In-flight Capture Maneuver of Drones Using Model Predictive Control

Diogo Oliveira[†] and Bruno J. Guerreiro[†,‡]

***Abstract*— This paper presents a model predictive control (MPC) strategy to enable a rotary-wing shuttle drone to cooperatively capture a fixed-wing target drone during flight and place it in the ground safely. Simplified models of the two types of drones are presented, along with simple controller for the each vehicle. The developed strategy defines a simple nonlinear MPC problem that models the relative dynamics between the two drones, enabling the capture maneuver without a preassigned trajectory or rendezvous point. Simulation results demonstrate the capabilities of the proposed cooperative capture strategy, whereas experimental trials with a real shuttle and a simulated target validate the hardware and software integration of the developed system, showing promising performance.**

I. INTRODUCTION

A technology that has been in continuous development is Unmanned Aerial Vehicles (UAVs) or, simply, drones. Aiding in simple and complex tasks in many different fields, UAVs can now have equipment to interact with objects and people, making them able to operate in more realistic environments. These systems are designed with a given application in mind, resulting in different characteristics, such as wing type and overall size, making it impossible for one UAV to efficiently tackle every assignment, which can include surveilance, aerial photography, agriculture, search and rescue missions, among many others [1], [2].

The work presented in this paper is within the scope of project CAPTURE[1], which addresses the challenges of using shuttle drones to launch and capture other vehicles or objects. This paper addresses the integrated planning and control of the maneuvers towards the capture of a fixed-wing cooperative target drone by a rotary-wing shuttle drone solely relying on position data from integrated GPS devices and controlled by onboard algorithms. This will enable an efficient fixed-wing to have VTOL capabilities provided by the shuttle rotary-wing vehicle, as depicted in Figure 1, in alternative to a, possibly less efficient, hybrid VTOL UAV.

Multiple state-of-the-art machine vision-based solutions are available for capturing flying objects using UAVs [3], [4]. These solutions often rely on systems comprised of sensing modules to detect the objects to capture and estimate their poses, high-level task planning modules usually implemented as state machines, and guidance and control units to compute

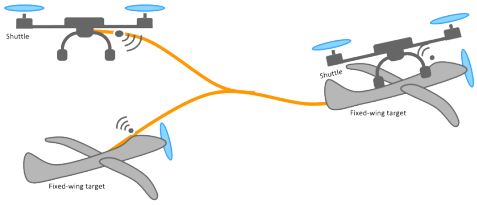


Fig. 1: Rotary-wing shuttle capturing a fixed-wing target.

the action variable values for the actuators of the UAV during its flight. One common control technique used for UAVs in these types of applications is Model Predictive Control (MPC), which has the advantages of predicting the state of a system through a certain horizon window, and of imposing constraints to a control system to model behaviors. Several authors [5], [6] proposed and tested MPC-based controllers that are integrated into Miniature Aerial Vehicles (MAVs) and rely on solving computationally complex problems in real-time. Similar to our problem, [7] explores the challenges of using an MPC to instruct a UAV to approach a moving ship from above and land on its back, whereas [8] showcases simulation results of two UAVs executing a parcel relay maneuver by meeting at an unspecified point.

In the scenario considered in this paper, the target drone performs an independent flight, following a predefined path, whereas the shuttle drone has knowledge about the target's state and desired path. To perform the capture maneuver, the shuttle drone will be hovering in a strategic position, $\boldsymbol{p}_{wait}$, activate its capture controller when the target drone reaches a specific point in its trajectory, $\boldsymbol{p}_{inform}$, which will control the shuttle to a position above the target and perform the capture. During the capture maneuver, the target drone is assumed to have a constant horizontal velocity vector. A virtual fence area is also considered during this phase, A_{stop}, where the drone will revert to the waiting phase if violated. If the capture is successful, the shuttle drone will carry the captured target drone to a dropping point, $\boldsymbol{p}_{target\_collection}$, before the landing, as illustrated in Figure 2.

To address this scenario, the shuttle system will be designed to have a simple prediction of the behavior of both UAVs performing the cooperative capture maneuver, and an optimal control problem involving this joint model is proposed, not only to achieve synchronized flight, but also autonomously decide when is the moment and place to have contact between vehicles. This is achieved through a nonlinear MPC algorithm that includes safety, physical, and operational constraints in its formulation, solved in real-time by the shuttle drone, which performance is firstly analyzed

[†] Diogo Oliveira and Bruno Guerreiro are with the Dep. of Electrical and Computer Eng., CTS-Uninova and LASI, NOVA University Lisbon, 2829-516 Caparica, Portugal. E-mails: diogo.nunes1973@gmail.com, bj.guerreiro@fct.unl.pt

[‡] B. Guerreiro is also with Institute for Systems and Robotics (ISR), LARSyS, Instituto Superior Técnico, Universidade de Lisboa, Portugal.

This work acknowledges the financial support of FCT project CAPTURE (DOI:10.54499/PTDC/EEIAUT/1732/2020).

[1]Website: https://capture.isr.tecnico.ulisboa.pt.

