2023 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
<AUTONOMY, ARTIFICIAL INTELLIGENCE, & ROBOTICS> TECHNICAL SESSION
AUGUST 15-17, 2023 - NOVI, MICHIGAN

ROBUST PATH PLANNING IN THE BATTLEFIELD

Thomas Jonsson Damgaard[1], Mikael Rittri, PhD[1], Patrick Franz[1], Anika Halota[1]

[1]Carmenta Geospatial Technologies

ABSTRACT

Autonomous vehicles rely on path planning to guide them towards their destination. These paths are susceptible to interruption by impassable hazards detected at the local scale via on-board sensors, and malicious disruption. We define robustness as an additional parameter which can be incorporated into multi-objective optimization functions for path planning. The robustness at any point is the output of a function of the isochrone map at that point for a set travel time. The function calculates the sum of the difference in area between the isochrone map and the isochrone map with an impassable semi-circle hazard inserted in each of the four cardinal directions. We calculate and compare two different Pareto paths which use robustness as an input parameter with different weights.

Citation: T. Jonsson Damgaard, M. Rittri, P. Franz, A. Halota "Robust Path Planning in the Battlefield," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 15-17, 2023.

1. INTRODUCTION

Mission planning in combat scenarios begins with the path planning phase. The quality of a planned path is based not only on its speed, but also on its ability to provide viable alternatives in the event of unexpected blockages. In April 2022, the Russian army was forced to abandon numerous tanks due to a lack of foresight in this respect. A planned offensive was thwarted as Ukrainian defenders blocked off the sole path along the main road, and vehicles diverted off-road into muddy terrain, which proved to be impassable [1].

Autonomous vehicles, like manned vehicles, are guided by a long-range path set during initial planning. The path can be updated dynamically based on actual travel and newly sensed information. Path planning is commonly performed using graph traversal algorithms such A* [2] or D* [3]. Edge costs are determined based on environmental parameters, including slope, terrain, and land cover, as well as road network data.

During the mission, on-board sensors provide a continuous stream of high-resolution data, including

obstacles that must be avoided and impassable slopes. Various techniques are under development with the goal of rapidly navigating a vehicle around local obstacles, while keeping it focused on the long-range path to its destination [4, 5, 6].

Existing path planning methods do not provide or incorporate any information on the susceptibility of the path to disruption by opposing forces, nor to the possibility of locally impassable zones. These obstructions could be due to newly sensed hazards or unexpectedly uneven terrain that could not be detected in advance with the lower resolution data used in the path planning algorithm. Depending on the severity of the obstruction, this could lead to a complete disruption of the mission when no alternative route is available.

In this paper, we define a new parameter, robustness, which can be incorporated into the path planning function. In general, robustness is a measure of the potential time implications of a reroute, should a hazard occur which prevents the vehicle from proceeding along its planned path. We briefly describe the relevant background for graph theory and isochrone maps, followed by a presentation of the novel region-based robustness algorithm. We display our results in the form of screenshots from a custom application, in which we implement multi-objective optimization to incorporate robustness into our path planning algorithm. We conclude with a discussion of the results and how they can be applied in mission critical applications.

2. BACKGROUND

This paper uses properties of graph theory specifically with the shortest path problem to generate a new objective function used in multiobjective optimization. The new objective function is generated using penalty time, a property defined in the quickest path problem (similar to the shortest path problem) when the solution deviates away from the optimal solution due to obstacles blocking off the original route. In this paper, we will refrain from distinguishing between shortest and quickest path and instead use the word shortest for the general case, meaning lowest cost. Using the duality gap between the shortest path problem and the shortest path tree (routing vs isochrones), penalty time is measured in changes to the respective area of the isochrones generated from the shortest path tree when obstacles block off a traversable area of arbitrary Euclidean size.

2.1 Graph Theory

In graph theory [7], a graph G is commonly written as $G = (V, E)$, where V represents the vertices (commonly referred to as nodes) and E represents the edges (commonly referred to as links). A vertex is defined as a position inside of a graph. An edge is defined as a pair of vertices. In this paper we use directed graphs, where the edges are ordered pairs of vertices.

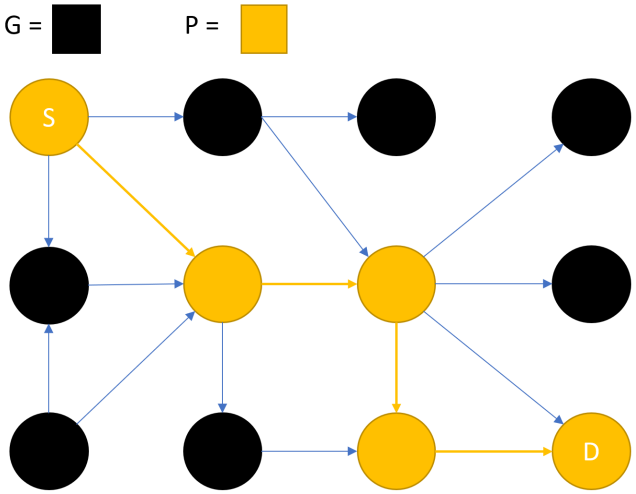

Figure 1: *A graph G in black and a path P from vertex S (start) to vertex D (destination) in yellow.*

A path (or a route) in a graph is a sequence of vertices, $P = (v_1, v_2, ..., v_n) \in V \times V \times ... \times V$ such that (v_i, v_{i+1}) is an edge of the graph for all $i < n$. A *path planning optimization problem* is

a problem where we seek one or more routes and where we prefer routes with a lower *cost* as defined by an *objective function*. This is commonly known as the shortest path problem [2]. The objective function for the shortest path problem is defined as,

$$\sum_{1 \leq i < n} f\left(v_i, v_{i+1}\right) \tag{1}$$

This objective function sums the cost of traversing each edge in the sequence P from v_1 to v_n where f gives the cost value measured as the distance between v_1 and v_{i+1}. In general, the objective function for a path is commonly the summed cost for all edges or all vertices in a path.

