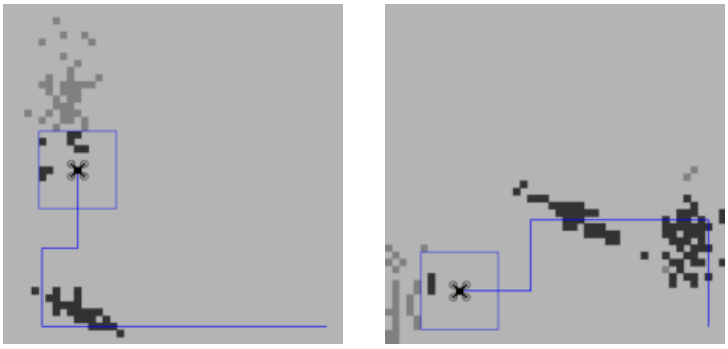


Figure 2: Two examples of the simulation environment with the field-of-view indicated by the blue rectangle around the drone, the flight path in blue, the detected weeds in dark-gray and the not-yet detected weeds in light-gray.

reactive to the information gathered from the environment, rather than learning the specific locations of each weed. Additionally, by introducing randomness in the weed distributions, the RL agent can learn to respond to a large variety of different weed distributions through domain randomization.

A drone is flying over the field at a fixed altitude and has a camera with a FoV of size $F \times F \in \mathbb{N}^2$ facing downward. The goal of the drone is to find all weeds in the field as fast as possible. The drone can start at the top-left or bottom-right part of the field, which is randomly selected. For experiments 1–3, the simulation terminates when all the weeds are found; in experiments 4 and 5, we use a learned stop signal. Figure 2 shows two examples of the simulation environment.

The output of the detection network is simulated by creating a map of the weeds that are visible within the FoV. Typical errors in the output of a detection network are False Positives (FPs), False Negatives (FNs), and positional errors. FP detections are simulated by adding $r_{\text{dt,fp}} \cdot F^2$ FP detections at a random location in the FoV, where $r_{\text{dt,fp}}$ is the fraction of FP detections with respect to the size of the FoV, F. FN detections are simulated by removing $r_{\text{dt,fn}} \cdot n_{\text{fov}}$ detections from the visible weeds in the current FoV of the drone, where $r_{\text{dt,fn}}$ is the fraction of FN detections with respect to the number of weeds visible in the current FoV, n_{fov}. Uncertainty in the position of the weeds in the detection map is simulated by adding an offset drawn from the normal distribution $\mathcal{N}_{\text{dt,pos}}(0.0, \sigma)$, independently in both x and y directions. These errors are simulated independently for each detection.

To simulate prior knowledge, inaccuracies are added to the map of ground-truth locations of weeds. These inaccuracies are added using a similar approach as described above. False positives in the prior knowledge are added by adding $r_{\text{pk,fp}} \cdot M^2$ weeds to the ground-truth map, where $r_{\text{pk,fp}}$ is the fraction of FPs in the prior knowledge with respect to the field size M. FNs in the prior knowledge map are simulated by removing $r_{\text{pk,fn}} \cdot n$ weeds from the ground-truth map with the weed locations where $r_{\text{pk,fn}}$ is the fraction

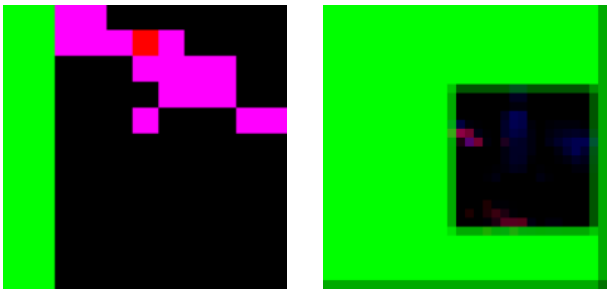
of FNs with respect to the number of weeds n. Same as for the detection network output, the location of the weeds is altered by drawing an offset from the normal distribution $\mathcal{N}_{\text{pk,pos}}(0.0, \sigma)$. To simulate a reduction in resolution, the map is down-sampled using average pooling with a kernel size of $\lfloor \frac{M}{P} \rfloor$ where $P \times P$ is the resulting prior knowledge size. The prior knowledge is simulated once for each simulation.

2.2. Problem definition

To solve the described goal, the simulation is implemented as a Markov Decision Process (MDP). An MDP is described by a state-space S, an action-space A, and a reward R (Sutton and Barto, 2018). In a state $s_t \in S$ at timestep t, an agent (the drone) performs an action $a_t \in A$ yielding a transition to state $s_{t+1} \in S$ and a reward $r_t \in R$.

The state-space representation, S, is adapted from Theile et al. (2021) and uses the same structure, containing a global and local map representation and a movement budget scalar. Maintaining the same structural representation, we encode different information in the global and local map: the global map contains down-sampled information about the complete field (the prior knowledge), and the local map contains detailed information about the current FoV of the drone (representing the output of the detection network). Both global and local maps consist of three layers: a field-area layer, a layer with the locations of the already detected weeds, and a third layer consisting of the prior knowledge for the global map and simulated detection network output for the local map. The field-area layer describes the area of the field (value of 0) and the area outside the field (value of 1). The layer with the already detected weeds contains a value of 1 at the places where weeds have been detected and 0 at all other locations. The movement budget scalar b equals the remaining battery capacity and is calculated by $b = b_{\text{init}} - s \cdot b_{\text{step}}$, where b_{init} is the initial battery level of the drone, s the number of flight actions made in the field, and b_{step} the battery usage of each step.

Both the global and the local map are drone-centric, meaning that the drone is always in the middle of the map. Work by Theile et al. (2021) showed that centering the local and global map makes it possible to scale to larger fields. To do so, the global map is padded to a size of $(2M-1) \times (2M-1)$ by adding padding values of 0 for both the detected weed and prior knowledge layer and values of 1 for the field-area layer (indicating that the drone is not allowed to fly outside the field). To decrease the size of the global map, the global map is down-sampled using average pooling with kernel size g_{global} resulting in a global map size of $\frac{2M-1}{g_{\text{global}}} \times \frac{2M-1}{g_{\text{global}}}$. The local map is padded to a size of $F \times F$ by adding padding values of 0 for both the detected weed map and output of the detection network, and values of 1 for the field-area layer when the FoV partly extends outside the field. An example of the global and local map is shown in Figure 3.



(a) (b)

Figure 3: Example of the local (a) and global (b) map. Red indicates the already detected weeds, green the area outside the field and blue the simulated output of a detection network for the local map and the prior knowledge for the global map. Note that purple is a combination of red and blue.

The action-space, A, contains the following actions: fly north, fly south, fly east, and fly west. Each fly action moves the drone one grid cell in the associated direction. The drone cannot move into the area outside the field. For experiments 4 and 5, we added another 'land' action that terminates the search.

For each timestep t, the reward function, $r(s_t, a_t)$, yields a positive reward, r_{dt}, for every detected weed, a negative reward, r_{nfz}, for trying to fly into the area outside the field and a small negative reward, r_{step}, for every action. A large negative reward, r_{crash}, is given when the drone runs out of battery before the task is completed, as it would crash.

The default parameters for the simulation are given in Table 1. The simulation is implemented using the OpenAI Gym API and is published on GitHub[1].

2.3. Policy learning

The goal of RL is to find a policy $\pi(s)$ that specifies an action $a \in A$ given state $s \in S$. The control policy of the drone can be described by:

$$\pi(s_t) = \underset{a \in A}{\operatorname{argmax}}\, Q(s_t, a), \tag{1}$$

where $Q(s, a)$ is the learned action-value function. Q-learning is a popular model-free method that learns the action-value function by iteratively optimizing a Q-table using the immediate reward r_t and the discounted future reward $\gamma \cdot \max_{a \in A} Q(s_{t+1}, a)$, where $\gamma \in [0, 1]$ is the discount factor determining the emphasis on the future reward (Sutton and Barto, 2018). Q-Learning is, however, unsuitable for learning high-dimensional state-spaces because the size of the Q-table grows exponentially with the number of possible states and actions.

[1] `https://github.com/wur-abe/rl_drone_object_search`

Table 1: Parameters for the default simulation environment.

