Utah State University

DigitalCommons@USU

All Graduate Theses and Dissertations | Graduate Studies

12-2022

Design of Environment Aware Planning Heuristics for Complex Navigation Objectives

Carter D. Bailey
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

 Part of the Computer Sciences Commons, and the Robotics Commons

Recommended Citation

Bailey, Carter D., "Design of Environment Aware Planning Heuristics for Complex Navigation Objectives" (2022). *All Graduate Theses and Dissertations*. 8657.
https://digitalcommons.usu.edu/etd/8657

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.

DESIGN OF ENVIRONMENT AWARE PLANNING HEURISTICS FOR COMPLEX NAVIGATION OBJECTIVES

by

Carter D. Bailey

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Mario Y. Harper, Ph.D. Major Professor	Shuhan Yuan, Ph.D. Committee Member
Steve Petruzza, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2022

Copyright © Carter D. Bailey 2022

All Rights Reserved

ABSTRACT

Design of Environment Aware Planning Heuristics for Complex Navigation Objectives

by

Carter D. Bailey, Master of Science

Utah State University, 2022

Major Professor: Mario Y. Harper, Ph.D.
Department: Computer Science

Heuristic methods are valued in many modern AI algorithms and decision-making architectures due to their ability to drastically reduce computation time. Particularly in robotics, path planning heuristics are widely leveraged to aid in navigation and exploration. As the robotic platform explores and navigates, information about the world can and should be used to augment and update heuristic functions to guide solutions. Complex heuristics that can account for environmental factors, robot capabilities, and desired actions provide optimal results with little wasted exploration, but are computationally expensive. This thesis demonstrates results of research into simplifying heuristics that maintains the performance improvements from complicated heuristics.

The research presented is validated on two complex robotic tasks: stealth planning and energy efficient planning. The stealth heuristic was created to inform a planner and allow a ground robot to navigate unknown environments in a less visible manner. Due to the highly uncertain nature of the world (where unknown observers exist) this heuristic implemented was instrumental to enabling the first high-uncertainty stealth planner. Heuristic guidance is further explored for use in energy efficient planning, where a machine learning approach

is used to generate a heuristic measure. This thesis demonstrates effective learned heuristics that simplify convergence time and accounts for the complexities of environment. A reduction of 60% in required time for planning was realized.

(46 pages)

PUBLIC ABSTRACT

Design of Environment Aware Planning Heuristics for Complex Navigation Objectives

Carter D. Bailey

A heuristic is the simplified approximations that helps guide a planner in deducing the best way to move forward. Heuristics are valued in many modern AI algorithms and decision-making architectures due to their ability to drastically reduce computation time. Particularly in robotics, path planning heuristics are widely leveraged to aid in navigation and exploration. As the robotic platform explores and navigates, information about the world can and should be used to augment and update the heuristic to guide solutions. Complex heuristics that can account for environmental factors, robot capabilities, and desired actions provide optimal results with little wasted exploration, but are computationally expensive. This thesis demonstrates results of research into simplifying heuristics that maintains the performance improvements from complicated heuristics.

The research presented is validated on two complex robotic tasks: stealth planning and energy efficient planning. The stealth heuristic was created to inform a planner and allow a ground robot to navigate unknown environments in a less visible manner. Due to the highly uncertain nature of the world (where unknown observers exist) this heuristic implemented was instrumental to enabling the first high-uncertainty stealth planner. Heuristic guidance is further explored for use in energy efficient planning, where a machine learning approach is used to generate a heuristic measure. This thesis demonstrates effective learned heuristics that simplify convergence time and accounts for the complexities of environment. A reduction of 60% in required compute time for planning was found.

To my lovely wife and parents

ACKNOWLEDGMENTS

I would like to thank Dr. Mario Harper for all the time he spent mentoring me. Without him this thesis would have never happened.

Carter D. Bailey

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

RRT	Rapidly-Exploring Random Tree
ML-Heuristic	Machine Learned Heuristic
SBMPO	Sampling Based Model Predictive Optimization
GBM	Gradient Boosted Machine
MLP	Multi-Layer Perceptron
RMSE	Root Mean Square Error
RF	Random Forest
SCA*	Stealth Centric A*
ROS	Robot Operating System
SCARS	Stealth Centric Autonomous Robot Simulator
DBSCAN	Density-Based Spatial Clustering of Applications with Noise

CHAPTER 1

Introduction

1.1 Heuristic Algorithms

Heuristic methods are useful in guiding many forms of complex decision-making algorithms, and is one of the necessary components behind most efficient planners used in modern robotics. Heuristics operate by guiding planning search towards regions estimated to be close to an optima. Without heuristic guidance, many planning and scheduling algorithms fail to converge or expends significant amounts of time to compute a solution. In many cases, simple Heuristic measures are useful for simplifying search space while maintaining optimal guarantees.

In robotic motion planning, two common heuristic methods are the A* and RRT planners. Variations of these planners are illustrated with a brief explanation on their differences.

1.2 Motion Planning and Heuristics

The A* planner is traditionally a grid based planner that uses a breadth first search coupled with a directional heuristic to expand nodes closer towards the goal. A* is a modified breadth first search algorithm that is deterministic, and always converges to the most efficient solution irrespective of initial conditions if the Heuristic function is admissible (a conservative estimate of the optimal solution). This algorithm is relatively fast, but can scale poorly in high dimensions.

RRT based planners operate by randomly placing nodes and building connections between them. Node connections are created with each iteration of planning with more nodes spawned and connections explored. This planner is anytime in nature, being able to converge to a solution relatively quickly, however optimal guarantees are asymptotic. RRT based planners become more optimal as the compute time increases, as further explorations

examine nodes which lower the cost of the best discovered solution. Common variations of RRT-type planners smooths paths as they are typically inefficient due to randomness in node generation.

1.3 Drawbacks and Difficulties Heuristic Design

While heuristics are crucial in guiding path planners they tend to be situation specific which causes them to be brittle, unscalable, and difficult to transfer. This difficulty in making general heuristics stems largely from the criteria of optimization changing and constraints placed by the situation. The heuristic designed for simple scenarios cannot handle changes (environment, problem) and often does not help improve computation time or loses its optimal guarantee.

Creating a global heuristic that works in all environments has proven nearly intractable, and research is focused on ways to increase the performance of heuristics. Recent work uses multiple heuristics that alternate based on the scenarios and objectives being planned. This enables the heuristic to be relatively resilient to environment changes.

This thesis will discuss new heuristics created to enable more robust and generalized planning. The second chapter discusses work that creates a machine learned heuristic (ml-heuristic). This ml-heuristic was found to lower the cost of computation for a complex energy-optimal robot path planner by 60% from traditional Heuristic methods, dramatically improving convergence times. The third chapter discusses the creation of a stealth focused heuristic. This heuristic guesses locations of potential observers in an unknown map (and updates as exploration reveals more knowledge) to help ground robots reach a goal location in a stealth-like manner. This heuristic was shown to reduce time in the line of sight of unknown observers in simulation by 37%.

CHAPTER 2

Improving Real-Time Energy-Efficient Trajectory Planning Via Machine Learning

2.1 Understanding Energy Constraints in Planning

Realtime energy optimal planning is improved through a lightweight machine learning-based heuristic function. This heuristic function efficiently guides planners while being aware of analytical physics models to honor constraints. Being lightweight, these functions do not require deep learning or significant data collection to train.

Energy-efficient trajectory planning must consider additional complexities compared to standard distance-optimal or time-optimal planners, causing them to be slower in convergence [1]. Power models evaluated during planning can be cumbersome as they must consider system kinematics and dynamics. This planning modality is important to improving the operating capacity of robots by extending their active mission time and avoids unstable maneuver's (such as sharp turns and difficult terrains). Designing an efficient machine-learning based heuristic function that accounts for vehicle models and environment significantly improves the planning speed needed for online energy efficient planning.

This machine learned heuristic (ML-heuristic) is trained on total energy cost of energy optimal trajectories as computed by an optimal A*-type kinodynamic planner. Several candidate algorithms were trained for use as a heuristic function: Light GBM, Random Forest, Multi-Layer Perceptron, and a Ridge Regressor. Training data for these algorithms were generated from randomly spawned start and goal poses along with an obstacle field.