In this paper, we will use the term shortest path problem as a collective word for all path planning algorithm where the objective function is to accumulate the lowest total cost. The cost can, however, be defined in units other than distance, such as travel time.

2.2 Isochrone Maps

An isochrone map displays the subset of a graph that can be reached from a certain start vertex within some set travel time. The single-source shortest path tree generates a tree of vertices connected to a starting vertex given a time cost threshold. It is comparable to the principle of path planning algorithms.

Unlike the shortest path problem, the isochrone algorithm creates isolines along constant values in a unit of time. Instead of looking at a sequence, the isoline expresses the vertices that can be reached from a starting vertex within a time threshold. In a graph G we denote the isochrone map that originates at vertex v and is bounded by the threshold t by G_v where we deliberately let t be implicit to reduce clutter in the formulas.

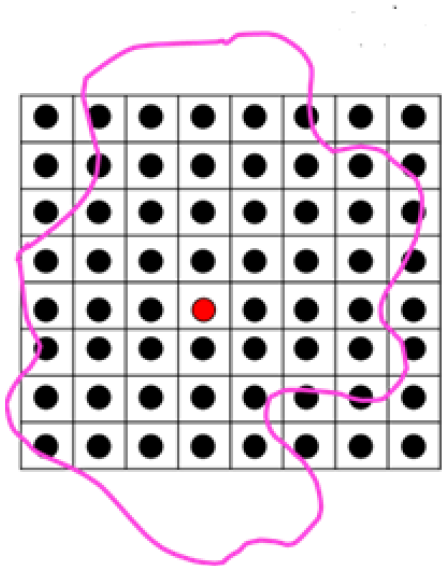

Figure 2: *An isochrone map originating at the red vertex and bounded by the pink outline (graph edges not shown).*

In this paper we also make some other assumptions, namely that:

1. each vertex has a geographical location, so distances between them are well-defined,
2. distances in 3D space are highly correlated with the corresponding distances in 2D space, so one can regard the search as taking place on a 2D plane except for bridges and tunnels,
3. the amount of ground covered from a start vertex within a certain cost threshold can be quantified as an area, for example in km^2.

2.3 Shortest Path Problem vs Shortest Path Tree

There is a mathematical connection between solving a routing problem (shortest path problem) and creating an isochrone access map (shortest path tree). The shortest path problem results in a path connecting two vertices with the minimal accumulated cost. The shortest path tree results in a subgraph that illustrates the possible vertices reached from the start vertex within a maximum accumulated

cost. The accumulated cost is calculated using equation 1.

In the shortest path problem, the accumulated cost is the objective function to the minimization problem. In the shortest path tree, this is a constraint to the maximization problem of generating a subgraph. From optimization theory, this is connection is similar to duality problems. The shortest path tree is the duality problem of the shortest path problem and can generate an upper bound under a certain condition. The shortest path tree needs to be large enough to where both vertices exist inside the subgraph, the starting vertex, and the destination vertex. If no such subgraph exists, one needs to increase the constraint limit on the shortest path tree. If no solution is found for large enough search bounds, then no feasible solution will exist for the shortest path problem.

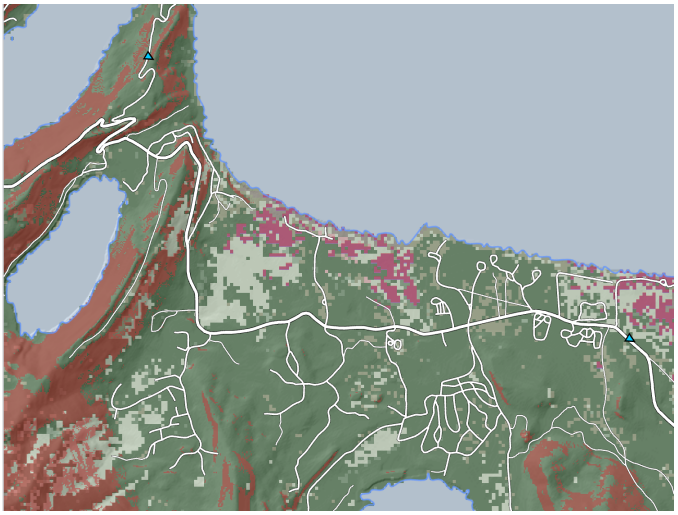

Figure 3: *A compact routing scenario using a mixed network of roads and traversable terrain together with a proposed vehicle. Traversable terrain is painted green while non-traversable is painted red. Roads are visualized using white lines with thickness illustrating road type class.*

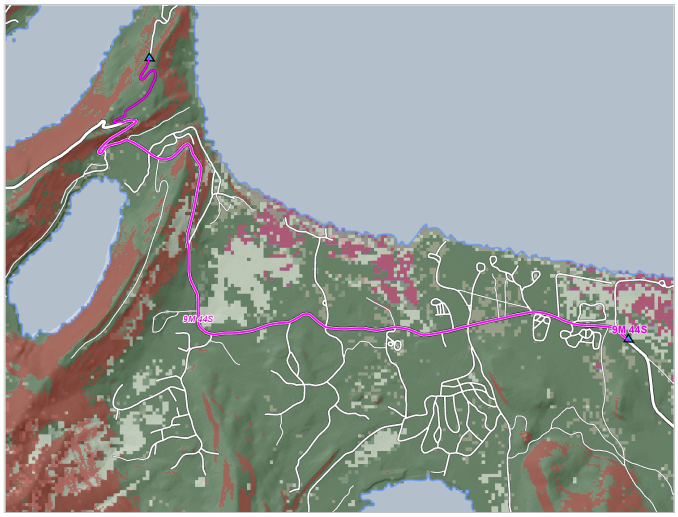

Figure 4: *A solution to the quickest path problem using the scenario described in figure 3. Solution was found using the D*-algorithm. The purple line visualizes the route and the travel time is printed in text along the route. The travel time is 9 minutes and 44 seconds.*

