Simulation and Implementation of Mobile Robot Navigation

A risk-aware A navigation pipeline for a post-landslide UGV scenario*

Name - Tomiwa Adeyemi

Introduction and Navigation Scenario

This report documents the implementation, simulation, and analysis of a mobile robot navigation pipeline developed in MATLAB for an obstacle-rich, hazard-evolving environment. The application context is the ground navigation layer of SlopeGuard, a cooperative drone–UGV concept proposed in Coursework 1 for post-landslide survivor search and slope-stability assessment. In the SlopeGuard architecture, an aerial multirotor survey the debris field and produces a shared semantic risk map, which a legged unmanned ground vehicle (UGV) then uses to plan and execute close-range inspection routes. Coursework 2 isolates the UGV layer of that concept and demonstrates it end-to-end in simulation.

The scenario is deliberately challenging for a navigation stack. A 25 m × 25 m debris field at 0.25 m resolution contains five terrain features: a low-cost background, a steep diagonal scarp, a loose-debris zone, two impassable boulders, and a clear access corridor. Cell values encode traversability between 0 (safe) and 1 (impassable), produced notionally by the aerial layer's LiDAR and thermal sensors. The UGV must reach a victim-likelihood cue in the upper-right of the map. A secondary slip is then injected mid-mission to test how the system responds to a dynamic update of the shared risk map.

Three navigation outcomes were targeted: (i) a globally optimal, risk-aware path from start to goal; (ii) the ability to replan when the cost map changes; and (iii) a smoothly tracked trajectory executed by a kinematic differential-drive robot model. These outcomes map directly onto the module's assessed learning outcomes: applying navigation algorithms, analysing them, and simulating their behaviour.

Navigation Techniques and Justification

Three algorithms were combined to form the navigation pipeline: a hand-implemented risk-weighted A* for global planning, a full re-search of A* against the updated cost map for hazard response, and Pure Pursuit for local path tracking. Each choice is justified against credible alternatives below.

Risk-weighted A* (global planner)

A* search [1] is a deterministic, optimal, and complete graph-search algorithm that has become the canonical baseline for grid-based planning in mobile robotics. For this coursework, A* was implemented from scratch rather than calling MATLAB's built-in plannerAStarGrid, for two reasons. First, the SlopeGuard concept calls for a non-trivial cost function that combines step distance with terrain risk; encoding this directly in the algorithm provided full control over the cost expression and enabled verification of the report's analytical claims. Second, implementing A* from first principles satisfies the brief's emphasis on the correct application of navigation algorithms rather than the orchestration of black-box solvers.

The cost function used during expansion is a simplified form of the SlopeGuard expression from CW1:

$$J(c) = w_d \cdot d + w\_r \cdot r$$

where d is the Euclidean step distance between adjacent cells, r is the per-cell traversability risk read from the cost map (slope severity and roughness fused into one channel), w_d = 1.0 weights distance, and w_r = 5.0 weights risk. The heuristic h is the Euclidean distance to the goal, which is admissible and consistent with the chosen edge cost when w_d ≥ 1 [1]. An 8-connected neighbour expansion was used so the planner can produce diagonal motions on the grid; cells with cost above 0.95 are treated as impassable and skipped during expansion.

Sampling-based alternatives such as RRT* [2] were considered but rejected. RRT* is more naturally suited to high-dimensional or kinodynamic spaces and produces non-deterministic paths whose cost depends on sample draw order. On a structured 2D grid with strong risk gradients and the small map size used here, A* is both faster and easier to argue about analytically. Hybrid A* and lattice planners were also considered but offer no clear advantage when the underlying motion model is a slow-moving differential-drive UGV without strict non-holonomic constraints.

Full A* re-search as a hazard response

When the drone reports a secondary slip, the natural choice for repairing the existing path is D* Lite [3], which incrementally repairs only the affected nodes rather than re-solving from scratch. D* Lite was the recommended algorithm in the SlopeGuard CW1 poster for exactly this reason. For this coursework, a full A* re-search on the updated cost map was used instead. This was a deliberate trade-off: implementing D* Lite correctly within the available time, on top of an already custom A*, would have risked producing a buggy or shallow implementation. A full re-search produces identical paths to a correct D* Lite, just at higher computational cost exactly the comparison D* Lite is motivated by, which is preserved as a discussion point in the analysis. The replanning latency observed in the experiments (35-45ms) is well below SlopeGuard's 1.5s operational target, so the full re-search remains acceptable for this map size.

