OPTIMIZING THE A* ALGORITHM:

A PATH TO EFFICIENT UAV NAVIGATION

Word Count: 4968

Introduction

Unmanned aerial vehicles (UAVs), popularly referred to as drones, fly long distances and conduct a myriad of tasks such as transporting payloads and conducting surveillance, without human pilots physically on the aircraft. Most modern drones are remotely controlled by humans or computers. In the recent past, research has been conducted to enable autonomous travel in drones, to achieve cheaper and quicker drone transportation. Sensory equipment, such as LiDAR (Light Detection and Ranging), RaDAR (Radio Detection and Ranging), or camera-based technology, can allow a UAV to interpolate its position and the locations of surrounding obstacles so that it can travel efficiently, while avoiding obstacles along the way. This paper is mainly focused on the autonomous aspect of drone navigation.

UAVs have many commercial applications for society. These include photography, mapping, agriculture, and live streaming, among others.[1] Technology delivery company Amazon is investing heavily in UAVs for cheaper and quicker package deliveries.[2] With these widespread applications in multiple sectors of the economy, UAVs have a growing influence on society, and their increasing numbers could become hazardous if they collide with each other or the structures around them. To handle this problem, UAVs need to be designed in order to ensure collision avoidance and operational safety, while staying fuel-efficient to keep costs low.

UAVs also have disaster relief applications, according to Eastern Kentucky University.[3] Such applications include mapping disaster zones, assessing the damage caused, transporting

---

[1] "Applications and Uses for UAV Multirotor Drones," Rise Above Custom Drone Solutions, https://www.riseabove.com.au/drone-services/uav-applications-and-uses/. Accessed December 8, 2019.

[2] "Amazon Prime Air," Amazon, https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011. Accessed December 8, 2019.

[3] "5 Ways Drones Are Being Used for Disaster Relief," EKU Online, November 29, 2018, https://safetymanagement.eku.edu/blog/5-ways-drones-are-being-used-for-disaster-relief/.

supplies, and extinguishing wildfires. In addition to the commercial applications of UAVs, there is a humanitarian aspect that can help dampen the impact on victims of natural disasters and help them recover faster. If a hurricane were to strike an area, drones could be deployed shortly afterwards to carry first-aid supplies to remote or vulnerable communities. These drones can often be used in groups (also known as *swarms*) working together in a coordinated effort to efficiently address the task at hand. Since they can help speed up the recovery process after natural disasters, much research has been done to develop safer and more efficient deployable solutions. However, much of the research done on drones is private and proprietary, hampering the use of drones for disaster relief in developing countries. A motivation of the research in this paper is to develop an open-source algorithm to meet these humanitarian objectives.

To know how to implement navigation techniques (algorithms) in UAVs, it is important to first understand how they work. The focus of this research is on a specific type of UAV known as a quadrotor drone, although it can be adapted to fit other models. According to Rhett Allain, an associate professor of Physics at Southeastern Louisiana University, a quadrotor drone uses torque, a rotational analog of force, to maneuver.[4] There are 4 rotors spaced out in a square formation, with two spinning clockwise and two spinning counterclockwise. Two adjacent rotors always spin in opposite directions, cancelling out rotational torque and inducing a translational force. Powering a pair of opposite rotors, however, creates a net torque on the body, allowing it to change direction. This facilitates drone motion in a desired direction.

In the quest to design an appropriate navigation algorithm for this research, many published path-finding and navigation algorithms were explored. Among these, the popular, open-source A* (pronounced “a-star”) appeared to be a good fit, given its simplicity and

[4] Rhett Allain, “How Do Drones Fly? Physics, of Course!” Wired, June 3, 2017, https://www.wired.com/2017/05/the-physics-of-drones/.

flexibility. A* is an algorithm that assigns a cost to each location and uses a heuristic (prediction of how cost will change) to find the least-cost solution.[5] Different variants of the A* algorithm, implemented by the researcher, are compared on the basis of collision avoidance and fuel efficiency, to determine which methods succeed along an obstacle-prone path to a destination.

Literature Review

In 2007, at the American Institute of Aeronautics and Astronautics, G Hoffman and a group of Stanford scholars analyzed the physics behind quadrotors and the thrust necessary to propel them forward.[6] Using a branch of physics known as Momentum Theory, they refined a model relating the freestream speed (true airspeed) of a quadrotor UAV and its induced velocity. The term *induced velocity* is defined as the tendency for an airfoil, such as a rotor blade, to pull down surrounding air molecules and propel itself upward. This model requires an equation to be solved for induced velocity, which is then substituted into another formula that computes the power of the rotor. Using this two-step process, one can compute the rate of drone energy use.

Another paper, written by C Samaras – an environmental engineer from Carnegie Mellon – and others, explores this exact topic.[7] It utilizes concepts from the Stanford research mentioned earlier, to create and test a model of energy consumption based on factors such as velocity and size. Using the Momentum Theory framework and experimental data they collected from specific supply-carrying quadrotor UAVs, the energy consumption per unit distance was plotted for varying velocities under different conditions, such as the number of propellers and the

---

[5] "A* Search Algorithm," GeeksforGeeks, September 7, 2018, https://www.geeksforgeeks.org/a-search-algorithm/.

[6] Gabriel Hoffmann, Haomiao Huang, Steven Waslander, and Claire Tomlin. "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment." *AIAA Guidance, Navigation and Control Conference and Exhibit* (2007), https://doi.org/10.2514/6.2007-6461.

[7] Joshuah K. Stolaroff, Constantine Samaras, Emma R. O'Neill, Alia Lubers, Alexandra S. Mitchell, and Daniel Ceperley, "Energy Use and Life Cycle Greenhouse Gas Emissions of Drones for Commercial Package Delivery." *Nature Communications* 9, no. 1 (2018), https://doi.org/10.1038/s41467-017-02411-5.

inclusion of a package. To maintain replicable results, Samaras and others provide specifications (such as mass, size, and drag coefficients) of the UAV model in its supplementary material. The paper defended the claim that UAV package transport can be more efficient compared to ground transport due to its relatively low fuel consumption.

Other literature explores algorithms for UAV motion and navigation. One very straightforward approach to navigation is the A* algorithm, described in a peer-reviewed thesis written by T Liao, an Auburn University graduate.[8] This algorithm works by exploring simple paths to a destination, using a heuristic to estimate how much cost (distance) has been covered and how much is left in its current path, by weighing different paths in an attempt to minimize the total distance from a start to the goal. Given that A* only considers its current state and plans a path to the destination, the path has to update itself regularly as new obstacle information is retrieved. Another paper, written by an Informational Engineering Faculty member in the China University of Geosciences, W Zeng, applies the concept of A* to road networks, detailing the procedure for A*.[9] Using an experimentally conducted procedure on seven different algorithms, the source concluded that A* is the quickest algorithm to navigate complicated networks, with the added bonus of always finding the shortest path.

