AIAA SciTech Forum
7-11 January 2019, San Diego, California
AIAA Scitech 2019 Forum

10.2514/6.2019-1169

Model Predictive Control for Autonomous Ship Landing in a Search and Rescue Scenario

Linnea Persson[*] and Bo Wahlberg[†]
Department of Automatic Control, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

This paper presents a Model Predictive Control approach for autonomous landing of a quadcopter on the deck of a moving boat. The research is motivated by a large-scale demonstrator arena equipped with autonomous boats and drones that should collaborate to perform various tasks related to search and rescue missions. The landing maneuver is executed in a cooperative manner where both the boat and the drone take actions to reach their common objective. The maneuver is designed to be feasible under a range of conditions, including scenarios where the boat is moving across the water or when it is subjected to disturbances such as waves and winds. During the landing, the vehicles must also consider various safety constraints for landing safely and efficiently. The algorithms are implemented both in hardware-in-the-loop simulations, where we demonstrate some of the different scenarios that the algorithm is expected to handle, as well as on a real boat-drone system, on which initial tests have been carried out.

I. Introduction

Cooperative autonomous agents have a range of application scenarios that a single autonomous agent might not be able to complete, and there is even more potential when we also let heterogeneous sets of agents work together. Search and rescue is one such application which strongly benefits from heterogeneous multi-agent collaboration. Using vehicles that have different capabilities to move in air, on land, and over water, and which are equipped with different types of sensors and actuators, we are able to gather and process information about the environment while simultaneously executing necessary actions fast and efficiently. However, coordinating such heterogeneous systems in a safe and predictable manner can be a challenge, as the control actions of all agents in such a collaborative maneuver affects the relative state. As such, the difference in dynamics needs to be taken into account when computing the control actions.

In this paper, we consider a search and rescue mission at sea, where the objective is to identify and rescue victims of a ship accident using a collaborative fleet of Unmanned Aerial Vehicles (UAVs) and Unmanned Surface Vehicles (USVs). During such a mission, it is essential to provide help to the victims as rapidly as possible, all the while navigating through the accident debris and potentially bad weather or even storms. In this scenario, the UAVs could contribute significantly, using its overhead view to identify areas of interest and guide the USVs safely through the rescue mission, while the latter can focus their efforts on the physical work that is demanded. Examples of UAVs assisting in disaster relief efforts can be found in e.g. [1] and [2].

A large-scale research demonstration of a search and rescue scenario is to be performed outdoors the coast of Sweden in the fall of 2019, as a collaboration between several Swedish universities and companies [3]. The arena provides a realistic and large scale demonstration environment for developing and testing algorithms related to search and rescue, as well as cooperative maneuvering for surface and aerial vehicles. A map of the area is shown in Figure 1. As a result of the limited battery lives that the UAVs have, they also have a very limited flying range from their take-off point. Even with a take-off and recharging station located relatively close to the point of interest, it would both be inefficient and risky to require the drone to return to the station whenever the battery charge is low; and if the area of interest is far from land, it would likely not at all be possible to use the UAVs. Therefore, to be able to extend the flying range and the search time, it is necessary to be able to recharge the batteries during the mission. For this purpose, we are interested in using the USVs as moving landing platforms for the UAVs. With such take-off and landing capabilities, the drones can be deployed as needed and later land on the closest USV for recharging its batteries. The research presented in this paper considers collaborative autonomous landings of UAVs on the decks of the USVs.

[*]PhD Student, Department of Automatic Control, laperss@kth.se
[†]Professor, Department of Automatic Control, bo@kth.se.
This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) and the Swedish Science Foundation.

We focus on the scenario when one UAV and one USV collaborate to perform a landing maneuver while the USV is moving. There are sereval motivating factors for performing the landing while the USV is moving; one is that it might be desirable to let the USV continue moving due to an ongoing rescue operation, another reason might be that the USV can assist the UAV in a difficult landing situation (e.g. in strong winds or during an emergency landing). Landing the UAV while the USV is moving adds complexity to the system but will in the end contribute to a more flexible and useful solution which is applicable in a wider range of scenarios.

A. Contributions

The contributions of this paper are as follows:

- A model predictive control algorithm for a quadcopter/ship cooperative landing is derived, together with a distributed implementation structure for estimating the relative states.
- Derivation and estimation of models for the two vehicles based on real-world data.
- The landing is analyzed in realistic simulations, with different sources of external disturbances.
- The physical setup is presented, on which initial tests have been made and which will be used for future experiments and demonstrations.

The paper begins by introducing some background on autonomous landings, including a review of some of the research that has been done in this field in the past years. The use of Model Predictive Control (MPC) is motivated and some background of specific MPC methods of interest are mentioned. In Section III, the autonomous cooperative landing problem is formally stated and the system dynamics of the vehicles are derived and estimated. Following this, the relative state estimation and controller design are explained in Sections IV and V respectively. The implementation and the experimental setup are explained in Section VI, and Section VII provides experimental results, both from hardware-in-the-loop tests and from real flight tests. Finally, in Section VIII, we conclude with an analysis of the current systems and provide some directions for future developments.

II. Background

A. Autonomous and cooperative multivehicle landings

The topic of autonomous multivehicle landings has been dealt with in various projects over the last few years. Several papers consider the issue of relative state estimation, often from a computer vision perspective. In [4], an image recognicion technique is used where patterns of different sizes are combined to accurately estimate the relative position and pose between a quadcopter and a moving ground platform in a lab. In [5], the approach is instead to use an infrared camera which enables tracking of IR-lights under limited lighting conditions. The focus of both these papers is the state estimation, and the controller used in both cases are based on PID loops. Further, the landing platform is moving but is not collaborative.

