Aerial View Localization with Reinforcement Learning:
Towards Emulating Search-and-Rescue

Aleksis Pirinen[1], Anton Samuelsson[2], John Backsund[2] and Kalle Åström[2]

[1]RISE Research Institutes of Sweden

[2]Centre for Mathematical Sciences, Lund University, Sweden

{aleksis.pirinen@ri.se, anton.b.samuelsson@gmail.com,
j.backsund@gmail.com, karl.astrom@math.lth.se}

Abstract— **Climate-induced disasters are and will continue to be on the rise, and thus search-and-rescue (SAR) operations, where the task is to localize and assist one or several people who are missing, become increasingly relevant. In many cases the rough location may be known and a UAV can be deployed to explore a given, confined area to precisely localize the missing people. Due to time and battery constraints it is often critical that localization is performed as efficiently as possible. In this work we approach this type of problem by abstracting it as an *aerial view goal localization* task in a framework that emulates a SAR-like setup without requiring access to actual UAVs. In this framework, an agent operates on top of an aerial image (proxy for a search area) and is tasked with localizing a goal that is described in terms of visual cues. To further mimic the situation on an actual UAV, the agent is not able to observe the search area in its entirety, not even at low resolution, and thus it has to operate solely based on partial glimpses when navigating towards the goal. To tackle this task, we propose *AiRLoc*, a reinforcement learning (RL)-based model that decouples exploration (searching for distant goals) and exploitation (localizing nearby goals). Extensive evaluations show that AiRLoc outperforms heuristic search methods as well as alternative learnable approaches, and that it generalizes across datasets, e.g. to disaster-hit areas without seeing a single disaster scenario during training. We also conduct a proof-of-concept study which indicates that the learnable methods outperform humans on average. Code and models have been made publicly available at https://github.com/aleksispi/airloc.**

I. INTRODUCTION

Recent technological developments of unmanned aerial vehicles (UAVs) and satellites have resulted in an enormous increase in the amount of aerial view landscape and urban data that is available to the public [4], [15], [28], [12], [38], [26], [36]. An important application area of UAVs is within search-and-rescue (SAR) operations, where the task is to localize and assist one or several people who are missing, for example after a natural disaster. It may often be the case that the people in need are known to be within a confined area, such as within a specific neighborhood or city block. In such a scenario, a UAV can be used to explore the area from an aerial perspective to precisely localize and subsequently assist the missing people. Obviously, controlling the UAV in an informed and intelligent manner, rather than exhaustively scanning the whole area, could significantly improve the likelihood of succeeding with the operation.

In this paper, we propose a novel setup and task formulation that allows for controllable and reproducible develop-ment of and experimentation with systems for UAV-based SAR operations.[1] More specifically, we abstract the problem within a framework that emulates a SAR-like setup without requiring access to actual UAVs. In this framework, an agent operates on top of an aerial image (proxy for a specific search area) and is tasked with localizing a goal for which coordinates are not available, but where some visual cues of the goal are provided. For our task, which we denote *aerial view goal localization*, we assume that the visual cues are given in terms of a top-view observation of the goal within the search area (see Fig. 1). This provides a streamlined proxy setup, but note that in a real SAR operation such cues could instead be provided e.g. by the missing people, assuming they have been able to send information about their surroundings (e.g. ground-level images). The active localiza-tion methodologies we propose can easily be extended to allow for more flexible goal specifications, for example by integrating an off-the-shelf geo-localization module.

There are many cases where GPS coordinates of the goal location are not available, or where such information is not reliable (e.g. because global satellite navigation systems are susceptible to radio frequency interruptions and fake signals). Hence there is a need for robust aerial localization systems that do not rely on global positional information, but that can operate reliably based on visual information alone. Moreover, to further mimic the situation on an actual UAV, it is assumed in our task that only a partial glimpse of the search area can be observed at the same time. In many cases, a UAV could elevate to a higher altitude to get a generic (lower-resolution) sense of the whole search area, but there are also conditions which makes this impractical, e.g. if the battery of the UAV is running low. Adverse weather conditions could also make it risky or impossible to operate at a high altitude.

To tackle our suggested aerial view goal localization task, we propose *AiRLoc*, a reinforcement learning (RL)-based model that decouples exploration (searching for distant goals) and exploitation (localizing nearby goals) – see Fig. 1. Extensive experimental results show that AiRLoc outper-forms heuristic search methods and alternative learnable approaches. The results also show that AiRLoc generalizes across datasets, e.g. to disaster-hit areas without seeing a

---

[1]Also relevant for many types of environmental monitoring applications, e.g. in forestry management.

single disaster scenario during its training phase. We also conduct a proof-of-concept study which indicates that this task is difficult even for humans.

II. RELATED WORK

Several prior works have proposed methods for autonomous control a UAVs [27], [13], [6], [3], [24], [41], [19]. Many of these works (e.g. [27], [24], [41]) revolve around methodologies for efficient scanning of large areas (e.g. agricultural landscapes) such that certain types of global-level downstream inferences – such as determining the health status of a field of crops – can be accurately performed based on a limited number of high-resolution observations. Aside from differing in task formulation (ours requiring precise localization of a particular goal, while the aforementioned works often revolve around global-level inference), these prior works assume access to a global lower-resolution observation of the whole area of interest, while we do not. There are also works that are closer to us in terms of task setup [3], [13], [6]. For example, [3] propose a hierarchical planning approach for a goal reaching task, where a rough plan is first proposed using A*. This rough plan is subsequently used as an initial guess by a finer-grained planner which parametrizes the initial trajectory as continuous B-splines and performs trajectory optimization. Different from us, their system assumes access to ground truth detections of moving objects and ground classifications.

Our work is also related but orthogonal to the increasingly studied problem of geo-localization [35], [30], [42], [40], [20], [33]. Such works aim to infer relationships between two or more images from different perspectives, e.g. predicting the satellite or drone view corresponding to a ground-level image. Most such methods perform this task by an exhaustive comparison within a large image set, and are thus very different to our setup which instead revolves around minimizing the amount of observations when performing localization. However, our proposed methodologies could further benefit from incorporating geo-localization methods. For example, if the goal location is specified from a ground-level perspective, which may be more realistic in practice, geo-localization methods can be used to match the top-view images observed by our proposed method during goal localization.

