Waypoint Generation based upon Sensor Aimpoint

Shannon M. Farrell, Capt, USAF*
Air Force Research Laboratory Munitions Directorate, Eglin Air Force Base, FL 32542
and David R. Jacques †
Air Force Institute of Technology, Wright-Patterson Air Force Base, OH 45433

ABSTRACT

This research developed open-loop waypoint generation algorithms required to keep a point of interest (POI), or target, in the field of view (FOV) of a fixed sensor on a micro air vehicle (MAV) in the presence of a constant wind. Two scenarios were explored: one where the MAV orbits the POI, and an overflight scenario in which the MAV flies to align the sensor at the POI along a preferred look angle. A commercial off-the-shelf (COTS) autopilot and COTS airframe were used for both hardware-in-the-loop and flight testing. The algorithms consistently generated flight plans of waypoints to align the sensor aimpoint on the POI. Limitations in the native autopilot control loops greatly affected the results for keeping the POI in the FOV. Future research will investigate solutions to help the MAV reach the commanded waypoints as well as solutions to help stabilize the sensor aimpoint.

1 INTRODUCTION

Fixed sensors tend to be cheaper, lighter, and more robust than gimbaled sensors. This makes them ideal for employment on MAVs, especially as MAVs continue to shrink in size and weight. A fixed sensor, however, is coupled to the flight dynamics of the micro aerial vehicle (MAV). An operator of a tactical intelligence, surveillance, and reconnaissance (ISR) MAV likely cares less about the physical location of the MAV and more about the information gathered by the sensor. This leads to the need to generate flight plans for MAVs such that the sensor remains aimed at the point of interest (POI). The operator may want to view the POI from multiple angles as in the case of an orbit or view the POI along a preferred sensor orientation angle, in essence having the sensor overfly the POI. This research developed waypoint generation algorithms to keep a fixed, vehicle-mounted sensor aimed at a POI for both scenarios in the presence of constant wind. The research further attempted to do this in a platform independent manner such that the algorithms could be implemented on almost any MAV without respect to the hardware used by the MAV. The POI was assumed fixed for the purposes of this research. The POI was defined as a set of GPS coordinates that were either known or derived from sensor imagery. The first scenario relies on a sensor aimed out the side of the MAV, while the second considers both a side-viewing sensor and a forward-viewing sensor.

*MAV Systems Engineer, email address: shannon.farrell@us.af.mil
†Assistant Professor, Department of Systems and Engineering Management, email address: david.jacques@afit.edu

2 BACKGROUND

2.1 Waypoint Guidance

To account for wind affects on MAV path planning, [1] used a lookup table in the MAV guidance system to determine the appropriate time to transition from one flight path segment to the next. This allowed the MAV to avoid reaching the waypoint before transitioning to the next flight segment, causing the MAV to overshoot the waypoint. A MAV with a sensor was not considered, so the sensor footprint was not part of their solution.

In addition to simulating the MAV flying a course to demonstrate the lookup table's use, [1] demonstrated the lookup table worked for the MAV to circle a target. The circle flown was a many-sided polygon made up of many individual waypoints. In the presence of wind, the ground speed was not held constant, but by anticipating a turn before each segment, the aircraft circled the target.

2.2 "Good Helmsman" Sensor Aimpoint

[2] explored flight path planning for MAVs that kept the sensor aimed at a point of interest while the aircraft maneuvered. [2] sought a trajectory which would maintain constant geometry between MAV and target. The algorithms developed allowed for wind corrections if the wind data was available. Since the sensor had a field of view larger than the aimpoint, it did not have to be directly pointed at the target to have the target visible in the sensor display. As [2] pointed out, this could lead to the target wandering in the sensor display. Implementing wind corrections could keep the target much closer to center in the display. Iterating the algorithms allowed the aircraft to adjust to slowly varying winds. While the solution developed by [2] was robust, it relied on the ability to change the camera angle to keep the target in the FOV as well as a MAV with the ability to coordinate turns.

2.3 Robust Wind Correction Algorithms

[3] considered the effects of wind on sensor aimpoint in a limited fashion. The sensor was assumed to point out in front of the MAV with some depression. The algorithms generated in [3] accounted for the MAV yawing into the wind, but did

not account for the aircraft banking into a turn in an orbit, nor did it generate usable flight plans to pass back to the MAV.

2.4 Constant Wind with a Bounded Turning Rate

Having found their test MAV, a SIG Rascal 110, had flight speeds of 20 m/s and could encounter winds of 5 m/s, [4] researched how to apply constant winds and a constrained turning rate of an aircraft to the Dubins path [5] solution. They noted that in the presence of wind, an aircraft's ground path was not a straight line. They had to iteratively solve the Dubins path problem with a virtually moving target. The work accomplished by [4] did not attempt to implement a solution on a MAV nor did it consider a sensor as part of the solution.