The starting condition of a path planning problem is described in Figure 3. Problem is to go from the starting position marked with a blue arrow in the top left corner and reach the destination in the bottom right corner marked with a blue arrow. To define the quickest path, a vehicle model is introduced that has an associated vehicle speed for each type and class of environment, such as terrain class or road class. Blue color illustrates water, however, the vehicle in use is said to not be amphibious. Using said vehicle, together with the combined network as input (roads and terrain) the quickest path problem can be solved visualized in figure 4. This scenario will be used to illustrate the difference between the shortest path problem and the shortest path tree.

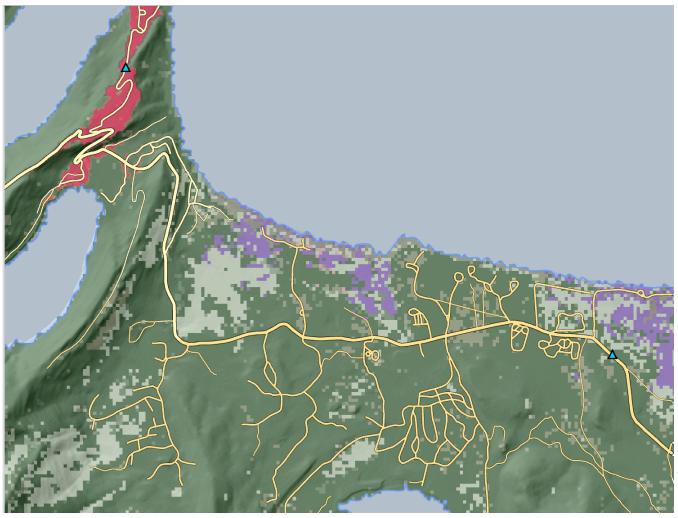

Figure 5: *Studying the shortest path problem using the shortest path tree visualized using isochrones. The conditions of this scenario is presented in figure 3. The isochrone is calculated using the top left starting position as the reference vertex and the solution is visualized as a red area. The red area visualizes all the positions reachable within the travel time constraint put on the isochrone. In this case the travel time constraint is 3 minutes.*

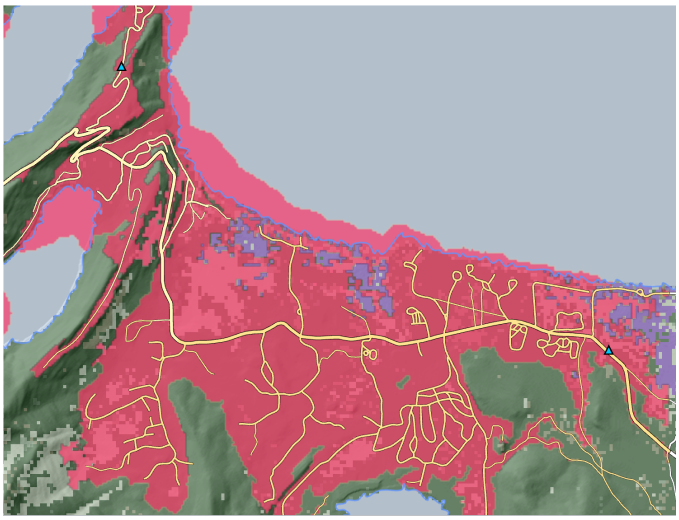

Figure 6: *A continuation of the problem described in figure 5. In this scenario the travel time constraint is increased from 3 minutes to 11 minutes. The red area visualizing the isochrone area has increased in size consequently.*

Figure 5 and figure 6 illustrates the duality gap between the shortest path tree and the shortest path problem. In the shortest path problem, the solution is a path illustrated in figure 4. Figure 5 visualizes the shortest path tree using isochrones painted in red. The isochrones are measured using the starting position as the reference vertex and a travel time constraint. If the starting position and the destination both exist within the red area, a duality gap exists and can be used to put an upper bound on the shortest path problem. This is depicted in figure 6, which tells us that there exists a path to our destination and it will take us at least 11 minutes to reach. Although, if the destination point does not exist inside the red area, nothing can be said about a feasible solution to the shortest path problem. This is shown in figure 5.

2.3.1 Penalty Time vs Shrinking Tree

In the shortest path problem, penalty time is defined as the additional time needed when rerouting away from the shortest path solution when finding alternative routes. Let's say an obstacle has appeared blocking the shortest path solution, deeming it no longer feasible. Instead, we need to reroute away from where the obstacle appeared and still try to reach the destination. The difference in travel time compared to the case without an obstacle is known as the penalty time.

$$t_{\text{Penalty}} = t_{\text{Alternative}} - t_{\text{Optimal}} \geq 0. \tag{2}$$