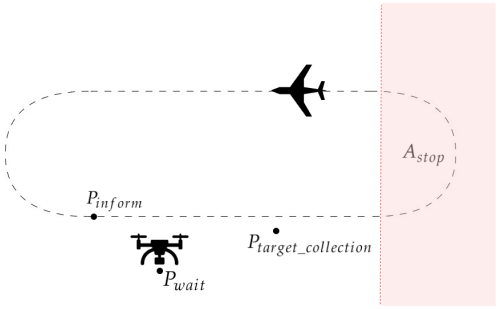


Fig. 2: Illustration of the cooperative capture scenario with the locations of interest identified.

through Matlab simulations. To ensure the robustness of the algorithm to unexpected disturbances during the capture maneuver, a state machine is used to control the shuttle high-level behavior. The developed hardware and software system is then tested under a more realistic scenario using the ROS and Gazebo frameworks to ensure that it's working with the software that is going to be implemented in the real drone. After that, the algorithms will be implemented in real hardware to evaluate their performance under real-world conditions, where the a real shuttle drone and a virtual target drone cooperatively perform the capture maneuver.

The main contributions of this paper include the nonlinear MPC formulation of the capture maneuver of fixed-wing by a rotary-wing drone, the validation of the proposed method in simulation, the hardware implementation of the system and the experimental validation of an online and real-time execution of a capture maneuver using MPC, considering a simulated target vehicle. It is demonstrated that the shuttle drone has the capabilities to process the algorithms in real-time using onboard computation, as well as communicate with the virtual target drone and with a ground station computer. This paper based on [9], whereas other components of the overall system to achieve the real capture are under development within the scope of project CAPTURE, such as the mechanism and vision system [10].

The remainder of this paper is structured as follows. Section II presentes the models used in the control algorithms, while Section III describes the MPC solution used for the capture maneuver and the respective simulation results. Section IV describes the developed system integrated into the real shutle drone and the experimental trials. Finally, Section V presents some concluding remarks and and future work directions.

II. VEHICLE MODELS

This section presents the models of both shuttle and target vehicles considered for the simulation results as well as in the internal models of the MPC problem.

A. Target and Shuttle Models

The model of the target drone is a simple guidance model based on the Dubins airplane model described in [11]. This assumes constant altitude and airspeed, as well as a negligible wind. Let an inertial reference frame I be defined as a local tangent plane by three orthonormal axes $\boldsymbol{x}_I = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ pointing north, $\boldsymbol{y}_I = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$ east, and $\boldsymbol{z}_I = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$ down. Considering a body frame for the target defined as T with origin at the vehicle center of mass and orthonormal axes, $\boldsymbol{x}_T$ pointing forward, $\boldsymbol{z}_T = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$ down, and$\boldsymbol{y}_T$ defined accordingly. As such, the UAV heading angle ψ_t is defined by the angle between the projection of the airspeed vector $\boldsymbol{v}_t$ onto the horizontal plane and the $\boldsymbol{x}_I$ vector, and the flight-path angle γ_t is defined by the angle between the horizontal plane and the airspeed vector.

Consider the position of the vehicle in the inertial frame as $\boldsymbol{p}_t = \begin{bmatrix}p_{tn} & p_{te} & p_{td}\end{bmatrix}^T$, $\boldsymbol{v}_t = \begin{bmatrix}v_{tn} & v_{te} & v_{td}\end{bmatrix}^T$ as the velocity, and the relationship between the heading angle ψ_t and the bank angle ϕ_t in a coordinate turn, we can get the equations that describe the guidance model for a fixed-wing UAV behavior as

$$\dot{\boldsymbol{p}}_t = \boldsymbol{v}_t \tag{1a}$$

$$\dot{\psi}_t = w_{tz} \tag{1b}$$

where g is the Earth gravity and the velocity vector is

$$\boldsymbol{v}_t = V_t \begin{bmatrix}\cos\psi_t \cos\gamma_t \\ \sin\psi_t \cos\gamma_t \\ -\sin\gamma_t\end{bmatrix} \tag{2}$$

where the velocity norm is $V_t = \|\boldsymbol{v}_t\|$, whereas the yaw rate can be defined as $w_{tz} = \frac{g}{V_t}\tan\phi_t$. Considering this guidance model, an attitude inner-loop control strategy can be used to control the thrust of the rotor to achieve the desired forward velocity V_t, and the aerodynamic surfaces to drive the dynamics of the fixed-wing aircraft to the desired behavior in therm of ϕ_t and γ_t.

Regarding the motion of rotary-wing drones, the typical configuration consists of four rotors capable of producing a thrust and the reaction torque, allowing the motion control of the vehicle is 3-D space, using different combinations of rotor angular velocities. A simple guidance model of such a vehicle also assumes an inner-loop for attitude control, for which acceleration vector and a yaw angle references can be used.

Considering the inertial reference frame I, we define the shuttle drone body frame S in a similar fashion to that of the target, for which the position of the origin is $\boldsymbol{p}_s = \begin{bmatrix}p_{sn} & p_{se} & p_{sd}\end{bmatrix}^T$ and $\boldsymbol{v}_s = \begin{bmatrix}v_{sn} & v_{se} & v_{sd}\end{bmatrix}^T$ is the vehicle linear velocity. A simple shuttle drone motion, considering the actuation as a force vector, $\boldsymbol{f}_s \in \mathbb{R}^3$, and the angular velocity around the z-axis, $w_{sz} \in \mathbb{R}$, can be described as

$$\dot{\boldsymbol{p}}_s = \boldsymbol{v}_s \tag{3a}$$

$$\dot{\boldsymbol{v}}_s = \boldsymbol{a}_s \tag{3b}$$

$$\dot{\psi}_s = w_{sz} \tag{3c}$$

where the acceleration vector can be defined as $\boldsymbol{a}_s = g\boldsymbol{z}_I + \frac{1}{m}\boldsymbol{f}_s$ in terms of the Earth gravity and the desired force vector, with m representing the vehicle mass.