An example of a collaborative landing maneuver can be found in [6], where a fixed wing UAV that cooperates with a semi-autonomous moving ground vehicle is considered. The UAV landing was implemented using a hybrid PID

Fig. 1 The search and rescue demonstrator is a large-scale arena for testing and developing algorithms for autonomous systems. Map data: Google/Lantmäteriet/Metria.

controller, considering safety sets and switching between “landing” and “go-around” modes until the drone is able to land safely. Experiments were carried successfully where the fixed-wing drone landed on a moving car at a velocity of around 70 km/h. The behavior of such a controller is safe but, as is noted in [7], it is also conservative since many retries might be required until a safe enough landing attempt is initialized. The same paper then builds on the previous system but with the PID controller replaced by a model predictive controller. The MPC approach is shown to improve the results by explicitly considering the system dynamics and taking the safety constraints into account directly in the computation of the input signals. This control method guarantees that the vehicles remain safe under typical circumstances.

An example of a project where a quadcopter landing on a boat is performed can be found in [8]. Here, the main focus is also on the visual tracking algorithm, and the sea vehicle takes no part in the control effort. The control method of choice was initially a PID controller, however, the authors note that the use of this controller results in a highly unstable performance. In the final landings, an ad hoc solution is instead used where the built-in hover mode is combined with adaptive “nudging” of the system in the correct direction.

B. Model Predictive Control

The autonomous cooperative landing is a challenging maneuver for many reasons. As has already been mentioned, both vehicles will simultaneously affect the relative state by adapting their velocities and heading angles. For this reason, the vehicles should in some way take the movement and dynamics of the other vehicle into account when computing its own control inputs. Further, in the example studied in this paper, both vehicles are subject to disturbances, such as waves and wind, and are traveling at potentially relatively high velocities. It is therefore critical that the system remains safe since an accident might harm equipment or even nearby people. As such, the autonomous landing will require a control algorithm that takes both disturbances and safety limitations into account. For this reason, we will in this paper present a model predictive control solution based on results in [7] but adapted to the environment and the specific safety criteria of the USV landing.

Model Predictive Control is an optimization-based control technique, where at every sampling instant a finite-horizon optimal control problem is solved. An overview of the control method can be found e.g. in [9]. There are several reasons for why MPC is a suitable controller for the landing maneuver. Because of the way that future trajectories are predicted, potential unwanted situations can often be avoided before they occur if the event is withing the planning horizon. MPC also has a natural way of dealing with constraints (such as actuator saturations) directly in the computation of the control inputs. As such, no ad hoc fix is needed and the controller will always return feasible control inputs. Additional constraints, e.g. for increasing the safety, can also be imposed, making the controller safe by construction.

In a model-based control approach such as MPC, having an accurate prediction model of the system dynamics is essential for how well the controller will perform. In MPC, an inaccurate model due to either modeling errors or external disturbances can make it difficult to achieve the desired control performance for the closed-loop system. If there are substantial errors in the model, then at each time step, the predicted trajectories and control inputs will be wrong, which might result in steady-state errors or even instabilities. To robustly handle a larger set of scenarios, the prediction needs to take these effects into account. Many different methods have been developed for handling robustness issues in MPC [10]. Offset free MPC is one flavor of model predictive control where external disturbances and modeling errors can be taken into account in the optimization [11], by including a disturbance term in the system dynamics. In [12], the results are extended to nonlinear MPC. In [13], a linear and a nonlinear MPC for quadcopter trajectory tracking under external disturbances using offset-free methods are compared. The results show that the controllers have comparable behavior but that the disturbance rejection capabilities are slightly better for the nonlinear MPC.

One of the potential issues that is often pointed out when using MPC is that the required computational load of solving finite-horizon optimal control problems at a high rate can be very large. However, the combination of modern hardware and recent algorithmic developments have made it possible to implement MPC also on systems that require very fast updates. Explicit MPC [14] is one such method where the optimization problem is computed off-line for the operating conditions of interest, turning the online optimization into a function evaluation. This method is mainly applicable to small problems due to the exponential growth of the dimension of the constraints. For larger problems, some methods for speeding the computations up can be found in e.g. [15].

III. Problem Statement

In this section, the cooperative landing problem is formally stated. First, the frames of references are derived, and then the dynamic of the quadcopter is stated. The parameters of the system are then estimated from flight data to get accurate models for implementation in the MPC framework in the next section.

A. Kinematics

We differentiate between two main frames – the inertial frame and the body frame, as illustrated in Figure 2. The inertial frame is here given in East-North-Up coordinates, and the body frame is given in Forward-Left-Up coordinates. This convention is chosen since it is the one which is provided in the the SDK of the quadcopter that we implemented the algorithms on, and as such, the data is already delivered in this frame of reference. More details on the quadcopter are provided in Section VI. With these reference frames, the yaw angle ψ is defined as the positive rotation between the forward-facing side of the drone and east. The relation between the body and inertial frame can be computed using the rotation matrix

$$\boldsymbol{R}_{ib} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\phi\sin\theta - \sin\psi\cos\phi & \cos\psi\cos\phi\sin\theta + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\phi\sin\theta + \cos\psi\cos\phi & \sin\psi\cos\phi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{1}$$

where ϕ, θ and ψ represent the roll, pitch and yaw angles respectively.

