Precise automated landing of a fixed-wing aircraft onto a moving platform

Mohamed Zahier Parker[1], and *Jacobus Adriaan Albertus* Engelbrecht[1]

[1]Department of Electrical and Electronic Engineering, Stellenbosch University, Stellenbosch, South Africa

Abstract. This paper presents flight and guidance control systems that can accurately land a fixed-wing unmanned aerial vehicle onto a small moving platform. The flight control system uses a hybrid architecture that combines classical control with model predictive control. The guidance control system uses a guidance algorithm, waypoint scheduler and state machine to allow the aircraft to complete circuits around the airfield. A prediction algorithm calculates the touchdown point between the aircraft and the moving platform. The state machine provides references to the flight control system to allow the aircraft to reach this touchdown point. The control system is implemented in PX4 autopilot software and verified using simulation. A fixed-wing aircraft is constructed using a model airframe with a Pixhawk 4 autopilot hardware and other additional hardware components. In simulation, the control system lands the aircraft on a moving platform with an accuracy of 22 cm, while for the practical moving platform landing, the control system lands the aircraft with an accuracy of 75 cm. Preparations are being done to perform the final practical moving platform landing tests.

1 Introduction

This section starts by introducing the background to this research project, thereafter the literature review is performed and finally, the research contributions produced from this research project are outlined.

1.1 Background

The demand for a system to perform an automated landing of fixed-wing unmanned aerial vehicles (UAVs) has increased in recent times. This is due to more use cases of UAVs being developed for both military and commercial applications. An example military application is to land fixed-wing UAVs on aircraft carriers. An example commercial application is to have a moving, ground-based delivery vehicle with multiple fixed-wing UAVs that take-off and land to perform local deliveries.

A significant amount of research and experimentation is being done with fixed-wing UAVs with many companies using UAVs for different applications. R Pavithran *et al* [1] researched a prototype fixed-wing UAV to deliver medical supplies such as vaccines to rural

areas. R Pavithran *et al* found that the use of the UAV allows for a decrease in transport time and an increase in power efficiency. P Manapongpun *et al* [2] presented a drone that is capable of performing fully autonomous drone missions for surveillance. This drone system does not require any direct human interaction. KN Tahar *et al* [3] utilised a fixed-wing drone to obtain images for aerial mapping. The system developed was able to map the surrounding area at a fairly reasonable accuracy.

One of the challenges of using a fixed-wing UAV is the space needed for take-off and landing. The use of a moving platform with an arrestor system can address this challenge and allow for easier retrieval of the UAV. This can open new applications for UAVs which were previously not viable due to space limitations. To achieve this goal, a system will be required to be able to automatically land the UAV on the moving platform. This paper presents a system that is capable of performing this landing.

1.2 Literature review

This research project is performed at the Electronic Systems Laboratory (ESL) at Stellenbosch University where many other fixed-wing UAV research projects have been previously performed. These research projects considered different scenarios for the fixed-wing UAV, including landings. The work in common between all these research projects is the design of a control system for the UAV. One of the first fixed-wing UAV research projects was produced by Iain Peddle [4] who designed an acceleration-based control architecture to control both the lateral and longitudinal dynamics of the UAV. Many of the later research projects built on Peddle's design and adapted it for their applications, for example, the research projects Hugo [5] and Goosen [6]. Hugo presented a system that could land a fixed-wing UAV with partial horizontal stabiliser loss, however, Hugo focused more on getting the aircraft down safely than a precision landing, which this research project focuses on.

Some of the latest research projects in the ESL for the fixed-wing UAV were produced by De Bruin [7] and Le Roux [8]. De Bruin's research project focused on landing the UAV on a runway in crosswind conditions by using Acceleration-Based Direct Lift control. De Bruin achieved great success with his design and tested it practically where he landed the UAV accurately on a runway. Le Roux's research project focused on landing the UAV on a moving platform by using Total Energy Control System(TECS) methods. He achieved success in simulation however, due to a hardware fault, his UAV crashed resulting in him being unable to verify his results practically.

UAV control has been researched by many institutions using different methods such as model predictive control (MPC), neural networks and vision-based control. Yi Feng[9] explores using MPC methods to land a rotary-wing UAV onto a moving platform. Even though the UAV used is a rotary-wing quadrotor, its MPC methods can be extended to a Fixed-wing UAV. Amadi [10] used an MPC to control a rotary-wing UAV's angular rates. His MPC successfully controlled the rates both in simulation and on a real quadcopter. Kayacan [11] uses neural networks to control a fixed-wing UAV and is successful in simulation tests. Brukarczyk [12] uses vision-based algorithms with an automatic landing system to land the UAV. These research projects use different methods but achieve the same goal of landing the UAV.

Our research project considers using a hybrid control system combining classical acceleration-based control and model predictive control (MPC). The classical control method

was chosen due to it being stable and predictable. The MPC method is added to provide highly accurate glideslope tracking which is important for precision landing.