Overall, the relevant literature deals with the physics of UAVs and how to navigate them quickly and safely. The work of Hoffmann and Samaras explains how UAVs navigate and how induced velocity can be used to calculate the energy consumed. Liao and Zeng's papers explore the A* algorithm, with benefits as a simple, open-source pathfinding algorithm for drones.

---

[8] Tingsheng Liao, "UAV Collision Avoidance using A* Algorithm," https://pdfs.semanticscholar.org/1d67/0367799f2aa2eef7998942ced9f97c265200.pdf. Accessed October 11, 2019.

[9] W. Zeng, and R. L. Church. "Finding Shortest Paths on Real Road Networks: the Case for A*." *International Journal of Geographical Information Science* 23, no. 4 (2009): 531–43. https://doi.org/10.1080/13658810801949850.

Although the sources mention fuel efficiency and collision avoidance strategies, no single source is able to provide a comprehensive analysis of both parameters.

Publicly available literature neglects to focus on the intrinsic connection between efficiency and safety. A focus on fuel efficiency results in a faster speed. This higher speed leads to increased momentum, which means that it would take longer to change direction if a new obstacle were to appear. On the other hand, an emphasis on safety would mean a slower speed, so that the drone has better responsiveness to obstacles. This would lead to more hovering, resulting in less fuel efficiency. To address the gap in the available knowledge, this research aims to focus on 3 researcher-created parameters, developing a variant of the A* algorithm that prioritizes both efficiency and safety, by formulating the research question: **Which combination of penalty function, step size, and input size will produce the best results for fuel efficiency and collision avoidance when the A* algorithm is optimized for drone pathfinding?**

Method

The method used is an iterative engineering design process, which involves developing a product, testing for deficiencies, and improving the design repeatedly. This process of optimization of the A* algorithm over time, by evaluating different models, was performed to make the simulation as realistic as possible. Once the simulation was designed, the plan involved finding the best combination of the following variables designed by the researcher: **input size**, **step size**, and the **penalty function** in terms of fuel efficiency and safety. These variables will be explained in more detail in the method description.

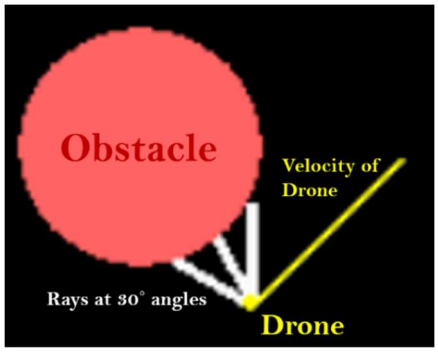
Simulation Design

To design the simulation, a specific model of a quadrotor UAV, the *3D Robotics Iris*, was chosen since data regarding this drone type is freely available and it is a good representation of

quadrotors in general. Next, data which influence the aerodynamics of the UAV were gathered, such as its mass, drag parameter, and the density of the surrounding air. By solving a quartic equation using Newton's method, a Calculus-based algorithm for solving polynomials, a value for induced velocity was calculated. Induced velocity was used to create an expression for the power (energy consumption rate) of the specific UAV based on its acceleration and velocity. The power was used to determine the energy efficiency of following a particular path.

After the energy model was constructed, the next step involved creating a simulation in PyGame (which requires Python modules) that would entail the basic movement of a UAV. The simulation uses a black screen, and refreshes at a default frame rate of around 10 frames per second, with each frame representing what will be termed a "step." A *player* object representing a drone was added, which could be moved linearly within a grid, moving a small amount each step, with 8 options to move to adjacent or diagonal cells. This movement was later replaced by a more complex system, based on Kinematic Equations which detail the impact of accelerations on velocity and position. The 8 directions controlled the lateral and longitudinal accelerations of the UAV in each frame, leading to a more realistic model. Each player object was designed to have a *goal*, a 1.5 meter wide circular region surrounding its destination. Getting to this region would mean that the algorithm succeeded in navigating the drone safely to the destination.

Other objects, called *entities*, served as obstacles. Random movement controlled by a *seed* (a value that controls the random number generation) was implemented in obstacles. Collision detection between the drone and its environment was implemented in the drone, so that crashing into an obstacle or a simulation boundary would result in a failure. If an obstacle were to collide with a wall, it would physically move to the other side of the screen, adding some degree of unpredictability that the UAV must deal with.



In the image above, there is an input size of 12, as 12 lines are drawn, 30 degrees apart, for obstacle detection.

Through a system called *ray-tracing*, a fixed *input size* (the first of 3 main parameters involved in optimizing the algorithm) was chosen and used to draw that number of rays emanating from the UAV, with trigonometry to detect the presence of nearby obstacles. The *input size* is an important factor to consider, as too few inputs make detection difficult, though having too many inputs adds more time complexity and makes the UAV more focused on obstacles instead of pursuing the goal in an efficient manner.

Original A* Algorithm

After the simulation was designed and implemented, the next step was to program the conventional A* algorithm. This involved generating f-costs (final path costs) as sums of g-costs (to get to the next point) and the h-costs (predicted costs for the remainder of the path). The options with the lowest f-costs were always chosen first, using a *binary heap (priority queue)* structure to organize the f-costs in increasing order from lowest to highest.

This grid-based implementation of the A* algorithm will be referred to as the *Original A* Algorithm*. Code for the Original A* algorithm was not implemented in the final research as it did not accurately describe how drones navigate in the real world, but it served instead as a demonstration of how the algorithm works. To apply these concepts to the drone model described before, the researcher created a variant of this algorithm, which would not function as

a grid, but would allow the drones to explore a continuous range of possibilities where they could be in the future. The diagram below shows the functionality of this implementation.