2.5 Optimal Wind Corrected Flight Path Planning

[6] evaluated dynamic flight path optimization for a MAV in a constant wind environment. The research sought the optimal minimum-time flight path for a MAV with fixed sensors mounted in both a forward-looking and side-view configurations. [6] differed from previous research because most previous research allowed for gimbaled sensors or focused on aircraft large enough to not be significantly affected by winds. The previous research placed the aircraft above the target instead of placing the target within the field of view (FOV) of the sensor.

One assumption in [6] was that the aircraft was a point mass. This made the analysis easier, but did not account for the aircraft banking and what that did to the sensor footprint during a turn. By not accounting for bank angle in an orbit, the sensor aimed at the incorrect location instead of the target.

Using MATLAB to conduct the discrete dynamic optimization, [6] was able to identify which of six potential Dubins paths [5] was the optimal solution. This solution was not perfect, however, since one of the assumptions was constant wind speed and direction for any given MAV flight. To account for the constant wind, [6] created a virtual target which moved at a speed equal to the wind speed but in an opposite direction. This allowed the algorithm to create a path that put the sensor footprint over the actual target. The algorithm used the Dubins path model as the initial estimate for the final flight path, then ran the various paths through the MATLAB optimization function *fmincon* to produce the optimal solution. The optimal solution relied on a first-order Euler discretization of the equations of motion to develop a series of state equations which could be optimized through MATLAB.

While the MATLAB optimized solution was too slow to implement on a MAV in real-time, [6] concluded that it could serve as a benchmark for future path planning algorithms designed for MAVs to account for wind.

3 USE CASES

A use case describes the interaction between the operator and the system as a sequence of simple steps. Use cases are useful for systems engineering analysis to translate system required capabilities to functions that can be coded in software or built into a system. The use cases associated with this research are discussed subsequently. While fully dressed use cases were developed in [7], only terse versions are presented here.

3.1 Orbit POI Use Case

The MAV starts on a pre-determined route surveillance mission. The Operator sees a POI in the user interface sensor display and directs the MAV to orbit the POI. The MAV exits the planned route, flies to the orbit, and orbits the POI, keeping the sensor aimpoint centered on the POI. The MAV may transition between sensors during this procedure to provide the best image to the Operator. After the Operator determines the MAV has orbited the POI sufficiently, the Operator commands the MAV to resume the pre-determined route.

3.2 Overfly POI Use Case

The MAV starts on a pre-determined route surveillance mission. The Operator sees a POI in the user interface sensor display and directs the MAV to overfly the POI. The MAV exits the planned route, flies to the POI overflight path, and flies over the POI using a specified sensor view angle. The MAV may transition between sensors during this procedure to provide the best image to the Operator. Upon completion of the overflight, unless otherwise directed by the Operator, the MAV will resume the planned route.

4 COORDINATE SYSTEMS

Throughout the research, a north-east-down reference system was maintained. This meant the x-axis pointed out the nose of the MAV, the y-axis pointed out the right wing, and the z-axis pointed towards the ground. This meant that all altitudes were technically in a negative z-direction, although anytime altitude was input into an algorithm, it was input as a positive value. If necessary, the equations added a negative sign to account for the altitude along the z-axis. All altitudes were above ground level altitudes, assuming a flat ground. This meant that z=0 at the ground.

Aircraft attitude angles were inertial aircraft attitudes. The right hand rule applied so that MAV yaw, ψ_a, went from

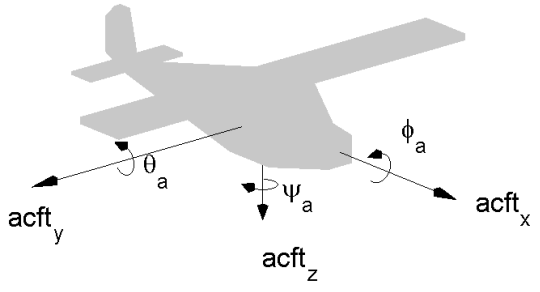

Figure 1: The coordinate system used for this research.

zero if the MAV's nose pointed due north, positive in a clockwise fashion; therefore, $\psi_a = 90^\circ$ had the MAV pointed due east and $\psi_a = -90^\circ = 270^\circ$ had the MAV pointed due west. Pitch, θ_a, became positive as the MAV's nose went away from the ground, keeping the conventional concept of positive pitch. Roll, ϕ_a, was positive as the left wing rose. Figure 1 shows the coordinate system used for the research.

