Journal of Intelligent & Robotic Systems (2021) 103: 73
https://doi.org/10.1007/s10846-021-01521-z

SHORT PAPER

Control System Architecture for Automatic Recovery of Fixed-Wing Unmanned Aerial Vehicles in a Moving Arrest System

Kristoffer Gryte[1] · **Martin L. Sollie**[1] · **Tor Arne Johansen**[1]

Received: 16 December 2020 / Accepted: 10 October 2021 / Published online: 29 November 2021

Abstract

Automatic recovery is an important step in enabling fully autonomous missions using fixed-wing unmanned aerial vehicles (UAVs) operating from ships or other moving platforms. However, automatic recovery in moving arrest systems is only briefly studied in the research literature, and is not yet an option when using low-cost, commercial off-the-shelf (COTS) autopilots. Acknowledging the reliability and low cost of COTS avionics, this paper adds recovery functionality as a modular extension based on non-intrusive additions to an autopilot with very general assumptions on its interface. This is achieved by line-of-sight guidance, which sends an augmented desired position to the autopilot, to ensure line-following along a virtual runway that guides the UAV into the arrest system. The translation and rotation of this line is determined by the pose of the arrest system, determined using two Global Navigation Satellite System (GNSS) receivers, where one is configured as a Real-Time Kinematic (RTK) base station. The relative position of the UAV and arrest system is also precisely estimated using RTK GNSS. Through extensive field testing, on two different fixed-wing UAVs, the system has shown its performance and reliability; 43 recovery attempts in a stationary net hit 0.01 ± 0.25m to the right and 0.07 ± 0.20m below the target in calm conditions. Further, 15 recoveries in a barge-mounted, ship-towed net hit 0.06 ± 0.53m to the right and 0.98 ± 0.27m below the target in winds up to 4 m/s. The remaining error is largely systematic, caused by communication delays, and could be reduced with more integral effect or through direct compensation.

Keywords Unmanned Aerial Vehicle · Line-of-sight guidance · COTS avionics · Automatic recovery

1 Introduction

Fixed-wing unmanned aerial vehicles (UAVs) are typically superior to similar-sized rotary-wing UAVs using the same energy source when it comes to range, endurance and speed, and is thus the preferred option for many scenarios. While many missions can be flown automatically, possibly interacting with an operator at a ground control station, recovery of fixed-wing UAVs is often a manual task performed by a highly skilled pilot. In addition to the economic benefits of removing the human pilot from the control loop, this also enables operations with a smaller margin of error, as the sensing and control loops of a UAV autopilot are faster and capable of simultaneously monitoring more mission-critical conditions. This enables operations in rougher conditions, such as in strong and gusty winds, and landing in confined and moving locations, such as aboard ships.

Landing of UAVs on a moving platform is often limited by available space. One viable approach to enable recovery on a space-limited, moving platform is to design the operation around a fixed-wing UAV with vertical takeoff and landing (VTOL) capabilities, i.e. a rotary-wing/fixed-wing hybrid [1]. The increased maneuverability and hover capabilities associated with VTOL UAVs make them easier to land, but this comes at a cost of increased drag, mass and complexity, and decreased payload capacity. While fixed-wing VTOL UAVs and rotary-wing only require a flat surface to land on a moving platform, as in [2–4], conventional fixed-wing UAVs, which this publication focuses on, relies on arrest recovery systems to land on a space-limited, moving platform. Arrest recovery systems are herein defined as some mechanical system that seeks to remove the kinetic energy from the fixed-wing UAV and

✉ Kristoffer Gryte
gryte@ieee.org

[1] Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, NTNU, Trondheim, Norway

bring it to a standstill. This enables the design of the UAV to be focused on the main mission, which is usually what adds value for the end user. Arrest recovery systems, can be divided into the following categories:

Net recovery flying into a tensioned, fixed net that absorbs the kinetic energy of the impact either vertically [5–7], horizontally mounted on the roof of a moving car [8, 9], or suspended between two multirotor UAVs [10].

Airbag recovery flying into an inflated cushion [11], from any direction.

Hook recovery attaching to a wire stretched between two points, e.g. horizontally [12], vertically [13] or between two multirotor UAVs [14].

One strategy to simplify recovery of fixed-wing UAVs on a moving platform is to control the platform itself, assisting or fully performing the alignment of the recovery system with the UAV. In both [10] and [14], where multirotor UAVs are used to capture the fixed-wing UAV, the accurate alignment of the recovery system with the flight path of the incoming UAV is performed by the multirotors, while the fixed-wing flies along a predetermined path. Another way to simplify the control for the recovery is to predict the motion of the platform [15] and when conditions are safe for landing [3]. In the ideal case, with perfect prediction, this simplifies the scenario to stationary landing, although in reality the prediction of e.g. ship motion is difficult [4, 16].

Recovery in an arrest system requires two types of navigation functions in the UAV; it has to *self-navigate*, i.e. keep track of its own position, velocity and attitude, while also keeping track of its position relative to the arrest system. While the self-navigation also is critical for the success of the main mission of the UAV, the relative navigation is only relevant for the recovery. Therefore, a large overlap in the hardware requirements for the two systems is ideal, to simplify avionics. UAV avionics typically consists of an inertial measurement unit (IMU) aided by GNSS position measurements, heading information from a magnetometer/compass, altitude information from a barometer or altimeter, and airspeed information from a pitot tube. These sensors are sufficient for the waypoint tracking involved in most missions, but the precision might not be sufficient for a recovery application. The required level of precision is largely governed by the error margins allowed by the geometry of the arrest system, and the dynamics of the moving arrest system compared to the agility of the UAV. Furthermore, as recovery is seen as a safety-critical phase of the operation, it may be required to add additional sensors to improve the robustness and resilience.

Visual navigation is a popular technique for relative navigation [17, 18]. What makes this approach tractable is the possibility to construct a self-contained system that does not rely on external communication or measurements, that delivers relative pose measurements at a high rate, with high precision when close to the arrest system, like e.g [6, 11, 19, 20]. Drawbacks include high processing requirements, risk of false detection, and sensitivity to visual conditions, such as light/weather conditions and distinctiveness of the arrest system relative to its background. The latter can to some extent be mitigated by using infrared (IR) cameras, either using natural landmarks [21, 22] or IR lamps in known locations [23].

The arrest system may also be equipped with position sensors such as GNSS receivers [8] or radio beacons, exemplified by ultra-wideband (UWB) [24–26], where the main advantage is low cost to the user, small footprint, all-weather availability and ease of use. This is especially true for GNSS, which is already part of most autopilot sensor suites. While the positioning accuracy of a single receiver without augmentation is in the order of meters, depending on whether one or more constellations and a single or dual frequency receiver is used [27], two independent receivers operating within a short distance will have atmospheric errors which are mostly common. This means that relative positioning accuracy between two receivers will generally be better than the absolute accuracy if both receivers track the same satellites and apply the same atmospheric corrections, although this also depends on the multipath situation for each receiver. Space-based augmentation systems (SBAS) can improve the positioning accuracy by transmitting corrections for satellite position errors, clock errors and atmospheric effects to the user from geostationary satellites [27]. Centimeter-level precision, between the UAV and a base station, can be achieved with real-time kinematic (RTK) GNSS [27], a technology that over the last decade has become available to the civilian market at a low cost. GNSS measurements are inherently absolute, so if the arrest system is moving, it must be fitted with a second receiver to obtain the relative position. This also calls for radio communication between the UAV and arrest system. Advantages with UWB, compared to GNSS, include robustness to interference, resistance to multipath [28], as well as high temporal resolution allowing for centimeter level precision of range measurements [29], but the ranges are typically only in the hundreds of meters. By positioning the UWB beacons to move with the arrest system, they can provide a relative navigation solution, possibly at the cost of weaker measurement geometry due to the limited size of the arrest system, leading to lower precision [24]. Drawbacks associated with GNSS include susceptibility to radio frequency interference (RFI), both natural RFI, such as ionospheric scintillations [30] and multipath, and intentional RFI, such as jamming [31] and spoofing [32].

Other relative navigation off-the-shelf options include the laser-based *Object Position and Tracking System* (OPATS) [33], GPS- and radar-based *Dual-Thread Automatic Takeoff and Landing System* (DT-ATLS) [34, 35], as well as the integrated navigation and control solution *UAV Common Automatic Recovery System* (UCARS) [36] for ship landing. Although these systems are well proven, they are also proprietary commercial systems with unknown algorithms and with little flexibility to make customizations.

To approach the arrest system from the correct direction, its (relative) heading must be found, from one of the seven ways to estimate heading [37]. The simplest is through a magnetometer/compass, which unfortunately is highly susceptible to magnetic anomalies and electromagnetic interference (EMI) [37]. With a camera, the relative heading angle can be found [6]. Another solution is to equip the arrest system with multiple position sensors to find the orientation of the baseline between them, see e.g. [38] or [39] that reports 0.27 ° precision for a baseline of about 0.5,m using GNSS. Depending on the dynamics of the arrest system and the precision requirements, a combination with inertial sensors may be required to provide smoother estimates at a higher rate.

Another important part of the recovery system is guidance and control. For an overview of different control algorithms for fixed-wing UAVs, see [40], and [41] for path following guidance algorithms. The navigation setup tends to dictate requirements for the guidance and control system, where visual servoing methods favor pure-pursuit guidance [6, 11], while with the relative navigation between the UAV and arrest system in an absolute frame, the guidance law can be chosen arbitrarily.

This paper seeks to investigate how precisely, accurately and reliably a fixed-wing UAV can land in a moving arrest system, using a control system architecture building modularly and non-intrusively on low-cost commercial off-the-shelf (COTS) hardware (HW) and software (SW). The main contribution is the design and implementation of the landing system SW and HW, in addition to extensive experimental testing. To our knowledge, there are no openly available publications that give a complete description of such a system, so we believe that the description and systematic evaluation herein is a solid foundation for an industrial implementation and future academic research. To be of any operational value, the system must be accurate and reliable enough that the operators have confidence in it, which boils down to repeatability and operability across a wide range of environmental conditions. It should also be precise enough to allow recovery in arrest systems that are of a manageable size. The presented system is based on COTS autopilot HW and SW as they are generally well tested, thus reliable, providing airworthiness and reducing the needs for implementation, possibly at the cost of performance, flexibility and licensing issues. However, they might not provide all the necessary features. Even though some commercially available autopilots are capable of automatic landing in fixed locations, such as [42], this does not suffice for a moving arrest system. Instead of adding the arrest system recovery functionality in a specific autopilot SW, by making possibly error-inducing changes to a working system, this work seeks to build on the existing interfaces of common autopilots by basing the extension on the very general assumption that all autopilots provide an estimate of its position, velocity and attitude, based on internal sensors and an external position measurement, while also providing a means to command the UAV to fly to a specific location. In this work the autopilot is provided with position measurements from RTK-GNSS, due to its simplicity and high precision, but it could in principle come from any position sensor with sufficient accuracy. In addition to non-intrusiveness, these assumptions also make the system modular so that it can be adapted to a wide range of autopilots and fixed-wing UAV configuration, through only a few tuning variables, although this work focus on the open source ArduPlane autopilot software, and is motivated by its current limitations. This is achieved by a line-of-sight (LOS) guidance controller, that ensures line-following of a virtual runway into the moving arrest system, by sending position commands to the autopilot.

The paper is split into two main parts. First, Section 2 describes the recovery system architecture for a general, idealized scenario. This includes the plan generation (Section 2.1), navigation (Section 2.2), motion prediction (Section 2.3), guidance and control (Section 2.4) and operator interface (Section 2.5) subcomponents. The second major part is Section 3, where the general architecture is adapted to a specific arrest system and a specific autopilot, where the implementation in a real-time system is discussed. The implemented system is experimentally validated in two experiments in Sections 4.1 and 4.2, first for a stationary arrest system and then for a moving arrest system mounted on a floating barge towed by a ship. Lastly, in Section 5.1 we discuss the results and mention possible improvements before drawing the conclusion in Section 5.2.