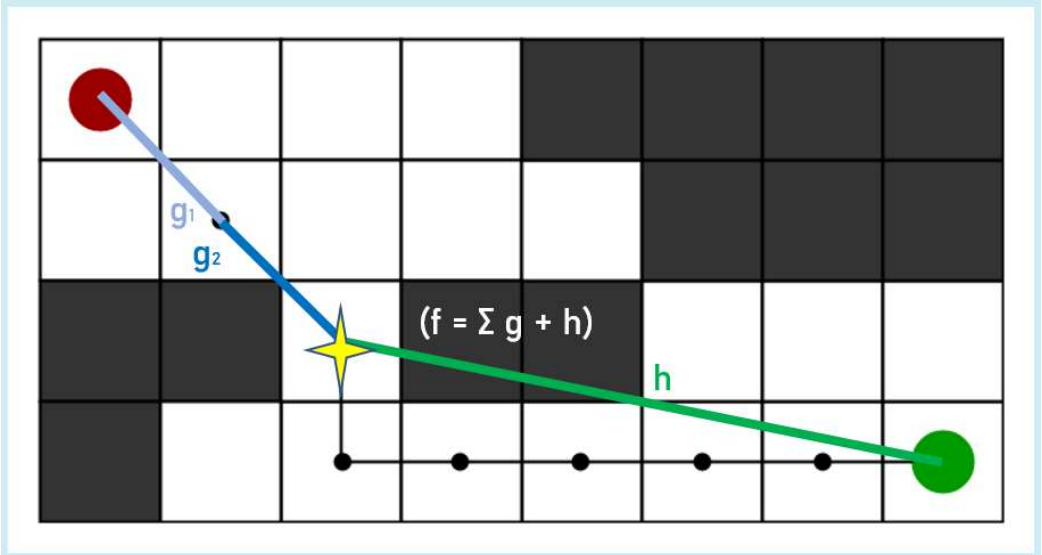


Photo Credit: GeeksforGeeks

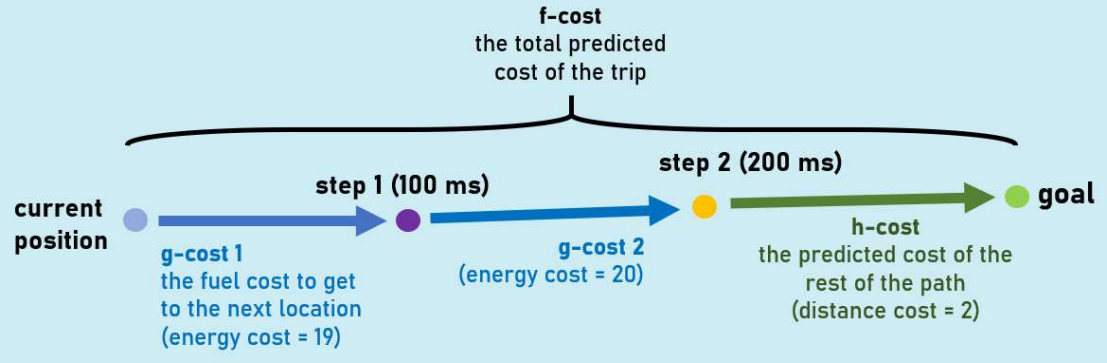
ORIGINAL A* ALGORITHM. In the image above, the yellow location is evaluated by the A* algorithm, by summing the distance cost, called the *g-cost*, to reach it from the starting point (in red), and the remaining distance cost to reach the goal (in green), the *h-cost*. The objective is to minimize the sum of these costs, known as the *f-cost*.

In what will be called the *Standard A** Algorithm, the g-costs in the Original A* algorithm, originally implemented as distance costs, were replaced by energy-based costs, proportional to the energy consumed per "step." The h-costs remained as distance costs. Since there were no longer specific locations where the drones could travel, the concept of *states*, describing the location, velocity, and f-costs after 100-millisecond intervals, was used instead.

Standard A* Algorithm

The concept of a grid used in the *Original A* Algorithm* would not fit the model designed for this research. Given that the drones designed for the simulation could control their accelerations in several directions due to the Kinematic Equations mentioned earlier in this section, there was no longer a fixed number of points where the drones could travel. By slightly altering their accelerations, an exponentially increasing number of paths could be generated, which proved to be an issue for the computational efficiency of the algorithm. A demonstration

of the *Standard A* Algorithm* is presented in the diagram below, showing only a single path for simplicity. In the actual implementation, there would be an infinite number of paths to explore.



Standard A* Algorithm implementation, where each *step* represents 100 ms of time.

With a choice to sacrifice the optimality of A* or the practicality of the algorithm in terms of efficiency, a decision was made by the researcher to use a *greedy* approach. Greedy algorithms, which often are able to optimize multi-step problems by choosing the best option at every point in time, were fitting to use in this situation as they could massively lower time and space complexity and were not implemented into the A* algorithm in the surrounding literature.

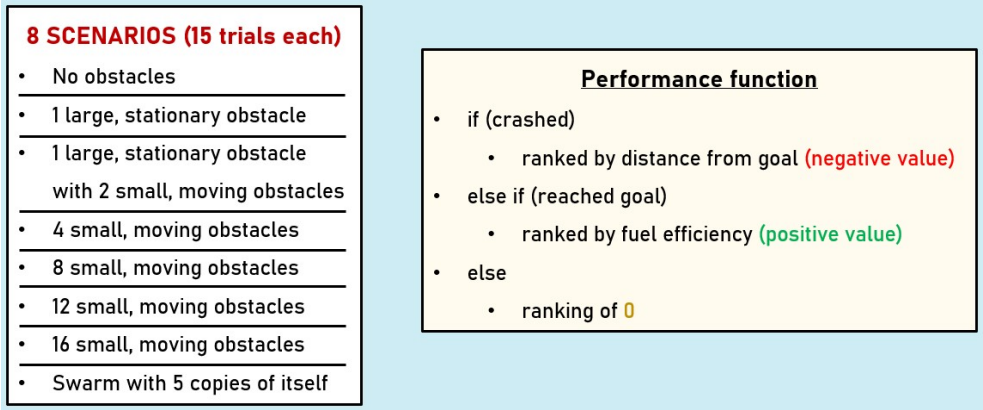
The greedy approach implemented in this research consisted of only running the first few "steps" of the A* algorithm, according to a *step size* which would determine how many "steps" ahead the drone would calculate before moving to the next location and calculating another short-term path. In this way, the algorithm was able to follow a subroutine that chose the best location to travel to within a certain amount of "steps;" then, after moving to that location, the process repeated. The value of this step size was another parameter in optimizing the algorithm, as more steps would require more computational time but would generate more optimal paths.

Improved A* Algorithm

The Standard A* algorithm worked well, but only in static environments. If moving obstacles were present, the UAV would only realize and move out of the way when it is too late, since gradual changes to velocity are necessary to bring the UAV to a stop. So, a *penalty*

function, the third parameter designed to improve the A* algorithm's functionality, was introduced by the researcher. This penalty function would increase the cost of travelling to a location if an obstacle is nearby, and the increase would be proportional to how close surrounding obstacles are. Stronger penalty functions would avoid obstacles better, but possibly to an extent that they ignore opportunities to reach the goal efficiently.

Each of the 3 parameters discussed was implemented in the program. Multiple values for each variable were chosen. There were 5 step sizes from 1 to 5, and 3 input sizes of 12, 24, and 36. Three penalty functions – *no penalty (Standard A* Algorithm as a control)*, *linear penalty*, and *hyperbolic penalty* – were used, with each name describing its properties. The *no penalty* choice implies that the input size does not matter, leading to 7 penalty-input pairs, for each of 5 step sizes. For all 35 combinations of these variables, a *performance function* was determined from 15 trials each, which would be positive if the drones reached the goal and negative if not.