Parameter	Value	Description
M	48	Field size
$\mathcal{N}_{\text{obj}}(\mu, \sigma)$	$\mathcal{N}(100, 30)$	Number of weeds
$\mathcal{N}_{\text{dist}}(\mu, \sigma)$	$\mathcal{N}(3, 2)$	Number of distributions
Σ_1	$\left[\begin{smallmatrix} 5 & 8 \\ 8 & 15 \end{smallmatrix}\right]$	Covariance distribution 1
Σ_2	$\left[\begin{smallmatrix} 15 & 0 \\ 0 & 5 \end{smallmatrix}\right]$	Covariance distribution 2
F	11	FoV size of the drone
$r_{\text{dt,fp}}$	0.05	Detection FP
$r_{\text{dt,fn}}$	0.0001	Detection FN
$\mathcal{N}_{\text{dt,pos}}(0.0, \sigma)$	$\mathcal{N}(0.0, 0.2)$	Detection offset
$r_{\text{pn,fn}}$	0.20	Prior knowledge FP
$r_{\text{pn,fn}}$	0.001	Prior knowledge FN
$\mathcal{N}_{\text{pk,pos}}(0.0, \sigma)$	$\mathcal{N}(0.0, 0.5)$	Prior knowledge offset
P	12	Prior knowledge resolution
g_{global}	3	Global map kernel size
b_{init}	75	Initial battery level
b_{step}	0.2	Battery usage per step
r_{dt}	1.0	Detection reward
r_{nfz}	-1.0	Hit no-fly-zone reward
r_{step}	-0.5	Step reward
r_{crash}	-150.0	Crash reward

To overcome this problem, we used a Deep Q-Network (DQN) introduced by Mnih et al. (2015). DQN uses a neural network to approximate the action-value function. Although more RL algorithms exist, we use DQN because of the discrete action space and its sample efficiency (relative to other RL algorithms). However, the RL algorithm can easily be changed since the simulation environment uses the standard OpenAI Gym API.

Figure 4 shows the training procedure for the DQN. During training, two identical neural networks are used: a policy network Q and a target network Q_{target}. Both networks have an equal architecture, which is described in section 2.3.1. During training, the parameters from the target network, θ_{target}, are updated by the policy network parameters, θ, using a soft update:

$$\theta_{\text{target}} \leftarrow (1 - \tau)\theta_{\text{target}} + \tau\theta, \tag{2}$$

where $\tau \in [0, 1]$ defines the update rate. This is done to stabilize the learning by reducing the correlation between the action values $Q(s, a)$ and the target value (immediate reward + discounted future rewards) (Mnih et al., 2015). Training data for the DQN is created using a replay buffer, which is explained in section 2.3.2. The optimization of the policy network is described in section 2.3.3.

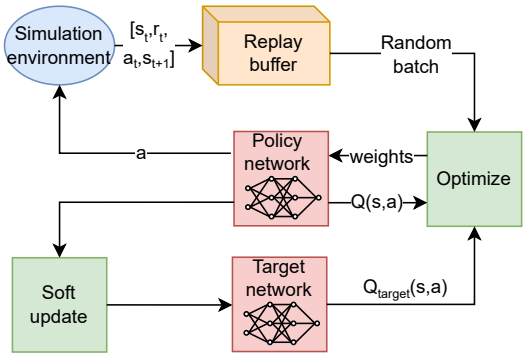


Figure 4: Training procedure for the Deep Q-Network.

2.3.1. Network architecture

Figure 5 shows the network architecture used for the Q-network. This architecture is based on the architecture used in Theile et al. (2021). The network gets the current state s_t as input and predicts the Q-value for every action $a \in A$. The network consists of a feature extractor and a fully connected network. The feature extractor consists of two parallel convolution blocks to ensure unique feature extraction from both the local and the global map. The output of both convolution blocks is flattened and concatenated with the movement budget to combine the extracted features of the local and global map with the movement budget. Four fully-connected layers convert these features into four outputs that correspond to the action values for each possible action $a \in A$. Compared to Theile et al. (2021), the input global map was larger (after down-sampling, see Section 2.2), resulting in a larger flattened layer and an increased number of trainable parameters.

2.3.2. Experience Replay Buffer

To create training data for the DQN, we used an experience replay buffer (Mnih et al., 2015). The replay buffer is a circular buffer of size n_{buffer} containing state transition vectors, $[s_t, a_t, r_t, s_{t+1}]$. From this buffer, mini-batches of size n_{batch} are sampled for training the Q-network. The buffer is filled by an agent that operates in the simulation following a probability-based training policy that takes random actions from the action space A using the probability vector p:

$$\forall a \in A, p(s,a) = \frac{\exp(Q(s,a)/\lambda)}{\sum_{\forall a_i \in A} \exp(Q(s,a_i)/\lambda)}, \tag{3}$$

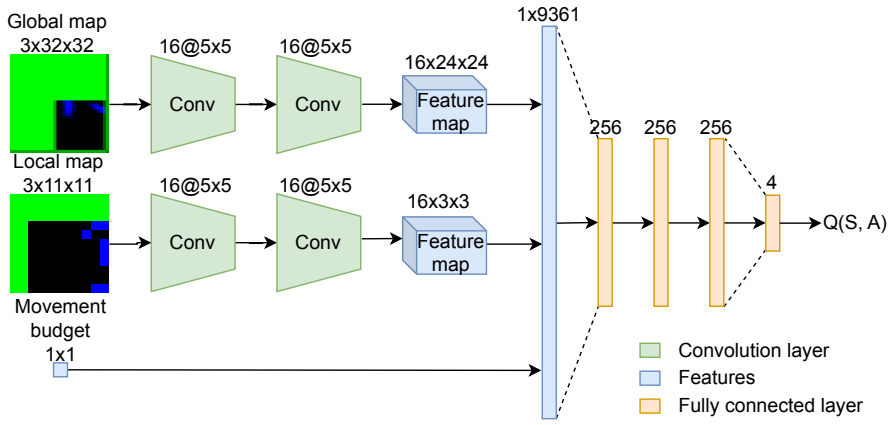


Figure 5: Network architecture for the DQN using the global and local map and the movement budget indicating the input size, the number of kernels and their size, the size of the flatten layer, and the size of the fully connected layers. The number of trainable parameters equals 2,544,548.

where $\lambda \in (0, \infty)$ is the temperature parameter. A high value for λ gives all actions equal probability, and a low value results in a greedy training policy (Equation 1). This probability-based training policy has an advantage over a standard ϵ-greedy policy because λ is independent of the number of training steps, whereas ϵ usually requires a schedule to decrease with the number of training steps. The training policy is equal to Theile et al. (2020).

2.3.3. Optimizing policy network

The optimal action-value function is approximated by minimizing the smooth L1 loss (Girshick, 2015) between the target, $y = r_t + \gamma \max_{a_i \in A}(Q_{\text{target}}(s_{t+1}, a_i))$, and the predicted value, $\hat{y} = Q(s_t, a_t)$, using $\beta = 1$. Optimizing the weights of Q is done using an Adam optimizer with learning rate α.

Optimization of the policy network was started after the replay buffer was filled for 50% and was optimized for n_{steps} training steps. The best weights were selected based on the highest mean reward observed over a validation set of n_{val} evaluation episodes (sequences of steps in a simulation).

Table 2 shows the parameters used during training. The implementation of DQN in Stable-Baselines3 (Raffin et al., 2021) was used with a custom feature extractor and custom training policy. Training was done on a computer with an AMD Ryzen 5950x CPU and NVIDIA RTX 3090 graphics card using 12 parallel simulation environments.

Table 2: Hyperparameters for training the Deep Q-Network.

Parameter	Value	Description
γ	0.95	Discount factor
τ	0.005	Update rate
n_{buffer}	50000	Experience replay buffer size
n_{batch}	128	Mini-batch size
λ	0.1	Temperature parameter
α	$3 \cdot 10^{-5}$	Learning rate
n_{steps}	10^7	Training timesteps
n_{val}	120	Number of validation episodes

2.4. Experiments

In this section, we present the experiments to validate the learned search policy. We evaluate the percentage of weeds found and the flight-path length. These values are compared to a traditional row-by-row coverage flight path, planned by Fields2Cover (Mier et al., 2023), using the size of the FoV, F, as path width without overlap. Because we expected the learned path planner to outperform a row-by-row flight path specifically for non-uniformly distributed weeds, we evaluated the impact of these weed distributions (section 2.4.1). To test the robustness of the RL-learned path planner, we evaluated the effect of detection errors (section 2.4.2) and prior knowledge quality (section 2.4.3). We also investigated the effect of different stopping criteria to mark the end of the search (section 2.4.4). Lastly, the transferability of the simulation-trained RL policy was evaluated on real-world image data (section 2.4.5).

2.4.1. Experiment 1: Impact of weed distributions

To study the influence of the weed distribution on the path length and percentage of found weeds, three weed distributions were defined. Distribution 'strong' was a strong clustered distribution and used the default field parameters from Table 1. Distribution 'medium' was a medium clustered distribution with $\mathcal{N}_{\text{dist}}(\mu, \sigma) = \mathcal{N}(4, 1)$ and $\mathcal{N}(\mu_i, \Sigma_i)$ has a random mean μ_i and a covariance Σ_i uniformly sampled from set $\{\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\}$ where $\Sigma_1 = \left[\begin{smallmatrix} 10 & 16 \\ 16 & 40 \end{smallmatrix}\right]$, $\Sigma_2 = \left[\begin{smallmatrix} 40 & 0 \\ 0 & 10 \end{smallmatrix}\right]$, $\Sigma_3 = \left[\begin{smallmatrix} 30 & 12 \\ 12 & 12 \end{smallmatrix}\right]$, and $\Sigma_4 = \left[\begin{smallmatrix} 15 & 4 \\ 4 & 20 \end{smallmatrix}\right]$. Distribution 'uniform' was a uniform distribution with each weed having a uniform random non-overlapping coordinate in the field. For all distributions, the number of weeds in the field was drawn from the same normal distribution $\mathcal{N}_{\text{obj}}(\mu, \sigma)$. An example of each distribution is shown in Figure 6. All other parameters were kept at their default value as defined in Table 1 and 2. A different policy was trained and evaluated using 1000 evaluation episodes for the three distributions.

