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 confined area to precisely localize the people. Due to time and battery constraints it is often critical that localization is performed efficiently. We abstract this type of problem 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 must localize a goal that is described through visual cues. To further mimic the situation on a UAV, the agent cannot observe the search area in its entirety, not even at low resolution, so it must operate based on partial glimpses alone. To tackle this task, we propose *AiRLoc*, a reinforcement learning (RL) model that decouples exploration (searching for distant goals) and exploitation (localizing nearby goals). Extensive evaluations show that AiRLoc outperforms various baselines as well as humans, and that it generalizes across datasets, e.g. to disaster-hit areas without seeing a single disaster scenario during training. Code and models are available at https://github.com/aleksispi/airloc.

1 INTRODUCTION

We propose a novel setup that allows for controllable and reproducible development of and experimentation with systems for UAV-based SAR operations.[1] In this framework, an agent operates on top of an aerial image (proxy for a 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 visual cues are given as a top-view observation of the goal in 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). To enable this, our approach can be modified to allow for more flexible goal specifications, e.g. by integrating an off-the-shelf geo-localization module (Wilson et al., 2021; Zeng et al., 2022).

There are many cases where GPS coordinates of the goal are not available or reliable (e.g. satellite navigation systems are susceptible to radio frequency interruptions). 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. To further mimic the situation on a 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 this is often impractical, e.g. if the battery of the UAV is running low. Adverse weather conditions could also make it risky to operate at a high altitude. To tackle our task, we propose *AiRLoc*, an RL-based model that decouples exploration (searching for distant goals) and exploitation (localizing nearby goals) – see Fig. 1. Extensive experimental results show

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

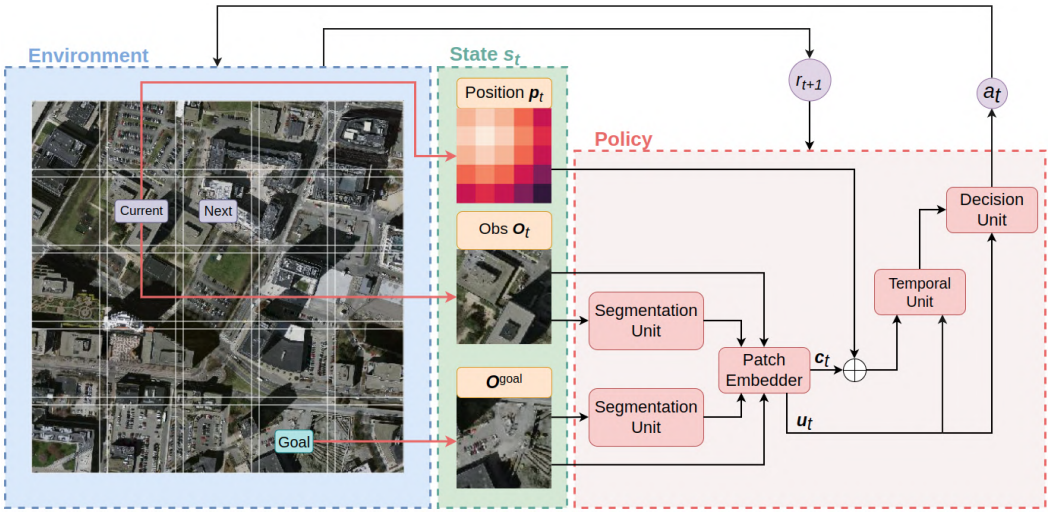

Figure 1: Overview of *AiRLoc*, our RL-based agent for aerial view goal localization. Note that AiRLoc only observes a glimpse of the full search area in each step. See § 2 for details.

that AiRLoc outperforms heuristic search methods and alternative learnable approaches. We also conduct a proof-of-concept study which indicates that this task is difficult even for humans.

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 (Rangrej & Clark, 2021; Rangrej et al., 2022). In contrast, most earlier RL-based methods that have been proposed for computer vision tasks – e.g. for object detection (Caicedo & Lazebnik, 2015; Gao et al., 2018; Pirinen & Sminchisescu, 2018) and aerial view processing (Uzkent & Ermon, 2020; Ayush et al., 2020) – assume access to at least a low-resolution version of the entire scene or image being processed.

2 Aerial View Goal Localization

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. Every grid cell in the search area corresponds to a valid position $\boldsymbol{p}_t$. The agent can only directly observe the image content $\boldsymbol{O}_t$ of its current cell. In each episode, one of the grid cells corresponds to the goal location. The image content of the goal cell is denoted $\boldsymbol{O}^{\text{goal}}$ and its position is denoted $\boldsymbol{p}^{\text{goal}}$. Note that the goal position $\boldsymbol{p}^{\text{goal}}$ is *never* observed by the agent; it is only used to determine if the agent is successful, which it is as soon as the agent's current position $\boldsymbol{p}_t$ and the goal position $\boldsymbol{p}^{\text{goal}}$ coincide ($\boldsymbol{p}_t = \boldsymbol{p}^{\text{goal}}$).

In each episode, the agent's start location $\boldsymbol{p}_0$ and goal location $\boldsymbol{p}^{\text{goal}}$ are randomly sampled within the search area ($\boldsymbol{p}_0 \neq \boldsymbol{p}^{\text{goal}}$). The agent then moves around until it either reaches the goal, or a maximum number of steps T have been taken. This limit T represents 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. 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 $\boldsymbol{p}_t$).

AiRLoc agent. The state s_t contains the currently observed patch $\boldsymbol{O}_t$, the goal patch $\boldsymbol{O}^{\text{goal}}$, and an encoding $\boldsymbol{p}_t \in \mathbb{R}^{256}$ of the agent's position. As described above, AiRLoc has eight possible actions a_t, which correspond to moving to any of its eight 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 $r_t = 3 \cdot \mathbb{1}\left(\boldsymbol{p}_t = \boldsymbol{p}^{\text{goal}}\right) - 1$ is provided ($\mathbb{1}$ is the indicator function).

