# Dynamic Path finding with A* and Bellman-Ford Algorithms for Drone Movement Simulation

Niranchna Natarajan, Lakshmi Nivetha A S, Esther Rachel Thomas

Vellore Institute of Technology, Chennai

niranchna.natarajan2022@vitstudent.ac.in, lakshminivetha.as2022@vitstudent.ac.in, estherrachel.thomas2022@vitstudent.ac.in

## Abstract

The project attempts a hybrid approach to dynamic path finding by combining the A* algorithm with the Bellman-Ford in order to simulate an adaptive navigation system in drones. The A* algorithm identifies an efficient, optimal route by using heuristic cost estimations. Bellman-Ford algorithm allows for the dynamic modification of path costs by being able to update them due to the introduction of new obstacles. The combination of these two methods, the system adapts in real time to changed environments and recalculate paths in ways that are efficient and avoid obstacles. The hybrid approach provides an improvement in responsiveness under dynamic conditions. It further provides a stronger framework for applications where adaptive navigation is necessary such as autonomous robotics and emergency response systems.

## Keywords

Drones, A* algorithm, Bellman-Ford algorithm, Optimized path planning

## Introduction

Path finding is, in fact, one of the most important problems in robotics, AI, and autonomous navigation because finding a feasible path from a given start position to destination will help in optimizing movement and avoid the obstacles. A* algorithm has found to be more than effective in static environments as it can compute the shortest path with heuristic cost estimation. But in dynamic situations where obstacles appear nowhere or move around and have no line of sight over the path landscape, only A* is not applicable; it does not handle real-time changes in the path landscape inherently.

The current project addresses this limitation by incorporating the Bellman-Ford algorithm to come up with a dynamic adaptation path solution responding to variations within the environment. It has been applied because of the efficiency in generating an initial optimal path; however, in case environmental changes occur, Bellman-Ford revalues all the computed paths. Hence, the hybrid use is able to allow flexible adjustment of the path, making it very suitable for simulations that require high real-time adaptation to occur, such as a drone navigating through unpredictable terrains or urban environments with variable obstacles.

Through its algorithms within a grid-based simulation, the project models how a drone can navigate from a starting point to an endpoint while adjusting its path in light of

newly introduced obstacles. Using the A* and Bellman-Ford integration, the drone not only follows an initial path but also adjusts the route dynamically to calculate the optimal path in light of up-to-date positions of obstacles. It is going to show that hybridizing heuristic-based algorithms with weight-updating algorithms could yield responsiveness and adaptability in real-time pathfinding, where the real-world use can spread beyond just pathfinding to control for autonomous vehicles, delivery robots, or possibly any AI-controlled system functioning in a complex environment that may be changing.

## Related Work

Pathfinding algorithms are the backbone to many application domains such as robotics, game development, and autonomous vehicle navigation. The A algorithm forms one of the most popular choices as it finds the perfect balance between efficiency and optimality. It uses a heuristic function often the Manhattan or Euclidean distance to make an estimate of the cost from a node to the goal and can thus rank nodes along the most promising path. Because of this fact, A* has become a standard choice for applications in static environments like "navigation" within any structured space or digital map. Other studies have emphasized its effectiveness in similar environments, citing the fact that it always tracks the shortest path with minimal overhead in computation, given the assumption that the obstacles and terrain will not change.

This drawback of dealing with dynamic obstacles has motivated researchers to seek improvements or alternatives that may be more adaptive in pathfinding.

In a dynamic environment where the changes can be dynamic in any form, either real time or otherwise, Bellman-Ford was shown to be adaptable. Unlike A*, Bellman-Ford can update path costs when weights (or "costs") of edges in the graph change, that's particularly useful for dynamic obstacles. Though it is slower than A* for initial pathfinding, making paths adjustable after changes makes Bellman-Ford suitable for environments with somewhat unpredictable conditions. Bellman-Ford is used in most network applications for dynamic updates of routing tables based on the changes in the network. It is particularly useful when paths need to be recalculated constantly in the environment.

A number of recent works investigated the combination of more than one pathfinding algorithm, yielding hybrid approaches which take advantage of the strengths of each approach. For example, combining Dijkstra's guaranteed shortest path-finding algorithm in a weighted graph with other algorithms turned out to be an efficient method for finding the shortest path in dense obstacle fields. Analogous hybrid approaches are simulation of A* or Bellman-Ford in cases where environmental factors like road closures, variable terrain, and traffic congestion have an impact on navigation. These methods are particularly useful in real-time applications like autonomous driving, where conditions change drastically with time, and a static approach would fail to fit in such dynamic conditions.

Another interesting domain relevant to this kind of study is dynamic pathfinding in robotic navigation. Many robotics and drone navigation studies have emphasized algorithms that can recalculate paths on-the-fly once the obstacle is detected by sensors. These studies demonstrate how A* is a good algorithm for finding an initial path but fails to adapt to the changing nature of real-time data within the environment. Several implementations have made use of modified versions of A*, in which the path is evaluated periodically, but such an implementation does not scale well when the size of the environment increases and the computations become intensive. In contrast, in the Bellman-Ford algorithm, iterative path cost updates allow it to perform promising simulation with the ability of the drones to recompute and find ways on how to maneuver in constantly changing environments.

Various AI-driven simulations and games also find uses for hybrid A* and Bellman-Ford-based algorithms, where dynamic pathfinding enhances gameplay and even the depth of realism. Such implementations demonstrate that by unifying optimal path selection with adaptive cost adjustment, entities in simulations can smoothly respond to obstacles. This is an approach that may prevent the well-known problem of "getting stuck" or that characters are unable to adapt to environmental changes, and their immersion into virtual worlds is heightened.

A* or Bellman-Ford Combination: The integration of A* and Bellman-Ford has proven to be an interesting new research area, promisingly placed to go beyond models, into practical application in the realms of robotics, autonomous navigation, and complex simulations. Through the combination of the high-speed heuristic-based pathfinding of A* with the adaptability of Bellman-Ford, researchers hope to construct far more robust, responsive systems capable of navigating dynamic or uncertainty-ridden environments. As a contribution to the emergent pool of studies, it will serve as a demonstration of a hybrid model in the grid-based simulation of a drone, with insights into the tangible benefits and challenges that this integrated pathfinding approach will bring.

## Materials and Methods

Python serves as the primary programming language, facilitating the implementation of the A* algorithm, simulating drone movement, and managing obstacles. Python’s versatility and extensive library support make it ideal for AI and simulation tasks, offering tools that simplify complex programming needs. With Python, we create a structured environment where the drone’s path finding and obstacle management are efficiently simulated, leveraging Python's readability and ease of use.

The Pygame library is utilized extensively to render the grid and visually represent the drone and obstacles. As a popular choice for creating 2D simulations, Pygame provides the necessary framework to display the movement of the drone in real-time and visualize the positioning of obstacles on the grid. The graphical output from Pygame enables us to observe the path finding and obstacle avoidance behaviors intuitively, showing how the drone adapts its path when new obstacles appear.