Improvement in planning time and computational complexity was compared between these machine learned heuristic functions and the original energy efficient path planner on several validation scenarios. The simplest algorithm (ridge regressor) outperformed in our key metrics: low evaluation latency (resulting in faster compute time) and accuracy, resulting in expanding 63% less nodes during planning than our baseline kinodynamic planner.

This research provides three contributions to the trajectory planning and robotics research fields. First, this paper explores various learning algorithms for real-time planning and compares how they perform on simulated maps. Second, it was found that specific machine algorithms lend itself well to estimating heuristics as they require little data. Third, the obstacle map information can be compressed into simple vectors which are beneficial features for the heuristic learning. We demonstrate work on a dynamically constrained legged robot (via simulation).

2.2 Core Concepts of Heuristic Approaches

Heuristic functions have served to simplify search problems and find solutions for potentially intractable problems. Heuristic or guidance functions underpin much of modern planning algorithms as they guide exploration to speed up convergence, most families of planners (A* and RRT*) utilize heuristics. Particularly for A* type algorithms, having an admissible heuristic guarantees the optimal solution will be found, potentially at the expense of some wasted computation [2]. Ongoing development of more efficient admissible heuristic functions have improved task and motion planning, computer security [3], and sequential logic problems [4].

Fig. 2.1: Legged system used for validating energy efficient navigation tasks. Unstructured terrain navigation poses unique constraints and requires rapid planning as new information becomes available.

Efficient heuristic functions must be able to guide the search as close as possible to the optimal path, without incurring significant computational overhead. Particularly when planning for dynamically constrained legged vehicles (Figure 1) online, a search must complete and return an actionable plan before footfall. In practice, analytically accounting for a robot's motion constraints, environment, obstacles, and payload/power constraints is difficult to construct and computationally expensive to evaluate.

Well-engineered heuristic functions that integrate dynamic models result in reduced convergence time, while adding valuable insight into robot performance. Heuristic designers use various techniques such as caching data, genetic algorithms, or self-education (evaluating feedback) [5–8], to improve performance and mediate these issues. Other researchers have shown that implementing a machine learned heuristic into their program increases the speed of convergence [9] and explored robot construction and design through the aid of a heuristic [10].

While engineering better heuristic functions are exceptionally useful in simplifying computational loads, there are drawbacks to its use. Primarily, well-engineered heuristics are not portable as they are often tailored for a specific problem and use case. Second, they can be challenging to design and implement as models need to be derived and tuned, particularly for energy-based planning. [11, 12]

Recent frameworks have combined heuristic functions with neural networks or machine learning tools to augment search functions and alleviate some of the drawbacks that heuristic functions usually have [13, 14]. Other advances in learned-heuristic approaches take advantage of iterative learning, effectively reducing the complexity of high-state space planning [7], guiding A* graph search with neural networks [15], and explicitly modeling heuristic functions based on deep learning [16].

As various techniques to mediate heuristics' have improved, popular heuristic-based search algorithms, such as A* have increased its convergence speed. Improved heuristic design allows a heuristic to reflect the true cost-to-goal accurately [17] and with reduced computation time. Collecting data for training newer machine-learning based approaches

can however be time consuming and suffer from the simulation-reality gap. This is particularly true in real-world systems that do not possess high-fidelity simulations.

We are interested in applying learned heuristic methods to the problem of energy-efficient navigation with vehicles that are dynamically constrained. This method of planning necessitates understanding of vehicle dynamics (power models) where approximations of traversal costs are difficult to quantify due to the kinodynamic complexity. Data collection on real energy consumption is collected via simulation, and optimal paths to generate training data are done through a sampling-based A* type algorithm.

2.3 Machine Learning Solutions

2.3.1 Gaining Data From SBMPO

We choose to employ a kinodynamic planner called Sampling Based Model Predictive Optimization (SBMPO) [11] to collect the training data. SBMPO was chosen for its ease of use with legged and dexterous mobility robots, and its demonstrated energy efficient planning capabilities on both legged and wheeled ground robots.

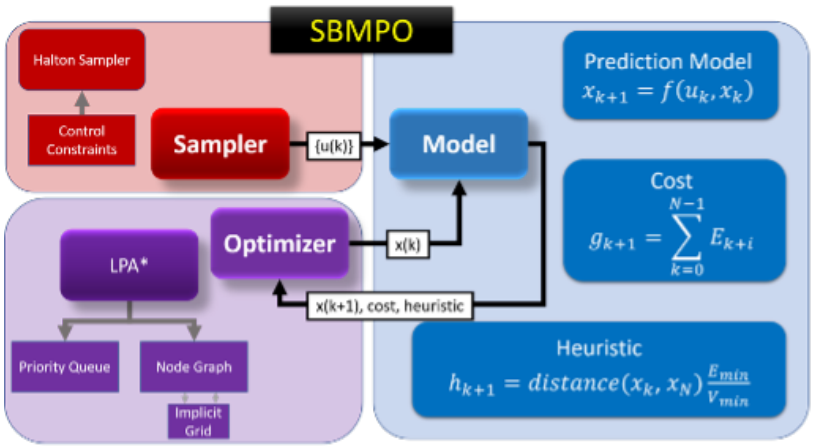

Fig. 2.2: SBMPO core components. SBMPO is comprised of three major components: A sampler that determines what control input is explored, model-specific functions, and an optimizer based on a A*-type algorithm.

SBMPO has three primary components (shown in Figure 2) and works with a variety

Table 2.1: Sample of data from the training set

Start X	Start Y	Start Angle	Goal X	Goal Y	Goal Angle	Diff Angle	Est. Dist	R1	R2	R3	R4	R5	R6	R7	R8
9.61	5.49	1.02	1.63	7.46	172.60	0.33	8.22	1	1	1	2	1	1	2	0
9.07	7.90	1.59	0.96	4.11	82.28	0.25	8.96	1	0	0	2	0	2	1	0
1.03	9.29	4.04	7.99	8.87	156.19	0.06	6.97	1	2	1	1	0	2	0	0
12.84	4.93	4.53	6.00	10.68	187.44	0.33	8.93	0	1	1	2	0	0	3	1
10.30	14.61	1.17	0.74	4.51	16.18	0.24	13.91	0	1	0	0	1	3	2	0

of dynamic holonomic or non-holonomic robot models, either linear or nonlinear. These models may be derived from first principles or learned, and can also be non-invertible as SBMPO samples the input (i.e., control) space directly, alleviating the cumbersome need for local connection planners or inverse kinematics.

A core tenant of SBMPO is flexibility in optimality criteria such as minimum distance, minimum time, or minimum energy; the latter relies on the use of power models derived from system dynamics coupled with electrical models of the actuators. Generated trajectories do not rely on smoothing and post processing stages to ensure that system dynamics are followed as control inputs are considered directly. This algorithm shares the same optimality and completeness guarantees of A*, with a heuristic function enabling rapid computational convergence where it is both informative and admissible.

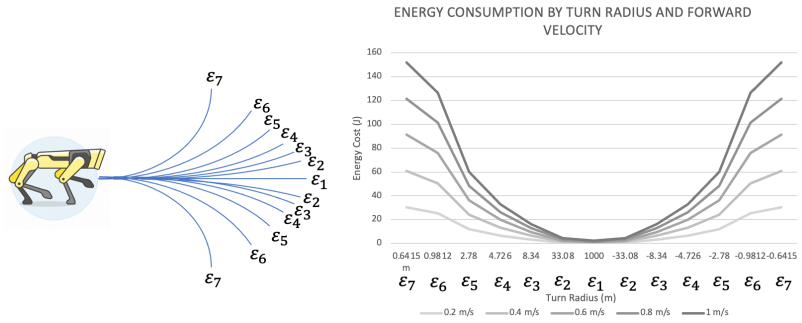

Fig. 2.3: Energy profile of simulated legged robot. As velocity increases and turns become tighter, the expended energy per step is several times more than for straight-line motion. Increased energy consumption stems from tight turns requiring increased motor torque from all 12 leg motors, particularly the hip which is not generally actuated in forward motion.

Energy models of this type of robot follow a near quadratic relationship [11] between

turn radius and power as shown in Figure 3. Sampled turn radii begin with a straight-line motion (turn radius of 1000m is used to approximate) and extend to tight turns of 0.6415m as the safest extreme maneuver. Power consumption increases dramatically with increased turn radii, as motors must respond with higher torque to complete maneuvers.

