UAV Trajectory Optimization with Machine Learning

Emerson Hall

Ramana M. Pidaparti

24 November 2025

Table of Contents

ABSTRACT **3**

INTRODUCTION **4**

BACKGROUND **5**

METHODS **7**

Simulation Environment 7

Baseline Trajectory Planner and Controller 8

Dataset Generation and Machine Learning Model 9

Multi-UAV Coordination and Wind Handling 10

Week 11 Validation Framework and Final Code 11

RESULTS **14**

Urban Terrain 14

Natural Terrain 16

Manmade Terrain 20

Overall Comparison 23

DISCUSSION **24**

FUTURE WORK **26**

CONCLUSION **28**

Abstract

This project explored how machine learning can be integrated with a classical trajectory planner to control unmanned aerial vehicles (UAVs) in cluttered and windy environments. Over twelve weeks, the work progressed from a baseline A* planner in a simple 2D world to a full simulation framework with three UAVs, three terrain types, and four wind profiles. A supervised learning model was trained to imitate the baseline controller using a dataset generated from planned trajectories. The final system compared a baseline mode, which used only classical planning and control, to a machine learning enhanced mode that replaced the core path following behavior with a learned controller. Week 11 focused on deterministic validation across urban, natural, and manmade terrains under calm, breeze, gusty, and storm conditions. The results showed that both modes performed similarly in calm scenarios, while the machine learning mode was often faster in mild wind but less reliable in high wind. Minimum obstacle clearance remained comparable for both controllers. Overall, the project demonstrated that a behavior cloned controller can match classical performance in easy conditions, but the baseline A* driven approach remains more robust when disturbances increase or the geometry becomes more complex.

Introduction

Autonomous UAVs are expected to navigate complex environments while avoiding obstacles, compensating for wind, and maintaining safe distances from structures and other vehicles. Traditional path planning algorithms such as A* can generate feasible trajectories on a grid map, but they can be conservative and may require frequent replanning when the environment changes. At the same time, machine learning methods have shown promise for learning control policies directly from data, but questions remain about how reliable they are compared to classical methods, especially in safety critical situations.

This project addressed those questions in the context of a semester long independent study centered on UAV trajectory optimization. The work was organized into weekly milestones that gradually increased the complexity of the system. Early weeks focused on building a robust baseline planner and simulation environment. Middle weeks introduced dataset generation, feature engineering, and supervised learning to imitate the baseline controller. Later weeks integrated the learned model into a real time simulation, added a wind model, and extended the system to multiple UAVs operating in a shared airspace.

Week 11 represented the final testing and validation stage. The goal was to systematically compare the baseline and machine learning enhanced controllers across multiple terrains and wind conditions using a deterministic framework. Week 12 brings all of this work together in a single final report and presentation. This report summarizes the overall methodology, describes the final simulation and validation scripts, presents key results, and highlights lessons learned and directions for future work.

Background

Path planning for UAVs in structured or semi structured environments often starts with a grid-based planner such as A*. A* searches for a least cost path from a start cell to a goal cell based on a combination of distance and heuristic estimates. When the grid is built from an occupancy map that marks obstacles as blocked cells, the planner can guarantee collision free paths as long as the map is accurate and the UAV tracks the path well. In practice, a raw grid path can be jagged and inefficient, so additional smoothing and interpolation are usually applied to make the trajectory more realistic and easier to follow.

Classical controllers for trajectory following are often designed in a “Frenet frame,” where motion is decomposed into progress along the path and lateral error relative to it. This makes it possible to correct cross track error smoothly while still moving forward toward the goal. Additional safety can be achieved by adding repulsion forces derived from distance fields so that the UAV naturally moves away from obstacles, combined with braking logic that slows down motion whenever a potential collision is detected.

Machine learning can augment or replace parts of this pipeline. In behavior cloning, a supervised model is trained to reproduce actions taken by an expert controller given local observations. If the dataset covers enough of the relevant state space, the model can learn to approximate the policy without needing explicit search at test time. In this project, the expert was the baseline A* planner combined with a hand tuned Frenet controller. The machine learning model was trained to predict velocity commands based on engineered features that encode goal direction, obstacle distances, and local motion information.

The central question is not only whether the learned controller can imitate the expert on the training data, but also how it behaves when embedded in a full closed loop simulation with

wind and complex map geometry. This motivated the final Week 11 validation framework that directly compared the baseline mode to a machine learning enhanced mode under identical test conditions.

Methods

Simulation Environment

The simulation environment is a two-dimensional square world that spans 20 by 20 meters. The continuous space is discretized into a grid using a configurable resolution, with 0.20 meters used in the final code. The grid is used for both planning and collision detection. Obstacles are represented as polygonal shapes that are rasterized into an occupancy mask. A second, inflated occupancy mask is generated by dilating the obstacles with a disk-shaped structuring element. This “thick” occupancy grid enforces safety margins around obstacles when planning and checking collisions.

Three terrain types were implemented to provide different navigation challenges. The urban terrain consists of a symmetric grid of rectangular building blocks, with a cross shaped road network in the center. Buildings are placed in all blocks that are not part of the main vertical or horizontal roads. The natural terrain includes a central pond modeled as an ellipse, along with randomly placed rocks or logs and trees that avoid the pond region. The manmade terrain is structured like a harbor, with a long water strip, piers extending into the land, a quay road, and several warehouses placed on the right side of the map.