2 Recovery System Architecture

The control system architecture is presented in this section, by considering each of the functional components that are needed to recover a fixed-wing UAV in a moving arrest system. The system creates the plan, seen in Fig. 1, from parameters set by the operator. As the arrest system is moving, so is the latter part of the plan, which is translated and rotated such that it lines up with the predicted pose of the arrest system at the time of impact.

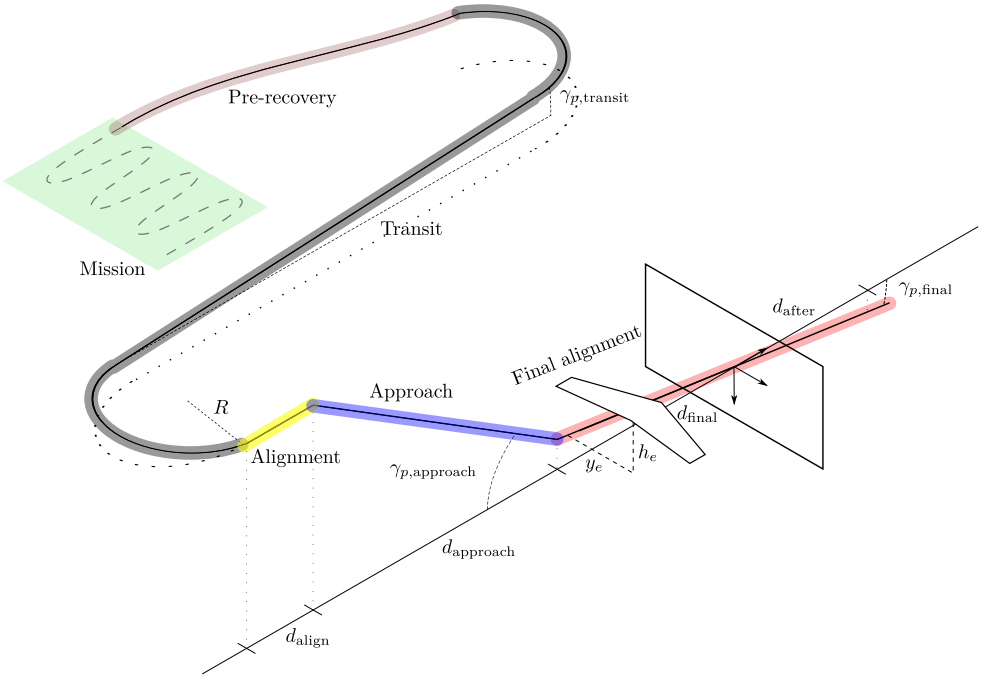

Fig. 1 The geometry of the arrest system recovery plan, illustrated with a net

This pose is predicted from precision navigation, that includes compensation for predicted arrest system motion to maximize the chances of impacting near its center. To allow straight-line path-following, while being limited to only sending a position reference to the autopilot, the system is augmented with a line-of-sight guidance that finds an appropriate carrot-point reference that will give the desired behavior. Before impact with the arrest system, the motor is turned off, to avoid damage and severe entanglement.

2.1 Plan Generation

A plan can be generated with different objectives in mind. The different objectives are usually a combination of minimizing risk and reducing the effect the recovery has on the rest of the mission, being the primary objective. The presented solution seeks to minimize risk, primarily in three ways. First by maximizing the predictability of the UAV motion, by having the final stages be straight line segments. Secondly, the risk is minimized by delaying the reduction in height as much as possible, to increase the probability of succesful abort in case of an emergency. Lastly, the risk is minimized by reducing the relative speed between the UAV and the arrest system before impact, to not jeopardize the structural integrity of the UAV. However, this speed reduction must not come at the cost of a too low airspeed, to avoid stall, and to maintain enough speed to penetrate wind gusts and shear, as well as overcoming any forces needed to be captured by the arrest system. Based on these strategical decisions, the recovery is divided into the following phases, illustrated in Fig. 1.

Pre-recovery This is when the UAV has finished its mission, and is initiating recovery by flying to the start of the transit phase along an arbitrary path.

Transit The path toward alignment with the arrest system ends in a 2D Dubins path [43], to bring the UAV to the start of the descent with a correct course angle and altitude. This is an interconnection of circular arcs and straight lines, which, under the assumption of a maximum curvature, is shown to be the shortest path between two poses in 2D, thus minimizing the effect the recovery has on the rest of the mission. The altitude is simply a linear descent at a prescribed angle, $\gamma_{p,\text{transit}}$, again delaying the descent as long as possible. If the desired altitude at the end of the transit phase is unreachable at this descent rate, the final circular arc of the Dubins path is extended into a spiral to shed the excess altitude.

Alignment When exiting the Dubins path (transit phase), the course should be aligned with the arrest system, but to be sure this course is held for a distance d_{align}, to ensure that the UAV has a stable course.

Approach While maintaining alignment, the UAV descends with a flight path angle γ_{approach}, for a distance d_{approach}.

Final alignment The UAV is now on a virtual runway, starting a distance d_{final} before the arrest system. This runway is guiding the UAV into the arrest system center by continuously aligning itself with the orientation and position of the moving arrest system and the desired landing flight

path angle γ_{final}. Speed is reduced to lower the impact, and the engine is turned off to avoid damage to the propeller or arrest system.

Catch After a successful recovery, with stopped/disarmed motors.

2.2 Navigation

The success of the recovery hinges on knowledge of where the UAV is, relative to the arrest system. For the presented approach, the self-navigation is assumed to be handled by the COTS autopilot, typically through a Kalman filter based on inertial navigation, aided by a GNSS receiver for position measurements, a compass/magnetometer for heading measurements, a barometer/altimeter for altitude measurements, and a pitot tube for airspeed measurements. These are considered as standard components, since they are part of most autopilot sensor suites.

Instead of only using a standalone GNSS receiver, this work uses Real-Time Kinematic (RTK) GNSS, which has been successfully utilized for similar applications [5, 10]. RTK GNSS works by continously sending all raw measurements including carrier phase measurements from a reference receiver, in addition to the reference receiver's own position estimate (as this may be changing over time), to the UAV. The UAV receiver then uses the measurements from both receivers, using a technique based on carrier phase interferometry, to estimate the baseline between them with high accuracy and precision. This works well as long as they are closer than about 20 km apart [44], as the atmospheric signal disturbances have a high degree of spatial correlation such that they are approximately equal for both receivers. It is important to note that the output format of the receiver is the same both when RTK is used and when it is used as a standalone receiver, and where there is a degradation of the RTK capability, e.g. GNSS signal strength drops so carrier phase measurements become unusable, this means that the precision and accuracy of the relative positioning is reduced. RTK GNSS was chosen based on the simplicity of its usage, availability and high precision. Whether the high precision of RTK GNSS is necessary, depends on the size of the UAV relative to the arrest system, but given the availability of low-cost RTK GNSS solutions, it seems tractable. These receivers output position and velocity estimates with high precision and accuracy directly into the autopilot, without the need for any additional computation. If using a widely supported output-data format, such as the NMEA standard, the receiver can easily be replaced, becoming a transparent source of high-precision and high-accuracy estimates to the autopilot.

2.2.1 Relative Navigation Setup

To reap the full potential of GNSS in terms of precision, it is important to make use of RTK processing, providing a precise *relative* position of the UAV and base. Therefore, the arrest system is equipped with one GNSS receiver acting as an RTK base station. For ship-based recovery in open waters, the only option is a moving-base configuration, where only the precision of the relative position is guaranteed. However, for recovery in stationary arrest systems, the base antenna position can be surveyed, allowing for both accurate and precise, global position measurements. Further, the arrest system is equipped with an additional RTK GNSS receiver, which is used to measure the orientation of the arrest system. During recovery, it is important that the position of the UAV and arrest system are reported in the same frame of reference, with the same origin. This also implies that a barometric pressure sensor onboard the UAV cannot be used as the only source of altitude measurements during the final stages of recovery, unless the arrest system is equipped with barometric pressure sensor that is calibrated to the same level as the onboard pressure sensor, pre-flight, as changes in ground level pressure would lead to drift in the altitude estimate during a flight, which ultimately would cause the UAV to aim above or below the physical arrest system target. Another option is to use a source of altitude measurements without long-term drift, such as RTK GNSS, either as a primary altitude sensor or to correct barometer drift over time.

The arrest system is instrumented with two GNSS antennas, with one antenna positioned in $\boldsymbol{p}^n_{\text{left}}$ on the left side as seen from the front, acting as the RTK base for the UAV and the second net antenna, which is placed in position $\boldsymbol{p}^n_{\text{right}}$ on the right side of the arrest system, as illustrated for a recovery net in Fig. 2. The position of the arrest system center in the NED frame $\{n\}$ with its origin at the position of the left antenna, $\boldsymbol{p}^n_{\text{arrest}}$, roll angle ϕ_{arrest} and heading angle ψ_{arrest} are calculated as

$$\psi_{\text{arrest}} = \text{atan2}\left(-b^n_x, b^n_y\right), \tag{1}$$

$$\phi_{\text{arrest}} = \text{atan2}\left(b^n_z, \sqrt{b^{n\,2}_x + b^{n\,2}_y}\right), \tag{2}$$

$$\boldsymbol{p}^n_{\text{arrest}} = \frac{1}{2}\boldsymbol{b}^n_{\text{arrest}} - \boldsymbol{R}^n_b \boldsymbol{p}^b_{\text{offset}}, \tag{3}$$

where $\boldsymbol{b}^n_{\text{arrest}} = \begin{bmatrix} b^n_x & b^n_y & b^n_z \end{bmatrix}^\mathsf{T} = \boldsymbol{p}^n_{\text{right}} - \boldsymbol{p}^n_{\text{left}}$ is the vector from the left antenna to the right antenna, estimated using moving-base RTK with the left antenna used as the base. $\{b\}$ here denotes the body frame of the arrest system, and the vector $\boldsymbol{p}^b_{\text{offset}}$ contains the position of the midpoint between the antennas relative the origin of $\{b\}$, allowing more flexibility in the mounting of the antennas if required,

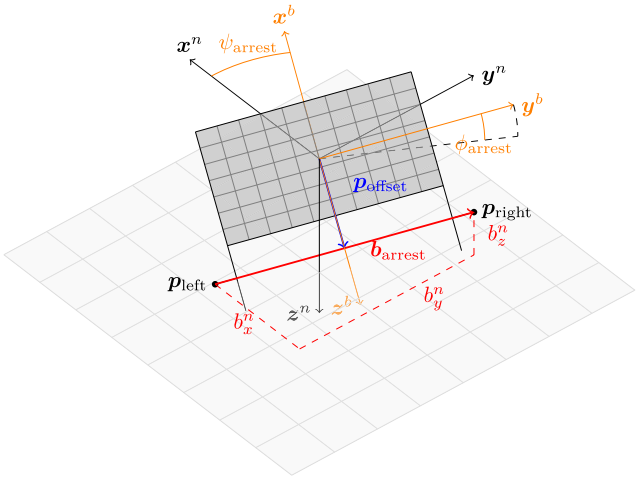

Fig. 2 Illustration of recovery net with antenna positions. The $\{n\}$ origin is here shown coincidental with $\{b\}$ to illustrate the roll and yaw angles

i.e. they can be positioned higher than the center or with different distance to the center on each side. With this antenna configuration, the pitch angle can not be calculated, forcing the approximation $\boldsymbol{R}_b^n \approx \boldsymbol{R}_b^n(\phi_{\text{arrest}}, \theta_{\text{nom}}, \psi_{\text{arrest}})$, where the pitch angle θ_{nom} is a constant nominal value. This encourages a small offset $\boldsymbol{p}_{\text{offset}}^b$ to minimize position errors. For excessive pitch motion, or large offsets in the xz-plane, an alternative would be to also estimate the pitch angle, using an IMU or a third antenna.