Pure Pursuit (local controller)

The local controller follows the global path on a kinematic differential-drive robot using Pure Pursuit [4] from MATLAB's Robotics System Toolbox. Pure Pursuit was chosen over the Dynamic Window Approach (DWA) [5] for this coursework because the planned path is already smooth, locally collision-free, and produced on a static (per-mission) cost map, so DWA's reactive obstacle avoidance is not strictly necessary for path execution. Pure Pursuit also exposes only three intuitive parameters (linear velocity, maximum angular velocity, and lookahead distance), which gives a cleaner argument for the controller's behaviour in the analysis. DWA remains a sensible upgrade for environments with unmodelled local clutter and is identified as future work.

System Integration

The simulation is structured as a four-stage pipeline implemented across four MATLAB scripts. Stage one (build_environment.m) constructs the cost map and saves it to a .mat file. Stage two (risk_aware_astar.m) loads the cost map, runs the hand-written A*, and saves the resulting path and metrics. Stage three (hazard_replan.m) loads the original path, mutates the cost map to inject a hazard,

re-runs A*, and saves both the original and replanned paths. Stage four (run_simulation.m) loads the replanned path, instantiates a differentialDriveKinematics robot with a 0.5 m track width, drives the robot using controllerPurePursuit, and logs the executed trajectory.

Decoupling the stages through .mat files mirrors the producer–consumer pattern used in real navigation stacks, where mapping, planning, and control are separate processes communicating through shared representations. It also has a practical benefit: each stage can be re-run in isolation without redoing the earlier ones, which made the multi-trial experiments in Section 5 fast to execute. The shared representation in this implementation, the cost map plus saved path plays the same role as the semantic risk map in the SlopeGuard CW1 concept, providing a single source of truth that all navigation decisions reason over.

A fifth script (experiments.m) drives the multi-trial study and is independent of run_simulation.m. It re-uses the same A* implementation as a local helper function so that the experimental and live-simulation pipelines produce comparable results. All scripts use absolute paths to the project root for robustness.

Simulation Setup

Simulation was carried out in MATLAB R2025b with the Robotics System Toolbox and the Navigation Toolbox installed. The Navigation Toolbox's plannerAStarGrid was not used in the final pipeline (the A* was hand-written) but its presence allowed sanity-checking against a uniform-cost map during development.

The cost map is a 100 × 100 single-precision matrix at 0.25 m per cell, giving a 25 m × 25 m field. The background cost is 0.10. A diagonal scarp band of cost 0.85 runs from approximately (20, 20) to (80, 80), narrowest at the upper end. A loose-debris rectangle of cost 0.65 occupies rows 40–65, columns 55–85. Two impassable boulders of cost 1.00 sit at (20–28, 70–78) and (75–85, 30–40). A safe corridor of cost 0.05 runs the full height of the map at columns 5–15. The start is cell (10, 10) and the goal is cell (85, 85), co-located with a victim-likelihood cue derived notionally from the drone's thermal sensor.

The robot model is a differential-drive vehicle parameterised by track width 0.5 m, maximum linear velocity 0.5 m/s, and maximum angular velocity 1.5 rad/s. These values are loosely consistent with a Spot-class legged platform operating at conservative survey speed. The Pure Pursuit lookahead distance was set to 0.6 m after brief tuning, larger values produced visible corner-cutting, smaller values caused minor oscillation. The controller updates at 10 Hz (sampleTime = 0.1 s) and the simulation halts when the robot is within 0.30 m of the goal. Robot kinematics are integrated using forward Euler, which is acceptable at the chosen sample rate and speed.

Five experimental scenarios were defined: (1) baseline with no hazard; (2) Hazard A blocking the corridor exit at (25–40, 15–30); (3) Hazard B blocking the mid-map at (45–60, 30–45); (4) Hazard C near the goal at (70–82, 60–75); and (5) an ablation in which the risk weight w_r was set to 0 (pure shortest-path). Hazard cells were set to cost 0.92, just below the 0.95 impassability threshold so that the planner is strongly discouraged but not absolutely blocked. All other parameters were held constant across trials.