The sensor orientation angles were relative pointing angles of the sensor to the body axis of the MAV. Sensor azimuth or yaw, ψ_s, went from zero if pointing out the nose and became positive in a clockwise fashion; therefore, $\psi_s = 90^\circ$ was out the right wing and $\psi_s = -90^\circ$ was out the left wing. Sensor elevation or pitch, θ_s, was negative as the sensor pointed away from the longitudinal axis of the MAV towards the ground; due to an assumption that the MAV would be used to surveil ground targets, positive sensor pitch was not considered. Sensor roll, ϕ_s, was always assumed to be zero. Figure 2 shows the sensor coordinate system used for the research.

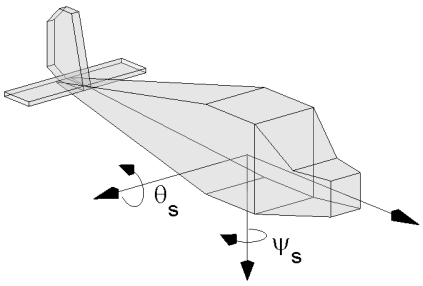

Figure 2: The sensor coordinate system used for this research. Note: the sensor vector is not displayed in this image.

The sensor was assumed to be located at the same Cartesian coordinates as the MAV. This was tantamount to assuming that the sensor was directly underneath the GPS antenna of the MAV. This was not a perfect assumption; however, the differences of centimeters from the actual location of the sensor to the location of the GPS antenna would not be noticeable in the sensor aimpoint determination, especially when the MAV was in the presence of wind.

5 ALGORITHMS

This research developed the algorithms to implement the *Orbit POI* and *Overfly POI* use cases, discussed Section 3 and fully developed in [7], along with supporting algorithms for estimating wind, for knowing the sensor aimpoint given a MAV position and attitude, and for knowing the footprint of the sensor field of view on the ground. The starting and ending points are a planned route for the MAV.

5.1 Sensor Aimpoint

The sensor aimpoint vector, $\vec{\text{aim}}$, is the ordered product of a sensor unit vector, $\vec{s}$, with two coordinate transformation matrices. The sensor unit vector, $\vec{s}$, aligns along the boresight or central axis of the sensor. The initial state of the sensor unit vector, as shown in Equation 1, aligns $\vec{s}$ with the longitudinal axis of the MAV. The unit vector relates to the airframe through a coordinate transformation matrix, $C_{a/s}$, that relates the sensor azimuth, ψ_s, and elevation, θ_s, to the airframe. This vector then relates to the inertial reference frame through a second coordinated transformation matrix, $C_{r/a}$, which accounts for the aircraft attitude. The sensor aimpoint vector points from the central point of the MAV to a location, or sensor aimpoint, in the inertial frame.

The angle between the ground and $\vec{\text{aim}}$ is found using Equation 2. The horizontal distance or radial distance, R_{aim}, from the position of the MAV above the ground to the sensor aimpoint is likewise found using Equation 3, where acft_z is MAV altitude above ground level. The bearing from MAV to aimpoint, ψ_{aim} is then calculated with Equation 4. Finally, the sensor aimpoint coordinates are found by summing the MAV position with the components of the horizontal vector from MAV to aimpoint and setting the altitude of the aimpoint to ground level, as indicated in Equation 5.

$$\begin{aligned} \vec{\text{aim}} &= C_{r/a}C_{a/s}\vec{s} \\ \vec{s} &= [1,0,0]^T \end{aligned} \tag{1}$$

$$\tan\theta_{aim} = \frac{\text{aim}_z}{\sqrt{\text{aim}_x^2 + \text{aim}_y^2}} \tag{2}$$

$$R_{aim} = \frac{\text{acft}_z}{\tan\theta_{aim}} \tag{3}$$

$$\psi_{aim} = tan^{-1}\frac{\text{aim}_x}{\text{aim}_y} \tag{4}$$

$$\text{sensor aimpoint} = \begin{bmatrix} (\text{acft}_x + R_{aim}\cos\psi_{aim}) \\ (\text{acft}_y + R_{aim}\sin\psi_{aim}) \\ 0 \end{bmatrix}^T \tag{5}$$

The set of waypoint coordinates at which the MAV must be to point the sensor at the sensor aimpoint coordinates is called the sensor-on-POI (SoP) Waypoint. The SoP Waypoint is found by taking the POI ground coordinates relative to the MAV, $(\text{POI}_x, \text{POI}_y)$ and subtracting the sensor aimpoint offsets shown in Equation 6. The waypoint altitude and airspeed remain the same as the MAV's altitude, acft_z, and airspeed, V_a, at the time the algorithm was initiated. The SoP Waypoint values are given in Equation 7.

$$\begin{aligned} \Delta x &= (\text{acft}_x + R_{aim}\cos\psi_{aim}) \\ \Delta y &= (\text{acft}_y + R_{aim}\sin\psi_{aim}) \end{aligned} \tag{6}$$

$$\text{SoP Waypoint} = [\text{POI}_x - \Delta x, \text{POI}_y - \Delta y, \text{acft}_z, V_a] \tag{7}$$

5.2 Sensor Footprint

The sensor footprint is the projection of the sensor field of view (FOV). The sensor was assumed to have a horizontal FOV (FOV_H) and a vertical FOV (FOV_V). For the untransformed sensor unit vector, $\vec{s}$, the sensor footprint would be a parallelogram in front of the MAV. Once rotated through the same coordinate transformation matrices used for the aimpoint vector described in Section 5.1, the footprint would describe the portion of the ground visible to the sensor. For the sake of this research, only the sensor footprint as projected on the ground is considered; if the footprint was in the air, it was considered invalid as the POI was assumed on the ground.

