Flight Paths for Fugitive Methane Detection

6 Degrees of Freedom

May 15th, 2020

Project Manager
Will Larson

Communications Lead
Jake Sansom

Technical Leads
Jennifer Zvonek
Frank Popelar

Table of Contents

Problem Statement

According to the XTO Unconventional Oil and Gas Inventory, equipment leaks comprise the second largest category of pollutants (Tullos et al.). For this reason, the prevention of such leaks is paramount to the continued success of the oil and gas industry.

As a result, many technologies have arisen over the past decade to enhance the detective capabilities of well pad technicians and engineers. Our client, SeekOps, has developed a drone-mounted sensor capable of detecting methane at unprecedented distances. Their sensor, paired with the drone technology they use, allows them to quickly survey a well pad to determine whether a leak is present, quantify the volume of the leak, and locate the emission source (SeekOps).

Currently, a SeekOps pilot flies a drone around a well pad to look for leaks. Using their intuition and experience, the pilot attempts to fly the drone in a path that delivers optimal results to the customer. Naturally, however, no amount of experience can consistently yield a mathematically optimal flight path. This limitation causes SeekOps and its customers to regularly incur inefficiencies in the forms of time, money, and sub-optimal detection results.

Project Objective

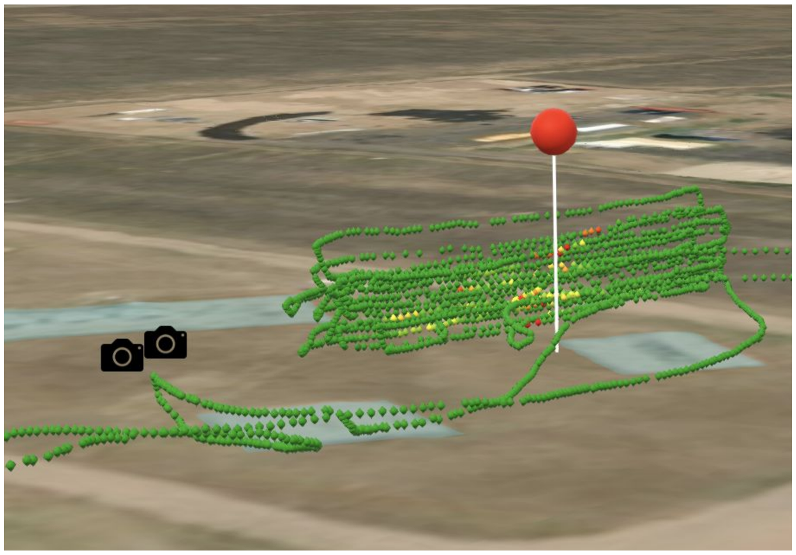
The objective of this project is to provide SeekOps with software built for their methane-detecting drone that automates the drone's flight. Instead of requiring a pilot to estimate the location of possible leaks, the drone will fly directly to the point of the highest leak probability. Additionally, the drone will navigate the well pad in an optimal flight time.

The result of this software not only removes the monotonous task of the pilot, but also maintains the same level of detection accuracy while decreasing flight time. The new software will also decrease human error dramatically, easing the mindsets of SeekOps and their customers.

To further increase the accuracy of the methane detection, the drone may update the methane detection probability in real-time to more precisely locate potential leaks. These updates are not a necessary addition to the software, however, and may be added at a later point.

Introduction and Background

To automate the flight of a methane detection drone, we first wanted to learn more about the current way that SeekOps performs flights to detect methane leaks. During a meeting with SeekOps, they detailed their process in several steps. First, the pilot flies a complete circuit around the well pad to establish "background" levels of methane. After doing that, the pilot flies the drone in the vicinity of emission-prone well pad components. By visually analyzing a live feed of methane concentration measurements downlinked from the drone, the pilot determines whether a leak is present or not. If a leak is present, the pilot flies the drone in a "flux plane" pattern (shown below) to quantify the total flow rate and to pinpoint the likely source of emission. All of these computations are done during post processing.



Given the time constraints of our project, we chose to create an automated version of the current process SeekOps utilizes rather than re-design the entire thing. Our objective, therefore, is twofold: (1) automate what the pilot already does and (2) optimize the process.

To choose waypoints that specify an optimal flight path, we must know quite a bit about how methane propagates from an emission source. Many mathematical and computational models exist that simulate this phenomenon. One of these methane models will be the backbone of our project, so it is paramount that we choose the most appropriate one.

We looked at two primary models. The first was a CFD model created by Dr. Dave Allen. After meeting with him, we gained insight into the great accuracy such a model would afford. The model could, for instance, simulate the downdraft of methane along the wall of a large tank, indicating the best region for measurement. This model, however, presented massive difficulties. Because it was a CFD model, it would have a long runtime. Given that we won't know the wind speed and direction until arrival, such simulations were deemed infeasible.

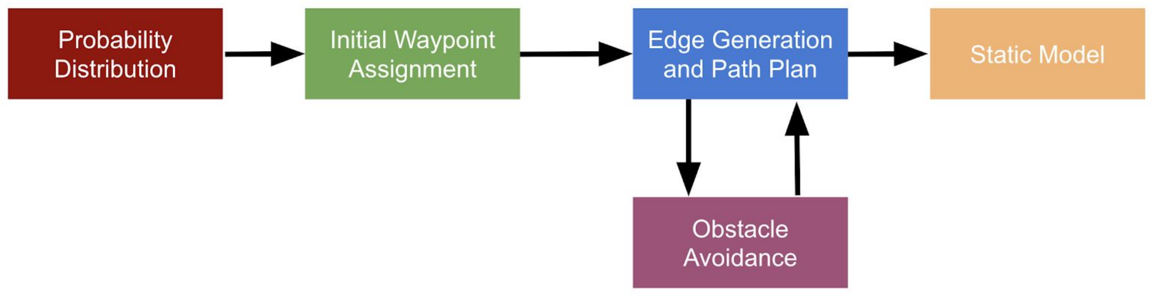
Alternatively, we found a paper titled "Detection of a Simulated Gas Leak in a Wind Tunnel" by Hodgkinson et al. This paper constructs a model of methane propagation as it flows downwind and becomes more diffuse. This model is incredibly useful due to its computational simplicity. It is only parameterized by a handful of variables including flow rate, wind direction, and temperature. These are all data points SeekOps is currently capable of taking in upon arrival at a well pad. By leveraging this model, we will be capable of automatically indicating areas that the drone should fly to, thereby automating the path selection.

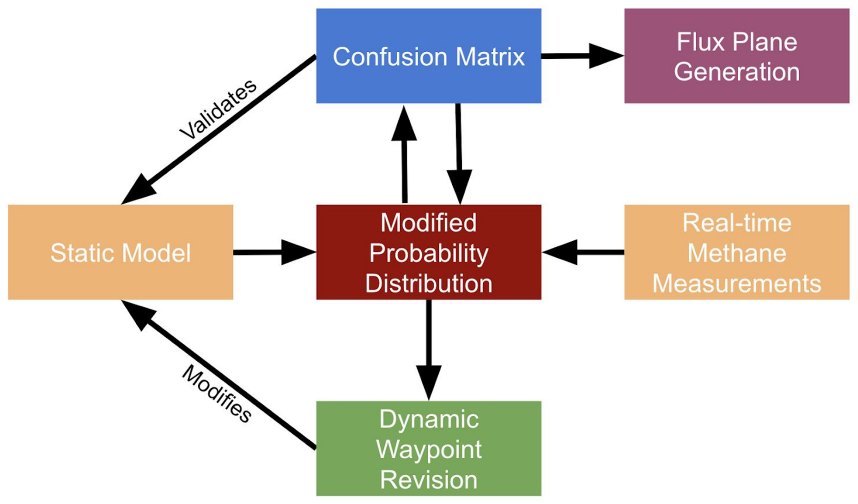
The travelling salesman problem (TSP) asks, "Given a set of random points, what is the shortest path that visits each point and returns to the starting location?" This same problem applies to finding the most optimal path that visits each leak-susceptible piece of equipment on a well-pad. The problem has been studied for the last several decades, resulting in several algorithms for solving the travelling salesman problem under certain circumstances. One particular early solution, Christofides algorithm, proved a simple approach with a near perfect solution. The algorithm featured a minimum spanning tree as the basis of the approach. A minimum spanning tree is similar to the TSP, but instead defines the shortest path from the starting point to each point individually via their neighbors (Christofides). This has served as the basis for many algorithms including those most suited to a drone's navigation around a well pad. Combined with the high-probability points determined using the Hodgkinson et al. model, it should be possible to create an optimal path.