From a pure task formulation perspective, and setting aside the application areas, our setup may be most closely related to embodied image goal navigation [1], [43], [14]. In this framework, an agent is tasked to navigate in a first-person perspective within a 3d environment towards a goal location which is specified as an image within the environment. On the one hand, the embodied setting may sometimes be more challenging than our setup, since the exploration trajectories are typically longer (as the agent moves a significantly smaller extent per action) and because exploration is performed among obstacles (e.g. walls and furniture). On the other hand, embodied first person agents may often observe the goal from far away (e.g. from the other side of a newly entered room), while our formulation

is more challenging in that the goal can never be observed in any way prior to reaching it.

To the best of our knowledge, in addition to us relatively few prior works have considered inference based solely on partial glimpses of an underlying image [21], [22]. In contrast, most earlier RL-based methods that have been proposed for computer vision tasks – e.g. for object detection [5], [8], [18] and aerial view processing [29], [2] – assume access to at least a low-resolution version of the entire scene or image being processed. Even the seminal work by [16] uses lower-resolution full image input in addition to high-resolution partial glimpses during its sequential processing, even though in principle it may be possible to re-design the system to operate based on high-resolution glimpses alone.

III. AERIAL VIEW GOAL LOCALIZATION

In this section we first explain in detail our proposed aerial view goal localization task and framework (§III-A). Then, in §III-B, we explain AiRLoc, our reinforcement learning (RL)-based approach for tackling this task. See Fig. 1 for an overview. Finally, §III-C describes the baseline methods we have developed and that we evaluate and compare with AiRLoc in §IV.

A. Task Description

The task is executed by an agent within a *search area*, which is discretized as an $M \times N$ grid that is layered on top of a given aerial image (with a small distance between each grid cell, to avoid overfitting models to edge artefacts). Every grid cell within the search area corresponds to a valid position p_t of the agent, and the agent can only directly observe the image content O_t of its current cell. In each episode, one of the grid cells corresponds to the goal that the agent should localize. The image content of the goal cell is denoted O^{goal} and its position is denoted p^{goal}. Note that the goal position p^{goal} is *never* observed by the agent; it is only used to determine if the agent is successful. The task is considered successfully completed as soon as the agent's current position p_t and the goal position p^{goal} coincide,[2] i.e. when $p_t = p^{\text{goal}}$.

In each episode, the agent's start location p_0 and the goal location p^{goal} are sampled at uniform random within the search area ($p_0 \neq p^{\text{goal}}$). The agent then moves around until it either reaches the goal ($p_t = p^{\text{goal}}$), or a maximum number of steps T have been taken. This limit T is included to represent time and resource constraints. In our task formulation, an agent has eight possible actions, which correspond to moving to any of its eight adjacent locations (grid cells). An agent may in general move outside the search area, and if so, the agent receives an entirely black observation. There is never any advantage to moving outside the search area, and thus it should be avoided (it is easy to avoid given p_t).

---

[2]A reasonable next step would be to require that an agent has to declare when it has reached its goal.
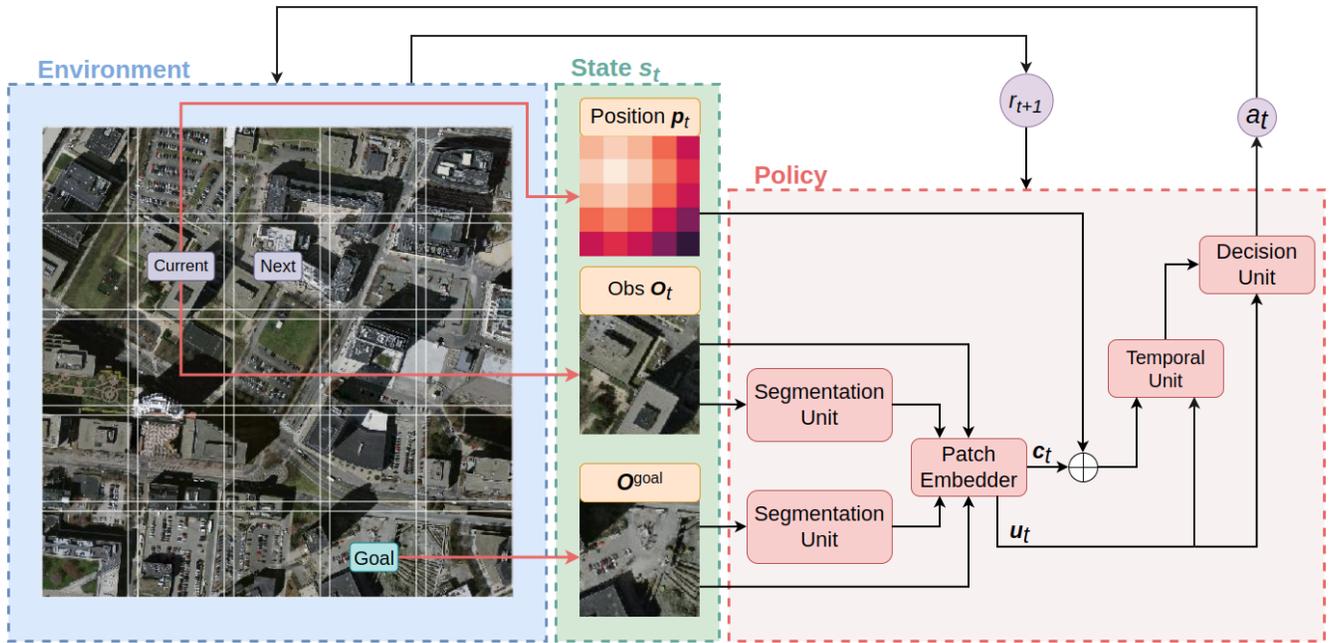
Fig. 1. Overview of *AiRLoc*, our RL-based agent for aerial view goal localization. The state s_t consists of the agent's current position p_t, its currently observed patch O_t, and the goal patch O^{goal}. First, segmentation masks for O_t and O^{goal} are computed, and O_t, O^{goal} and their segmentations are then fed through a patch embedder to generate a common representation c_t. The positional encoding p_t is then added to c_t, and the sum, together with an exploitation prior u_t (see §III-B), are subsequently processed by an LSTM, whose output is fed to a decision unit. The decision unit also receives u_t and outputs an action probability distribution $\pi(\cdot|s_t)$. A movement a_t is then sampled from $\pi(\cdot|s_t)$, which results in the next state s_{t+1} and reward r_{t+1} (a reward is provided during training only). The process is repeated, either until the agent reaches the goal, or until a maximum number of steps T have been taken. Note that AiRLoc never observes the full search area, not even at a low resolution.