2.3 Motion Prediction

In principle, for a successful recovery the UAV only needs to know the relative pose of the arrest system at the time of recovery. However, this can be difficult to predict for moving arrest systems, as the recovery location might not be determined uniquely and with certainty at the start of the recovery plan. Not only is this a chicken-and-egg problem, where the relative position of the arrest system, at the time of recovery, is needed to calculate the time it takes to fly to it (which again is needed to predict the relative position of the arrest system at the time of recovery), but it is also highly dependent on the dynamics of the arrest system. The arrest system can either have accurate, actively controlled motion, such as [8–10], which calls for synchronization between the UAV and arrest system, or be passively attached to a moving platform without accurate control, such as a moving ship [4]. Particularly for recovery in smaller arrest systems in space-restricted environments, using less agile UAVs with smaller error margins, good predictions of the arrest system motion will be more important, but the prediction quality naturally depends on how well the arrest system dynamics can be modeled. Given good predictions, the recovery controllers may be less reactive to rapid arrest system motion, thus allowing less agile UAV dynamics.

The proposed system only makes a rough prediction of the position of the arrest system at the time of impact, given its current velocity. The relative position of the UAV and arrest system is calculated as

$$\Delta \boldsymbol{p}^n = \boldsymbol{p}_{\text{arrest}}^n + \Delta \boldsymbol{p}_{\text{arrest}}^n - \boldsymbol{p}_{\text{UAV}}^n, \tag{4}$$

where $\boldsymbol{p}_{\text{arrest}}^n$ and $\boldsymbol{p}_{\text{UAV}}^n$ are current position estimates for the arrest system and UAV, respectively, and $\Delta \boldsymbol{p}_{\text{arrest}}^n$ is the predicted arrest system movement during the time t_{impact} remaining until impact, using an initial guess. Given this relative position, an estimate of the time until impact is found using the current UAV velocity and a simplifying assumption of straight line flight to the arrest system,

$$t_{\text{Impact}} = \frac{\|\Delta \boldsymbol{p}^n\|_2}{\|\boldsymbol{v}_{\text{UAV}}\|_2}, \tag{5}$$

which is then used to improve the prediction of arrest system movement,

$$\Delta \boldsymbol{p}_{\text{arrest}}^n = \boldsymbol{v}_{\text{arrest}} t_{\text{Impact}}. \tag{6}$$

Then, Eq. 4 is used to compute a new prediction of the relative position at the time of impact, and the process is iterated until sufficiently converged. $\Delta \boldsymbol{p}_{\text{arrest}}^n$ is initialized as 0, which corresponds to a stationary arrest system, but using the final result from the previous time step at the next time step can reduce the number of iterations needed for convergence.

In the above, no prediction or filtering of the attitude and heave motion of the arrest system is performed, as the time horizon for reliable wave-induced motion prediction for ships may be in the order of a few seconds, due to the stochastic nature of waves [16]. Furthermore, the motions of the arrest system are assumed small compared to the agility of the fixed-wing UAV.

2.4 Guidance and Control

To ensure that the UAV follows the final alignment stage of the recovery plan in a manner that is easy to predict by the operator, line-following guidance [41], such as line-of-sight (LOS), is applied in the approach and final alignment stages. LOS guidance [45] mimics an experienced navigator, by aiming to intercept the desired path a time-varying lookahead distance Δ ahead of the current position, see Fig. 3. The UAV, with position $\boldsymbol{p}_{\text{UAV}}^n = [x, y, z]^\intercal$ in the NED frame, follows the line segment that starts in waypoint $\boldsymbol{x}_k^n = [x_k, y_k, z_k]^\intercal$ and ends in waypoint $\boldsymbol{x}_{k+1}^n = [x_{k+1}, y_{k+1}, z_{k+1}]^\intercal$, illustrated in 2D in Fig. 3, while the different segments are illustrated in Fig. 1. As the virtual runway is moving with the position $\boldsymbol{p}_{\text{arrest}}^n$ and orientation

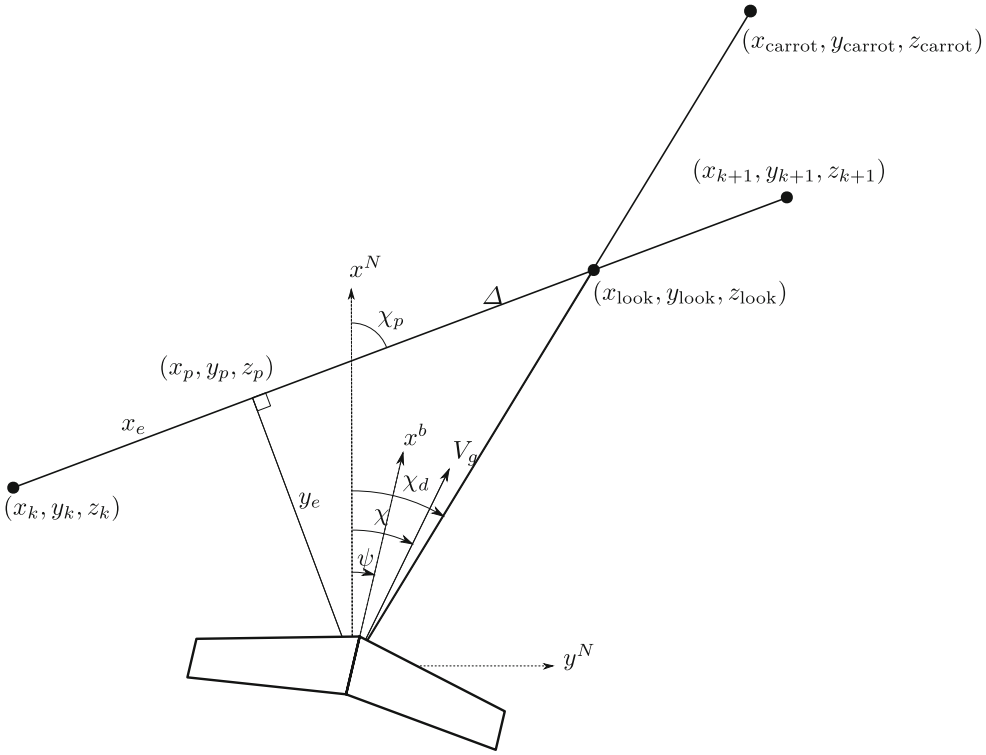

Fig. 3 Geometry of the lateral line-of-sight guidance

ψ_{arrest}, the waypoints that define the start and end of its line segment are also moving. The lateral LOS guidance law

$$\chi_{\text{LOS}} = \text{atan}\left(\frac{-y_e}{\Delta} + K_i \int \left[\text{atan}\left(\frac{-y_e}{\Delta}\right) - \chi\right] \text{d}t\right), \quad (7)$$

$$\chi_d = \chi_{\text{LOS}} + \chi_p, \quad (8)$$

where Δ is the lateral lookahead distance, where $y_e = -\sin(\chi_p)(x-x_k)+\cos(\chi_p)(y-y_k)$ is the lateral cross-track error, and where $\chi_p = \tan\left(\frac{y_{k+1}-y_k}{x_{k+1}-x_k}\right)$ is the course angle of the line segment [45]. This guidance law is extended from [46, 47] to also include integral effect. Although the guidance law is formulated using the course angle, which inherently accounts for wind effects, integral effect is still needed to overcome stationary cross-track errors as a result of e.g. uncompensated misalignment of the navigation system with respect to the airframe. Specifically, there might be a small, uncompensated angular difference in how the navigation system is mounted compared to what roll and pitch angles correspond to trimmed level flight. This has the effect that to fly level, the autopilot should command a nonzero roll angle, which only can be achieved with a zero cross-track error if the integral term in Eq. 7 is nonzero. In Eq. 7, the integral term accounts for the error between the actual course χ and the integral-free desired LOS-angle.

To ensure modularity of the system, and applicability to a wide variety of autopilot interfaces, the guidance controller set-point should be formulated as a desired position. For the lateral axis, this is achieved by considering the desired course χ_d as the direction of the vector from the UAV to the lookahead point $\boldsymbol{p}^n_{\text{look}}$, of length $\sqrt{y_e^2 + z_e^2 + \Delta^2}$, see Figs. 3 and 4, it is clear that the desired lateral position $\boldsymbol{p}^n_{\text{h,look}} = [x_{\text{h,look}}, y_{\text{h,look}}, z_{\text{h,look}}]^\mathsf{T}$ is found from the UAV position by

$$\boldsymbol{p}^n_{h,\text{look}} = \boldsymbol{p}^n_{\text{UAV}} + \boldsymbol{R}_z(\chi_d)\boldsymbol{R}_y(\gamma_d)\boldsymbol{R}_z(\chi_{\text{LOS}})\boldsymbol{R}_y(\gamma_{\text{LOS}}) \times \begin{bmatrix} \sqrt{y_e^2 + z_e^2 + \Delta^2} \\ 0 \\ 0 \end{bmatrix}, \quad (9)$$

where $\boldsymbol{R}_*$ is the rotation matrix representing rotation about the $*$-axis, and where γ_d, z_e and γ_p are defined in Eqs. 12, 10 and 13. By making similar geometric considerations in the longitudinal plane, an analogous longitudinal LOS guidance law can be formulated as [48]

$$\gamma_d = \gamma_{\text{LOS}} + \gamma_p = \text{atan}\left(\frac{-z_e}{\Delta_{\text{v}}}\right) + \gamma_p, \quad (10)$$

where the longitudinal cross-track error z_e and the flight path angle of the line segment γ_p are given by

$$z_e = \cos(\chi_p)\sin(\gamma_p)(x-x_k) + \sin(\chi_p)\sin(\gamma_p)(y-y_k) + \cos(\gamma_p)(z-z_k) \quad (11)$$

$$= \sin(\gamma_p)\sqrt{(x-x_k)^2 + (y-y_k)^2} + \cos(\gamma_p)(z-z_k), \quad (12)$$

$$\gamma_p = \text{atan2}\left(z_{k+1} - z_k, \sqrt{(x_{k+1}-x_k)^2 + (y_{k+1}-y_k)^2}\right), \quad (13)$$

and where Δ_{v} is the longitudinal lookahead distance. To ensure modularity, the desired flight path angle is

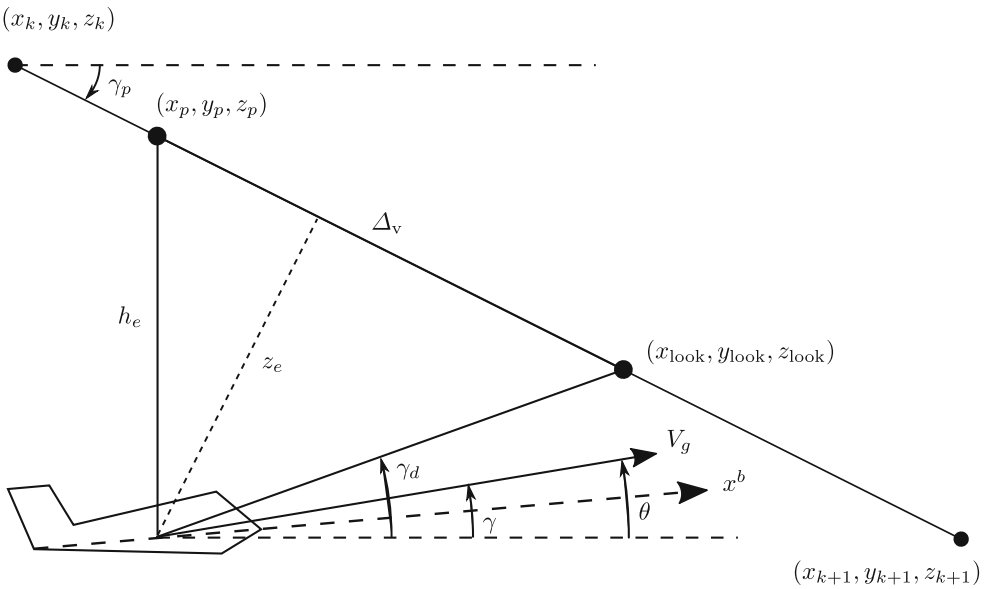

Fig. 4 Geometry of the longitudinal line-of-sight guidance

translated into a desired vertical position using the same LOS principles behind (10). The UAV still aims at a point a distance Δ_v ahead of the projection $\boldsymbol{p}^n_{\mathrm{p}} = \left[x_{\mathrm{p}}, y_{\mathrm{p}}, z_{\mathrm{p}}\right]^{\mathsf{T}}$, but considers the projection point to be on the the vector $\boldsymbol{l}^n = \boldsymbol{x}^n_{k+1} - \boldsymbol{x}^n_k$, which represents the line segment that the UAV is tracking, directly above or below the UAV $\boldsymbol{p}^n_{\mathrm{UAV}}$. Now, the height error h_e is vertical, as illustrated in Fig. 4, in contrast to the longitudinal cross-track error from Eq. 10, which is orthogonal to $\boldsymbol{l}^n$.