5.3 Orbit Algorithm

The orbit algorithm relies solely on the side sensor while the MAV is in the orbit. An operator could use the front sensor to get the MAV into position, but only the side sensor would keep the POI in focus throughout the orbit. The side sensor is assumed to be situated with ψ_s of $\pm 90^\circ$ so that it is directly perpendicular to the longitudinal axis of the MAV. If this assumption is not made, then the MAV will have a spiral flight path around the POI, spiraling towards the POI if the absolute value of $\psi_s$is less than 90°, and spiraling away from the POI if the absolute value of $\psi_s$is greater than 90°. The algorithm changes the side sensor azimuth to $\pm 90^\circ$, based on if the sensor azimuth is less than 90° or not.

The orbit algorithm finds a series of n-waypoints in an approximate circle. Each waypoint is found using the sensor aimpoint algorithm discussed in Section 5.1. The inputs for the sensor aimpoint algorithm are sensor attitude (ψ_s, θ_s), MAV attitude $(\psi_a, theta_a, phi_a)$, MAV position $(\text{acft}_x, \text{acft}_y, \text{acft}_z)$, MAV airspeed (V_a), and POI coordinates $(\text{POI}_x, \text{POI}_y)$.

This research assumes sensor attitude and POI coordinates are fixed. The algorithm assumes that altitude, acft_z, and airspeed, V_a, will remain constant throughout the orbit. The waypoints are calculated relative to the MAV allowing two of the position coordinates, acft_x and acft_y, to be assumed 0. MAV pitch, θ_a, is assumed to be negligible in the turn, thus facilitating the use of a small angle approximation. This is not a perfect assumption, but the pitch angle would be sufficiently small to not heavily impact the sensor aimpoint.

MAV heading, ψ_a, depends on the wind speed, V_w, and direction χ_w, MAV airspeed, V_a, and the instantaneous ground track, χ_g for the MAV at each waypoint in the orbit. Since the orbit is a series of n segments or legs instead of a circle, ground track is assumed constant for the duration of that segment. The known values, $V_w, \chi_w, V_a,$ andχ_g, do not produce fully realized vectors for either the ground or the air information; in each case, only one of the two required values are known. Two cases exist, one for $\psi_a < \chi_g$ and the other for $\psi_a > \chi_g$. The solutions for ψ_a are shown in Equation 8.

$$\begin{gathered} \chi_{rel} = \sin^{-1}\left(\frac{V_w}{V_a}\sin\left(\chi_g - \chi_w\right)\right) \\ \psi_a = \begin{cases} \chi_g - 2\pi + \chi_{rel} & \text{, for } \psi_a < \chi_g \\ \chi_g - \chi_{rel} & \text{, for } \psi_a > \chi_g \end{cases} \end{gathered} \tag{8}$$

The algorithm calculates the bank angle, ϕ_a, required by the MAV at the SoP waypoint. The bank angle is found as a function of orbit radius. The orbit radius for an air vehicle can be found using two methods, as indicated in Equations 9 and 10. Equation 9 calculates the orbit radius as a function of ground speed, V_g, the gravity constant, g, and the bank angle, ϕ_a. Equation 10 calculates the orbit radius as a function of aircraft altitude, acft_z, bank angle, ϕ_a, and sensor elevation, θ_a. By taking Equations 9 and 10 and setting them equal to each other, ϕ_a can be solved using the quadratic equation for the solution given in Equation 11.

$$R_1 = \frac{V_g^2}{g\tan(\phi_a)} \tag{9}$$

$$R_2 = \frac{\text{acft}_z}{\tan(\phi_a + \theta_s)} \tag{10}$$

$$\begin{gathered} a = (\text{acft}_z g \tan\theta_s) \\ b = (-V_g^2 + \text{acft}_z g) \\ c = (V_g^2 \tan\theta_s) \\ \phi_a = \tan^{-1}\left(\frac{b \pm \sqrt{b^2 - 4ac}}{2a}\right) \end{gathered} \tag{11}$$

It is possible to result in bank angles that have complex or imaginary values based on the input ground speed and altitude. As the airspeed decreases, the bank angle required to maintain the same radius of an orbit also decreases; therefore, when this situation occurs, the algorithm iterates the process to solve for heading, ground speed, and bank angle by incrementally changing the commanded airspeed for the waypoint in -0.25 m/s steps.