Policy overview: In each step, the state s_t is processed by four modules to generate the current action distribution $\pi_{\boldsymbol{\theta}}(*|s_t)$, where $\boldsymbol{\theta}$ denotes all learnable parameters. First, $\boldsymbol{O}_t$ and $\boldsymbol{O}^{\text{goal}}$

Table 1: Results on the test sets of *Masa* (col 2-3), *Dubai* (col 4-5) and *xBD-disaster* (col 6-7; col 8-9). Movement budget $T = 10$, $T = 14$ for setups of sizes 5×5, 7×7. Results in col 2-7 correspond to models trained on *Masa*-train. For col 8-9, models were trained on *xBD-pre* at different geographical locations than those in *xBD-disaster*. AiRLoc outperforms the baselines across all datasets and search area sizes, and localizes goals using fewer steps on average.

Agent type	**Success**	**Steps**	**Success**	**Steps**	**Success**	**Steps**	**Success**	**Steps**
AiRLoc (5x5)	67.6 %	6.2	68.8 %	6.3	66.1 %	6.5	72.8 %	6.1
Priv local (5x5)	64.2 %	6.5	65.6 %	6.5	63.8 %	6.7	67.3 %	6.4
Priv rand (5x5)	41.0 %	8.0	41.0 %	8.0	40.8%	7.9	40.8%	7.9
AiRLoc (7x7)	59.0 %	9.4	57.2 %	9.7	50.7 %	10.2	55.7 %	9.9
Priv local (7x7)	56.3 %	9.9	53.7 %	10.2	50.5 %	10.2	53.6 %	10.0
Priv rand (7x7)	25.2 %	12.3	26.9 %	12.0	25.5 %	12.2	25.5 %	12.2

are passed through a pretrained *segmentation unit* (a U-net) which predicts building segmentation masks for $\boldsymbol{O}_t$ and $\boldsymbol{O}^{\text{goal}}$. Second, $\boldsymbol{O}_t$ and $\boldsymbol{O}^{\text{goal}}$ and their segmentations are passed through a *patch embedder* which yields a low-dimensional embedding $\boldsymbol{c}_t \in \mathbb{R}^{256}$ of what the agent observes and what it aims to localize. The patch embedder also outputs an exploitation prior $\boldsymbol{u}_t \in \mathbb{R}^8$ (see below). Third, $\boldsymbol{p}_t$ is added to $\boldsymbol{c}_t$ and the result and $\boldsymbol{u}_t$ are passed to an LSTM-based *temporal unit* which integrates information over time. Finally, the LSTM output and $\boldsymbol{u}_t$ are passed to a *decision unit* which yields the probability distribution $\pi_{\boldsymbol{\theta}}(*|s_t)$. This decision unit first projects the LSTM's output into the action space dimensionality, then adds the exploitation prior $\boldsymbol{u}_t$, 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: To extract relevant information about the relationship between $\boldsymbol{O}_t$ and $\boldsymbol{O}^{\text{goal}}$, we use an architecture similar to that by Doersch et al. (2015), 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. We pretrain the patch embedder in the same self-supervised fashion as Doersch et al. (2015). During pretraining, another dense layer (with input $\boldsymbol{c}_t$) is attached to produce an 8-dimensional output $\boldsymbol{u}_t$, which is fed to a softmax function. The eight outputs correspond to the possible locations of $\boldsymbol{O}^{\text{goal}}$ relative to $\boldsymbol{O}_t$, assuming these are adjacent. When using the patch embedder within AiRLoc, we take advantage of both $\boldsymbol{c}_t$ and $\boldsymbol{u}_t$, cf. Fig. 1. Note that $\boldsymbol{u}_t$ can be interpreted as an *exploitation prior*, as it is specifically tuned towards localizing ('exploiting') adjacent goals. Thus, feeding $\boldsymbol{u}_t$ to the temporal unit as well as directly to the decision unit allows AiRLoc to learn when to explore and when to exploit.

Positional encoding: Positional information is represented similarly to Transformers (Vaswani et al., 2017). 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.

3 Experiments

Baselines. ***Priv random*** selects actions randomly, with two exceptions: i) it cannot move outside the search area; ii) it avoids previous locations. ***Priv local*** selects actions by repeatedly calling the pretrained patch embedder (which assumes the goal is adjacent to the current location), subject to the privileged movement restrictions of *Priv random*. Finally, ***Human*** represents the average human performance from a proof-of-concept evaluation with 19 subjects (details in the supplement).

Datasets. We mainly use *Massachusetts Buildings* (*Masa*) by Mnih (2013) for development and evaluation. The data contains images of Boston and the surrounding suburban and forested areas. Models are also evaluated on the *Dubai* dataset (the Loop), which depicts urban regions surrounded by dry deserts. Finally, we also train and evaluate on the *xBD* dataset by Gupta et al. (2019) which contains aerial images from various regions both before (*xBD-pre*) and after (*xBD-disaster*) various natural distastes (e.g. wildfires).