From the vertical component of the projection point, $z_{\mathrm{p}} = \frac{\boldsymbol{p}^n_{\mathrm{h,UAV}} \cdot \boldsymbol{l}^n_{\mathrm{h}}}{\boldsymbol{l}^n_{\mathrm{h}} \cdot \boldsymbol{l}^n_{\mathrm{h}}} l^n_{\mathrm{v}}$, where $\cdot$ represents the dot product, and where subscripts $_{\mathrm{h}}$ and $_{\mathrm{v}}$ indicate the horizontal and vertical components, respectively, the vertical lookahead position is computed as

$$z_{\mathrm{v,look}} = z_{\mathrm{p}} + \frac{\Delta_{\mathrm{v}}}{\|\boldsymbol{l}^n\|} l^n_{\mathrm{v}} \tag{14}$$

Similarly to Eq. 7, to account for possible steady-state vertical errors, the height component of Eq. 14 is extended with an integral term

$$\bar{z}_{\mathrm{v,look}} = z_{\mathrm{v,look}} + K_{v,i} \int h_e \mathrm{d}t, \tag{15}$$

where $h_e = z_p - z_{\mathrm{UAV}}$.

To make the guidance performance similar both in downwind and leewind, both the lateral and longitudinal lookahead distances should be functions of the ground speed, i.e.

$$\Delta = V_g \Delta_t \tag{16}$$

$$\Delta_{\mathrm{v}} = V_g \Delta_{\mathrm{v},t} \tag{17}$$

$$K_{v,i} = \frac{\bar{K}_{v,i}}{V_g}, \tag{18}$$

where Δ_t, $\Delta_{\mathrm{v},t}$ are constant, tunable lookahead-time parameters, and where V_g is the estimated ground speed. The longitudinal lookahead time, $\Delta_{\mathrm{v},t}$, can be considered as a compensation for the response of the UAV, including communication delays and time constants in lower-level controllers and reference filters.

The guidance laws (8) and (10) can be implemented through different interfaces to the autopilot. As a consequence of the modular design goals, and under the assumption that all autopilots provide an interface to receive position references, the presented solution simply send the aggregate of the lateral and longitudinal lookahead points, $\boldsymbol{p}^n_{\mathrm{look}} = \left[x_{\mathrm{h,look}}, y_{\mathrm{h,look}}, \bar{z}_{\mathrm{v,look}}, \right]^{\mathsf{T}}$, to the autopilot. However, if the autopilot provides an interface to receive e.g. desired course angle, desired flight path angle and desired airspeed, then χ_d and γ_d from the Eqs. 8 and 10 can be used directly. Furthermore, if an interface that accepts desired roll angle, desired pitch angle and desired throttle, these values can be calculated on the basis of χ_d, γ_d and airspeed error [48, 49]. The lower-level control, regardless of the interface, is assumed provided by the autopilot.

2.4.1 Recovery Prediction and Detection

To avoid damage or entanglement in the arrest system, the motor should in some cases be stopped before it hits the arrest system. For a fixed-wing UAV in puller configuration, the propeller is the first thing that hits the arrest system, which forces the motor stop to be triggered by distance to the arrest system, not by impact detection. This implies a small risk of missing the arrest system while also deactivating the motor, which is mitigated by a starting a watchdog timer. If no impact is detected briefly after the deactivation of the motor, the recovery is deemed unsuccessful and the motor is re-activated, if possible. For an UAV using an internal combustion engine

without a starter motor, reactivation in-air would not be possible and an alternative would be to set the engine to idle before recovery, although this could lead to propeller entanglement. Upon detection of impact the motor is disarmed. This impact is detected based on the longitudinal acceleration of the UAV, which is typically $5 - 10$g during impact.

2.5 Operator Interface

Generally, an increased level of autonomy decreases the requirements to the user interface for the operator. To allow for automatic recovery, the operator interface should let the operator initiate the recovery, while also enable performance monitoring and possibly intervention. This requires radio communication, such that the UAV and arrest system can report their states, and a user interface that displays the essence of this information, including cross-track errors, as a performance metric of the guidance controllers, desired values for the control, motion of the arrest system, the status of any automatic abort monitors, described in Section 2.5.1, as well as the ability to abort the landing.

There are many COTS UAV flight management graphical user interfaces (GUI) available that are used during normal UAV operations, and it is an advantage if the same interface is used in the recovery, as long as the recovery-specific requirements are met.

2.5.1 Aborted Recovery Framework

If the operator initiates abort, an emergency plan is executed. This is a simple dynamic plan, designed by the operator, that consists of a series of waypoints and a loiter, positioned relative to the current position of the arrest system. Thus, initiation of the emergency plan should bring the UAV to a loiter in a safe location, regardless of how the arrest system has moved.

In addition to being triggered by the operator, different abort triggers, that monitor different situations automatically, are implemented to relieve the burden on the operator. Examples of such situations, that make recovery impossible or highly risky, are

Loss of radio communication or loss of arrest system pose measurement making it impossible for the UAV to know the pose of the arrest system.

Severe weather conditions such as strong and/or unpredictable wind, increase the risk involved with recovery. A coarse wind estimate, e.g. [50], is typically monitored by the UAV autopilot, or can be implemented separately.

Poor recovery performance The ultimate objective is to hit the arrest system, so the system predicts if this is not achievable. This is monitored by the UAV by considering its cross-track errors.

Large relative speed from e.g. a strong tail wind can lead to high impact that jeopardizes the structural integrity of the UAV. This is monitored by the UAV, by comparing its own ground speed by that of the arrest system.

Missed catch If the UAV passes the arrest system without registering a catch, its state is undefined, so the emergency plan is started.

However, not all situations allow for an abort [5]. Therefore, the abort framework also acknowledges the UAVs state *severity* level. This level is increased by another set of triggers e.g. if the UAV is too close to the arrest system to make a successful emergency maneuver, or if the fuel or battery is running so low that a go-around is impossible. An elevated state severity level causes the UAV to over-ride aborts, and continue the recovery regadless of some risks.

3 Implementation

To evaluate the arrest recovery control system architecture, the system was implemented for net recovery on a ship. A net was chosen since it occupies little space on a ship deck, which is the landing site of primary interest. Ship decks tend to be crammed, and recovery nets can be removed when not in use. Nets can also be held off the side of the ship by a crane, further reducing the space requirements and risk to the ship. The following subsections describe how the generic arrest recovery control system architecture from Section 2 is implemented for the specific scenario of net landing, and what adaptations have been made to accommodate a specific autopilot software and hardware.

3.1 Net Hardware and Software

The components of the net instrumentation are pictured in Fig. 5 and illustrated in the upper left part of Fig. 6. Both the base and the rover GNSS receivers are U-blox ZED-F9P, which are multi-constellation (configured to use the four global systems GPS, Galileo, GLONASS and BeiDou), multi-frequency receivers with built-in Real-time-kinematic (RTK) processing. The first UART connection of each receiver is configured to send position and velocity estimates to a SenTiBoard sensor interface and timing board [51] at 5 Hz rate, while the second UART is used for a RTCM3 correction data stream also with a rate of 5 Hz, needed for RTK. The only difference in configuration

Fig. 5 Picture of the instrumented net case, showing the two GNSS receivers on top of the SenTiBoard and an ethernet switch. In the front is the embedded computer with a custom cape, on top of the 5 GHz radio

between the receivers is that the base outputs RTCM3 data on the second UART, while the rover uses it as an input. The base receiver sends the correction stream to a BeagleBone Black (BBB) single-board computer, which is set up to distribute this on the network using a TCP server, and to the rover receiver of the net over UART. The SenTiBoard sends the received estimates to the BBB over USB for processing. On the BBB the position data is parsed, translated into net center position and heading according to Eqs. 3 and 1. This is implemented as a task in the *DUNE Unified Navigation Environment* robotic middleware framework of the LSTS Toolchain [52], while the resulting net position and heading, are distributed over the network in terms of Inter-Module Communication (IMC) [53] protocol over UDP.

3.2 UAV Hardware and Software

The UAVs used in the experiments use ArduPlane 3.9.9 [42], running on a Pixhawk autopilot hardware in the X8 UAV, and on a Pixhawk 2.1 for the Dolphine UAV. The rest of the landing-specific payload is common to all the experiments, and is illustrated in the upper right of Fig. 6. In both situations, the autopilot is connected to a Ublox ZED-F9P GNSS receiver, that is used to aid its INS, and to both an ethernet switch and an Odroid XU4 single board computer over UART, to send and receive Mavlink telemetry data. The GNSS receiver receives the correction data from the net base receiver, through the network, via the Odroid, into the receivers second UART port. The GNSS receiver onboard the UAV essentially has the same configuration as the net rover receiver, but outputs a few other messages required by the autopilot.

In order to maintain consistent altitude estimates which are comparable between the UAV and the net, the UAV cannot rely on the internal barometer alone, which is the default behaviour in ArduPilot. The barometer is calibrated

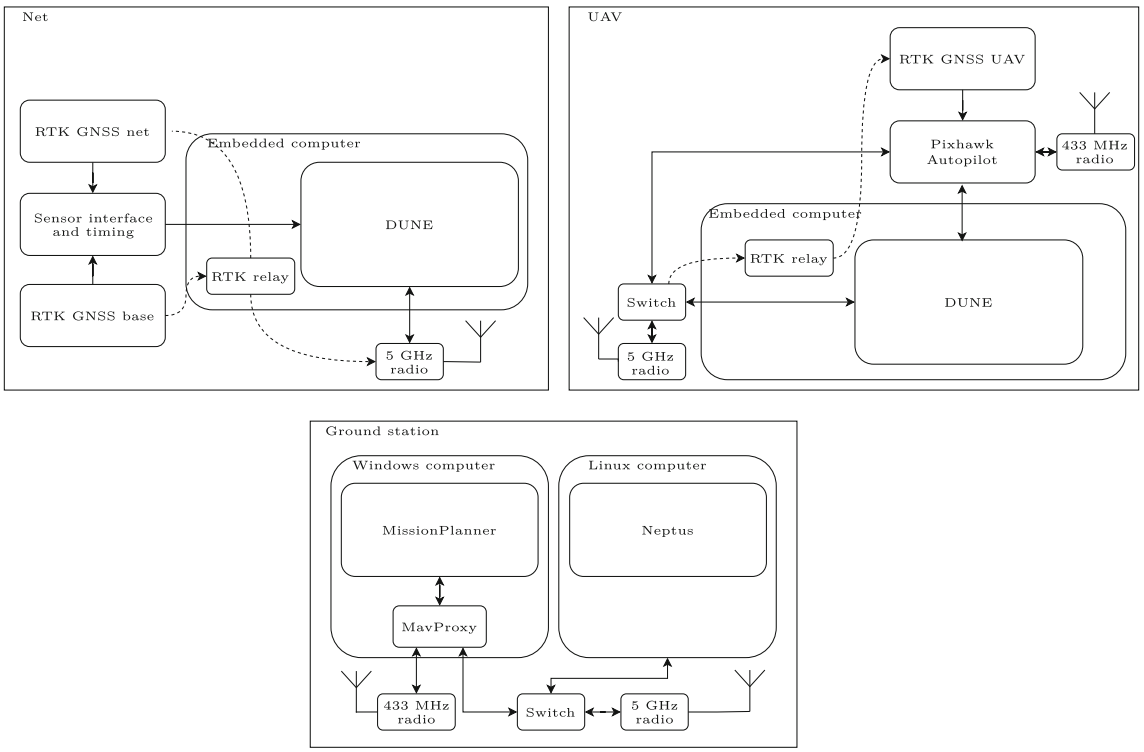

Fig. 6 The different subcomponents of the UAV, the ground station and the instrumented net, that are relevant for the recovery