Wind is represented as a combination of a background vector and a stochastic gust component. Four wind profiles were defined: calm, breeze, gusty, and storm. Each profile specifies the base wind vector, gust variance, correlation time, and limits on airspeed and minimum ground progress. The gust component evolves as an Ornstein Uhlenbeck process so that it changes smoothly rather than randomly jumping between values. The final simulation uses these wind profiles to generate realistic disturbances that vary over time.

Baseline Trajectory Planner and Controller

The baseline mode uses a classical A* planner operating on the inflated occupancy grid. Each UAV has a start and goal location that are guaranteed to be connected by free space using a breadth first search check. The planner allows motion to all eight neighbors of a grid cell, with diagonal moves having a higher cost. A heuristic based on Euclidean distance is used to guide the search and keep it efficient.

After A* returns a path in grid coordinates, the path is converted to world coordinates and passed through a string pull smoothing procedure. The string pull algorithm removes unnecessary intermediate nodes as long as straight line segments between remaining waypoints remain free of obstacles in the thick occupancy grid. A final shortcut step attempts to replace sequences of segments with longer segments whenever the straight line remains collision free. This produces paths that are shorter and smoother than the original grid path while preserving safety.

For tracking, the baseline controller uses a Frenet frame representation. At each time step, the controller computes the closest point on the smoothed path, the tangent direction along the path, and the lateral error perpendicular to it. The desired direction of motion is a combination of the path tangent and a lateral correction term proportional to the cross-track error. The controller also blends this direction with a vector pointing directly to a lookahead waypoint that lies a fixed distance ahead along the path. The lookahead distance is slightly larger in the machine learning mode but similar in structure.

To improve safety, a distance transform of the thin occupancy map is computed, and the gradient of this distance field is used to define a repulsion direction. When the UAV gets close to obstacles, the controller smoothly blends between pure path following and a repulsive sliding

direction that keeps it away from walls and building edges. Ray casting is used to measure the distance to obstacles in a set of directions around the UAV. If any ray detects an obstacle within a threshold, the desired speed is reduced using a braking gain.

The baseline also includes several robustness features. Sub step integration is used when the proposed motion in a single time step is large relative to the grid resolution. If a segment between two consecutive states intersects the thick occupancy grid, a bisection search finds the collision point, and the controller attempts to slide along the obstacle rather than passing through it. Starts and goals are kept out of obstacles, and a stall recovery mechanism prevents the UAV from staying stuck in one region for too long.

Dataset Generation and Machine Learning Model

The machine learning component was built from Weeks 6 through 8. In Week 6, trajectories generated by the baseline planner and controller were used to construct a dataset. Each sample in the dataset consisted of a feature vector describing the local state of the UAV and the environment, along with the corresponding expert velocity command. Features included the relative position to the goal, components of the current velocity, local obstacle distance information, and wind related terms. The features were normalized, and the dataset was split into training, validation, and test sets.

In Week 7, a behavior cloning model was trained using ridge regression. The problem was posed as a multi output regression task that mapped the feature vector to the three components of the velocity command. Ordinary least squares failed due to singularities caused by collinear or constant features, but ridge regression resolved this by adding a small regularization term. The model achieved effectively zero mean squared error and perfect cosine

similarity between predicted and true velocities on the training, validation, and test splits. This confirmed that the feature engineering and preprocessing steps were consistent.

Week 8 focused on systematic evaluation and hyperparameter tuning. The ridge coefficient was swept over a logarithmic range, and validation root mean squared error was used to pick the best value. The final model maintained extremely low error and near perfect directional agreement across all splits. A strict success metric based on an error tolerance showed high success rates, which matched the behavior of the baseline planner on the deterministic trajectories used for dataset generation.

In the final system, the machine learning mode uses this trained model to replace the core path following step. Instead of using the hand tuned Frenet controller alone, the simulation computes the engineered features at each step and feeds them to the learned model to obtain velocity commands. The rest of the pipeline, including planning, smoothing, collision checks, and wind compensation, remains the same.

Multi-UAV Coordination and Wind Handling

Weeks 9 and 10 extended the single UAV system to handle multiple agents in the same environment and to make the wind integration more realistic. The simulation supports two or three UAVs that all share the same static geometry and moving obstacles. Each UAV maintains its own planned path and smoothed trajectory but reasons about other UAVs as moving obstacles during planning.

To accomplish this, a shared occupancy grid is maintained, and a separate dilation around each UAV position creates local keep out zones. When a UAV calls A*, it uses a planning grid that combines static obstacles, inflated obstacles, and dilated UAV footprints. This discourages

overlapping paths and reduces the chance of in flight collisions. In Week 11, an additional "reserved corridor" mechanism was added. Each smoothed path is rasterized into grid cells, dilated by a radius that depends on the terrain, and marked as temporarily occupied for other UAVs. This encourages the planner to assign distinct corridors to different UAVs, especially in the urban roads and manmade harbor channels.

Wind handling uses a combination of feedforward and feedback concepts. For each UAV, the controller computes a desired ground velocity based on the path following direction and an emergency braking term. The required air velocity is obtained by subtracting the current wind vector from this desired ground velocity. The airspeed is then clipped based on a maximum factor relative to the nominal cruise speed. If the projected ground progress along the path direction falls below a threshold, an additional component is added along the path to ensure forward motion. Different wind profiles scale the airspeed limit, minimum ground progress fraction, and thrust boost so that the UAV can push harder into stronger winds.