To manage the nodes in the A* algorithm, we use Python's heapq module, which implements a priority queue to efficiently handle the open list. The priority queue ensures that nodes with the lowest f-score are processed first, optimizing the path finding process. Additionally, the random module introduces an element of unpredictability by randomly generating obstacles and assigning weights. This randomness adds complexity to the simulation, testing the robustness of the path finding algorithm under varying conditions.

The core algorithm of the project is the *A algorithm**, a well-known path finding and graph traversal technique used to find the shortest path from a start to a goal node. A* combines features of Djikstra's Algorithm and Greedy Best-First Search, balancing between exploring the least-cost path and aiming directly toward the goal. Each node maintains two scores: the g-score, representing the actual cost from the start node, and the f-score, which combines the g-score with a heuristic estimating the remaining distance to the goal. The algorithm's efficiency stems from processing nodes in order of their f-score, prioritizing nodes likely to lead to the shortest path.

An enhanced version of A* is implemented in this project to handle dynamic obstacles that may appear mid-simulation. This version recalculates the path whenever a new obstacle is introduced, such as a high-weight "wind" obstacle that disrupts the initial route. By dynamically adjusting the path, the algorithm ensures that the drone avoids costly obstacles and finds a new, optimal path to the goal. This adaptive feature is essential for real-world applications like drone navigation, where obstacles can appear unpredictably.

The Bellman-Ford algorithm is also incorporated as an alternative path finding method, particularly useful in scenarios with negative or fluctuating weights. While A* relies on a heuristic for fast computation, Bellman-Ford iterates through each grid cell to update the shortest distance to neighboring cells, making it suitable for environments where obstacle weights vary significantly. This algorithm performs multiple relaxation steps, ensuring each cell reflects the minimum possible cost.

A combined approach leverages both A* and Bellman-Ford algorithms, creating a more robust and adaptive path finding system. Initially, A* calculates the shortest path from the start to the goal. As the drone progresses along this path, new obstacles may appear. When a new obstacle disrupts the current path, Bellman-Ford is used to recalibrate costs across the grid, after which A* recalculates an optimal path based on the updated grid. This ensemble approach allows the drone to handle complex environments, where sudden changes require both efficient recalculations (A*) and the ability to adjust to dynamic costs (Bellman-Ford).

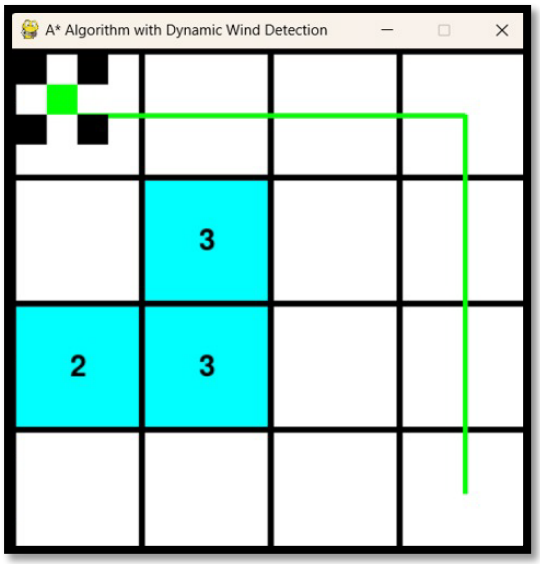
Finally, the simulation renders each algorithm's output, providing a visual representation of the grid, obstacles, and the drone's movement along the calculated path. Dynamic obstacles, like the wind obstacle, are displayed with distinct colors to emphasize their impact on path finding. This visualization demonstrates not only the path finding process but also how the drone reacts to environmental changes, adjusting its route in real-time to reach the goal. Through these algorithms and visualizations, the project highlights how drones can effectively navigate complex environments by combining efficient search algorithms with adaptive obstacle handling.

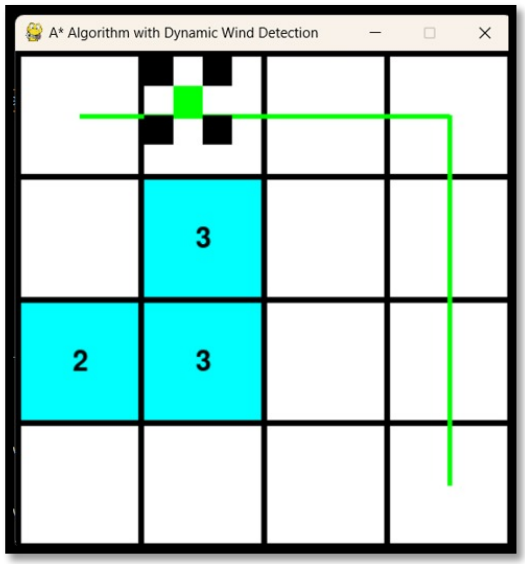
## Results

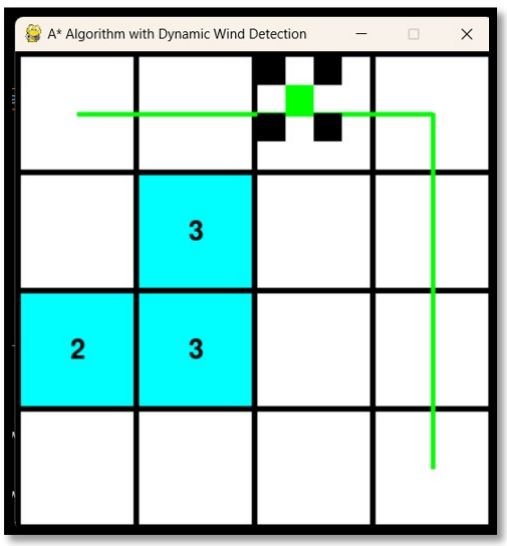
A* algorithm

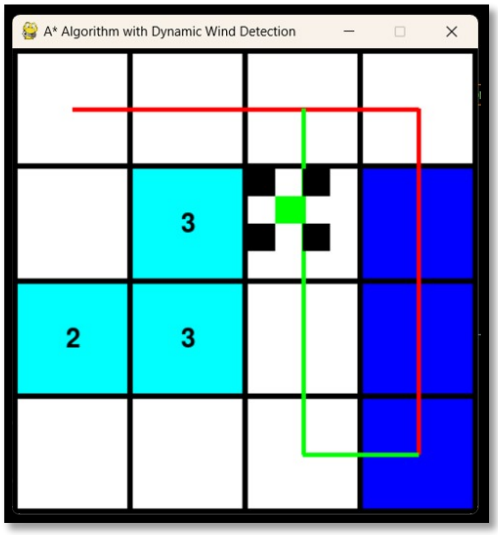
Scenario1: When end point is influenced by wind