1.3 Research contributions

Some of the mentioned literature either focused on landing a rotary-wing UAV on a moving platform or landing a fixed-wing UAV onto a runway. Leroux attempted to land a fixed-wing UAV onto a moving platform however he could not verify his system on a practical vehicle. This research project aims to design a system that is capable of executing the moving platform landing on a practical vehicle. By achieving this aim, the following research contributions will be provided:

- A Flight Control System (FCS) will be designed that is capable of controlling the local states of the aircraft to keep it in stable flight.
- A Guidance Control System (GCS) will be designed that is capable of allowing the aircraft to follow a trajectory and land on the moving platform.
- A fixed-wing UAV will be assembled that can be used to practically test the FCS and GCS performance in the real world.
- A moving platform will be assembled that, together with the fixed-wing UAV, can be used for the practical moving platform landing.

2 Methodology

This section first introduces the system used to control the aircraft to perform the automated landing. After that, the moving platform landing procedure is demonstrated. The software environment used to implement the Software in the Loop (SITL) simulation is then presented. Finally, the hardware and location used to perform the practical tests are then discussed.

2.1 Automated landing system development

A Flight Control System (FCS) and Guidance Control System (GCS) are developed to control the local states of the aircraft and allow it to follow a trajectory.

2.1.1 Flight control system development

The fixed-wing UAV is represented by a non-linear model which is derived from the UAV's equations of motion. To develop the flight control system, the non-linear model is linearised around an equilibrium point which is when the aircraft flies straight in level flight. At this equilibrium point, the UAV has a trim thrust and elevator deflection applied to it and the FCS applies its commands with respect to these trim values. The FCS is developed in MATLAB using the linearised model of the fixed-wing UAV. The non-linear UAV model is then used to verify the FCS's performance in Simulink. The block diagram in figure 1 shows a simplified view of the controllers applied to the non-linear fixed-wing UAV model.

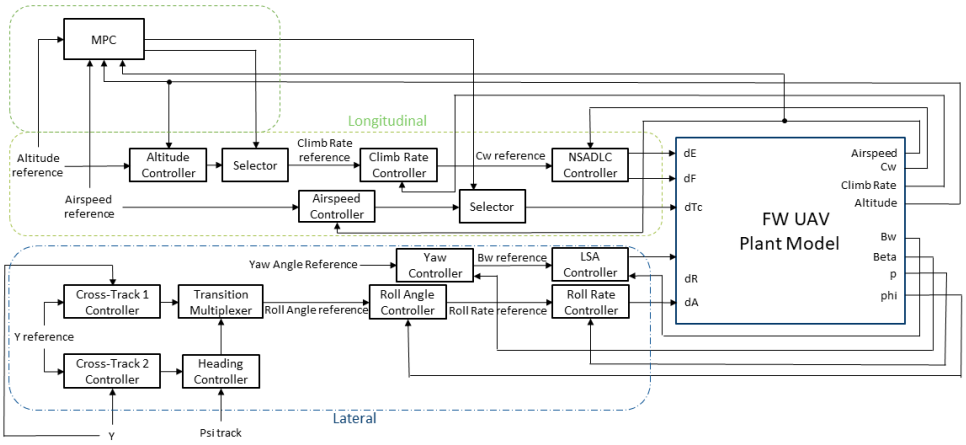

Fig. 1. Block diagram showing all the controllers of the flight control system. The controllers are split into longitudinal and lateral groups.

The FCS consists of a hybrid architecture combining classical control and model predictive control (MPC). The classical controllers, which are derived from de Bruin [7], Peddle [4] and Le Roux [8], are operated in an enclosed loop configuration where the outer loops feed the inner loops. The classical controllers are split into two sets, namely the longitudinal and lateral sets so that they can control the longitudinal and lateral dynamics of the aircraft independently of each other.

The longitudinal controllers consist of airspeed, Normal Specific Acceleration Direct Lift Control (NSADLC), climb rate, and altitude controllers. The airspeed controller controls the airspeed of the UAV using the thrust command (ΔT_c). The NSADLC, climb rate, and altitude controllers control the vertical characteristics of the aircraft by manipulating the elevator and flap deflections.

The lateral controllers consist of Lateral Specific Acceleration (LSA), yaw, roll rate, roll angle, heading, and two cross-track controllers. The LSA and yaw controllers use the rudder deflection to control the yaw characteristics of the aircraft. The remaining controllers are used to control the direction that the aircraft is flying. The transition multiplexer is used to select which roll angle reference to give to the roll angle controller based on the aircraft's cross-track error. This allows the aircraft to have both zero cross-track error at steady-state and to follow the trajectory while being far from the track.

2.1.2 Classical altitude and airspeed controller landing configuration

The classical altitude and airspeed controllers are normally replaced by the MPC during the landing phase however they are still capable of landing the UAV themselves. It is decided to develop the classical altitude and airspeed controllers for landing so that their results can be compared to the MPC's results to determine if there is an improvement in using the MPC.

The classical airspeed controller does not need any modification from its normal configuration for landing and therefore it is left unchanged. The classical altitude controller does however need a change as it struggles to track the glideslope with its normal configuration during landing. The modification to the altitude controller is shown in figure 2 below.

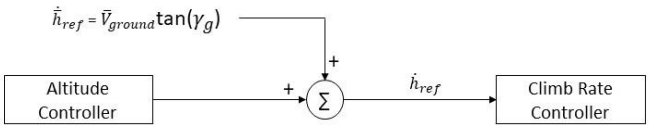

Fig. 2. Block diagram showing the modification to the classical altitude controller to allow it to follow a glideslope. The modification consists of adding the desired climb rate for glideslope tracking to the altitude controller output.