Finally, during the flight it will be necessary to automate the process of determining what constitutes a "significant" methane detection. Currently, a SeekOps pilot does this by visually examining the live feed of data from the drone. The pilot then flies a flux plane if they feel that the methane detection indicates a likely emission. The computations are all done during post processing.

To automate the flux plane trigger, we needed to research probability models. We met with a statistics professor about performing Bayesian analysis in conjunction with the probability model determined by the Hodgkinson et al. paper to quantify our certainty of detection (Mueller). We also looked into Gaussian process models for the comparison of our data with the Hodgkinson et al. model. By implementing the results of our research, we will not only be able to trigger the creation of a flux plane, but we will also be able to quantify the probability of a false negative–the situation wherein we do not detect a leak even though one is present. By quantifying the probability of these values, it will be possible to measure the success of our algorithm.

Feasibility Studies




Confusion Matrix
Flux Plane Generation
Validates
Static Model
Modified Probability Distribution
Real-time Methane Measurements
Modifies
Dynamic Waypoint Revision

One of SeekOps' requests is that we create a probability distribution for where we expect to detect a leak. This distribution will allow us to quantify the uncertainty in our detection of leaks, which then allows us to quantify the probability of a false negative in our results. After the literature review phase, our group identified two potential methods for creating the probability distribution. The first method is to use the statistics given by Allen et al., which tells us how often each component on the well pad leaks, in conjunction with the known geometry of the well pad. From this information, we could create a distribution that gives higher probability for a leak near components, with components that leak more often given a higher probability. The second method is to use the model described by Hodgkinson et al. to create a more detailed probability distribution around each component on the well pad. We chose the second option since it (1) took wind velocity into account to predict where the methane would flow to, (2) allows for more precision than the first method, and (3) allows us to quantify uncertainties. We are aware that the second option is much more computationally intensive. However, since the initial probability distribution may be computed before each flight, we decided that this is not a significant concern.

Two methods were considered for initial waypoint assignment, either basing the waypoint locations on the geometry of the well pad or on a computed probability distribution. The geometry-based method would identify the components most likely to leak and assign waypoints 4-5 meters downwind of the object to maintain the safety buffer. Moving directly downwind of the component would put the drone in the location most likely to detect methane, if we neglect buoyancy effects on the methane particles. The probability distribution-based method would assign waypoints at the local maxima of the given probability distribution. Because we already chose to create a probability distribution, there is no sufficient reason to use the geometry-based method over the probability-based method for assigning the waypoints, and as such the probability-based method was chosen.

After the waypoints are assigned, the connectivity between these waypoints must be defined. If there exists a certain number of obstacles blocking the direct line between two waypoints, it is infeasible to generate a path around these obstacles. If two waypoints are too great a distance apart compared to nearby waypoints, it is computationally expensive to include the path as an option for the pathfinding software.

During the initial path planning, the software can either generate additional waypoints around a main waypoint and include them in the pathfinding software, or the software can simply find a path through the main waypoints and run a short, pre-programmed sweep path near each main waypoint. However, the latter requires the inclusion of real-time obstacle avoidance. The second option also is more closely related to SeekOps' current model. Nevertheless, the reduction in waypoints greatly reduces the runtime for the path-planning algorithm, and obstacle avoidance can easily be done in real-time with simplified assumptions such as assuming circular obstacles.

For the path-planning algorithm, several options were considered. Because the path must be a loop ending at the starting waypoint, the algorithm must solve the Travelling Salesman Problem. Several solutions are available, and four were seriously considered for the drone software. These four are a brute force approach, Christofides algorithm, a modified A* search, and the 2-opt approach (asu.edu; Saythan). After considering runtime, solution optimality, and drone compatibility, the A* approach was chosen. This approach uses a heuristic formula to weigh the option of each path, and always finds the most optimal path. Although the heuristic formula is based on a minimum spanning tree similar to the Christofides algorithm, the algorithm is more easily implemented and suited to the average number of waypoints defined on a well-pad. Splines and polynomial fitting can then be applied to make the path compatible with the drone's limited dynamics (Usenko).

To ensure accurate detection of methane emissions, two additional calculations are necessary during flight. The first calculation determines the statistical significance of a particular group of measurements. If the points test positive for a given region downstream of a well pad component, a “flux plane” is generated and flown. This flux plane is a rectangular grid flown by the drone that allows for the quantification of methane flow rate and emission source. To quantify the statistical significance of a set of measurements, the measurements in question will be compared to a set of “background” measurements collected at the very beginning of the flight. Using a two sample t-test, it is possible to quantify the significance of a particular set of measurements with respect to the background methane levels. The second calculation SeekOps requested presents greater challenges: the probability of a false negative must be quantified, as this would be the worst-case scenario for a customer. To perform, another two sample t-test was considered, but the “background” dataset would be unknown. The measurement data in question needs to be compared against the concentration model delineated by Hodgkinson et al. Effectively, this concentration model is a parameterized prior that can be utilized in a Gaussian process model (Benavoli and Mangili; Rasmussen). By building such a model, it will be possible for the algorithm to safely conclude that no methane emissions under a certain level were missed.

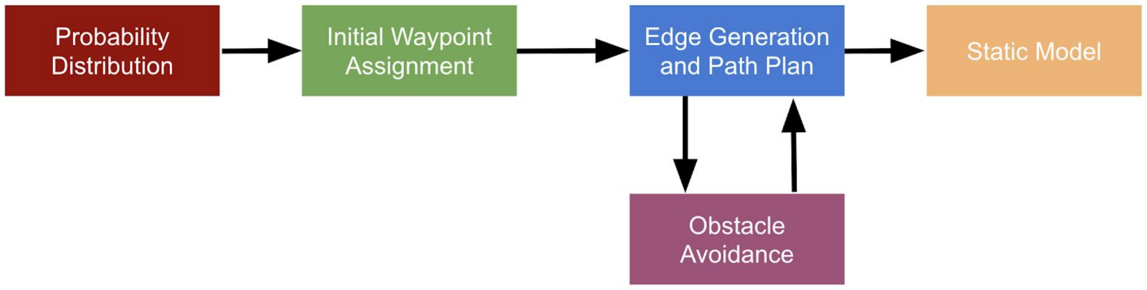
For the definition of flux plane boundaries, four methods were taken into consideration: a geometry based method currently employed at SeekOps; a probability distribution based method; a ‘rolling’ statistical significance method; and the Pasquill-Gifford method outlined by Hodgkinson, et al. SeekOps currently employs a method based on the dimensions of the potential source objects that attempts to greatly overestimate the potential size of the flux plane so as to ensure that it is fully characterized in each flight. The probability distribution method would assign the boundaries of the flux plane as the area around the point of detection where the probability of methane detection was above a certain percent based on a given probability distribution. The ‘rolling’ statistical significance method seeks to use a statistical significance test on groups of samples collected while defining the flux plane to determine the bounds for the flux plane by measuring data until there is no statistically significant detection. The Pasquill-Gifford method is an application of the dispersion coefficients outlined by Pasquill-Gifford and the simplifications by DB Turner to define the area of potential methane concentrations. This method was successfully implemented by Hodgkinson et al., and the assumptions made in the definition of the method are valid for our cases. Because the Pasquill-Gifford method minimizes computational cost of rigorously defining the flux plane boundaries while maintaining valid assumptions, this method was chosen.

Several of the previous design choices require a probability model. We already described how we will compute an initial probability distribution. However, should we have time in the project we will also update this distribution throughout the flight. We will complete this update using Bayesian inference. Because each member of our group has limited background in probability, we consulted a probability professor, Dr. Mueller, at UT to determine this approach. By using Bayesian inference, we will be able to utilize the model from Hodgkinson et al. and real time data to improve upon our initial distribution.

Final Design

Our algorithm design can be split into two main parts: the portion of the code that is run before the drone starts flying, and the portion that is used while the drone is in flight. Thus, we choose to explain the before-flight design and then the during-flight design.