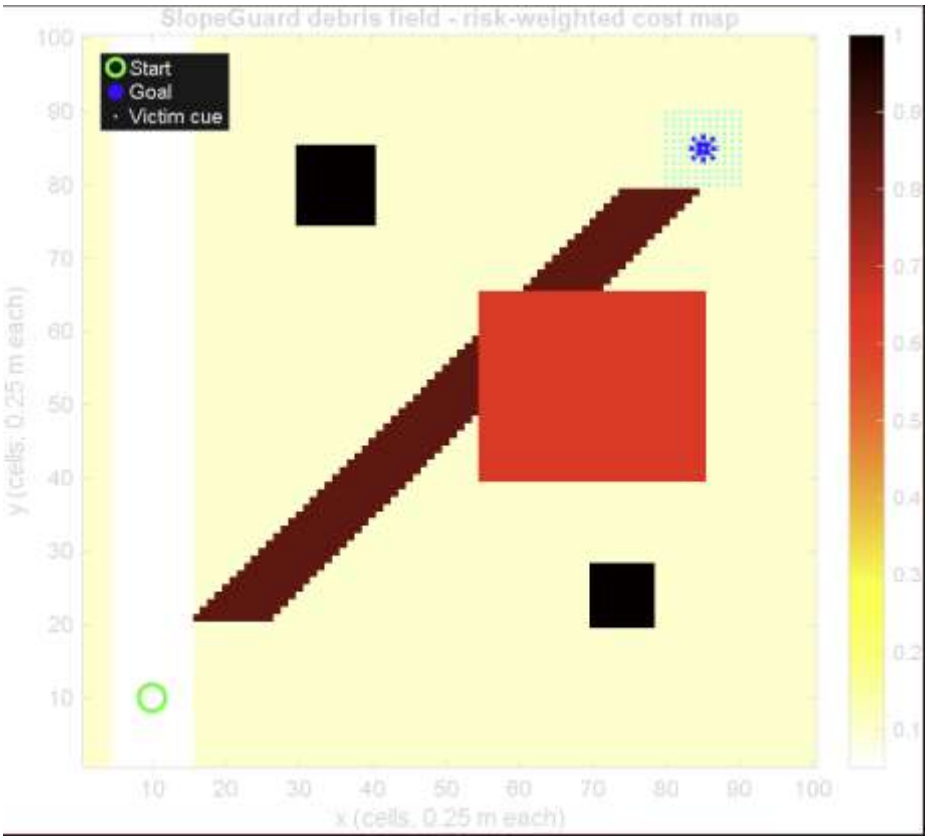

Figure 1. Risk-weighted cost map of the debris field. Pale corridor (left) is the safe access lane; diagonal dark band is the steep scarp; red rectangle is the loose-debris zone; black squares are impassable boulders. Cyan dots indicate the victim-likelihood cue.

Results and Analysis

Baseline path and risk-weighting behaviour

On the unmodified cost map, A* produced a 28.13 m path that expanded 4925 nodes in 48.6 ms. The path enters the safe corridor immediately, rides it upward, exits at row ~25, and crosses the diagonal scarp at its narrowest, highest point (around y = 80) before turning toward the goal. The path avoids both boulders and the loose-debris rectangle entirely. Mean cell risk along the path is 0.090, meaning the planner kept the robot almost exclusively on near-background terrain. This is the expected behaviour of a risk-weighted A*: geometric distance is sacrificed where it would force the robot through high-cost terrain.