B. Quadcopter Dynamics

The forces and moments acting on the quadcopter are the result of gravity, drag and thrust from the four rotor blades, as well as external disturbances. The total thrust is directed in the negative z-direction in the body frame and is the sum of the thrust from the four individual motors. In the inertial frame, the total force can be expressed as

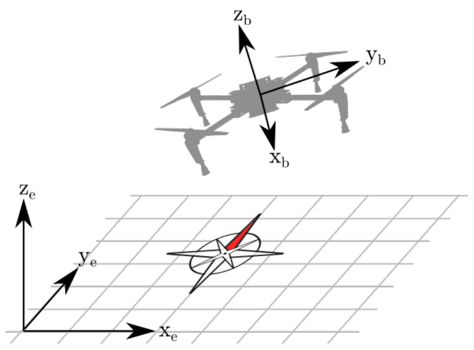
$$\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} = - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \boldsymbol{R}_{ib} \begin{bmatrix} 0 \\ 0 \\ \sum_{k=1}^{4} T_k \end{bmatrix} + F_{drag} + F_{ext},$$

where g is the gravitational acceleration, F_{drag} is the force from drag, and F_{ext} contains all external forces, e.g. due to wind. The control input at the lowest level is the thrust command to the four motors. For simplicity and safety, there is a built in attitude control system which is used to compute the individual motor thrust commands. Now instead the control inputs are the commanded roll angle ϕ_{cmd}, pitch angle θ_{cmd}, yaw rate $\dot{\psi}_{cmd}$, and vertical velocity w_{cmd}. The response of the vertical velocity w to a vertical velocity command is in the form of a Laplace transfer function expressed as

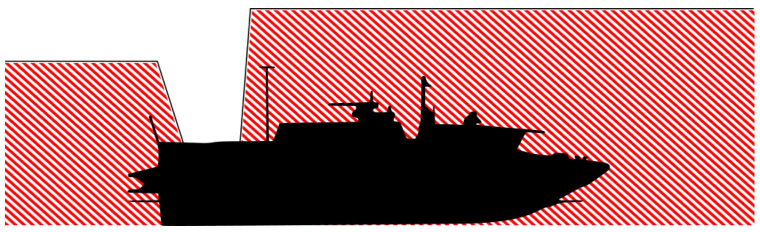
$$W(s) = \frac{k_w}{\tau_w s + 1} W_{cmd}(s), \tag{2}$$

meaning that the total thrust at time t can be written as

$$T(t) = \frac{g + \dot{w}(t)}{\cos\theta(t)\cos\phi(t)} = \frac{g + 1/\tau_w\,(k_w w_{des}(t) - w(t))}{\cos\theta(t)\cos\phi(t)}. \tag{3}$$



(a) The inertial and body frames.



(b) The quadcopter is constrained from entering areas too close to the boat. This is modelled as a safety constraint in the model predictive controller.

Fig. 2 Frames and constraints for the problem definition.

The attitude dynamics are approximated as second-order systems. For pitch and roll, the transfer functions from command to attitude are in the form of Laplace transfer functions expressed as

$$\Phi(s) = \frac{k_\phi \omega_\phi^2}{s^2 + 2\xi_\phi \omega_\phi s + \omega_\phi^2} \Phi_{cmd}(s) \tag{4}$$

$$\Theta(s) = \frac{k_\theta \omega_\theta^2}{s^2 + 2\xi_\theta \omega_\theta s + \omega_\theta^2} \Theta_{cmd}(s) \tag{5}$$

For the yaw, the rate control is modeled as

$$\Psi(s) = \frac{1}{s} \frac{k_\psi}{\tau_\psi s + 1} \dot{\Psi}_{cmd} \tag{6}$$

Now, the dynamics can be summarized as

$$\begin{aligned} \dot{\boldsymbol{p}}(t) &= \boldsymbol{v}(t) \\ \dot{\boldsymbol{v}}(t) &= \boldsymbol{a}(t) = -\boldsymbol{g} + \boldsymbol{R}_{ib}(t)\boldsymbol{T}(t) - \boldsymbol{D}\boldsymbol{v}(t) + \boldsymbol{a}_{ext}(t) \\ \dot{\boldsymbol{\theta}}(t) &= \boldsymbol{\omega}(t) \\ \dot{\boldsymbol{\omega}}(t) &= f_\omega(\boldsymbol{\omega}(t), \boldsymbol{\theta}(t), \boldsymbol{\theta}_{cmd}(t)) \end{aligned} \tag{7}$$

where $\boldsymbol{p}$ denotes the position, $\boldsymbol{v}$ the velocity, $\boldsymbol{a}$ the acceleration, $\boldsymbol{\theta}$ the attitude and $\boldsymbol{\omega}$ the angular acceleration. There are in total 12 states and 3 sources of external disturbances. The function f_ω is given by equations (4–6). It can be seen that this model is nonlinear only in the velocity derivative, where the input term $\boldsymbol{R}_{ib}\boldsymbol{T}$ is given by the vector

$$\boldsymbol{R}_{ib}\boldsymbol{T} = (g + \dot{w}) \begin{bmatrix} \frac{\cos\psi\cos\phi\sin\theta + \sin\psi\sin\phi}{\cos\theta\cos\phi} \\ \frac{\sin\psi\cos\phi\sin\theta - \cos\psi\sin\phi}{\cos\theta\cos\phi} \\ 1 \end{bmatrix} = (g + \dot{w}) \begin{bmatrix} \cos\psi\tan\theta + \frac{\sin\psi\tan\phi}{\cos\theta} \\ \sin\psi\tan\theta - \frac{\cos\psi\tan\phi}{\cos\theta} \\ 1 \end{bmatrix}$$

The system equations (7) can be written on state space form as follows

$$\frac{d}{dt}\begin{bmatrix} u \\ v \\ w \end{bmatrix} = -\begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \boldsymbol{R}_{ib}\begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}\begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{8a}$$

$$\frac{d}{dt}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = -\begin{bmatrix} \omega_\phi^2 & 0 & 0 \\ 0 & \omega_\theta^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} - \begin{bmatrix} 2\omega_\phi\xi_\phi & 0 & 0 \\ 0 & 2\omega_\theta\xi_\theta & 0 \\ 0 & 0 & 1/\tau_\psi \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{8b}$$

$$+ \begin{bmatrix} k_\phi\omega_\phi^2 & 0 & 0 \\ 0 & k_\theta\omega_\theta^2 & 0 \\ 0 & 0 & k_\psi/\tau_\psi \end{bmatrix}\begin{bmatrix} \phi_{des} \\ \theta_{des} \\ \psi_{des} \end{bmatrix}. \tag{8c}$$