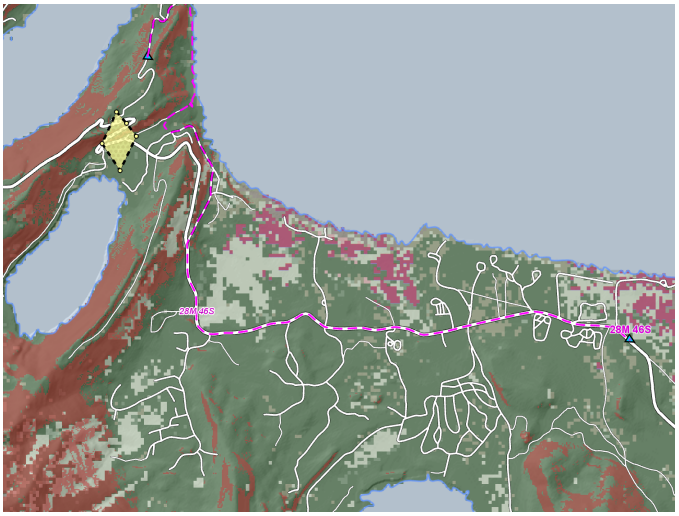

Figure 7: *Based on the scenario depicted in figure 3 the shortest path solution from figure 4 will change if an obstacle appears along the original planned route. An obstacle area painted in yellow has been placed in the first traffic situation just below the starting position. The new solution, painted in purple dashed lines, circumvents this obstacle by going along the shore. The new travel time is 23 minutes and 46 seconds. Using equation 2 the penalty time is 14 minutes and 2 seconds.*

What is the equivalent behavior in the shortest path tree? If an obstacle appears inside the isochrone area generated partly from the shortest path tree around a vertex v, the isochrone area will shrink in size consequently. This illustrates that one can predetermine the potential penalty time around a vertex v by studying how the shortest path tree shrinks in size when obstacles appear around the vertex.

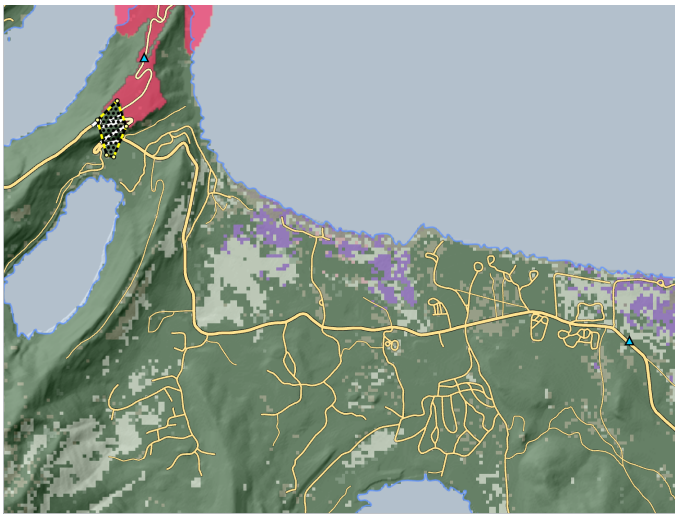

Figure 8: *Based on the solution presented in 6. When applying the same obstacle from figure 7, the isochrone area painted in red has decreased in size drastically from figure 6. Using the 11-minute travel time constraint, the starting position and the destination is no longer contained inside the red area. The dramatic shrinking of isochrone area is due to the large penalty time added from applying the obstacle, which is observed in figure 7.*

This phenomenon is the key to the robustness cost-map. The shortest path problem is context sensitive; you need a starting vertex and a destination vertex. This works fine when one wishes to analyze a route for potential weakness using penalty time as an indicating factor. This does not however, work for region-based understanding. For a region-based understand, studying the shrinking behavior of the shortest path tree around each vertex results in an equal overall understanding of the region with regards to potential penalty times when routing. Using this observation, we can develop a new cost-map from a robustness objective function that can be used in multi-objective optimization.

3. REGION-BASED ROBUSTNESS

Our goal in defining robustness is to generate a path that is minimally impacted by unexpected hazards. We suggest that this can be described by the reduction in size of accessible area from a point, should a section be blocked off. We therefore define region-based robustness as the sum of the difference in area between the isochrone map and the isochrone map with an impassable hazard inserted in each of the four cardinal directions.

3.1 Robustness Cost-map

Let's define a region $G(E, V)$ as a directed graph as in section 2.1. Each node $v \in V$ will then be used to generate a local isochrone map $G_v \subseteq G$, see figure 2. In each cardinal direction (north, south, west, east) a no-go zone is applied. The no-go zone can be arbitrarily defined, but in this algorithm, its defined as a half circle with a Euclidean radius and thickness. The no-go zone will describe the problematic accessibility towards that cardinal direction. For each cardinal direction, we generate a new isochrone G_v^D where $D \in \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}\}$ denotes that cardinal direction.

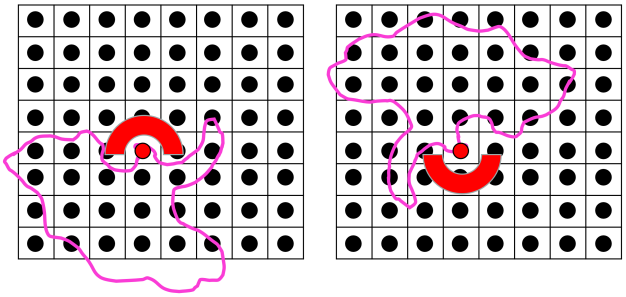

Figure 9: *The isochrone maps G_v^N (obstacle north) and G_v^S (obstacle south). The node v is marked in red and the areas $A(G_v^N)$ and $A(G_v^S)$ are highlighted in pink. The obstacle applied is the red half circle.*

The difference in isochrone areas with and without the obstacles can be used as a measure of the robustness for the vertex v, but since larger values for the difference mean worse robustness, it makes sense to call the measure the *fragility* for v, which we denote by $F(v)$. Formally, we calculate the area difference for each of the four obstacles and normalize it. Finally, we sum the differences:

$$F(v) = \frac{1}{4A(G_v)} \sum_{D \in \{\mathrm{N,S,E,W}\}} \left(A(G_v) - A(G_v^D) \right)$$

The unit of $F(v)$ is area loss as a percentage. We will use the same notation also for the fragility of an entire path P, namely $F(P)$, which is defined as the sum of the fragility values for each vertex of the path.