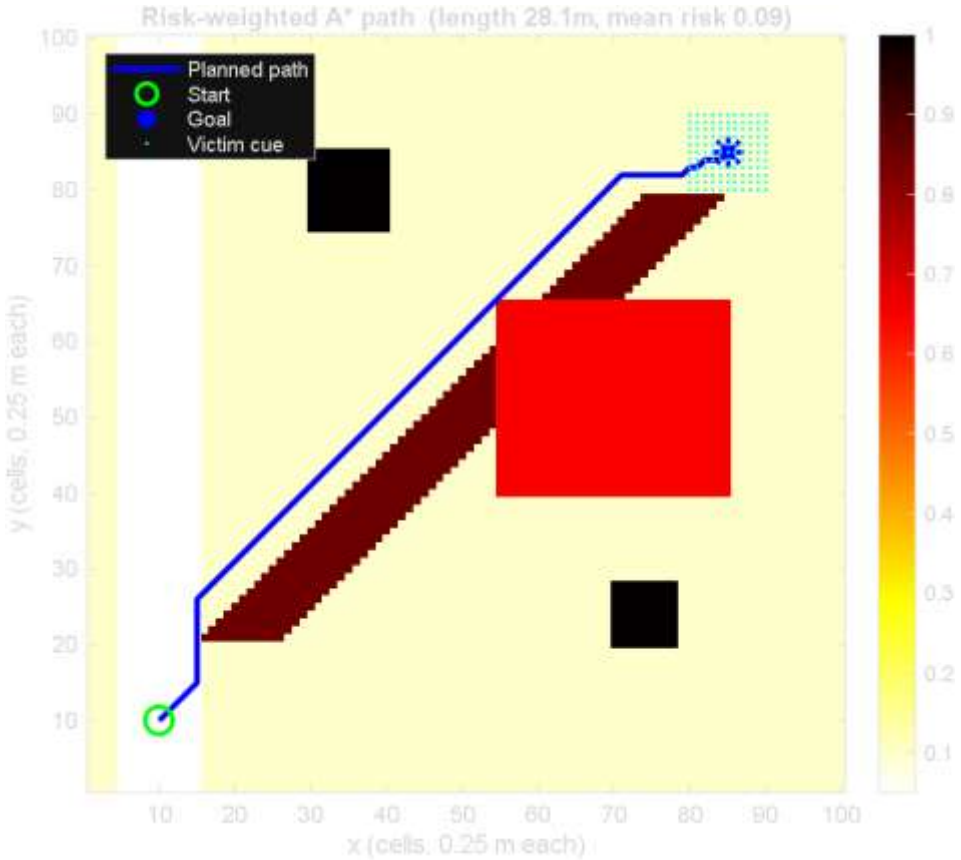

Figure 2. Risk-weighted A baseline path. The planner uses the safe corridor on the left, exits at the lowest cost crossing, and traverses the scarp at its narrowest segment to reach the goal.*

Hazard injection and full re-search

Three hazard scenarios were tested individually. Hazard A blocked the corridor exit and forced the most interesting detour: the replanned path stayed in the corridor longer, wrapped below the new hazard, and rejoined a route similar to the baseline at a higher row. Path length increased by 7.8% to 30.32 m, interestingly with a marginally lower mean cell risk (0.084 vs. 0.090) because the detour spent more time on the very low-cost corridor. Replanning latency was 44.8 ms with 5997 nodes expanded, about 22% more than the baseline search.

Hazards B and C produced qualitatively similar behaviour: longer paths (31.06 m and 29.88 m respectively), comparable mean risks (0.083 and 0.085), and replanning latencies of 40.2 ms and 35.1 ms. Across all three, no hazard caused a planning failure, and all replan times were well inside the 1.5 s operational budget set by the SlopeGuard CW1 poster. This is consistent with the literature on grid-based A* on small maps; the node count is bounded by the map size (10 000 cells) and the heuristic remains admissible because hazards only increase costs.