B. AiRLoc Model

In this section we describe *AiRLoc*, the reinforcement learning (RL)-based model we propose for tackling the aerial view goal localization task; see an overview in Fig. 1.

States, actions and rewards. The state s_t contains the currently observed patch O_t, the goal patch O^{goal}, and an encoding $p_t \in \mathbb{R}^{256}$ of the agent's position. As described above, AiRLoc has eight possible actions a_t, which correspond to moving to any of its adjacent locations. During training, a negative reward is provided for each action that does not move the agent into the goal location, and a positive reward is provided when the goal is found. Specifically, after taking action a_{t-1} in state s_{t-1} the reward $r_t = 3 \cdot \mathbb{1}\left(p_t = p^{\text{goal}}\right) - 1$ is provided, where $\mathbb{1}$ is the indicator function.

Policy overview In each step, the state s_t is processed by four modules to generate the current action distribution $\pi_{\theta}(*|s_t)$, where θ denotes all learnable parameters. First, O_t and O^{goal} are passed through a pretrained *segmentation unit* (a U-net [23]) which predicts building segmentation masks for O_t and O^{goal}, respectively. Second, O_t and O^{goal} and their segmentations are passed through a *patch embedder* which yields a low-dimensional embedding $c_t \in \mathbb{R}^{256}$ of what the agent observes and what it aims to localize. The patch embedder also outputs an exploitation prior $u_t \in \mathbb{R}^8$ (described more below). Third, p_t is added to c_t and the result and u_t are passed to an LSTM-based *temporal unit*

[10] which integrates information over time. Finally, the LSTM output and u_t are passed to a *decision unit* which yields the probability distribution $\pi_{\theta}(*|s_t)$. This decision unit first projects the LSTM's output into the action space dimensionality, then adds the exploitation prior u_t, and finally generates an action distribution using softmax. Note that we use an LSTM rather than a Transformer for the temporal unit, since we want to keep the overall architecture lightweight – the model weights occupy less than 4 MB of memory, and inference can be efficiently performed even without a GPU.

Patch embedder. The patch embedder should extract relevant information about the relationship between O_t and O^{goal}. To achieve this, we use an architecture similar to that by [7], who consider a self-supervised visual representation learning task where the spatial displacement between a pair of adjacent random crops from an image should be predicted. Note that when the start location p_0 is adjacent to the goal location p^{goal}, and when the movement budget $T = 1$, our task becomes equivalent to the representation learning task introduced by [7]. Our patch embedder architecture consists of two parallel branches with four convolutional layers (ReLUs and max pooling are applied between layers). First, O_t and O^{goal}, with their segmentations channel-wise concatenated, are fed separately into one branch each. To enable early information sharing between the agent's current patch and the goal patch, after two convolutional layers, the outputs of the two branches

are concatenated and sent through the rest of their respective branches. The two resulting 128-dimensional embeddings are then concatenated and the result is passed through a dense layer with output $c_t \in \mathbb{R}^{256}$.

Pretraining backbone vision components is common in RL setups, since it often yields a higher end performance [25], [17], [32], [37], [39]. We therefore pretrain the patch embedder in the same self-supervised fashion as [7]. During pretraining, another dense layer (with input c_t) is attached to produce an 8-dimensional output u_t which is fed to a softmax function. The eight outputs correspond to the possible locations of O^{goal} relative to O_t, assuming these are adjacent. When using the patch embedder within AiRLoc, we take advantage of both c_t and u_t, cf. Fig. 1. Note that u_t can be interpreted as an *exploitation prior*, as it is specifically tuned towards localizing ('exploiting') adjacent goals. Thus, feeding u_t to the temporal unit as well as directly to the decision unit allows AiRLoc to learn when to explore and when to exploit (without u_t, the same policy must be able to both localize adjacent goals *and* explore far-away goals). The choice of using both c_t and u_t is empirically justified in §IV-B.

Positional encoding. Positional information is represented similarly to Transformers [31]. Note that AiRLoc never receives global positional information, i.e. it is always relative to a given search area. Such information may be available during SAR within a confined area, where a UAV can keep track of its location relative to the borders of this area. Let (x, y) denote the agent's coordinates within the $M \times N$-sized search area (thus $x \in \{0, \ldots, M - 1\}$, $y \in \{0, \ldots, N - 1\}$). Then the i:th element p_t^i of the positional encoding vector $p_t \in \mathbb{R}^d$ (with d even; for us $d = 256$) is given by:

$$
p_t^i = \begin{cases}
\cos\left(x/100^{2(i-1)/(d/2)}\right) & \text{if } i \in \{1, 3, \ldots, d/2 - 1\} \\
\sin\left(x/100^{2i/(d/2)}\right) & \text{if } i \in \{2, 4, \ldots, d/2\} \\
\cos\left(y/100^{2(i-1)/(d/2)}\right) & \text{if } i \in \{d/2 + 1, \ldots, d - 1\} \\
\sin\left(y/100^{2i/(d/2)}\right) & \text{if } i \in \{d/2 + 2, \ldots, d\}
\end{cases}
\tag{1}
$$

Policy training. To learn the parameters of AiRLoc, we first pretrain the patch embedder in a self-supervised fashion (without RL) as described above. We then freeze the patch embedder weights and train the rest of AiRLoc using REIN-FORCE [34]. We employ within-batch reward normalization based on distance left to the goal, i.e. rewards associated with states of equal distance to the goal are grouped and normalized to zero mean and unit variance. We use a pretrained segmentation unit (one can simply use an off-the-shelf aerial view segmentation model) and it is not refined during policy training.

C. Baselines

In §IV we compare AiRLoc with the following baselines:

- **Priv random** selects actions randomly, with two exceptions: i) it cannot move outside the search area; ii) it avoids previous locations.

- **Local** selects actions by repeatedly calling the pretrained patch embedder (which assumes the goal is adjacent to the current location).
- **Priv local** is the same as *Local* but with the privileged movement restrictions of *Priv random*.
- **Human** represents the average human performance from a proof-of-concept evaluation with 19 subjects (see details in the appendix).

IV. EXPERIMENTS

In this section we extensively evaluate and compare AiRLoc and the various baselines described in §III-B and §III-C, respectively. First we however describe what datasets and evaluation metrics we use, explain different variants of AiRLoc, and provide some further implementation details.