Week 11 Validation Framework and Final Code

The Week 11 script provides a deterministic validation harness that compares baseline and machine learning modes across all terrains and weather conditions. At the top level, the script loops over selected terrains, the four wind profiles (calm, breeze, gusty, storm), and the two modes. For each combination, it constructs or reuses a scenario and calls a simulation function that returns performance metrics. All results are stored in a table and written to a CSV file for post processing.

Scenario generation uses the generate_scenario function. This function builds the selected terrain, rasterizes obstacles, and constructs inflated occupancy masks. It then attempts to place

three UAV start and goal pairs using an adaptive sampling strategy. The placement algorithm enforces a minimum separation between starts and goals, requires a minimum clearance from obstacles, and uses a breadth first search on the inflated free space mask to guarantee connectivity between each start and goal. For the natural terrain, corner boxes and tree placement are adjusted so that no objects land inside the pond, and for the manmade terrain, hard overrides are used to give the blue UAV a long path along the quay. A special override is also used to place an additional obstacle between the yellow UAV's start and goal in the natural terrain to create a more challenging path.

The main simulation logic is implemented in simulate_scenario. This function sets up the state variables for all UAVs, including positions, velocities, planned paths, smoothed paths, and replanning timers. It initializes wind parameters based on the selected profile and allocates storage for trajectories and speeds. The function then runs a time stepping loop until either all UAVs reach their goals or the maximum simulation time of 160 seconds is exceeded.

Inside the loop, the script updates the gust component of the wind and rebuilds both thin and thick occupancy masks. It computes a smoothed distance map and its gradients, which are used for repulsion and clearance calculations. Reserved corridors from existing smoothed paths are reconstructed and combined with static obstacles and other UAVs' footprints to form the planning mask for each vehicle. When a UAV needs to replan, the code calls an A* routine that prefers higher clearance and, when appropriate, penalizes motion directly into the wind.

The controller then computes lookahead waypoints, Frenet based path following directions, repulsion vectors based on distance gradients, and braking behavior using ray casts. A terminal capture rule prevents orbiting near the goal by damping lateral motion when the UAV is close to its target. Wind compensation computes a desired air velocity, caps the airspeed, and

enforces a minimum ground progress along the path direction. Sub step integration and bisection based collision resolution ensure that the thick occupancy grid is respected even when velocities are high in gusty and storm conditions.

At the end of each simulation, the script computes the total time to goal, accumulated path length, minimum clearance across all time steps, mean speed, number of replans, and a Boolean success flag for each run. Separate plotting functions create bar charts of success rate by terrain and mode, box plots of time and path length, and cumulative distribution functions of minimum clearance. The script also records a video of each run showing the terrain, smoothed paths, trajectories, starts, goals, and a wind inset.

Results

Urban Terrain

The urban terrain consists of a grid of rectangular buildings with a cross shaped road network. The start and goal pairs were configured to force long routes along and across these roads, so the UAVs needed to pass through tight gaps between blocks. In calm conditions, both the baseline and machine learning modes completed the runs successfully. The baseline mode reached all goals in about 62.9 seconds, while the machine learning mode achieved a slightly faster time of 61.4 seconds. The paths were smooth, and all UAVs stayed within the planned road corridors.

In breeze conditions, which added a moderate wind along one axis, clear differences appeared. The baseline mode required about 95.2 seconds to complete, showing that the vehicles often slowed down and replanned more frequently when pushed sideways by the wind. The machine learning mode finished in around 81 seconds under the same conditions, suggesting that its path following behavior handled moderate drift more efficiently.

However, in gusty and storm conditions, the wind magnitude and variability were high enough that both modes struggled. In both profiles, all UAVs failed to reach their goals before the 159.8 second time limit. The success rate for urban terrain was therefore fifty percent for both modes because only the calm and breeze runs counted as successful which is shown in Figures 1 and 2. The clearance distributions in Figure 3 for urban terrain were very similar across modes. Both controllers maintained lower quartile clearances near 0.32 meters and upper quartile clearances near 0.42 meters, which indicates that the safety margin logic worked consistently in this map. Shown in Figure 4, the time and path length distributions showed long upper tails caused by repeated replanning and detours in wind.