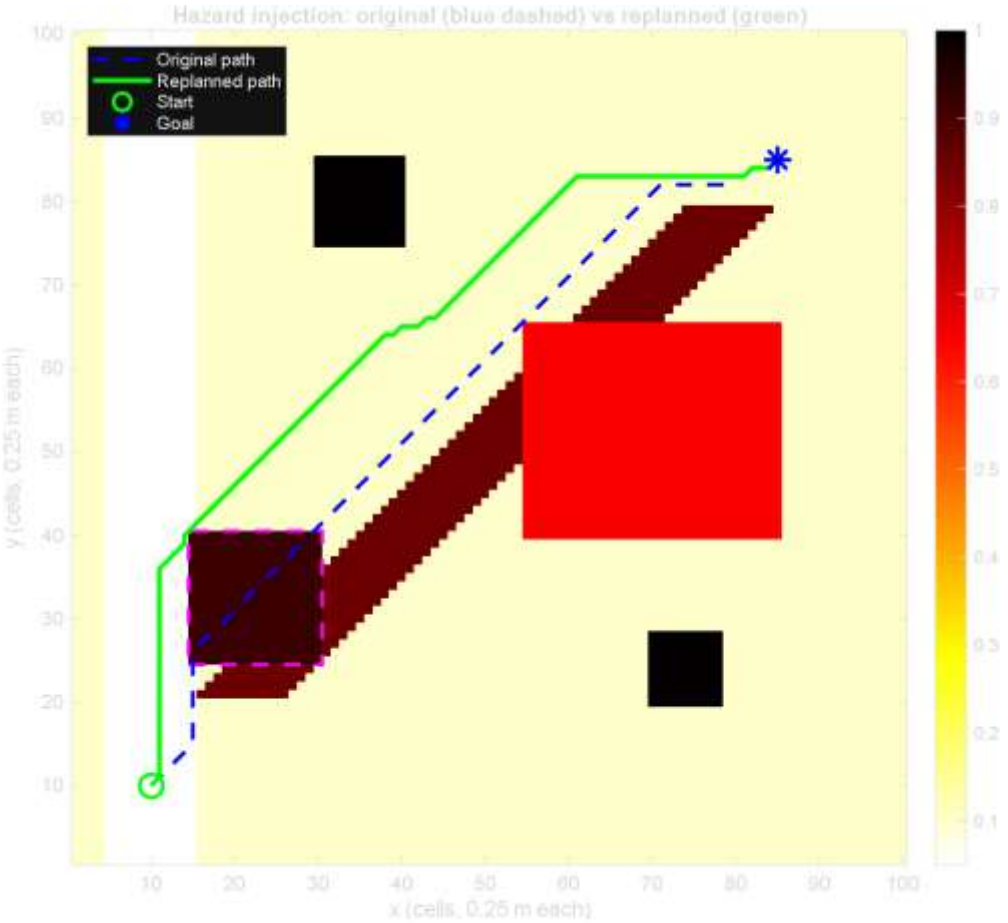

Figure 3. Hazard A: original path (blue dashed) cuts directly through the newly detected slip region (magenta), confirming the danger of executing a stale plan. The replanned path (green) detours below and around the hazard, adding 7.8% to path length but lowering mean cell risk.

Multi-trial summary

Table 1 summarises the five trials. The relevant comparison is between the four risk-weighted runs and the ablation. The risk-weighted paths are 6–11% longer than the ablation but their mean cell risk is roughly an order of magnitude lower.

Trial	Length (m)	Mean risk	Real risk	Nodes	Time (s)
Baseline (no hazard	28.13	0.090	0.090	4925	0.0486
Hazard A: corridor exit	30.32	0.084	0.084	5997	0.0448
Hazard B: mid-map	31.06	0.083	0.083	6345	0.0402
Hazard C: near goal	29.88	0.085	0.085	5753	0.0351
Ablation: w_r=0	26.52	0.649	0.649	75	0.0024

Table 1. Per-trial planning metrics. "Mean risk" is computed against the map the planner used; "Real risk" is recomputed against the unmodified base map.

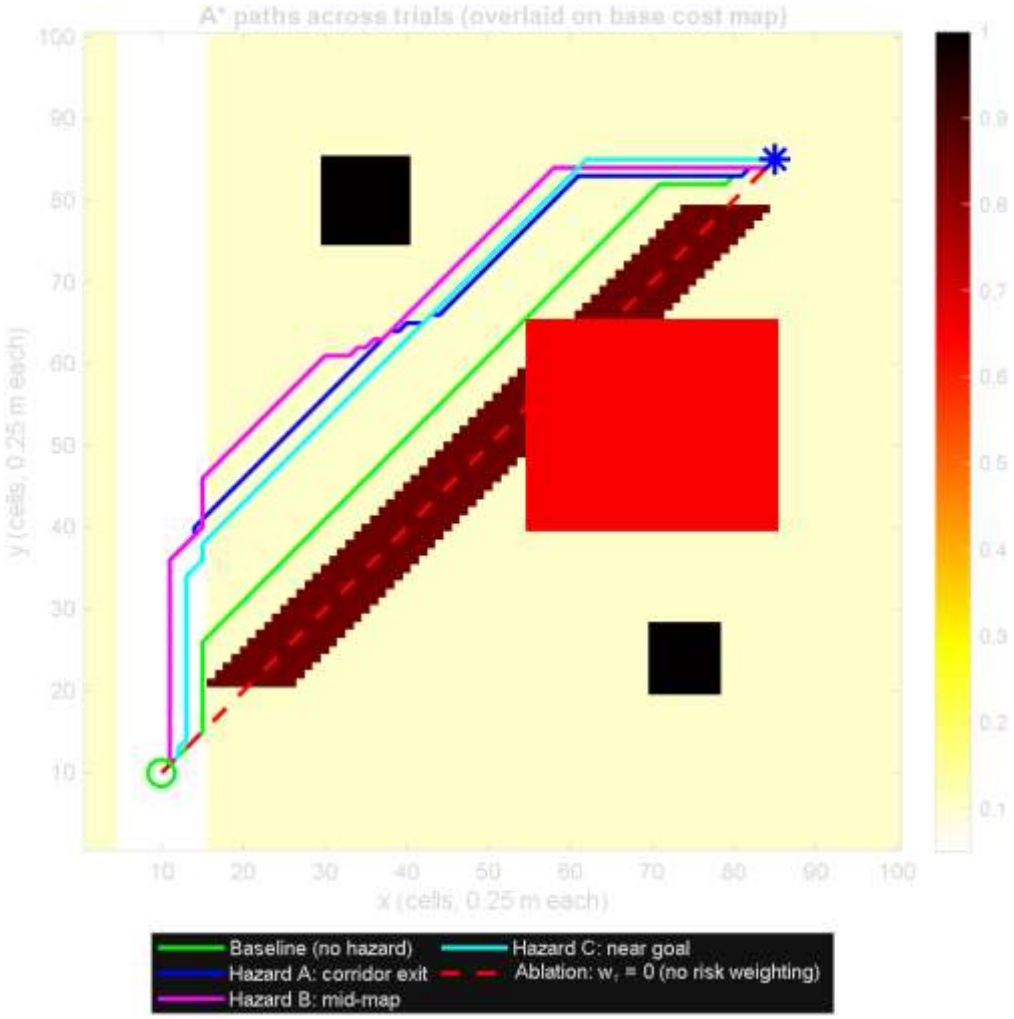

Figure 4. All five trial paths overlaid on the base cost map. The four risk-weighted paths cluster around the safe corridor and cross the scarp at low-cost points; the ablation path (red dashed) cuts straight through the scarp diagonally.