(a) (b) (c)

Figure 6: Example of a field with (a) a strong distribution, (b) a medium distribution, and (c) a uniform distribution of weeds.

Table 3: Error levels for the simulated detection network.

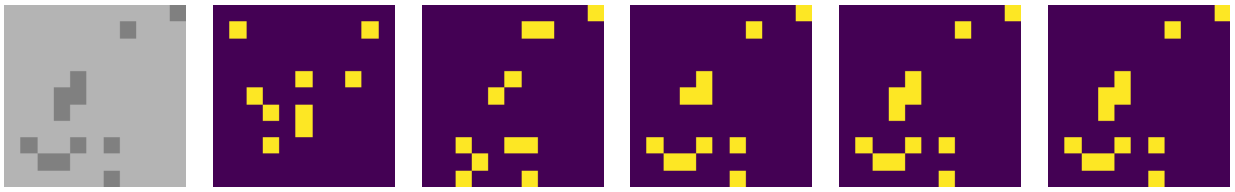
Error level	$r_{\text{dt,fp}}$	$r_{\text{dt,fn}}$	$\mathcal{N}_{\text{dt,pos}}(0, \sigma)$
very high	0.01	0.5	$\mathcal{N}(0, 0.5)$
high	0.001	0.1	$\mathcal{N}(0, 0.1)$
moderate	0.0001	0.05	$\mathcal{N}(0, 0.05)$
low	0.00005	0.02	$\mathcal{N}(0, 0.02)$
perfect	0.0	0.0	$\mathcal{N}(0, 0)$

2.4.2. Experiment 2: Influence of detection errors

To assess the influence of errors of the simulated detection network on the policy, we defined five detection error levels, ranging from a very high number of errors to no errors. The parameters for each level of detection errors are given in Table 3. The default simulation environment from Table 1 is equivalent to the moderate level in Table 3. Figure 7 shows an example of the simulated output of the detection network. All other parameters were kept at their default value as defined in Table 1 and 2. For each level of detection errors, a policy was trained and evaluated using 1000 evaluation episodes.

2.4.3. Experiment 3: Influence of prior knowledge quality

We assessed two aspects of quality in the prior knowledge: the uncertainty related to resolution and the inaccuracy in the prior knowledge. When the resolution of the prior knowledge is low, only sparse information about the location of the weeds is available. This could, for example, be information from other sources with a lower spatial resolution, such as satellite images or images taken from a higher altitude. Another aspect of prior knowledge is the inaccuracy. The higher the inaccuracy, the less reliable the prior knowledge becomes. This could, for example, be due to sensor noise or detection errors. We defined five levels of prior knowledge quality, ranging from no prior knowledge (level none) to perfect prior knowledge (level perfect), expressing an increasing quality. Table 4 defines these levels. The default simulation environment from Table 1 is equivalent to the moderate level in Table 4. To keep the same number of trainable parameters in the DQN, we kept the same number of cells in the prior knowledge map for all quality levels by using nearest neighbor upsampling to resize the prior knowledge map to 48×48. Figure 8 shows an example of



(a) (b) (c) (d) (e) (f)

Figure 7: Example of field-of-view (a) and the corresponding simulated detection network output for a very high (b), high (c), moderate (d), and low (e) number of detection errors, and a perfect (f) detector. For visibility, a colormap is applied.

Table 4: Prior knowledge quality levels.

Quality level	P	$r_{\text{pk,fp}}$	$r_{\text{pk,fn}}$	$\mathcal{N}_{\text{pk,pos}}(0, \sigma)$
none	0x0	-	-	-
low	2x2	0.002	0.40	$\mathcal{N}(0, 1.0)$
moderate	12x12	0.001	0.20	$\mathcal{N}(0, 0.5)$
high	24x24	0.0005	0.05	$\mathcal{N}(0, 0.25)$
perfect	48x48	0.0	0.0	$\mathcal{N}(0, 0)$

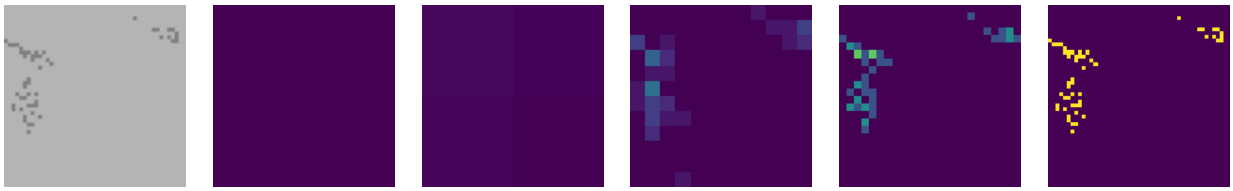
the prior knowledge map for each level. All other parameters were kept at their default value as defined in Table 1 and 2. For each quality level, a different policy was trained and evaluated using 1000 evaluation episodes.

2.4.4. Experiment 4: Effect of different stopping criteria

In a real-world application, the number of weeds might be unknown, which makes the termination of the simulation when all weeds are found infeasible. Therefore, we evaluated the effect of different stopping criteria, specifically:

1. Stop searching when a certain percentage of the field is covered. Two levels were tested, 50% and 75%.
2. Stop searching when there are no new weeds detected for a certain number of flight actions. To avoid constantly resetting this counter due to FP detections, at least 2 weeds need to be detected. Three thresholds were tested: 15, 25, and 50 steps.
3. A learned landing action, by extending action space A with a 'land' action that terminates the search.

These stopping criteria were compared with the default stopping criterion, which stops searching when all weeds are found. We compared both the percentage of found weeds and the path length. For each stopping criterion, a separate policy was trained and evaluated on 1000 evaluation episodes.



(a) (b) (c) (d) (e) (f)

Figure 8: Example of the prior knowledge of world (a) for quality level none (no prior knowledge) (b), low (c), moderate (d), high (e), and perfect (f). Level perfect has high-resolution knowledge about the locations of the weeds, while level low has limited knowledge about the locations of weeds. For visibility, a colormap is applied.

2.4.5. Experiment 5: Transferability of simulation-learned policy to a real-world application

To test the transferability of the learned policy to a real-world scenario, the planner was tested on real-world image data without further training. To do so, we evaluated its performance on the four real-world datasets presented in van Essen et al. (2025). Each dataset consists of a high-resolution orthomosaic of a grass field containing artificial plants distributed in clusters in the field, as a proxy for a weed detection application. Images were captured at four different dates in spring and summer using a DJI M300 drone with a Zenmuse P1 RGB camera, each date having a different weed distribution. The orthomosaics were made using Agisoft Metashape (Agisoft, 2023) using the high-quality settings. Compared to van Essen et al. (2025), we only use a part of the orthomosiacs to match the squared input format of the RL agent.

Figure 9 illustrates the generation of the local weed map using real-world orthomosaic data. To align with the state representation of the RL agent, we rasterize the fields in a raster of $M \times M$ grid cells with a grid-cell size of 1x1 m. At each time step, a camera image was generated by mapping the UAV position to real-world coordinates, cropping an area of $F \times F$ m around this position from the orthomosaic, and resizing the crop to an image of 2048×2048 pixels using linear interpolation. Weeds were detected in the camera images using YOLOv8-nano (Jocher et al., 2023). All detected weeds with a confidence score larger than 0.5 were mapped to the local map coordinates to create the local weed map. The detection network was trained for 250 epochs on the training dataset used in (van Essen et al., 2025), which consisted of real-world drone images from the grass field with artificial plants taken from 12m, 24m, and 32m altitude. In total, 1618 images with 2000 annotations were available. Compared to (van Essen et al., 2025), both classes were merged into a single 'plant' class.

To create the global map, prior knowledge was generated using a high-altitude row-by-row flight path with a field-of-view of 24x24 grid cells. In these images, weeds were detected using the trained detection network. All detected weeds with a confidence score larger than 0.05 were mapped to the global map coordinates.