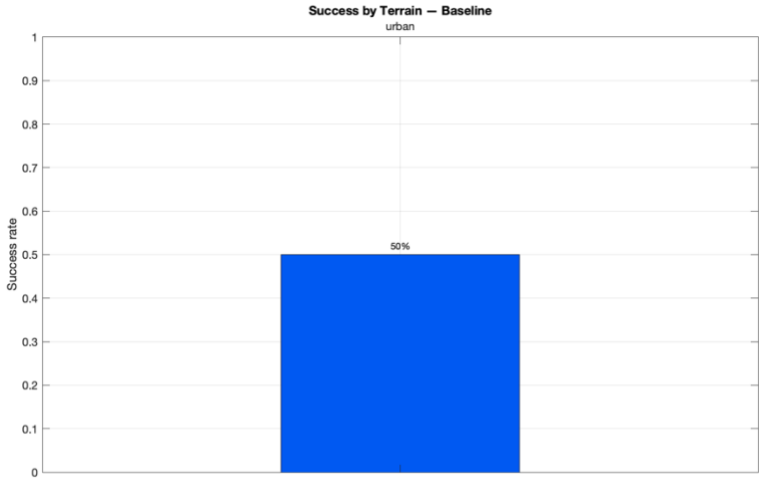

Figure 1: Urban: Success by Terrain – Baseline

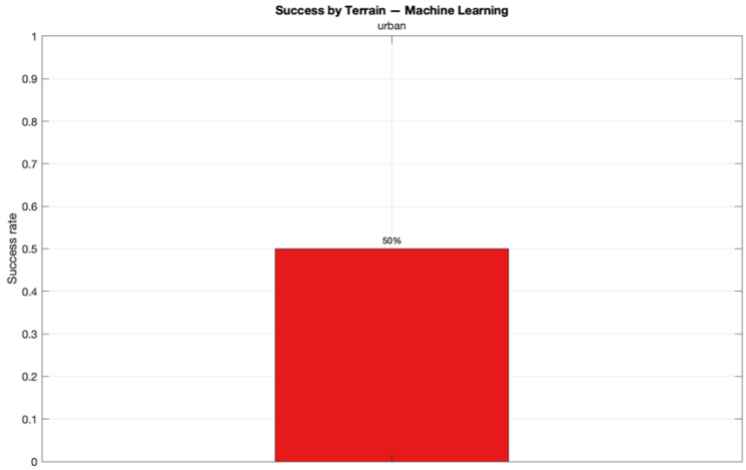

Figure 2: Urban: Success by Terrain – Machine Learning

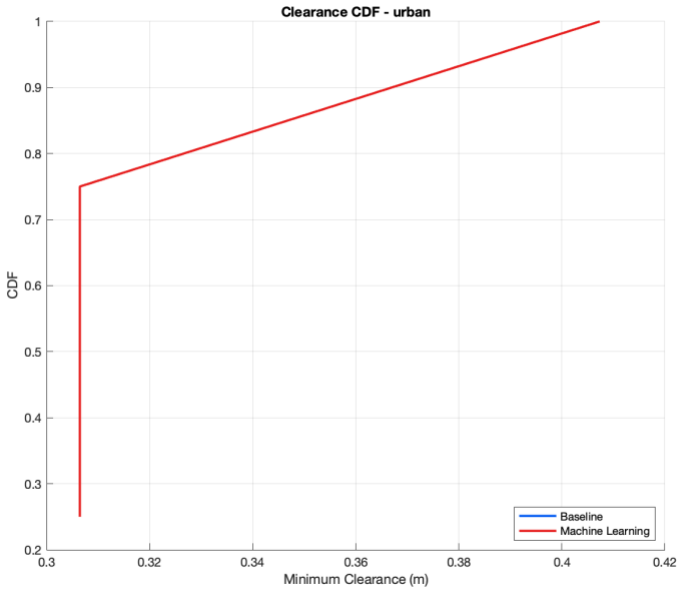

Figure 3: Clearance CDF - Urban

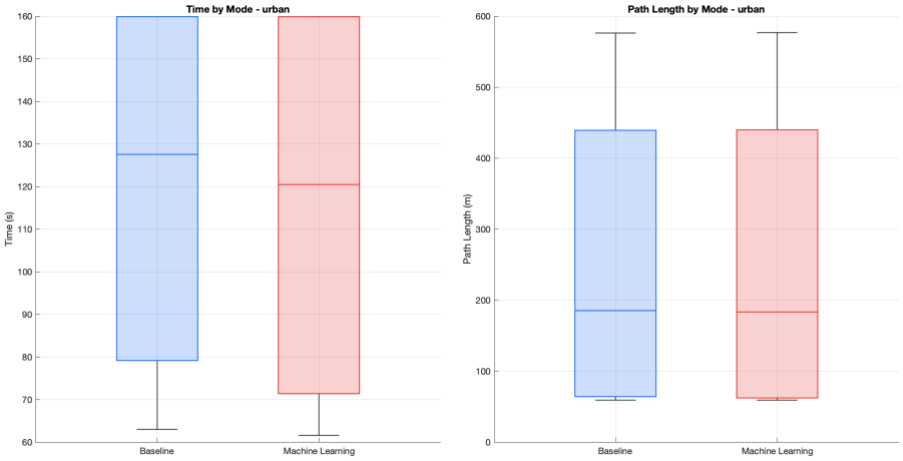

Figure 4: Time and Path by Mode - Urban

Natural Terrain

The natural terrain placed a central pond in the middle of the map with rocks and trees scattered around it. The start and goal pairs were chosen to force UAVs to route around the pond and navigate between obstacles while staying out of the water. Additional logic ensured that

buildings and trees did not overlap the pond region, and the yellow UAV had an extra obstacle inserted between its start and goal to create a more challenging path.

In calm conditions, both modes performed similarly, with times around 53.8 seconds for the baseline and 54.8 seconds for the machine learning mode. Breeze conditions again produced almost identical behavior in the two modes, with times near 71 seconds. In gusty conditions, the baseline completed in about 70.8 seconds and the machine learning mode in about 71.5 seconds, so the difference was small.

Storm conditions were too aggressive for both controllers. All UAVs timed out at 159.8 seconds, and no runs in that profile were counted as successful. Overall, the natural terrain achieved a seventy five percent success rate for both baseline and machine learning modes since three out of four weather profiles had all vehicles reaching their goals which is shown in Figures 5 and 6. The clearance distributions in Figure 7 showed that the baseline started with minimum clearances around 0.375 meters at the lower quartile and around 0.4675 meters at the upper quartile. The machine learning mode maintained slightly higher clearances, starting around 0.4375 meters and reaching approximately 0.4725 meters. Time and path length distributions in Figure 8 were compact in successful runs, with the longest path lengths around 187.5 meters.