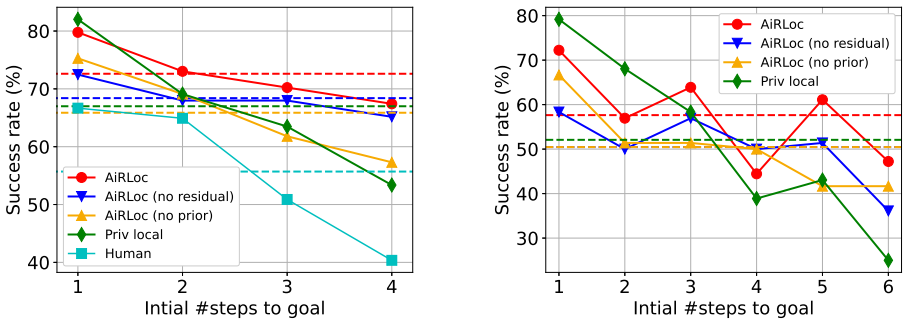

Figure 2: Success rate versus start-to-goal distance on the validation set of *Masa* (averages are dashed). Left: $M \times N = 5 \times 5$, $T = 10$. The methods are generally more successful when the start is closer to the goal. AiRLoc performs roughly on par with *Priv local* when the goal and start are adjacent and outperforms it at larger distances. AiRLoc is also more successful than its ablated variants in all settings and on average. Right: $M \times N = 7 \times 7$, $T = 14$. 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.

Main results. From Table 1 we see that 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), as well as across different spatial regions and circumstances. Particularly relevant from a SAR-perspective in disaster-hit areas, columns 6-9 contain results on *xBD-disaster*. Column 6-7 show that AiRLoc generalizes quite well from having been trained on an entirely different dataset (*Masa*) containing satellite images of non-disaster-hit urban areas to disaster-hit areas at various other spatial locations. Results are however improved further (col 8-9) if models are first trained on non-disaster-hit images from the same dataset (*xBD-pre*) and then evaluated at different locations depicting disaster-hit scenarios. The runtimes per action are 20 ms, 13 ms and 6 ms, respectively, for AiRLoc, *Priv local* and *Priv random* (thus negligible compared to the movement overhead of an actual UAV). Please see the supplementary material for visualizations of agent behaviors in disaster zones.

AiRLoc variants. Fig. 2 shows ablation results[2] for AiRLoc. ***No residual*** omits the prior $\boldsymbol{u}_t$ in the decision unit (but not in the temporal unit). ***No prior*** entirely omits $\boldsymbol{u}_t$ in the architecture.

Human performance evaluation. The results of the proof-of-concept human performance evaluation in Fig. 2 (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.

4 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 be useful for automated search-and-rescue operations, e.g. in regions that are heavily affected by climate-induced disasters. 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 several challenges of real use-cases are not fully reflected in the task. 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. Extensive experimental evaluations clearly showed the benefits of our AiRLoc agent over the learnable and heuristic baselines. Code and models have been made publicly available, so that others can further explore and extend our proposed task towards real use-cases within disaster relief and management.

[2]See the supplementary material for more ablation results and a seed sensitivity analysis.

REFERENCES

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.

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

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction, 2015. URL https://arxiv.org/abs/1505.05192.

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*, pp. 6926–6935, 2018.

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 10–17, 2019.

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

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

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.

Aleksis Pirinen and Cristian Sminchisescu. Deep reinforcement learning of region proposal networks for object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6945–6954, 2018.

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

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.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

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.

Humans In the Loop. Semantic segmentation of aerial imagery. URL https://www.kaggle.com/datasets/humansintheloop/semantic-segmentation-of-aerial-imagery.

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*, pp. 12345–12354, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

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.

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

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

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

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.

Zelong Zeng, Zheng Wang, Fan Yang, and Shin'ichi Satoh. Geo-localization via ground-to-satellite cross-view image retrieval. *IEEE Transactions on Multimedia*, pp. 1–1, 2022. doi: 10.1109/TMM.2022.3144066.

A SUPPLEMENTARY MATERIAL

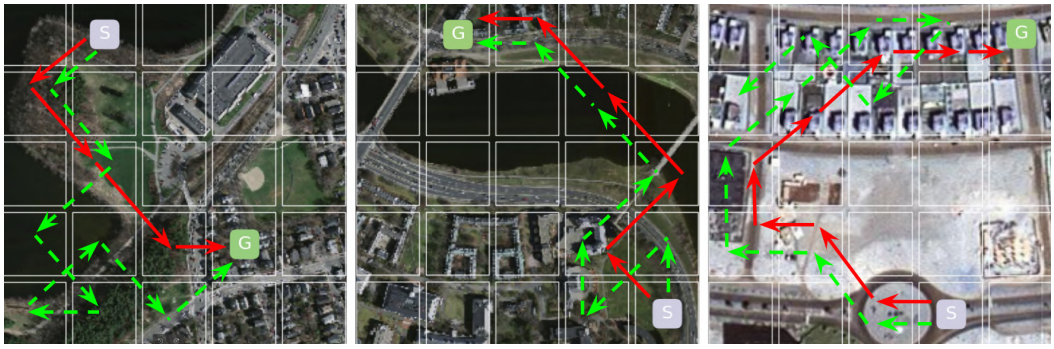

Figure 3: Examples of AiRLoc (red) and *Priv local* (dashed green) on the test set of *Massachusetts Buildings* (left, middle) and *Dubai* (right). Models were trained on the training set of *Massachusetts Buildings*. Search area size 5×5, movement budget $T = 10$. Recall that the full underlying search area is never observed in its entirety (they are shown here for visualization purposes only), i.e. the agents must operate based on partial glimpses alone. Left: AiRLoc takes the same action as *Priv local* in the first two steps 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.

Visualizations of agent trajectories. In Fig. 3 we show qualitative examples of AiRLoc and the best alternative learnable approach *Priv local* on the datasets *Massachusetts Buildings* and *Dubai*. In this case the models were trained on the training set of *Massachusetts Buildings*. Fig. 4 - 15 show several visualizations of AiRLoc and *Priv local* on disaster-hit search areas from the dataset *xBD-disaster*. In this case the models were trained on non-disaster-hit data from *xBD-pre*, where we have ensured that this training data depicts other geographical areas than those in *xBD-disaster*. See more detailed information about each dataset further down in this supplement.