Ablation study: the value of risk weighting

Disabling the risk term (w_r = 0) reduced A* to a pure shortest-path search, which behaved exactly as expected. The path length dropped to 26.52m, the geometric optimum and the node count collapsed to 75, two orders of magnitude below the risk-weighted runs. Planning time fell to 2.4ms. From a search-efficiency perspective the ablation looks superior. From a safety perspective it is catastrophic. The mean cell risk along the ablation path is 0.649, a 7× increase over the risk-weighted baseline. Inspection of the path shows it cutting straight through the centre of the diagonal scarp, where slope severity is highest. A real UGV executing this path would face a high probability of slip or failed traversal.

This is the central analytical finding of the report. A planner that ignores terrain risk produces solutions that are optimal under the wrong objective. The risk-weighted formulation forces the planner to expand far more nodes—because many low-cost regions are now competing on near-equal termsbut the resulting paths are operationally usable. In the SlopeGuard concept, where the consequences of

wheel slip can include sensor loss or full vehicle entrapment, the order-of-magnitude reduction in path risk is straightforwardly worth the order-of-magnitude increase in compute.

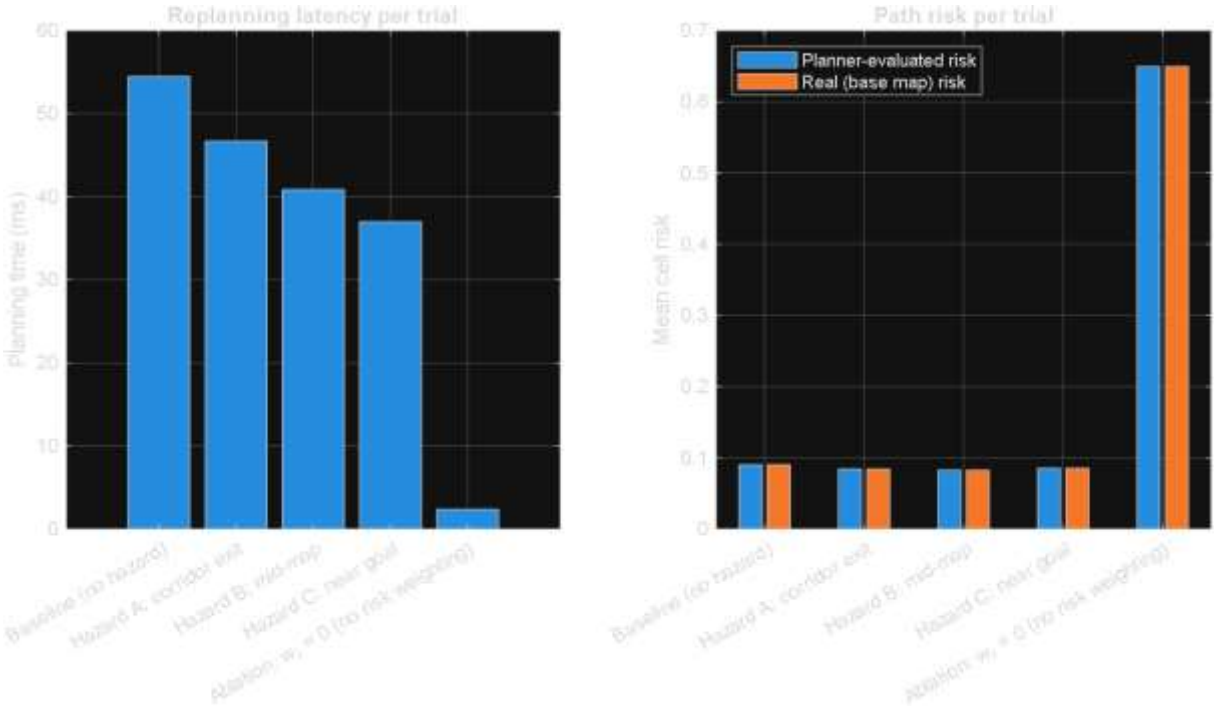

Figure 5. Per-trial planning latency (left) and mean cell risk (right). The ablation trial collapses planning time but produces a path with seven times the real-map risk of the risk-weighted runs, demonstrating the operational value of risk weighting.