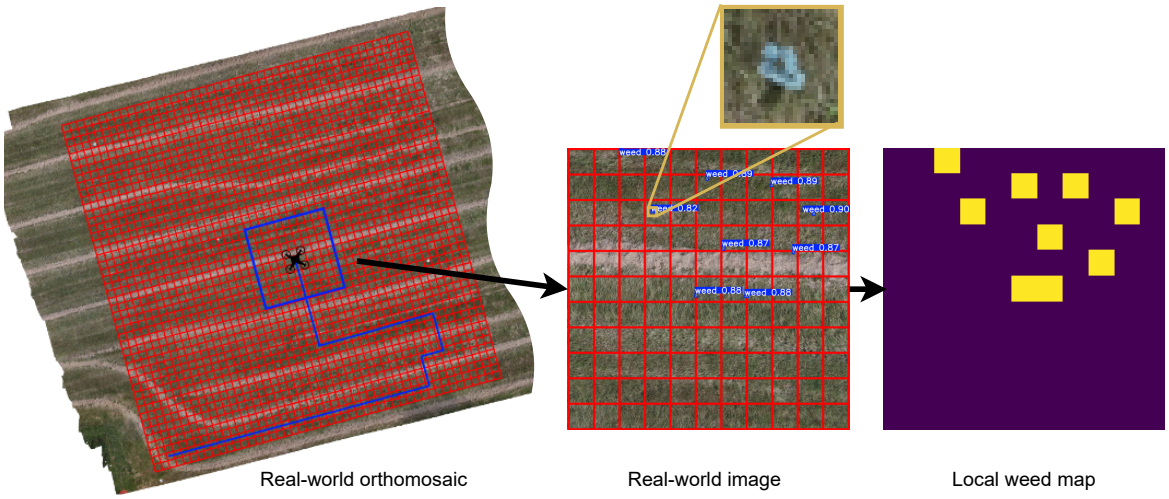


Figure 9: Applying the simulation-trained Reinforcement Learning policy to real-world image data by generating an image from the orthomosaic (with the raster shown in red) corresponding to the drone's field of view (blue rectangle), and converting it into a local weed map using an object detection network.

For the experiment, we used the trained RL policy with learned landing action from experiment 4. The results of this policy on the four datasets were compared to the baseline row-by-row flight path. We compared both the percentage of found weeds and the flight-path length.

3. Results

3.1. Experiment 1: Impact of weed distributions

Figure 10 shows the relation between flight-path length and the percentage of found weeds for different distributions of weeds. The more uniformly distributed the weeds were, the more linear the relation between path length and the number of found weeds. For the strong and medium distributions, the learned policy outperformed the baseline row-by-row flight path, having found more than 80% of the weeds in 73 and 94 steps on average, respectively, compared to 209 steps for the baseline row-by-row flight path. However, the learned policy often had trouble finding all weeds before the battery was empty.

Table 5 shows the percentage of found weeds at 100, 200, and 300 flight steps for the policy learned by DQN and the baseline row-by-row flight path. Up to 200 flight steps, the DQN policy significantly found more weeds than the baseline. However, in contrast to the baseline, it did not find all weeds.

Figure 11 shows examples of a single flight path for the strong, medium, and uniform distributions, respectively. It can be seen that the drone flew in quite straight lines till around 80% of the weeds were

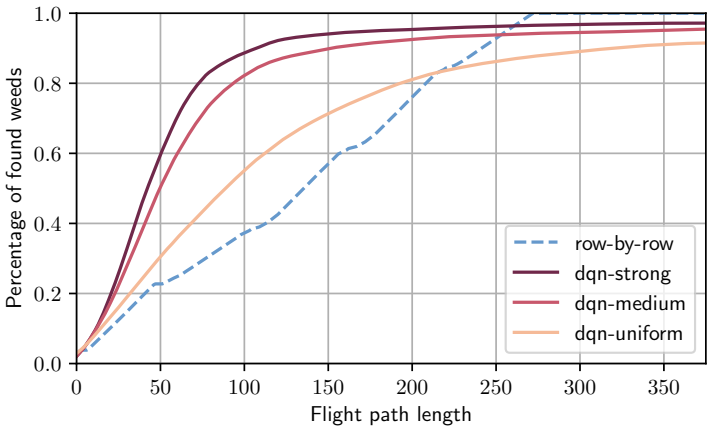


Figure 10: Effect of a strong, a medium, and a uniform weed distribution on both the number of found weeds and the path length for the policy learned by DQN, and the baseline row-by-row flight path. The lines show the mean over 1000 episodes.

Table 5: Percentage of found weeds at 100, 200, and 300 flight steps for a strong, a medium, and a uniform weed distribution for the policy learned by DQN, and the baseline row-by-row flight path. The values show the mean ± the standard deviation. Values indicated with a '*' have a significant ($\alpha = 0.001$) higher percentage of found weeds than the baseline row-by-row flight path (Welch's t-test).

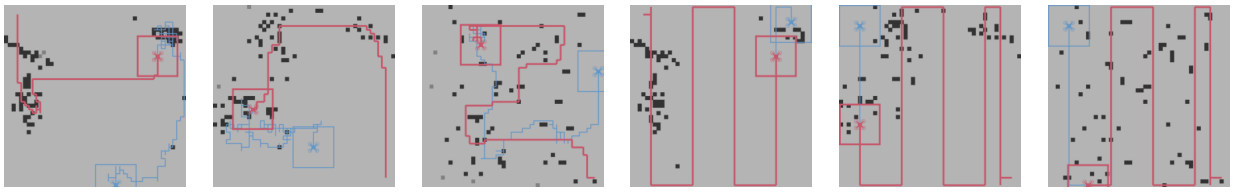
Method	100 steps	200 steps	300 steps
row-by-row	0.37±0.08	0.76±0.07	1.00±0.00
dqn-strong	0.88±0.14*	0.95±0.08*	0.97±0.06
dqn-medium	0.82±0.13*	0.93±0.08*	0.95±0.05
dqn-uniform	0.55±0.06*	0.81±0.07*	0.89±0.05

found. After 80% the drone started wandering around till the battery was empty and crashed, because the drone missed some weeds, and the environment only terminated when all weeds were found.

3.2. Experiment 2: Influence of detection errors

Figure 12 shows the relation between the different levels of detection errors and the percentage of found weeds and flight-path length. All the levels outperformed the baseline row-by-row flight path. The learned policy is robust and only starts to drop performance for a very high number of detection errors. All levels outperform the baseline by having found 80% of the weeds in 70–98 steps compared to 206 steps for the baseline row-by-row flight path.

Table 6 shows the percentage of found weeds at the different detection error levels corresponding to 100, 200, and 300 flight steps for the DQN policy and the baseline row-by-row flight path. The differences between the levels were small, only the very-high level of detection errors performed slightly lower, although it still found significantly more weeds than the baseline until 200 flight steps. Up to 200 flight steps, the DQN policy had found significantly more weeds than the baseline.



(a) (b) (c) (d) (e) (f)

Figure 11: Single flight paths of the RL agent (a-c) and the baseline row-by-row flight path (d-f) for strong (a,d), medium (b,e), and uniform (c,f) distribution of weeds. The detected weeds are indicated with black dots and the undetected weeds with gray dots. The red line indicates the flight path till 80% of the weeds are found, the blue line the complete flight path till all weeds were found or the battery was empty.

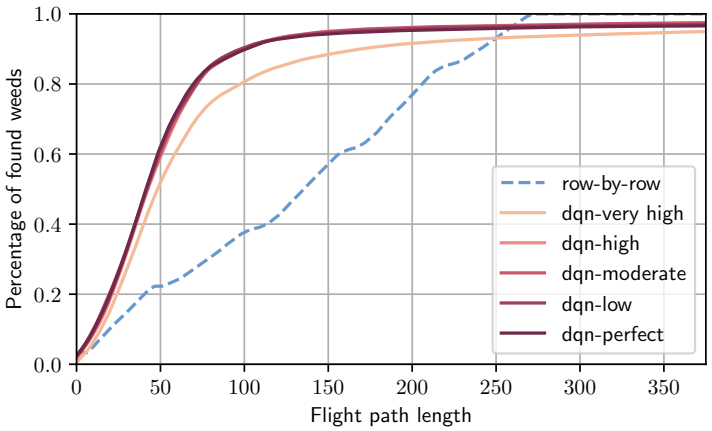


Figure 12: Effect of the different levels of detection errors on the percentage of found weeds for the policy learned by DQN, and the baseline row-by-row flight path. The lines show the mean over 1000 episodes.

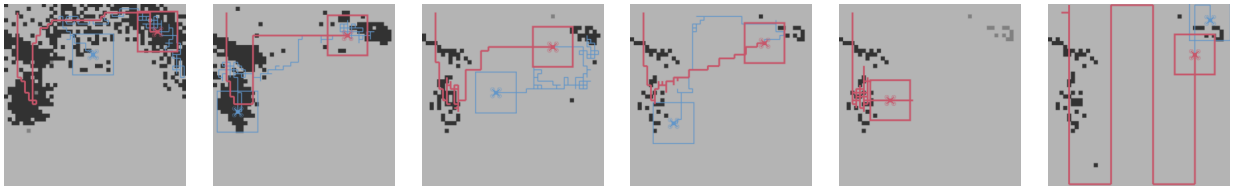
Figure 13 shows some flight paths for the different levels of detection errors. For a very high number of detection errors (Figure 13a), there are many false positive detections visible. When having a perfect detection network (Figure 13e), the agent missed some weeds, however, this is just an example of a single flight path. On average, this level of detection errors performed comparable to the levels low-high (Figure 12).