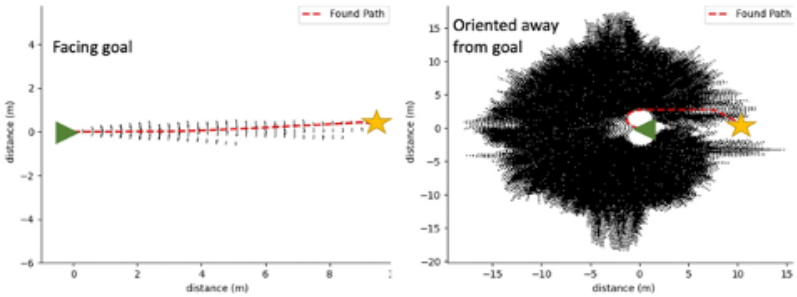

Fig. 2.4: On the straight facing path, the cost is 1198.53 J. On the path where the heading angle is oriented away from the goal the cost is 6365.66 J.

Consideration of energy optimality also complicates heuristic function design. As shown in Figure 4, a simple near-straight line path can be approximated well by an admissible heuristic, resulting in rapid convergence with little exploration burden. For many non-holonomic vehicles, the orientation matters tremendously as turns induce higher energy consumption, thus, orienting 180 degrees away from the goal pose can cause expended energy to increase by a factor of 6 and increase computation time by a factor of 1000.

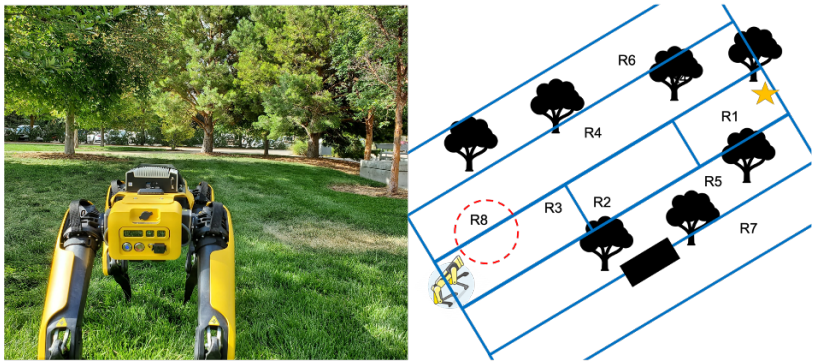

Fig. 2.5: Example of obstacle regions used for the machine learning to know where obstacles are. Onboard lidar can sufficiently cover a 360 degrees field of view.

While using a convolutional neural network to take advantage of state-of-the-art computer vision aids in processing the world (particularly of lidar points), this increases the burden of online training and evaluation, and fails to capture distance relationships relative to the robot accurately [18, 19]. We opted for a simpler, novel feature reduction technique which accounts for all obstacles as 1-meter wide circular objects in specified regions respective to the robot and goal position.

Eight regions are defined by different sections of a robot's field of view (Figure 2.5). Regions 1-3 corresponds to the corridor directly between the robot and the goal, region 4-7 corresponds to a wider area around the central corridor, and region 8 represents a circle of minimum turn radius directly in front of the robot. All these regions (excepting region 8) are 1 meter wide and expand to form a rectangle of finite length.

Every region reports its obstacle count which is used by the machine learning rather than the explicit coordinates and size of an obstacle. Thus, obstacle count in the corridors defined by R1-R3, being directly between the robot and the goal, has a higher impact to the machine learning. Obstacle count in R4-R7 respectively has greater impact in predicting energy costs if the initial heading angle is oriented towards them. Another area found to inform the machine learning was R8 as immediate obstacles will necessitate a higher energy maneuver to avoid. Thus, rather than using a machine learning processing technique, the obstacle map was simplified into a 1D vector of obstacle counts by region. An example of this simplification can be seen in Table 2.1 in columns R1-R8.

2.3.2 Training With Data From SBMPO and Multiple Maps

SBMPO was run on many randomized scenarios to obtain training data for real energy requirements to complete a navigation task. To collect this data 40,000 individual start and goal poses were randomly generated for 13 distinct maps, resulting in a total dataset pool of 520,000 optimal trajectories. Optimality was guaranteed during data collection by using the base admissible energy heuristic used in SBMPO.

Training features were comprised of start pose (robot x position, y position, and orientation), goal pose, the angular difference between start orientation and goal (Diff Angle),

Table 2.2: Accuracy scores of the ML-heuristic integrated SBMPO

	Ridge Regressor	Random Forest	Light GBM	MLP
Score	50.69%	84.85%	46.25%	87.28%

Euclidean distance to goal (Est. Dist), and obstacle count by region vector [R1-R8]. A sample of a typical feature set is shown in Table 1.

We limited the suite of machine learning algorithms to those with low latency (speed of evaluation) and/or high human interpretability, these are critical factors in autonomous systems. Specifically, we compare the Light GBM (Gradient Boosted Machine), Random Forest, Ridge Regressor, and the Multi-Layer Perceptron (MLP) [20–23] in both training accuracy, computation (based on nodes expanded), and resulting energy of optimized trajectory.

The ML-heuristics parameters were tuned based on RMSE accuracy. Due to the differences in ML algorithms employed, the number and type of parameters are different, we list these as follows rather than tabulating due to formatting concerns. Ridge Regressor: alpha=5; Random Forest: estimators=100; Light GBM: leaves=31, estimators=20, learning rate=0.05; MLP: hidden layer sizes=(30,30), random state=1, max iteration=300

Accuracy of the models (Table 2.2) ranged from 50.7% to 87.3%. The new heuristic functions were integrated into SBMPO and compared against the baseline heuristic function they were trained from. The ML-heuristics were run on new scenarios with randomized start and goal locations to test the robustness of the new ML-heuristic SBMPO compared to SBMPO.

2.4 Improvements from Machine Learning Integration

Resulting explorations and computed trajectories are shown in Figures 2.6, 2.7, the ridge regressor computed quicker and had the lowest exploration cost of all tested ML-

heuristics. Even though the ridge regressor computed the quickest it converged to a suboptimal trajectory which increased traversal costs by 5%.

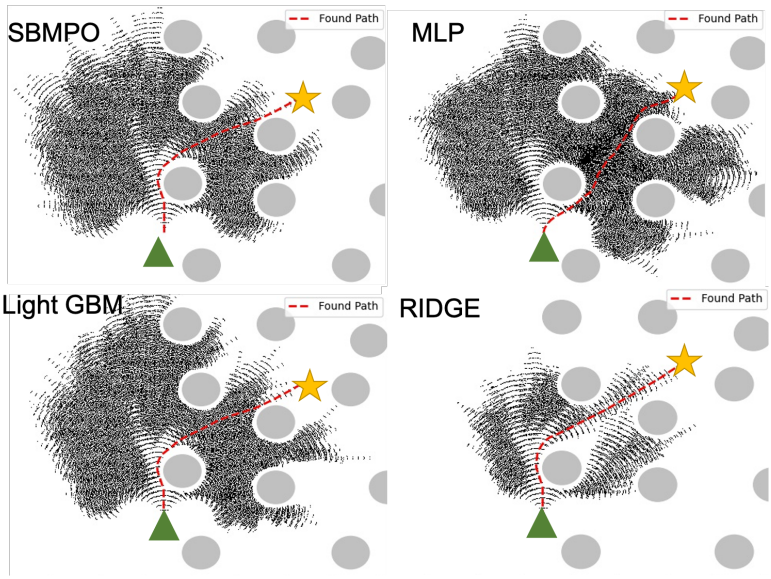

Fig. 2.6: Comparison of exploration space. Node expansion for SBMPO and the three highest scoring models showing the path chosen and the node expansion to find the path.

Results of integrated tests (Figure 2.6) shows that the ridge regressor heuristic was able to compute a trajectory to the goal with only 37% of explored nodes of SBMPO. The convergence took 36% of the baseline time and came at a slightly increased energy cost of 4.5%. SBMPO's average time of convergence (on a desktop computer) for a long-distance plan in a complex obstacle field was 5.1 seconds. Utilizing a ridge regressor heuristic function decreases convergence to 1.7 seconds on average. This improvement in convergence speed allows for more replanning as a robot is traversing it's environment, key for navigating in dynamic scenes.