When inspecting these visual examples, keep in mind the connection between AiRLoc and *Priv local*, where *Priv local* is essentially an 'exploit only' model that is optimized to localize adjacent goals. The action distribution of *Priv local* is obtained[3] by feeding its final output $\boldsymbol{u}_t \in \mathbb{R}^8$ through a softmax. Our full AiRLoc agent takes advantage of this exploitation prior $\boldsymbol{u}_t$ and decouples exploration from exploitation, as explained in the main paper. AiRLoc thus decides when to resort to *Priv local*'s exploitative behavior (although without the privileges) and when to explore independently.

[3]Subject to privileged movement constraints, without which it performs abysmally (see Table 3).

Figure 4: Successful examples of AiRLoc (left) and *Priv local* (right) on a flooding scenario in *xBD-disaster* (5×5 setup, movement budget $T = 10$). 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 (they are shown here for visualization purposes only), i.e. the agents must operate based on partial glimpses alone. AiRLoc takes a different and slightly shorter path towards the goal location.

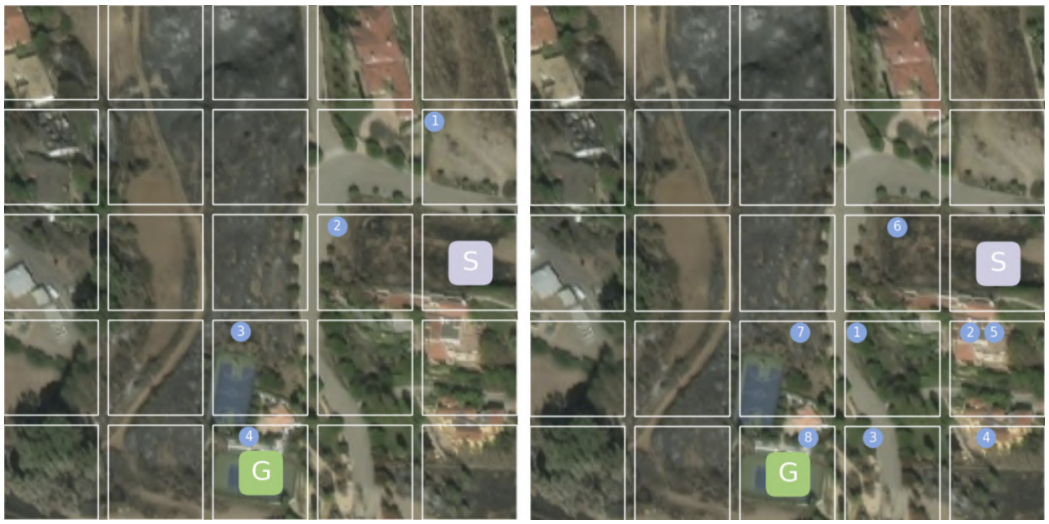

Figure 5: Successful examples of AiRLoc (left) and *Priv local* (right) on a post-wildfire scenario in *xBD-disaster* (5×5 setup, movement budget $T = 10$). AiRLoc takes a different and significantly shorter path towards the goal location. Note that *Priv local* visits the location below the start location after 2 and 5 steps, despite its privileged movement constraints which tries to avoid previous locations. However, in this example, after the 4th step there are no unvisited locations to move to, and so it has to move somewhere.

Figure 6: A Successful example of AiRLoc (left) and an unsuccessful example of *Priv local* (right) on a flooding scenario in *xBD-disaster* (5×5 setup, movement budget $T = 10$). AiRLoc's location #7 shares forest-structure with the goal location, which may have been an important visual cue in the last step.

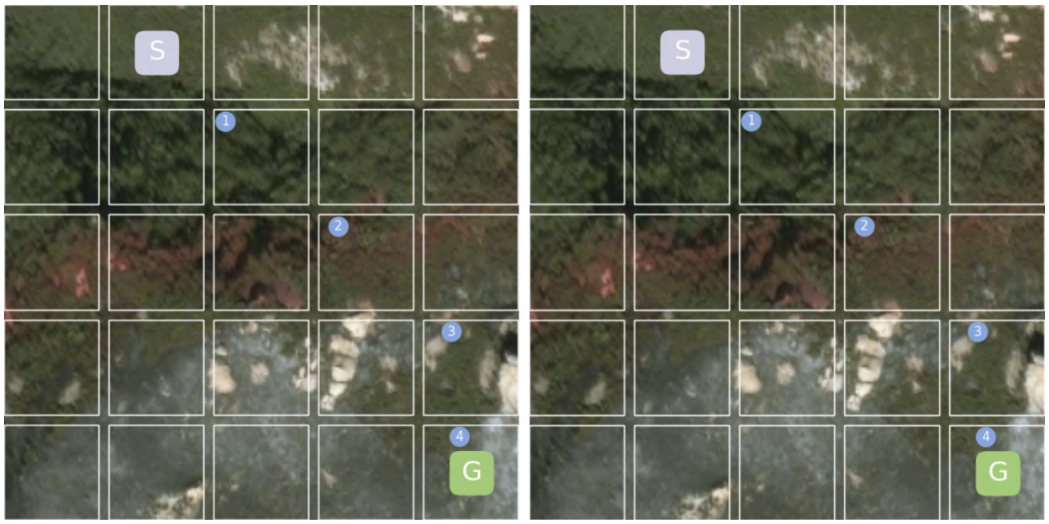

Figure 7: Successful examples of AiRLoc (left) and *Priv local* (right) on a wildfire scenario in *xBD-disaster* (5×5 setup, movement budget $T = 10$). Both agents take the exact same (and shortest) path towards the goal, i.e. AiRLoc fully resorts to the exploitation prior in this case.