3.2 Balancing Robustness And Speed

Now that we have two cost maps, one for fragility and one for travel time, we can think about how to find a path that is both robust and fast. In other words, we want to minimize both the fragility and the travel time of the path. This is an example of *multi-objective optimization* [8], which is difficult in general since there may be no single solution that simultaneously optimizes each objective. Some compromises must be made, but there is no established way to know whether a certain decrease of one objective is worth a certain increase of some other objective; in other words, two different objectives can be seen as two currencies without any known exchange rate.

In our setting, if a path P_1 has less fragility but longer travel time than path P_2, then which path is best? If we do not want to declare an official exchange rate, then we do not know: the two solutions are not comparable. On the other hand, if path P_1 has both less fragility and shorter travel time then path P_2, then it is obvious that P_1 is better than P_2, and we say that P_1 *dominates* P_2.

For a given path optimization problem, the paths that are not dominated by any other path are called *Pareto optimal*. Each Pareto optimal point represents an optimal choice in the multi-dimensional space of objective functions. Each objective function, also known as cost, can be drawn as a dimension

in the multi-dimensional diagram and each Pareto point becomes a position in this space. In our case, the problem exists on a two-dimensional space with travel time along the horizontal axis and fragility along the vertical axis. Each point in this diagram represents a solution to our minimization problem. The Pareto points can be connected by a line known as the *Pareto front*. The Pareto front is a set of Pareto optimal points, in our case closest to the axes due to minimization. The Pareto front would go from the upper left to the lower right of the diagram, with all other paths above and to its right. See figure 10 for an illustration.

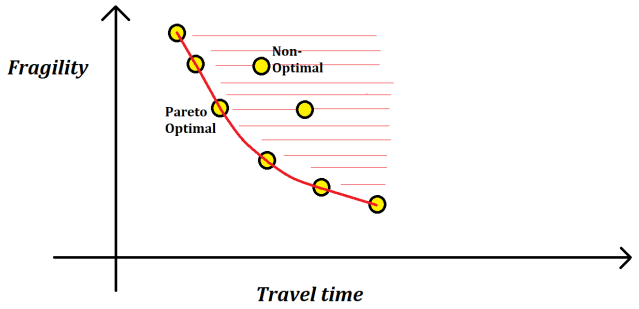

Figure 10: *Pareto optimal points displayed on a two-dimensional diagram. The horizontal axis displays travel time, and the vertical axis displays fragility. Any point in this space represents a solution to our minimization problem. The Pareto optimal points lie closest to the axis and dominates all other solutions. The Pareto front is the thick red line going through all Pareto optimal points. The thin red lined area showcases the region where all non-optimal solutions exist.*

One way to handle this situation is to generate all the Pareto optimal routes, or at least a large enough subset of them, and then present them to a human expert who can make the final choice. This approach is taken by Clawson et al. [9]. However, a part of their design is to preprocess the terrain into a graph with few vertices but many edges between them, which we think improves the performance of their sophisticated algorithm. And we think their algorithm could become much slower in our setting, where vertices correspond to a large number of terrain raster cells that have only eight outgoing edges to neighbors.

The purpose of this paper is not to explore different ways of doing multi-objective optimization. To be blunt, our main purpose is to explore whether our formula that defines fragility makes sense at all. A numerical definition of robustness can look sensible in theory, but when one uses it for route planning, one could possibly discover unexpected quirks that are not obvious at first glance. Since we only want to rule out the existence of confusing or counter-intuitive quirks at this stage, we will handle multi-objective optimization in a simple way: *scalarization* [8].

Scalarization in multi-objective optimization is the idea of defining a *superposition* of the set of objective functions to approximate the behavior of Pareto optimal points. A superposition is defined as a linear combination of multiple functions. We simply choose numeric weights w_F and w_T and declare that we want to minimize the weighted sum of the two objective functions, in other words we want minimize the single objective:

$$w_F F(P) + w_T T(P) \tag{3}$$

where P is a possible path and $T(P)$ is defined as the accumulated travel time for path P. Without loss of generality, one can assume that the weights are normalized, $w_F + w_T = 1$.

By using superposition, the objective function is now one dimensional with a clear ordering of cost values for each path. This simplifies the optimization problem: traditional one-dimensional path planning algorithms can now solve this. Although, the choice of values for the weights will remain a difficult task. By setting $w_F = 0$, the problem returns to the quickest path problem where travel time is the only concern. Vice versa, if we set $w_T = 0$ the problem now only concerns robustness. Each choice of pair of weights will generate a Pareto optimal point.

However, scalarization cannot find all solutions in a Pareto front that is not convex. Such limitation is due to the superposition of the objective function space, which cannot reach Pareto points if they occur on a concave Pareto front. Although, in this paper it will be good enough to generate alternative paths when we sweep over a set of different weight values for w_T and w_F.

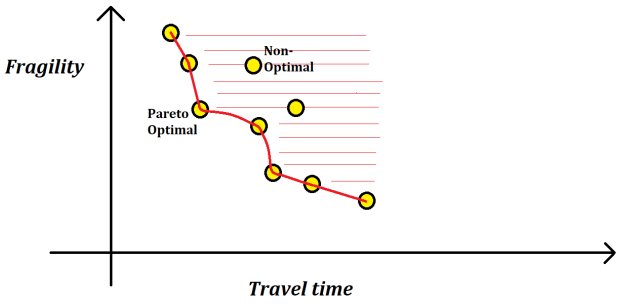

Figure 11: *Pareto optimal points displayed on a two-dimensional diagram of the two-dimensional minimization problem. The horizontal axis displays travel time and the vertical axis displays fragility. In figure 10 the Pareto front is convex while in this figure the Pareto front is concave. Pareto optimal points lying inside the concave part cannot be reached using scalarization.*

From a physics perspective, an interesting consequence of taking a superposition of objective functions is the difference in unit of measurement. In one objective we measure travel time of 600 seconds while in the other we measure a difference in area of 40%. The weights w_T and w_F will need to account for this in order to generate interesting results, simply putting $w_T = w_F = 0.5$ will not yield a solution where robustness is valued equally as high as travel time.