Even though the ridge regressor had the lowest learning rate compared to all the machine learned models it outperformed all the other models in our three key metrics. We believe this is largely due to difficulty in overfitting the ridge regressor as well as its lack of capacity to learn more than general trends, allowing it to generalize better than its peer algorithms.

The simplifications engineered to reduce data complexity caused some issues for the other ML-heuristics. Both the neural network (MLP) and forest based (RF, LGBM) machine learning algorithms fared much worse than expected in validation runs. These validations were an entirely new obstacle map, and it was immediately apparent that the simplifications were not sufficient to represent the real complexities of unknown environments well. These larger capacity algorithms will likely do better with less data simplification.

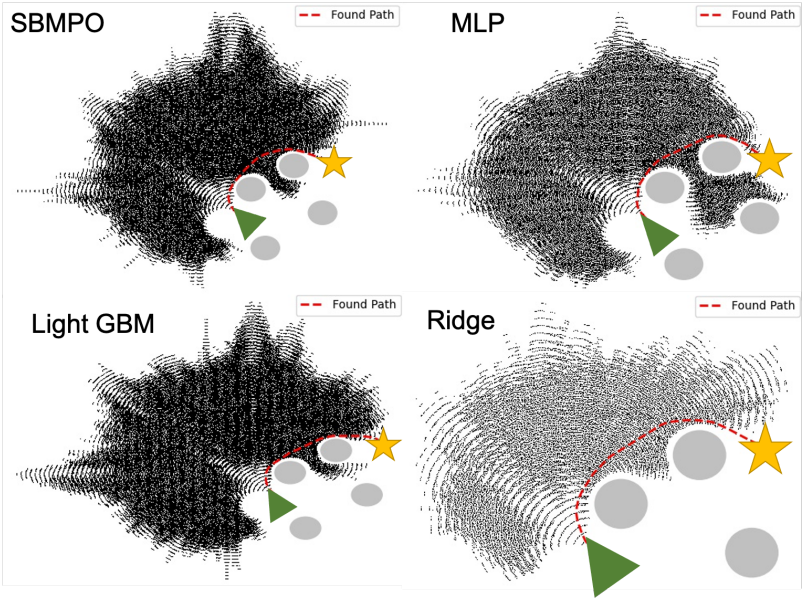

Fig. 2.7: Another example of the ML-heuristics expanding less than SBMPO.

The ridge regressor ML-heuristic expanded dramatically less nodes than SBMPO overall (Table III). Light GBM and MLP also generally expanded less nodes than SBMPO while maintaining a very similar cost to goal as SBMPO. All the ML-heuristics outperformed SBMPO in computing time as it alleviated some computational complexity of the standard model, but the ridge regressor's consistency with outperforming SBMPO in all regards allowed it to be the best ML-heuristic overall, as visualized in Figure 2.8.

We note that at times, MLP would expand nodes very quickly to find the goal location beating all other ML-heuristics. This leads us to believe that if given a more significant amount of data MLP would be able to outperform the ridge regressor. Our research being

Table 2.3: Comparison of time, cost, and node expansion

	Percentage Nodes	Percentage Cost	Percentage Time
SBMPO	100	100	100
RF	119.357	117.946	72.766
MLP	111.321	128.754	56.770
RIDGE	36.840	104.562	36.393
Light GBM	99.488	105.434	92.434

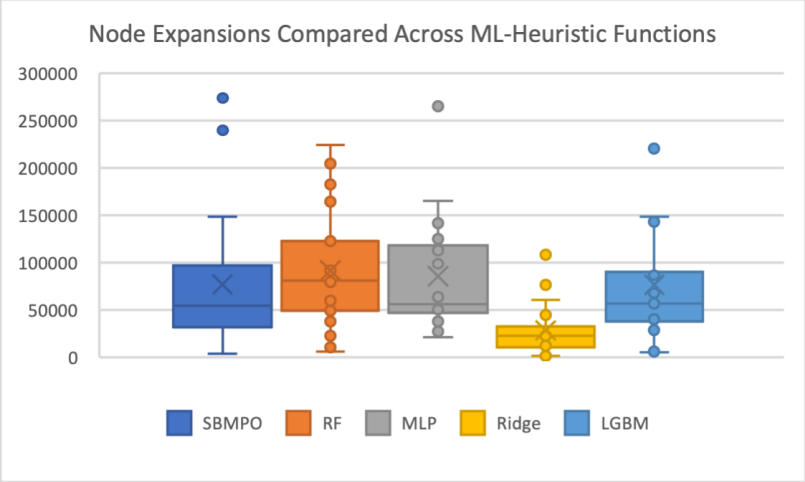

Fig. 2.8: Node expansions compared across different ML heuristic functions. Average expansions are appreciably lower for the ridge regressor. Other algorithms span a wider range of expansion costs, often similar in performance to SBMPO.

focused on creating a ML-heuristic with a low needs may impact the usefulness of MLP as it typically requires more data to train.

Additionally, we wish to state that this method no longer guarantees the optimal solution. As can be seen in the trajectory Figures 2.6, 2.7 and comparison Table III, the heuristic is no longer admissible and costs generally increase by a small percentage. Users of this method must consider if the computational gains in convergence time are worth the trade-off in optimality.

The comparison of the ML-heuristics (Figure 2.8) explored in this research shows the improvement in performance from a simple ridge regressor which is consistently lower in expansions than any other method. Other algorithms perform similarly to SBMPO and perhaps learned too closely to emulate the original algorithm.

2.5 Key Insights

This research demonstrates the value of machine learned heuristics in improving performance of planning tasks. This is particularly true in cases where non-trivial optimization is required, as in the case of energy efficient planning.

Within the span of machine learning approaches, simple algorithms like the ridge regressor is sufficient to improve the computation time of finding a solution. This is primarily due to the significant reduction of nodes expanded during a search, while keeping evaluation costs low. Our approach improves on the basic admissible heuristic by reducing computation time to 34% for an average planning task. This method particularly is useful for complex obstacle fields and kinodynamically constrained vehicles.

The approach described in this research is general and applicable for any heuristics-based planner. It is particularly beneficial in situations requiring reduced planning time such as navigation in dynamic scenes which require frequent replanning. In particular, this research is crucial for high-speed legged systems which moves in a discrete nature (due to actions limited to brief moments of footfall) by allowing optimal trajectories to be computed before next footfall.

Acknowledgment

We would like to thank the NSF ASPIRE ERC (1941524) for help in supporting parts of this research.

REFERENCES

[1] S. Liu and D. Sun, "Minimizing energy consumption of wheeled mobile robots via optimal motion planning," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 2, pp. 401–411, 2013.

[2] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[3] D. J. Sanok Jr, "An analysis of how antivirus methodologies are utilized in protecting computers from malicious code," in *Proceedings of the 2nd annual conference on Information security curriculum development*, 2005, pp. 142–144.

[4] R. E. Korf, "Recent progress in the design and analysis of admissible heuristic functions," in *International Symposium on Abstraction, Reformulation, and Approximation*. Springer, 2000, pp. 45–55.

[5] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[6] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2872–2879.

[7] S. J. Arfaee, S. Zilles, and R. C. Holte, "Learning heuristic functions for large state spaces," *Artificial Intelligence*, vol. 175, no. 16-17, pp. 2075–2098, 2011.

[8] Y. F. Yiu, J. Du, and R. Mahapatra, "Evolutionary heuristic a* search: Heuristic function optimization via genetic algorithm," in *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, 2018, pp. 25–32.

[9] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.

[10] S. Ha, S. Coros, A. Alspach, J. M. Bern, J. Kim, and K. Yamane, "Computational design of robotic devices from high-level motion specifications," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1240–1251, 2018.

[11] M. Y. Harper, J. V. Nicholson, E. G. Collins, J. Pusey, and J. E. Clark, "Energy efficient navigation for running legged robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6770–6776.

[12] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Energy-efficient motion planning for mobile robots," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 4344–4349.

[13] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning a-star," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[14] B. M. Reese and E. G. Collins Jr, "A graph search and neural network approach to adaptive nonlinear model predictive control," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 250–268, 2016.

[15] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki, "Path planning using neural a* search," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 029–12 039.

[16] T. Takahashi, H. Sun, D. Tian, and Y. Wang, "Learning heuristic functions for mobile robot path planning using deep neural networks," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, 2019, pp. 764–772.