Datasets. We mainly use *Massachusetts Buildings* (*Masa*) by [15] for development and evaluation (70% for training; 15% each for validation and testing). The data contains images of Boston and the surrounding suburban and forested areas. It depicts houses, roads and other clearly identifiable man-made structures, but also woods and less developed regions. The data also includes segmentation masks for buildings, which are used to separately train the segmentation unit (cf. Fig. 1) that is used by most of the learnable models in the results below. Models are also evaluated on the *Dubai* dataset [28], which also depicts urban regions, although the surrounding areas are instead dry deserts. This dataset is hence used to assess the generalization of the various methods. Finally, we also train and evaluate on the *xBD* dataset by [9], which contains satellite images from various regions both before (*xBD-pre*) and after (*xBD-disaster*) various natural distastes, e.g. wildfires and floods. In this case the models are trained on non-disaster-hit data from *xBD-pre* and evaluated on *xBD-disaster*, where we also ensure that the training data depicts other geographical areas than those in *xBD-disaster*.

Evaluation metrics. We use the following evaluation metrics. *Success* is the percentage of episodes where the goal is reached. *Steps* is the average number of actions taken per episode (for failure episodes this is set to the movement budget T). *Step ratio* measures the average ratio between the taken number of steps and the minimum number of steps required (lower is better). It is only computed for successful trajectories. *Residual distance* measures the average distance between the final location relative to the goal location in unsuccessful episodes (lower is better). Finally, *Runtime* is the average runtime per episode.

AiRLoc variants. We also train and evaluate several ablated variants of AiRLoc. *No sem seg* omits the segmentation unit and uses only RGB patches in the patch embedder (which is instead pretrained with RGB-only inputs). *No residual* omits u_t in the decision unit, but not in the temporal unit, cf. Fig. 1. Finally, *no prior* entirely discards the prior u_t in the architecture.

TABLE I

RESULTS ON THE TEST SET OF *Massachusetts Buildings* (MOVEMENT BUDGET $T = 10$ AND $T = 14$ FOR SETUPS OF SIZES 5×5 AND 7×7, RESPECTIVELY). FOR BOTH SEARCH AREA SIZES, THE SUCCESS RATE OF AiRLoc IS HIGHER THAN FOR THE BASELINES. MID-LEVEL VISION CAPABILITIES (SEMANTIC SEGMENTATION) ARE CRUCIAL FOR AiRLoc'S PERFORMANCE. THE STANDARD LOCAL APPROACH PERFORMS POORLY AND IS SIGNIFICANTLY IMPROVED BY IMPOSING THE PRIVILEGED MOVEMENT CONSTRAINTS. THE TIME PER EPISODE IS LOW FOR ALL METHODS.

Agent type	Success	Step ratio	Steps	Residual distance	Runtime
AiRLoc (5x5)	67.6 %	1.45	6.2	2.4	120 ms
Priv local (5x5)	64.2 %	1.59	6.5	2.4	117 ms
Local (5x5)	24.7 %	1.47	8.1	7.0	138 ms
Priv random (5x5)	41.0 %	2.56	8.0	1.6	48 ms
AiRLoc (7x7)	59.0 %	1.52	9.4	3.3	188 ms
Priv local (7x7)	56.3 %	1.72	9.9	3.4	178 ms
Local (7x7)	17.8 %	1.20	11.9	8.7	202 ms
Priv random (7x7)	25.2 %	1.82	12.3	3.5	74 ms
AiRLoc (no sem seg, 5x5)	61.7 %	1.54	6.7	2.4	94 ms
Priv local (no sem seg, 5x5)	61.6 %	1.67	6.8	2.4	88 ms
Local (no sem seg, 5x5)	20.5 %	1.28	8.4	6.2	92 ms
AiRLoc (no sem seg, 7x7)	52.5 %	1.61	10.1	3.5	141 ms
Priv local (no sem seg, 7x7)	51.1 %	1.89	10.2	3.3	133 ms
Local (no sem seg, 7x7)	14.1 %	1.37	12.4	8.0	136 ms

Fig. 2. Examples of AiRLoc (red) and *Priv local* (dashed green) on the test set of *Masa* (left, middle) and *Dubai* (right). Left: AiRLoc takes the same first two actions as *Priv local* and then takes the shortest path to the goal ('G'). *Priv local* also reaches the goal. Middle: AiRLoc first deviates from *Priv local* and then follows the same path. AiRLoc reaches the goal faster. Right: AiRLoc follows the same path as *Priv local* until it is adjacent to the goal and then moves into the goal, while *Priv local* fails.

Implementation details. All methods are implemented in, trained and evaluated using PyTorch. Training AiRLoctakes 30h on a Titan V100 GPU. To learn the parameters of the policy networks, we use REINFORCE [34] with Adam [11], batch size 64, search area size $M \times N = 5 \times 5$, movement budget $T = 10$, learning rate 10^{-4}, and discount $\gamma = 0.9$. The grid cells of the search areas are of size $48 \times 48 \times 3$, with 4 pixels between each other to avoid overfitting models to edge artefacts (each cell corresponds to roughly 100×100 meters). Each model is trained until convergence on the validation set (typically happens within 50k batches). We apply left-right and top-down flipping of images (search areas) as data augmentation. The AiRLoc variants are trained with five random network initializations each, and the results for the median-performing models on the validation set are reported below. AiRLoc is not seed

sensitive, as shown in §IV-C. Unless otherwise specified, all models are evaluated in deterministic mode, i.e. the most probable action is selected in each step. All models are evaluated on the exact same start configurations for fair comparisons.