3.3. Experiment 3: Influence of prior knowledge quality

Figure 14 shows the relation between flight-path length and the number of found weeds for different levels of prior knowledge quality. When having a perfect prior knowledge map without mistakes, the DQN learned a policy that found all weeds. Even with prior knowledge of low quality, the DQN learned a policy to quickly find most weeds. However, not all weeds were found before the battery was empty. The higher the quality of the prior knowledge, the more weeds were found. In complete absence of prior knowledge (level none) the agent still found some weeds, however, a row-by-row flight path is more efficient in that

Table 6: Percentage of found weeds at 100, 200, and 300 flight steps for different levels of detection errors for the policy learned by DQN, and the baseline row-by-row flight path. The values show the mean ± the standard deviation. Values indicated with a '*' have a significant ($\alpha = 0.001$) higher percentage of found weeds than the baseline row-by-row flight path (Welch's t-test).

Method	100 steps	200 steps	300 steps
row-by-row	0.37±0.30	0.76±0.25	1.00±0.00
dqn-very high	0.80±0.17*	0.92±0.08*	0.94±0.06
dqn-high	0.90±0.11*	0.96±0.05*	0.97±0.04
dqn-moderate	0.90±0.12*	0.96±0.05*	0.97±0.04
dqn-low	0.90±0.13*	0.96±0.05*	0.97±0.04
dqn-perfect	0.90±0.13*	0.95±0.07*	0.96±0.06



(a) (b) (c) (d) (e) (f)

Figure 13: Single flight paths of the RL agent for detection error levels very low (a), low (b), moderate (c), high (d), and perfect (e) and the baseline row-by-row flight path (f). The detected weeds are indicated with black dots and the undetected weeds with gray dots. The red line indicates the flight path till 80% of the weeds are found, the blue line the complete flight path till all weeds were found or the battery was empty.

case. The levels moderate and above outperformed the baseline by finding more than 80% of the weeds in 71, 69, and 67 steps, respectively, compared to 206 steps for the row-by-row flight path. Level low needed around 300 steps to find 80% of the weed.

Table 7 shows the percentage of found weeds for the different levels of prior knowledge corresponding to 100, 200, and 300 flight steps for the DQN policy and the baseline row-by-row flight path. Levels 'moderate' and higher had a significantly higher number of found weeds than the baseline at 200 flight steps. Level 'low' only outperformed the baseline at 100 flight steps. Without prior knowledge, the row-by-row flight path was more efficient even at a low number of flight steps. At 300 flight steps, the baseline found all weeds, compared to 97%, 98% and 99% for the levels 'moderate', 'high', and 'very-high' respectively.

Figure 15 shows some flight paths for the DQN agents with different levels of prior knowledge quality. Without prior knowledge (Figure 15a), the agent quickly found the weeds around the start location of the drone based on information from the local map, but failed to find weeds further away from the start location. When having perfect prior knowledge (Figure 15e), the agent quickly finds all weeds without wandering around.

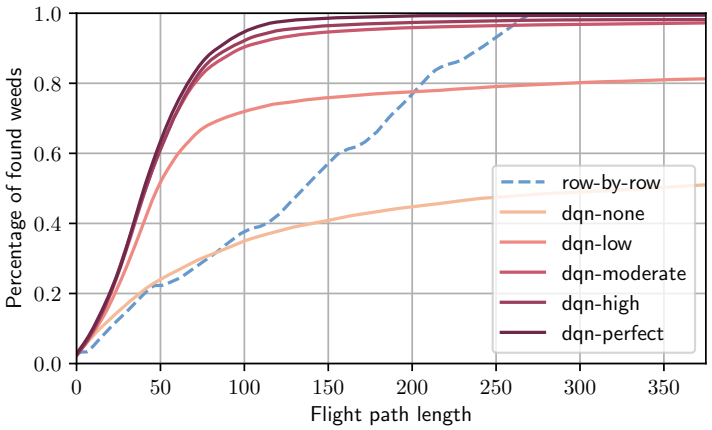


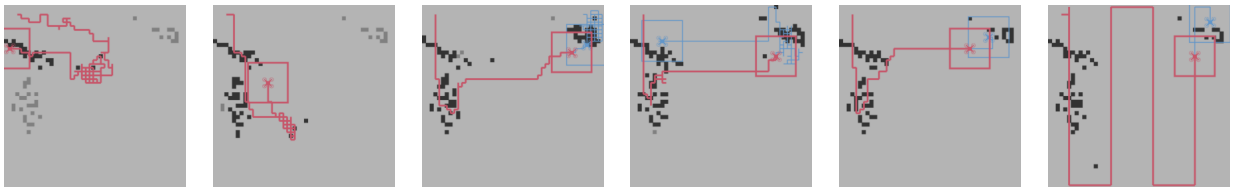
Figure 14: Effect of the different prior knowledge quality levels on the percentage of found weeds for the policy learned by DQN, and the baseline row-by-row flight path. The lines show the mean over 1000 episodes.

Table 7: Percentage of found weeds at 100, 200, and 300 flight steps for different prior knowledge quality levels for the policy learned by DQN, and the baseline row-by-row flight path. The values show the mean ± the standard deviation. Values indicated with a '*' have a significant ($\alpha = 0.001$) higher percentage of found weeds than the baseline row-by-row flight path (Welch's t-test).

Method	100 steps	200 steps	300 steps
row-by-row	0.37±0.30	0.76±0.25	1.00±0.00
dqn-none	0.35±0.32	0.45±0.33	0.49±0.33
dqn-low	0.72±0.24*	0.78±0.22	0.80±0.20
dqn-moderate	0.90±0.12*	0.96±0.05*	0.97±0.05
dqn-high	0.92±0.10*	0.97±0.03*	0.98±0.03
dqn-perfect	0.94±0.08*	0.99±0.02*	0.99±0.01

3.4. Experiment 4: Effect of different stopping criteria

Table 8 shows the percentage of weeds found and flight-path length using different stopping criteria. Setting a threshold on coverage resulted in a high percentage of found weeds, but also a long path length. This indicates that the agent had difficulty fulfilling the coverage threshold before the battery was empty. Stopping the search task when there were no new detections during 15, 25, or 50 consecutive steps resulted in a significantly shorter path length than the baseline row-by-row flight path. However, terminating the task 15 or 25 steps after the last detection was too soon, indicated by the lower percentage of found weeds. Terminating the search task when there were no new weeds detected in the previous 50 steps resulted in a high number of found weeds and a shorter flight path than the default stopping criterion (stop searching when all weeds were found). Using a learned land action that terminates the search yielded a high percentage of found weeds and a very short path length. Compared to the baseline row-by-row flight path, it yielded a 74% shorter flight path at the cost of a 12% lower percentage of found weeds. When it is not essential to find all weeds and a short flight-path length is important, using a learned land action to terminate the



(a) (b) (c) (d) (e) (f)

Figure 15: Single flight paths of the RL agent for prior knowledge quality level none (no prior knowledge) (a), low (b), moderate (c), high (d), and perfect (e), and the baseline row-by-row flight path (f). The detected weeds are indicated with black dots and the undetected weeds with gray dots. The red line indicates the flight path till 80% of the weeds are found, the blue line the complete flight path till all weeds were found or the battery was empty.

Table 8: Percentage of found weeds and path length for the different stopping criteria and the baseline row-by-row flight path. The values show the mean ± the standard deviation. Values indicated with a '*' have a significantly shorter ($\alpha = 0.001$) path length than the baseline row-by-row flight path (Welch's t-test).

Method	Percentage found weeds	Path length
row-by-row	1.00±0.00	276±0
all weeds (default)	0.98±0.03	281±130
coverage 50%	0.94±0.08	239±97*
coverage 75%	0.97±0.06	371±14
15 steps no new weeds	0.45±0.42	52±40*
25 steps no new weeds	0.72±0.38	98±52*
50 steps no new weeds	0.94±0.14	193±79*
land action	0.88±0.16	71±31*

search is suitable.

Figure 16 shows the distribution of the action value of the learned land action (after the softmax layer) compared to the percentage of found weeds. As was expected, the number of times the land action got a high action value increased when the percentage of found weeds increased, indicating that the agent learned that landing is only profitable after finding most weeds. In 2% of the episodes, the drone landed directly without taking any flight actions.

3.5. Experiment 5: Transferability of simulation-learned policy to a real-world application

Figure 17 shows the percentage of found weeds against the flight-path length on the four real-world datasets for the DQN policy and the baseline row-by-row flight path. Till 200 flight steps, the DQN policy found more weeds than the baseline. When the drone landed, the percentage of found weeds on the real-world image data ranged between 68% to 97% for the DQN policy. Due to detection errors, the baseline did not find all weeds, with the total percentage of found weeds ranging between 82% to 97%. On average, the DQN policy found $81.2 \pm 10.4\%$ of the weeds before landing in 94 ± 31 flight steps, whereas the row-by-row flight path found $90.9 \pm 5.8\%$ of the weeds in 276 ± 0 flight steps. This corresponds to a 66% shorter flight path compared to the row-by-row flight path at the cost of a 10% lower percentage of found weeds.