C. Ship Dynamics

The USV has been modeled with motion around the yaw axis and in the forward and heave (vertical) direction. This way, the velocity and position are modeled using forward thrust and yaw angle, with an additional wave motion in the vertical direction. The motions in the other axes are for now assumed to be small enough to ignore. Worth noting is that the boat has a considerably larger moment of inertia and mass than the drone, and will as such be a significantly slower changing system. The state of the boat is indicated by the subscript b.

$$\begin{aligned} \dot{\boldsymbol{p}}_{\boldsymbol{b}}(t) &= \boldsymbol{v} \\ \dot{\boldsymbol{v}}_{\boldsymbol{b}}(t) &= \begin{bmatrix} \cos\psi_b(t)T_b(t) & \sin\psi_b(t)T_b(t) & 0 \end{bmatrix}^T - \boldsymbol{D}_b\boldsymbol{v}_b(t) + \boldsymbol{a}_{ext}(t) \\ \psi_b(t) &= \omega_b(t) \\ \dot{\omega}_b(t) &= f_\omega(\omega_b(t), \psi_b(t), \psi_{cmd}(t)) \end{aligned}, \tag{9}$$

where the notation is the same as for the drone (9). The yaw dynamic is modeled in the same way as the drone, using a first-order yaw rate control (6).

D. Safety constraints

There are some constraints with regards to the relative positioning that need to be fulfilled during the landing. One such constraint is related to the direction of approach. Because we do not wish to have the drone too close to the water, and because of the various equipment sticking up from the boat, we formulate a spatial constraint which forces the drone to approach the boat from approximately above, as illustrated in Figure 2b. This sort of nonconvex constraint can be expressed as

$$\boldsymbol{A}\begin{bmatrix} d(i) \\ h(i) \end{bmatrix} \geq \boldsymbol{k} - M\begin{bmatrix} b_1(i) \\ b_2(i) \end{bmatrix} \tag{10}$$

$$b_1(i) + b_2(i) \leq 1 \tag{11}$$

where the matrix A and the vector k together describe the shape of the region, and $b_1, b_2 \in \{0, 1\}^N$ are binary decision variables indicating which part of the constraint is active at a given time, and M is a sufficiently large constant. Details on how these can be chosen can be found in [7]. In addition to this spatial constraint, we also place a constraint on the vertical velocity at touchdown, which is expressed as

$$\begin{aligned} w(t) &\geq w_{min} \\ w(t) &\geq k_1 h(t) - w_{min,land} \end{aligned}$$

where $w_{min,land}$ is the limit of the (negative) vertical velocity at touchdown. This constraint makes the vertical velocity limitation dependent on the altitude.

The safety constraints, together with the state and input constraints, are expressed as

$$\begin{bmatrix} \boldsymbol{x}_{uav}^T(i) & \boldsymbol{x}_{usv}^T(i) & \boldsymbol{u}_{uav}^T(i) & \boldsymbol{u}_{usv}^T(i) \end{bmatrix} \in \mathcal{X}_i(t) \qquad \text{for } i = 1, \ldots, N. \tag{12}$$

In addition to this, we have a terminal state constraint expressed as

$$\begin{bmatrix} \boldsymbol{x}_{uav}^T(N) & \boldsymbol{x}_{usv}^T(N) \end{bmatrix} \in \mathcal{X}_N \qquad \text{for } i = 1, \ldots, N. \tag{13}$$

E. System identification

Accurate system models are needed both for the prediction models in the control algorithm and for the state estimation. The parameters of the equations (4–6) have been approximated using experimental data from real flight tests. The model parameters have been tuned to give suitable responses. Examples of the input-output responses along with the estimated system responses are shown in Figures 3–4. The resulting values are provided in Table 1.

$k_\phi = 0.95$	$\omega_\phi = 9.0$	$\xi_\phi = 0.72$
$k_\theta = 1.02$	$\omega_\theta = 11.0$	$\xi_\theta = 0.7$
$k_\psi = 1.0$	$\tau_\psi = 0.1$	
$k_w = 1.0$	$\tau_w = 0.4$	

Table 1 Model parameters for the drone dynamics, which have been estimated using flight data.

For the boat, no input-output data has been recorded since the input data is not yet accessible. However, GPS data has been collected during different maneuvers to approximate the responses and to estimate maneuverability limitations.

IV. State estimation

For the MPC to work as desired, the combined state of the drone and the boat must be known. To this end, a modified Extended Kalman filter has been implemented which predicts the drone and the boat states and combines them into a synchronized estimate. This section describes the state estimation filter design, and how different disturbance terms are approximated.