The classical altitude controller is too slow in providing the correct climb rate reference to the climb rate controller to allow the UAV to descend at a constant rate so that it can follow the glideslope(ramp). Since the glideslope angle (γ_g) is known beforehand, the desired climb rate reference ($\dot{\bar{h}}_{ref}$) can be fed forward to the climb rate controller using the UAVs ground velocity ($\bar{V}_{ground}$) as shown in figure 2. The classical altitude controller will then work around this desired value to ensure the UAV tracks the glideslope.

2.1.3 Model predictive control development

The MPC controller replaces the classical altitude and airspeed controllers during the final phases of flight to improve landing accuracy performance. The MPC controls the aircraft's altitude and airspeed by commanding the climb rate reference and thrust command respectively during the landing phase of flight. The climb rate reference is used by the climb rate classical controller and the thrust command is used by the motor. During the landing phase, the MPC will decrease the aircraft's altitude while maintaining airspeed to ensure the aircraft is on the predetermined glideslope to accurately intersect the moving platform. The MPC can track the glideslope faster and more accurately compared to classical controllers hence why it is added to the flight control system.

The MPC controller uses an architecture described in Wang [13] and Amadi [10] which utilises Hildreth's quadratic programming to obtain the control actions applied to the system. The MPC is operated in a multiple input multiple output (MIMO) configuration to control both the altitude and airspeed simultaneously. Figure 3 shows a simplified overview of the MPC operation.

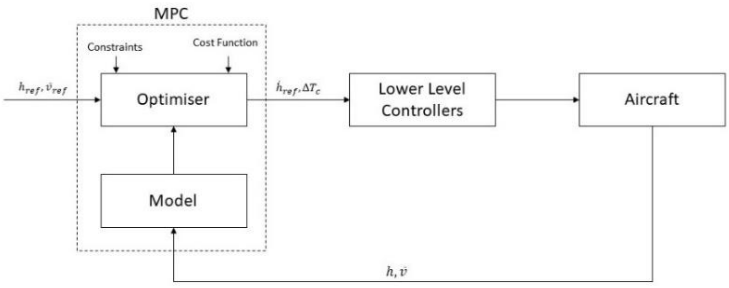

Fig. 3. Block diagram showing the MPC operation. The MPC consists of the optimiser, model, constraints, and cost function.

Figure 3 shows that the MPC uses a model of the fixed-wing UAV to predict the UAV's future behaviour. The optimiser uses this model with the cost function and constraints to

generate the ideal control actions (climb rate reference and thrust command) to apply to the UAV. The optimiser used by the MPC is Hildreth's quadratic programming method.

The MPC controller uses the following compact form of the model.

$$\dot{\mathbf{x}}_{mpc}=A_{mpc}\mathbf{x}_{mpc}+B_{mpc}\mathbf{u}_{mpc} \quad (1)$$

$$\mathbf{y}_{mpc}=C_{mpc}\mathbf{x}_{mpc} \quad (2)$$

where,

$$\mathbf{x}_{mpc}=[\bar{v},\alpha,q,\theta,\Delta T,e_f,e_c,e_r,h\,]^T\ ;\ \mathbf{u}_{mpc}=\left[\dot{h}_{ref},\Delta T_c\right]^T\ ;\ \mathbf{y}_{mpc}=[h,\bar{v}]^T \quad (3)$$

The MPC model consists of the linear longitudinal UAV model augmented with the NSADLC and climb rate controllers. The MPC state vector ($\mathbf{x}_{mpc}$) consists of airspeed ($\bar{v}$), angle of attack (α), pitch rate (q), pitch angle (θ), change in thrust (ΔT), NSADLC controller integrators (e_f , e_c), climb rate controller integrator (e_r) and altitude (h) states. The MPC's manipulated variables ($\mathbf{u}_{mpc}$) are the climb rate reference ($\dot{h}_{ref}$) and the thrust command (ΔT_c) while the measured outputs ($\mathbf{y}_{mpc}$) are altitude (h) and airspeed ($\bar{v}$).

The MPC uses the cost function shown below to calculate the ideal control actions to apply to the aircraft.

$$J=\frac{1}{2}\Delta\mathbf{U}^T E\Delta\mathbf{U}+\Delta\mathbf{U}^T F \quad (4)$$

where,

$$E=2\left(H^T H+W\right) \quad (5)$$

$$F=-2H^T\left(R_S-P\mathbf{x}(k)\right) \quad (6)$$

$\Delta\mathbf{U}$ is the control action, R_S is the reference matrix, H and P are the prediction matrices, W is the weight matrix, $\mathbf{x}$ is the state vector, and k is the time step. The E and F matrices are quadratic programming variables used in the Hildreth programming algorithm. The weight matrix(W) is tuned until the output response has the desired settling time, overshoot and oscillation.

The constraints are only considered for the input variables ($\dot{h}_{ref}$, ΔT_c) and their limits are shown below:

$$-2 \le \dot{h}_{ref} \le 2 \quad (7)$$

$$-26.5513 \le \Delta T_c \le 13.4487 \quad (8)$$

The climb rate is limited to ±2 m/s to ensure that the aircraft does not rapidly change its altitude which can put strain on the airframe. Since the UAV model is linearised around trim, the trim thrust of 26.5N is added to the MPC thrust command. Therefore, the MPC thrust command has to account for this by offsetting its thrust constraints by the trim thrust. This means that with respect to the motor, the MPC can command a thrust between 0N to 40N which is the physical thrust range that the motor can supply.