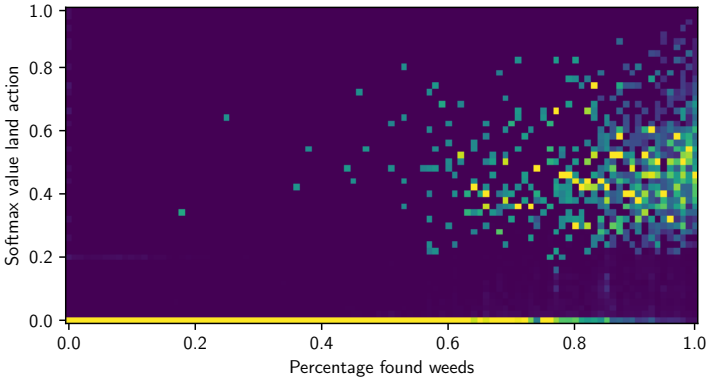


Figure 16: Histogram showing the relation between the action value after the softmax layer for the landing action and the percentage of found weeds. The colors are normalized column-wise.

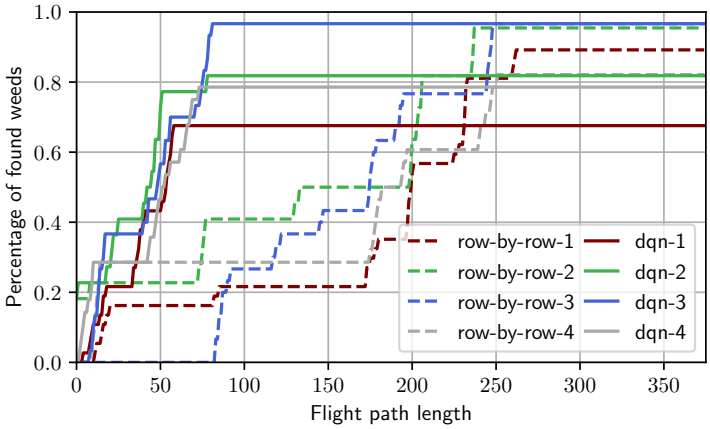


Figure 17: Percentage of found weeds and the flight-path length for the policy learned by DQN and the baseline row-by-row flight path on the four real-world datasets.

Figure 18 shows the flight path of the DQN policy and the baseline row-by-row flight path for real-world datasets. On dataset 1, the DQN policy missed a cluster of weeds, which explains the lower percentage of found weeds for dataset 1 in Figure 17. On datasets 2 and 4, the DQN policy found all clusters, but missed some weeds within the clusters. Both the DQN policy and the baseline made some false-positive and false-negative detections.

4. Discussion

The results showed that, with prior knowledge, the RL agent was able to find a shorter path than a baseline row-by-row flight path for finding the weeds when they were non-uniformly distributed. Even with low-quality prior knowledge and a high number of detection errors, the learned search policy still

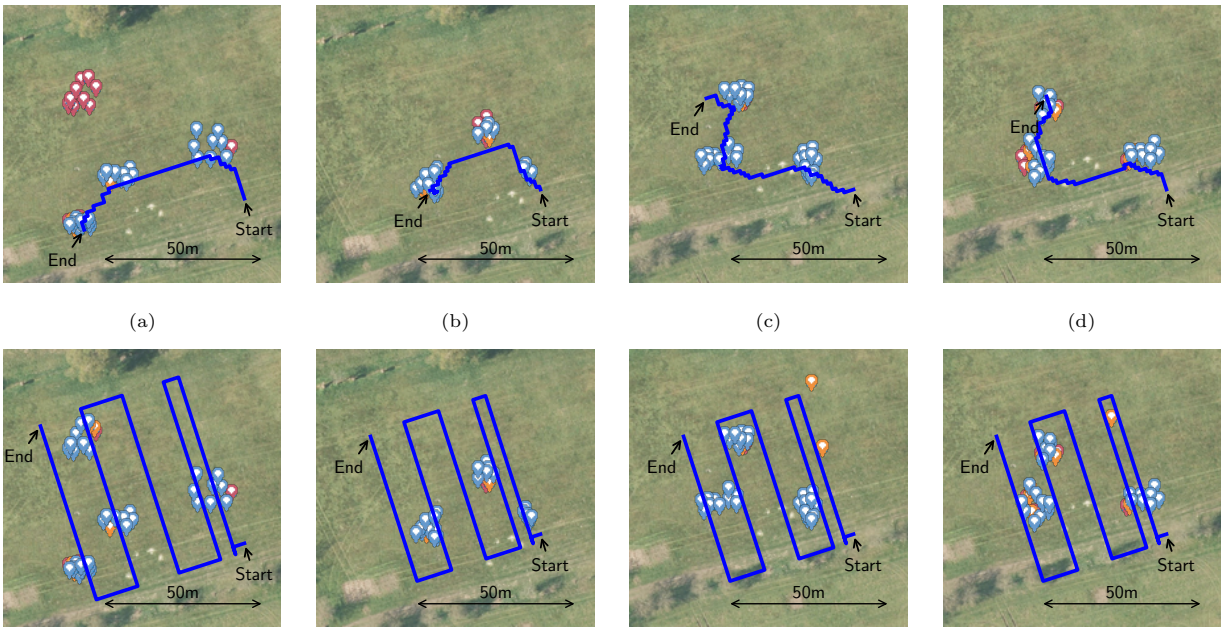


Figure 18: Flight path on dataset 1 (a,e), 2 (b,f), 3 (c,g) and 4 (d,h) for the DQN policy (a–d) and the baseline row-by-row flight path (e–h). Blue, orange, and red markers indicate a true positive, false positive, and false negative object detections, respectively. The start and end position of the UAV is indicated by an arrow.

outperformed a baseline row-by-row flight path in terms of path length and was able to find most weeds. Especially when the weeds are not uniformly distributed, there is great potential in a learned search policy over a traditional row-by-row flight path if a perfect mapping of weeds is not required. When there was no perfect prior knowledge available, the RL agent still found most weeds quicker than the row-by-row flight path, but was not able to find all weeds. The simulation-learned policy is transferable to a real-world application, as was shown in experiment 5. Section 4.1 discusses the DQN policy results compared to the literature, section 4.2 the action selection by the agent, section 4.3 the used action space, section 4.4 learning in absence of prior knowledge, section 4.5 the application and finally, section 4.6 discusses the influence of the assumptions on the applicability in the real world.

4.1. Comparison with literature

All results in this paper were compared to a baseline row-by-row flight path, which is most commonly used in practice. On real-world data, the DQN policy yielded a 66% shorter flight path at the cost of a 10% lower percentage of found weeds. Several alternative UAV path planning methods for object detection in fields are proposed in literature. For instance, van Essen et al. (2025) showed a rule-based adaptive path planner that resulted in a 37% shorter flight path at the cost of a 2% lower F1-score compared to a

row-by-row flight path. Popovic et al. (2017) introduced a path planner for active weed classification that found 85% of the weeds in 150 seconds, whereas a row-by-row flight path took 300 seconds to find the same number of weeds. Both papers use different data, which makes a direct comparison impossible, but they do show the same trade-off between accuracy and flight path length. Similarly to the results in this paper, they both show that the flight paths were more efficient when the objects were non-uniformly distributed.

4.2. Action selection

Towards the end of some episodes, the agent sometimes got stuck in the same spot when not all weeds were found by repeating the same two actions, such as 'fly north' and 'fly south' or 'fly east' and 'fly west' (Figure 15b for an example). Using a learned land action in experiment 4 solved this issue as it allowed the RL agent to decide when to terminate the search task, and thus, when it was not profitable anymore to search for the last weeds. However, we observed some cases where the drone directly terminated the search task. This only happened in 2% of the episodes, in contrast to Yang et al. (2018) and Druon et al. (2020). Yang et al. (2018) indicated a 20-30% lower performance when using a stopping action, and Druon et al. (2020) attributed around 50 percent of the failure cases to selecting the stopping action at the wrong moment. The difference in performance may be due to their use of an end-to-end network architecture that takes images as input, unlike the abstract representation used in this work. As these images have a larger variability than the abstract representation we use, the network is likely to be more uncertain about its predictions, which makes it more profitable to terminate earlier. To reduce the remaining 2% of direct terminations, a penalty for missing weeds when landing could be introduced during training. This would encourage the agent to explore at least parts of the field before landing.

4.3. Action space

Similar to most research on RL-based path planning for UAVs (e.g. Gugan and Haque (2023); Husnain et al. (2023)), in this work, the environment is simplified into a 2D representation, where the UAV flies at a fixed altitude. To further increase the efficiency of the RL learned policy, the environment could be changed to 3D by involving actions for altitude changes, which can have several advantages for a search policy. In the context of weed detection, it can be used to get a (low resolution) overview of the whole field at a high altitude, which can be used as prior knowledge in the global map. The drone can then learn to fly lower to inspect parts of the field in high resolution.

4.4. Learning in the absence of prior knowledge

Experiment 3 showed that learning a search policy without any prior knowledge remained a challenge. Because the agent had no clue where the weeds were, gathering useful state transition vectors for the experience replay buffer was difficult, which prevented the agent from learning useful actions. Possibly, curriculum learning can be used to gradually build up the challenge for the agent (Narvekar et al., 2020), starting with a simulation that has full prior knowledge and slowly building up to a simulation without prior knowledge.