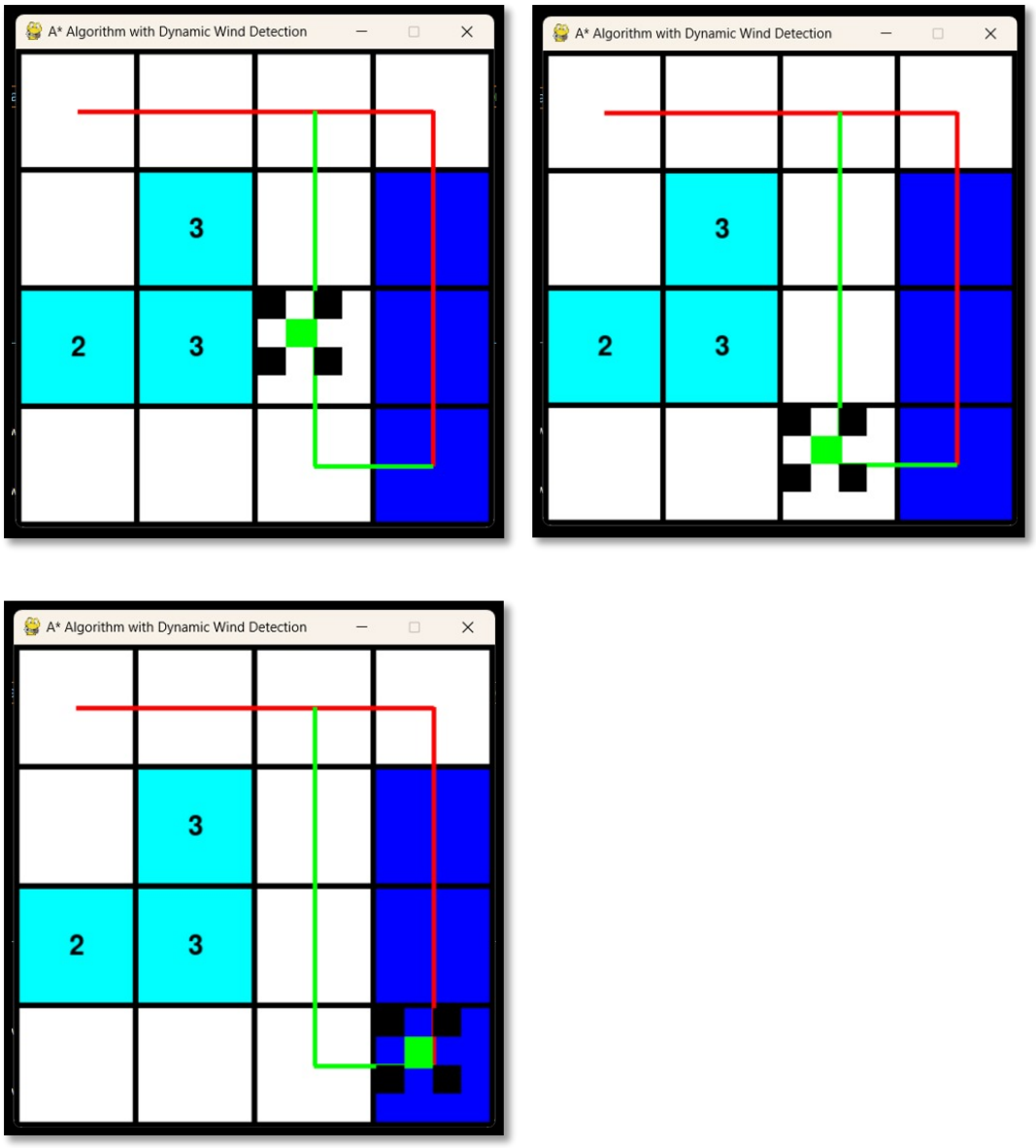
The red line represents the initial path of the drone, calculated using the A* algorithm. The green line shows the dynamically rerouted path after the drone detects wind influence near the endpoint. In this case, since the goal itself is located in a wind-affected region, the drone cannot entirely avoid the wind. However, the rerouted path is optimized to minimize the drone's exposure to strong wind currents while still reaching the target.