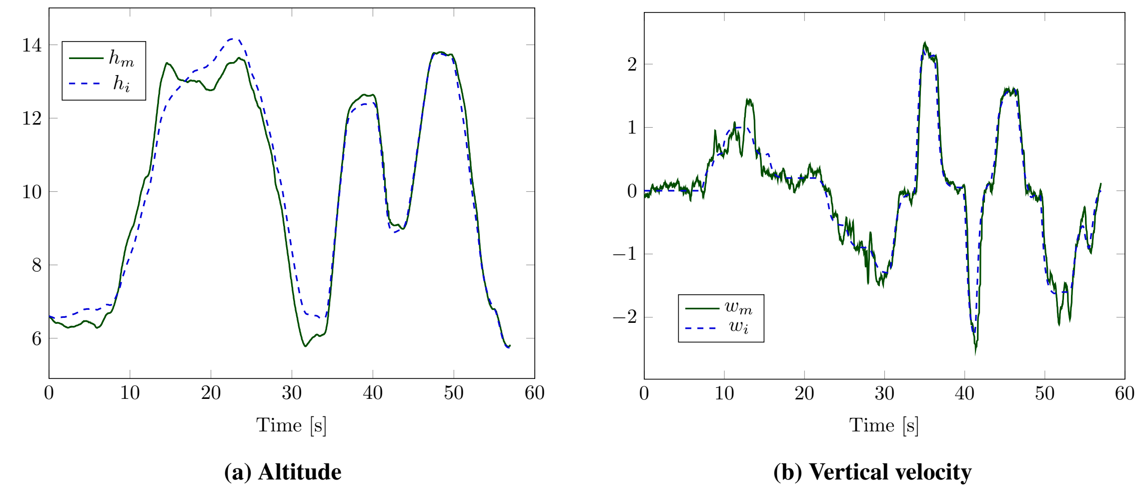


Fig. 3 Measured data (green) and generated model data (blue, dashed), from the input w_{cmd}.

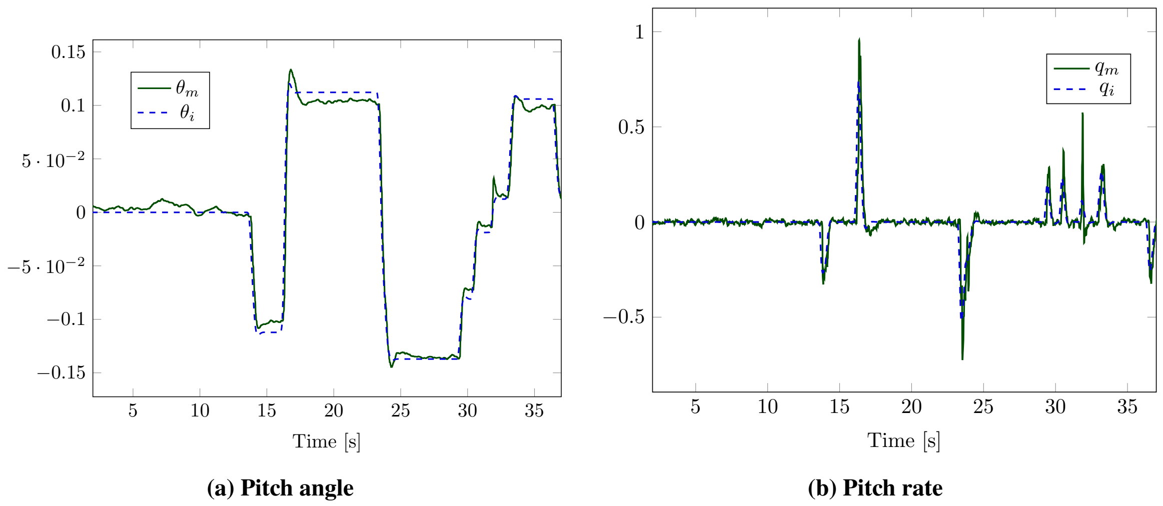


Fig. 4 Measured data (green) and generated model data (blue, dashed), from the input θ_{cmd}.

A. Kalman filter

The state estimation is run at a rate dt, which is chosen to be higher than the update frequency of the fastest sensor. The Kalman filter is implemented so that partial updates of the system states are allowed when not all sensor information is available, similar to the work described in [16]. The predicted values are corrected once every iteration if there are new measurements available from any of the sensors. One state estimation filter runs locally on each vehicle so that each vehicle at any instance have an estimation of the state of the global system. If for some reason no new data has been received from the other vehicle for a certain amount of time, then the state estimate of that vehicle is flagged as unreliable and this is included in the subsequent control and decision making. If a vehicle can estimate a reliable state for itself and the other vehicle, then a combined system state is computed. The state estimation setup is illustrated in Figure 5.

B. Heave estimation

The disturbances acting on the boat are assumed to be limited to heave motion, that is a vertical cyclic motion. The heave motion is for simplicity assumed to be a single sine wave with amplitude A and frequency ω.

$$p_z = A\sin(2\pi\omega \cdot t + \xi)$$

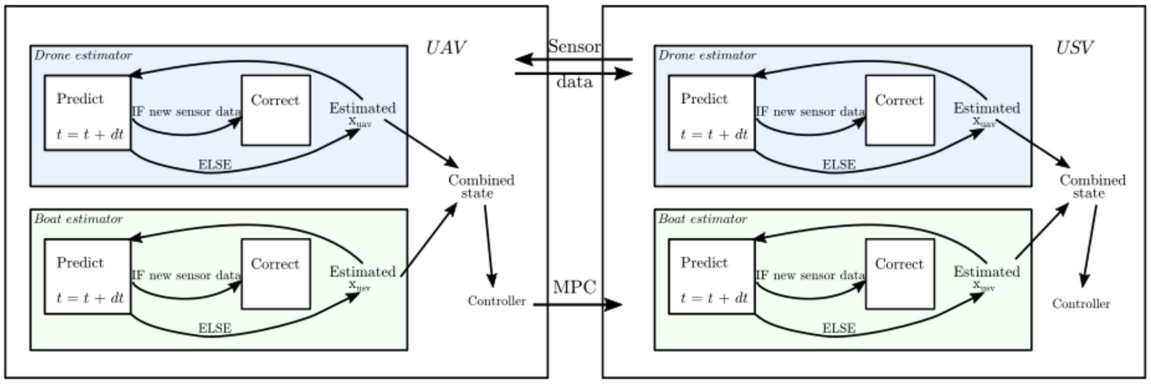


Fig. 5 The data of the modified Kalman Filter. Each vehicle has one Kalman filter of its own state and one for the other vehicle.