Before Flight



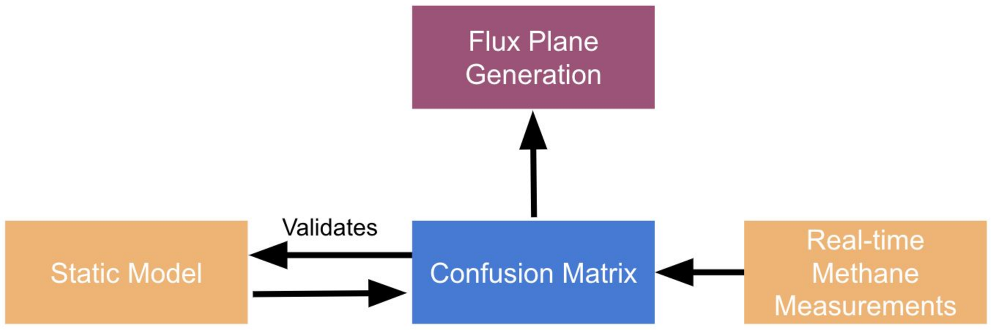
The before-flight algorithm begins by using the Hodgkinson et al. algorithm to compute the probability distribution for the well pad. The program takes well pad component locations, wind speed, and wind direction as inputs. Based on the coordinates of the well pad components, an area including the well pad components and an additional 30m in the x and y directions (to allow for methane flow away from the components) is discretized into a grid of points. The Hodgkinson et al. model is then used to determine the potential concentration of methane at each point due to all components on the well pad. These values are then weighted such that the sum over all points is 1. The resulting array gives us a discrete initial probability distribution for where we expect to possibly find methane on the well pad. Note that this model uses the wind speed and direction to predict where the methane might flow from each well pad component.

Next, a set of initial waypoints is found by identifying the local maxima of the probability distribution. The local maxima are computed using a simple comparison of matrix elements in a given neighborhood. The function looks at the neighbors of each element in the matrix on an adjustable window size, and does not include any mirroring of the matrix along the boundaries, meaning for an NxNxN matrix, the index (-1,0,0) does not map to (N,0,0). Once the indices of the local maxima are found, the function then computes the coordinates of each waypoint given the x, y, and z spacings.

After the initial waypoints are assigned, the shortest path visiting every waypoint is calculated through a Travelling Salesman Problem solver. The particular solver employed uses a heuristic function to account for the unique nature of solving a path around a well pad fitted to the drone’s dynamics. This heuristic function is used to find the shortest path similar to the A* path planning algorithm.

The path generation takes into account the obstacles around the well pad. By simplifying the obstacles as cylindrical structures, the path can easily be manipulated to circle around an obstacle. This circular trajectory must be taken into account when planning the path, and trajectory waypoints delivered to the drone must clearly state the object avoidance. Once a shortest path has been found, a buffer distance is added between waypoints to remain a specified distance away from any objects.

During Flight



The in-flight portion of the algorithm follows the process shown above. As its inputs, it takes in the ordered list of waypoints the drone is going to visit (the static model) along with the real-time methane measurements. At each point of time during the drone’s flight, the algorithm computes the confusion matrix, detailed below. The results of the confusion matrix computation are used to decide whether or not to fly a flux plane pattern. Additionally, the confusion matrix can be used to validate the efficacy of the original static model.

The table below details what a confusion matrix actually is. Each row specifies whether a particular well pad component is actually leaking or not. Each column specifies whether the algorithm believes that the component in question is leaking or not. For a client, the worst possible situation would be a false negative, wherein the algorithm reports the absence of a leak when one is actually occuring. For this reason, the confusion matrix code was designed to minimize this probability.

	Measured a leak	**No leak detected**
Leak present	True Pos.	False Neg.
No leak present	False Pos.	True Neg.

The quantification of each element within the confusion matrix relies on a Bayesian model. Based on the methane measurements taken within a particular window of time, the confusion matrix code compares the probability that such measurements resulted from a leak to the probability that they resulted from normal background noise. Each situation (leak present and normal background noise) can be denoted mathematically as a particular probabilistic model M_i. The leak-present model is based on the model outlined by Hodgkinson et al. and Turner. The normal background noise model is simply a normal distribution around the baseline methane concentration.

Given some dataset D and a particular methane flow model M_i, the probability can be quantified as follows:

$$\mathbb{P}(M_i|D) \propto \mathbb{P}(D|M_i)\mathbb{P}(M_i)$$

In summary, the confusion matrix code compares the probabilities that a particular window of measurements resulted from a leak or resulted from normal background noise. Given a sufficiently high probability that a leak is present, the algorithm triggers a flux plane.

The confusion matrix code also enables the algorithm to quantify its certainty about the absence of a leak. Given a particular window of measurements, if the probability of a leak is sufficiently small, the algorithm can reliably conclude that no leak is present.

If methane is detected at a location on the well pad, the flux plane generation function is called and a vertical curtain pattern is drawn out for the drone to follow. The size of this flux plane is based on the Pasquill-Gifford dispersion coefficients laid out by Hodgkinson et al., and uses the forms given by the assumptions made by DB Turner along with some of our own additional assumptions. The first of these additional assumptions is that the object closest to the drone is the leaking object. This makes intuitive sense, but does not account for the fringe scenario where the drone would detect methane downwind of the actually leaking object, but closer to a separate object. In this case, the 4 meter safety buffer would likely mitigate, if not eliminate, any largescale impact this would have on the computed dispersion coefficients, since the pair of objects would have to be quite close for this to reasonably happen. Additionally, tests are assumed to take place during the day, which eliminates several Pasquill stability categories, and it is assumed that the drone detects methane at distances of less than 100m from their sources, which drastically reduces the complexity of computing the dispersion coefficients.

Once the dispersion coefficients are computed, a vertical curtain pattern is generated with an adjustable spacing in the z direction. However, this component of the flux plane generation is defined in a completely separate function, which allows for easy change of the actual pattern used to define the flux plane at a later time, should SeekOps ever wish to consider a more random definition of the flux plane to more fully eliminate time dependence in the flux plane generation.

Analysis Plan

The analysis plan can be split into two subsections. First, an analysis of each component in the algorithm was necessary to ensure its proper functioning in isolation. Second, an overall analysis was performed by stringing together the entire algorithm.

Probability Distribution Analysis

To verify that our algorithm finds a reasonable probability distribution, we planned to graph the distribution for a given well pad with corresponding wind velocity data. Since we had access to limited data, we also planned to visualize the distribution for simulated leaks (with wind velocity manually chosen) on other known well pad maps.

We planned to test different grid resolutions for the probability distribution. The goal was to achieve the best distribution resolution within a reasonable amount of time. Thus, we wished to balance the trade-off between accuracy and computation time.

Waypoint Assignment Analysis

Since the waypoint assignment is based solely on finding the local maxima of a 3D array, validating the waypoint assignment function is as simple as running test arrays with known local maxima through the function and comparing the output from the code to the known results. In order to fully validate the code, arrays were carefully chosen to: 1) ensure that the function is correctly making comparisons in each of the three coordinate directions, and 2) test that the function is correctly taking into account window size.

Path Generation Analysis

The path generation was run several times to ensure a path near the shortest possible path was always found. Varying parameters including the distance between points and the number of points was used during basic testing. The addition of objects further pushed the program, as larger buffer distances pushed the abilities of the program.

Confusion Matrix Analysis

To validate the proper operation of the confusion matrix computation, it was easiest to analyze its performance at detecting a single leak. To do this, a single leak was simulated and the confusion matrix code was tested at all of the surrounding locations.

Flux Plane Analysis

In order to fully validate the flux plane generation method, the following test cases were considered:

1. Pre-assumed stability class and distance from object
2. Pre-assumed distance from object, wind direction, wind speed, and weather conditions
3. Simplified well-pad with only one object and known wind speed, wind direction, and weather conditions
4. Well-pad with multiple components and known wind speed, wind direction, and weather conditions

The first two test cases check whether the code is correctly computing σ_y and σ_z, the dispersion coefficients given by Hodgkinson et al. and Turner, as well as the Pasquill stability conditions, given by DB Turner, respectively. The third test case is to validate the orientation of the flux plane, which is defined to be perpendicular to the wind direction, and the fourth test case is to make sure that this all still integrates when considering multiple components.