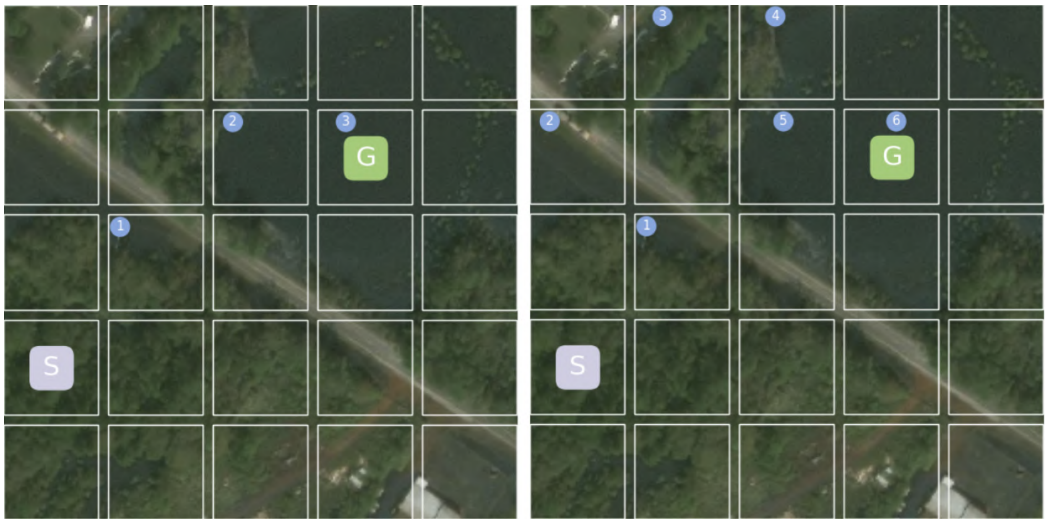

Figure 8: Successful examples of AiRLoc (left) and *Priv local* (right) on a flooding scenario in *xBD-disaster* (5×5 setup, movement budget $T = 10$). AiRLoc takes the first same step as *Priv local*, then deviates and takes a shortest path towards the goal. *Priv local* reaches the goal using several more steps.

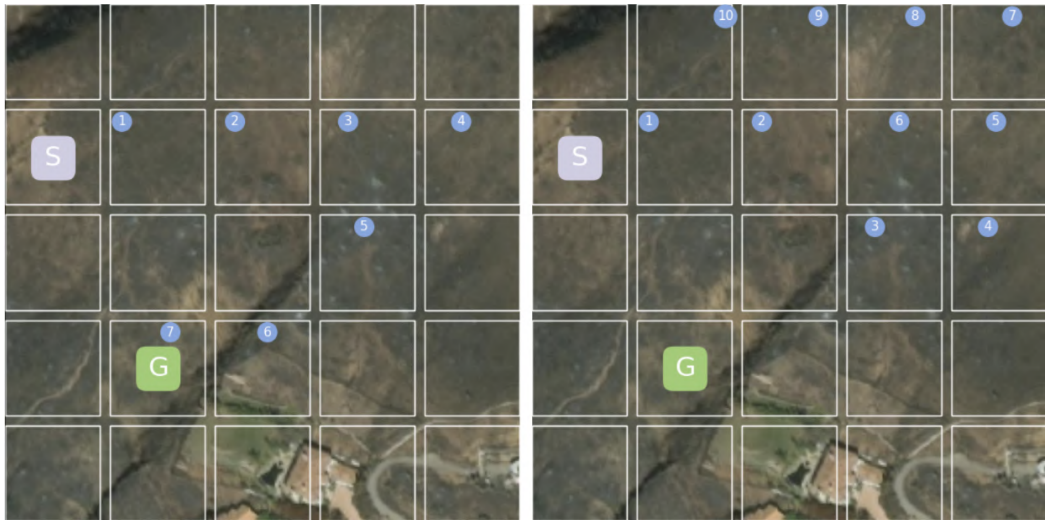

Figure 9: A successful example of AiRLoc (left) and an unsuccessful example of *Priv local* (right) on a post-wildfire scenario in *xBD-disaster* (5×5 setup, movement budget $T = 10$). AiRLoc moves in the same way as *Priv local* for the first two steps and then deviates. Note that AiRLoc does not take the shortest path towards the goal but nonetheless reaches it well within the movement budget.

Figure 10: An unsuccessful example of AiRLoc (left) and a successful example of *Priv local* (right) on *xBD-disaster* (5×5 setup, movement budget $T = 10$). AiRLoc takes the same path as *Priv local* for the first six steps and then deviates (it is adjacent to the goal when the budget $T = 10$ is exhausted). *Priv local* precisely manages to reach the goal within the budget.

Figure 11: Successful examples of AiRLoc (left) and *Priv local* (right) on a flooding scenario in *xBD-disaster* (5×5 setup, movement budget $T = 10$). AiRLoc takes a different and much shorter path towards the goal location.