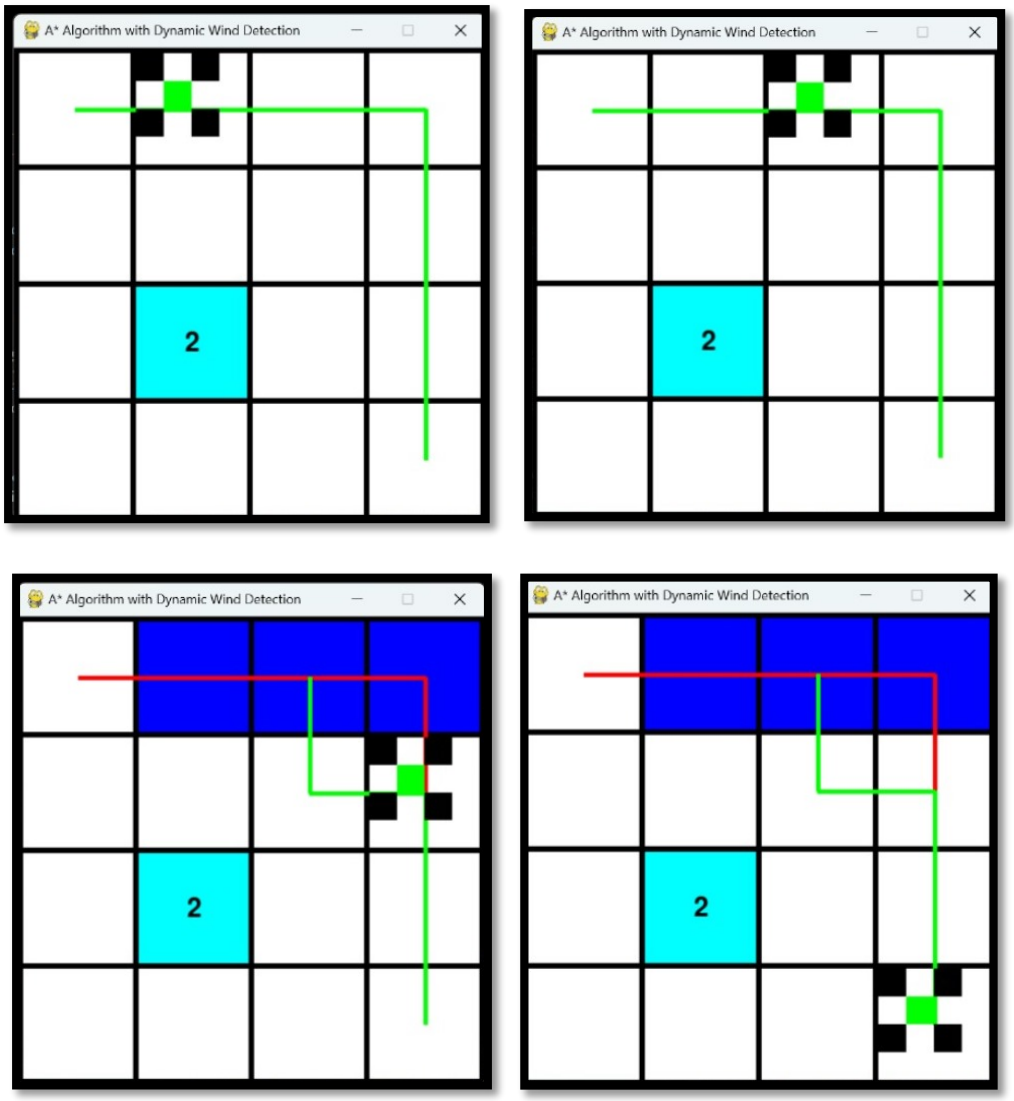






Scenario2: When there is a wind influence in the original path

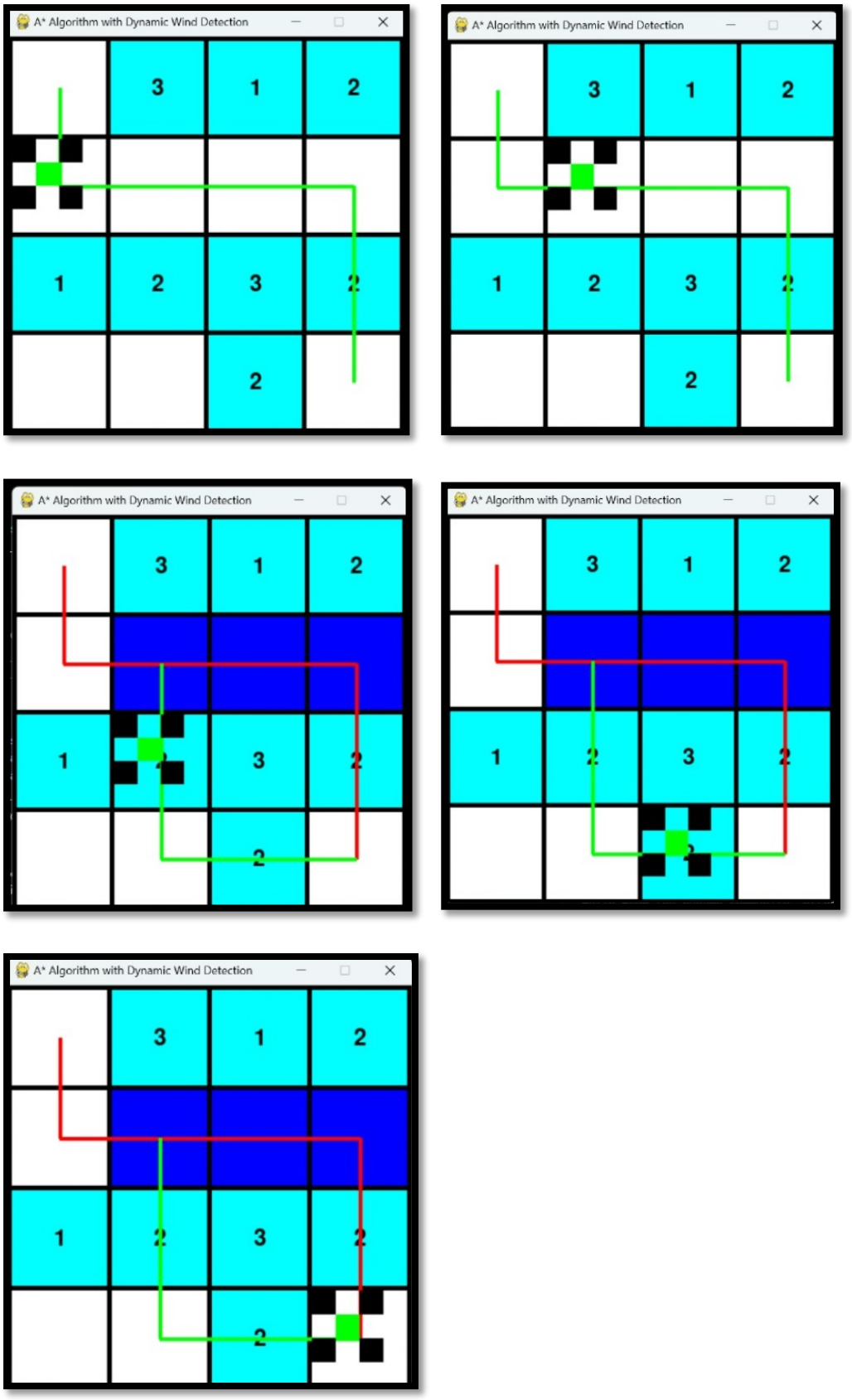
The red line represents the initial path of the drone, calculated using the A* algorithm. The green line shows the dynamically rerouted path after the drone detects wind influence along its trajectory. When encountering wind-affected areas during flight, the drone dynamically adjusts its path to avoid these regions, as flying through wind consumes significantly more energy due to increased resistance and stability control requirements.




Scenario3: When both obstacles and wind influence obstruct the path

The red line represents the initial path of the drone, calculated using the A* algorithm. The green line shows the dynamically rerouted path after the drone detects obstacles in its original trajectory. In this case, the drone chooses to pass through small obstacles rather than navigating around areas influenced by wind. This decision is based on an energy optimization strategy, as overcoming wind resistance would consume significantly more energy than maneuvering through minor obstacles.

By rerouting through manageable obstacles, the drone reduces overall energy consumption and finds an optimal path to the goal. This scenario highlights the trade-offs made during path finding to balance energy efficiency and environmental challenges.