Overall Analysis

In addition to the individual analysis of each component of the algorithm, the algorithm needed to be tested as a whole. Three methods were considered for the overall analysis of the algorithm's capabilities:

1. Field test: Test the algorithm by using it to pilot an actual drone around a well pad. See if it detects methane from known leak locations.
2. Real-Data Simulation: Test the algorithm by “flying” the drone around a simulated version of a real well pad. Interpolate the real methane concentration data collected during an actual flight around that well pad to create a continuous field of methane concentrations. As the drone “flies” around the well pad in simulation, feed it the interpolated methane concentration values. See if it detects methane from the known leak locations.
3. Simulated-Data Simulation: Using a simulated version of a real well pad, choose certain well pad components to serve as leaks. Using the methane flow models outlined by Hodgkinson et al. and Turner, create a noisy field of methane concentration values over the entire well pad. As the drone “flies” around the well pad in simulation, allow it to sample methane concentrations from this simulated methane field. See if it detects methane coming from the wellpad components simulated as leaks.

Of course, a field test would be the most desirable of the three. It would accurately survey all of the difficulties that might arise during the implementation of the algorithm. On the other hand, the algorithm could be in a stage too early for a field test. The introduction of all the variables involved during flight could overcomplicate the project, especially given its single-semester scope. Additionally, COVID-19 certainly did not make in-person testing any easier.

Rather than devote time towards field testing, the team decided to pursue options 2 and 3: the real-data simulation and the simulated-data simulation. Such simulations would provide insight into the performance of the algorithm under more controlled conditions. Hopefully, by tweaking the algorithm during these tests, it could be readied for field testing that SeekOps may eventually want to perform.

Analysis Results and Discussion

Probability Distribution Analysis Results

Below is the probability distribution produced by our algorithm for the well pad geometry and wind data provided to us by SeekOps. Note that this distribution is just a single layer of the full probability distribution, which is in 3 dimensions. The distribution shown here corresponds to a height of 2.7m.

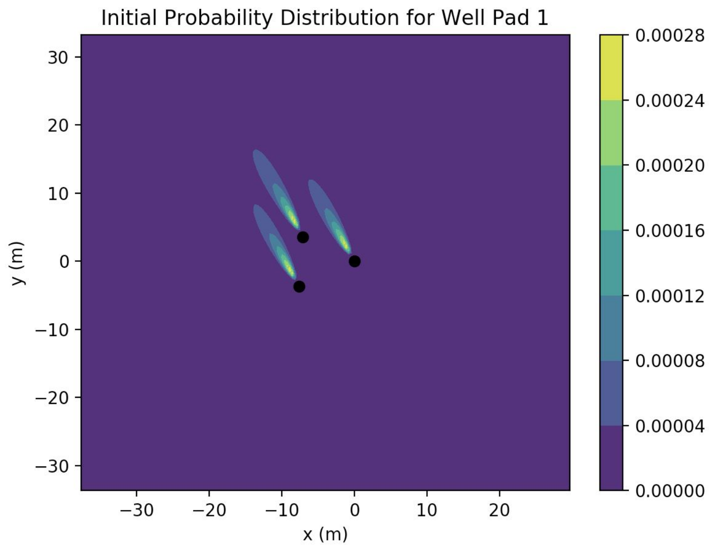


Figure 1: Initial probability distribution for well pad 1

This distribution displays potential methane leaks coming from the well pad components (shown in black) in the same direction as the wind, which was blowing 117^{o} counterclockwise from the x axis. Thus, the probability distribution looks as expected so we believe this part of our program works correctly. For further validation, we simulated a leak with wind speed 4 m/s at an angle of 220° relative to the x-axis on a different well pad. Our code produced the following distribution:

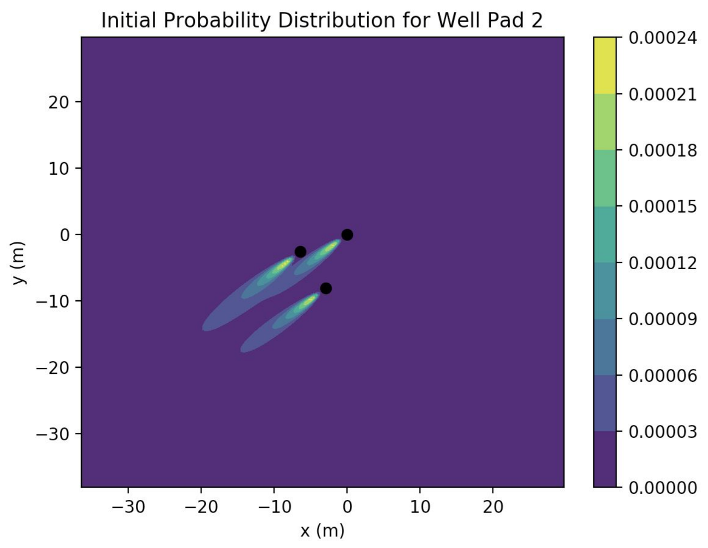
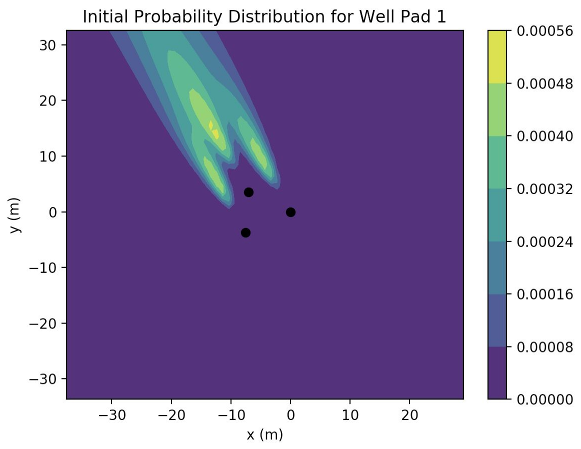
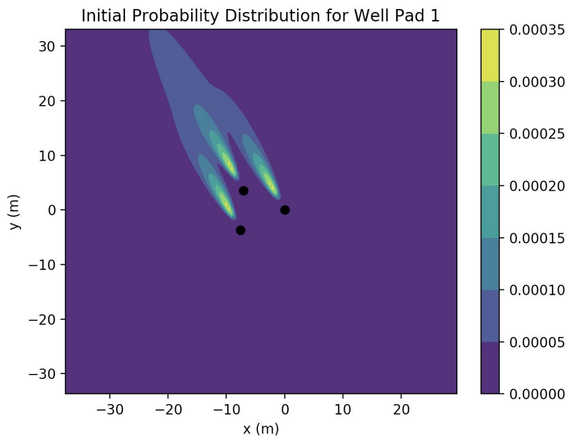
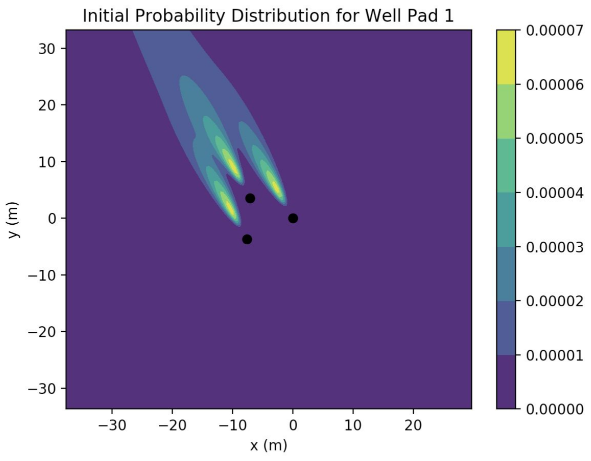


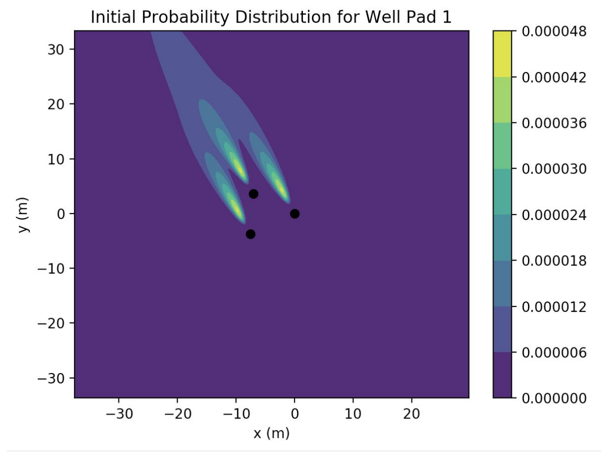
Figure 2: Initial probability distribution for well pad 2

This distribution again shows an increased probability of methane in the direction of the wind, which is expected. The distribution shown is also at a height of 2.7 m.