4.5. Application

The use case in this paper is weed detection in agricultural fields. However, the learned planner could be applied to any task where the target objects are non-uniformly distributed in the environment and the goal of the application is to quickly find most of them. For these applications, the learned policy clearly outperforms the baseline row-by-row flight path. However, there is no guarantee that all objects are found, which limits the usability of such a learned policy in applications where it is crucial to find all objects. For the use case of weed detection in arable fields, it may be fine to detect most weeds and miss some. Since weed detection requires multiple flights throughout the season, weeds that were not detected in this flight can be detected in a later flight.

The agent was trained on a large variety of different simulated object distributions. The trained agent may work for real-world applications where the number of clusters, the number of objects, and the spatial distribution within the clusters fall within the randomized ranges used during training. However, when there are large differences between the real-world distribution of the weeds and the distribution in the training simulation, changing the simulation parameters and retraining the agent may be required.

During a real-world application, the inference time equals the sum of both the detection network and the DQN. On embedded devices, such as a NVIDIA Jetson Orin, the used detection network, YOLOv8-nano, has an inference time of around 20 ms (Rey et al., 2025). Since the DQN is even smaller, running both the object detection network and the DQN on board a drone in real time should be feasible. Future work will focus on applying the simulation-trained RL policy to a real drone on a real-world application.

4.6. Impact of assumptions on applicability in the real world

Although most of the presented work was based on an abstracted simulated environment and pre-recorded datasets, the proposed method can easily be applied to a real-world scenario with a few additions. The simulator was designed to be used in combination with an object-detection system, prior knowledge

information, and a flight controller (Figure 1). Using a trained object detector, weeds can be detected from a camera image to generate the local weed map as input to the RL, as demonstrated in experiment 5. The resulting discrete flight actions can be executed on a real drone using the drone's flight controller. Below, we discuss the underlying assumptions of the simulation and RL policy, as mentioned in the introduction, and discuss the implications for real-world use:

Drone can accurately execute flight actions: the flight controller of the drone should be able to translate the discrete flight actions into motor signals. The discrete actions are in four directions with a distance determined by the grid-cell size. Due to inaccuracies in the drone's position and attitude, the estimated location of the drone is imperfect. The localization accuracy of the drone should therefore be higher than the grid-cell size, to be able to accurately move the drone to a specific grid-cell. Although not tested in this paper, when using grid-cells of, for example, 1x1 m, a drone equipped with RTK-GPS (accuracy of around 2-3cm) should be more than accurate enough to execute the RL policy. As the use of RTK-GPS is quite standard on agricultural drones, this assumption can be met.

Field size: since the prior knowledge map is related to the field size and the Q-network uses a combination of convolutions and fully-connected layers (Figure 5), a larger field increases the size of the flattened layer and thereby the number of trainable parameters. This will require more training iterations. Experiment 3 showed that the number of found weeds only dropped at a very low prior knowledge quality (and thus very low resolutions), which indicates that the resolution of the prior knowledge may be low. Therefore, larger fields can be used with the same number of trainable parameters by applying a larger down-sampling factor (increasing g_{global}), or by incorporating prior knowledge in a different way.

Prior knowledge of field boundaries: field boundaries are static and available on forehand. In this work, we limited the experiments to squared fields. Although not specifically tested, applying the learned search policy in non-squared fields can be achieved by changing the field-area map, containing information on whether the cell is inside or outside the field. Additionally, obstacles within the field, such as trees, can be avoided by mapping them as non-field areas. As the violation of the field-area map results in a negative reward, it is very likely that the RL agent will learn to deal with it. Work of Theile et al. (2021) showed that an RL-learned policy with a comparable architecture is able to avoid buildings in a city when visiting specific locations using drones by encoding these positions in a no-fly-zone, which is in principle comparable with our field-area map.

A decently trained detection network: due to factors such as variation in the natural environment and motion blur, the output of an object detection network contains errors. The results of experiment 2

showed that the RL agent is robust to these errors in the object detection. Hence, the detection network does not have to be very accurate. Even when the detection network generates a lot of false positives, the RL policy was able to find most of the true positives. Typical object detection performance in agriculture lies around 0.7-0.9 F1-score (Rai et al., 2023; Ruigrok et al., 2023; Rehman et al., 2024). This roughly corresponds with level 'high' in experiment 2, which indicates that the RL agent can deal with input from a real field. Experiment 5 confirmed this by finding most of the weeds in the real-world datasets using the RL agent in combination with a real object detection network.

Some prior knowledge of location of objects: although some level of prior knowledge is required, it does not need to be very detailed. Experiment 3 showed that the prior knowledge did not need to be very detailed or very accurate, however, some prior knowledge was still required. In practice, this prior knowledge can be created by, for example, using the detections from a few higher altitude images (as done in experiment 5), data from previous flights, or other data sources such as satellite images. For many agricultural tasks, like weed detection and disease detection, such rough prior knowledge is available. For instance, the location of many weeds may be predicted by previous years' location (Colbach et al., 2000), and spread very locally. Information from the previous year can then be used as prior knowledge.

The above analysis indicates that all the underlying assumptions of the abstracted RL agent and simulation environment are not problematic for a real-world application of an RL learned search policy. Future research should focus on applying the DQN policy on a real drone to check the influence of the drone's positional and attitude errors on the execution of the task. Especially in use-cases where the target objects are distributed non-uniformly in the field and when it is not crucial to find all objects, there is much to gain when specifically searching for objects instead of flying over and covering the whole field.

5. Conclusion

In this study, we showed that a learned search policy can increase the efficiency of finding weeds using a UAV, particularly when there was some prior knowledge available. When the weeds were non-uniformly distributed, the learned search policy outperformed the baseline row-by-row flight path. The learned policy was robust against errors in the detection, and different qualities in prior knowledge only had a minor influence on the number of weeds found. In addition, the RL agent was able to learn when the search task had to be terminated and when it was not profitable to continue searching. Finally, we demonstrated the transferability of the learned policy to real-world data, achieving a 66% shorter flight path compared to a

row-by-row flight path at the cost of a 10% lower percentage of found weeds.

The major assumptions beneath the presented approach have limited impact on application in real-world agricultural search tasks. For tasks that meet the assumptions as discussed in section 4.6, the method is expected to be applicable.

In conclusion, learning a search policy improves the efficiency of a search task for UAVs for applications where the target objects, such as weeds, are non-uniformly distributed and when there is some level of prior knowledge. Learning to search these objects in the absence of prior knowledge and evaluation of the RL policy on a real drone remains a topic for future work.

CRediT author statement

Rick van Essen: Conceptualization, Methodology, Formal analysis, Software, Visualization, Writing - Original Draft. **Eldert van Henten:** Conceptualization, Writing - Review & Editing, Funding acquisition. **Gert Kootstra:** Conceptualization, Methodology, Writing - Review & Editing, Funding acquisition.

Declaration of Competing Interest

This research is part of the research program SYNERGIA (project number 17626), which is partly financed by the Dutch Research Council (NWO).

Data availability

The simulation and the RL network are made available on `https://github.com/wur-abe/rl_drone_object_search`.

References

Agisoft, 2023. Metashape Professional. URL: `https://www.agisoft.com`. version: 2.0.3.

Albani, D., Manoni, T., Arik, A., Nardi, D., Trianni, V., 2019. Field Coverage for Weed Mapping: Toward Experiments with a UAV Swarm, in: Compagnoni, A., Casey, W., Cai, Y., Mishra, B. (Eds.), Bio-inspired Information and Communication Technologies, Springer International Publishing, Cham. pp. 132–146.

Anul Haq, M., 2022. CNN Based Automated Weed Detection System Using UAV Imagery. Computer Systems Science and Engineering 42, 837–849. doi:`10.32604/csse.2022.023016`.

Azar, A.T., Koubaa, A., Ali Mohamed, N., Ibrahim, H.A., Ibrahim, Z.F., Kazim, M., Ammar, A., Benjdira, B., Khamis, A.M., Hameed, I.A., Casalino, G., 2021. Drone Deep Reinforcement Learning: A Review. Electronics 10, 999. doi:`10.3390/electronics10090999`.

Cardina, J., Johnson, G.A., Sparrow, D.H., 1997. The Nature and Consequence of Weed Spatial Distribution. Weed Science 45, 364–373. Publisher: [Cambridge University Press, Weed Science Society of America].

Castro, G.G.R.D., Berger, G.S., Cantieri, A., Teixeira, M., Lima, J., Pereira, A.I., Pinto, M.F., 2023. Adaptive Path Planning for Fusing Rapidly Exploring Random Trees and Deep Reinforcement Learning in an Agriculture Dynamic Environment UAVs. Agriculture 13, 354. doi:10.3390/agriculture13020354.

Chin, R., Catal, C., Kassahun, A., 2023. Plant disease detection using drones in precision agriculture. Precision Agriculture 24, 1663–1682. doi:10.1007/s11119-023-10014-y.