This could be extended to e.g. a combination of several sinusoids. We use the accelerometer data to approximate the frequency since the update rate of the accelerometer is much higher than the update rate of the GPS. The vertical acceleration measurements are given by

$$a_z = \frac{d^2}{dt^2} p_z = -(2\pi\omega)^2 A \sin(2\pi\omega \cdot t + \xi).$$

Data from the past M accelerometer measurements are saved into an array which is processed using the Fast Fourier Transform (FFT). The frequency ω is then chosen as the dominant frequency of the spectrum. For the amplitude estimation, the wave height is estimated using the power of the signal

$$A = \sqrt{\frac{2}{M} \sum_{i=0}^{M} p_z(i)^2}.$$

In Figure 6, it is illustrated how the heave motion is being estimated in real-time. To begin with, not enough data has been measured and so the estimation is not accurate. Then, as more data is collected, the estimation improves. The number of saved elements M and the sampling frequency h will affect the accuracy of the estimation and what frequencies can be detected. In Figure 6, $M = 1000$ and $h = 0.05$, and the period of the wave is 7 seconds.

V. Landing algorithm

This section will describe the implemented algorithm in detail. First, an overview of the landing procedure is given, then the details of the cooperative model predictive control algorithm are described.

A. Landing procedure

The suggested landing procedure is as follows. First, the UAV gets an instruction to return to the sea vehicle. The UAV then navigates towards the USV, and after it has reached a feasible landing set, the landing is initiated. This feasible landing set can for example be computed using backwards reachable set theory, and from within this set the optimization problem is known to be feasible which means that the problem will have a valid solution.

After the landing has been initiated, the vehicles share their respective states with eachother. Each vehicle has an internal model of its own dynamics and the dynamics of the other vehicle. The UAV uses this relative state information and solves the optimization problem which will be stated in the next section. The optimal trajectories are then returned to the USV, and both vehicles apply the first input in the optimal input sequence.

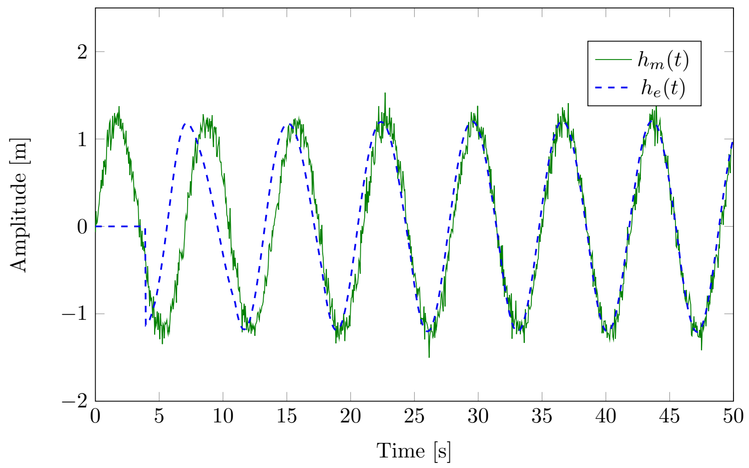


Fig. 6 The heave motion being estimated in real time. Up to $M = 1000$ values of the measured height and vertical acceleration are being saved and used to compute the current frequency, amplitude and phase shift of the wave. In this example, the sampling rate is 20 Hz.

B. Model predictive control algorithm

We will in this paper follow the approach taken in [7], where the cooperative controller is divided up into two parts; one corresponding to the vertical motion and one to the horizontal motion. The formal definition of the cooperative MPC problem is as follows. Given the current state of the UAV x^0_{uav} and the state of the USV x^0_{usv}, solve the horizontal optimization problem

$$\begin{aligned} \underset{\boldsymbol{x}_{uav},\, \boldsymbol{x}_{usv},\, \boldsymbol{u}_{uav},\, \boldsymbol{u}_{usv}}{\text{minimize}} \quad & \textstyle\sum_{i=1}^N l(x_{uav}(i),\ x_{usv}(i),\ u_{uav}(i),\ u_{usv}(i)) + l_f(x_{uav}(N+1),\ x_{usv}(N+1)) \\ \text{subject to} \quad & x_{uav}\,(0) = x^0_{uav}, \quad x_{usv}\,(0) = x^0_{usv} \\ & [x_{uav}(N+1),\ x_{usv}(N+1)]^T \in \mathcal{X}_f \\ & \left.\begin{aligned} & x_{uav}\,(i+1) = f_{uav}\,(x_{uav}(i), u_{uav}(i)) \\ & x_{usv}\,(i+1) = f_{usv}\,(x_{uav}(i), u_{usv}(i)) \\ & [x_{uav}(i),\ x_{usv}(i),\ u_{usv}(i),\ u_{usv}(i)]^T \in \mathcal{X}_{hor} \end{aligned}\right\} \text{for } i = 0, \dots, N \end{aligned} \tag{14}$$

where we optimize over the entire state trajectories $\boldsymbol{x}_{uav},\ \boldsymbol{x}_{usv}$ and corresponding input trajectories $\boldsymbol{u}_{uav},\ \boldsymbol{u}_{usv}$. Here, l and l_f are cost functions, typically chosen to be quadratic, and f_{usv} and f_{usv} are the state transition functions given in (7–9), and $\mathcal{X}_{hor}$ is the constraint set. The horizontal MPC is in this implementation chosen to be a nonlinear controller. This is to fully utilize the movements that the drone is capable of, and to let it reach high angles without making the linearization inaccurate.