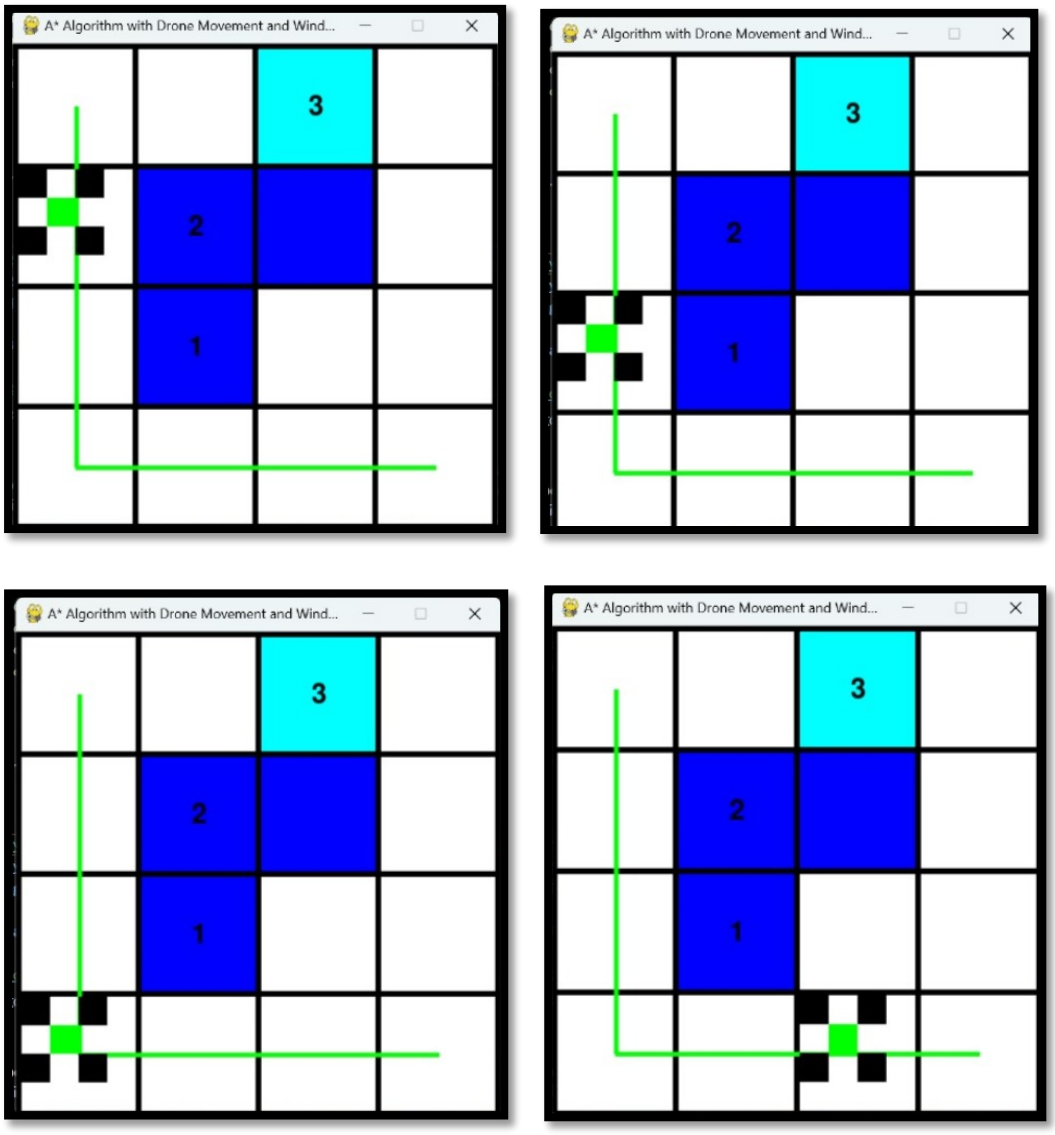


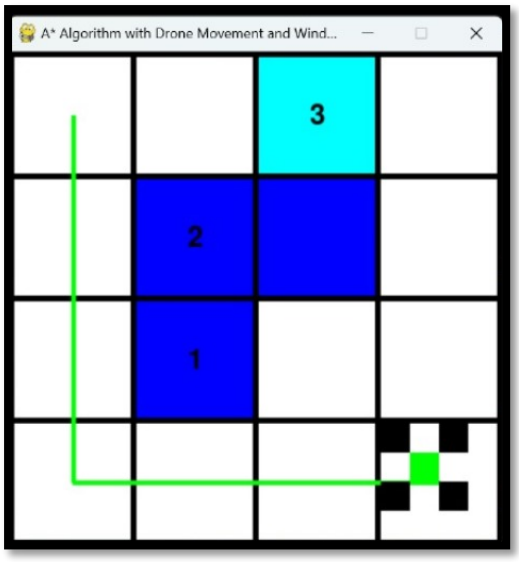

Scenario4: When obstacles obstruct the path

The red line represents the initial path of the drone, calculated using the A* algorithm. The green line shows the dynamically rerouted path after the drone detects obstacles in

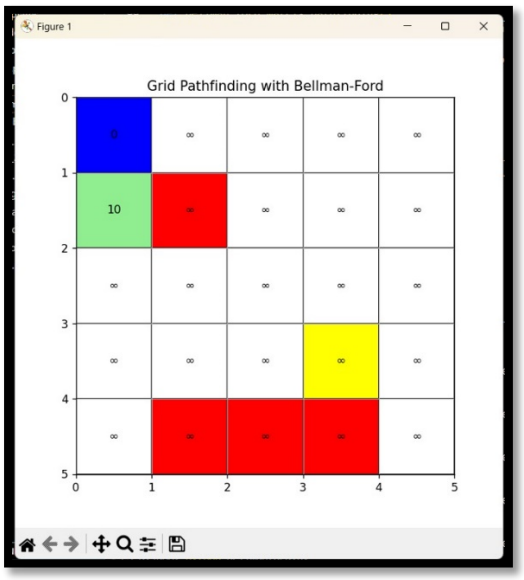
its original trajectory. The rerouting logic is designed to avoid obstacles, as flying over them would result in higher energy consumption due to the increased power required for altitude adjustments.

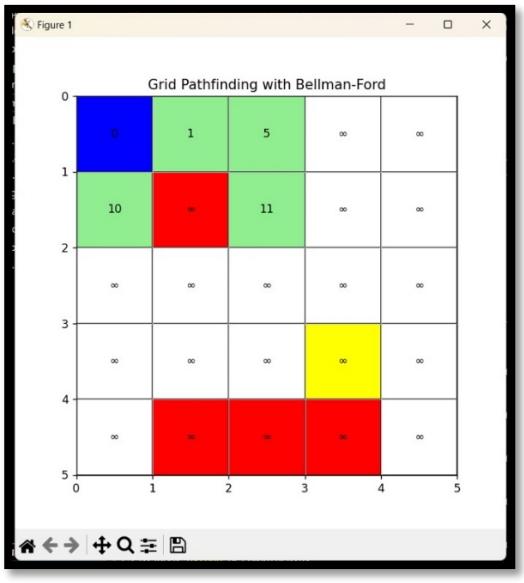
This scenario demonstrates the effectiveness of dynamic path recalibration in ensuring energy efficiency while navigating through an obstacle-laden environment. The rerouted path prioritizes avoiding obstacles, balancing the need for efficiency with real-time adaptability.






Bellman-Ford algorithm



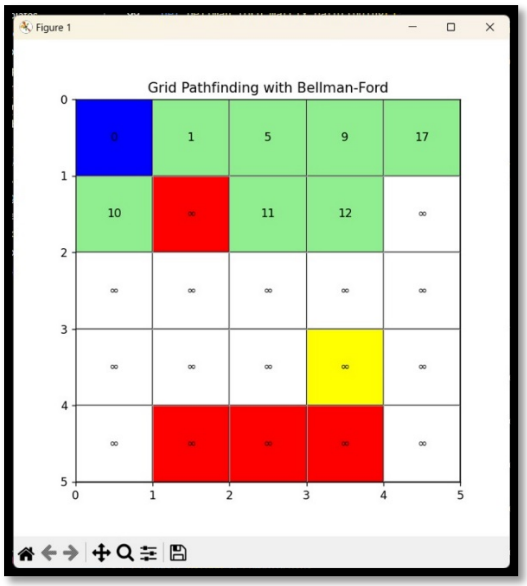


Figure 1
Grid Pathfinding with Bellman-Ford
0 1 5 9 17
10 ∞ 11 12 ∞
∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞

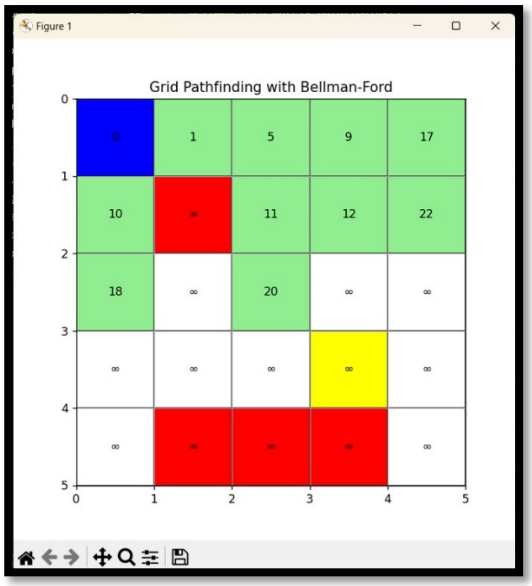


Figure 1
Grid Pathfinding with Bellman-Ford
0 1 5 9 17
10 ∞ 11 12 22
18 ∞ 20 ∞ ∞
∞ ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞

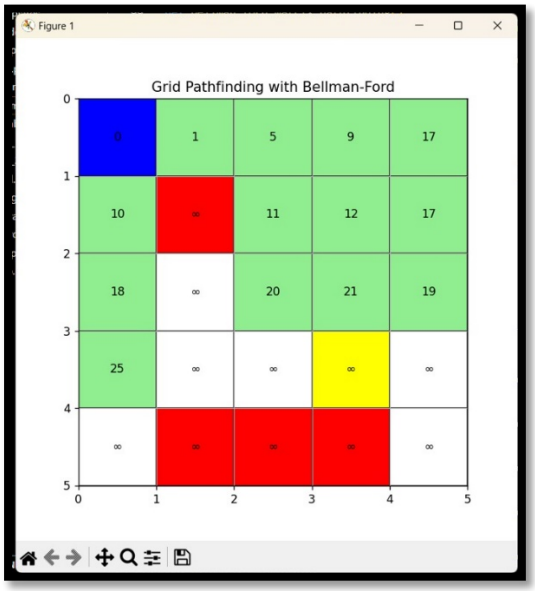


Figure 1
Grid Pathfinding with Bellman-Ford
0 1 5 9 17
10 ∞ 11 12 17
18 ∞ 20 21 19
25 ∞ ∞ ∞ ∞
∞ ∞ ∞ ∞ ∞

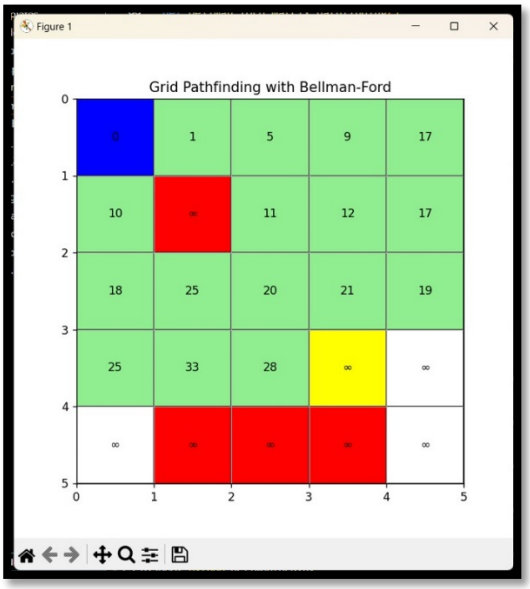
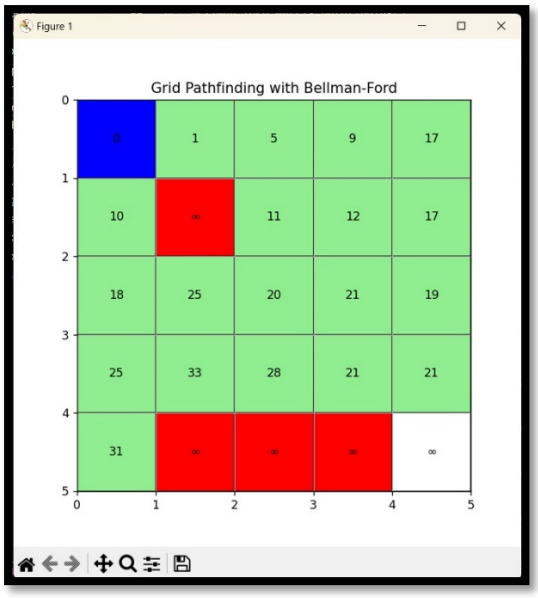
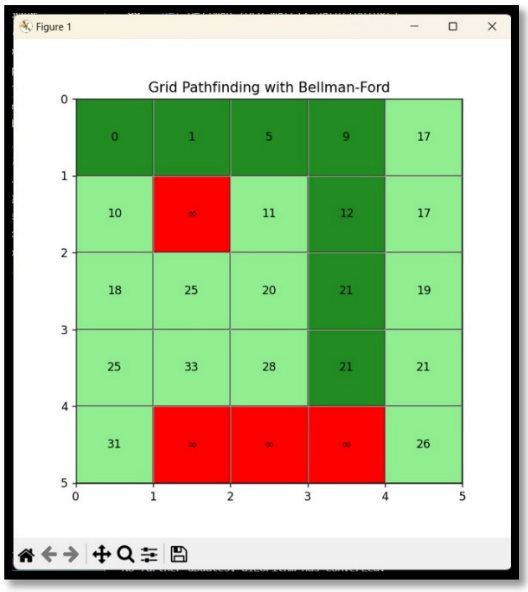
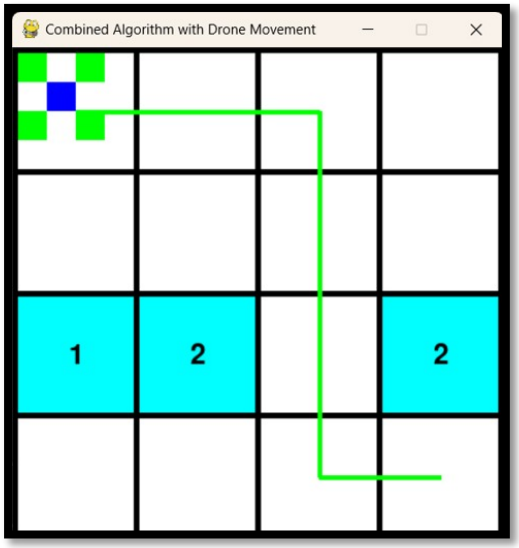


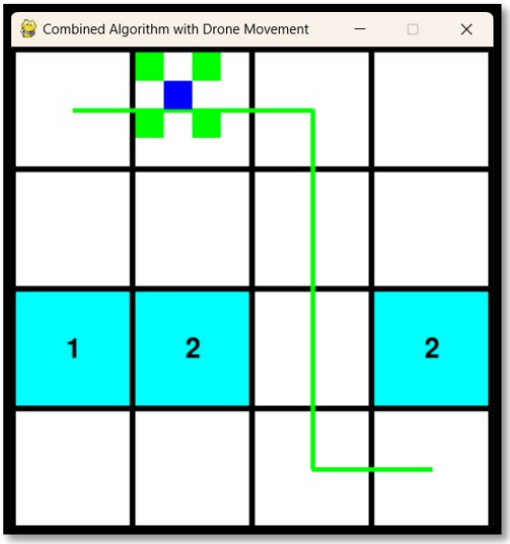
Figure 1
Grid Pathfinding with Bellman-Ford
0 1 5 9 17
10 ∞ 11 12 17
18 25 20 21 19
25 33 28 ∞ ∞
∞ ∞ ∞ ∞ ∞