The solution to Equation 11 produces two results for ϕ_a, both of which are valid bank angles. The algorithm chooses the lesser of the two bank angles, or the bank angle closest to zero, for the desired bank at the waypoint. This is a practical decision which allowed a better view of the side of a POI rather than a more overhead image. The algorithm could be modified if the user desired overhead images instead of side images.

The process to find a single waypoint in the orbit is repeated n-times to produce n-waypoints around the POI. The number of waypoints pass as an input to the algorithm, so the algorithm can handle any number of desired waypoints. As the process iterates to generate the waypoint list, the algorithm increments the commanded ground track, χ_g, as indicated in Equation 12. If the sensor points out the right wing,

χ_g increases so that the MAV will make right hand or clockwise turns about the POI; if the sensor points out the left wing, χ_g decreases with each increment.

$$\chi_g|_{new} = \chi_g|_{old} \pm \frac{360^\circ}{n} \tag{12}$$

5.4 Overflight Algorithm

The overflight algorithm focuses on establishing waypoints for a preferred sensor orientation or look angle on the POI. The overflight algorithm finds the necessary MAV heading, ψ_a, from the preferred look angle, χ_L, and the sensor azimuth, ψ_s. The relation is given in Equation 13. Neither the look angle nor the heading are dependent on wind in this case, since the ground track heading, χ_g, has not been established.

$$\psi_a = \chi_L - \psi_s \tag{13}$$

Given the MAV heading and altitude, the overflight algorithm calls the sensor aimpoint algorithm to get a necessary offset from the POI for the sensor-on-POI waypoint. The overflight algorithm assumes the MAV is in steady, level flight at this waypoint; therefore, the MAV has no pitch and no bank. While the actual orientation of the MAV at that waypoint may have pitch and bank, for planning purposes, pitch and bank are not necessary.

The sensor-on-POI waypoint coordinates are found using Equation 7, as discussed in Section 5.1. To increase the likelihood the MAV will have the sensor aimed at the POI at the sensor-on-POI waypoint, two additional waypoints are found, one "upstream" of or before the sensor-on-POI waypoint, and the other "downstream" or past the sensor-on-POI waypoint, as indicated in figure 3. This creates a flight path that has the sensor aimed at the POI at the preferred sensor orientation by allowing the sensor to sweep over the POI as the MAV flies along the path. The "upstream" waypoint is an initial point prior to the POI being in the sensor FOV. It is intended to ensure the MAV is in a stead, level flight when the MAV reaches the SoP waypoint. The "downstream" waypoint is intended to ensure the POI has completed passed through the FOV before the MAV initiates a new flight command.

The additional waypoints need to be along a straight line ground track for the MAV to fly in the presence of wind. The required heading, ψ_a, for the MAV along the projected path is found by Equation 13. MAV airspeed, V_a, the wind speed, V_w, and wind direction, χ_w, were assumed constant. The ground track, χ_g, for the projected path, was calculated as a component of the resultant of summing the wind vector and the airspeed/heading vector.

The "upstream" and "downstream" waypoints are then found by taking offsets from the sensor-on-POI waypoint. As before, the altitude and airspeed remain the same as the MAV's altitude, acft_z, and airspeed, V_a at the time the algorithm was initiated. A magnitude, l, is required to space the "upstream" and "downstream" waypoints from the sensor-on-POI waypoint. This magnitude could be any value as long as the additional waypoints are far enough from the sensor-on-POI waypoint so that the MAV flies straight and level as it encountered the sensor-on-POI waypoint. The magnitude, l, is set equal to the maximum distance the sensor footprint reached along the ground. In this manner, if the MAV flies in a no-wind situation while using the front sensor, the "upstream" waypoint is just before the POI entered the sensor field of view. The "downstream" waypoint is an equal distance away, but after the sensor aimpoint passed over the POI. Equation 14 gives the solution for the "upstream" and "downstream" waypoints, with subtraction for the former and addition for the later.

Figure 3: The waypoints from the overflight algorithm.

$$\text{Overflight Waypoints} = \begin{bmatrix} \text{SoP}_x \mp l\cos\chi_g \\ \text{SoP}_y \mp l\sin\chi_g \\ \text{acft}_z \\ V_a \end{bmatrix}^T \tag{14}$$

6 Test Equipment

The algorithms were written in MATLAB. Inputs were generated in script files which then called the algorithms. Outputs were waypoint flight plans compatible with the flight management software used for testing. The test equipment was all commercial off-the-shelf (COTS) equipment, including a Procerus Technologies Kestrel autopilot, Procerus Technologies Virtual Cockpit flight management software, the Aviones environmental simulator, and a SIG Rascal 110 radio-controlled aircraft modified to use the Kestrel autopilot. A hardware-in-the-loop (HIL) configuration was used by simulating the flight environment with Aviones. This data was input into the Kestrel autopilot which determined command responses to navigate to the waypoints commanded through Virtual Cockpit. Virtual Cockpit also received telemetry from the Kestrel; the Kestrel calculated this telemetry both in HIL tests and flight tests.