4. RESULTS

We present the results in the form of screenshots taken from a custom application that implements the region-based robustness algorithm. We display region-based robustness as a heatmap, and show the output of two differently weighted paths.

4.1. The Robustness Of A Region

The region-based robustness algorithm results in heat maps which visualize the robustness over a region of interest. The algorithm uses both road network as well as terrain data at a 10m resolution. Poor robustness represents areas where a disruption would cause a significant increase in travel time.

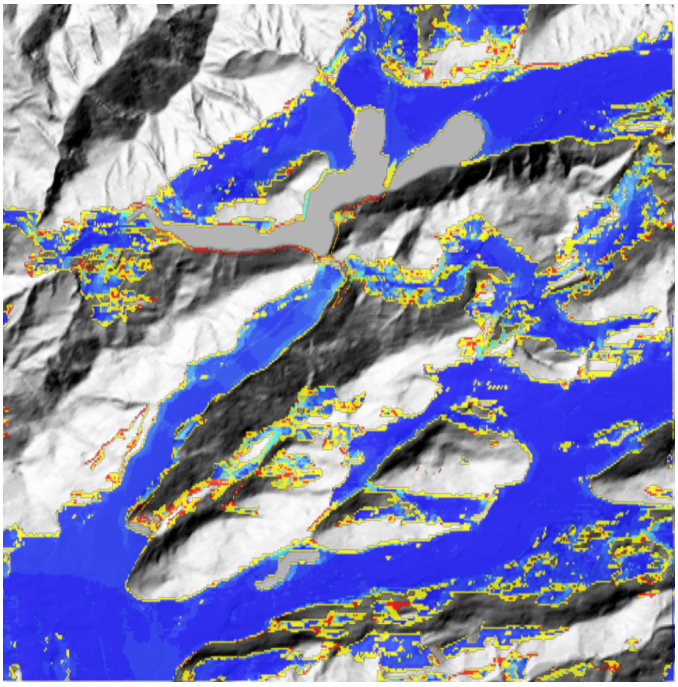

Figure 12: *A heat map visualizing the robustness over a region. The heat map's coloring ranges from blue (good robustness) to red (poor robustness).*

4.2. Robust Routes

We combine the the region-based robustness cost-map with a standard path planning algorithm using multi-objective scalarization. We study two routes: One which heavily values travel time, and one which heavily values robustness.

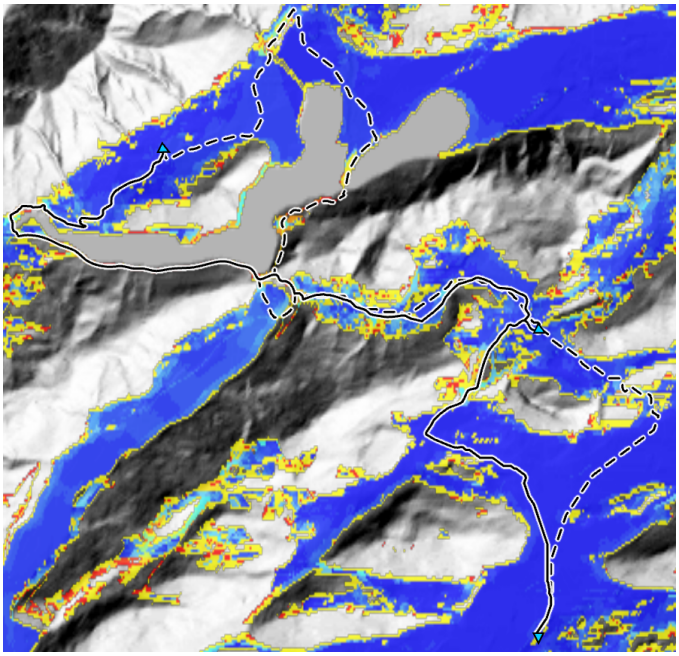

Figure 13: *Two routes visualized on a region-based robustness heat map. The solid route values travel time higher while the dashed route values robustness higher. As the solid route enters red and yellow areas (denoting poor robustness) the dashed route avoids the areas by diverting through terrain.*

5. DISCUSSION

The output robustness maps allow us to visually interpret our results. In general, robust regions are found in open, flat terrain, while less robust regions occur at valley edges, and especially on roads that may be surrounded by impassable terrain. In Figure 13, we see that when we weigh robustness heavily in our path planning algorithm, the resulting path is more likely to avoid roads in narrow valleys.

We now consider additional factors that may affect the use of the robustness cost-map in an active mission.

5.1. Impact of Size of No-Go Zones

We generated our initial robustness cost-map over a $25km^2$ region, with a $10m^2$ pixel resolution. We inserted a semicircle hazard, large enough to cover the adjacent and oncoming vertices, $75m^2$. Our robustness map is therefore a representation of changes in travel time, should a small hazard occur at that point.

The size of anticipated obstacles will vary based on the situation. We now compare two robustness maps, one generated with a small obstacle, and another generated with a large obstacle. The former could represent uneven, impassable terrain that was not detectable at the resolution of the original input data, while the latter may represent a boulder field or large enemy obstruction.