B. Target Path Following Controller

A simple control for the target drone guidance model can be defined using a vector-field path-following controller similar to the one presented in [11]. A path to be followed in $\mathbb{R}^3$ by a fixed-wing UAV can be specified as the intersection of two 2-D surfaces, α_1 and α_2, according to the path type, defining

$$V(\boldsymbol{p}_t) = \frac{1}{2}\alpha_1^2(\boldsymbol{p}_t) + \frac{1}{2}\alpha_2^2(\boldsymbol{p}_t) \ . \tag{4}$$

Based on this function, a desired velocity vector can be given by

$$\bar{\boldsymbol{v}}_t' = -K_1\frac{\partial V}{\partial \boldsymbol{p}_t} + K_2\frac{\partial \alpha_1}{\partial \boldsymbol{p}_t} \times \frac{\partial \alpha_2}{\partial \boldsymbol{p}_t} \tag{5}$$

where K_1 and K_2 are tuning matrices. Considering a vector with the same direction of $\bar{\boldsymbol{v}}_t'$ with the norm of V_t defined as $\bar{\boldsymbol{v}}_t = V_t\bar{\boldsymbol{v}}_t'/\left\|\bar{\boldsymbol{v}}_t'\right\|$, we can infer the desired yaw, roll, and flight-path angles as

$$\gamma_t = -\text{sat}_{\bar{\gamma}_t}[\sin^{-1}(\frac{\bar{v}_{td}}{V})] \tag{6a}$$

$$\psi_t^d = \text{atan2}(\bar{v}_{te}, \bar{v}_{tn}) \tag{6b}$$

$$\phi_t = \text{sat}_{\bar{\phi}_t}[k_\phi(\psi_t^d - \psi_t)] \tag{6c}$$

where k_ϕ is a positive constant, $\text{atan2}(b, a)$ is the four quadrant inverse tangent, and $\text{sat}_a[b]$ a saturation function acting on a based on b. The functions α_1 and α_2 can be specified to define simple paths, such as those provided in [11] for straight-line and helical paths, whereas to obtain more complex behaviors this controller is usually integrated in a path manager as the one provided in the next sections.

III. MODEL PREDICTIVE CONTROL FOR CAPTURE

This section describes the MPC controller for the capture maneuver between the two drones, validating the strategy with simulation results.

A. MPC Problem Formulation

Based on the scenario illustrated in Figure 2, the shutle and target drones models considered for the MPC strategy are given, respectively, in (3) and (1). Nonetheless, the target model is further simplified by considering that the velocity vector remains constant, resulting in

$$\dot{\boldsymbol{p}}_t = \boldsymbol{v}_t \tag{7a}$$

$$\dot{\boldsymbol{v}}_t = \boldsymbol{0} \tag{7b}$$

$$\dot{\psi}_t = w_{tz} \ . \tag{7c}$$

This assumption is plausible, as we assume a cooperative scenario where the target drone can try to maintain a straight line trajectory while being captured, and an estimate of this path can be updated in the MPC algorithm at each sampling interval. As such, we can define the overall system state vector as $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{p}_s & \boldsymbol{v}_s & \psi_s & \boldsymbol{p}_t & \boldsymbol{v}_t & \psi_t\end{bmatrix}^T$ and the control action vector as $\boldsymbol{u} = \begin{bmatrix}a_s & w_{zs}\end{bmatrix}^T$, resulting in the model for both vehicles defined as $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}_c(\boldsymbol{x}(t), \boldsymbol{u}(t))$. The model is then discretized along an horizon of N samples by a 4th order Runge-Kutta method, and added to the optimization problem as equality constraints of the form

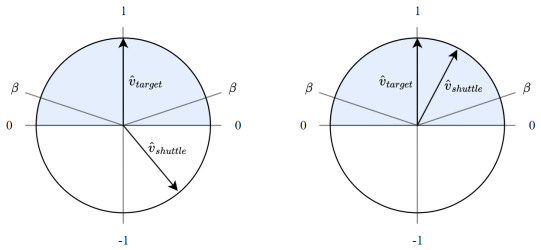


Fig. 3: Velocity direction alignment constraint (12): left, constraint not satisfied; right, constraint satisfied.

$$\boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i) \ . \tag{8}$$

The MPC problem further considers state and actuation saturations constraints, such as

$$-\boldsymbol{v}_{max} \leq \boldsymbol{v}_s \leq \boldsymbol{v}_{max} \tag{9}$$

$$-\boldsymbol{a}_{max} \leq \boldsymbol{a}_s \leq \boldsymbol{a}_{max} \tag{10}$$

whereas the goal to have the shuttle drone approach the target drone from above for the capture maneuver results in the constraint

$$p_{sd} - p_{td} < -0.7 \tag{11}$$

where the dimensions of the grasping mechanism considered for the capture maneuver was taken into account [10]. In addition to the above, a velocity alignment constraint is also imposed to ensure efficiency and prevent the drones from moving in opposite directions, defining

$$\hat{\boldsymbol{v}}_s^T\hat{\boldsymbol{v}}_t \geq \beta \tag{12}$$

representing the dot product between the drones normalized velocity vectors, and $\beta \in \mathbb{R}$ is a positive constant, as illustrated in Figure 3.