A. Main Results

In Table I we compare AiRLoc to the heuristic random and learnable local baselines on the test set of *Massachusetts Buildings* (*Masa*). AiRLoc obtains a higher success rate than the baselines, both in search areas of size 5×5 and 7×7 (AiRLoc is only trained in the 5×5 setting). AiRLoc and *Priv local* have roughly the same runtime per trajectory, and note that all methods have runtimes that would be negligible compared to the movement overhead of an actual UAV. It is also clear that the segmentation model is crucial, which is in line with prior works that find that mid-level vision capabilities are important for high performance in

Agent type	Success	Step ratio	Steps	Residual distance	Runtime
AiRLoc (5x5)	68.8 %	1.52	6.3	2.4	126 ms
Priv local (5x5)	65.6 %	1.59	6.5	2.4	113 ms
Local (5x5)	23.5 %	1.23	8.2	6.6	136 ms
Priv random (5x5)	41.0 %	1.96	8.0	2.5	48 ms
AiRLoc (7x7)	57.2 %	1.54	9.7	3.4	194 ms
Priv local (7x7)	53.7 %	1.85	10.2	3.6	184 ms
Local (7x7)	15.5 %	1.25	12.2	7.9	207 ms
Priv random (7x7)	26.9 %	1.64	12.0	3.5	72 ms
AiRLoc (no sem seg, 5x5)	67.1 %	1.59	6.5	2.4	91 ms
Priv local (no sem seg, 5x5)	65.1 %	1.67	6.6	2.5	86 ms
Local (no sem seg, 5x5)	23.3 %	1.25	8.2	6.6	90 ms
AiRLoc (no sem seg, 7x7)	48.6 %	1.56	10.3	3.3	144 ms
Priv local (no sem seg, 7x7)	41.9 %	1.69	10.8	3.4	140 ms
Local (no sem seg, 7x7)	15.0 %	1.28	12.3	7.6	135 ms

Agent type	Success	Steps	Runtime	Success	Steps	Runtime
AiRLoc (5x5)	66.1 %	6.5	130 ms	72.8 %	6.1	122 ms
Priv local (5x5)	63.8 %	6.7	121 ms	67.3 %	6.4	115 ms
Priv random (5x5)	40.8 %	7.9	48 ms	40.8 %	7.9	48 ms
AiRLoc (7x7)	50.7 %	10.2	204 ms	55.7 %	9.9	198 ms
Priv local (7x7)	50.5 %	10.2	184 ms	53.6 %	10.0	180 ms
Priv random (7x7)	25.5 %	12.2	74 ms	25.5 %	12.2	74 ms

RL-vision setups [25]. As seen in Table II, AiRLoc and the best alternative learnable approach *Priv local* generalize excellently to an entirely new dataset.

Table III contains results on *xBD-disaster*; these results are particularly relevant from a perspective of SAR-operations in disaster-hit areas. Columns 1-3 show that AiRLoc generalizes quite well from having been trained on an entirely different dataset (*Masa*), which depicts non-disaster-hit urban areas, to disaster-hit areas at various other spatial locations. Results are however improved further (columns 4-6) if models are first trained on non-disaster-hit images from the same dataset (*xBD-pre*) and then evaluated at different locations that depict disaster-hit scenarios.

In summary, AiRLoc outperforms the baselines across all datasets and search area sizes, and localizes goals in fewer steps on average. See Fig. 2 and Fig. 5 - 6 (the latter two are on the last page) for visualizations of AiRLoc and *Priv local*.

Human performance evaluation. The results of the proof-of-concept human performance evaluation in Fig. 3 (left) indicate that our proposed task is in general difficult, since only slightly above half of all human controlled trajectories are successful. We also see that AiRLoc and *Priv local* achieve significantly higher success rates compared to human operators. Details about the human performance evaluation are found in the appendix.

B. Ablation Study: Motivating the Exploitation Prior

In Fig. 3 we evaluate the various AiRLoc variants described earlier, together with the best non-RL-based model *Priv local* and the human baseline. AiRLoc is better than its ablated variants on average in both settings (5×5 and 7×7), as well as for most start-to-goal distances (exception at distance 4 in the 7×7 setting). This motivates the design choice of fully utilizing the exploitation prior within the policy architecture – see also Table IV.

Recall that *Priv local* is trained solely in the setting where the start and goal are adjacent, so it can be interpreted as an 'exploitation only' model, where the action distribution is obtained by feeding the exploitation prior $\boldsymbol{u}_t$ through a softmax, cf. Fig. 1. Conversely, the *no prior* variant of AiRLoc is trained without any exploitation prior, so the policy must simultaneously learn to explore (search for the goal when it is further away) and exploit (move to the goal when it is adjacent), which may be ambiguous. As seen in
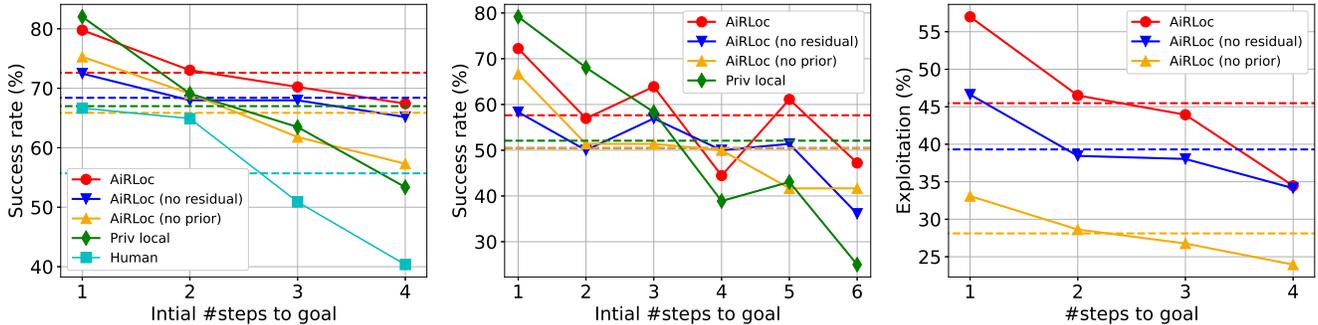
Fig. 3. Left and mid: Success rate versus start-to-goal distance on the validation set of *Masa* (averages are dashed). Search areas are of size $M \times N = 5 \times 5$ and $T = 10$ (left) or 7×7 and 14 (middle). Left: The methods are generally more successful when the start is closer to the goal. AiRLoc and *Priv local* achieve higher success rates than human operators. AiRLoc performs roughly on par with *Priv local* when the goal and start are adjacent (*Priv local* is trained only in this setting) and outperforms it at larger distances. AiRLoc is also more successful than its ablated variants in all settings. Middle: AiRLoc is best on average, despite having only been trained in the 5×5 setting. *Priv local* is better when the start and goal are close to each other, while AiRLoc is better when they are three or more steps apart. Right: How frequently AiRLoc selects the same action as the exploitation prior (argmax of $\boldsymbol{u}_t$) versus goal distance. The full AiRLoc agent has the largest variability in exploitation versus exploitation depending on distance to goal.