A diagram to represent how testing was done to determine the best combination of the 3 variables. Sometimes, the algorithm might circle the goal without colliding with obstacles; in that case, there would be a performance of 0.

The procedure was replicated with 8 different arrangements of static and moving obstacles. In one arrangement, there were 16 birds (small, moving obstacles), while another had 6 identical copies of the same drone, all with different starting points and goals. From all scenarios, the data was gathered, in an attempt to decide which of the 35 combinations (including the Standard A* Algorithm as a control) best optimized the A* algorithm.

Data

Figure 1 (Key for the Collision-Avoidance Color Coding)

The primary color associated with each distribution represents which penalty function it corresponds to (blue = no penalty, red = linear, green = hyperbolic), whereas the shade signifies how many sensory inputs the drone used (light = 12, medium = 24, dark = 36). Blue represents the standard A* algorithm, and therefore it is not impacted by input size, justifying the inclusion of only one blue group.

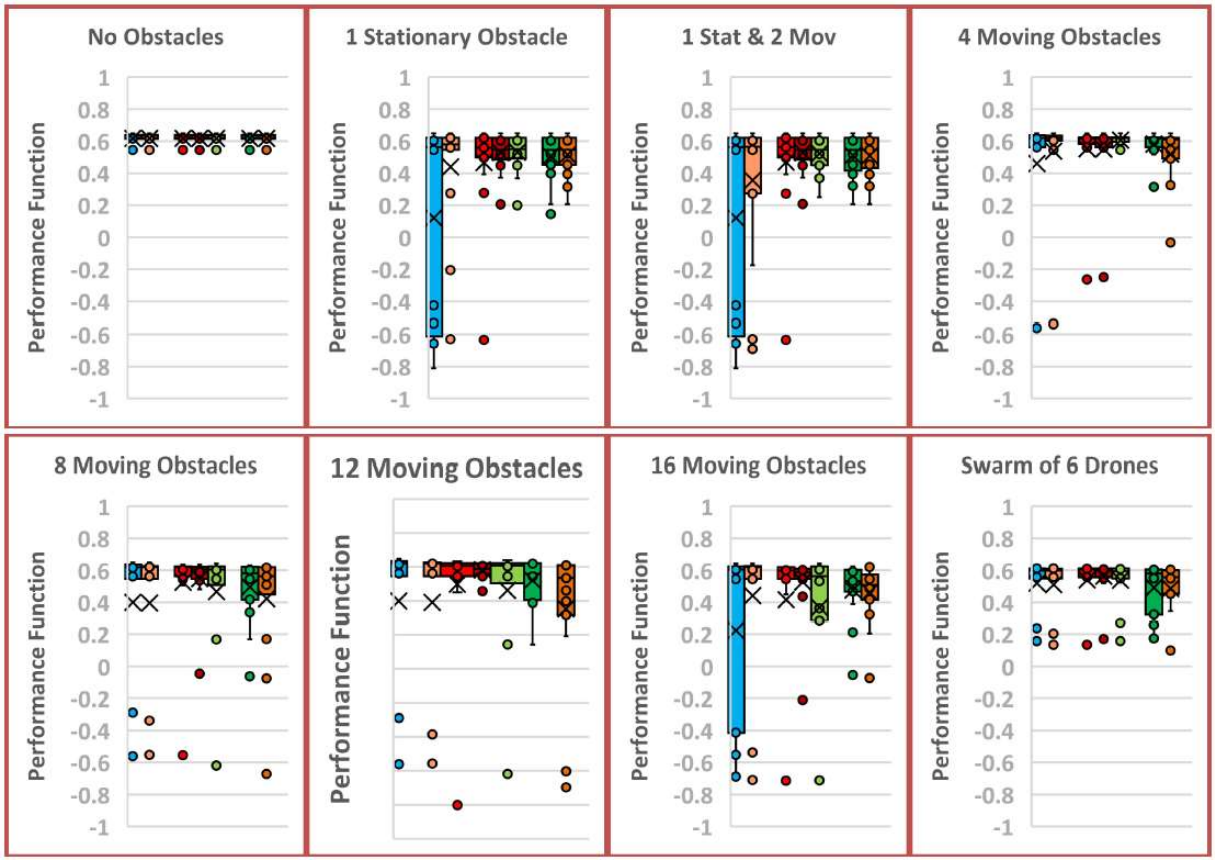


Figure 2.1 (Step Size = 1)

Performances of 7 algorithmic variants (with step size 1) in 8 different scenarios, with 15 trials each (840 total)