The shuttle drone MPC strategy aims to follow the path of the target drone, flying in a straight line, continuously approaching the target's position from above and orienting itself in a way that it can safely perform the capture maneuver. Defining the state and control sequences for a time horizon of N samples, $k = 0, \ldots, N$ for simplicity, as $\boldsymbol{X} = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N\}$ and $\boldsymbol{U} = \{\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}\}$, respectively, an objective function $J(\boldsymbol{X}, \boldsymbol{U})$ for this problem can be defined as

$$J(\boldsymbol{X}, \boldsymbol{U}) = M(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} L(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{13}$$

where the Lagrange and Mayer terms of the cost function are respectively given by

$$L(\boldsymbol{x}, \boldsymbol{u}) = \tilde{\boldsymbol{p}}^T Q_p \tilde{\boldsymbol{p}} + \tilde{\boldsymbol{v}}^T Q_v \tilde{\boldsymbol{v}} + \tilde{\psi}^T Q_\psi \tilde{\psi} + \boldsymbol{u}^T R \boldsymbol{u} \tag{14}$$

$$M(\boldsymbol{x}) = \tilde{\boldsymbol{p}}^T P_p \tilde{\boldsymbol{p}} + \tilde{\boldsymbol{v}}^T P_v \tilde{\boldsymbol{v}} + \tilde{\psi}^T P_\psi \tilde{\psi} \tag{15}$$

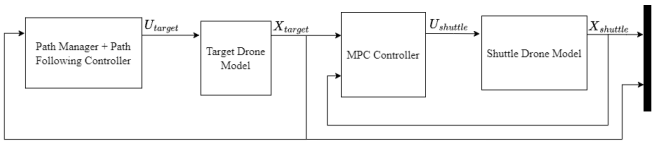


Fig. 4: Simulation model diagram for the complete system.

and

$$\tilde{\boldsymbol{p}}_k = \boldsymbol{p}_{s_k} - \boldsymbol{p}_{t_k} - \bar{\boldsymbol{p}}_k \tag{16}$$

$$\tilde{\boldsymbol{v}}_k = \boldsymbol{v}_{s_k} - \boldsymbol{v}_{t_k} \tag{17}$$

$$\tilde{\psi}_k = \psi_{s_k} - \psi_{t_k} \tag{18}$$

$$\bar{p}_k = \begin{bmatrix} 0 & 0 & dZ_k \end{bmatrix}^T \tag{19}$$

The remaining undefined quantities of the cost function are parameters of the algorithm, namely, the positive definite matrices P_p, Q_p, P_v, Q_v, scalars P_ψ, Q_ψ, and positive semi-definite matrix R. The relative distance variable dZ_k in the position error penalty allows the MPC to dynamically choose the height reference for the shuttle during the flight, defining the function

$$h(\boldsymbol{x}_k) = \begin{cases} h_{safe} & \text{if } \|\boldsymbol{p}_{s_k} - \boldsymbol{p}_{t_k}\|_{xy} \geq d_{near} \\ 0 & \text{if } \|\boldsymbol{p}_{s_k} - \boldsymbol{p}_{t_k}\|_{xy} < d_{near} \end{cases} \tag{20}$$

where $\|\boldsymbol{a}\|_{xy}$ is the norm of the 3D vector projected onto the xy-plane, while h_{safe} and d_{near} are distance parameters.

Considering the above, the optimal control problem solved at each MPC iteration is given by

$$\min_{\boldsymbol{X},\boldsymbol{U}} \quad J(\boldsymbol{X},\boldsymbol{U}) \tag{21a}$$

$$s.t \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \quad , \forall\, k \in \{0,\dots,N-1\} \tag{21b}$$

$$dZ_k = h(\boldsymbol{x}_k) \quad , \forall\, k \in \{0,\dots,N\} \tag{21c}$$

$$(9),(10),(11),(12) \quad , \forall\, k \in \{0,\dots,N-1\}. \tag{21d}$$

B. Matlab Capture Simulations

Using the previously described models and controllers, the Matlab simulation environment represented in Figure 4 was developed in order to get the following results. The simulations shown in this section were run on an ASUS F515 laptop with an Intel® Core™ i5-1135G7 CPU, a NVIDIA GeForce MX330 GPU, and 8 GB of RAM. These simulations were not run on the real drone's hardware, so, when implementing the controller on the drone, performance may vary from these results.

The parameters used for these simulation results are $\boldsymbol{v}_{max} = \begin{bmatrix} 20 & 20 & 20 \end{bmatrix}^T$ m/s, $N = 25$, $d_{near} = 1.5$ m, $h_{safe} = 2$ m, $\beta = 0$, $P_\psi = Q_\psi = 10$, $P_p = Q_p =$ blkdiag$(30, 30, 20)$, $R =$ blkdiag$(2, 2, 1, 10)$, and $P_v = Q_v = 20I_3$.

Figure 5 shows the drones' trajectory results from the simulations. The graphs show the shuttle drone hovering in its initial position, which in this case also represents P_{wait}, and its behavior after the target drone reaches the P_{inform} location represented in the graph as a green circle. The trajectory performed by the target drone is as expected and

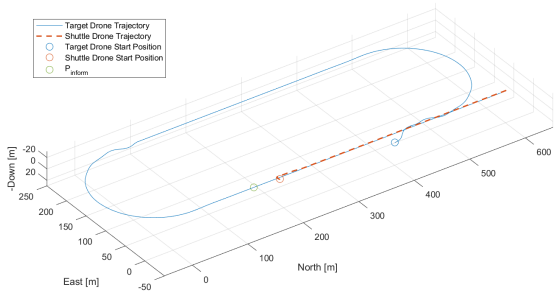


Fig. 5: Cooperative capture simulation results: trajectories.

after the MPC is activated, the shuttle is able to approach the target drone from above and follow the trajectory performed since its velocity is greater than the target's cruise speed.