Fig. 3, the *no residual* variant, which allows $\boldsymbol{u}_t$ to guide the agent's decision making by feeding $\boldsymbol{u}_t$ to the temporal unit, is only marginally better. Our full AiRLoc agent, which clearly outperforms the other variants, takes this a step further by decoupling exploration and exploitation and only has to learn a residual between the two (since $\boldsymbol{u}_t$ is added within the softmax of the decision unit). Hence, during RL training AiRLoc essentially learns when to explore and when to exploit.

C. Random Seed Sensitivity Analysis

Table IV shows the results of a seed sensitivity analysis (regarding policy network initilization) for AiRLoc and its ablated variants on the validation set of *Massachusetts Buildings*. The AiRLoc variants are trained with five random network initializations each until convergence on the validation set, and the results for the median-performing models on the validation set are the ones reported within the rest of the paper. The seed sensitivity is low overall. Furthermore, our full AiRLoc agent outperforms *Priv local* even for the worst-performing seed.

V. CONCLUSIONS

In this work we have introduced the novel *aerial view goal localization* task and framework, which allows for controllable and reproducible development of methodologies that can eventually be useful for automated search-and-rescue operations, e.g. in regions that are heavily affected by climate-induced disasters. Naturally, as with most technologies, there are also possible applications that may be unethical. We strongly discourage extending our research in such directions, and instead call for extensions towards benign use-cases.

The difficulty for humans to perform well on our proposed task shows that it is a reasonable first step for model development and evaluation, even though the setup avoids some challenges of real use-cases. Relevant next steps toward making the proposed methodologies more practically useful include making the goal specification more flexible (e.g. allowing for a ground-level image description of the goal); requiring the agent to explicitly declare when it has reached its goal; and considering even larger search areas.

An RL-based approach, *AiRLoc*, was developed to tackle the proposed task, in addition to several other learnable and heuristic methods. Key components of the policy architecture include a mid-level vision module and an explicit decoupling between exploration and exploitation, both of which were shown to be crucial for AiRLoc's performance. Extensive experimental evaluations clearly showed the benefits of our AiRLoc agent over the learnable and heuristic baselines. In particular, our methodology can be used to localize goals in aerial images depicting disaster zones, despite being trained only on scenarios without disasters. Code and models have been made publicly available[3] so that others can further explore and extend our proposed task towards real use-cases, for example within disaster relief and management.

APPENDIX

In this appendix we provide further details about the human performance evaluation. To compare the performance of AiRLoc with a human operator in a similar setting, a game version of the task was developed. For fair comparisons, this game was designed to resemble how AiRLoc perceives the search area. Therefore, in addition to receiving the current and goal patches, the human operator is also aware of the borders of the search area, and knows the current position as well as the history of all previously visited positions within the confined area – see Fig. 4. In fact, the human operator can even see all the previously visited patches, while this information is not provided to AiRLoc. We decided to provide humans with this additional information as they have not been trained for the task at hand. Based on this input, the human operator can move to any of the eight adjacent patches. The movement is selected by clicking with a mouse cursor on one of the eight dark squares surrounding the current location in the *Player Area*, shown on the left in Fig. 4. The game uses search areas of size 5×5 and ends either when the movement budget $T = 10$ is exhausted or

---

[3]https://github.com/aleksispi/airloc

TABLE IV

SEED SENSITIVITY ANALYSIS OF THE VARIOUS AiRLoc VARIANTS ON THE VALIDATION SET OF *Massachusetts Buildings* (SEARCH AREA SIZE 5×5, MOVEMENT BUDGET $T = 10$). THE RESULTS ON THE FIRST LINES OF EACH BLOCK ARE THE MEDIAN-PERFORMING AiRLoc MODELS AND ARE THE ONES WE HAVE EVALUATED IN THE REST OF THE PAPER. NONE OF THE AiRLoc VARIANTS ARE SENSITIVE TO THE RANDOM SEED USED FOR POLICY NETWORK INITIALIZATION. THE WORST PERFORMING SEED OF THE *no residual* VARIANT OF AiRLoc PERFORMS BETTER THAN THE BEST PERFORMING SEED OF THE *no prior* VARIANT, AND IT IS ALSO SOMEWHAT BETTER THAN THE ALTERNATIVE LEARNABLE APPROACH *Priv local*. SIMILARLY, THE WORST PERFORMING SEED OF OUR FULL AiRLoc OUTPERFORMS THE BEST PERFORMING SEED OF BOTH THE ABLATED VARIANTS AND *Priv local*, WHICH AGAIN MOTIVATES OUR DESIGN CHOICES.

Agent type	Success	Step ratio	Steps	Residual distance
AiRLoc	72.6 %	1.49	6.0	2.4
AiRLoc (other seed #1)	72.2 %	1.45	6.1	2.4
AiRLoc (other seed #2)	72.2 %	1.51	6.2	2.5
AiRLoc (other seed #3)	74.3 %	1.56	6.2	2.4
AiRLoc (other seed #4)	75.9 %	1.53	6.1	2.5
AiRLoc (average)	73.4 %	1.51	6.1	2.5
AiRLoc (no residual)	68.5 %	1.49	6.3	2.2
AiRLoc (no residual, other seed #1)	68.6 %	1.52	6.3	2.2
AiRLoc (no residual, other seed #2)	69.5 %	1.52	6.3	2.2
AiRLoc (no residual, other seed #3)	68.2 %	1.60	6.4	2.2
AiRLoc (no residual, other seed #4)	67.2 %	1.57	6.4	2.2
AiRLoc (no residual, average)	68.4 %	1.54	6.3	2.2
AiRLoc (no prior)	65.9 %	1.56	6.5	2.4
AiRLoc (no prior, other seed #1)	64.8 %	1.56	6.7	2.4
AiRLoc (no prior, other seed #2)	66.6 %	1.56	6.5	2.5
AiRLoc (no prior, other seed #3)	66.6 %	1.50	6.4	2.3
AiRLoc (no prior, other seed #4)	64.9 %	1.50	6.6	2.4
AiRLoc (no prior, average)	65.8 %	1.54	6.5	2.4
Priv local	67.0 %	1.54	6.3	2.3

when the player moves into the goal location (just as for AiRLoc and the other baselines). Moreover, different to the other approaches, the human participants have a limited time to complete each game (60 seconds). Such a time limit was used for the convenience of the participants – we wanted to avoid that the participants felt like they had to spend several minutes per action to squeeze out the maximum possible performance. The 60 second time limit was assessed to be more than sufficient for completing each game, and the participants agreed with this.