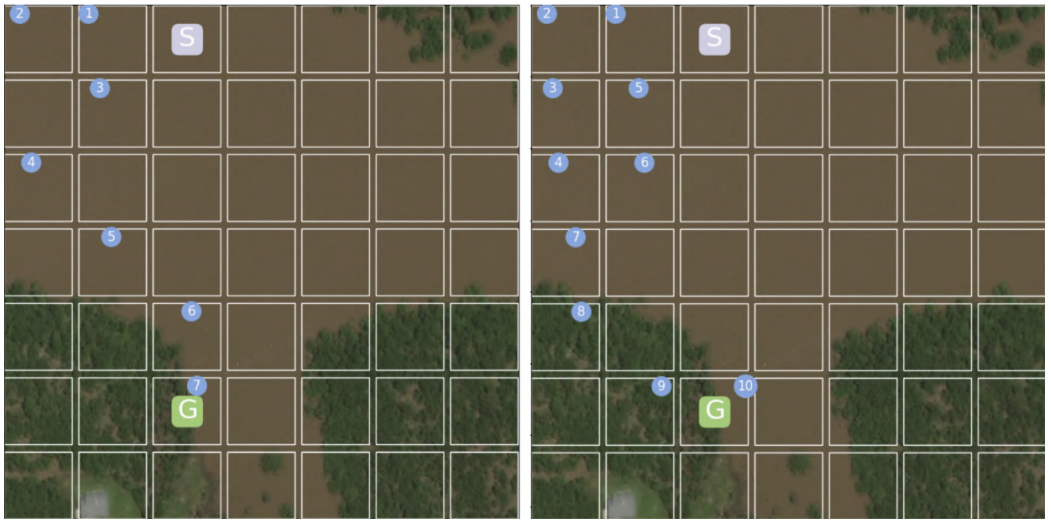

Figure 12: Successful examples of AiRLoc (left) and *Priv local* (right) on a flooding scenario in *xBD-disaster* (7×7 setup, movement budget $T = 14$). Note that AiRLoc was only trained on search areas of size 5×5 and movement budget $T = 10$. AiRLoc takes the same first two steps as *Priv local*, then deviates and reaches the goal in fewer steps than *Priv local*.

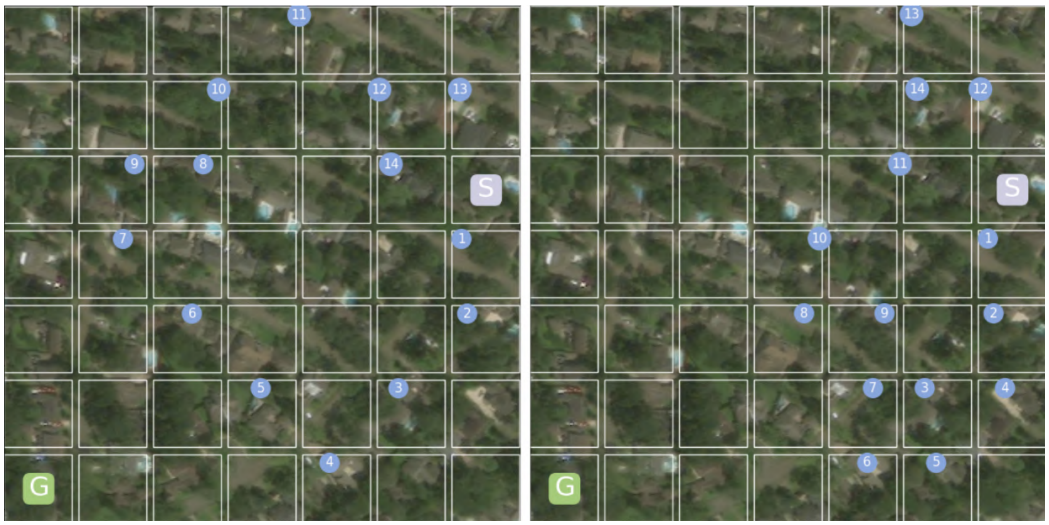

Figure 13: Unsuccessful examples of AiRLoc (left) and *Priv local* (right) on a flooding scenario in *xBD-disaster* (7×7 setup, movement budget $T = 14$). AiRLoc takes the same first two steps as *Priv local*, then deviates, but (like *Priv local*) fails to reach the goal.

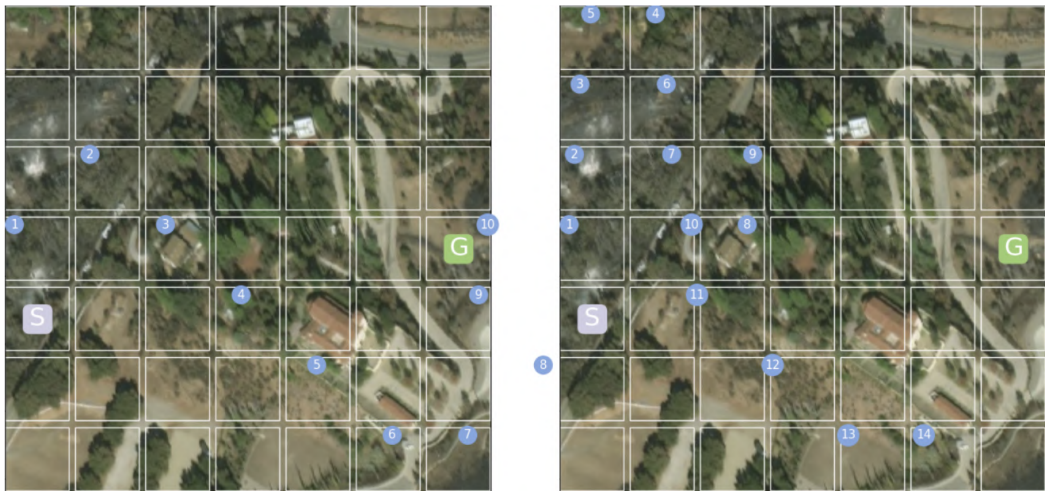

Figure 14: A successful examples of AiRLoc (left) and an unsuccessful example of *Priv local* (right) on *xBD-disaster* (7×7 setup, movement budget $T = 14$). AiRLoc takes the same first step as *Priv local* and then deviates. Note that AiRLoc even moves outside the search are at one occasion (location #8), but still manages to reach the goal well within the movement budget.