The SIG Rascal 110 is a COTS radio-controlled aircraft that employs a high-wing and conventional landing gear,

more commonly known as a tail-dragger configuration. It has a four-cycle aircraft engine as the main power plant. The Rascal uses a Kestrel autopilot that connects to a pitot-static system and a GPS antenna. It has a cruise speed of 40 kts (20.6 m/s) and a stall speed of 20 kts (10.3 m/s). The maximum bank angle was established as $40°$. More information regarding the test equipment is available in [7].

One limitation of the Aviones was that it could not simulate variable winds. The physics parameter file could be manually changed while the simulation was running, including winds, but this was a tedious and slow process. An external program, *e.g.*, MATLAB, could not change the file and have Aviones reread the physics parameters. This meant that only constant winds were available for HIL testing.

7 TESTING

As mentioned above, testing involved both HIL testing and flight testing. Flight test data was gathered early in the research. The test points accomplished during flight testing were repeated as HIL test points; however, due to the limitations from Aviones, constant wind was assumed for the HIL versions of the test points. The test data was compared and indicated that HIL testing provided an accurate evaluation of the utility of the algorithms. The comparison is not given here for the sake of brevity; those interested in better understanding the comparison are referred to [7].

7.1 Orbit Algorithm Testing

The orbit algorithm was tested through 15 different configurations, varying altitude, from 50 m to 200 m in 50 m increments, and wind speed, from 0 kts (0 m/s) to 25 kts (12.9 m/s) in 5 kts (2.6 m/s) increments. In all cases, the number of waypoints was set to 18 and the wind was out of the east with a heading, χ_w, of $270°$. Since wind speed can force the orbit algorithm to iterate altitude, certain test points were not necessary to evaluate. For example, for a desired altitude of 50 m and a wind speed of 10 kts (5.1 m/s), the algorithm generates waypoints at 110 m altitude. The same is true for a desired altitude of 100 m and a 10 kts (5.1 m/s) wind; therefore, the 50 m desired altitude test case was not evaluated. In between each change in wind speed, the MAV was instructed to fly at least 2 orbits at the rally point to allow the Kestrel to recalibrate for the new wind speed.

The test results shown in Table 1 demonstrate how wind speed and altitude affect the ability for the MAV to keep the POI in the FOV. The trend is that, for a given altitude, as wind speed increased, the ability to keep the POI in the FOV decreased. When the MAV had a tail wind at the top of the orbit, it continually overshot its turn and had to correct to get back on the commanded path. The correction required a bank larger than the waypoint generation algorithm calculated and the rate of the POI being in the FOV decreased.

Figures 4-7 illustrates the overshoot for the 150 m altitude orbit case, which shows the plots of the orbit paths for each of the 150 m altitude cases described in Table 1. It must be recalled that the orbit algorithm calculates the location of the waypoint to keep the MAV sensor pointed directly at the POI. The nature of the open-loop algorithm did not have it correct the flight controls when the MAV was blown off course.

Table 1: Orbit Aimpoint Statistics

Des Alt	Cmd Alt	V_w	RMS	% Time in FOV
50m	50m	$0kts$	14.46m	98.11%
50m	80m	$5kts$	31.58m	86.78%
100m	100m	$0kts$	18.02m	100.00%
100m	100m	$5kts$	47.68m	82.26%
100m	110m	$10kts$	86.23m	67.39%
150m	150m	$0kts$	18.03m	100.00%
150m	150m	$5kts$	34.86m	97.12%
150m	150m	$10kts$	113.32m	71.35%
150m	150m	$15kts$	128.28m	55.33%
200m	200m	$0kts$	21.70m	99.44%
200m	200m	$5kts$	49.12m	99.32%
200m	200m	$10kts$	68.66m	95.32%
200m	200m	$15kts$	80.53m	88.46%
200m	200m	$20kts$	94.94m	80.19%
200m	250m	$25kts$	181.14m	71.43%

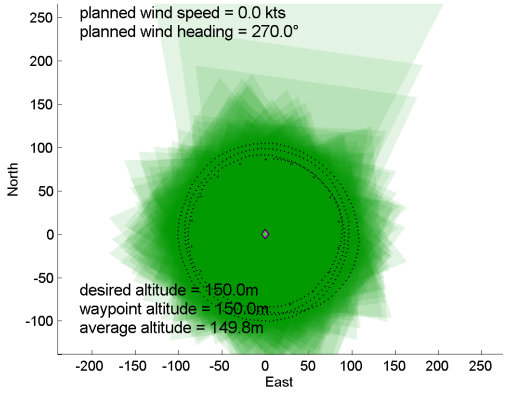

Figure 4: There was little to no overshoot in the zero wind condition.