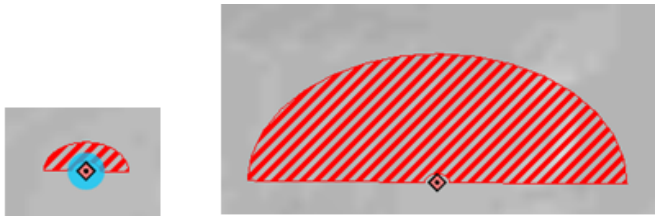

Figure 14: *A $75m^2$ and a $375m^2$ obstacle used to generate two robustness maps.*

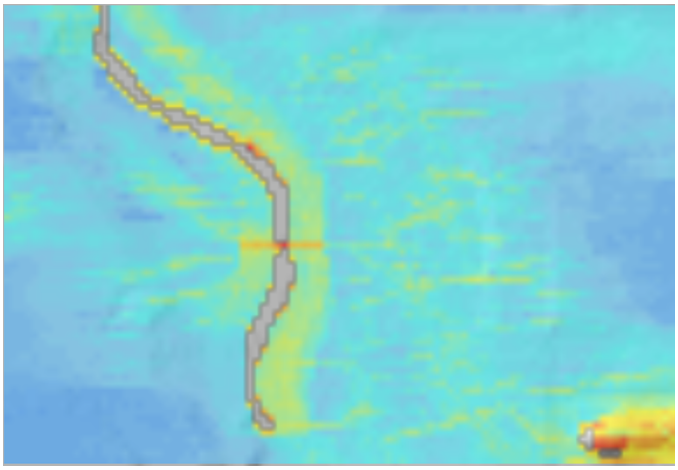

Figure 15: *A robustness map which was generated with a $75m^2$ no-go zone. Most areas show a fairly high robustness value, meaning they are minimally susceptible to disruption by small obstacles.*

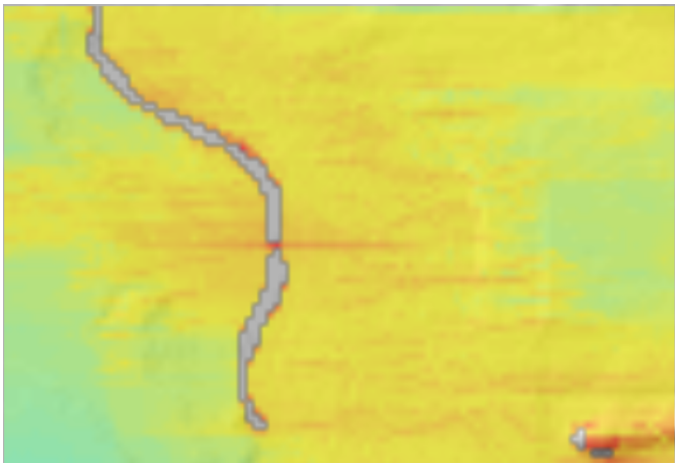

Figure 16: *A robustness map over the same area, which was generated with a 375m^2 no-go zone. A lower robustness value is calculated for the entire area, yet the same general distribution can be observed.*

The results in Figure 15 and Figure 16 demonstrate how the selected size of the obstruction affects the robustness map. Ultimately, care must be taken in the mission planning phase to select an appropriate size for the no-go zone, which represents the most likely size of the hazard that will be encountered. It is also possible to generate multiple robustness maps in advance, and select the appropriate one based on the situation encountered.

5.2 Real-Time Vehicle Speed vs Constant Speed

The robustness algorithm was altered using different types of obstacles of varies sizes to study change in the fragility cost-map. In addition to this, the vehicle model was altered to further study changes in the fragility cost-map. The two profound results were the difference between ”real-time” vehicle speed vs constant vehicle speed. Real-time speed changed more drastically depending on environment, speeds on roads were a lot higher than speeds on off-road terrain. A key observation for real-time speed is that the change in isochrone area became more drastic with respect to changing environment. Regular roads were seen as less robust compared to terrain due to the sensitive change in vehicle speed in the surroundings. If an obstacle occurred on the road, the vehicle would maneuver using the terrain, drastically decreasing the travel time. Vice versa, the vehicle could maneuver around the terrain by using the road and thereby gain vehicle speed and lose less travel time.

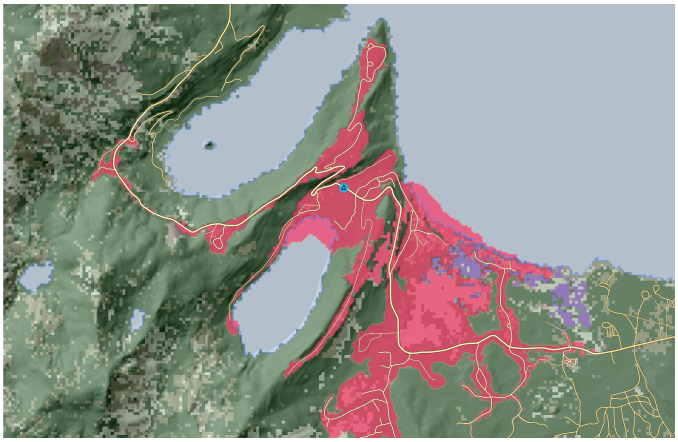

Figure 17: *An isochrone map generated from a network consisting of roads and terrain for a vehicle with real-time speed. The isochrone is generated from the blue circle as reference point. The isochrone is generated from a road as initial point.*