To better analyze the results, Figure 6 show the temporal evolution of both drones' positions, distances between them and yaw angles. When observing the temporal evolution graphs of the drones' positions in the horizontal and vertical planes simultaneously, it's possible to see the shuttle first adjusting its position until it is above the target with the imposed height safety distance, and then starting to descend until it reaches the target. Considering the grasping mechanism's size, the capture in this simulation is considered a success seeing as the shuttle manages to descend until it reaches a distance of 0.6725 m of the target (enough to activate the capture mechanism on the descent, and not to much to destabilize the target), and during this capture its distance of the target in the horizontal plane is of 0.0328 m and keeps converging to zero as it keeps following the target.

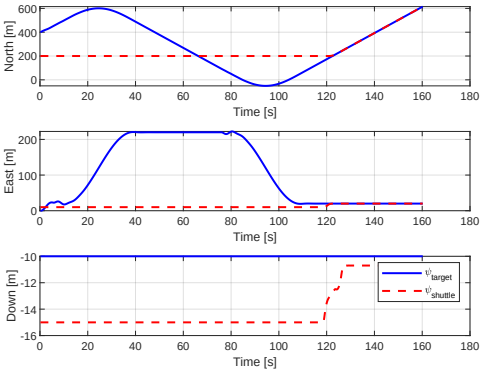
Since the MPC controller is a heavy computation process, it's relevant to analyze the performance of the controller during its execution. During the capture moment simulation, the controller had a relatively low average computation time of 14.34 ms, and a maximum time well below the sampling period ($T_s = 0.2$ s), which makes it plausible to be possible to integrate it into a real-time controller.

IV. ARCHITECTURE AND EXPERIMENTAL RESULTS

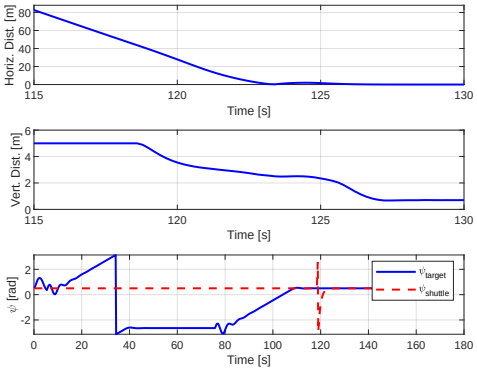
The final system architecture is illustrated in Figure 7, where the main components include the ground station computer, the on-board computer and the PX4 autopilot. Throughout this section, the integral components of the system are going to be described in detail.

A. Architecture and Experimental Setup

On of the main components of this architecture is the PX4 Autopilot, an open-source firmware and ecosystem for the control and development of different types of UAVs. In the



(a) NED Position.



(b) Relative distances and yaw.

Fig. 6: Capture simulation results: pose and relative distances.

system's architecture from Figure 7, the PX4 serves as an interface between the low-level components of the system, and the high-level commands produced from the developed algorithms. The PX4's controller modules take setpoints specified by the pilot or by an offboard algorithm and the state estimates from the estimator modules, and compute the values of the control action variables for the UAV's actuators. The ground control station in our architecture uses QGroundControl which is a GUI application that provides full flight control and mission planning for vehicles with PX4 integration, and communicates with the PX4 flight computer through the MAVLink protocol.

A first step towards experimental trials is to perform simulations using Gazebo, a realistic robotics simulator with ROS integration, a 3D rendering engine, an accurate physics engine, and an extensive set of sensors and vehicle models. This simulator is also used in this paper to replace the fixed-wing target aircraft as only the shuttle drone in considered in the experimental trials in this preliminary validation.

We use a drone control module based on [12] to aid the development of drone applications and the control of both simulated and real vehicles. This module, consisting of a collection of ROS packages, provides the developer with a level of abstraction for interacting with the UAVs

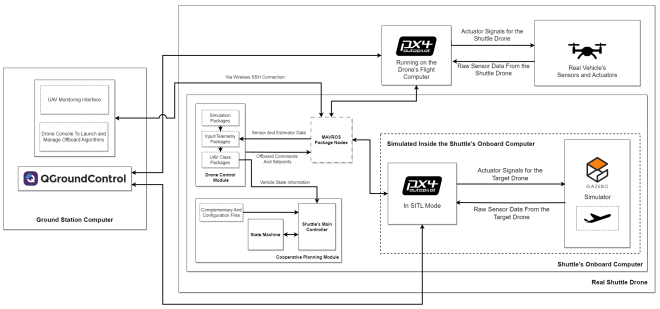


Fig. 7: System's architecture for the field tests using the real vehicle.

from PX4 by creating an interface that provides C++ instances of each UAV with all the required methods for the possible interactions that would be required with the UAV objects. These interactions include the collection of all the information of the drone system, such as sensor readings and state estimations, and some commands of interest for a developer when creating offboard control applications, for example, commands for the arm and takeoff of the drone, and commands to define position setpoints for the PX4 flight controllers.

For the experimental tests shown in this paper, the control of the target drone was controlled by the PX4 autopilot in Mission Mode, which allows the user to pre-plan a mission with the QGroundControl GUI application and upload it to the drone so PX4 can autonomously control it throughout the whole flight. The created mission instructs the target drone to follow a similar trajectory to the one illustrated in Figure 6, and is based on the real world location where the test were conducted. The MPC used for the shuttle's controller was implemented in Matlab with the aid of the CasADi framework. From there, a CasADi C++ library can be automatically generated and exported in Matlab so it can be used with the ROS-based controller by running the CasADi's API in the ROS controller node.