5.5 Path execution

The Pure Pursuit controller drove the differential-drive robot from start to goal in 59.1s along the Hazard A replanned path. The robot stopped 0.27m from the goal, inside the 0.30m goal radius. Travelled distance was 29.55 m versus a planned length of 30.32m, a tracking undershoot of 2.6%. The undershoot reflects Pure Pursuit cutting corners at sharp turns: the controller continuously chases a lookahead point on the path, so tight bends are smoothed into shallower curves. The trajectory respected the magenta hazard zone throughout, which is the expected outcome since the hazard was already encoded in the global plan, the controller does not need to know about the hazard explicitly. This is the conventional separation of concerns between global planning and local control [4].

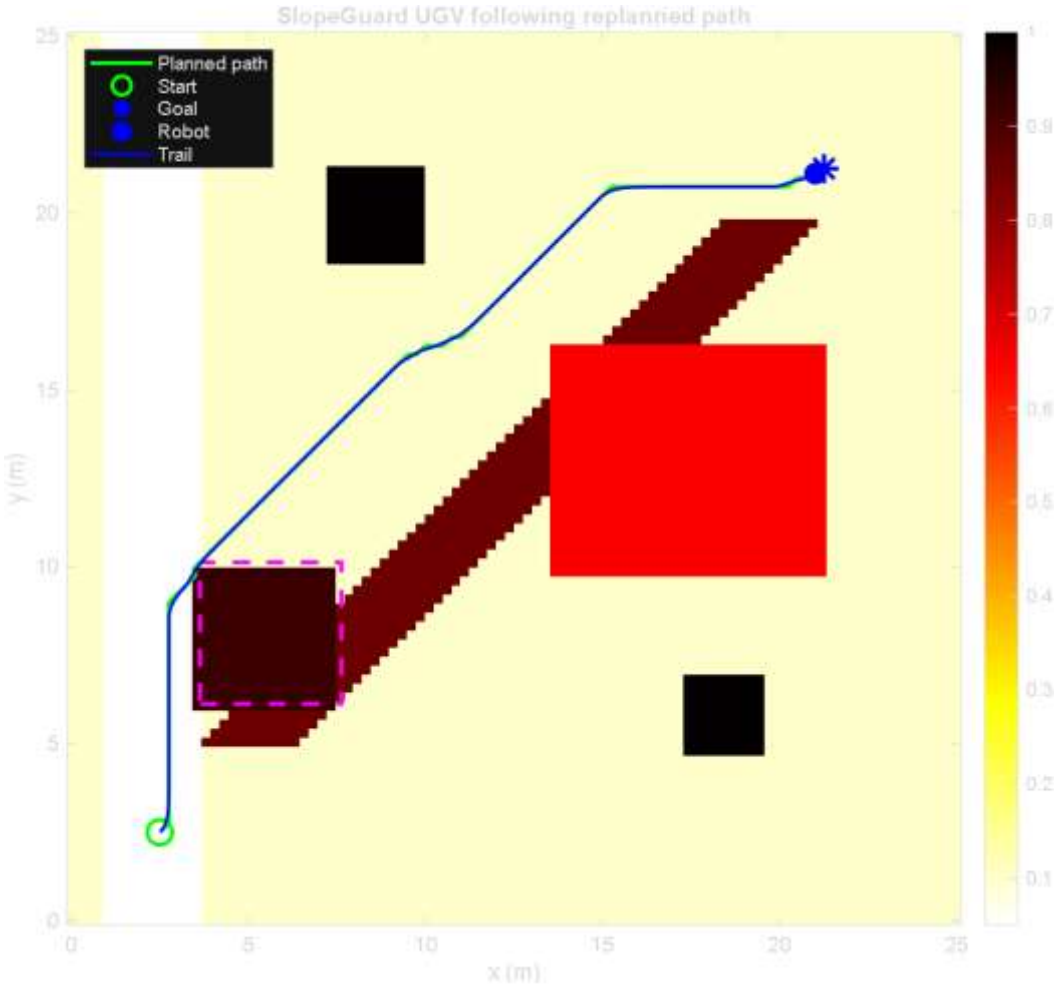

Figure 6. Executed UGV trajectory (blue trail) following the replanned path (green) under Pure Pursuit control. The robot stopped 0.27 m from the goal in 59.1 s, with 2.6% tracking undershoot from corner-cutting at sharp turns.