The following table shows the resulting probability distribution found for well pad 1 when a different number of grid points per meter were used. Note that because the distribution is defined on a discrete set of points, the distributions are not all at the same height. However, this analysis was meant to be qualitative and therefore this difference should not affect our conclusions.

Grid points per meter	Resulting distribution (Well pad 1)

1	 h=2m
2	 h=2.5m
3	 h=2.4m

4	 h=2.5 m

The plot for 1 grid point per meter is much less defined than the other distributions, while the plot for 4 grid points per meter is not significantly different from the less-refined plots. Thus, based on the above grid resolution study, we concluded that 2-3 grid points per meter is sufficient. The other results included in this report use approximately 3 grid points per meter, but we wish to inform SeekOps that 2 grid points per meter should be sufficient if they need to further reduce computation time.

Waypoint Assignment

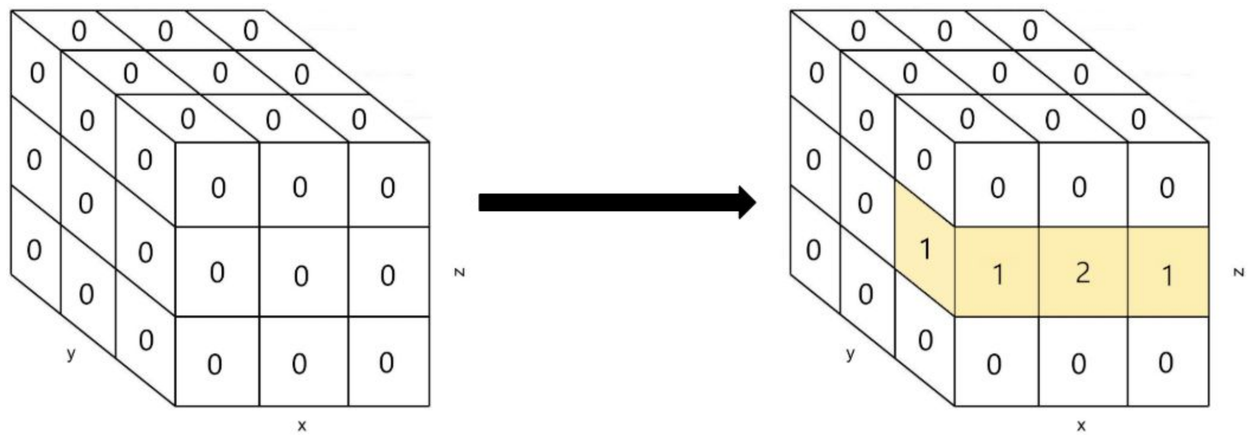


Figure 3: construction of the matrix used to check comparisons in the x direction

The matrices chosen to validate that the function correctly makes comparisons in each coordinate direction were sparse 3x3x3 matrices with the column [1 2 1] inserted in either the x, y, or z coordinate directions, as illustrated in the case of the x direction in Fig. 3 above with the inserted column highlighted in yellow. The results of this testing are shown below in Fig. 4.

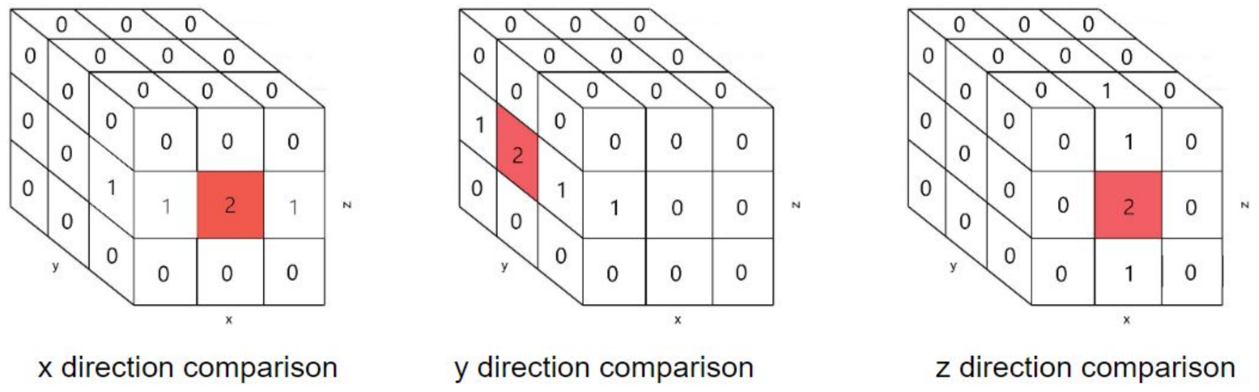


Figure 4: local maxima results from the three matrices used to test comparison directions

The square highlighted red in each of the three matrices in Fig. 4 is the local maxima as identified by the waypoint assignment function. This is, also, clearly the actual local maxima which indicates that the method is making comparisons in each of the coordinate directions accurately.

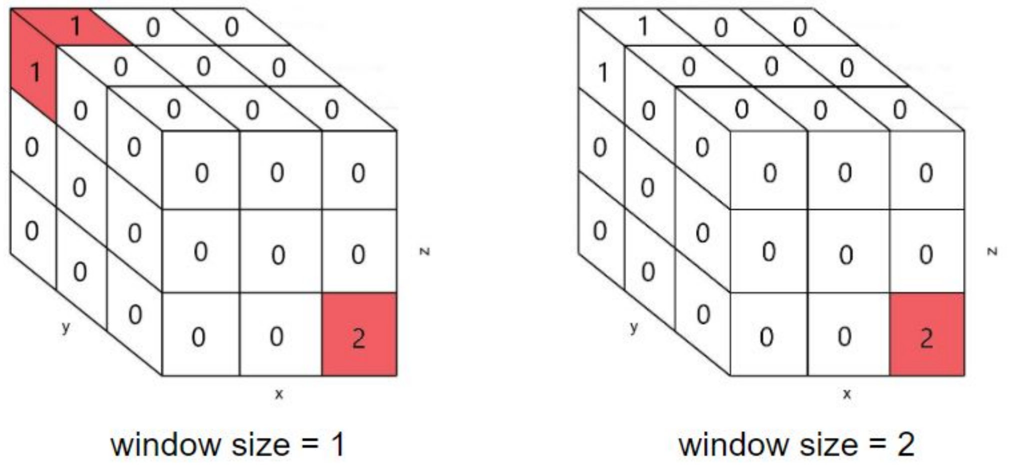


Figure 5: local maxima results with differing window sizes

In order to test that the method was correctly accounting for window size, the matrix shown above in Fig. 5 was chosen. Again, the local maxima identified by the waypoint assignment method, indicated by the square highlighted red, for each of the two window sizes matches with the expected results, which indicates that the function is correctly taking into account window size.

Path Generation

Testing for the path generation began with the Travelling Salesman Problem solver. During development, each run generated ten random points on a 2D plane. The solver then attempted to fit a path and trajectory over the path to study the feasibility of the drone's ability to fly between the waypoints as specified. An example of this initial path and trajectory generation is shown in Fig. 6.