To model the high-level tasks for the shuttle drone's complete mission in the cooperative capture scenario, it was developed a state machine using SMACH, a ROS package for the design and implementation of state machines in ROS applications, that is implemented as a Python ROS node and will be directly communicating with the shuttle's main controller which implements all the logic for the drone's behaviors and establishes the communications with the remaining components. The shuttle's main controller module will communicate with the state machine to get the current state and execute the desired tasks for each state. The state machine is shown in Figure 8. This state machine models the autonomous behavior for the shuttle drone for the whole mission to ensure that integrates failsafe measures to prevent the possibility of reaching a deadlock state because of unexpected behavior.

The shuttle drone's components were selected taking into account the requirements for the tasks that the drone will undergo, ensuring that the shuttle can carry another target

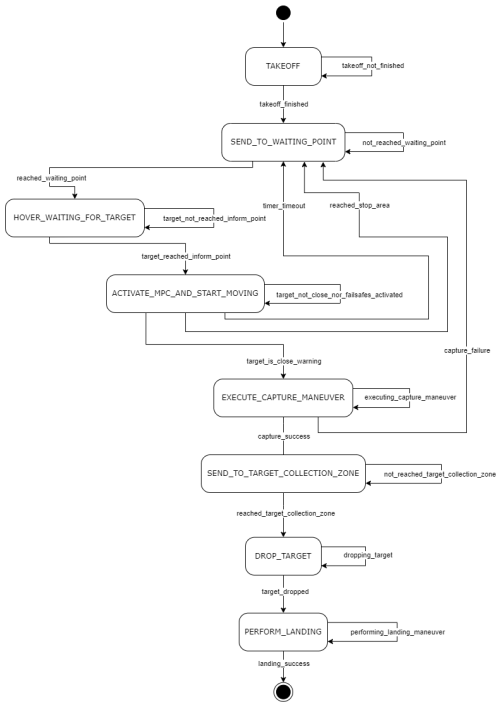


Fig. 8: State machine for the shuttle drone.

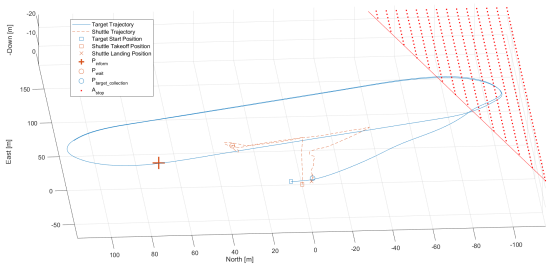


Fig. 9: Capture field tests: trajectories.

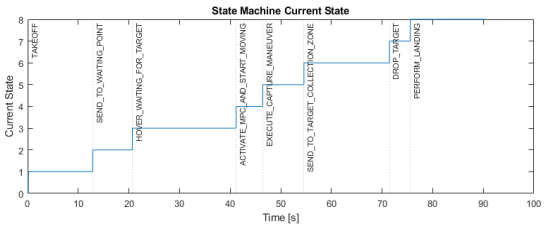


Fig. 10: Capture field tests: state machine.

drone, will be able to communicate with the target drone and with the ground infrastructure, and will be integrated with sensors and on-board computing for optimization and computer vision. The shuttle drone is based on a T-Motor M690B frame, a Holybro Pixhawk 6C autopilot, an NVIDIA Jetson Orin Nano computer, and in further developments related to this research effort, a RealSense D455 camera.

B. Field Test Results

At the moment of writing this paper, the only tests conducted so far were to test the controller's performance under different values of the d_{near} variable from Equation (20), so the tests presented do not represent the possible final performance of the developed algorithms. To improve the controller's performance, more real-world tests need to be conducted to properly tune the rest of the MPC parameters. However, the current data already shows promising results and shows signs of a possible successful capture for the final version of the algorithms developed when correctly tuning the MPC parameters for the real hardware. For the field tests shown in this section, the target drone was simulated inside the onboard computer of the real shuttle drone, allowing for a high messaging rate between both drones, which might impact the algorithm's performance when the target drone's states come from an outside machine.

The MPC parameters used for the tests presented are $\boldsymbol{v}_{max} = \begin{bmatrix}20 & 20 & 20\end{bmatrix}^T$ m/s, $N = 25$, $d_{near} = 1.5$ m, $h_{safe} = 2$ m, $\beta = 0$, $P_\psi = Q_\psi = 10$, $P_p = Q_p = \text{blkdiag}(90, 90, 70)$, $R = \text{blkdiag}(2, 2, 1, 10)$, and $P_v = Q_v = \text{blkdiag}(10, 10, 0)$.

The trajectory results of the field tests conducted for the capture scenario shown in Figure 9. The shuttle drone takes off and performs each step of the programmed mission according to the state machine from Figure 10. The shuttle reaches the hovering point and stays waiting for the target drone, and when the target reaches the inform point, the shuttle drone activates its MPC and stays close to its position until the target drone gets close, abiding by the constraint from Equation (12). When this moment comes, it's possible to see the shuttle trying to perform the capture maneuver but not with enough accuracy for it to be considered success.