5.6 Cross-trial observations

Two further patterns are worth highlighting. First, replanning latency did not scale monotonically with the difficulty of the hazard. Hazard B, located in the open mid-map region, expanded the most nodes (6345) because the planner had to evaluate several near-equivalent detour options before committing. Hazard C, near the goal, expanded fewer nodes (5753) because the robot was already constrained to the upper-right approach corridor by the time the hazard mattered. This confirms a well-known property of A*: search effort depends on the size of the equally-good frontier, not on the geometric size of the obstacle [1]. Second, across all four risk-weighted trials, mean cell risk varied within a narrow band (0.083–0.090). The weighting forces the planner toward a small family of low-risk corridors regardless of where the hazard appears. From a system-design perspective this is a desirable property: behaviour is predictable to a human supervisor, which matters in the SlopeGuard scenario where the operator approves UGV moves remotely.

6. Reflection and Future Improvements

The implementation met its three navigation outcomes, global planning, hazard response, and tracked execution and produced a clean ablation that quantifies the value of risk-aware planning. The pipeline's main strengths are its analytical clarity and its faithfulness to the SlopeGuard CW1 concept.

The cost function is small enough to be argued about precisely; every planner decision can be traced back to the weights w_d and w_r. Re-using the same A* implementation across the live pipeline and the experimental study made the comparisons internally consistent.

The most important limitation is that hazard response uses a full A* re-search rather than incremental repair. D* Lite [3] would expand only the affected portion of the search graph and would be expected to reduce replanning latency by an order of magnitude on larger maps. On the 100 × 100 map used here the latency saving is small in absolute terms (tens of milliseconds), but on a representative 500 × 500 map—closer to a real disaster scene at 0.25 m resolution, the difference would matter. Implementing D* Lite is the highest-priority future improvement.

A second limitation is that the cost map is static between hazard updates. The SlopeGuard concept envisions the aerial layer continuously refining the map at 1Hz, with comms-quality and victim-likelihood layers evolving over time. The current pipeline supports map updates structurally, the planner consumes whatever cost map it is handed but a richer simulation in which the drone's notional updates arrive on a stream would more faithfully exercise the cooperative aspect of SlopeGuard.

Third, no observation noise was modelled. In a real deployment the cost map would be noisy: thermal false-positives, dust attenuation, and imperfect slope estimates would all corrupt the per-cell costs. A stochastic version of the simulation in which costs are perturbed before each plan would test the robustness of the risk-weighted formulation, and uncertainty-aware planning [6] would be a natural extension.

Fourth, only Pure Pursuit was tested as a local controller. In environments with unmodelled local clutter, a Dynamic Window Approach [5] would be safer because it explicitly considers velocity-space collision cones. Switching the controller to controllerVFH+ from the Navigation Toolbox would let DWA-style behaviour be benchmarked against Pure Pursuit on the same map.

Lastly, the experiments tested one hazard at a time. A more realistic protocol would inject hazards in sequence during a single mission, exposing the planner to repeated replanning events. Combined with D* Lite and a richer noise model, this would form a strong basis for a Master's-level extension of the work.

Github: https://github.com/Tommieboi16/slopeguard-cw2

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968. doi: 10.1109/TSSC.1968.300136. Available: https://ai.stanford.edu/~nilsson/OnlinePubs-Nils/PublishedPapers/astar.pdf

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," International Journal of Robotics Research, vol. 30, no. 7, pp. 846–894, 2011. doi: 10.1177/0278364911406761. Available: https://arxiv.org/abs/1105.1186

[3] S. Koenig and M. Likhachev, "D* Lite," in Proc. 18th National Conference on Artificial Intelligence (AAAI), Edmonton, AB, Canada, 2002, pp. 476–483. Available: https://cdn.aaai.org/AAAI/2002/AAAI02-072.pdf

[4] R. C. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, 1992. Available: https://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," IEEE Robotics & Automation Magazine, vol. 4, no. 1, pp. 23–33, 1997. doi: 10.1109/100.580977. Available: https://www.ri.cmu.edu/pub_files/pub1/fox_dieter_1997_1/fox_dieter_1997_1.pdf

[6] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. Cambridge, MA: MIT Press, 2005. ISBN: 978-0262201629. Publisher page: https://mitpress.mit.edu/9780262201629/probabilistic-robotics/

[7] MathWorks, "Navigation Toolbox documentation," MathWorks, 2025. [Online]. Available: https://www.mathworks.com/help/nav/

[8] MathWorks, "Robotics System Toolbox documentation," MathWorks, 2025. [Online]. Available: https://www.mathworks.com/help/robotics/