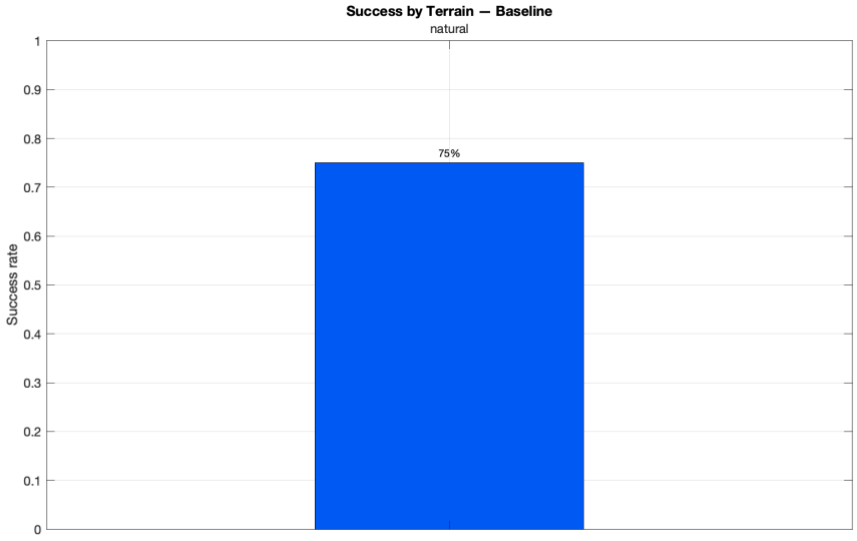

Figure 5: Natural Success by Terrain - Baseline

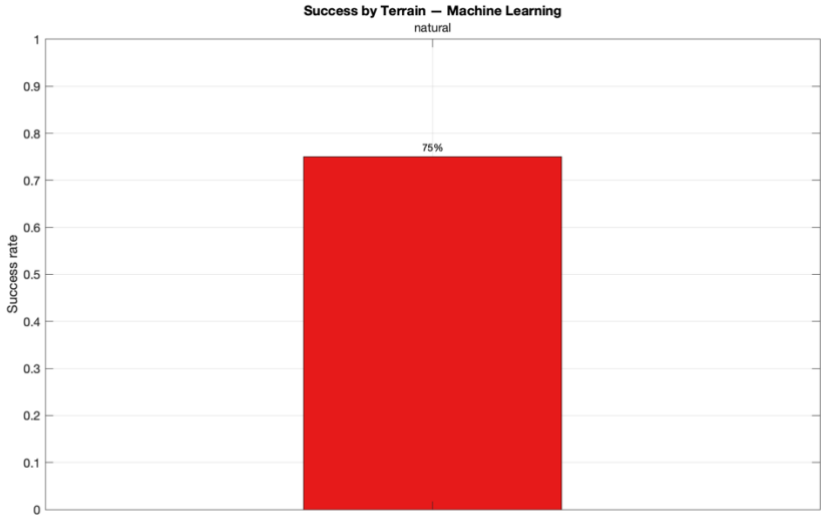

Figure 6: Natural Success by Terrain – Manmade

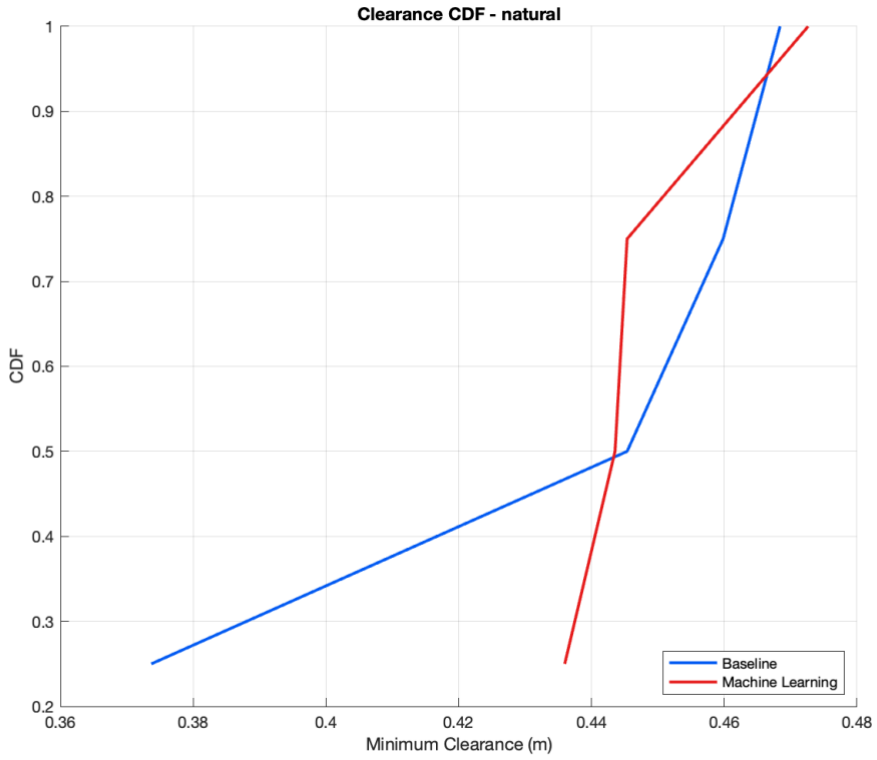

Figure 7: Clearance CDF - Natural

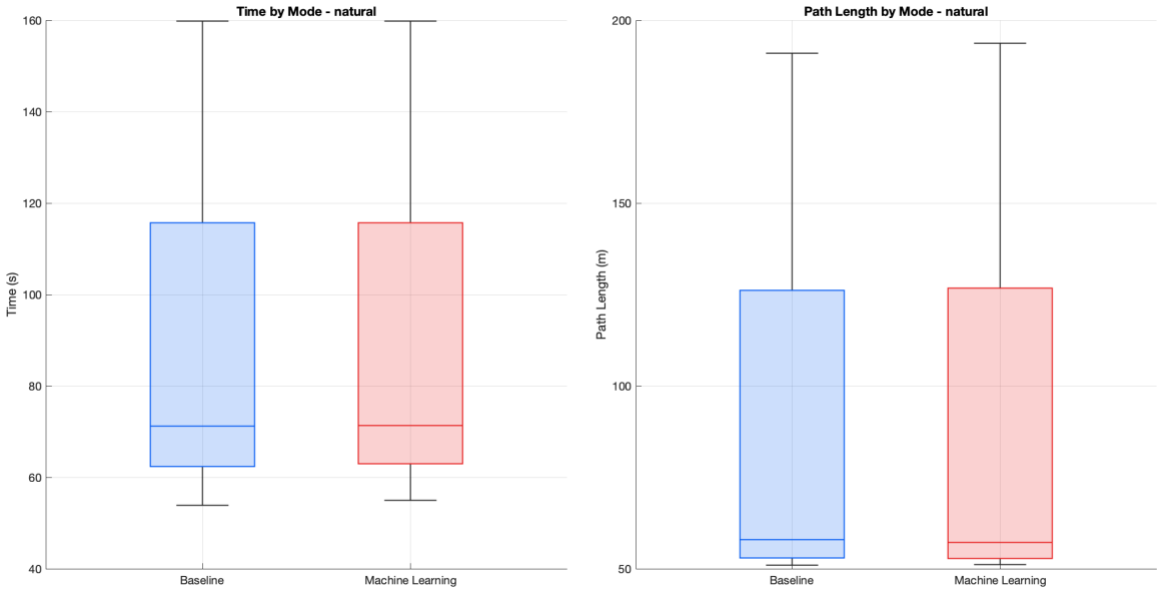

Figure 8 –Time and Path by Mode - Natural

Manmade Terrain

The manmade terrain produced the clearest differences between the two modes. This map includes a long harbor strip, several piers extending into the land, a narrow quay road, and warehouses placed on the opposite side. The geometry creates channels and corners that can be challenging under wind.

In calm conditions, both the baseline and machine learning modes succeeded with times near 65 seconds. Both controllers briefly produced planned paths that cut through buildings before correcting themselves based on the inflated occupancy grid and collision checks, but these errors were resolved quickly and did not lead to failures.

In breeze conditions, the machine learning mode showed a noticeable advantage, finishing in about 54.2 seconds compared to 61.2 seconds for the baseline. The learned controller appeared to track the smoothed paths more decisively along the harbor channels and quay, which reduced detours and the number of replans.

In gusty conditions, the difference reversed. The baseline mode still completed the run successfully in about 56.3 seconds, while the machine learning mode failed because the red UAV did not reach its goal within the time limit. This suggests that the learned controller is more sensitive to crosswind drift and may struggle to recover when pushed off the center of narrow corridors. Storm conditions again caused both controllers to time out with two UAVs failing in each mode.

The final success rate for the manmade terrain was seventy five percent for the baseline mode but only fifty percent for the machine learning mode shown in Figures 9 and 10. Clearance distributions in Figure 11 remained similar, with the baseline starting around 0.25 meters and reaching about 0.5145 meters, while the machine learning mode started near 0.3065 meters and

reached roughly 0.527 meters. In Figure 12, time and path length distributions showed that the machine learning mode had faster completion times in mild wind but higher variance and more failures in strong wind.