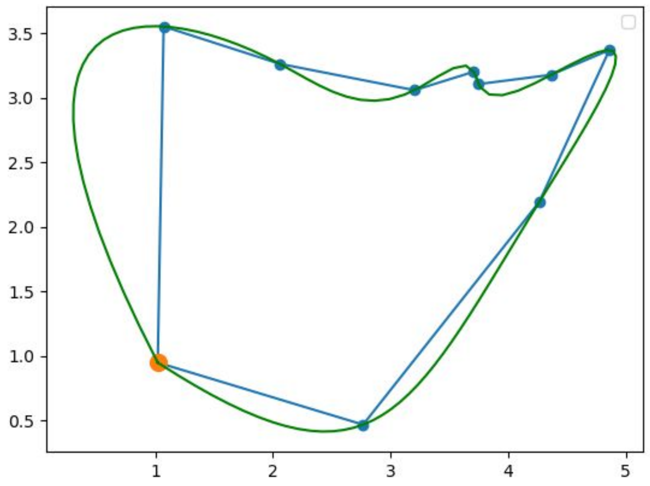


Figure 6: Trajectory Fit Over Travelling Salesman Problem Path

After the path generation successfully computed for several random scatterings of waypoints, testing moved on to object avoidance. The heuristic function involved accounting for objects on the well pad and the path requirements to avoid these objects. After testing with object shapes, buffer distances, and object avoidance procedures, the idea of modeling all objects as cylindrical was deemed most optimal. A circular buffer around objects was then used in addition to the initial path planning to create a path around objects while maintaining a quick trajectory around the well pad. An example of this object avoidance implemented on a real well pad map is shown in Fig. 7.

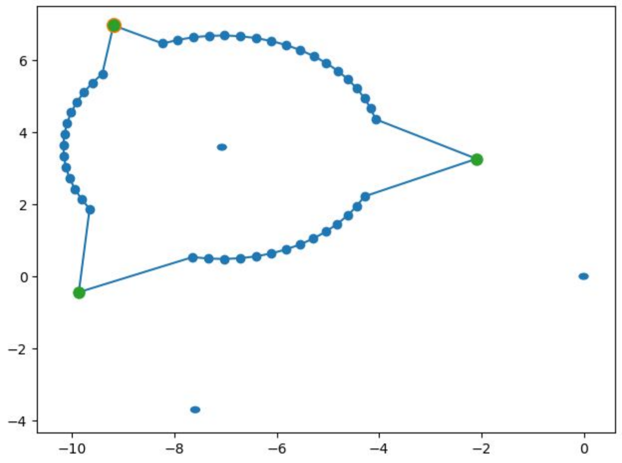
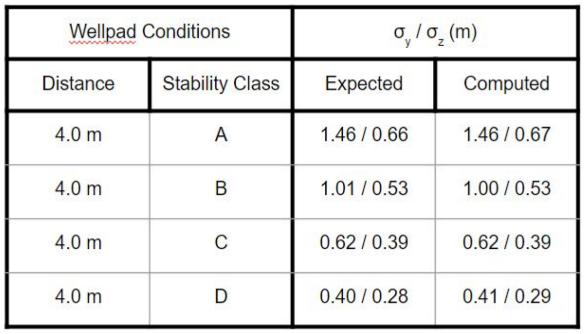


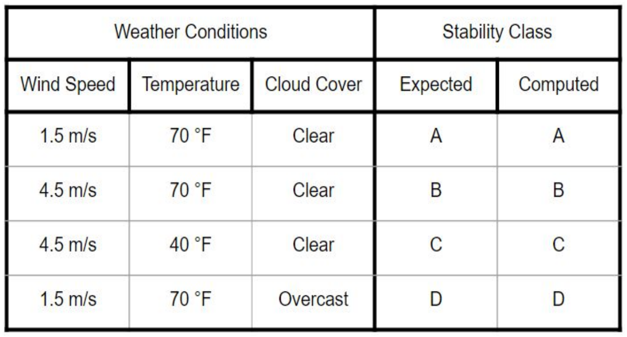
Figure 7: Object Avoidance in Path Generation

The object avoidance is limited by the cylindrical object assumption. However, for only an initial path assignment, this proves to not be an issue. The drone implements its own, in-flight object detection, and can account for any unexpected path changes such as wind disturbance and differing object shapes.

Flux Plane Analysis Results



Wellpad Conditions		σ_y / σ_z (m)	
Distance	Stability Class	Expected	Computed
4.0 m	A	1.46 / 0.66	1.46 / 0.67
4.0 m	B	1.01 / 0.53	1.00 / 0.53
4.0 m	C	0.62 / 0.39	0.62 / 0.39
4.0 m	D	0.40 / 0.28	0.41 / 0.29



Weather Conditions			Stability Class	
Wind Speed	Temperature	Cloud Cover	Expected	Computed
1.5 m/s	70 °F	Clear	A	A
4.5 m/s	70 °F	Clear	B	B
4.5 m/s	40 °F	Clear	C	C
1.5 m/s	70 °F	Overcast	D	D

Figures 8 (left) and 9 (right): conditions for test cases 1 (left) and 2 (right) for the flux plane analysis

For test case 1 in flux plane analysis, with known stability class and distance from object, the computed results match with the expected results, which were computed by hand, almost exactly. The small error in these numbers is very small, and is assumed to be due to rounding. For test case 2, with known distance from object, wind direction, wind speed, and weather conditions, the code was able to correctly identify the Pasquill stability class in all of the tested combinations.

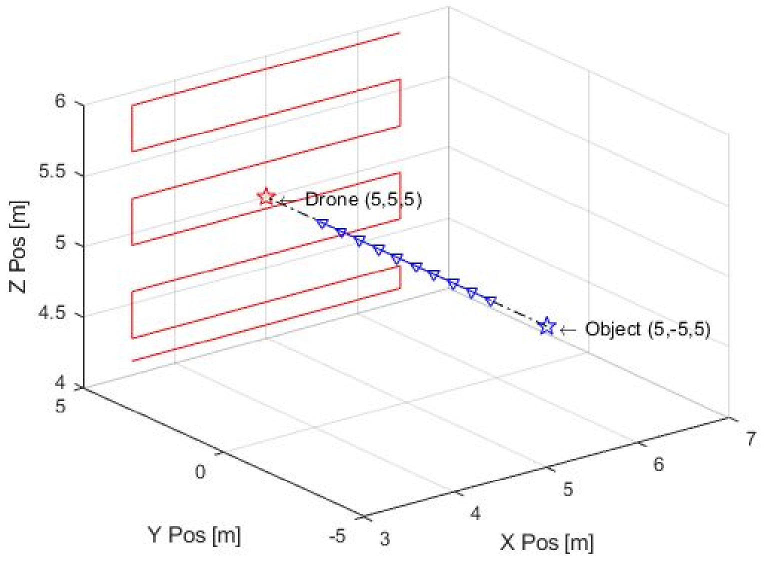
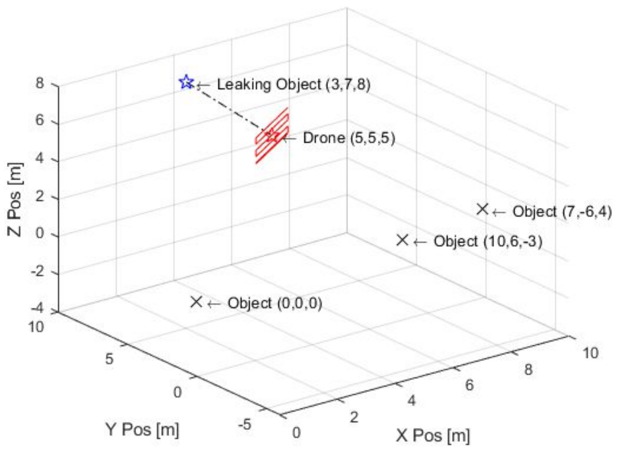
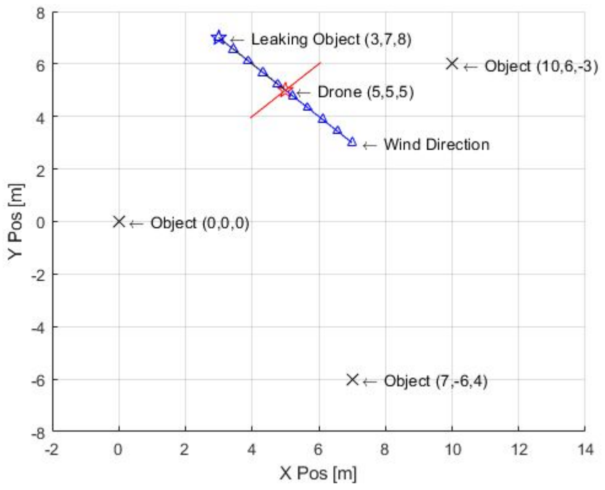


Figure 10: vertical curtain pattern for the third test case in flux plane generation

For the third test case, which was a simplified well pad with only one component, the vertical curtain pattern is shown above in Fig. 10 in the solid red line. Comparing this to the wind direction, which is shown as the blue line with arrowheads pointing in the wind direction, shows that the vertical curtain is formed perpendicular to the wind direction, which is exactly what is expected. Additionally, the path traced by the red line does follow a recognizable curtain pattern, meaning the waypoints are being generated in the correct order.