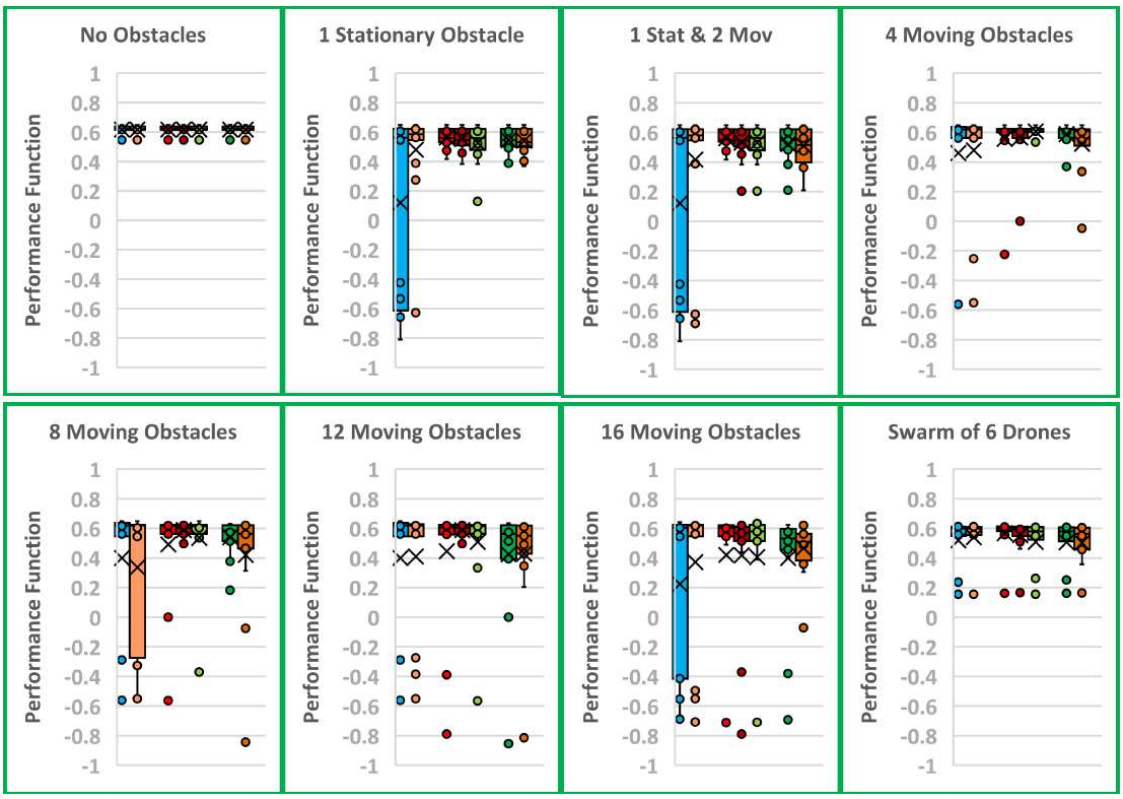


Figure 2.2 (Step Size = 2)
Performances of 7 algorithmic variants (with step size 2) in 8 different scenarios, with 15 trials each (840 total)

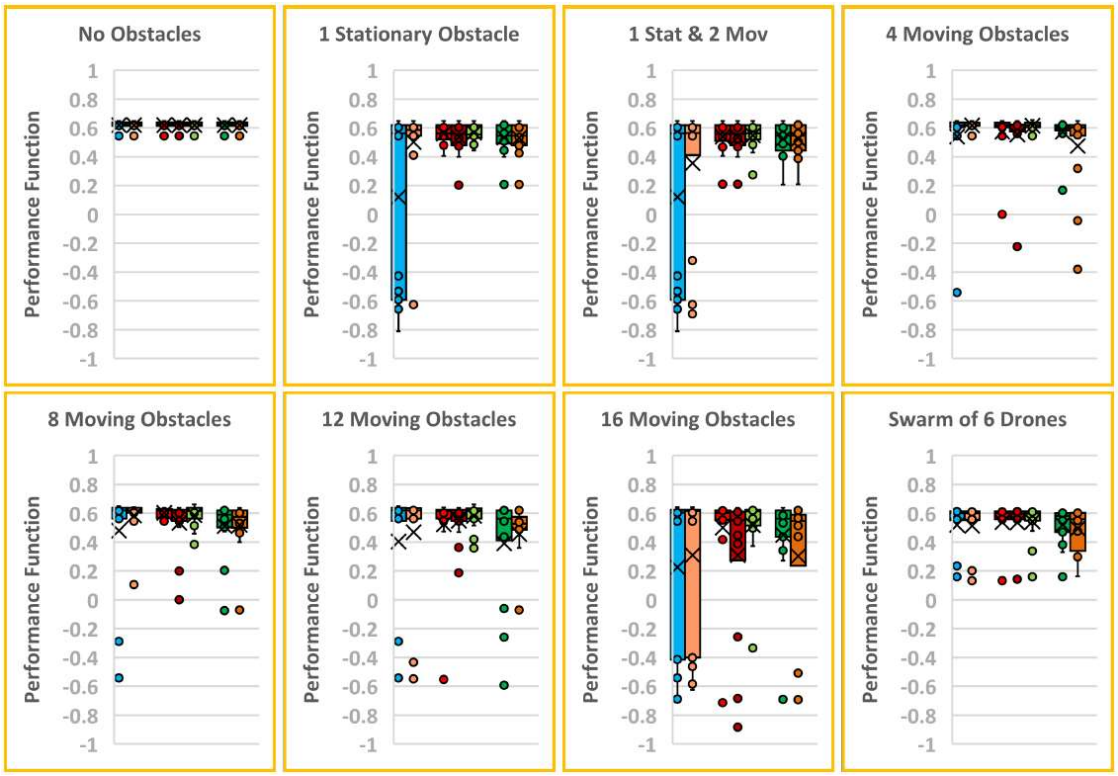


Figure 2.3 (Step Size = 3)
Performances of 7 algorithmic variants (with step size 3) in 8 different scenarios, with 15 trials each (840 total)

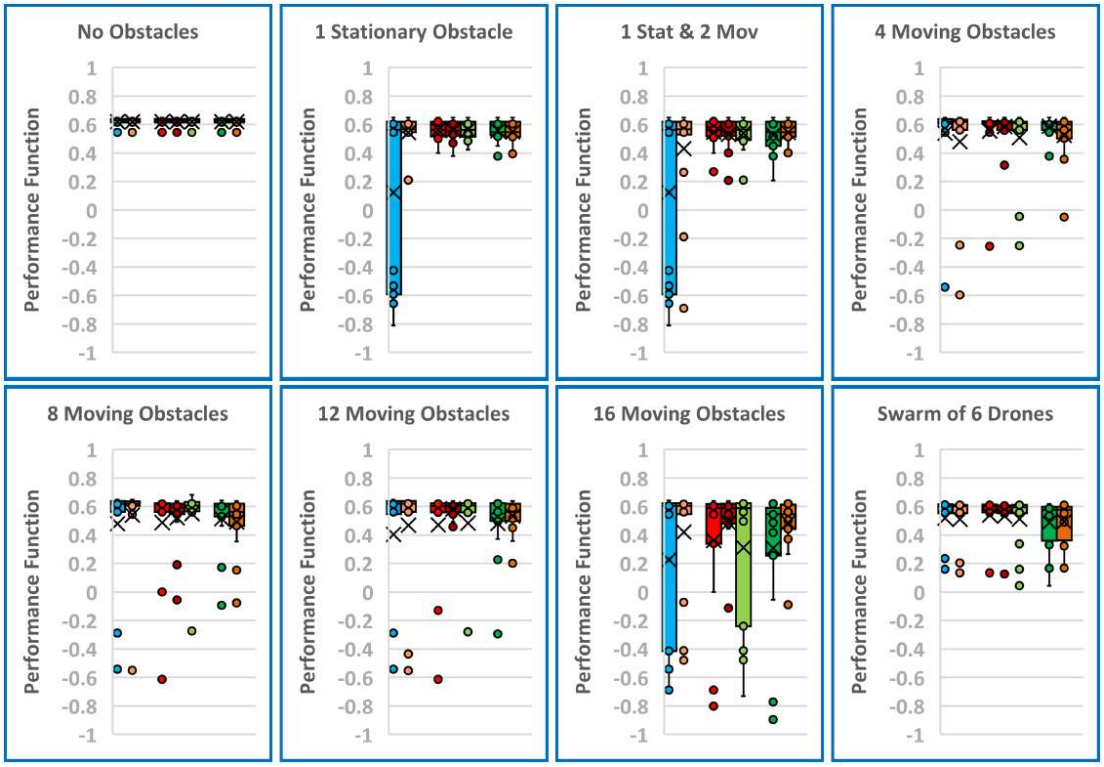


Figure 2.4 (Step Size = 4)