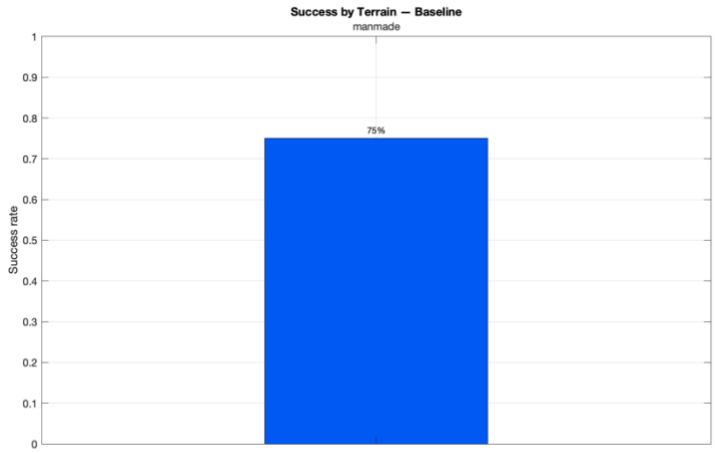

Figure 9: Manmade Success by Terrain - Baseline

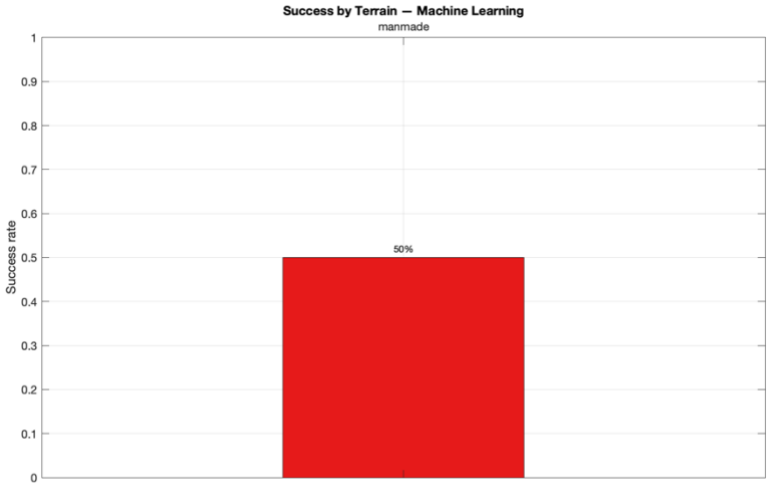

Figure 10: Manmade Success by Terrain – Machine Learning

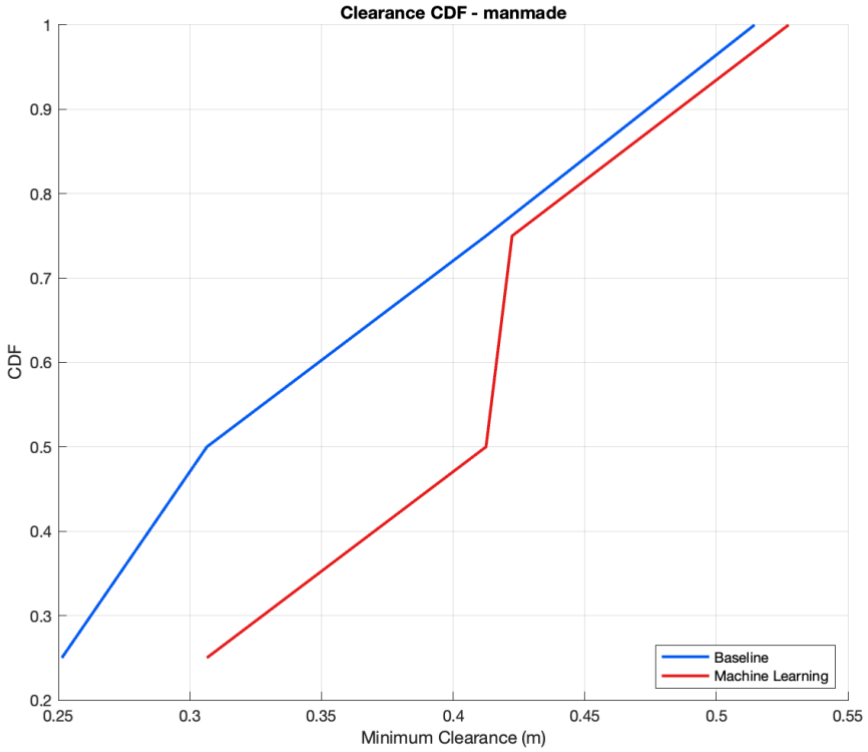

Figure 11: Clearance CDF - Manmade

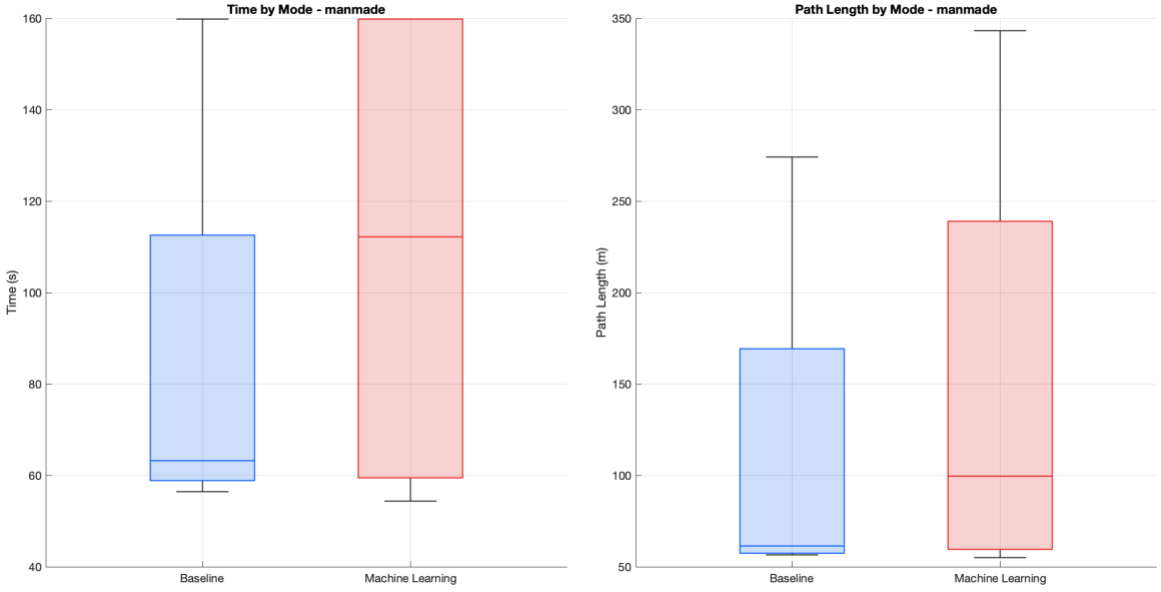

Figure 12: Time and Path by Mode - Manmade

Overall Comparison

Across all terrains, the baseline and machine learning controllers behaved almost identically under calm conditions. Both achieved one hundred percent success for those runs, with small variations in time and path length caused by minor differences in replanning and smoothing. In mild breeze conditions, the machine learning mode often completed faster, especially in urban and manmade terrains, which suggests that the learned policy produced slightly more efficient local motions.

