7 Supplementary Materials

7.1 Code

Anonymized code link: S.T.A.R. Search

7.2 Real Hardware Performance

While it is true that the STAR algorithm has been tested on the physical systems shown in the supplementary video, our aim with this paper was to properly ablate and assess the performance of the STAR algorithm against the state of the art from an algorithmic standpoint as physical runs cannot usually be conducted in quantities that show statistical significance. However, we can provide some statistics from running the algorithm on physical systems.

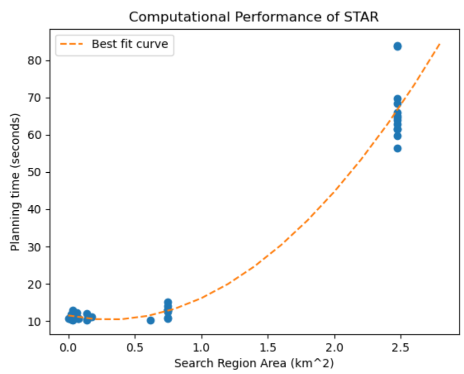

Figure 7: Planning time vs Search region Size

The Fig. 7 shows a plot of the planning time for one decision of the STAR algorithm against the size of the search space. The planning time in search regions under a sq km is around 10-15 secs. At 2.5 sq. km (search region size in the paper), it rises to over a minute. In practice the robot can just start planning its next decision slightly before it expects to arrive at its next goal location so this planning time doesn't impact search performance. This was an engineering detail which we didn't include because throughout the paper we evaluate the algorithms on their sample complexity rather than wall clock time to remain agnostic to such implementation details. The compute on the robot is a Nuvo-8108GC with Intel Xeon E-2278GEL (Coffee Lake R) 2.0 GHz Processor.

7.3 Terrain Visibility Prior

Since our use case is outdoor spaces we use Depth Elevation Maps (DEMs) to represent the terrain (See Fig. 3) since it is more memory efficient than voxels.

To determine what portion of the map is visible given location and direction of facing we use a reference plane-based approach [42] which can compute the viewable region in the map given any point in constant time as opposed to ray casting methods [43, 44, 45, 46, 47] which take variable time. We assume that the topography remains unchanged over the course of the run; however, our physical systems are capable of dynamically updating the topography using point clouds generated by stereo cameras. Hence, having a constant time algorithm for viewshed computation allows for efficient onboard updating if there are differences between the prior and the dynamic observations made by the robot on the ground.

Once we have the viewable region from a given point on the map we discretise it and apply viewability limits on in accordance with our physical system as shown in Fig. 4-left and described ahead.

7.4 UGV Sensing Action model

We use a array of 5MP RGB cameras with an effective lateral field-of-view (FOV) of 193° for the ground vehicles. This allows the perception system to pick up detections several hundred metres out.

We model the sensing action model in the grid representation as a trapezium of fifteen cells along the bearing of the UGV as shown in Fig. 2-a. Its full extent is upto $210m - 300m$ in front of the robot subject to occlusions. The motivation behind this is that beyond a certain distance even if the terrain is visible, it is not possible to make accurate detections of targets as they are just a few pixels in the image. Fig. 4 shows an example of the viewshed computed at two locations in the map shown earlier assuming a 360° FOV.

7.5 Target Sensing Action Model

Fig. 2-b shows a representative example of the viewshed of the targets. Since we don't have information on the direction of facing of the targets, we model the FOV such that targets see in all directions subject to the topography and the $210m - 300m$ viewing limit but without depth aware noise.

7.6 Visibility Risk Aware Path Planning

Since our robot and target viewing models are symmetric, it implies that detecting a target is accompanied by the target detecting the search agent, however being identified once does not mean the task is over, there could be more targets to locate and known targets should be avoided for the remainder of the search. We expect to minimize the stealth penalty over the course of the run but don't expect it to be zero. As an aside, were we to employ asymmetric viewing models such that viewing targets without being viewed was possible, we might aim to have zero risk policies but we outline this for future work.

In order for the search agents to respect the visibility risk map when path planning (See Fig. 1c), we use the OMPL planner [9] on the physical system and for the realistic simulation and the A-star planner [48] for our simplified simulations. Both planners can plan paths within time constraints and subject to state costs, which in our case is the visibility risk map, and an occupancy map of obstacles.