Performances of 7 algorithmic variants (with step size 4) in 8 different scenarios, with 15 trials each (840 total)

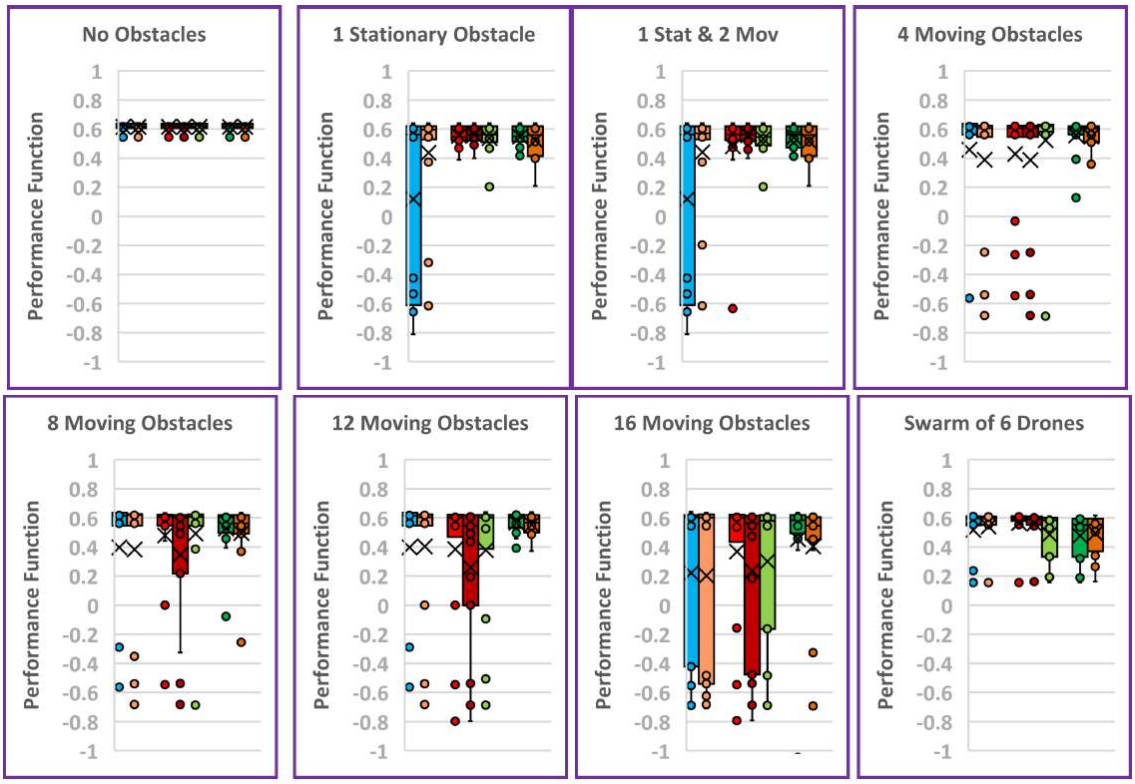


Figure 2.5 (Step Size = 5)

Performances of 7 algorithmic variants (with step size 5) in 8 different scenarios, with 15 trials each (840 total)

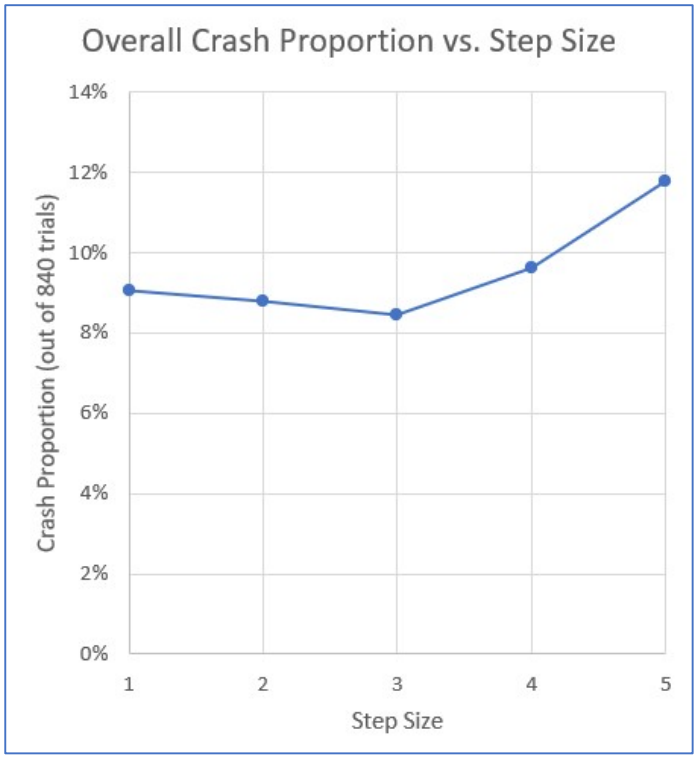


Figure 3 (Graph of Crash Proportion vs Step Size)
Shows the crash rates of the combined data for each of 5 different step sizes tested in this research. The graph indicates a minimum crash rate is experienced at a step size of 3.