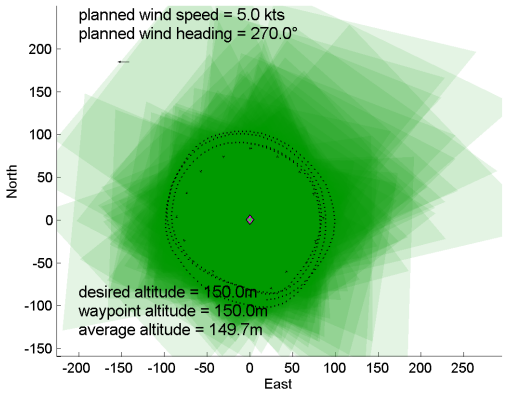

Figure 5: There was limited overshoot in 5kts of wind.

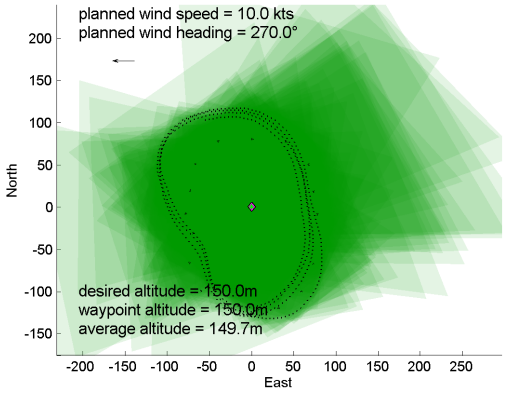

Figure 6: The overshoot increased at 10kts of wind.

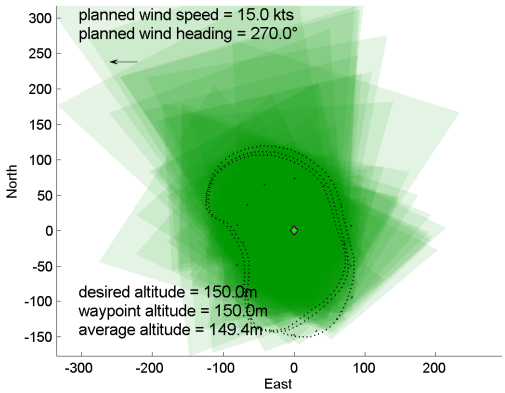

Figure 7: The MAV was blown off course and over-corrected at 15kts of wind.

For a given altitude, as the wind speed increased, the average error distance, described as RMS in Table 1, between the actual sensor aimpoint location and the POI location increased. The increase in RMS directly couples to the inability of the MAV to keep to the commanded flight plan. As the MAV overshot the path and corrected back to course, the actual sensor aimpoint moved away from the POI. The more aggressively the MAV had to correct, the larger the RMS became.

From the perspective of an operator, operational useful data likely comes any time the POI is in the sensor FOV, not just when it is centered, or aligned to the sensor boresight, in the FOV. Therefore, the metric was calculated for the percentage of the time the POI was in the FOV. The data rate was captured at 1 Hz, so a summary was done of the number of seconds the POI was in the sensor FOV. This was divided by the total duration of the given orbit. For example, on the 150 m orbit with 5 kts (2.6 m/s) of wind, the POI was in the FOV for a total of 135 seconds during the 139 second orbit, yielding the rate of 97.12%. As a related note, as the MAV altitude increased, the area captured by the sensor footprint increased, thus the likelihood that the POI remained in the FOV also increased.

7.2 Overflight Algorithm Testing

The overflight algorithm was tested at nine different test points with the front sensor only. The algorithm generates a flight plan based on the selected sensor, but that does not affect how the MAV flies the commanded path. Capturing telemetry for the MAV for both the front and the side as the MAV flies the same path through the air was equivalent to post-processing the data; therefore, testing was simplified by only considering the front sensor. All test points were flown at 100 m altitude, at the cruise speed for the Rascal of 40 kts (20.6 m/s), and with a wind out of the east with a heading of 270°. A total of nine test points considered three wind speeds, 0 kts (0 m/s), 10 kts (5.1 m/s), and 20 kts (10.3 m/s), and three look angles, 045°, 225°, and 360°. The look angles of 090° and 270° were not considered as they provided a pure headwind and tailwind respectively. This would slow or speed up the MAV, but not change the output of the overflight path. The look angles of 135°, 180°, and 315° provided similar crab angles to the path as those from the three look angles used.