[17] A. Koubaa, H. Bennaceur, I. Chaari, S. Trigui, A. Ammar, M.-F. Sriti, M. Alajlan, O. Cheikhrouhou, and Y. Javed, "Background on artificial intelligence algorithms for

global path planning," in *Robot Path Planning and Cooperation*. Springer, 2018, pp. 13–51.

[18] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[19] D. C. Duro, S. E. Franklin, and M. G. Dubé, "A comparison of pixel-based and object-based image analysis with selected machine learning algorithms for the classification of agricultural landscapes using spot-5 hrg imagery," *Remote sensing of environment*, vol. 118, pp. 259–272, 2012.

[20] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[21] H. Taud and J. Mas, "Multilayer perceptron (mlp)," in *Geomatic Approaches for Modeling Land Change Scenarios*. Springer, 2018, pp. 451–455.

[22] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[23] A. E. Hoerl, R. W. Kannard, and K. F. Baldwin, "Ridge regression: some simulations," *Communications in Statistics-Theory and Methods*, vol. 4, no. 2, pp. 105–123, 1975.

CHAPTER 3

Stealth Centric A* (SCA*): Bio-Inspired Navigation for Ground Robots

3.1 Stealth In Robotics

Intelligent navigation is a critical aspect in advancing the autonomy of robots, particularly ground robots that operate in unstructured environments. Ground robot navigation and guidance have increasingly adopted strategies inspired by nature, such as the navigation of bats, ants, bees, and large foragers [1,2]. Although the robotics community has benefited greatly from studying biological systems to help guide navigation strategies, particularly for swarm-based systems, additional insights can be gained to develop stealth-based ground motion.

Animals, particularly those predominantly preyed upon, have demonstrated an enhanced ability to navigate through complex environments, pursue resources, and stay relatively safe. An example is a rabbit foraging, where the rabbit must navigate many unknown dangers during exploration to obtain food while avoiding aerial or land-based predators. To protect itself, the rabbit minimizes the time spent in open areas with a clear line of sight to potential predators and maximizes the time spent near foliage and large obstacles capable of providing cover [3, 4]. This project aims to emulate these characteristics inside of a stealth-aware planner for a ground robot.

Designing and testing a planner that avoids regions of observer line of sight was accomplished through a new simulation software built in Unity. This was chosen due to the efficient ray casting capabilities built into the Unity engine. The environment was connected with a novel stealth-focused planner and tested on multiple environment configurations (Figure 3.1). In the simulation software, random obstacles and observers are generated, placed throughout the map, and a start and goal position is defined. The ground robot used in the simulation is equipped with a Lidar, camera, and pose data, with easy portability to ROS

for efficient sim-to-hardware transfer.

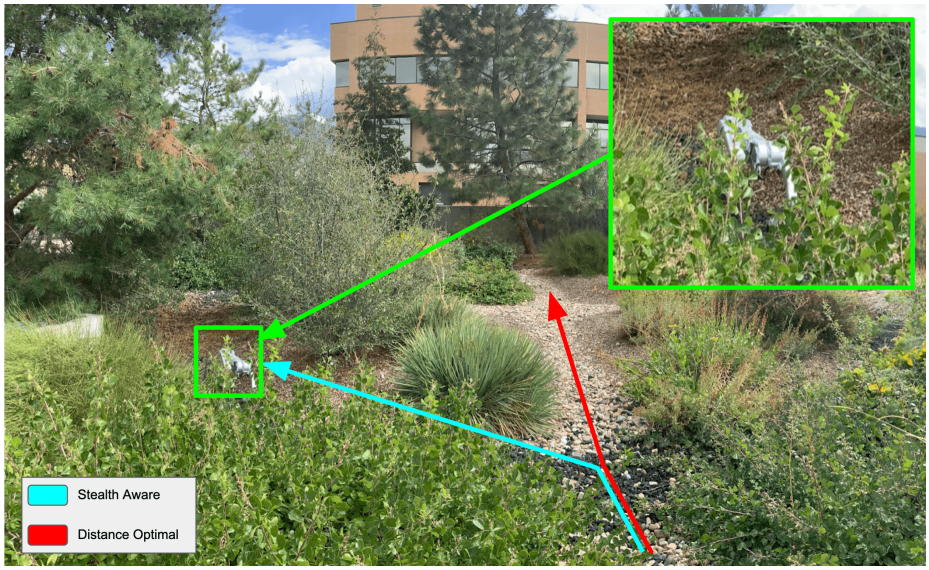

Fig. 3.1: An example of a biologically inspired stealth aware path compared to a distance optimal path. The robot in the image is hard to see due to the additional foliage providing cover. Prey favor paths with greater occlusion at the cost of limited visibility and slower map-building.

An A-Star based planner was implemented with multiple heuristic functions designed to emulate a biological systems preference for high-cover regions. The first heuristic function provides a baseline using a standard minimum-distance measure. The second heuristic function balances the distance measure with a line of sight metric. This line of sight metric randomly generates observers from the robots field of view and penalizes regions where the robot is visible. These observers exist only as guesses in the planner as the robot is not certain if and where any observers might be. The third heuristic function attempts to balance minimum-distance with proximity to the nearest obstacle, causing the robot to move near obstacles. These heuristic functions are assessed by comparing the distance traveled while being observed by randomly placed sensors.

Simulation experiments were conducted in multiple environment and observer configurations. The results show that the line-of-sight heuristic function significantly reduced the total visibility to observers compared to both other methods. Performance improved by

nearly 10 meters difference in traversal under detection.

3.2 Core Concepts of Stealth

Stealth-aware planning has several applications, many of which stem from defense-driven needs (surveillance, pursuit and interception, stealthy navigation, and target following), but they also serve purposes in civilian situations (law enforcement, animal tracking, and efficient autonomous delivery). Significant attention has been directed to stealth navigation for Unmanned Aerial Vehicles (UAVs) [5–8] which avoids known observer installations or minimizes exposure to ground-based or aerial systems.

Avoiding capture or visibility on ground robots is less developed. Work has been demonstrated on ground robots to monitor the behavior of an intruder on a predetermined map. The ground system follows a hostile actor while avoiding detection with the objective of maintaining surveillance. [9]. Additionally, research has been conducted into situations where the map and observers are known to the planner, and an optimal route that avoids detection is constructed [10]

Additional research has been conducted on scenarios where observers are known, but a map is not made available to a planner on run-time [11]. Similarly, extended work on a multi-agent system detected intruders without a map when starting a search. This planner does not construct a map internally, only navigating given its immediate environment [12] and is able to locate hidden intruders relatively quickly, however, the agents themselves are not concerned with stealth. Several other research projects have set out to accomplish stealth-aware motion [13], but none explicitly attempts to navigate in a stealthy manner in highly uncertain environments with unknown hostile actors.

This research focuses on aspects of these prior works, with added complexity as neither map nor observer information is available. The robot is also unaware of when its motion is within the view of an observer. The construction of a predicted danger map or biasing towards obstacles is built with the robot's guesses as to where an observer might be (if an observer exists). This is to emulate real-world situations in unstructured environments as autonomous systems never truly know where potential danger may be.

3.3 Stealth Centric A* Design

Stealth Centric A-Star (SCA*) is based on the A-Star planner [14] with modifications to return a stealth-aware route while dynamically updating and replanning based on new environmental data from sensor feeds. A priority queue is implemented that sorts based on stealth heuristics at each node.

To provide optimal stealth functionality, two stealth-aware heuristic functions were implemented in the SCA*. These two methods (obstacle proximity and line of sight) are compared against each other and a distance-minimizing standard A* algorithm to benchmark performance.

Obstacle Proximity Heuristic

The obstacle proximity heuristic function calculates the distance between the nearest obstacle and the expanded node. This calculated distance is passed into a step function which multiplies by a scalar based on obstacle proximity as seen in equation 3.1. This step function induces exploration near obstacles by shifting the priority queue to favor regions of close obstacle proximity. This heuristic function causes the robot to stay out of the line of sight of many observers by effectively maximizing time near occlusions. While this method effectively reduces the distance traveled under observation, it does not directly consider stealth.

$$F(x) = \begin{cases} distance * 0.1 & \text{if } x < 0.5 \\ distance * 0.3 & \text{if } x < 0.75 \\ distance * 0.5 & \text{if } x < 1.0 \\ distance * 0.7 & \text{if } x < 1.25 \\ distance * 0.9 & \text{if } x < 1.5 \\ distance & \text{if } x \geq 1.5 \end{cases} \tag{3.1}$$

Line of Sight Heuristic

The line of sight based heuristic similarly does not receive a map or true observer

positions. This method generates virtual observers throughout the global map based on its observable space. We note that this observer generation can potentially place them in an obstacle that has not yet been discovered by the robot's sensors. The observer generator segments the world into squares of $3m$ and places an observer in the center if the location is not occupied. After generating all observers, a line of sight collision detection based on ray casting checks if a node is visible to an observer.