The age span of the 19 people who participated is between 14 and 42 years, with an average of 26.4 years and a median of 25 years. There were 13 men and 6 women (68% and 32%, respectively). For each human operator, 12 unique search areas from the validation set of *Massachusetts Buildings* were used, as well as a few sample search areas for the player to get acquainted with the controls of the game – the participants were able to practice as long as they desired, and no statistics were tracked during this warm up phase. The exact games provided span a subset of the games that AiRLoc and the other baselines are evaluated on, to ensure that the comparison is as fair as possible. However, each human is not tested on the entire dataset since it is impractically large, and hence there is a higher uncertainty in the human performance evaluation. The difficulty settings were split equally over these twelve games, with three games per difficulty (here difficulty is the distance between the start and goal patches, ranging from 1 to 4 steps away).
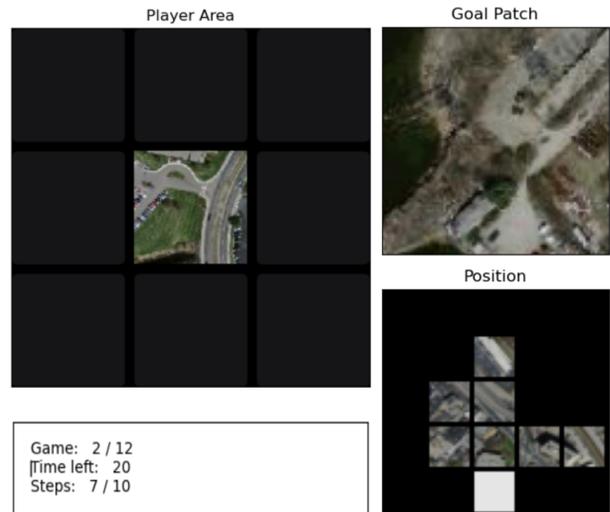

Fig. 4. An example of the human performance evaluation setup. Each participant was given a set of 12 different such games (a game is a search area and an associated start and goal location), and there was no overlap in the games played by different participants. Each search area was of size 5×5 and the movement budget was $T = 10$.

Even though the human setup is very similar to that of AiRLoc, there are some concepts that do not translate well to a human controlled setup. First, the positional encoding of AiRLoc is difficult to translate to human visual processing, and instead a map of the positions was implemented (thus the participants receive explicit information from past locations,

different from AiRLoc). Second, the human participants have not trained on the task like AiRLoc, and their visual systems are likely not tailored towards handling the quite low resolution patches. On the other hand, humans have implicitly conducted a lifetime worth of generic visual pretraining, which AiRLoc has not. These discrepancies, in conjunction with the limited number of human controlled trajectories, somewhat limit the reliability of the human baseline. Nonetheless, it is still a useful indication of the human performance on our proposed task.

REFERENCES

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[2] Kumar Ayush, Burak Uzkent, Kumar Tanmay, Marshall Burke, David Lobell, and Stefano Ermon. Efficient poverty mapping using deep reinforcement learning. *arXiv preprint arXiv:2006.04224*, 2020.

[3] Luca Bartolomei, Lucas Teixeira, and Margarita Chli. Perception-aware path planning for uavs using semantic segmentation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[4] Adrian Boguszewski, Dominik Batorski, Natalia Ziemba-Jankowska, Tomasz Dziedzic, and Anna Zambrzycka. Landcover.ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery, 2020.

[5] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2488–2496, 2015.

[6] Tung Dang, Christos Papachristos, and Kostas Alexis. Autonomous exploration and simultaneous object search using aerial robots. In *2018 IEEE Aerospace Conference*, pages 1–7. IEEE, 2018.

[7] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction, 2015.

[8] Mingfei Gao, Ruichi Yu, Ang Li, Vlad I Morariu, and Larry S Davis. Dynamic zoom-in network for fast object detection in large images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6926–6935, 2018.

[9] Ritwik Gupta, Bryce Goodman, Nirav Patel, Ricky Hosfelt, Sandra Sajeev, Eric Heim, Jigar Doshi, Keane Lucas, Howie Choset, and Matthew Gaston. Creating xbd: A dataset for assessing building damage from satellite imagery. In *CVPR workshops*, 2019.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[12] Danil Kuzin, Olga Isupova, Brooke D Simmons, and Steven Reece. Disaster mapping from satellites: damage detection with crowdsourced point labels. *arXiv preprint arXiv:2111.03693*, 2021.

[13] Ajith Anil Meera, Marija Popović, Alexander Millane, and Roland Siegwart. Obstacle-aware adaptive informative path planning for uav-based target search. In *ICRA*, 2019.

[14] Lina Mezghani, Sainbayar Sukhbaatar, Thibaut Lavril, Oleksandr Maksymets, Dhruv Batra, Piotr Bojanowski, and Karteek Alahari. Memory-Augmented Reinforcement Learning for Image-Goal Navigation. working paper or preprint, March 2022.

[15] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013.

[16] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *NeurIPS*, 2014.

[17] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. *arXiv preprint arXiv:2203.03580*, 2022.

[18] Aleksis Pirinen and Cristian Sminchisescu. Deep reinforcement learning of region proposal networks for object detection. In *CVPR*, 2018.

[19] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Jen Jen Chung, Inkyu Sa, Roland Siegwart, and Juan Nieto. An informative path planning framework for uav-based terrain monitoring. *Autonomous Robots*, 44(6):889–911, 2020.

[20] Shraman Pramanick, Ewa M Nowara, Joshua Gleason, Carlos D Castillo, and Rama Chellappa. Where in the world is this image? transformer-based geo-localization in the wild. *arXiv preprint arXiv:2204.13861*, 2022.

[21] Samrudhdhi B Rangrej and James J Clark. A probabilistic hard attention model for sequentially observed scenes. *arXiv preprint arXiv:2111.07534*, 2021.

[22] Samrudhdhi B Rangrej, Chetan L Srinidhi, and James J Clark. Consistency driven sequential transformers attention model for partially observable scenes. *arXiv preprint arXiv:2204.00656*, 2022.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[24] Seyed Abbas Sadat, Jens Wawerla, and Richard Vaughan. Fractal trajectories for online non-uniform aerial coverage. In *ICRA*, 2015.

[25] Alexander Sax, Bradley Emi, Amir R Zamir, Leonidas Guibas, Silvio Savarese, and Jitendra Malik. Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. *arXiv preprint arXiv:1812.11971*, 2018.