To further analyze the results of the capture maneuver attempt, Figure 11 shows the temporal evolution of both drones' positions during the field test. From these graphs, it's possible to see that the real drone can perform the initial approximation to the target and maintain its position close to the target, as seen from the tracking in the horizontal plane. However, when analyzing the values during the capture moment (identified by the overlay with number 5), the results show that the capture maneuver would probably not be successful because of the deviations of the horizontal and vertical positions during this moment. The shuttle makes a first attempt to get closer to the target's position in the horizontal plane but the abruptness of this movement makes the tracking of the target's position unstable and the first descent attempt happens when the target drone is not close under the shuttle. As the drones keep flying, the shuttle starts converging to the target's trajectory and attempts to descend into the target again while better aligned in the horizontal plane.

Analyzing the graphs from Figure 12, it's easier to understand the capture attempts made by the shuttle. These values represent the closest point that the shuttle got to the target

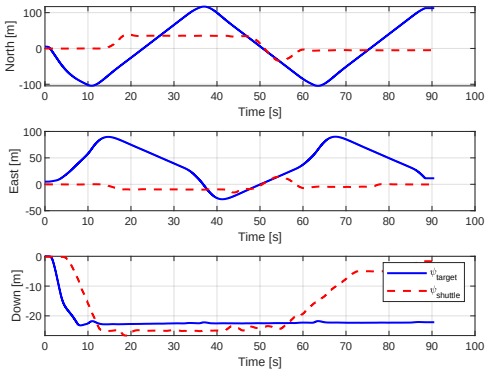


Fig. 11: Capture field tests: positions.

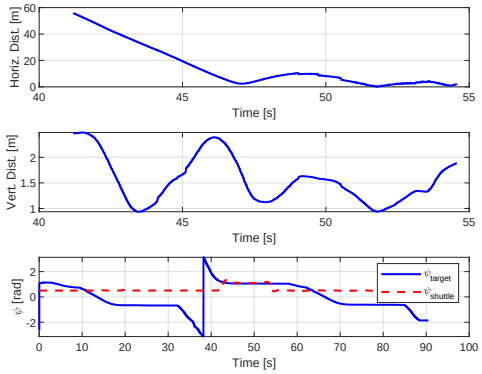


Fig. 12: Capture field tests: relative distances and yaw.

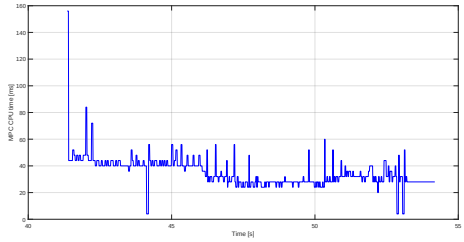


Fig. 13: Capture field tests: MPC computation times.

in the horizontal and vertical plane during the portion of the shuttle's mission that the MPC is active. From the vertical plane graph, it's possible to identify three local minimums with the first representing the adjustment of the drone's vertical position as soon as the MPC activates, and with the following two representing the capture descent attempts. If we compare both graphs at the same time, the first capture descent attempt happens after the shuttle tries to get close to the target and is getting further away in the horizontal plane because the shuttle's movements in the horizontal plane have an initial overshoot, and when the shuttle starts to lower itself, it's already at a horizontal distance from the target of more than $6\ m$. In the second capture attempt at $51.79\ s$, the shuttle is able to converge its trajectory to the target's and is better aligned in the horizontal plane, getting close to the target at a distance of $0.2566\ m$. In this instant, the shuttle descends and reaches a distance of $0.9392\ m$ from the target in the vertical plane.

Considering these values, the capture maneuver would not be considered a success because the values obtained were slightly above the metrics established for the success of the capture maneuver ($200\ mm$ to $250\ mm$ for the distance in the horizontal plane to the target because of the gripper's hand size, and $814.4\ mm$ for the distance in the vertical plane because of the gripper's arm length). With only these position values, it's not possible to predict what the drones' behavior would be if the tests were using two real drones with the capturing mechanism integrated in the shuttle, but even if by these results the capture would not be considered a success, the values obtained got real close to the required values under the real-world environment and it's plausible to assume that the performance of the controller would be more satisfactory if the MPC parameters are properly tuned in future tests.

The shuttle's yaw angle tracking, represented in Figure 12, also follows the expected behavior from the earlier tests, with the shuttle's yaw angle successfully converging to the target's during the capture moment and being aligned when the descent happens.

The other important aspect of the system's performance under the cooperative capture scenario, the MPC onboard computation time, is represented in Figure 13. Here it's shown the values obtained while the algorithms ran onboard in the NVIDIA Jetson Orin Nano. As seen in the figure, the onboard computer from the real drone is capable of running the developed algorithms in real-time while keeping small computation times, with the average time calculated from the results during the time of the field tests that the shuttle's MPC was active being of $35.1\ ms$. This conclusion is also confirmed by the stability of the shuttle's control.

As stated earlier, the field tests conducted in the real world for the cooperative capture scenario tested the controller's performance under different configurations of the MPC controller, more specifically, under different values for the d_{near} variable from Equation (20), with and without the velocity constraint from Equation (12). Table I showcases the results obtained from all the tests conducted. Here it's shown the Root Mean Squared Error (RMSE) from the shuttle's tracking of the target's position in all the axes during the execution of the capture maneuver (correspondent to state 5 from Figure 10), calculated using

$$PT_{RMSE} = \sqrt{\frac{1}{n}\sum_{k=1}^{n} \|p_{target_k} - p_{shuttle_k}\|^2},$$

where k represents each sample recorded for the time window considered, n represents the number of samples recorded for the time window considered, and $p_{drone_k} \in \mathbb{R}^3$ represents the respective drone's positions in each sample. The table also shows the minimum distances achieved by

TABLE I: Capture results from field tests.