The constraints are applied to the control action ($\Delta\mathbf{U}$) in the Hildreth quadratic function so that it satisfies the equation below:

$$CC\Delta\mathbf{U} \le d \quad (9)$$

These constraints are used to restrain the MPC so that it is not overly aggressive on the aircraft.

2.1.4 Guidance control system

The guidance control system consists of the guidance algorithm, waypoint scheduler, landing position predictor and state machine.

The guidance algorithm is used to generate states for the FCS to allow it to make the aircraft follow a trajectory. It does this by comparing the aircraft's current state with respect to the desired track it should maintain. The states it generates are the aircraft's cross-track error, in-track distance and track heading. A waypoint scheduler is used to change the current waypoints being tracked.

The landing position predictor calculates the predicted touch-down point between the UAV and moving platform based on their relative positions and velocities. A state machine is used to provide references to the FCS which change depending on the current state of flight. The state machine is also used to determine when the decrab manoeuvre should occur and if the aircraft is stabilised to land.

2.2 Landing procedure

The landing procedure used for the research project tries to imitate a real-life scenario, such as the delivery fixed-wing UAVs with a moving home base, while also accounting for the physical limitations of the hardware and environment used for the practical test flights.

The aircraft will first complete a circuit around the airfield and align itself with the runway. In the delivery UAV scenario, this would represent the UAV returning to the home base after it has completed its delivery. When the UAV is on final approach it will start to prepare for the intersection with the moving platform. Figure 4 shows the landing procedure of the aircraft with the moving platform.

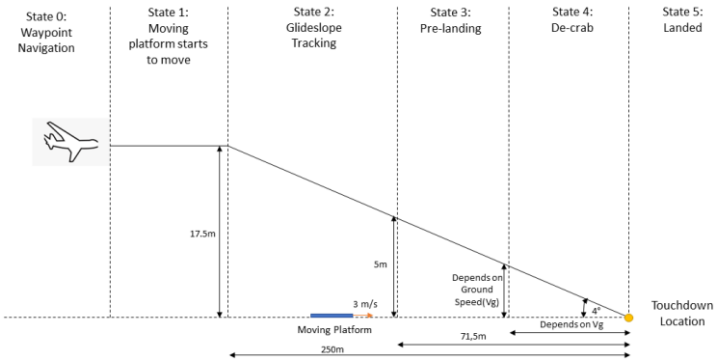

Fig. 4. Diagram showing the different phases of the landing procedure on the UAV's trajectory.

A prediction algorithm is used to get the intended touchdown point of the UAV with the moving platform. The algorithm updates the touchdown point based on the UAV and moving platform states.

During the waypoint navigation stage (state 0) of flight shown in figure 4, the state machine provides constant altitude, airspeed, cross-track, and yaw reference to the FCS. When the aircraft is on final and in line with the runway (state 1 and above in figure 4), the state machine changes these references to allow the aircraft to follow a glideslope, so that it gently decreases its altitude and intersects the moving platform. At state 1 the moving platform begins to move at a constant speed of 3 m/s while being in line with the runway. In the delivery scenario, the moving home base would already be moving however, due to the runway length being a limitation, it is decided to let the platform move when the UAV is close to it. At state 2 the aircraft begins to descend and track the glideslope. At the start of state 3, the aircraft checks whether it has captured the glideslope by comparing its variable values to a threshold. If its values are outside the bound, it will abort the landing attempt and return to the waypoint navigation stage (state 0). This will allow it to go around and try the landing again. If its values are within the bound, then it continues the landing and moves to state 4. At state 4 the aircraft decrabs to align itself with the runway. This is required to prevent the aircraft from going off the moving platform if it were to land on it. The point at which the aircraft starts to decrab in figure 4 depends on the ground speed of the aircraft (V_g). At state 5 the aircraft touches down on the moving platform. For the delivery scenario the UAV would get captured by an arrestor system however this cannot be done due to the time and cost constraints of the research project. Since the moving platform used in this research project is much smaller than the UAV, the aircraft cannot actually land on the platform. It is decided that the "touchdown" will consist of the aircraft intersecting a virtual platform a few meters above the moving platform. The position where the aircraft intersects the virtual platform's altitude will be considered the touchdown point. This minimises the risk of the aircraft becoming unstable on touchdown. After state 5 (touchdown) the aircraft returns to waypoint navigation mode (state 0) and goes around so that it can land on the runway itself. This time however it does physically touchdown on the runway.

2.3 Software in the loop (SITL) implementation

SITL implementation is performed to verify the controllers' performance in a more realistic environment. This is used to increase confidence in the controllers' ability to perform in real life.

SITL is implemented using PX4 Autopilot software, Robot Operating System (ROS), and Gazebo simulator. PX4 runs the controllers, a ROS node runs the MPC algorithm, and Gazebo simulates the physics of the UAV. The advantage of using PX4 is that it is used in commercial applications and is open-sourced. This provides confidence in the software as it has been thoroughly tested in different applications with many safety systems in place in case of component failure. ROS and Gazebo are used in many applications besides UAVs, and this results in a large community being available for troubleshooting issues.