For the vertical MPC, we use a linear MPC. This controller must take into account the safety constraints defined in Section III.D. Because we can precompute the binary decision variables and the relative distance using the trajectories of the horizontal MPC, we are left to solve a completely linear system in the vertical direction.

$$\begin{aligned} \underset{\boldsymbol{x}_{vert},\, \boldsymbol{u}_{vert}}{\text{minimize}} \quad & \textstyle\sum_{i=1}^N l(x_{vert}(i),\ u_{vert}(i)) + l_f(x_{vert}(N+1)) \\ \text{subject to} \quad & x_{vert}\,(0) = x^0_{uav} \\ & x_{vert}(N+1) \in \mathcal{X}_f \\ & \left.\begin{aligned} & x_{vert}\,(i+1) = f_{vert}\,(x_{vert}(i), u_{vert}(i)) \\ & [x_{vert}(i),\ u_{vert}(i)]^T \in \mathcal{X}_{vert}(b(i)) \end{aligned}\right\} \text{for } i = 0, \dots, N \end{aligned} \tag{15}$$

The major steps of the cooperative MPC algorithm are summarized in Algorithm 1.

Algorithm 1: The Collaborative MPC algorithm. Here, the horizontal MPC solver is represented by MPC$_{hor}$ and the vertical solver by MPC$_{ver}$. The complete state is given by x_0 and the collected external disturbances by d_0.

```
$t = 0$;
$x_0 = x(0)$;
$d_0 = d(0)$;
if landing then
    while $x \in \mathcal{X}_0$ do
        if $h \leq 0.1\ m$ then
            cut of motors;
            break;
        end
        else
            $x^\star_{uav}(i),\ u^\star_{uav}(t), x^\star_{usv}(i),\ u^\star_{usv}(t) \leftarrow \text{MPC}_{hor}(x_0, d_0)$ ;
            for $i = 0 \ldots N$ do
                $b(i) = (x^\star_{usv} \in \mathcal{X}_{land})?\ 1 : 0$;
            end
            $x^\star_{vert}(i),\ u^\star_{vert}(t) \leftarrow \text{MPC}_{ver}(x_0, d_0, b, x^\star_{usv})$;
            $t \leftarrow t + 1$;
            $x_0 \leftarrow x(t)$;
            $d_0 \leftarrow d(t)$;
        end
    end
end
```

VI. Implementation and experimental setup

The setup runs on two computers, one NUC 7i7BNB flight computer, which also runs all the low level drone control, and one laptop to be situated on the boat. Both computers are running Ubuntu 16.04 and are communicating using the message passing system of the Robot Operating System (ROS) in the Kinetic Kane distribution[*]. The vehicles are communicating using a local 2.4 GHz Wi-Fi network, with a measured average latency of less than 75 ms. All experiments, including the simulations, have been run using this setup meaning that the simulations run on the real hardware. For simulations, an additional computer running the DJI Assistant 2 Simulator have been used. The simulator mimics the dynamics of the drone and includes the possibility to test under various wind conditions. The boat is simulated using a separate Python script running on the boat laptop.

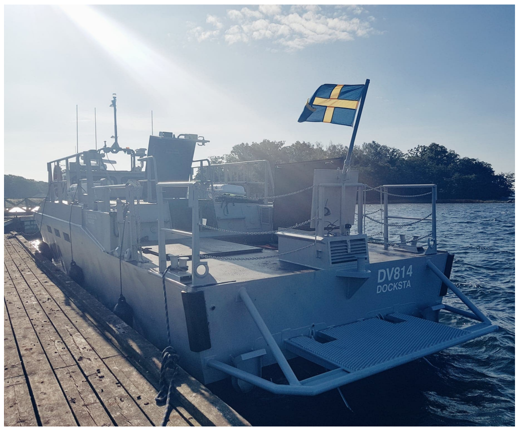
The MPC solvers have been implemented using the built-in code-generation tool of ACADO [17] (for the horizontal, nonlinear controller) and code-generation tool CVXGEN [18] for the vertical, linear controller.

A. Experimental setup

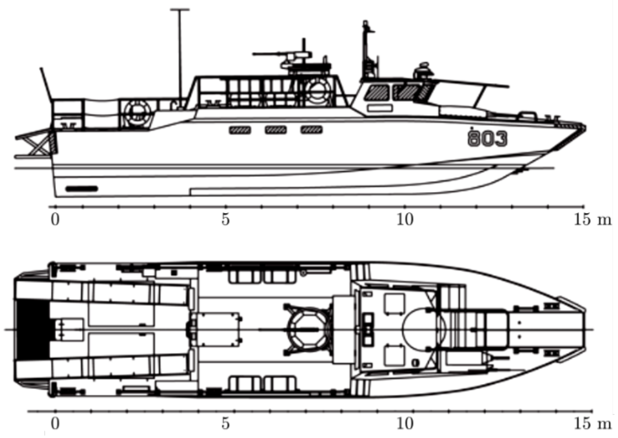
Experimental data has been collected both as preparation for future experiments, and for initial algorithmic evaluations. Experiments have been carried out outdoors with the boat docked at a pier. The Wi-Fi antenna and a laptop connected to the boat GPS system are placed on the deck of the boat.

Surface vehicle The boat that we have used for these experiments is a rebuilt CB90-class craft, as shown in Figure 7a. This is a highly agile boat, which can perform very sharp turns at high speeds. It can decelerate from its top speed of 40 knots to a full stop in 2.5 boat lengths. It is possible to connect directly to the boat GPS, from which NMEA sentences can be accessed. Specifically, this gives information on the boat latitude, longitude, speed and heading at a rate of 1 Hz. The boat is currently not fully equipped to be autonomous, but this is planned to happen within the next year. The designated landing area is on the stern deck, and is approximately 4×4 m^2, with the actual landing platform taking up approximately 2×2 m^2. Protective borders are attached to the fence to protect the drone from falling off.