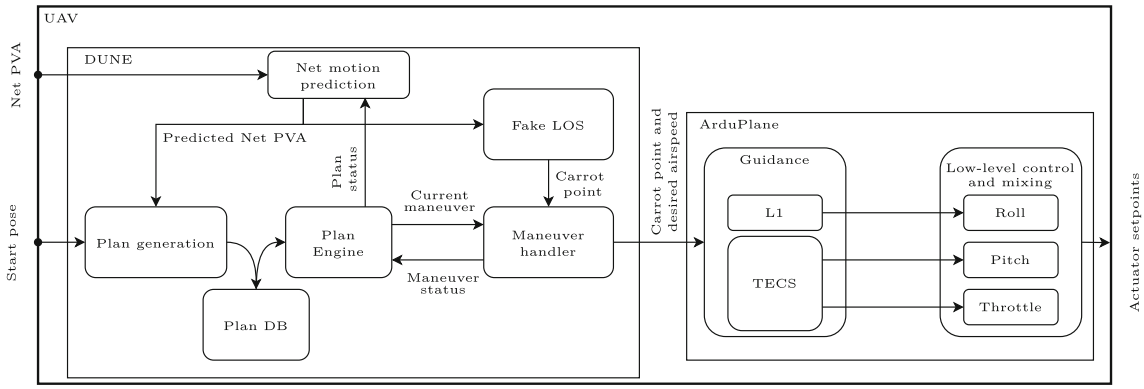

Fig. 7 Block diagram of the UAV control architecture. Each block within DUNE roughly corresponds to a separate task, with IMC messages being passed between them

once at the time of launch, but the ground level pressure can change during a flight, leading to drift in the altitude estimate. In order to avoid this problem, ArduPlane is set to use the height from the RTK GNSS receiver as the primary altitude sensor.

The Odroid also runs DUNE, which includes the net motion prediction, plan generation, guidance, and interface to the autopilot using the MAVLink protocol, in addition to publishing the state of the UAV and net recovery system as IMC messages over UDP, making it available for the Neptus GUI (see Section 3.3).

The recovery plan, as described in Section 2.1, is generated upon request by the operator, according to the parameters, start location, and the location of the recovery system. While the transit phase is static, based on the initial estimate of the landing location[1] and heading, the remainder of the plan is dynamic, and will update as the UAV receives position updates from the net. As the heading of the arrest system has a significant effect on the location of the endpoint of the transit phase, which is static from the time it is generated, it is assumed that the heading of the arrest system does not change significantly during the transit phase. This is reasonable for short, ship-based recoveries, under the assumption that the ship will either be in transit, with a clearly defined heading, actively kept stationary, using dynamic positioning, or slowly drifting. To some extent, an increased uncertainty in the yaw motion of the arrest system can be accounted for by increasing the length of the final alignment stage. For simplicity, the Dubins path, which is computed using the Dubins path library provided in [54, 55], is represented as a sequence of waypoints. This makes the path piecewise linear, and thus not *flyable* according to [56], but by adjusting the parameter that sets the distance between the points, the performance is sufficient for this application. After it has been generated, the plan is stored in the plan database, see Fig. 7, and may be inspected by the operator. Upon initiation of recovery, the plan is loaded into the plan engine, which tracks the progress of the plan, and divides it into separate maneuvers. Each of the static waypoints of the transit phase are represented as a single maneuver, while the dynamic waypoints of the remainder of the plan is its own maneuver. Upon completion of one maneuver, the plan engine starts the next maneuver, by sending it to the maneuver handler. For static waypoints, the desired location is sent directly to the ArduPlane lateral L_1 guidance [57] and longitudinal TECS guidance [58] controllers, operating in GUIDED-mode. In AUTO-mode, the L_1 lateral guidance controller already supports line following. However, it is limited to static lines. So to achieve line following of dynamic lines, like the virtual runway, the ArduPlane guidance controllers are fed a desired location and an airspeed that is continuously updated by the *Fake LOS* block in Fig. 7.

Based on the current position of the net, and the UAVs progression along the dynamic part of the plan, the *Fake LOS* block calculates the desired destination for the UAV

Table 1 Parameters and typical values for small UAVs

Parameter	Typical range
R	50 to 200 m
d_{align}	0 to 100 m
$\gamma_{p,\text{transit}}$	3 to 10°
d_{approach}	100 to 400 m
$\gamma_{p,\text{approach}}$	5 to 20°
d_{final}	100 to 400 m
$\gamma_{p,\text{final}}$	3 to 10°
d_{after}	20 to 150 m
$\boldsymbol{p}_{\text{offset}}$	0 to 10 m in each axis
$\boldsymbol{p}_{\text{start}}$	(Geodetic) start position of the recovery plan

[1] For a slowly moving arrest system, or for short recovery plans, the location of the recovery system could be a reasonable estimate of the landing location.

Fig. 8 The recovery plan generation GUI, with the start position (blue), the arrest system (white), the UAV (green) and the plan inbetween. The small black circle ahead of the UAV corresponds to the carrot point, while the dynamic plan is not in the map view

based on Eqs. 8 and 10. However, when in GUIDED-mode, ArduPlane interprets a desired location as "go here, then loiter". As a consequence, when horizontally close to the desired location, closer than the distance set by the parameter `LOITER_RAD`, the UAV will turn to one side and start a loiter. To avoid this, `LOITER_RAD` is set low, and the horizontal components $x_{\text{h,look}}$ and $y_{\text{h,look}}$ of the desired location $\boldsymbol{p}^n_{\text{look}}$ are extended in the direction χ_d to form a carrot point $\boldsymbol{p}^n_{\text{carrot}}$ for the UAV to follow, when combined with the original desired height $\bar{z}_{\text{v,look}}$, see Fig. 3. Essentially, the *Fake LOS* block transforms the desired position interface into a desired course and height interface.

The carrot point $\boldsymbol{p}^n_{\text{carrot}}$ from the LOS guidance is converted into a WGS84 reference, consisting of latitude, longitude and height, before it is passed to ArduPlane. As ArduPlane does not do line following in this setup, the integral effect in the L_1 guidance controller is disabled. To reduce the need for the integral term in Eqs. 7 and 15, it is advantageous to precisely determine the correct attitude misalignment of the autopilot, and account for this in the `AHRS_TRIM_X` and `AHRS_TRIM_Y` parameters to reduce the cross-track error.

From the desired height, and the desired airspeed, the ArduPlane TECS guidance controller calculates the desired pitch angle and throttle command based on the energy balance.[2] One important parameter is `TECS_SPDWEIGHT`, which weighs the importance of speed tracking against the importance of altitude tracking. During recovery, altitude tracking becomes relatively more important than airspeed tracking, compared to normal flight, so `TECS_SPDWEIGHT` is set to zero. In this configuration, airspeed is controlled by the slow throttle dynamics, while altitude is controlled by the fast elevator dynamics. Another important adjustment to TECS is to set `GLIDE_SLOPE_MIN` to zero. By default, ArduPlane smooths all jumps in altitude that are larger than this value, so by setting it to zero DUNE is given greater authority and less delay. In addition to these parameters, the L_1 and TECS controllers, and the lower level pitch and roll controllers, should also be tuned for a fast response, to compensate for rapid movement of the arrest system.

The recovery detection of Section 2.4.1 is implemented as a separate task in DUNE that subscribes to the distance to the net. Once this value is below a threshold, the ArduPlane parameter `THR_MAX` is set to zero, effectively cutting the electric motor. The threshold is set to be dependent on the speed of the UAV relative the net, to be invariant to wind. When setting this threshold, communication rates from the net to the UAV should be considered, so that

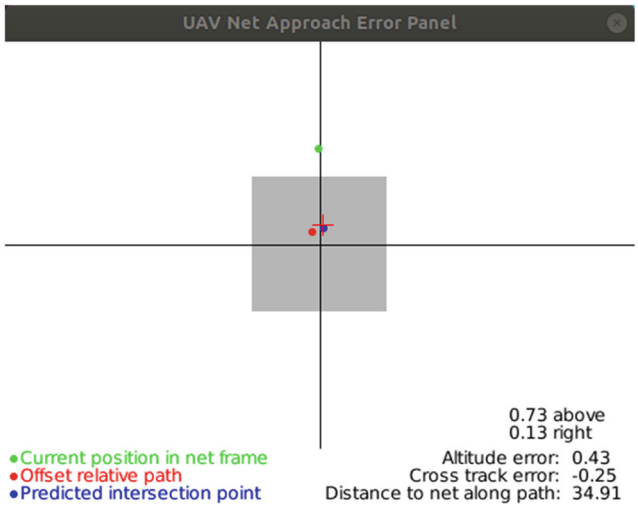

Fig. 9 The GUI recovery profile plugin, illustrating the net (here drawn with a size of 5 by 5 meters), current UAV position relative the path consisting of the errors y_e and z_e (red dot), predicted net impact point (blue dot) and current position in a NED-frame rotated around the z-axis to align with the net heading (green dot). The red cross marks the net impact point of the latest completed recovery

[2] A bug was discovered in the ArduPlane TECS implementation, which lead to poor altitude tracking. It was fixed, and has been included in the ArduPlane 4.0.6 release. See https://github.com/ArduPilot/ardupilot/pull/12822.

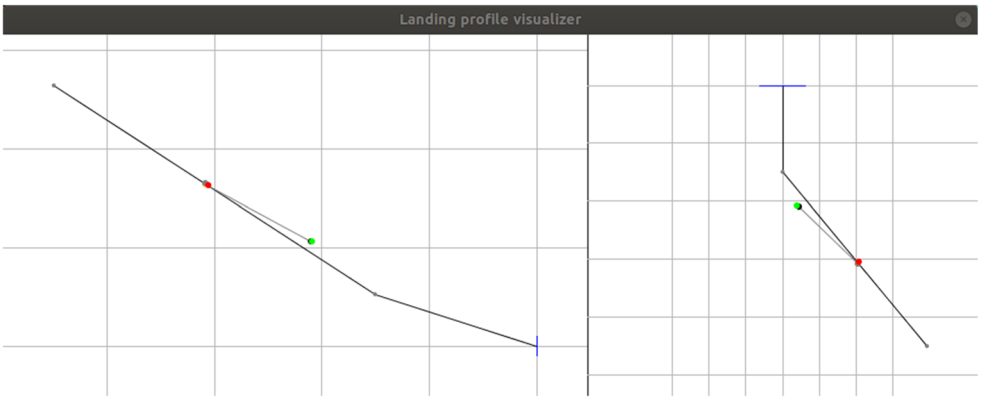

Fig. 10 The arrest system approach path visualization plugin, illustrating the segments of the approach path, as well as the current UAV position $\boldsymbol{p}^n_{\text{UAV}}$ (red dot) and the carrot point $\boldsymbol{p}^n_{\text{carrot}}$ sent to the autopilot (green dot). Because of the integral effect in Eq. 7, the horizontal component of the carrot point does not have to lie on the line segment even if the cross-track error is zero. Similarly, the carrot-point height does not have to lie on the path due to differences in longitudinal and lateral lookahead distances, and lateral carrot point extension. In order to better show the errors in height and in cross-track, these are scaled independently of the horizontal distance towards the net, filling the available window space. The left side shows a vertical profile of the path, with grid marks with 100m spacing in the horizontal and 25m in the vertical direction. The right side shows a horizontal plane view, with grid marks with 4m spacing sideways and 100m in the direction towards the net. The arrest system, exemplified by a net, was rotated after the plan generation, to also show the lateral changes

the motor is stopped before the net impact even with slow communication.