The ground control station software used for both SITL and practical testing is QGroundControl. This software is used to obtain live data of the aircraft as it is flying. Figure 5 shows QGroundControl and the aircraft model simulated in Gazebo.

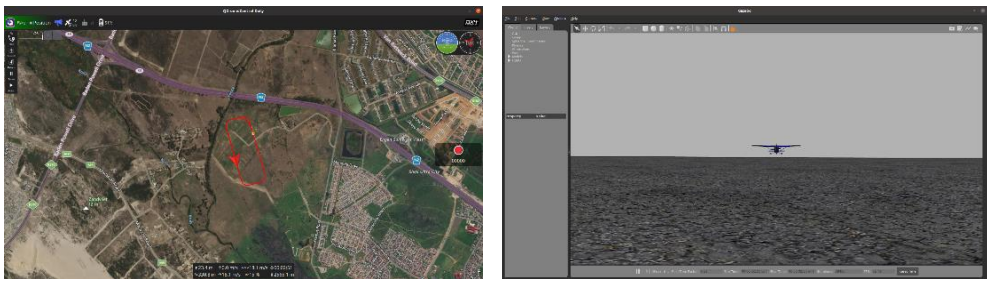

Fig. 5. A screenshot of QGroundControl is shown on the left and the aircraft simulated in Gazebo is shown on the right. In QGroundControl the arrow is the aircraft, and the loop indicates the path that the aircraft has flown.

2.4 Practical testing

Practical flight testing is performed to verify the simulation results in real life. A model remote-control aircraft is used as the UAV for the practical flight tests. The fully assembled aircraft is shown in figure 6.

Fig. 6. Fixed-Wing UAV used for practical flight testing.

The hardware used for the practical implementation of the autopilot system is as follows:

- A Phoenix 0.60 size trainer is used as the airframe of the UAV.
- A Pixhawk 4 is used to run the PX4 autopilot software.
- An Nvidia Jetson Nano runs the ROS node.
- An RTK D-GPS is used to obtain centimetre level position accuracy.
- Various sensors are used to obtain data about the states of the UAV.
- A remote-control car acts as the moving platform on the runway.
- A laptop is used as the ground station.

The moving platform can only be represented by a small RC car, shown in figure 7**,** due to the size of the runway. The small size of the car prevents the UAV from actually landing on it therefore a virtual platform a few meters above the RC car is used as the landing target, as was previously highlighted in the Landing Procedure section.

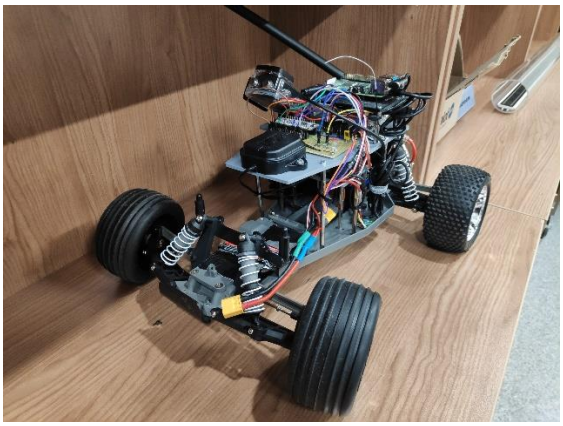

Fig. 7. Moving Platform used for practical flight testing.

The practical tests are performed at the Helderberg Radio Flyers airfield due to its open space and runway for take-off and landing. A safety pilot is present to take off the aircraft from the runway and also take over in case of autopilot failure.

3 Simulation testing results

Controller step responses are generated from the control design, Simulink block diagram and the SITL implementation. They are compared and are found to be similar to each other, however, there is a slight degradation in the controller performance as the testing environment becomes more realistic, which is expected. The MPC controller does perform better than the classical altitude and airspeed controllers in simulation.

The reason for performing these step responses is so that the controller's performance can be verified individually to ensure that the controllers will perform well together during the actual landing. To generate the step responses, the aircraft is kept at level flight flying in one direction with the airspeed and altitude remaining constant. The appropriate controller's reference is then changed, and the output is recorded.

After all the controllers' step responses are deemed acceptable, the full landing system is tested in simulation in both Simulink and SITL environments. The landing system is first tested for a static landing, which is landing on a runway. Thereafter the moving platform landing is tested. The static landing is done first because it tests if the UAV can land on a stationary target which is much easier to do than a moving target. Performing the simulation with the moving platform test gives increased confidence that the system will be able to land the UAV on the platform in real life.

The full landing system test in simulation for static and moving platform landing consists of the aircraft first completing a circuit and when it is aligned with the target it will then descend and intersect the target. The exact procedure followed is discussed in the Landing Procedure section.

The trajectory of the UAV for a static landing using the classical airspeed and altitude controllers is shown in figure 8.

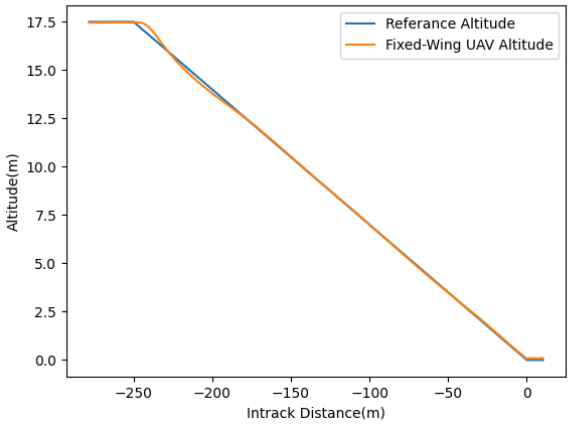

Fig. 8. Classical controllers SITL landing trajectory for a static landing.