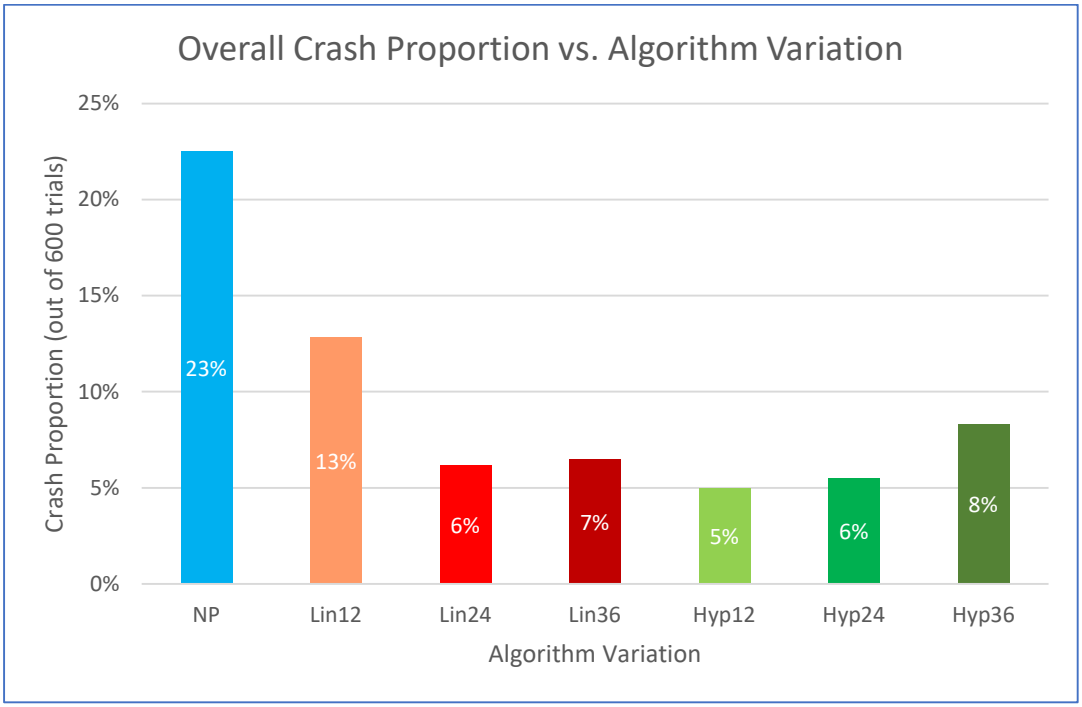


Figure 4 (Graph of Crash Proportion vs Penalty Function)
Shows the crash rates of the combined data for each of 5 different step sizes tested in this research. The graph indicates a minimum crash rate is experienced with the light green bar (hyperbolic with 12 inputs).

Results

Overview

Each side-by-side boxplot compared the performances of various input sizes and penalties, with 7 different distributions, color-coded based on the penalty function and the number of inputs, as shown in the key (Data: Figure 1).

The boxplot distributions contain values between -1 and 1, which represent the performance function output of each of the 15 trials. The median (8^{th} trial in sorted order) shows the central tendency of each skewed distribution. The *inter-quartile range*, or IQR (between the 4^{th} and 12^{th} trials in sorted order), shows the variation of the middle 50% of the data. Points which extend beyond the outer whiskers are outliers, which typically represent crashes (performance values that are negative). In the case of swarms, the average performance of all 6 drones was taken, and since most of the drones in the swarm did not crash, crashes were represented by lower-than-usual performances rather than negative numbers (Data: Figure 2).

Two different forms of analyses were conducted based on the results of this experiment. The first pertained to analyzing the crash rates of each of the combinations in broader categories, while the second looked at the specific distributions themselves to discern how well each individual algorithm performed in each scenario.

Quantitative Analyses of Crashes

To focus on the combinations that resulted in the 401 total crashes out of the 4200 trials, two specific quantitative analyses of the crashes were conducted:

1. For each of the step sizes from 1 to 5, 840 trials from all penalties, inputs, and testing scenarios were combined in order to find the step size with the lowest crashes overall. The sizes, in increasing order from 1 to 5, had 76, 74, 71, 81, and 99 crashes, respectively

(Data: Figure 3). These were then converted into proportions out of the 840 trials, and represented as percentages in the line graph shown in Figure 3.

2. For each group of variables associated with collision avoidance (i.e. the penalties and inputs) shown in the key (Data: Figure 1), the 600 trials for each penalty-input pair were combined. The light green group had 30 total crashes out of the 600 trials in which it was tested, a crash rate of 5% (Data: Figure 4). In comparison, the highest number of crashes was present in the blue distribution: a total of 135, a crash rate of 23% (Data: Figure 4).

Quantitative Analyses of Distributions

Analyses of the specific distributions themselves was also conducted, based on the number of crashes as well as the overall center (median) and spread (IQR). For each distribution in Figure 2, the number of crashes, the median, and the IQR were compared with other distributions in order to discern which combination of the algorithm performed better.

The results of this analysis showed that a specific combination, with step size 3, hyperbolic penalty, and 12 inputs, performed better consistently compared to the others (Figure 2.3 in light green). Out of the 120 trials in which it was tested, it was able to reach the goal successfully 117 times (97.5% success rate). The unaltered A* algorithm (3 steps without a penalty) reached the goal 94 times in total (78.3% success rate), representing the significantly worse performance the standard A* algorithm achieved compared to the variant in this research.

Discussion

Overall Trends

The overall results showed that the variations of the A* algorithm created in this research perform better compared to the standard A* algorithm. This is evidenced by comparison of the crashes and additional features of the no penalty distributions with other distributions.

In terms of crashes, 23% of all trials involving no penalty resulted in crashes (blue distribution in Figure 4) which translates to about 1 in 5 drones destroyed due to crashes. Similarly, when focusing specifically on step size 3, the lack of any penalty led to a 21.7% crash rate (Figure 2.3). Extending from the trend seen here, if 1000 drones were to be sent out on a natural disaster recovery mission, more than 200 drones would be predicted to crash because of improper sensing of their environment. Although this is extrapolation beyond the work in this simulation, it still highlights the impact of flawed collision detection capabilities in drones. Crashes lead to significant economic losses since each individual drone is itself expensive and equipped with costly LiDAR equipment. Without proper collision avoidance, drones cannot function as intended and they may end up hampering relief efforts rather than helping them.

When compared to the no penalty (blue) distributions, the hyperbolic penalty with 12 inputs (light green distribution in Figure 4) significantly lowers the crash rate. The light-green distributions had a crash rate of 5%, a decrease by more than a factor of 4 compared to the 23% seen when there is no penalty. When focusing on just step size 3, the light green distribution had a crash rate of 2.5% compared to the 21.7% seen in the blue distribution, which is almost a tenfold drop in the crash rate, showing just how effective penalties and sensory inputs can be in reducing collisions (Figure 2.3).

Furthermore, an interesting trend emerges when crash rates across different step sizes are compared in the line graph shown in Figure 3. The number of crashes seems to be minimal at step size 3 (Figure 3). From close inspection of the crashes for the low step sizes (1 and 2), it appears that the quick recalculation is too focused on the short term, not allowing the algorithm to predict future collisions and only reacting immediately before a collision is about to occur. For high step sizes (4 and 5), the drone focuses too much on a long-term prediction of where

collisions will be, not recalculating quickly enough to compensate for moving obstacles. At step 3 (Figure 3), a balance of the two extremes, the crash rate settles to a minimum of 8.45%, compared to 9.05% (step 1) and 11.8% (step 5), hinting that step size 3 may be best overall step size for dynamic recalculation of the paths in an efficient manner.

When considering other factors relevant to the distributions, medians are a good metric for determining overall tendencies. However, since each distribution had fewer crashes compared to successes, medians remained positive, at around a performance of 0.6 (with 60% fuel efficiency compared to ideal conditions) for all distributions. Since uniform fuel-efficiency calculations were implemented in all the algorithms, including the Standard A* Algorithms, fuel efficiencies would be similar for all algorithms. The medians were therefore discarded as a metric for comparison. The IQRs, another comparison standard across the distributions, were large in the blue distributions, showing that the large number of crashes heavily skewed the data in the negative direction. In other words, the standard A* algorithm has less reliability and is more prone to crashes compared to the other variants of the algorithm.