3.3 Ground Station Software

In this prototype implementation, Neptus [59] was chosen as the basis for an arrest system recovery GUI module. Neptus is the ground station component of the open-source LSTS Toolchain, allowing communication with DUNE using the IMC protocol. Specially made plugins provide two main features. First, the operator is able to decide the parameters that are used when the recovery plan is generated. This includes

- selecting the starting point for the recovery plan, or select that it should start from the current UAV position,
- select the desired recovery system, which enforces the UAV to only receive arrest system position messages from the selected recovery system,
- the different distances and angles illustrated in Figs. 1 and 2, with typical values given in Table 1.

This interface also presents the generated plan to the operator, to allow validation, see Fig. 8. Secondly, the GUI contains a display where the operator can monitor the progress of the UAV along the recovery plan, including cross-track errors and a prediction of where the UAV would hit the arrest system given its current position, course and flight path angle, see Figs. 9 and 10. Neptus also includes a button that will abort the recovery attempt.

4 Experimental Validation

Initial verification of the software running in DUNE during development was performed using simulation, with Ardupilot running as software-in-the-loop on a laptop and the net position and attitude coming from a simulated vehicle in DUNE. In simulation the net was tested both as stationary and moving, with playback of logged position and attitude from a seismic support vessel motion reference unit providing realistic movement. Two physical experiments are described in this section. The first demonstrates the

Fig. 11 The fixed-wing UAVs used in the experiments

(a) The NTNU Skywalker X8 flying wing UAV **(b)** The Maritime Robotics Dolphine UAV

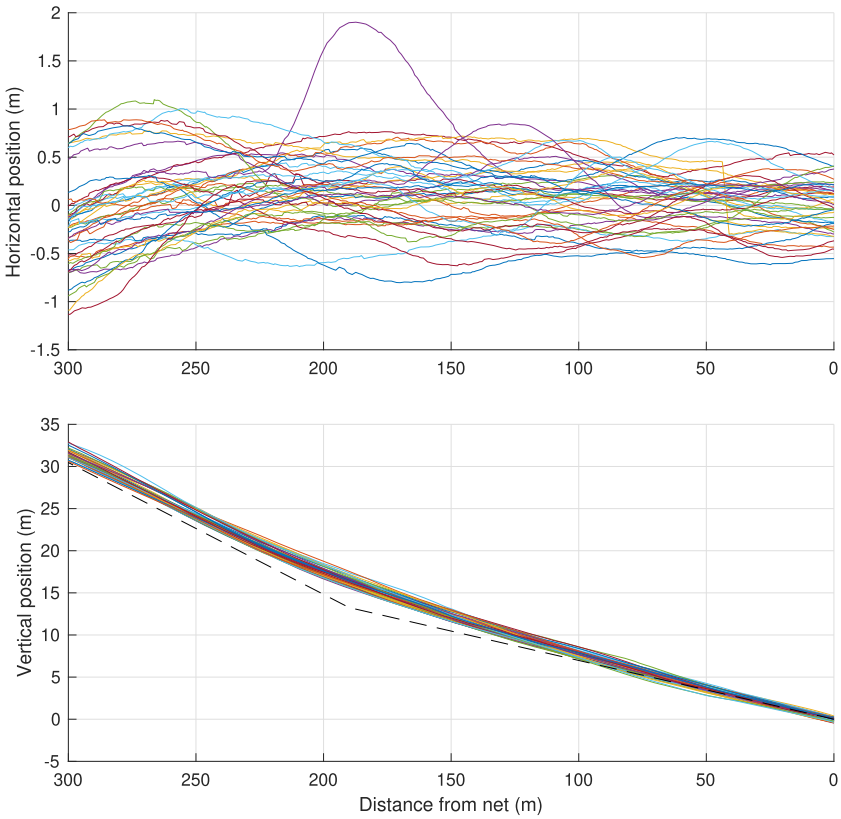

Fig. 12 The position of the UAV in the arrest system frame while approaching the net, for all attempts. The planned descend profile is shown as a dotted line

performance of the system with a stationary net, to isolate the control performance from the motion of the net. The second series of tests involve recovery when the net is placed on a barge, towed behind a moving ship. The two tests also use two different airframes, to demonstrate the flexibility of the system. The first test used a Skywalker X8 styrofoam flying wing UAV with an electric motor and a pusher propeller, see Fig. 11a, that has a wingspan of 2.1 m, a takeoff weight of about 3.5 kg, and cruise speed of 18 m/s. The second test used a Maritime Robotics Dolphine, see Fig. 11b, with an electric motor and a puller propeller, elevator and ailerons. Its wingspan is 1.8 m, the takeoff weight is 9.3 kg, while its cruise speed is 26 m/s.

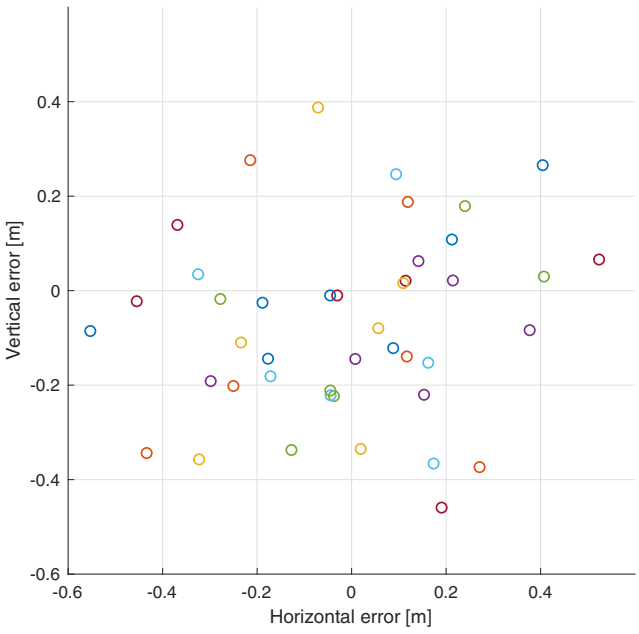

Fig. 13 The position of the UAV when impacting the net, for all 43 attempts in a stationary net, as seen into the net from the approaching UAV

The airframe, actuators and autopilot hardware of the two UAVs are different, while the hardware that the recovery software runs on is simply moved from one UAV to the other, which illustrate the modularity of the system. The same net instrumentation is used in all the experiments.

During both the experiments, the UAV and net are connected to a ground control station over a data link based on Ubiquiti Rocket radios, using the AirMax

Table 2 Net impact performance of 43 recovery maneuvers using a stationary net

	Vertical	Horizontal
Mean	−0.07m	−0.01m
RMS	0.21m	0.25m
Std. dev	0.20m	0.25m

Fig. 14 UAV approaching the barge-mounted net towed by the ship

communication protocol. The ground station, depicted in the lower part of Fig. 6, consists of two computers; one running Ubuntu Linux and another running Windows 10. The Linux computer runs the Neptus ground control station, with the recovery GUI, which is used to visualize information and to interface the payload on both the UAV and the net. The Windows computer runs MissionPlanner, the ArduPlane ground control software, which is used as a backup for the Neptus GUI for communication with the UAV. In addition to communicating with the UAV using Mavlink messages over UDP, the Windows computer also communicates using a 433 MHz telemetry radio, for

Table 3 Results from moving net recoveries. Directions are relative ψ_{arrest}

	Net		Wind		Impact position error	
Recovery #	Speed (m/s)	Dir (°)	Speed (m/s)	Dir (°)	Horizontal (m)	Vertical (m)
1	0.9	−2	<1	–	0.13	−0.87
2	0.8	−2	<1	–	0.03	−1.32
3	0.8	−7	<1	–	0.03	−1.44
4	1.0	6	2	−157	−0.83	−0.79
5	0.9	1	4	10	−1.11	−1.00
6	1.0	−22	4	−28	0.53	−1.13
7	0.6	−25	4	−74	0.05	−1.38
8	0.9	−4	<1	–	0.15	−0.52
9	1.0	−2	<1	–	0.24	−0.69
10	0.9	3	<1	–	0.24	−0.57
11	2.5	−32	1	72	0.25	−0.91
12	1.1	2	3	6	0.08	−1.01
13	1.7	−85	4	−94	0.94	−1.03
14	1.7	−85	4	−99	0.63	−0.88
15	1.7	−33	2	−40	−0.52	−1.12
Average					0.06	−0.98

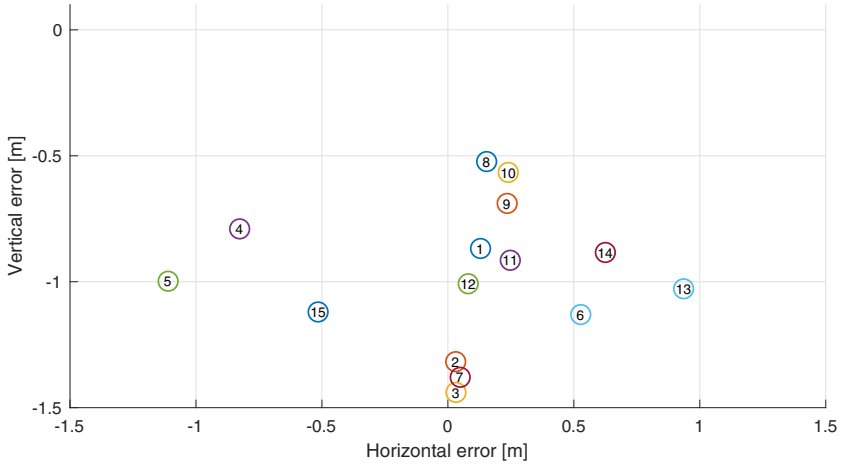

Fig. 15 The position of the UAV when impacting the net, for all 15 attempts in a moving net, as seen into the net from the front

redundancy. The Mavlink messages are then fused using MavProxy.

4.1 Experiments with Stationary Net

A preliminary test of the system with stationary net instrumentation, but without a physical net catching the UAV, was performed with the X8 UAV shown in Fig. 11a. This allowed looping the recovery plan to perform multiple recovery attempts in a single flight with lower risk. 43 recovery maneuvers were performed with a 220,m long approach with 9 ° glideslope and a 190,m long final alignment with a 4 ° glideslope. Winds were calm without significant gusts. The position of the UAV for all attempts are shown in Fig. 12. The top plot showing the sideways movement of the UAV indicates weak oscillating motion which could be caused by too high integral gain or too short lookahead distance in combination with time delays in the communication between the UAV and DUNE.[3]

The position where the UAV would have impacted the net is depicted in Fig. 13, which shows a tendency to hit slightly below the target, but no clear tendency sideways. This is also supported by the average impact position, as reported by performance statistics in Table 2.

4.2 Experiments with Moving Net

To test the arrest recovery system in a more challenging and realistic environment, the net was mounted on a barge, towed behind a ship, depicted in Fig. 14. For these experiments a modified net rig with telescoping poles and fixed antenna mounting points was used, to simplify the mounting on the barge. The barge is about 8,m wide and 5,m long. When towed by the ship, it has a maximum speed of about 2.5 m/s. By adjusting the ropes used in the towing, the angle between the net velocity vector and heading can be adjusted, to test different scenarios.[4]

An overview of the different scenarios used in the 15 recoveries of the Dolphine UAV made in the barge-mounted net is given in Table 3, which lists the horizontal and vertical error in the point of impact, as seen from the UAV flying into the net, for the different recoveries. All the scenarios use an approach of 225,m at a flight path angle of 7 ° downward, while the final alignment is 225,m long, descending with 3 °. To illustrate the environmental conditions for each recovery attempt, the speed of the net and its relative direction, as well as the wind speed and direction relative the net, are also given. In recovery 4, a yaw motion of the net of approximately 0.5 ° per second was initiated approximately 40 seconds before impact, meaning the UAV had to correct its approach course by 20 degrees while approaching the net. The same yaw rate was initiated approximately 20 seconds before impact in recovery 5. In recovery 15, the net was yawed by 5 degrees over a 10 seconds period starting 18 seconds before impact, then yawed back again before impact.