As with the orbit testing, in between each change in wind speed, the MAV was instructed to fly at least 2 orbits at the rally point to allow the Kestrel to recalibrate for the new wind speed.

The data in Table 2 indicate that the overflight algorithm can capture the POI in the FOV in each of the wind situations. Data was not collected at higher winds, however, because the MAV could not reach the commanded flight path. Depending on the relation between the wind and the flight path angle, the wind can actually help the MAV turn and keep the POI

Table 2: Overflight Aimpoint Statistics

χ_L	V_w	**Duration POI in FOV**	**% Time in FOV**
$45°$	$0kts$	10 s	100.00%
$225°$	$0kts$	3 s	100.00%
$360°$	$0kts$	10 s	100.00%
$45°$	$10kts$	6 s	100.00%
$225°$	$10kts$	5 s	100.00%
$360°$	$10kts$	4 s	100.00%
$45°$	$20kts$	9 s	100.00%
$225°$	$20kts$	6 s	100.00%
$360°$	$20kts$	4 s	100.00%

in the FOV for a longer duration. Additionally, the relation between the approach heading and the flight path heading affects how soon the POI enters the FOV. The later the POI enters the FOV, the shorter duration an operator can view the POI. Future research will investigate using a Dubins path to get the MAV to the flight path, allowing the MAV to align to the flight path and place the POI in the FOV for a longer period of time.

The lack of time of the POI in the FOV may reduce the utility of the algorithm to operators. As mentioned above in the orbit section, the POI can be kept in the FOV longer by flying the MAV at a higher altitude. The trade off in this case is a reduction in pixel density on the POI. This trade space was not a consideration in the development of the algorithm. Another potential factor to increase the amount of time the POI remained in the FOV for the overflight case would be to reduce the cruise speed for the MAV, an option not explored by this research.

8 Conclusion

Fixed sensors are ideal for use on MAVs as they tend to be lighter, cheaper, and more robust than gimbaled sensors. However, the coupling of the fixed sensor to the flight dynamics of the MAV can make their use more complicated in an operational environment. Most operational uses of MAVs for ISR depend on the sensor output rather than MAV location. This research generated flight plans of a series of waypoints. At these waypoints and the presumed attitude, the fixed sensor on the MAV will aim at the POI. Two scenarios were explored. The first was for a MAV orbiting the POI. The second scenario was for the sensor to overfly the POI along a preferred sensor look angle.

The conclusion from the HIL orbit tests was that the flight plan generated by the algorithm kept the POI in the FOV at least two-thirds of the time as long as the wind speed was less than one-quarter of the cruise speed. For the test equipment used, this meant that as long as the wind speed was less than 10 kts (5.1 m/s), the open-loop flight plan generated by the algorithm could keep the POI in the FOV for greater than 66% of the time. The conclusion from the HIL overflight tests was that the flight plan generated by the algorithm put the POI in the FOV every time, but the duration the POI remained in the FOV was dependant on the ground speed of the MAV; in headwinds, the POI may remain in the FOV longer, but with tailwinds the duration shrinks.

The HIL testing both confirmed the results seen in test flight as well as evaluated the algorithms at many more conditions than available in test flight. One limitation of the HIL testing was the inability to simulate variable winds; therefore, outside of flight test, the algorithms were only tested in a constant wind environment. The imperfect results for keeping the POI in the sensor FOV stem from the nature of this open-loop control. The waypoints generated by these algorithms presume the MAV attitude properly adjusting for constant wind. Reliance on the closed-loop controls of the Kestrel autopilot resulted in the MAV not reaching some of the commanded waypoints.

Future research will consider closed-loop control to provide improved performance in the presence of high wind speed relative to the airspeed of the MAV.

References

[1] J. Osborne and R. Rysdyk. Waypoint Guidance for Small UAVs in Wind. *AIAA Infotech@Aerospace*, pages 1–12, 2005.

[2] R. Rysdyk. Unmanned Aerial Vehicle Path Following for Target Observation in Wind. *Journal of Guidance, Control, and Dynamics*, 29(5):1092–1100, September-October 2006.

[3] B. Robinson. An Investigation into Robust Wind Correction Algorithms for Off-the-Shelf Unmanned Aerial Vehicle Autopilots. Master's thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH, 45433-7765, June 2006.

[4] T. G. McGee, S. Spry, and J. K. Hedrick. Optimal Path Planning in a Constant Wind with a Bounded Turning Rate. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005.

[5] L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497–516, July 1957.

[6] M. Zollars. Optimal Wind Corrected Flight Path Planning for Autonomous Micro Air Vehicles. Master's thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH, 45433-7765, March 2007.

[7] S. Farrell. Waypoint Generation based on Sensor Aimpoint. Master's thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH, 45433-7765, March 2009.