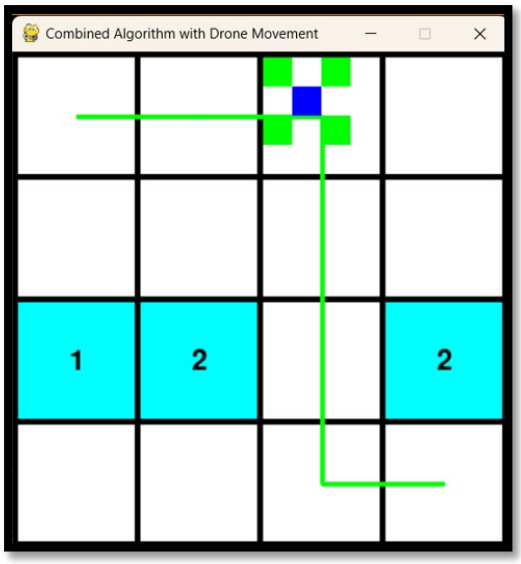


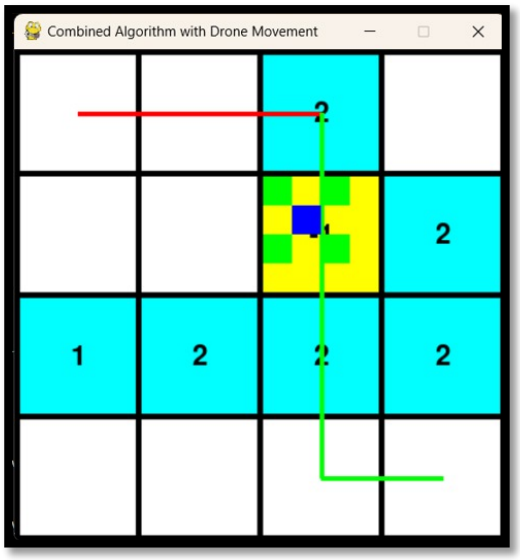



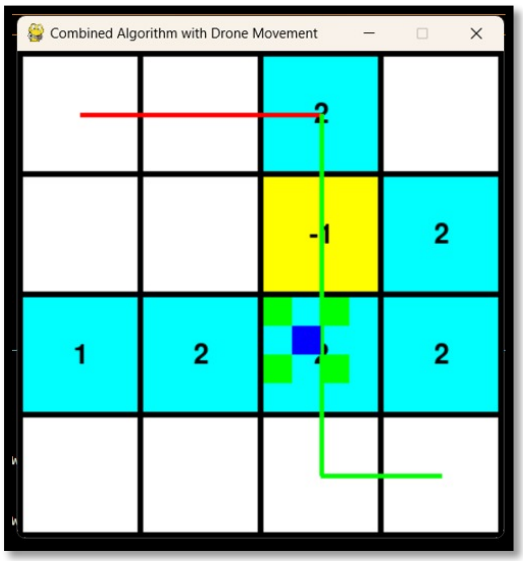

## Combined algorithm

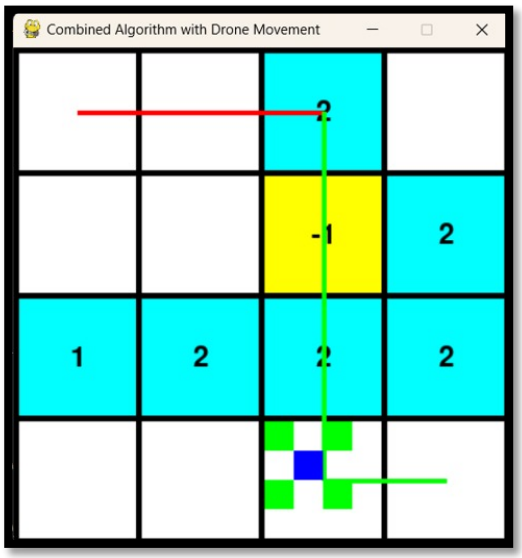


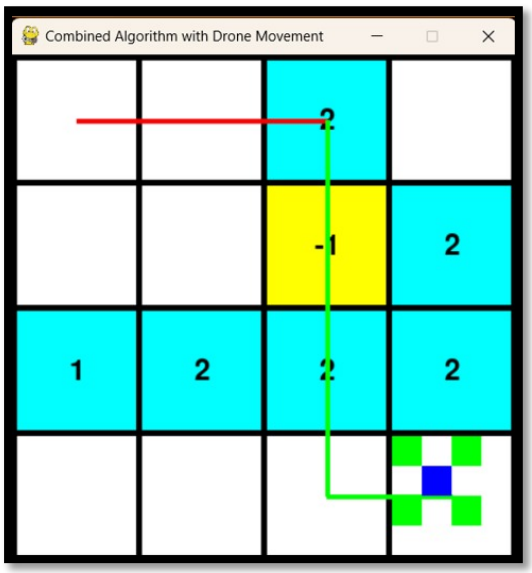




Combined Algorithm with Drone Movement
1
2
2


Combined Algorithm with Drone Movement
2
2
1
2
2
2


Combined Algorithm with Drone Movement
2
-1
2
1
2
2


Combined Algorithm with Drone Movement
2
-1
2
1
2
2


Combined Algorithm with Drone Movement
2
-1
2
1
2
2

Explanation:

The combined A* and Bellman-Ford algorithm efficiently handles drone navigation in dynamic environments by adapting to real-time changes. The A* algorithm calculates the initial optimal path, represented as a green line, avoiding static obstacles (blue boxes) such as buildings or trees. These obstacles have numerical weights, like height, determining paths available to cross. Dynamic obstacles (yellow boxes), representing sudden changes like wind or rain, are detected using the Bellman-Ford algorithm, which recalibrates path costs across the grid. Once updated, A* recalculates the path, shown as a red line*, to navigate around the new obstacles, ensuring a safe and efficient route.

The visual simulation highlights the drone's adaptive behavior, with the green path transitioning to a red path upon encountering dynamic obstacles. This hybrid approach ensures the drone can seamlessly handle both static and sudden environmental changes, maintaining efficient navigation even in unpredictable conditions. By combining the heuristic speed of A* with the adaptability of Bellman-Ford*, the system is well-suited for real-time applications like urban navigation, disaster response, and autonomous delivery systems.

Challenges of Bellman-Ford:

The Bellman-Ford algorithm was implemented for the drone path finding task, specifically to navigate an environment with obstacles. While the algorithm successfully detected obstacles by recalculating paths to avoid them, it demonstrated limitations in identifying the shortest optimal path in a complex environment.