The classical controllers have a slight oscillation when capturing the glideslope and this does reduce its landing accuracy. The UAV lands with an in-track error of 23 cm and a cross-track error of 85 cm. This is a good result even with the oscillation due to the UAV's high speed when landing occurs. The maximum allowable in-track and cross-track errors are 1.5 m for both, and the UAV is well within these limits.

The trajectory of the UAV for a static landing using the MPC controller is shown in figure 9.

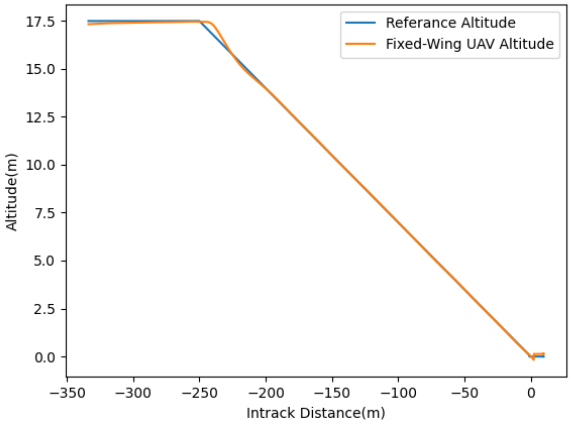

Fig. 9. MPC controller SITL landing trajectory for a static landing.

The MPC controller tracks the glideslope better than the classical controllers as there is less oscillation. This results in better landing accuracy with the UAV landing with an in-track error of 11 cm and a cross-track error of 20 cm. This is an excellent result and is well within the acceptable error limits.

The simulated trajectories of the aircraft and the moving platform for a moving platform landing are shown in figure 10. The MPC controller is active during this landing.

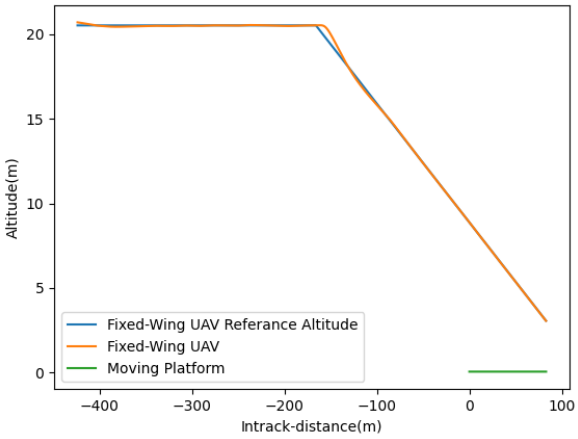

Fig. 10. MPC controller SITL landing trajectory for a moving platform landing. The altitude (in metres) is shown on the y-axis while the in-track distance (in meters) is shown on the x-axis.

The virtual platform altitude is set to 3 m above the moving platform, and this is where the aircraft aims to land. For the moving platform landing using the MPC, the UAV lands with an in-track error of 22 cm and a cross-track error of 9 cm. This in-track error is slightly worse than the static landing, which is expected as the predicted landing location slightly changes continuously as the UAV gets closer to the platform. Since the maximum allowed error is 1.5 m, these results are acceptable.

The designed controlled system was able to land the UAV on a static and moving platform in simulation and all that remains is testing the system in real life.

4 Practical testing results

The practical flight tests are used to verify the simulation results on the practical vehicle. The practical tests consist of first performing step responses on the individual controllers to examine their performance. Thereafter static runway landing tests are performed with both the classical and MPC controllers. Finally, the moving platform landing test is done for both sets of controllers.

4.1 Controller step responses

The airspeed, climb rate, altitude, roll angle, cross-track, yaw and MPC controllers are practically tested by performing a step response on the appropriate controller while the aircraft is flying around the airfield. The controller steps are activated on long straights where the aircraft is flying at level flight.

The practical results mostly match the simulated results however there are performance penalties such as a slower settling time and oscillation. This is due to the model not being entirely accurate as well as sensor measurement inaccuracies. Even with these penalties, the classical controller and MPC's practical results are still considered acceptable.

4.2 Static runway landing

The static runway landing is performed practically for both the classical controllers and the MPC controller to verify if they both can land the aircraft in real life. This is important to do as it will determine if the model used for the real environment and the assumptions made are sound. The practical static landing follows the same procedure as the moving platform landing highlighted in the Landing Procedure section with the exception being that the touchdown point is a chosen point on the runway.

The UAV trajectory for a static landing using the classical airspeed and altitude controllers is shown in figure 11 below.