The priority queue updates nodes (based on equation 3.2) where distance, observer count, and a constant scalar (LosConstant) are utilized. By modifying the LosConstant, the simulated observers' influence on the overall cost can be adjusted. This method directly considers stealth aspects, but remains unaware of if and where any real observers might be. This method can account for the influence of multiple obstacles acting as potential points of occlusion.

$$F(x) = distance + (LosConstant * ObserverCount) \tag{3.2}$$

3.4 Simulation Tests

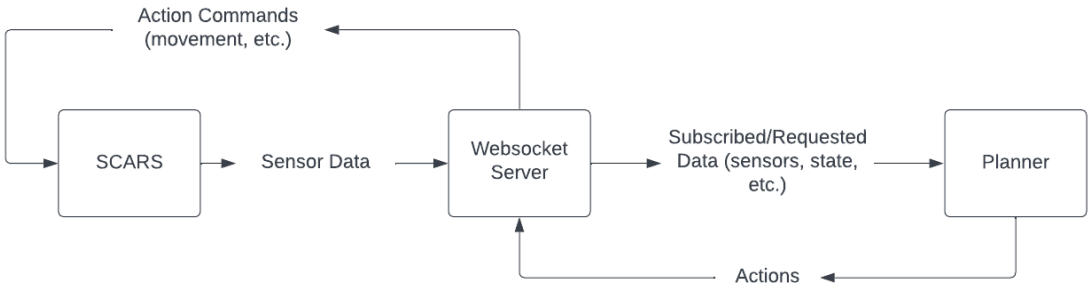

Fig. 3.2: System architecture of the SCARS framework. This communication protocol allows simple realtime data transfer between the SCARS and the planner.

There are excellent 3D software simulators for ground robotics that were compared prior to the construction of the Stealth Centric Autonomous Robot Simulator (SCARS). Of the notable open-source simulators, Webots was an option considered but did not have

the scripting and ray cast capabilities required for quick calculation of observer visibility. Mujoco has necessary ray cast capabilities, but lacks an editable environment in runtime, requiring XML edits that would not allow for the dynamic placement of observers. Isaac Sim has detailed physics simulations for robotics, but does not include the desired multi-object ray cast ability. Gazebo provides great integration with ROS, but lacked environment editor abilities possible with Unity. The SCARS framework was ultimately built on the Unity platform as it provided all the scripting and ray casting abilities we desired, along with methods to generate observers in runtime. Development time is also lower as new testing environments could be procedurally generated. Additionally, Unity is cross-platform and easier to distribute as software, allowing easier extension for continued research.

The SCARS and the planner communicate via a centralized websocket server which allows for asynchronous communication. The websocket is used to update the planner and SCARS through realtime subscription to specific data feeds, e.g. Lidar (See Figure 3.2). The current state of the robot is published by SCARS to the planner, and visualizations are constantly updated to reflect movement and action commands. This architecture allows our planner to be easily ported to a physical robot as we mirror ROS-style commands, all sensor feeds are similarly designed to be ROS compatible with topics built in line with hardware integration in mind.

Sensor Simulation

Within SCARS, a realistic Lidar scanner was built based on emulating the ROS LaserScan message. This virtual Lidar uses a ray scan to check for obstacles, similar to a physical sensor. On each frame, the simulation publishes collected data to update the central server with its current sensor state. On each planning cycle, the current Lidar feed and location data are used to generate an obstacle point map. The Lidar locations were translated into global space for the robot and SCARS to visualize.

Camera feeds are published as a base-64 string easily parsed into a ROS camera message. The base Unity camera object was used to generate this realtime feed. While this camera object was created, for the purposes of this research, the camera object is not

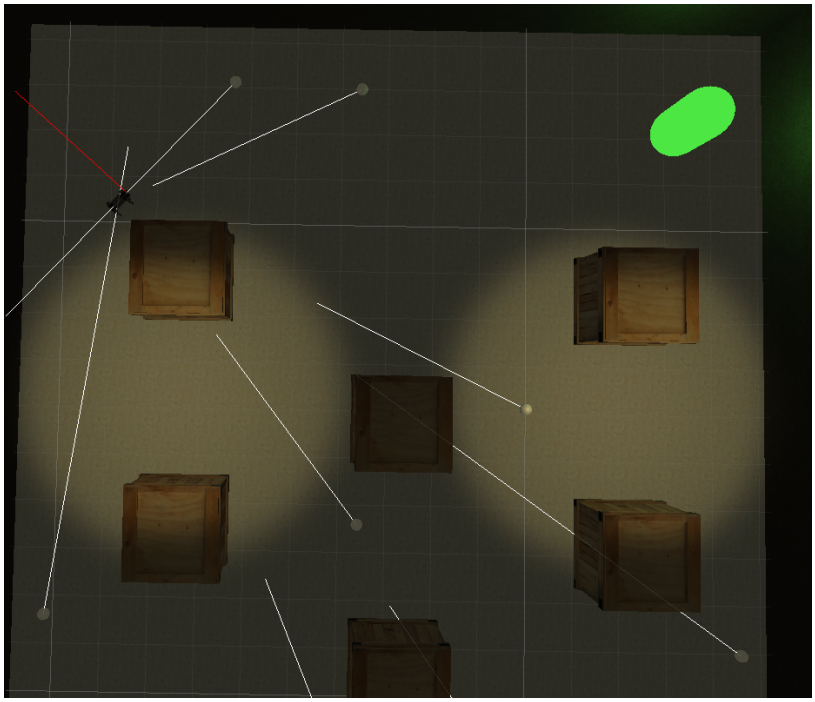

Fig. 3.3: Example of the SCARS ray cast being used for realtime calculation of observer visibility during a simulation run. The robot (denoted with a red line) is currently visible to two observers with an additional six unable to see the robot.

utilized. We note that all of the sensor objects are provided as a usable Unity asset.

Map Generation and Collision Detection

As the planner has no a priori knowledge of the map, sensor data from the virtual Lidar is processed to update the world representation during navigation. Lidar data does not provide information on the depth of an obstacle, leaving the full determination of obstacle geometry in doubt.

To remedy this issue, a Density-Based spatial clustering of applications with noise (DBSCAN) was used. DBSCAN leverages relative proximal locations to discover clusters of arbitrary shapes efficiently and effectively. [15] [16] This clustering algorithm was utilized on Lidar points to create obstacle regions. This greatly simplified the simulation and reduced the computational load of collision detection.

The DBSCAN requires two parameters, an ϵ and the threshold (minimum number of

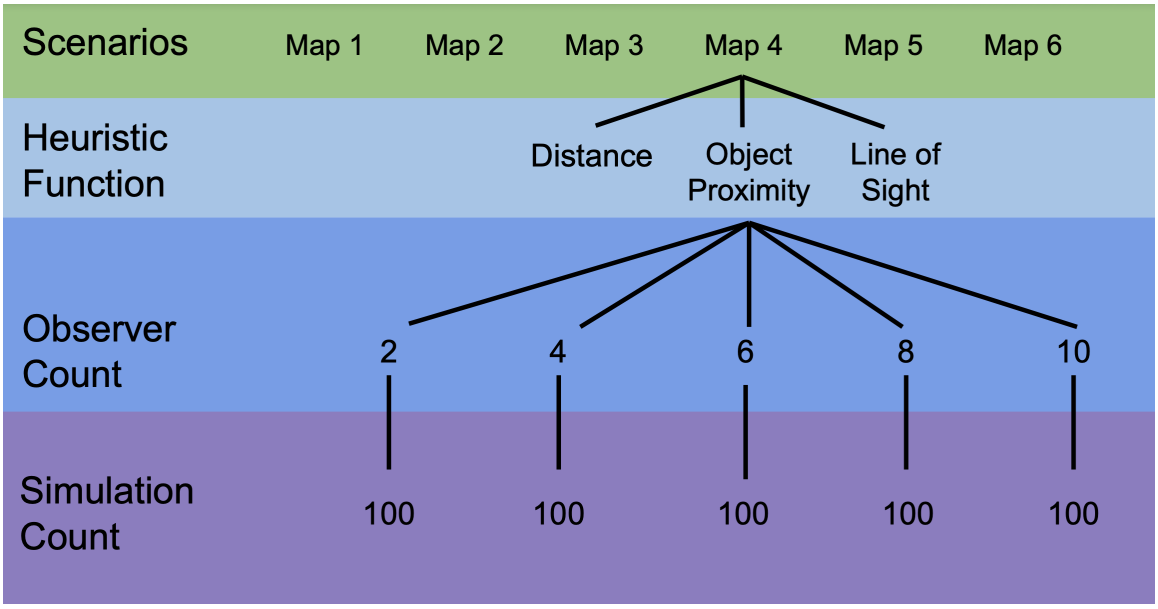

Fig. 3.4: Tree displaying the simulation methods. A total of 9000 simulations are run with each map scenario testing all three heuristic methods. Each heuristic method tests a random placement of n observers (2,4,6,8,10). Only a single branch is fully extended in this illustration.