All the points-of-impact are plotted in Fig. 15, where the numbers correspond to the recovery attempt number given in Table 3. The vertical and horizontal trajectories of the UAV as it approaches the net are seen in Fig. 16, where the trajectories have been rotated by ψ_{arrest} around the down-axis to ease the comparison. As the trajectories in these figures are relative to the net, some of the error can be attributed to the movement of the net. This is particularly true for the recovery attempts with large sideways velocity, as only the approach and the final alignment stages utilize the prediction of the ship motion. This is materialized as a larger initial cross-track error to the right of the path, as the barge and net are moving to the left while the start point of the approach phase is fixed when the plan is generated,

[3] To illustrate the development progress and these preliminary results, see https://youtu.be/nMON_udjtiE.

[4] A video showing the setup and some of the recoveries can be found at https://youtu.be/n4XhzcKLgm8.

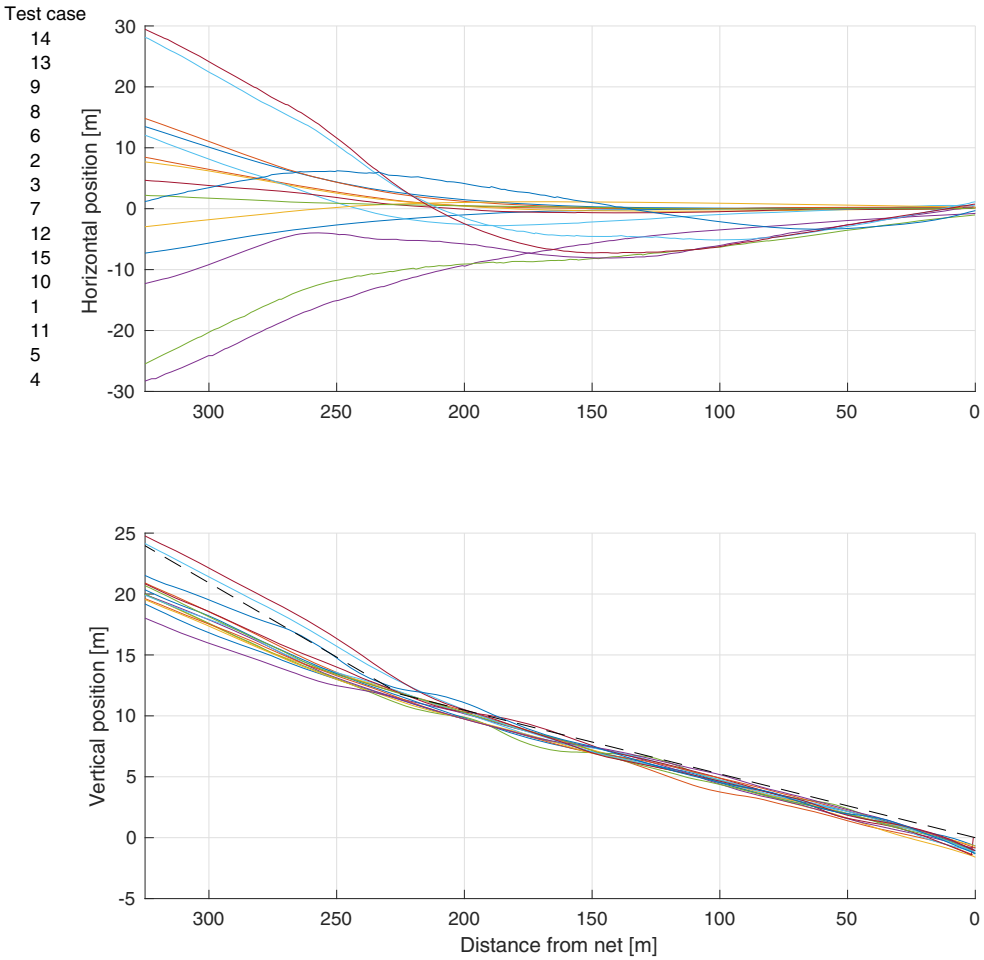

Fig. 16 The position of the UAV in the arrest system frame while approaching the net, for all 15 attempts in a moving net. Positive horizontal position errors are to the right as seen from the UAVs perspective. For the horizontal position plots the test cases are numbered by the horizontal position each case has when entering the figure at 325m distance from the net, with the case having the largest positive position listed first

and is not considering net motion afterwards. The cross-track error in the final alignment phase for the recovery attempts with a large sideways velocity or yaw rate seem large and to the left of the path, as Fig. 16 consider the error relative to the *actual* position of the net, while the UAV aims at the *predicted* position for the time of impact. As the UAV approaches the net, these errors approach zero, as the predicted positions approach the actual positions. However, due to some communication delays that are not accounted for in the net motion prediction, the impact of the UAV ends up to the right of the net center, as it is lagging slightly behind the net. Furthermore, the simple net motion prediction does not account for any rotation of the net, which causes a larger error in scenarios 4, 5 and 15, where the a yawing motion was performed by the barge. In the straight approaches, the communication delays will also cause the UAV to believe that the net is closer than what it actually is, as the net has moved slightly during the communication delay. This could explain some of the height error. Another small contribution to the height error is the power-off of the motor before the impact. From the vertical position it would seem like there is a large error in the approach phase, ending 225,m before the net, as most of the trajectories approach at a much lower angle than the dashed line. This, however, is simply an artifact caused by the forward motion of the net, and the stationary Dubins path. As the net moves forward, the virtual runway moves with it, while the end of the Dubins path remains fixed. This causes the descent of the approach phase to be more gentle. As the height error seems systematic, it could possibly have been reduced by increasing the height integral effect or directly compensated for.

It is noted that the wind in Table 3 is based on the autopilot wind estimate, which is believed to give a reasonable representation of the average wind conditions during the approach and final alignment stages, but is not fast enough to accurately estimate wind gusts. From the results, there is no clear tendency in how this average wind affects the impact error, which is reasonable given the course-based guidance, and it is believed to be dominated by the effects of the communication delays. An example of this is the similar recoveries 7, 13 and 14, where the larger impact error in recoveries 13 and 14 are attributed to the larger sideways velocity of the net.

5 Concluding Remarks

5.1 Discussion

The results show that the presented recovery system is able to reliably recover the UAVs in an arrest system of a size that would fit on many ships, in the tested environmental conditions which had even winds without significant gusts, and small waves. A test on a ship in more challenging conditions would give better understanding of the limits of the system and the net size required to cover all reasonable flight conditions.

Industrialization of the proposed architecture would require changes to increase its robustness to equipment failure. Software or hardware failure of the computer running the recovery software or the serial communication link with the autopilot is not handled in the implemented system, as the autopilot mode used is intended for single "go here, then loiter"-behaviour, not inputs at a fixed rate. A loss of position input is therefore not considered a failure, although it could lead to unfortunate situations in our case, with the UAV starting a small loiter around the position last received. This could be mitigated by extending the autopilot to include a watchdog that triggers a pre-defined action in the event that it stops receiving setpoints.

In a case where the arrest system can move and yaw significantly during the transit phase, the transit phase should be made dynamic. Re-planning of the transit could be done either continuously or if the arrest system movements pass set thresholds.

The communication delays causing increased arrest system impact position errors should be compensated for by improving the timestamping and clock synchronization of the UAV and the net case computer, for example by using UTC timestamps from the GNSS receivers.

5.2 Conclusion

Two test campaigns, with two different UAV platforms, demonstrated the modularity, reliability and performance of the presented arrested recovery system, where the average error norm of 43 recovery attempts in a stationary net was 0.30 ± 0.14m, while 15 recoveries in a moving net had an average error norm of 1.10 ± 0.30m. Although the results are deemed sufficiently accurate for the presented setup, remaining error sources are mostly systematic, like the simplistic motion prediction and communication delays, were discussed, as correction for these will enable recovery of larger UAVs or use of smaller arrest systems.

Acknowledgements This work has been carried out at the NTNU Center for Autonomous Marine Operations and Systems. The authors are grateful for the support from Maritime Robotics, in particular Lars Semb, Carl Erik Stephansen and Morten Einarsve, for piloting the UAVs, constructing the net rig and organizing the moving-net experiments. The authors also thank UAV pilot Pål Kvaløy from NTNU, as well as Einar Nielsen from Petroleum Geo-Services (PGS).

Author Contributions Kristoffer Gryte and Martin L. Sollie have contributed equally to the ideas, theories, implementation and testing presented in this paper. Tor Arne Johansen has contributed with ideas and a theoretical foundation.

Funding Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital). This work was supported by the Research Council of Norway through the Centers of Excellence funding scheme, project number 223254, and the Marlander project, project number 282427. The Marlander project is a collaboration between Maritime Robotics, Equinor, PGS, The Norwegian Clean Seas Association for Operating Companies (NOFO) and NTNU.

Code Availability The code is considered an intellectual property of the Marlander project, and therefore not publicly available.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Pratt, K.S., Murphy, R., Stover, S., Griffin, C.: Conops and autonomy recommendations for vtol small unmanned aerial system based on hurricane katrina operations. J. Field Robot. **26**(8), 636–650 (2009)
2. Rodriguez-Ramos, A., Sampedro, C., Bavle, H., De La Puente, P., Campoy, P.: A deep reinforcement learning strategy for UAV autonomous landing on a moving platform. J. Intell. Robot. Syst. **93**(1-2), 351–366 (2019)
3. Abujoub, S., McPhee, J., Irani, R.A.: Methodologies for landing autonomous aerial vehicles on maritime vessels, vol. 106, p. 10616 (2020). https://www.sciencedirect.com/science/article/pii/S1270963820308518
4. Sanchez-Lopez, J.L., Pestana, J., Saripalli, S., Campoy, P.: An approach toward visual autonomous ship board landing of a vtol uav. J. Intell. Robot. Syst. **74**(1), 113–127 (2014). https://doi.org/10.1007/s10846-013-9926-3
5. Skulstad, R., Syversen, C., Merz, M., Sokolova, N., Fossen, T., Johansen, T.: Autonomous net recovery of fixed-wing UAV with single-frequency carrier-phase differential GNSS. Aerosp. Electron. Syst. Mag. IEEE **30**(5), 18–27 (2015).