Test	d_{near} (m)	PT_{RMSE} (m)	Min. Distance Horizontal (m)	Min. Distance Vertical (m)
With the constraint from Equation (12)				
1	1.5	5.8009	0.2566	0.9392
2	1.0	7.8626	0.5339	1.3609
3	0.5	6.5085	0.4480	1.3143
Without the constraint from Equation (12)				
4	1.5	10.4465	0.0198	1.5042
5	1.0	19.9091	2.1186	1.6259
6	0.5	9.9298	0.6283	1.4790

the shuttle during the execution of the capture maneuver calculated in the same way as the graphs from Figure 12.

From the results obtained, the performance of the MPC controller with the velocity constraint from Equation (12) was better than the performance of the controller without it, and the d_{near} distance values that produced the best approximation results were of $1.5\ m$, meaning that distances smaller than this resulted in the shuttle executing the capture maneuver not quickly enough and not being able to approximate itself closer to the target. The Position Tracking RMSE, PT_{RMSE}, values can be used to compare the performance of the different tests, where the values for the tracking during the whole execution of the capture maneuver are concordant with the achieved distances during the capture moment, meaning that the shuttle behaved better in the capture maneuver when it was able to better follow the target's position. The reason that these RMSE values aren't lower is because the considered values are calculated for the whole execution of the capture maneuver, where the shuttle starts a little far away from the target and only manages to get closer in specific instants, as seen in Figure 12 for the test 1.

The tests of the results shown in this table did not account for different values for the rest of the MPC controller's parameters, and it's expected that the tuning of the remaining parameters will affect greatly the controller's performance.

V. CONCLUSIONS

In this paper, we presented a state machine-based controller running a Model Predictive Control (MPC) strategy for a shuttle drone operating in a cooperative capture scenario with another target drone. The results for the simulations using the drone's mathematical models and for the simulations in a realistic environment with ROS+Gazebo and the PX4 software running in Software In The Loop mode (SITL) were deemed successful as it was possible to achieve the conditions to assume a successful capture maneuver between the drones. When testing the developed controller in the real world, the results obtained were not accurate enough to consider a successful capture, however, the tests validated the system's integration into real hardware running in real time and showed promising results for future tests when the controller is correctly tuned.

When it comes to future work, more field tests with the algorithms need to be conducted in order to better tune the controller's parameters and achieve better results for the capture maneuver. Furthermore, it's also important to test the system's behavior when using different vehicles, as opposed to simulating the target drone inside the shuttle's onboard computer, to understand the consequences of relying on WiFi communications for the exchange of information. With the results of this work, it's also evident the benefits that a complementary pose estimation system could bring to our controller's performance that could be implemented with the drone's onboard computer and camera to use for a machine vision tracking application.

ACKNOWLEDGMENT

In addition, we would also like to acknowledge the projects FirePuma (DOI:10.54499/PCIF/MPG/0156/2019), CTS (DOI:10.54499/UIDP/00066/2020), and LARSYS (DOI:10.54499/UIDB/50009/2020), as well as thank Francisco Velez Santos and Daniel Silvestre for their collaboration in obtaining the experimental results.

REFERENCES

[1] S. Dastgheibifard and M. Asnafi, "A review on potential applications of unmanned aerial vehicle for construction industry," 10 2018.

[2] V. Spurny, V. Pritzl, V. Walter, M. Petrlik, T. Baca, P. Stepan, D. Zaitlik, and M. Saska, "Autonomous firefighting inside buildings by an unmanned aerial vehicle," *IEEE Access*, vol. 9, pp. 15 872–15 890, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9328798

[3] M. Garcia, R. Caballero, F. Gonzalez, A. Viguria, and A. Ollero, "Autonomous drone with ability to track and capture an aerial target," *2020 International Conference on Unmanned Aircraft Systems, ICUAS 2020*, pp. 32–40, 9 2020.

[4] M. Vrba, Y. Stasinchuk, T. Báča, V. Spurný, M. Petrlík, D. Heřt, D. Žaitlík, and M. Saska, "Autonomous capture of agile flying objects using uavs: The mbzirc 2020 challenge," *Robotics and Autonomous Systems*, vol. 149, 3 2022.

[5] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, pp. 3463–3469, 11 2016. [Online]. Available: https://arxiv.org/abs/1611.09240v2

[6] M. Neunert, C. D. Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 1398–1404, 6 2016. [Online]. Available: https://www.researchgate.net/publication/303885303_Fast_nonlinear_Model_Predictive_Control_for_unified_trajectory_optimization_and_tracking

[7] L. Persson, "Model predictive control for cooperative rendezvous of autonomous unmanned vehicles," Ph.D. dissertation, KTH, Decision and Control Systems (Automatic Control), 2021, qC 20210518.

[8] F. Matos and B. Guerreiro, "Model predictive control strategies for parcel relay manoeuvres using drones," in *2021 International Young Engineers Forum (YEF-ECE)*, 2021, pp. 32–37.

[9] D. Oliveira, "Planning and control for cooperative launch and capture of drones," 2024.

[10] A. Dietrich, "Fast grasping mechanism for aerial and marine drone automatic capture."

[11] T. M. R. Beard, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012. [Online]. Available: https://github.com/randybeard/uavbook

[12] F. Santos, "Sensor network using a mix of cheap sensors and drones," 2022. [Online]. Available: https://fenix.tecnico.ulisboa.pt/downloadFile/1126295043840323/90077-Francisco_Santos-Thesis.pdf