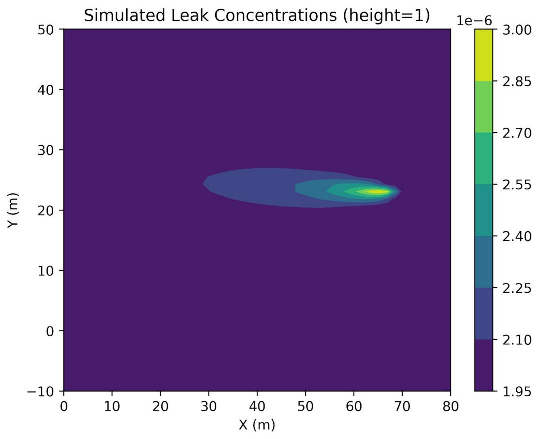


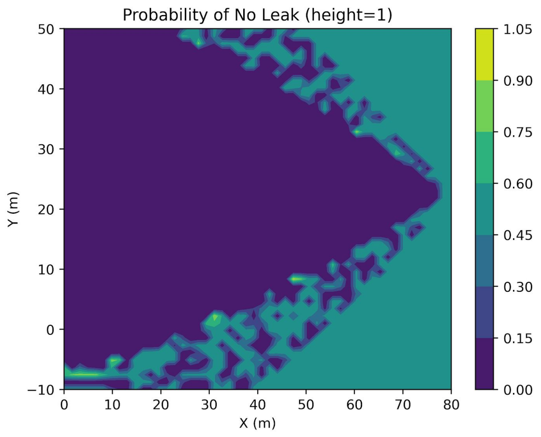
Figures 11 (left) and 12 (right): 3D (left) and X-Y projection (right) of the results of the fourth test case for flux plane generation

This leaves the fourth and final test case for flux plane generation, which is the well pad with multiple components. Again, as illustrated most clearly in Fig. 12, the flux plane is generated perpendicular to the wind direction, again shown in red and blue respectively.

Confusion Matrix Analysis Results

Recall that, to test the confusion matrix code in isolation, a single leak was simulated. Graphed below, on the left, are the concentration values simulated over the mock well pad. The graph on the right displays the results of the confusion matrix code over the entire mock well pad. At each (x,y) value, the methane concentration predicted by the model on the left was fed into the confusion matrix code. This value was used to compute the probability that a leak was present. Displayed on the right is the inverse of this value: the probability that a leak is not present. The scattered regions on the border between upstream and the downstream cone display where the methane concentration becomes statistically significant. Downstream of the leak, the confusion matrix code computes a probability of 0. In other words, the model is statistically certain that a leak is present. Upstream of the leak, the model cannot determine whether or not a leak is present, yielding a probability of 0.5.





Figures 13 (left) and 14 (right): Test results for the confusion matrix code

Overall Analysis

Following our initial analysis plan, we decided to try a real-data simulation test. To do this, it was necessary to interpolate the real methane concentration values across the well pad. This process created a scalar field predicting the methane concentration at any point on the well pad, even if it wasn’t along the real path that the drone took. By doing this, the flight path generated by our algorithm (which would inevitably be different from that originally flown by SeekOps), could sample methane concentrations as it flew around.

Implementation of the real-data simulation test proved difficult, if not impossible. The data did not behave nicely, and it certainly did not behave in the manner predicted by the Hodgkinson et al. and Turner flow models. To understand why this was the case, the drone’s real flight path was plotted in three dimensions.

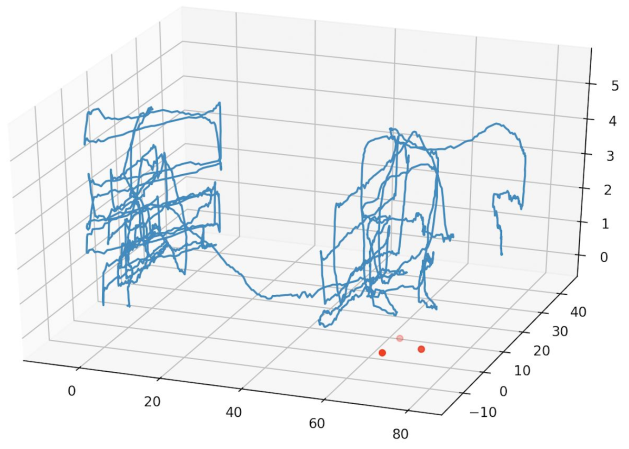


Figure 15: The real flight path (blue) of a drone around a real well pad. Three of the components on the well pad are shown in red.

By visually analyzing the plot above, the error source became almost immediately clear: the data was too sparse. Because the drone covers such a small fraction of the total well pad, the interpolation method is questionable at best. To further corroborate this claim, a Gaussian process regression model was used to analyze the effect of the sparsity. Gaussian process models interpolate n-dimensional fields of data using a statistical model. This model allowed for the visualization not only of the interpolated methane concentrations, but also of the variance of these concentrations. Below is a graph displaying the standard deviation of the interpolated methane concentration values. Relative to the background methane concentration value (on the order of 10^-6 ppm), these standard deviations were immense.

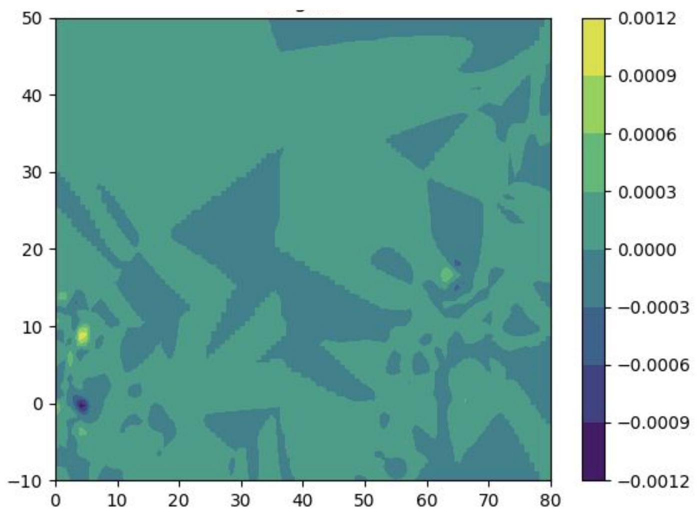


Figure 16: The standard deviation of the predicted methane concentrations across the entire well pad. Graph shown at a constant height of 4m.

Due to the sparsity of the real data, the simulated-data simulation seemed to be the best way to test the algorithm’s performance. Plotted below are the results of a simulated flight using real well pad components and simulated leaks. The background displays the methane concentrations over the entire well pad (a wind direction was chosen randomly). The blue, orange, and green dots show the positions of the well pad components. Finally, the blue line displays the path of the drone as it flies through the waypoints.

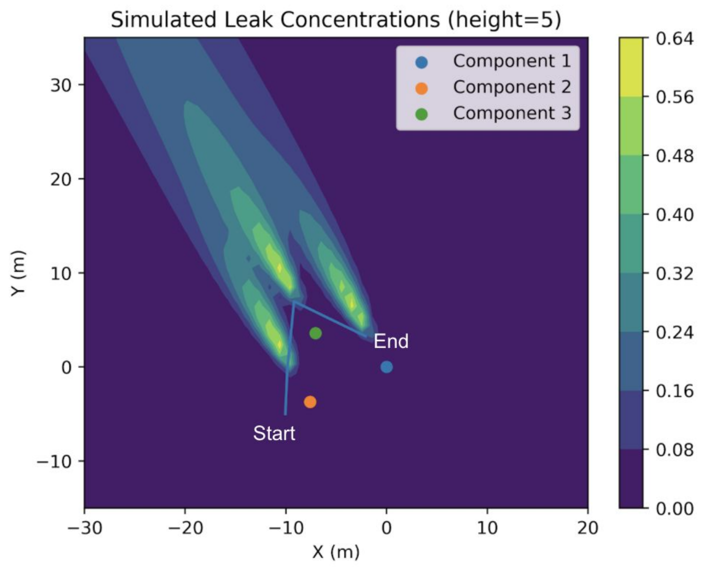


Figure 17: The path of the drone around three real well pad components plotted over the simulated concentration distribution.