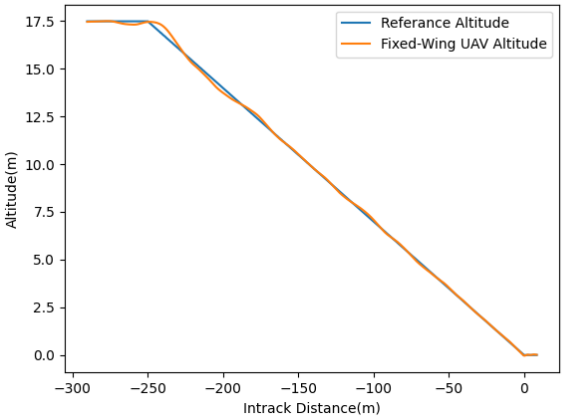

Fig. 11. Plot showing the aircraft's landing trajectory and reference glideslope for a runway landing using the classical airspeed and altitude controllers.

The classical controllers performed very well at landing the aircraft onto the runway as they tracked the glideslope with minimal oscillation. The classical controllers landed the UAV with an in-track error of 42 cm and a cross-track error of 1 cm. This is a great result even though it is slightly worse than the simulation. The real environment has wind and other uncertainties that can affect the result however the classical controllers managed to overcome these differences and produced a fairly accurate landing. This landing is impressive when you consider that the aircraft’s length is almost 2 m and it lands at 16 m/s.

The image in figure 12 shows the fixed-wing UAV landing on the runway at the airfield using the classical controllers.

Fig. 12. Image of the fixed-wing UAV landing on the runway with the classical controllers. A 3m red square(1.5m from the centre point) is made around the expected landing location of the UAV. The image shows the UAV just before touchdown.

The image shows that the fixed-wing UAV lands within 1.5 m of the expected landing point which is great as it is within the red square.

A practical static landing test using the MPC controller was performed however it was found that the MPC behaved too aggressively. The MPC was retuned and retested however, for the safety of the aircraft, it was decided to first examine the MPC on a go-around test. This test consisted of the MPC following a ramp reference at a high altitude that mimics the glideslope it would follow for a static landing. The UAV trajectory for this test is shown in figure 13 below.

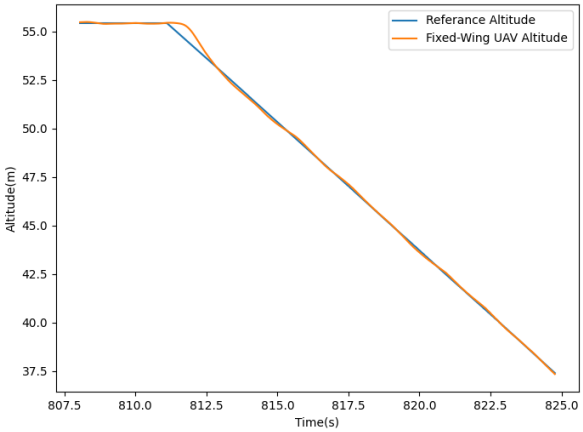

Fig. 13. Plot showing the aircraft's trajectory and altitude reference for a go-around test using the MPC controller.

The MPC can track the altitude ramp reference with minimal oscillation and performs slightly better than the classical controller. Note that the X-axis for the MPC go-around graph is measured in time as the in-track distance is reset during the descent. The in-track distance changes almost constantly with time therefore the MPC go-around graph can be compared with the classical controller static landing graph. As the MPC performed well for the go-around test, it can now be retested for the static landing test which will be done on the next flight day.

4.3 Moving platform landing

The practical moving platform landing consists of using the RC car as the platform on the runway. Unfortunately, during the flight test day, the RTK D-GPS did not work correctly which meant the RC car could not be used. Luckily, a virtual car was available to replace the RC car so that the practical moving platform test could continue. The virtual car simulates the RC car's position on the UAV instead of obtaining it from the GPS. The virtual car's simulation is valid as the RC car is constrained to move in a straight line at a constant speed which makes its position predictable.

The UAV and moving platform trajectories for a moving platform landing are shown in figure 14. The UAV uses the classical airspeed and altitude controllers for this landing.

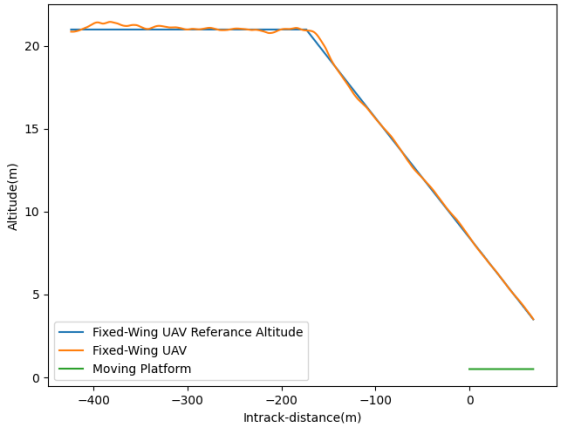

Fig. 14. Plot showing the aircraft and moving platform trajectories for a moving platform landing test using the classical airspeed and altitude controllers.

The virtual platform altitude is also set to 3 m above the moving platform for the practical moving platform landing test. The classical controllers are able to track the glideslope with fairly minimal oscillation and this performance is similar to the runway landing from figure 11 which is desired. The classical controllers land the UAV on the virtual platform with an in-track error of 69 cm and a cross-track error of 4.5 cm which are within the 1.5 m limit. These in-track and cross-track errors are slightly worse than the runway landing values, which is expected as the touchdown point continuously changes making tracking more difficult.

The UAV and moving platform trajectories for a moving platform landing using the MPC controller are shown in figure 15.