points needed to be considered as a cluster). The ϵ was defined as 1.25 meters as this allowed the larger obstacles to be reconciled as one single object. The minimum threshold value chosen was 2, to limit the stored Lidar points to every .1 meters saving memory and improving computation speed. This collection of points is then created into an obstacle with a collision box represented by a polygon, which enables rapid collision detection.

Collision detection between nodes is determined using the Separated Axis Theorem. This method projects obstacles down to a new axis, if the projections do not overlap on any projected axis, the obstacles do not overlap [17]. Additionally, the polygon representation of obstacles is two-dimensional, decreasing the algorithmic complexity.

Motion Prediction

Upon receiving a location from SCARS, the planner maps a route from the current position to the goal. However, the transmitted location and actual position has differences due to the constant motion of the robot. The simulated robot does not stop for replanning

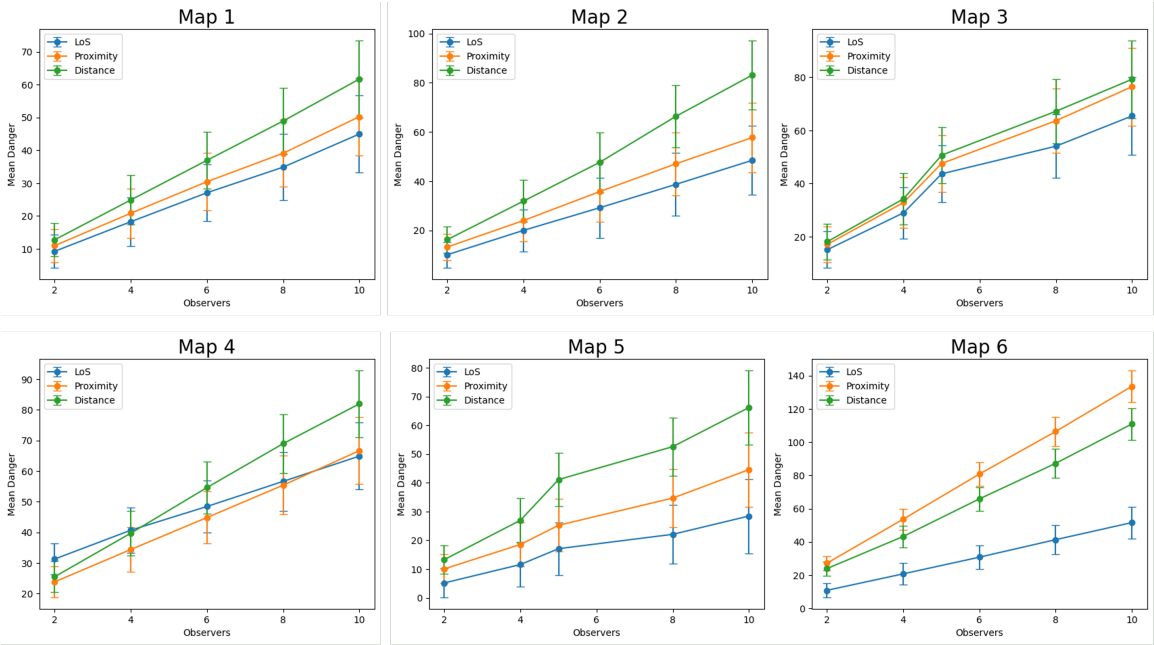

Fig. 3.5: Comparisons of map scenarios. The maps show that the Line of Sight (LoS) heuristic generally has the lowest amount of distance seen by observers. Proximity and distance tend to be closer in performance to each other. The distance heuristic outperforms the proximity on map 6 where the proximity heuristic navigates close to obstacles that face an open area, exposing the robot to multiple observers.

and maintains motion along its active plan if there isn't an obstacle preventing navigation. The planner accounts for the previous heading, position, walking speed, goal, and elapsed time since the last plan and current position. Using these parameters, an estimate of the robot's new position is derived. Thus, a motion prediction estimate was created to recognize the difference and reconcile new plans generated on updated map data.

3.5 SCA* Results

The SCARS framework generated a total of 9000 simulations under different circumstances. Each heuristic function (distance, obstacle proximity, and line of sight) was tested an equal amount on six general scenarios with varying numbers of randomly placed observers (see figure 3.4).

Each heuristic method was tested with many configurations of hidden observers. We randomly generated 100 sets of observers throughout the map in scenarios where the sets

Table 3.1: Danger metric averaged over 6 maps for each Heuristic Function

Heuristic Function	2 Observers	4 Observers	6 Observers	8 Observers	10 Observers
Line Of Sight	13.6213m	23.37036m	32.7000m	41.2731m	50.6259m
Obstacle Proximity	17.0520m	30.7108m	44.0932m	57.7072m	71.5252m
Distance Only	18.2929m	33.4865m	49.4729m	65.1997m	80.5068m

were 2, 4, 6, 8, or 10 observers. The hidden observers were not placed in obstacles or too close to them. Placement was restricted to regions sufficiently removed (2.5 meter radius) from obstacles to prevent situations where observers' viewscape only included an obstacle. We generated the average observer visibility region as the robot moved through the randomized environments.

$$danger = \sum_{i=1}^{n} observedDistance_n \tag{3.3}$$

Where n is the observer id.

A measure of stealth was computed by comparing the distance traveled while being visible to an observer. The total visible distance measure may be large as it is the summation of all traversed distance under each observer's field of view (see equation 3.3).

The results of the 9000 runs can be seen in table 3.1. The table illustrates the total distance traveled within the view of all unknown observers. The simulations are not aware of how many observers there are, or their locations and field of view. As the number of hidden observers increases, the line of sight heuristic method changes from a 25.58% to 37.12% improvement over the baseline distance planner. While the obstacle proximity heuristic method outperformed the distance planner (going from 6.79% to 11.16% improvement), performance was not as high as the line of sight heuristic. This is likely due to not expressly considering stealth which reduced the effectiveness of avoiding open areas.

General scenarios (by map) are illustrated in Figure 3.5 with each subgraph showing the average observed distance by the number of observers. We note that in some cases (map 4), the distance optimal baseline does perform reasonably well in a few situations. This is

Fig. 3.6: Examples of the different heuristic paths. The distance path (in red) tends to move through open areas while the proximity (green) and line of sight (blue) avoid regions without occlusion.

due to the nature of the obstacle placement where the distance-baseline planner happened to move through tighter obstacle areas and reached a goal state much faster (thus reducing the total distance observed) compared with other methods. These differences were mitigated as the number of observers increased.

Differences in planned paths (as shown in Figure 3.6) indicate the impact of the various heuristic functions. As evidenced in map 1, the distance-based baseline traverses through an open region, while the other methods gravitate towards occlusions. In both maps 2 and 3, similar performance differences between the methods are still evident.