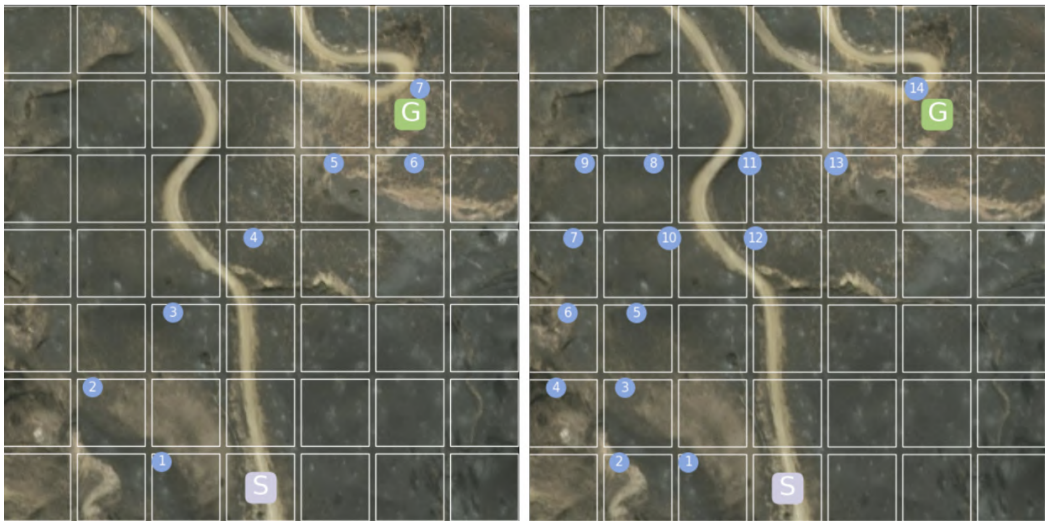

Figure 15: Successful examples of AiRLoc (left) and *Priv local* (right) on a post-wildfire scenario in *xBD-disaster* (7×7 setup, movement budget $T = 14$). AiRLoc takes the same first step as *Priv local*, then deviates, and reaches the goal twice as fast. *Priv local* precisely manages to reach the goal within its movement budget.

More details about AiRLoc. Positional information is represented in a similar way as in Transformers (Vaswani et al., 2017). Note that AiRLoc never receives global positional information, i.e. it is always relative to a given search area. Such information may be available e.g. during SAR within a confined area, where a UAV could easily 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 $\boldsymbol{p}_t \in \mathbb{R}^d$ (with d even; for us $d = 256$) is given by:

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

Policy training: To learn the parameters of AiRLoc, we first pretrain the patch embedder in a self-supervised fashion (without RL; see details below). We then freeze the patch embedder weights and train the rest of AiRLoc using REINFORCE (Williams, 1992) with Adam (Kingma & Ba, 2015), batch size 64, search area size $M \times N = 5 \times 5$, movement budget $T = 10$, learning rate 10^{-4}, and discount $\gamma = 0.9$. Training AiRLoc takes 30h on a Titan V100 GPU.[4] We apply left-right and top-down flipping of images (search areas) as data augmentation. We also 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 – see below for details.

More details about the patch embedder and segmentation unit: Our patch embedder architecture consists of two parallel branches with four convolutional layers (ReLUs and max pooling are applied between layers). First, $\boldsymbol{O}_t$ and $\boldsymbol{O}^{\text{goal}}$, with their segmentations channel-wise concatenated, are fed separately into one branch each (cf. Fig. 1). 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 $\boldsymbol{c}_t \in \mathbb{R}^{256}$.

Pretraining backbone vision components is common in RL setups, since it often yields a higher end performance (Sax et al., 2018; Parisi et al., 2022; Wang et al., 2022; Xiao et al., 2022; Yadav et al., 2022). Before training the rest of AiRLoc with reinforcement learning, the patch embedder is therefore pretrained (in the same self-supervised fashion as Doersch et al. (2015)) on the training set of *Massachusetts Buildings* using the categorical cross-entropy loss. This loss is computed using the 8-dimensional patch embedder prediction and a one hot encoding of the true goal direction relative to the start location (recall that during this pretraining stage, the start and goal are assumed to be adjacent). The Adam optimizer with batches of 256 pairs of image patches (start and goal) and a learning rate of 10^{-3} is used during this pretraining phase.

For the segmentation unit, we use and adapt the U-net model for biomedical segmentation applications (Ronneberger et al., 2015). A publicly available implementation of this U-net[5] is used as a starting point. However, since the patches (partial glimpses of the search area) are smaller than in the original U-net, the network structure is altered. This altered network consist of four downsampling convolutional blocks, which reduce the spatial dimensions of the input into a $3 \times 3 \times 64$ embedding. Then, four upsampling convolutional blocks are used to recreate the spatial dimension of the input patch (thus the segmentation unit outputs a binary $48 \times 48 \times 1$ building segmentation mask, although in general the segmentation unit could obviously include more classes as well). The segmentation network is pretrained on the training set of *Massachusetts Buildings* using a cross-entropy loss with Adam, batch size 128, and learning rate 10^{-4}, and is kept frozen when training the rest of AiRLoc.

Random seed sensitivity analysis. Table 2 shows the results of a policy network initilization

[4] All methods and baselines are implemented in, trained and evaluated using PyTorch.

[5] `https://github.com/milesial/Pytorch-UNet`

Table 2: 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 main paper and in the extended ablation study in Table 3. 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	Res. dist.
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

seed sensitivity analysis for AiRLoc and its ablated variants on the validation set of *Massachusetts Buildings*. The AiRLoc variants are trained[6] with five random network initializations each, and the results for the median-performing models on the validation set are reported in the main paper. The seed sensitivity is low overall. Even at worst, our full AiRLoc agent outperforms *Priv local*. Note that we report two additional metrics (aside from the success ratio and the average number of steps taken per episode). ***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).

Extended ablation study. See Table 3 for an extended ablation study of AiRLoc on the validation set of *Massachusetts Buildings*. In addition to the *no prior* and *no residual* variants, we also show results for the following AiRLoc variants. ***Priv*** applies the same heuristic as the privileged baselines (never moves outside, avoids visiting past locations). ***No sem seg*** omits the segmentation unit and uses only RGB patches in the patch embedder (which is pretrained with RGB-only inputs).