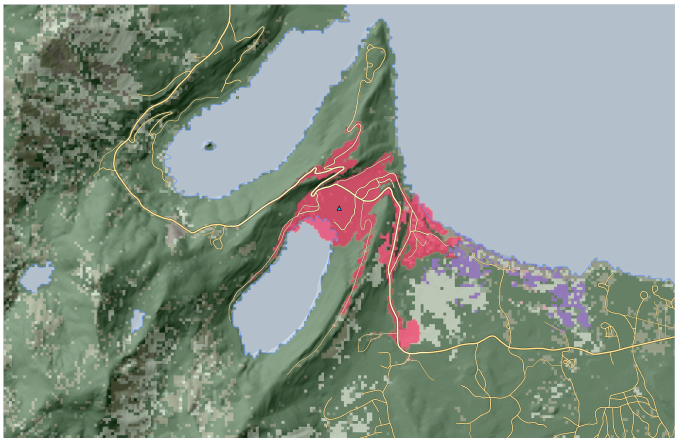

Figure 18: *An isochrone map generated from a network consisting of roads and terrain for a vehicle with real-time speed. The isochrone is generated from the blue circle as reference point. This figure is similar to figure 17 yet the isochrone area is much smaller due to the starting position being on terrain. Terrain has a lower vehicle speed.*

The real-time vehicle speed resulted in robustness maps that illustrated areas where any change would result in negative travel time, a more travel time sensitive approach. An alternative to this, the more preferred option in this paper, was the use of constant vehicle speed. By using constant vehicle speed, traveling on the road compared to in terrain were seen as an equally good alternative, no travel time penalty. By doing this, the shortest path objective function now measures distance, as travel speed is constant, and penalty time now becomes penalty distance. Penalty distance can be seen as additional distance to traverse in order to maneuver around obstacle.

Using this method, the resulting robustness maps illustrated pure maneuverability compared to maneuverability and vehicle speed sensitive areas. Both maps highlight different issues. With constant travel time the clear edge cases would be more profound, such as narrow roads (with no access to terrain) or bridges. Narrow roads and bridges are more critical areas, blocking these ones clearly limits alternatives and ability to traverse forward.

5.3. Performance

We ran the robustness map calculation on an Intel I7-8850H processor running at 2.6Ghz, using 10 m^2 resolution. A 1 km^2 region took 4 minutes to process. A 4 km^2 region took 20 minutes to process. The 25 km^2 region shown in the results took 2 hours to process. We argue that this performance time is acceptable for use in the mission planning phase. Robustness maps must be generated only once and can then be reused across the region for any path planning processes. Due to this processing time, care must be taken to generate appropriate robustness maps in advance for any vehicle type that will be used.

5.4. Inverse Relationship to Exposure

One consequence of the region-based robustness parameter is that the route will divert to more open areas. These are deemed more robust with respect to potential obstacles occurring within the region. For certain mission-critical operations this is not a viable option due to the exposure risk. Depending on the mission, the exposure factor could be a bigger risk than potential obstacles along the route.

One way to balance this out is to introduce exposure as a third objective to the path planning algorithm. Exposure is calculated using the visibility index within a region [10]. Combining this with the robustness factor would lead to a more suitable balance for certain missions, especially smaller vehicles on stealth missions.

6. CONCLUSION

Autonomous vehicles rely on path planning to guide them towards their destination. The quality of a planned path is based not only on its speed, but also on its ability to provide viable alternatives in the event of unexpected blockages. We have defined robustness as an additional parameter for path planning which describes the impact a potential disruption would have on travel time at each point in the region. By combining a calculated robustness

map with a travel time map in multi-objective optimization path planning, we are able to generate a route that balances robustness with travel times.

Generating a robustness map using the algorithm defined here cannot be done in real-time, so the operator must decide in advance the region of interest, the sizes of potential hazards we expect to encounter, and the characteristics of vehicles that will be traversing the region. Once a robustness map has been generated, however, it can be used for near real-time path planning.

When using a robustness map in combination with other environmental variables to determine the ideal path through a region, the operator must analyze the situation to determine how much to weigh each factor. Robustness tends to have a positive relationship to visual exposure, which may be unacceptable in certain scenarios. In such cases, a lower weight should be applied to the robustness map. The robustness map itself is not a replacement for subject matter expertise, and the operator's judgement is critical in determining its application.

The robustness map is a valuable tool during the mission planning phase, as it allows the operator to quickly spot possible trouble spots along the route. When used in reverse, it can also be used to determine ideal regions along an enemy route to target, especially when combined with on-the-ground information from additional sources such as UAV surveys, which may not have been included in the path planning function.

7. REFERENCES

References

[1] D. Hambling, "Mud season in Ukraine leaves Russian tanks stuck in mire," *The Guardian*, Apr. 2022. [Online]. Available: https://www.theguardian.com/news/2022/apr/12/mud-season-in-ukraine-leaves-russian-tanks-stuck-in-mire

[2] 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, Jul. 1968.

[3] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, vol. 4. San Diego, CA, USA: IEEE, May 1994, pp. 3310–3317.

[4] E. Martinson, B. Purman, and A. Dallas, "Topography dependent path planning using Deep Q-learning," in *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*. Novi, Michigan: NDIA, Aug. 2021.

[5] J. Ramsey, R. Brothers, and J. Hernandez, "Creation of a ground slope mapping methodology within the robotic technology kernel for improved navigation performance," in *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*. Novi, Michigan: NDIA, Aug. 2022.

[6] A. Lacaze, E. Mottern, and B. Brilhart, "Off-road autonomous mobility," in *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*. Novi, Michigan: NDIA, Aug. 2019.

[7] W. T. Tutte, *Graph Theory*, ser. Encyclopedia of Mathematics and its Applications. Addison-Wesley Publishing Company, 1984, vol. 21.

[8] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, pp. 369–395, 2004.

[9] Z. Clawson, X. Ding, B. Englot, T. A. Frewen, W. M. Sisson, and A. Vladimirsky, "A bi-criteria path planning algorithm for robotics applications," 2017.

[10] W. R. Franklin, "Siting observers on terrain," in *Advances in Spatial Data Handling: 10th International Symposium on Spatial Data Handling*. Springer-Verlag, 2002, pp. 109 – 120.