http://www.scopus.com/inward/record.url?eid=2-s2.0-84933041204&partnerID=40&md5=51ea460a66b515b034f377f13158fecb
6. Kim, H.J., Kim, M., Lim, H., Park, C., Yoon, S., Lee, D., Choi, H., Oh, G., Park, J., Kim, Y.: Fully autonomous vision-based net-recovery landing system for a fixed-wing UAV. IEEE/ASME Trans. Mechatron. **18**(4), 1320–1333 (2013). http://www.scopus.com/inward/record.url?eid=2-s2.0-84880572499&partnerID=40&md5=40283be167bea79af3a3bc1748a6fdde
7. Yoon, S., Kim, H.J., Kim, Y.: Spiral landing trajectory and pursuit guidance law design for vision-based net-recovery UAV. In: AIAA Guidance, Navigation, and Control Conference, p. 5682 (2009). https://doi.org/10.2514/6.2009-5682
8. Muskardin, T., Balmer, G., Persson, L., Wlach, S., Laiacker, M., Ollero, A., Kondak, K.: A novel landing system to increase payload capacity and operational availability of high altitude long endurance uavs. J. Intell. Robot. Syst. **88**(2), 597–618 (2017). https://doi.org/10.1007/s10846-017-0475-z
9. Persson, L., Muskardin, T., Wahlberg, B.: Cooperative rendezvous of ground vehicle and aerial vehicle using model predictive control. In: 2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017, pp. 2819–2824 (2017)
10. Klausen, K., Fossen, T.I., Johansen, T.A.: Autonomous recovery of a fixed-wing uav using a net suspended by two multirotor uavs. J. Field Robot. **35**(5), 717–731 (2018). https://doi.org/10.1002/rob.21772
11. Huh, S., Shim, D.H.: A vision-based automatic landing method for fixed-wing uavs. J. Intell. Robot. Syst. **57**(1), 217 (2009). https://doi.org/10.1007/s10846-009-9382-2
12. Khantsis, S., Bourmistrova, A.: Uav controller design using evolutionary algorithms. In: Zhang, S., Jarvis, R. (eds.) AI 2005: Advances in Artificial Intelligence, pp. 1025–1030. Springer, Berlin (2005)
13. The Boeing Company: Scaneagle product page. Accessed: 2020-02-30. https://www.boeing.com/defense/autonomous-systems/scaneagle/index.page (2020)
14. Bornebusch, M.F., Johansen, T.A.: Autonomous recovery of a fixed-wing UAV using a line suspended between two multirotor UAVs. IEEE Trans. Aerosp. Electron. Syst., 1–1 (2020)
15. Meng, Y., Wang, W., Han, H., Ban, J.: https://www.sciencedirect.com/science/article/pii/S1270963817315614. Aerosp. Sci. Technol. **85**, 474–480 (2019)
16. Halvorsen, H.S., Øveraas, H., Landstad, O., Smines, V., Fossen, T.I., Johansen, T.A.: Wave motion compensation in dynamic positioning of small autonomous vessels. J. Mar. Sci. Technol. https://doi.org/10.1007/s00773-020-00765-y (2020)
17. Kong, W., Zhou, D., Zhang, D., Zhang, J.: Vision-based autonomous landing system for unmanned aerial vehicle: A survey. In: 2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI), pp. 1–8 (2014)
18. Lu, Y., Xue, Z., Xia, G.-S., Zhang, L.: A survey on vision-based uav navigation. Geo Spat. Inf. Sci. **21**(1), 21–32 (2018). https://doi.org/10.1080/10095020.2017.1420509
19. Thurrowgood, S., Moore, R.J.D., Soccol, D., Knight, M., Srinivasan, M.V., biologically inspired, A.: vision-based guidance system for automatic landing of a fixed-wing aircraft. J. Field Robot. **31**(4), 699–727 (2014). https://doi.org/10.1002/rob.21527
20. Rodin, C.D., Johansen, T.A., Stahl, A.: Skyline based camera attitude estimation using a digital surface model. In: 2018 IEEE 15th international workshop on advanced motion control (AMC), pp. 306–313. IEEE (2018)
21. Khattak, S., Papachristos, C., Alexis, K.: Keyframe-based thermal–inertial odometry. J. Field Robot. https://doi.org/10.1002/rob.21932 (2019)
22. Yakimenko, O.A., Kaminer, I.I., Lentz, W.J., Ghyzel, P.: Unmanned aircraft navigation for shipboard landing using infrared vision. IEEE Trans. Aerosp. Electron. Syst. **38**(4), 1181–1200 (2002)
23. Gui, Y., Guo, P., Zhang, H., Lei, Z., Zhou, X., Du, J., Yu, Q.: Airborne vision-based navigation method for UAV accuracy landing using infrared lamps. J. Intell. Robot. Syst. **72**(2), 197–218 (2013). https://doi.org/10.1007/s10846-013-9819-5
24. Gezici, S., Tian, Z., Giannakis, G.B., Kobayashi, H., Molisch, A.F., Poor, H.V., Sahinoglu, Z.: Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. IEEE Signal Process. Mag. **22**(4), 70–84 (2005)
25. Mahfouz, M.R., Zhang, C., Merkl, B.C., Kuhn, M.J., Fathy, A.E.: Investigation of high-accuracy indoor 3-D positioning using UWB technology. IEEE Trans. Microw. Theory Tech. **56**(6), 1316–1330 (2008)
26. Gryte, K., Hansen, J.M., Johansen, T., Fossen, T.I.: Robust navigation of UAV using inertial sensors aided by UWB and RTK GPS. In: AIAA Guidance, Navigation, and Control Conference. American Institute of Aeronautics and Astronautics (AIAA), pp. 1–16 (2017). https://doi.org/10.2514/6.2017-1035
27. Groves, P.D.: Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems. Norwood, Artech House (2013)
28. Monica, S., Ferrari, G.: An experimental model for UWB distance measurements and its application to localization problems. In: 2014 IEEE international conference on ultra-wideBand (ICUWB), pp. 297—302 (2014)
29. MacGougan, G., O'Keefe, K., Klukas, R.: Tightly-coupled GPS/UWB integration. J. Navig. **63**(01), 1–22 (2010)
30. Yeh, K.C., Liu, C.-H.: Radio wave scintillations in the ionosphere. Proc. IEEE **70**(4), 324–360 (1982)
31. Pinker, A., Smith, C.: Vulnerability of the GPS signal to jamming. GPS Solut. **3**(2), 19–27 (1999). https://doi.org/10.1007/PL00012788
32. Kerns, A.J., Shepard, D.P., Bhatti, J.A., Humphreys, T.E.: Unmanned aircraft capture and control via GPS spoofing. J. Field Robot. **31**(4), 617–636 (2014). https://doi.org/10.1002/rob.21513
33. RUAG: Object position and tracking system (OPATS). Accessed: 14.03.2020. https://ruag.picturepark.com/Go/eemLIBDs/V/7623/1 (2020)
34. Sierra Nevada Corporation: Dual-thread automatic takeoff and landing system (DT-ATLS). https://www.sncorp.com/media/2000/dual-thread-product-sheet2010.pdf. Accessed: 14.03.2020 (2010)
35. Harker, R., Gilligan, J.: Dual thread-automatic takeoff and landing system (dt-atls). In: Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006), pp. 1146–1150 (2001)
36. Sierra Nevada Corporation: Unmanned aerial vehicle common automatic recovery system (UCARS) - version 2. Accessed: 14.03.2020. https://www.sncorp.com/media/1998/ucars-v2product-sheet.pdf (2013)
37. Gade, K.: The seven ways to find heading. J. Navig. **69**(5), 955–970 (2016)
38. Sollie, M.L., Bryne, T.H., Johansen, T.A.: Pose estimation of UAVs based on INS aided by two independent low-cost GNSS receivers. In: 2019 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1425–1435. IEEE (2019)
39. Hemisphere: Vector VR500 Smart Antenna. Accessed: 14.03.2020. https://www.hemispheregnss.com/wp-content/uploads/2019/02/hemispheregnss_vr500_userguide_875-0375-0_a4.pdf (2019)
40. Michailidis, M.G., Rutherford, M.J., Valavanis, K.P.: A survey of controller designs for new generation uavs: The challenge of

uncertain aerodynamic parameters. Int. J. Control Autom. Syst. https://doi.org/10.1007/s12555-018-0489-8 (2019)
41. Sujit, P., Saripalli, S., Sousa, J.B.: Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. IEEE Control. Syst. **34**(1), 42–59 (2014)
42. ArduPilot community: Ardupilot. http://ardupilot.org. Accessed: 30.02.2020 (2009)
43. Dubins, L.E.: 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 (1957)
44. Farrell, J.A.: Aided Navigation: GPS with High Rate Sensors. McGraw Hill, New York (2008)
45. Fossen, T.I.: Handbook of Marine Craft Hydrodynamics and Motion Control. Wiley, Hoboken (2011)
46. Lekkas, A.M., Fossen, T.I.: Minimization of cross-track and along-track errors for path tracking of marine underactuated vehicles. In: Control Conference (ECC), 2014 European, pp. 3004–3010. IEEE (2014)
47. Fossen, T.I., Pettersen, K.Y.: On uniform semiglobal exponential stability (USGES) of proportional line-of-sight guidance laws. Automatica **50**(11), 2912–2917 (2014)
48. You, D.I., Jung, Y.D., Cho, S.W., Shin, H.M., Lee, S.H., Shim, D.H.: A guidance and control law design for precision automatic take-off and landing of fixed-wing UAVs. In: AIAA Guidance, Navigation, and Control Conference, p. 4674 (2012)
49. Gryte, K.: Precision control of fixed-wing UAV and robust navigation in GNSS-denied environments. Ph.D. dissertation, Norwegian University of Science and Technology (NTNU). https://hdl.handle.net/11250/2657113 (2020)
50. Johansen, T.A., Cristofaro, A., Sørensen, K.L., Hansen, J.M., Fossen, T.I.: On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors. In: International Conference on Unmanned Aircraft Systems, Denver, pp. 510–519 (June 2015)
51. Albrektsen, S.M., Johansen, T.A.: User-configurable timing and navigation for uavs. Sensors **18**(8). https://www.mdpi.com/1424-8220/18/8/2468 (2018)
52. Pinto, J., Dias, P.S., Martins, R., Fortuna, J., Marques, E., Sousa, J.: The LSTS toolchain for networked vehicle systems. In: OCEANS-Bergen, pp. 1–9 (2013 MTS/IEEE. IEEE, 2013)
53. Martins, R., Dias, P.S., Marques, E.R.B., Pinto, J., de Sousa, J.B., Pereira, F.L.: IMC: A communication protocol for networked vehicles and sensors. In: OCEANS 2009-EUROPE, pp. 1–6 (2009)
54. Walker, A.: Hard real-time motion planning for autonomous vehicles. Ph.D. dissertation, Swinburne University (2011)
55. Walker, A.: Dubins-Curves GitHub repository. Accessed: 14.03.2020. https://github.com/AndrewWalker/Dubins-Curves (2013)
56. Tsourdos, A., White, B., Shanmugavel, M.: Cooperative Path Planning of Unmanned Aerial Vehicles. Wiley, Hoboken (2011)
57. ArduPilot community: Arduplane L1 guidance. https://github.com/ArduPilot/ardupilot/pull/101, 2013, accessed 03.12.2019
58. ArduPilot community: Arduplane TECS controller. https://github.com/ArduPilot/ardupilot/blob/ArduPlane-3.9.11/libraries/AP_TECS/AP_TECS.cpp. Accessed: 02.12.2020 (2019)
59. Dias, P.S., Fraga, S.L., Gomes, R.M., Goncalves, G.M., Pereira, F.L., Pinto, J., Sousa, J.B.: Neptus - a framework to support multiple vehicle operation. In: Europe Oceans 2005, vol. 2, pp. 963–968. IEEE (2005)

Kristoffer Gryte received his MSc and PhD in Engineering Cybernetics in 2015 and 2020, respectively, both from the Norwegian University of Science and Technology (NTNU). He currently works as a researcher on software for autonomus systems at NTNU. His research interests currently evolve around mobile robots, particularly aerial, including guidance, navigation and control.

Martin L. Sollie received his MSc in Engineering Cybernetics in 2018 from the Norwegian University of Science and Technology (NTNU), where he is currently a PhD candidate. Research interests include guidance, navigation and control for unmanned aerial vehicles.

Tor Arne Johansen received the MSc degree in 1989 and the PhD degree in 1994, both in electrical and computer engineering, from the Norwegian University of Science and Technology, Trondheim, Norway. From 1995 to 1997, he worked at SINTEF as a researcher before he was appointed Associated Professor at the Norwegian University of Science and Technology in Trondheim in 1997 and Professor in 2001. He has published several hundred articles in the areas of control, estimation and optimization with applications in the marine, aerospace, automotive, biomedical and process industries. In 2002 Johansen co-founded the company Marine Cybernetics AS where he was Vice President until 2008. Prof. Johansen received the 2006 Arch T. Colwell Merit Award of the SAE, and is currently a principal researcher within the Center of Excellence on Autonomous Marine Operations and Systems (NTNU-AMOS) and director of the Unmanned Aerial Vehicle Laboratory at NTNU and the SmallSat Laboratory at NTNU. He recently co-founded the spin-off companies Scout Drone Inspection, UBIQ Aerospace and Zeabuz.