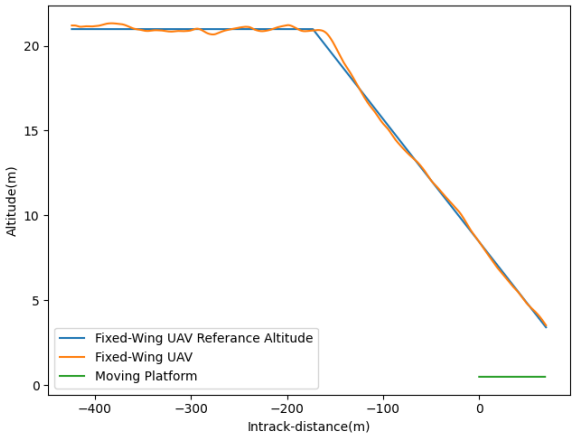

Fig. 15. Plot showing the aircraft and moving platform trajectories for a moving platform landing test using the MPC controller.

The MPC controller has more oscillation in tracking the glideslope compared to the classical controllers. The MPC controller lands the UAV on the platform with an in-track error of 75 cm and a cross-track error of 14 cm. These errors are significantly higher than the SITL moving platform landing however they are still within the 1.5 m limit for the errors. It is still desired to obtain an improved accuracy result with the MPC controller for the practical moving platform landing. The MPC go-around test from figure 13 had minimal oscillation in tracking the glideslope therefore it is likely that there is an external component influencing the results. The wind on this flight test day was quite high and the UAV could have experienced a gust during the MPC moving platform landing test. This MPC moving platform test is therefore planned to be redone when conditions are more favourable.

The GPS on the RC car is currently being configured to work correctly and then it is planned to use the RC car for the next moving platform landing test in the future.

5 Conclusion

A flight and guidance control system was designed and implemented to land a fixed-wing UAV onto a moving platform. The flight control system combined both classical control and model predictive control. The guidance control system used a guidance algorithm and a state machine to allow the aircraft to circuit the airfield and land on the platform. The state machine and prediction algorithm executed the landing procedure which allowed the UAV to follow a glideslope and intersect the moving platform. In simulation the MPC landed the UAV onto the moving platform with an accuracy of 22 cm which is within the 1.5 m limit. This verifies that at least in simulation the designed system is capable of landing a UAV onto the moving platform. For the practical moving platform landing test, the classical controllers landed the UAV with an accuracy of 69 cm while the MPC controller landed the UAV with an accuracy of 75 cm. These accuracies are within the 1.5 m limit and are therefore acceptable however it is desired to improve these results. All the research contributions mentioned in the introduction section have been met with the exception of the moving platform hardware contribution as it is currently being worked on. The practical moving platform landing tests were done with a virtual car, however, preparations are being done to perform the next practical flight tests with the RC car.

References

1. R. Pavithran *et al* 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **995** 012015 (2020)
2. P. Manapongpun *et al.*, "DroneBox: A Fully Automated Drone System for Surveillance Application," *Offshore Technology Conference Asia*. p. D031S017R003, Mar. 22, 2022, doi: 10.4043/31685-MS. (2022)
3. Tahar, Khairul Nizam & Ahmad, Anuar & Akib, W.A.A.W.M. & Naim, Wan. Aerial mapping using autonomous fixed-wing unmanned aerial vehicle. 164-168. 10.1109/CSPA.2012.6194711. (2012)
4. I. K. Peddle, "Acceleration based manoeuvre flight control system for unmanned aerial vehicles" (2008)
5. G. L. Hugo and J. A. A. Engelbrecht, "Autonomous landing of a fixed-wing aircraft with partial horizontal stabiliser loss," 2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), 2016, pp. 1-7, doi: 10.1109/RoboMech.2016.7813184. (2016)
6. G. J. Goosen, "Automatic Upset Recovery for Small Fixed-Wing UAVs" (2018)
7. A. de Bruin and T. Jones, "Accurate Autonomous Landing of a Fixed-Wing Unmanned Aircraft under Crosswind Conditions," (2017)
8. C. T. Le Roux, "Autonomous landing of a fixed-wing unmanned aerial vehicle onto a moving platform," (2016)
9. A. Mohammadi, Y. Feng, C. Zhang, S. Baek, and S. Rawashdeh, "Autonomous landing of a UAV on a moving platform using model predictive control," Drones, vol. 2, no. 4, pp. 1–15, 2018, doi: 10.3390/drones2040034. (2018)
10. C. A. Amadi and W. J. Smit, "Design and Implementation of Model Predictive Control on Pixhawk Flight Controller," (2018)
11. E. Kayacan, M. A. Khanesar, J. Rubio-Hervas, and M. Reyhanoglu, "Learning Control of Fixed-Wing Unmanned Aerial Vehicles Using Fuzzy Neural Networks," Int. J. Aerosp. Eng., vol. 2017, 2017, doi: 10.1155/2017/5402809. (2017)
12. B. Brukarczyk, D. Nowak, P. Kot, T. Rogalski, and P. Rzucidło, "Fixed wing aircraft automatic landing with the use of a dedicated ground sign system," Aerospace, vol. 8, no. 6, 2021, doi: 10.3390/aerospace8060167. (2021)
13. L. Wang, Model Predictive Control System Design and Implementation Using MATLAB. Springer, (2009)