As wind strength increased to gusty and storm levels, the baseline controller maintained higher reliability, particularly in the manmade harbor where the machine learning mode failed in gusty wind. In urban and natural terrains, both modes eventually reached a point where the wind magnitude overwhelmed their ability to make forward progress, and all runs in the strongest wind profile timed out.

Minimum clearance distributions were consistent between modes for all terrains. This indicates that the overall safety behavior is driven more by the planning and inflation logic than by the exact details of the path following controller. Differences between the modes show up more clearly in time to goal and success rate, especially when wind interacts with tight geometry.

Discussion

The validation results highlight the strengths and weaknesses of both controllers. The classical baseline approach, built around A* planning and a hand tuned Frenet follower, provides predictable and stable behavior across a wide range of conditions. It is conservative, especially when wind is strong, but it rarely fails in mild conditions and maintains good safety margins. The supervised machine learning controller is capable of matching the baseline behavior when the conditions remain similar to the dataset used for training. In calm and mild breeze environments, it can even produce slightly faster completion times by making smoother or more decisive local adjustments.

However, the machine learning mode is limited by the diversity of its training data and by the fact that it is a behavior cloning model rather than a reinforcement learning policy trained to optimize long term performance. When the environment or wind conditions deviate from the patterns seen in the dataset, the model can produce actions that do not align well with the assumptions of the planner. This was most visible in the manmade harbor under gusty winds, where the red UAV failed to reach its goal even though the baseline controller succeeded.

Another important observation is that many aspects of overall safety depend on the planning and map inflation stages rather than on the details of the local controller. Both modes use the same static obstacles, thick occupancy dilation, and distance-based repulsion logic. As a result, their minimum clearance statistics are similar across all experiments. The main performance difference lies in how each controller responds to wind induced drift and how efficiently it moves along smoothed paths in the presence of disturbances.

The deterministic validation framework used in Week 11 was valuable for revealing terrain specific behaviors. Because the start and goal pairs, obstacle layouts, and wind profiles

were fixed and repeatable, it was possible to make direct comparisons between modes and isolate the effect of the controller. The collected metrics and generated plots made it easier to see where the learned controller was helpful and where it introduced new failure modes.

Future Work

Several improvements could make the system more robust and demonstrate a clearer advantage for machine learning methods. First, the dataset could be expanded to include more varied environments, wind conditions, and failure cases. Instead of training only on deterministic baseline trajectories, the data could incorporate noisy perturbations and scenarios where the baseline recovers from disturbances. This would help the supervised model learn more about how to respond when it is off the nominal path.

Second, a reinforcement learning approach could be used to train a policy that directly optimizes long term success, time to goal, and clearance rather than simply copying the baseline controller. Techniques such as curriculum learning and domain randomization could gradually expose the policy to harder conditions and reduce overfitting to a single terrain or wind profile. A hybrid training setup could also encourage the learned controller to mimic the baseline in easy conditions while discovering improved strategies in more complex cases.

Third, the simulation could be extended to three dimensional motion and more realistic UAV dynamics. The current model treats the UAVs as point masses moving in a 2D plane with simple velocity constraints. Incorporating altitude, attitude limits, and dynamic constraints would make the problem more realistic and would test whether the planner and controller can scale to more complex state spaces.

Finally, the hybrid controller idea suggested by the Week 11 results could be formalized. In this approach, the system would monitor metrics such as wind magnitude, clearance margins, and path curvature, and switch between a primary machine learning mode and a fallback baseline mode depending on the measured difficulty of the current situation. This would allow the system

to use the learned controller when it is safe and beneficial while retaining the reliability of the classical planner when conditions become extreme.

Conclusion

This project developed and evaluated a complete UAV trajectory optimization framework that combined classical A* planning, Frenet based control, wind modeling, multi UAV coordination, and a supervised machine learning controller. Over twelve weeks, the work progressed from simple baseline simulations to a comprehensive validation across three terrains and four wind profiles. The final Week 11 framework provided a deterministic way to compare a baseline mode with a machine learning enhanced mode using shared scenarios and common metrics.

The results showed that the behavior cloned controller could match and sometimes slightly outperform the baseline in calm and mild wind conditions, especially in the urban and manmade terrains. However, the baseline remained more reliable in strong wind and in tight harbor channels where crosswind drift and narrow corridors increased the difficulty of the problem. Safety margins, as measured by minimum clearance, remained similar for both modes because they shared the same planning and map inflation logic.

Overall, the project demonstrated both the promise and the limitations of plugging a supervised learning controller into a classical planning pipeline. It confirmed that careful dataset engineering and validation are essential for machine learning in safety critical control tasks. It also pointed toward future work on reinforcement learning, hybrid control architectures, and more realistic dynamic models that could further improve robustness and performance.