Chronis, C., Anagnostopoulos, G., Politi, E., Garyfallou, A., Varlamis, I., Dimitrakopoulos, G., 2023. Path planning of autonomous UAVs using reinforcement learning. Journal of Physics: Conference Series 2526, 012088. doi:10.1088/1742-6596/2526/1/012088.

Colbach, N., Forcella, F., Johnson, G.A., 2000. Spatial and temporal stability of weed populations over five years. Weed Science 48, 366–377. doi:10.1614/0043-1745(2000)048[0366:SATSOW]2.0.CO;2.

Dessaint, F., Chadoeuf, R., Barralis, G., 1991. Spatial Pattern Analysis of Weed Seeds in the Cultivated Soil Seed Bank. Journal of Applied Ecology 28, 721–730. doi:https://doi.org/10.2307/2404578.

Druon, R., Yoshiyasu, Y., Kanezaki, A., Watt, A., 2020. Visual Object Search by Learning Spatial Context. IEEE Robotics and Automation Letters 5, 1279–1286. doi:10.1109/LRA.2020.2967677.

Gao, J., Ye, W., Guo, J., Li, Z., 2020. Deep Reinforcement Learning for Indoor Mobile Robot Path Planning. Sensors 20, 5493. doi:10.3390/s20195493.

Girshick, R., 2015. Fast R-CNN. ArXiv:1504.08083 [cs].

Gugan, G., Haque, A., 2023. Path Planning for Autonomous Drones: Challenges and Future Directions. Drones 7, 169. doi:10.3390/drones7030169.

Husnain, A., Mokhtar, N., Shah, N.M., Dahari, M., Iwahashi, M., 2023. A systematic literature review (slr) on autonomous path planning of unmanned aerial vehicles. Drones 2023, Vol. 7, Page 118 7, 118. doi:10.3390/DRONES7020118.

Jocher, G., Qiu, J., Chaurasia, A., 2023. Ultralytics YOLO. URL: https://github.com/ultralytics/ultralytics. version: 8.3.49.

Liu, C., Jian, Z., Xie, M., Cheng, I., 2021. A Real-Time Mobile Application for Cattle Tracking using Video Captured from a Drone, in: 2021 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, Dubai, United Arab Emirates. pp. 1–6. doi:10.1109/isncc52172.2021.9615648.

Lodel, M., Brito, B., Serra-Gomez, A., Ferranti, L., Babuska, R., Alonso-Mora, J., 2022. Where to Look Next: Learning Viewpoint Recommendations for Informative Trajectory Planning, in: 2022 International Conference on Robotics and Automation (ICRA), IEEE, Philadelphia, PA, USA. pp. 4466–4472. doi:10.1109/ICRA46639.2022.9812190.

Mier, G., Valente, J., de Bruin, S., 2023. Fields2cover: An open-source coverage path planning library for unmanned agricultural vehicles. IEEE Robotics and Automation Letters 8, 2166–2172. doi:10.1109/LRA.2023.3248439.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518, 529–533. doi:10.1038/nature14236.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M.E., Stone, P., 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. J. Mach. Learn. Res. 21. Publisher: JMLR.org.

Niroui, F., Zhang, K., Kashino, Z., Nejat, G., 2019. Deep Reinforcement Learning Robot for Search and Rescue Applications:

Exploration in Unknown Cluttered Environments. IEEE Robotics and Automation Letters 4, 610–617. doi:10.1109/LRA.2019.2891991.

Panov, A.I., Yakovlev, K.S., Suvorov, R., 2018. Grid Path Planning with Deep Reinforcement Learning: Preliminary Results. Procedia Computer Science 123, 347–353. doi:10.1016/j.procs.2018.01.054.

Pei, H., Sun, Y., Huang, H., Zhang, W., Sheng, J., Zhang, Z., 2022. Weed Detection in Maize Fields by UAV Images Based on Crop Row Preprocessing and Improved YOLOv4. Agriculture 12, 975. doi:10.3390/agriculture12070975.

Popovic, M., Hitz, G., Nieto, J., Sa, I., Siegwart, R., Galceran, E., 2017. Online informative path planning for active classification using uavs. Proceedings - IEEE International Conference on Robotics and Automation , 5753–5758doi:10.1109/ICRA.2017.7989676.

Popović, M., Ott, J., Rückin, J., Kochenderfer, M.J., 2024. Learning-based methods for adaptive informative path planning. Robotics and Autonomous Systems 179, 104727. doi:10.1016/j.robot.2024.104727.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N., 2021. Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research 22, 1–8.

Rai, N., Zhang, Y., Ram, B.G., Schumacher, L., Yellavajjala, R.K., Bajwa, S., Sun, X., 2023. Applications of deep learning in precision weed management: A review. Computers and Electronics in Agriculture 206, 107698. doi:10.1016/j.compag.2023.107698.

Rehman, M.U., Eesaar, H., Abbas, Z., Seneviratne, L., Hussain, I., Chong, K.T., 2024. Advanced drone-based weed detection using feature-enriched deep learning approach. Knowledge-Based Systems 305, 112655. doi:10.1016/j.knosys.2024.112655.

Rejeb, A., Abdollahi, A., Rejeb, K., Treiblmaier, H., 2022. Drones in agriculture: A review and bibliometric analysis. Computers and Electronics in Agriculture 198, 107017. doi:10.1016/j.compag.2022.107017.

Rey, L., Bernardos, A.M., Dobrzycki, A.D., Carramiñana, D., Bergesio, L., Besada, J.A., Casar, J.R., 2025. A Performance Analysis of You Only Look Once Models for Deployment on Constrained Computational Edge Devices in Drone Applications. Electronics 14, 638. doi:10.3390/electronics14030638.

Rivas, A., Chamoso, P., González-Briones, A., Corchado, J.M., 2018. Detection of Cattle Using Drones and Convolutional Neural Networks. Sensors 18, 2048. doi:10.3390/s18072048. publisher: MDPI AG.

Ruigrok, T., Van Henten, E.J., Kootstra, G., 2023. Improved generalization of a plant-detection model for precision weed control. Computers and Electronics in Agriculture 204, 107554. doi:10.1016/j.compag.2022.107554.

Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: an introduction. MIT press, Cambridge, Massachusetts, USA.

Tang, J., Liang, Y., Li, K., 2024. Dynamic Scene Path Planning of UAVs Based on Deep Reinforcement Learning. Drones 8, 60. doi:10.3390/drones8020060.

Theile, M., Bayerlein, H., Nai, R., Gesbert, D., Caccamo, M., 2020. UAV Coverage Path Planning under Varying Power Constraints using Deep Reinforcement Learning, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Las Vegas, NV, USA. pp. 1444–1449. doi:10.1109/IROS45743.2020.9340934.

Theile, M., Bayerlein, H., Nai, R., Gesbert, D., Caccamo, M., 2021. UAV Path Planning using Global and Local Map Information with Deep Reinforcement Learning, in: 2021 20th International Conference on Advanced Robotics (ICAR), IEEE, Ljubljana, Slovenia. pp. 539–546. doi:10.1109/ICAR53236.2021.9659413.

Tu, G.T., Juang, J.G., 2023. Uav path planning and obstacle avoidance based on reinforcement learning in 3d environments. Actuators 2023, Vol. 12, Page 57 12, 57. doi:10.3390/ACT12020057.

van Essen, R., van Henten, E., Kooistra, L., Kootstra, G., 2025. Adaptive path planning for efficient object search by uavs in

agricultural fields. Smart Agricultural Technology 12, 101075. doi:https://doi.org/10.1016/j.atech.2025.101075.

Westheider, J., Rückin, J., Popović, M., 2023. Multi-UAV Adaptive Path Planning Using Deep Reinforcement Learning, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Detroit, MI, USA. pp. 649–656. doi:10.1109/IROS55552.2023.10342516.

Xu, K., Shu, L., Xie, Q., Song, M., Zhu, Y., Cao, W., Ni, J., 2023. Precision weed detection in wheat fields for agriculture 4.0: A survey of enabling technologies, methods, and research challenges. Computers and Electronics in Agriculture 212, 108106. doi:10.1016/j.compag.2023.108106.

Yang, W., Wang, X., Farhadi, A., Gupta, A., Mottaghi, R., 2018. Visual Semantic Navigation using Scene Priors. ArXiv:1810.06543 [cs].

Yu, J., Su, Y., Liao, Y., 2020. The path planning of mobile robot by neural networks and hierarchical reinforcement learning. Frontiers in Neurorobotics 14, 63. doi:10.3389/FNBOT.2020.00063/BIBTEX.

Zhang, C., Valente, J., Wang, W., Guo, L., Tubau Comas, A., Van Dalfsen, P., Rijk, B., Kooistra, L., 2023. Feasibility assessment of tree-level flower intensity quantification from UAV RGB imagery: A triennial study in an apple orchard. ISPRS Journal of Photogrammetry and Remote Sensing 197, 256–273. doi:10.1016/j.isprsjprs.2023.02.003. publisher: Elsevier BV.