Implications

This work has several implications which contribute to the literature existing in the field. Much of the current A* algorithm usage involves using a grid.[10] The extension of this algorithm to contexts such as fuel consumption and acceleration-driven motion allows this research to better model the real world, bridging a gap between the A* and the fuel efficiency perspective.

The inclusion of a "penalty cost" for approaching obstacles in this study provides the added benefit of slightly modifying the A* algorithm to increase collision avoidance without increasing computational complexity. The addition of the penalty *function* allows the drone to

[10] Liao, "UAV Collision Avoidance Using".

react to obstacles in specific ways; choosing the right penalty function can significantly reduce computational time and increase the safety of the drones. This adds a new understanding to the field, of how to avoid obstacles in a fuel-efficient way.

The benefit of using a greedy step-based approach in this study, contrary to the static pre-calculation done by other implementations of the A*, involves significantly more efficient computation that can adapt to the environment while generating and choosing optimal paths several times within a single second. This contributes to achieving the research goal, which involves bridging the gap between the research done on dynamic pathfinding algorithms and the open-source A* algorithm, in order to create an affordable, adaptable, and freely available solution for safely navigating drones.

Of the 35 different combinations tested in this research, one performed significantly better than the rest. To answer the research question, the 3-step, 12-inputs, hyperbolic penalty combination performed the best, with the lowest crash rate of 2.5% out of 120 trials. This claim is supported by the quantitative analyses of both crashes and the individual distributions themselves, and it helps reach a new understanding that a moderate step size, with few inputs and a harsh penalty, is the best way to handle drone navigation in the drone model presented.

Limitations

A limitation of this study, however, is the fact that a simulation, rather than actual testing of a drone, was used to collect data. Although the simulation is programmed to use models of fluid flow and kinematics to model the path of a drone as accurately as possible, more testing is necessary to show that this algorithm has similar practical applications. For the case of simplicity, the z-dimension (altitude) was excluded from the simulation, restricting the motion of

the drone to motion within a 2D plane of space. Further research should be conducted on a 3-dimensional model in order to more confidently generalize the simulation to the real world.

Another limitation of the study is the inclusion of only 8 directions for drone acceleration, in the 4 cardinal directions and their diagonal variants (NE, NW, SE, SW). Real drones have a continuous range of choices for directing their motion, and this should be expanded upon in the future. Since the filtering of different states by the A* algorithm runs somewhat exponentially, the goal of the 8-direction limit was to avoid too much computation. However, it is true that this boost in computational speed comes at the cost of restricting the range of options a drone has. Quicker programming languages than Python, which was used in this simulation, like C++ or Java, could be used to ameliorate the effects of slow computation.

These limitations do not discredit the research process itself, but they instead serve to caution against extrapolating the results beyond the simulation's boundaries. Since this research is a simplification to avoid the possible destruction of expensive drones during testing, there remains work be done to comprehensively test the efficacy of these A* variants in the real world.

Conclusion

A combination of 3 steps, 12 inputs, and hyperbolic penalty, performed best with only 3 crashes out of the 120 trials in which it was tested in this research. This conclusion is supported by analysis of the individual crashes across step sizes, input sizes, and penalty functions. Additionally, as most of the data is clustered at high performances, and the IQR is small, the performance is shown to be reliable. The new crash rate, using this combination, was significantly reduced to around 12% of the standard A* algorithm's crash rate. This improvement can revolutionize the use of A* as a drone navigation technique and broaden the

capabilities of the algorithm, and drones in general, helping communities in disaster relief and allowing for cheaper, quicker, safer, and more efficient deliveries.

The research itself addresses gaps present in the surrounding literature and helps to merge fuel-efficiency and collision-avoidance, two perspectives that are very important to consider when testing the practicality of drones. Adding parameters such as the *penalty function* and *step size* allows for more comprehensive models for autonomous drone navigation, that can efficiently compute small, manageable subroutines that consider distances to obstacles. With the use of an *input size*, this research is able to model the LiDAR capabilities of drones, to detect obstacles and avoid collisions while still keeping an emphasis on fuel. The collected data not only answers the research question, but it contributes to the overall understanding of how drone navigation models should operate.

There is still more work that must be done to further improve the algorithm discussed in this research. By adding a third dimension of altitude, the research can more accurately model the applications of drones in the real world. Other techniques, such as artificial intelligence, may help to improve the A* algorithm by helping with obstacle detection, such as identifying the shape of an obstacle or interpolating the speed at which obstacles are traveling. Eventually, the algorithm will need to be tested in real world conditions in an actual drone, in order to test its efficacy in drone swarm operations. As a first step, though, this research will help pave the way for more sophisticated, open-source, and affordable use of powerful drone technologies.

Bibliography

"5 Ways Drones Are Being Used for Disaster Relief." EKU Online, November 29, 2018. https://safetymanagement.eku.edu/blog/5-ways-drones-are-being-used-for-disaster-relief/.

"A* Search Algorithm." GeeksforGeeks, September 7, 2018. https://www.geeksforgeeks.org/a-search-algorithm/.

Allain, Rhett. "How Do Drones Fly? Physics, of Course!" Wired. Conde Nast, June 3, 2017. https://www.wired.com/2017/05/the-physics-of-drones/.

"Amazon Prime Air." Amazon. Accessed December 8, 2019. https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

"Applications and Uses for UAV Multirotor Drones." Rise Above Custom Drone Solutions. Accessed December 8, 2019. https://www.riseabove.com.au/drone-services/uav-applications-and-uses/.

Hoffmann, Gabriel, Haomiao Huang, Steven Waslander, and Claire Tomlin. "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment." *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007.

Liao, Tingsheng. "UAV Collision Avoidance using A* Algorithm." Accessed October 11, 2019. https://pdfs.semanticscholar.org/1d67/0367799f2aa2eef7998942ced9f97c265200.pdf.

Stolaroff, Joshuah K., Constantine Samaras, Emma R. O'Neill, Alia Lubers, Alexandra S. Mitchell, and Daniel Ceperley. "Energy Use and Life Cycle Greenhouse Gas Emissions of Drones for Commercial Package Delivery." *Nature Communications* 9, no. 1 (2018). https://doi.org/10.1038/s41467-017-02411-5.

Zeng, W., and R. L. Church. "Finding Shortest Paths on Real Road Networks: the Case for A*." *International Journal of Geographical Information Science* 23, no. 4 (2009): 531–43. https://doi.org/10.1080/13658810801949850.