Heat maps were generated to illustrate where the total visible distance is greatest, with a gradient from green (invisible to observers) moving to red (locations most visible to observers). As seen in Figure 3.7, randomly positioned observers have greater visibility over open areas, which the stealth heuristic approaches avoid. Map 1 again shows the relative safety of both the obstacle proximity and line of sight approaches. Conversely, the distance path moves through the open area, in full view of multiple hidden observers. Heat maps

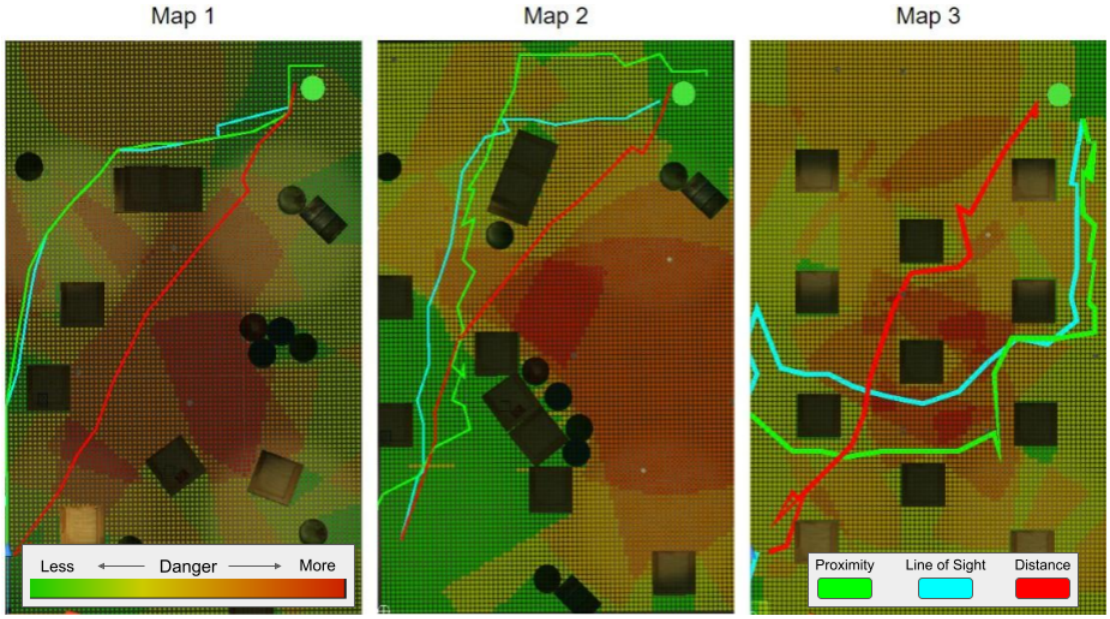

Fig. 3.7: Heat maps showing the observer visibility regions. The distance path (in red) navigates in largely dense observer areas as open areas are easier to observe. The proximity (green) and line of sight (blue) avoid these regions to a greater extent than the distance heuristic method.

allow for simple representation of areas less desirable from a stealth perspective and is useful for demonstrating planner stealth performance.

3.6 Key Insights

We present a novel bio-inspired motion planning algorithm that avoids areas with little to no cover. This algorithm is developed theoretically and tested in a newly developed simulation software: Stealth Centric Autonomous Robot Simulator (SCARS). A legged robot is implemented and used in SCARS for validation experiments. The development of a line of sight cost produces a path that avoids detection from unknown observers more than other methods compared.

The SCARS framework developed custom perceptive sensors, flexible procedurally generated global maps, and efficient collision detection for complex geometry obstacles. A planner easily connects to SCARS and artifacts developed can be quickly ported into ROS for physical hardware tests. The SCARS framework is provided for public use and is open

source.

3.7 Acknowledgment

We would like to thank the NSF ASPIRE ERC (1941524) for supporting this research.

REFERENCES

[1] A. G. Roy and P. Rakshit, "Motion planning of non-holonomic wheeled robots using modified bat algorithm," in *Nature-inspired algorithms for big data frameworks*. IGI Global, 2019, pp. 94–123.

[2] A. K. Kashyap and A. Pandey, "Different nature-inspired techniques applied for motion planning of wheeled robot: a critical review," *Int. J. Adv. Robot. Autom*, vol. 3, no. 2, pp. 1–10, 2018.

[3] F. M. Jaksic and R. C. Soriguer, "Predation upon the european rabbit (oryctolagus cuniculus) in mediterranean habitats of chile and spain: a comparative analysis," *The Journal of Animal Ecology*, pp. 269–281, 1981.

[4] F. M. Jaksic, "Predation upon small mammals in shrublands and grasslands of southern south america: ecological correlates and presumable consequences," *Revista Chilena de Historia Natural*, vol. 59, no. 209.22, 1986.

[5] Z. Zhang, J. Wu, J. Dai, and C. He, "A novel real-time penetration path planning algorithm for stealth uav in 3d complex dynamic environment," *IEEE Access*, vol. 8, pp. 122 757–122 771, 2020.

[6] X. Li, H. Huang, and A. V. Savkin, "Autonomous navigation of an aerial drone to observe a group of wild animals with reduced visual disturbance," *IEEE Systems Journal*, vol. 16, no. 2, pp. 3339–3348, 2022.

[7] F. W. Moore, "Radar cross-section reduction via route planning and intelligent control," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 5, pp. 696–700, 2002.

[8] T. Inanc, M. K. Muezzinoglu, K. Misovec, and R. M. Murray, "Framework for low-observable trajectory generation in presence of multiple radars," *Journal of guidance, control, and dynamics*, vol. 31, no. 6, pp. 1740–1749, 2008.

[9] J. Park, J. S. Choi, J. Kim, S.-H. Ji, and B. H. Lee, "Long-term stealth navigation in a security zone where the movement of the invader is monitored," *International Journal of Control, Automation and Systems*, vol. 8, no. 3, pp. 604–614, 2010.

[10] R. Geraerts, E. Schager *et al.*, "Stealth-based path planning using corridor maps," in *Computer animation and social agents*, 2010.

[11] S. Ravela, R. Weiss, B. Draper, B. Pinette, A. Hanson, and E. Riseman, "Stealth navigation: Planning and behaviors," in *Proceedings of ARPA Image Understanding Workshop*, vol. 10931100, 1994.

[12] A. Kolling and S. Carpin, "Multi-robot pursuit-evasion without maps," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 3045–3051.

[13] M. Al Marzouqi and R. A. Jarvis, "Robotic covert path planning: A survey," in *2011 IEEE 5th international conference on robotics, automation and mechatronics (RAM)*. IEEE, 2011, pp. 77–82.

[14] 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.

[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[16] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[17] J. Huynh, "Separating axis theorem for oriented bounding boxes," *URL: jkh. me/-files/tutorials/Separating% 20Axis% 20Theorem% 20for% 20Oriented% 20Bounding% 20Boxes. pdf*, 2009.

CHAPTER 4

Conclusion: Improved Heuristic Generation

This thesis represents the work accomplished in creating more robust and faster Heuristic functions useful for the robot planning community. Learning generalizable Heuristic functions often benefits from simple ML approaches that find general trends, similar to the simple hand-made heuristics used in many algorithms today. In particular, the ridge regressor ml-heuristic model lowered computation time by 67% compared to the basic analytical heuristic as compared in the SBMPO planning algorithm. This speedup allows SBMPO to be used on more complex maps and converges to solutions quicker with less compute resources. With the lowering of computational requirements, smaller robots with limited power payloads are now able to run planners such as SBMPO effectively to create energy efficient plans.

A general trend was in performance improvements generally for simple machine-learning models. The ridge regressor does not take long to train compared to other ensembles of ML algorithms used in the field and tested, and data requirements are much lower. Perhaps more importantly, the evaluation time costs only a few FLOPs and thus did not impact the convergence speed of the algorithm. Conversely, the performance gains from using deep learning and other parameter intensive ML approach often resulted in longer total convergence time. As machine learning becomes more prevalent in heuristic design, using simple models to quickly create efficient ml-heuristics will be beneficial for practical implementation.

As heuristics were refined, the focus shifted to a Heuristic designed for stealth traversal. The novel stealth planning algorithm outperformed a basic Dijkstra's path planner when in terms of time observed by hidden observers. The developed stealth heuristic reduced time in the line of sight of hidden observers by over 35%. This reduction of time under observation is key in situations where the enemy could spot a robot, the new Heuristic approach is

crucial for ground robots performing critical stealth-based missions. Designing a heuristic estimate where the function directly predicts possibilities it is not aware of was key to the success of the algorithm. There is still much refinement required for stealth planning and work is ongoing in this line of research.