[26] Michael Schmitt, Pedram Ghamisi, Naoto Yokoya, and Ronny Hänsch. Eod: The ieee grss earth observation database. In *IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium*, pages 5365–5368. IEEE, 2022.

[27] Felix Stache, Jonas Westheider, Federico Magistri, Cyrill Stachniss, and Marija Popović. Adaptive path planning for uavs for multi-resolution semantic segmentation. *arXiv preprint arXiv:2203.01642*, 2022.

[28] Humans In the Loop. Semantic segmentation of aerial imagery.

[29] Burak Uzkent and Stefano Ermon. Learning when and where to zoom with deep reinforcement learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12345–12354, 2020.

[30] Andrea Vallone, Frederik Warburg, Hans Hansen, Søren Hauberg, and Javier Civera. Danish airs and grounds: A dataset for aerial-to-street-level place recognition and localization. *CoRR*, abs/2202.01821, 2022.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[32] Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *arXiv preprint arXiv:2202.10324*, 2022.

[33] Tingyu Wang, Zhedong Zheng, Yaoqi Sun, Tat-Seng Chua, Yi Yang, and Chenggang Yan. Multiple-environment self-adaptive network for aerial-view geo-localization. *arXiv preprint arXiv:2204.08381*, 2022.

[34] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[35] Daniel Wilson, Xiaohan Zhang, Waqas Sultani, and Safwan Wshah. Visual and object geo-localization: A comprehensive survey. *arXiv preprint arXiv:2112.15202*, 2021.

[36] Junshi Xia, Naoto Yokoya, Bruno Adriano, and Clifford Broni-Bediako. Openearthmap: A benchmark dataset for global high-resolution land cover mapping. *arXiv preprint arXiv:2210.10732*, 2022.

[37] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[38] Zhitong Xiong, Fahong Zhang, Yi Wang, Yilei Shi, and Xiao Xiang Zhu. Earthnets: Empowering ai in earth observation. *arXiv preprint arXiv:2210.04936*, 2022.

[39] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. *arXiv preprint arXiv:2204.13226*, 2022.

[40] Zelong Zeng, Zheng Wang, Fan Yang, and Shin'ichi Satoh. Geo-localization via ground-to-satellite cross-view image retrieval. *IEEE Transactions on Multimedia*, pages 1–1, 2022.

[41] Leyang Zhao, Li Yan, Xiao Hu, Jinbiao Yuan, and Zhenbao Liu. Efficient and high path quality autonomous exploration and trajectory planning of uav in an unknown environment. *ISPRS International Journal of Geo-Information*, 10(10):631, 2021.

[42] Runzhe Zhu. Sues-200: A multi-height multi-scene cross-view image benchmark across drone and satellite, 2022.

[43] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017.

Fig. 5. Successful examples of AiRLoc (left) and *Priv local* (right) on a flooding scenario in *xBD-disaster* (7×7 setup, movement budget $T = 14$). The start and goal locations are denoted 'S' and 'G', respectively. The numbered circles show which locations are visited and in what order. Recall that the full underlying search area is never observed in its entirety, i.e. the agents must operate based on partial glimpses alone. Also note that AiRLoc was only trained on search areas of size 5×5 and movement budget $T = 10$. AiRLoc takes the same first two steps as *Priv local*, then deviates and reaches the goal in fewer steps than *Priv local*.
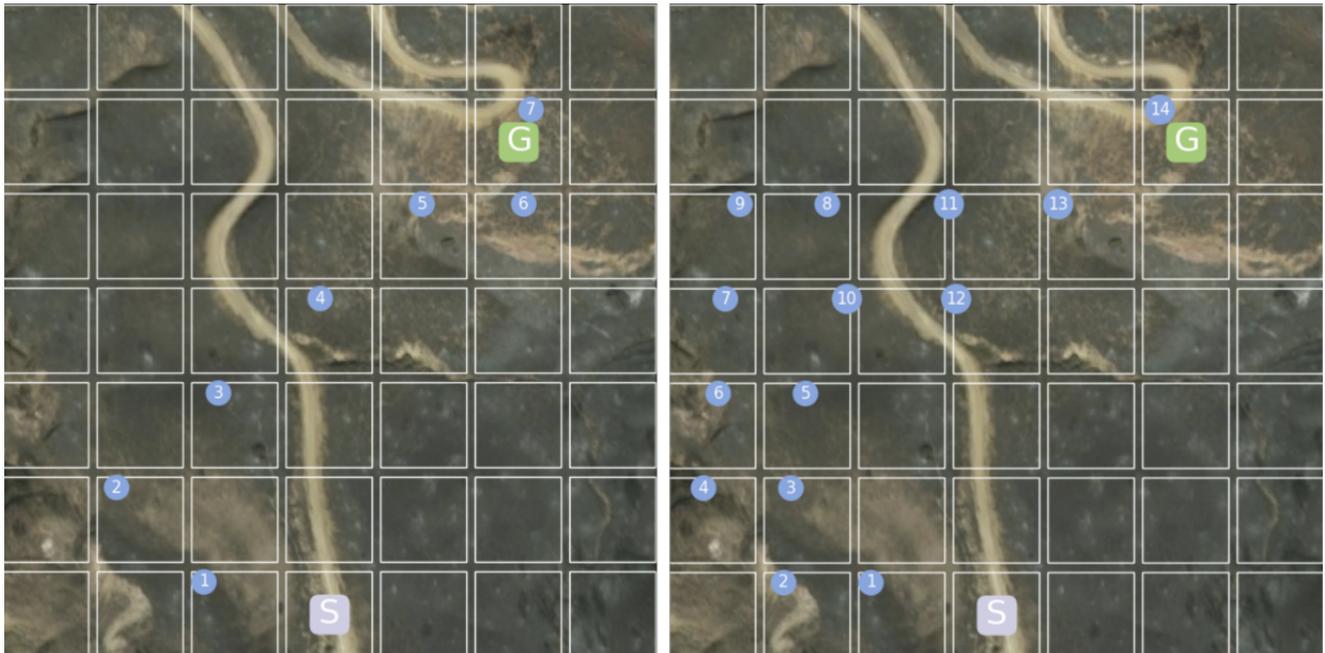

Fig. 6. Successful examples of AiRLoc (left) and *Priv local* (right) on a post-wildfire scenario in *xBD-disaster* (7×7 setup, movement budget $T = 14$). AiRLoc takes the same first step as *Priv local*, then deviates, and reaches the goal twice as fast. *Priv local* precisely manages to reach the goal within its movement budget.