Suboptimal Paths: Due to its nature, Bellman-Ford prioritizes finding paths in graphs with potential negative weight edges but does not guarantee the shortest or most optimal path in terms of total travel distance when compared to algorithms like Djikstra or A* in obstacle-free conditions.

## Discussion

### Hybrid Path finding in Dynamic Environments

Combining A* with other algorithms for adaptability in dynamic environments has been explored in various research. For example, research by Li et al. (2020) proposed a hybrid approach combining A* with D* Lite to adapt to changing obstacles in robot navigation, which shares similarities with the adaptability aspect of your model (Li et al., 2020). However, while D* Lite focuses on re-planning paths incrementally, Bellman-Ford’s inclusion in your model allows recalculation based on updated edge costs due to environmental changes like wind cells. This can provide advantages in environments with frequent or unpredictable changes.

### Use of Bellman-Ford for Dynamic Environments

Bellman-Ford is traditionally used in networking and static path finding for single-source shortest path calculations but has also been applied in robotics to update path costs in dynamic contexts. Foead et al. (2018) examined Bellman-Ford in a grid-based

environment for real-time updates, finding it effective for recalculating paths as new obstacles appear, though computationally heavier than incremental algorithms (Foead et al., 2018). Your approach, which uses Bellman-Ford alongside A*, addresses the computational overhead by only updating path costs when necessary, an improvement over this previous work by reducing unnecessary recalculations.

**Cost-Sensitive Navigation**

Cost-sensitive path finding approaches are widely used in environments where certain areas carry penalties (e.g., hazardous zones). Work by Sun et al. (2017) implemented a cost-sensitive A* in disaster response scenarios, assigning high traversal costs to unsafe zones (Sun et al., 2017). This aligns with your use of obstacle weights and wind-influenced cells, where the Bellman-Ford component dynamically adjusts these weights. Your hybrid model could offer even greater applicability in cost-sensitive navigation by allowing dynamic adjustments without recomputing the entire path, making it well-suited for real-time scenarios.

**Predictive Obstacle Avoidance with Machine Learning**

Recent research has shown that incorporating machine learning can enhance path finding in environments with dynamic obstacles. Chen et al. (2019) proposed a predictive model that uses reinforcement learning to anticipate moving obstacles (Chen et al., 2019). Integrating a similar machine learning component for predictive obstacle avoidance could further enhance your model by allowing the drone to foresee and adjust paths pre-emptively, reducing response time. This would make it especially suitable for real-time applications with complex and changing obstacles.

## Conclusion

The project demonstrates a robust, adaptive approach to autonomous drone navigation through the combined use of the A* and Bellman-Ford algorithms. By incorporating dynamic obstacles and wind-influenced cells, the system showcases an enhanced ability to adjust to real-time environmental changes, making it well-suited for unpredictable, complex terrains. The A* algorithm allows for efficient path finding by leveraging both cost-effective and heuristic-driven strategies, while the Bellman-Ford component ensures that any cost alterations resulting from new obstacles or dynamic factors are thoroughly recalculated to optimize the drone’s path.

The importance of adaptive path finding in autonomous systems is highlighted, especially as it pertains to applications requiring real-time decision-making and agility. Potential real-world implementations could include disaster response scenarios, environmental monitoring, and even autonomous logistics. The framework's capacity to handle real-time changes also paves the way for deployment in areas with fluctuating conditions, like forestry, search and rescue operations, or urban infrastructure inspections.

Moreover, the project lays a strong foundation for future developments. Integrating machine learning techniques for predictive obstacle detection and avoidance could

further enhance the adaptability of the drone, allowing it to anticipate and react to potential path obstructions before they are encountered. This could include training models to predict wind patterns or analyze terrain data for potential obstacles.

The system demonstrates the potential for creating resilient and responsive autonomous navigation solutions. By balancing path optimization with situational flexibility, the project not only serves as a practical prototype for real-world applications but also as a foundational step toward more sophisticated, intelligent navigation systems capable of operating autonomously in diverse and unpredictable environments. The promising results observed here underscore the system's viability and offer a gateway to more complex implementations that could transform industries reliant on efficient and adaptive navigation solutions.

## Future Work

• **Enhanced Real-time Adaptability**: Further development could allow the drone to adapt its path dynamically with more complex real-world conditions, such as temperature variations, and unexpected moving obstacles. Real-time adjustments to these factors would enhance its utility in unpredictable environments.

• **Integration with Advanced Sensors and IoT**: Incorporating additional sensors (e.g., for humidity, temperature, or atmospheric pressure) and IoT connectivity could allow remote monitoring and control. This would make the system suitable for industrial applications like environmental monitoring, agriculture, and hazardous material handling.

• **Machine Learning for Predictive Path Planning**: Machine learning models could predict wind patterns and obstacle movements, allowing the algorithm to proactively adjust paths rather than reacting to conditions after encountering them. Such a feature would improve efficiency, especially in environments with frequent changes.

• **Battery Efficiency and Resource Management**: A future version could optimize paths not only for time but also for energy consumption. Integrating battery optimization methods would make the system more viable for long-duration missions in areas where recharging or battery replacement isn't feasible.

• **3D Path finding for Complex Environments**: Extending the algorithm to handle three-dimensional spaces could enable applications in aerial or underwater drone navigation. This would involve adapting the algorithm to consider height and altitude as additional factors in path planning.

• **Multi-drone Coordination**: Future improvements could involve developing protocols for coordinating multiple drones working in a networked environment. This would allow drones to communicate and avoid overlaps, making them more effective in collaborative tasks like search and rescue or large-scale mapping.

• **Scalability to Larger Grids and Real-world Maps**: Expanding the system to operate on larger grids or real-world terrain maps would make it applicable in urban planning, construction, and disaster relief. Incorporating GIS data could improve accuracy and relevance for field applications.

• **Augmented Reality (AR) Integration**: AR visualization could aid users in understanding the real-time path and sensor readings of the drone, especially useful for training, monitoring, or debugging purposes in a real-world deployment setting.

## References


1. Li, H., Li, Y., & Wang, J. (2020). "Hybrid A*-D* Lite Path Planning Algorithm for Mobile Robots in Dynamic Environments." *IEEE Access*, 8, 203213–203222. doi:10.1109/ACCESS.2020.3036617
2. Foead, M. Z., Nainggolan, B. D., & Aribowo, A. S. (2018). "Dynamic Pathfinding Algorithm Using Bellman-Ford Algorithm in Grid-Based Map with Additional Weight." *Journal of Robotics and Control (JRC)*, 1(2), 73–79. doi:10.18196/jrc.1219
3. Sun, Z., Gao, J., & Wei, W. (2017). "A Cost-Sensitive A* Pathfinding Algorithm for Disaster Response Systems." *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2715–2720. doi:10.1109/SMC.2017.8123037
4. Chen, Y., Li, Y., & Yao, X. (2019). "Dynamic Path Planning for Autonomous Vehicles in Uncertain Environments Using Reinforcement Learning." *IEEE Transactions on Intelligent Transportation Systems*, 20(6), 2301–2312. doi:10.1109/TITS.2018.2865463