Although this well pad seems relatively small, it provided us with an excellent opportunity to test the code's performance once it was all integrated together. Shown below are the confusion matrix results plotted as the drone flies through its path. As it passes through each waypoint (displayed as stars), it is able to detect the leak coming from the nearby well pad component. As displayed clearly by this diagram, there is a flaw with the confusion matrix algorithm. Namely, it cannot differentiate between leaks coming from well pad components close to one another. Still, however, a quantitative measure of statistical significance is an improvement over visual analysis (currently used by SeekOps pilots), and the model could be modified and improved in the future.

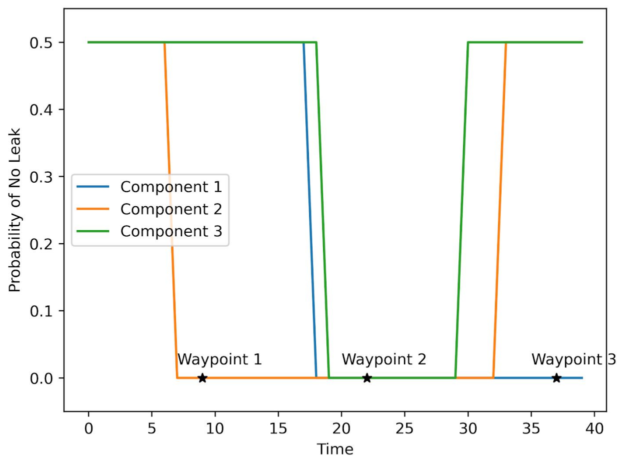


Figure 18: The predictions of the confusion matrix code over the course of the simulated flight.

Requirements Analysis

Displayed below are the quantitative project requirements this team first intended to achieve. Marked in green are the requirements tested for and met. Marked in yellow are the requirements most likely met, but not tested for. Finally, marked in red are the requirements this team was not able to meet. These requirement evaluations are based on the analysis results detailed above, along with the natural design of the algorithm.

By design, the algorithm assigns waypoints so as to maintain a 4m buffer between well pad objects and the drone. The simulated flight around the well pad was run at 2.5m/s, and it took less than 5 minutes. The total flight time is impossible to accurately compute, however, because this flight did not include the additional time required to fly flux plane patterns, and it surveyed a relatively small well pad. Despite this limitation, the algorithm seems to produce timely flight paths. It flies in a manner similar to current SeekOps pilots, but its path is optimized. The team has no reason to believe the algorithm will fall short of time requirements. Finally, by design the algorithm triggers a flux plane upon a high enough false negative probability.

	Requirement	**Value**	**Achieved?**
Needs	Distance maintained between drone and well pad components	4m	✓
	Average flight time	<10m	✓
	Flight speed	<5 m/s	✓
	Probability of false negative detection	<10%	✓
Wants	Average flight time	<5m	✓
	Account for periodic emission cycles		X
	Web-based GUI to control the program and view data		X

The qualitative requirements for this project were

1. ✓ Drone should fly around well pad to detect methane emissions
2. ✓ Algorithm should define waypoints on the well pad
3. ✓ Minimize probability of a false negative
4. ✓ As fast or faster than current pilots
5. ✓ Avoid obstacles on well pad

6. ✓ Determine background methane levels
7. ✓ Fly in a vertical-curtain pattern near possible detection
8. ✓ Determine if methane detection is statistically significant
9. ✓ Determine boundaries of flux plane to capture probability distribution
10. X Update waypoints real time
11. ✓ Code written in Python

Requirements 1, 2, and 11 were the goal of the entire project and thus were satisfied. Requirements 3, 6, and 8 are satisfied by the use of the confusion matrix. Requirement 4, as explained above, is expected to be satisfied but since we were unable to test on a real drone we cannot know for sure if it is met. Requirement 5 is satisfied by the obstacle avoidance part of our algorithm. Requirements 7 and 9 are satisfied by the flux plane generation. Requirement 10 was not satisfied due to lack of time and difficulties due to online courses.

Conclusions and Future Work

We created a program in Python that outputs a set of waypoints to be traversed by a drone to detect methane leaks on a well pad. The path created by these waypoints avoids obstacles, passes through areas with high probability of methane being present, and takes the drone efficiently through areas of interest on the well pad. Furthermore, our program determines the statistical significance of detected methane levels and uses this information to determine where along the path to trigger a flux plane. This flux plane is then automatically generated to capture the full methane concentration distribution from the corresponding well pad component that may be leaking.

Our program finds a probability distribution for where we expect potential leaks to be. This distribution is found using the model described by Hodgkinson et al. and Turner. The original intention was to create this initial probability distribution and then update it every few time steps using new methane detection data. Unfortunately, due to limited time and the shift to online classes, we were not able to implement the probability distribution update. However, we did explore the theory behind this part of the project and intend to share this information with SeekOps. We are hopeful that SeekOps, should they choose to, can add this part of the code without too much difficulty.

Due to the circumstances with COVID-19, we were also unable to test our algorithm with a physical drone. We tried our best to test our algorithm using simulations and test data given to us by SeekOps. However, we can’t know for sure how accurate and useful our program is until it is tested on a real drone. However, based on our simulations we are hopeful that the path produced by our algorithm will be efficient and allow SeekOps to transition to autonomous drone flight on well pads.

Future work for this project includes testing our current algorithm on a physical drone. Once this testing is done and if the flight path is successful, we suggest that the algorithm be modified to include probability distribution updates using Bayesian inference. The accuracy of the program could also be improved if more detailed well pad maps are used. For instance, the height of well pad components would be useful in determining the height of potential leaks, and the location of pipes and other, smaller equipment would be useful in locating additional potential leak sites.

References

Allen, D. T. (2020, February 21). Meeting with Dr. Dave Allen. Austin, TX.

Allen, D. T., et al. “Measurements of Methane Emissions at Natural Gas Production Sites in the United States.” Proceedings of the National Academy of Sciences, vol. 110, no. 44, 2013, pp. 17768–17773., doi:10.1073/pnas.1304880110.

Benavoli, A. and Mangili, F. “Gaussian process for Bayesian hypothesis tests on regression functions.” Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, PMLR 38:74-82, 2015.

Christofides, Nicos, Worst-case analysis of a new heuristic for the travelling salesman problem, Report 388, Graduate School of Industrial Administration, CMU, 1976.

Hodgkinson, J, et al. “Detection of a Simulated Gas Leak in a Wind Tunnel.” Measurement Science and Technology, vol. 17, no. 6, 2006, pp. 1586–1593., doi:10.1088/0957-0233/17/6/041.

Jung, D., & Tsiotras, P. (2013). On-line path generation for unmanned aerial vehicles using B-spline path templates. Journal of Guidance, Control, and Dynamics, 36(6), 1642-1653.

Mueller, Peter. (2020, March 31). Meeting with Dr. Mueller. Austin, TX.

Rasmussen, C. E. (2006). Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning. (Doctoral dissertation, Technische Universität Darmstadt, Darmstadt, Germany).

Sathyan, A., Boone, N., & Cohen, K. (2015). Comparison of approximate approaches to solving the travelling salesman problem and its application to UAV swarming. International Journal of Unmanned Systems Engineering., 3(1), 1.

SeekOps. (2020, February 19). Meeting with SeekOps. Austin, TX.

SeekOps.com

Tullos, E. et al. (2019, December 13). Insights from a field trial of methane detection technologies. *American Geophysical Union Fall Meeting*. San Francisco, CA.

Turner, D B. Workbook of Atmospheric Dispersion Estimates: An Introduction to Dispersion Modeling. Boca Raton: Lewis Publishers, 1994. Print.

Usenko, V., von Stumberg, L., Pangercic, A., & Cremers, D. (2017, September). Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In 2017

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 215-222). IEEE.

http://www.public.asu.edu/~huanliu/AI04S/project1.html