Dataset details. As described in the main paper, *Massachusetts Buildings* is used as the main dataset for model development and evaluation. There are 832 different search areas in training (70%), 178 in validation (15%), and 178 in testing (15%). Since top-right and left-right flipping of search areas is performed during training, and since search a area of size $M \times N = 5 \times 5$ has $25 \cdot 24$ different start-goal configurations, there are in total $832 \cdot 4 \cdot 25 \cdot 24 \approx 2 \cdot 10^6$ unique training configurations. As the various agents are trained for roughly 50k batches each, and since each batch consists of 64 episodes, this amounts to $3.2 \cdot 10^6$ training episodes.

[6]Each model is trained until convergence on the validation set (typically happens within 50k batches).

Table 3: Extended ablation study on the validation set of *Massachusetts Buildings* (movement budget $T = 10$ and $T = 14$ for setups of sizes 5×5 and 7×7, respectively). Adding the movement constraint privileges of the random baseline and *Priv local* does not yield any significant improvements for AiRLoc - it even reduces the success rate for our full AiRLoc agent. Conversely, in the bottom of this table we report results for *Local*, which is the same as *Priv* local but without the privileged movement constraints (thus *Local* may visit the same location multiple times and move outside the search area). Different to AiRLoc, which also lacks any privileged movement constraints, *Local* performs abysmally when it not given such constraints. Mid-level vision capabilities (semantic segmentation) are crucial for AiRLoc's performance. The fact that the ablated AiRLoc variants generally result in a lower success rate motivates our design choices.

Agent type	Success	Step ratio	Steps	Res. dist.
AiRLoc (5x5)	72.6 %	1.49	6.0	2.4
AiRLoc (priv, 5x5)	68.8 %	1.47	6.2	2.3
AiRLoc (no residual, 5x5)	68.5 %	1.49	6.3	2.2
AiRLoc (no residual, priv, 5x5)	71.9 %	1.52	6.2	2.5
AiRLoc (no prior, 5x5)	65.9 %	1.56	6.5	2.4
AiRLoc (no prior, priv, 5x5)	67.1 %	1.56	6.4	2.6
AiRLoc (no sem seg, 5x5)	62.6 %	1.52	6.6	2.4
AiRLoc (no sem seg, priv, 5x5)	64.6 %	1.56	6.7	2.5
AiRLoc (no residual, no sem seg, 5x5)	61.1 %	1.56	6.8	2.4
AiRLoc (no residual, no sem seg, priv, 5x5)	62.6 %	1.59	6.8	2.5
AiRLoc (no prior, no sem seg, 5x5)	60.7 %	1.67	6.9	2.5
AiRLoc (no prior, no sem seg, priv, 5x5)	62.2 %	1.69	6.9	2.6
AiRLoc (7x7)	57.6 %	1.54	9.6	3.4
AiRLoc (priv, 7x7)	51.4 %	1.49	10.1	3.3
AiRLoc (no residual, 7x7)	50.5 %	1.54	10.1	3.4
AiRLoc (no residual, priv, 7x7)	52.3 %	1.54	9.9	3.5
AiRLoc (no prior, 7x7)	50.5 %	1.59	10.2	3.6
AiRLoc (no prior, priv, 7x7)	52.1 %	1.59	10.1	3.6
AiRLoc (no sem seg, 7x7)	48.4 %	1.56	10.4	3.3
AiRLoc (no sem seg, priv, 7x7)	47.7 %	1.69	10.7	3.2
AiRLoc (no residual, no sem seg, 7x7)	46.1 %	1.59	10.4	3.6
AiRLoc (no residual, no sem seg, priv, 7x7)	48.8 %	1.64	10.4	3.7
AiRLoc (no prior, no sem seg, 7x7)	42.4 %	1.79	11.1	3.4
AiRLoc (no prior, no sem seg, priv, 7x7)	44.4 %	1.82	11.0	3.4
Priv local	67.0 %	1.54	6.3	2.3
Local	19.9 %	0.79	8.5	6.1

During evaluation, one randomly generated but fixed configuration of each start-to-goal distance is used per search area, which results in 712 fixed validation and test configurations, respectively, in the 5×5 setting ($4 \cdot 178 = 172$). Similarly, when evaluating on the *Dubai* dataset (the Loop), there are 196 search areas and thus 784 fixed evaluation configurations. 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).

As for the *xBD-pre* and *xBD-disaster* data, they again depict data from non-disaster-hit (*xBD-pre*) and disaster-hit (*xBD-disaster*) areas,[7] and the respective data splits are from different geographical areas (thus there is no spatial overlap). There are 902 different search areas in training (*xBD-pre*), and since top-right and left-right flipping of search areas is performed during training, and since search a area of size $M \times N = 5 \times 5$ has $25 \cdot 24$ different start-goal configurations, there are in total $902 \cdot 4 \cdot 25 \cdot 24 \approx 2.2 \cdot 10^6$ unique training configurations. During evaluation (on *xBD-disaster*), one randomly generated but fixed configuration of each start-to-goal distance is used per search area, which results in 5212 evaluation configurations in the 5×5 setting (there are 1303 search areas in

[7]More specifically, *xBD-pre* contains the satellite image subset depicting various locations prior to hurricane Michael (found in the `tier1` subset of the *xBD* dataset), and *xBD-disaster* contains the satellite image subset depicting various locations after various natural disasters (also found in the `tier1` subset of the *xBD* dataset).

Figure 16: 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 $M \times N = 5 \times 5$ and the movement budget was $T = 10$.

xBD-disaster and $4 \cdot 1303 = 5212$).

Description of 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. 16. 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. 16. The game ends either when the movement budget $T = 10$ is exhausted or when the player moves into the goal location.

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 validation set 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).

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. Third, 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. 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.

Acknowledgments: This work was partially supported by the strategic research projects ELLIIT and the Swedish Foundation for Strategic Research project, Semantic Mapping and Visual Navigation for Smart Robots (grant no. RIT15-0038).