[*]`http://wiki.ros.org/kinetic`



(a) The CB90 boat



(b) Drawing of the CB90

Fig. 7 The CB90 craft is 16 m long, with a landing platform on the stern of the boat.

Aerial vehicle The aerial vehicle that we use is a DJI matrice 100, a platform with open-access to much of the code base. The platform includs an SDK, granting access to sensor data coming from the IMU, GPS and barometer, as well as providing low-level control of the attitude and thrust of the drone. The drone is equipped with an inertial measurement unit, as well as a GPS and a pressure sensor. The pressure sensor outputs an estimated altitude. Some of the operating limitations are summarized in Table 2.

Max. Speed	22 m/s
Max. Speed of ascent/descent	5 m/s and 4m/s
Max. Pitch angular velocity	300°/s
Max. Yaw angular velocity	150°/s
Max. Tilt angle	35°

Table 2 Platform operating limitations of the quadcopter

VII. Experiments and results

A. Outdoot experiments

Tests have been performed where the drone has been controlled using the cooperative MPC algorithm, both in real-world experiments and in simulations. Figure 8 shows data from an outdoor experiment where the drone lands on a simulated, moving boat. The plot shows the relative distance in x and y, as well as the altitude h of the drone above the simulated boat. Instead of landing directly when the drone reaches an altitude of 10 cm, we let it hover above ground for approximately 5 seconds while following the boat. The controller both manages to drive the relative distance close to zero and keep the error below 50 cm. Further, no constraints are violated during the experiments. One noticeable effect is the ground effect when the drone gets close to landing. Although it can handle it without violating the constraints in most cases, this is something that could be improved for future landings.

B. Disturbance compensation simulations

The disturbance compensation has been tested by adding different types of disturbances, mainly represented as wind gusts acting on the quadcopter, or as vertical waves on the boat. Figure 9 shows the result of two simulations with identical wind conditions. The red dotted line shows the distance dependent altitude constraint. In the plot to the left it is shown how the system reacts to a strong wind when the MPC has no disturbance compensation. The result is a steady-state error which the MPC cannot compensate for, and as such the UAV is not able to land without violating the safety constraints. The vehicle then repeatedly plans to land, fails, and then has to rapidly increase its altitude to avoid violating the safety constraints. In the plot to the right, disturbance compensation is turned on. There is an overshoot in δy due to the wind, however, as the UAV gets a better and better estimate of the wind disturbance, it is able to compensate for it and land without violating the safety constraint.

Figure 10 shows the altitude of the UAV and the USV separately when the USV is subjected to wave disturbances. The waves have a period of 10 seconds and an amplitude of 0.8 m. At around $t = 8$ seconds, the UAV temporarily pauses the descent to allow the vehicles to align better. Then, as it continues to descend, it lands with a safe relative vertical velocity using the estimated heave motion as a reference for the altitude.

VIII. Conclusions and Future Work

This paper has presented the experimental setup of a cooperative landing algorithm to be applied in a search and rescue scenario. The system and landing algorithm have been derived and necessary parameters have been estimated using experimental data. The algorithm which has been proposed both plans the trajectories and computes the control inputs for both vehicles simultaneously. The system and algorithm have been tested both in hardware-in-the-loop simulations and in real flight tests.

The controller can already at this point handle many different types of errors and disturbances, as was shown in Section VII. We could potentially improve the robustness and performance even more if we take known and predictable disturbances into account directly. These disturbances include the ground effect from the rotor blades, and the drag from the boat. We also need to extend the current model to include other types of motions on the boat, including sideways rolling wave motions and drifting. Another approach to the disturbance modeling could be to use methods from machine learning to estimate the disturbances. This could potentially allow us to estimate the disturbances faster than we are able to at the moment.

Although the cooperative MPC algorithm and the simulations are all made using a simulated cooperative coat, the real boat does not currently have full support for autonomous driving implemented. As such there have been no tests where the system has been truly cooperative, other than in simulations. After the autonomous features have been added to the boat, there will be full-scale tests of the controller. We expect to have the autonomous boat available this spring and so autonomy experiments can be carried out some time after this.

The state estimation of the real-world boat is based only on GPS data, since there have been no other sensors available up until this point. This is in contrast to the simulated boat which is assumed to have access also to measurements from accelerometer and gyro. This is needed to get an accurate enough relative position estimate of the vehicles. For the future we are planning both to add more sensors to the boat, and to make better use of the GPS data in order to

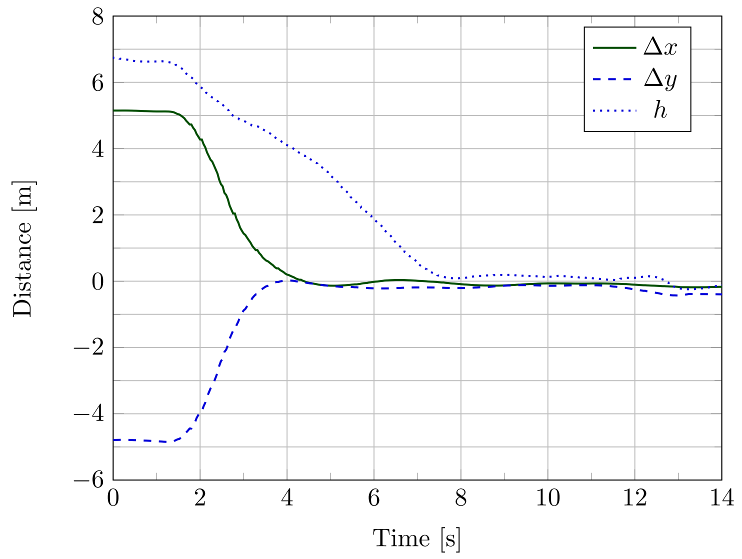


Fig. 8 Outdoors test of the quadcopter control with a simulated boat. The relative distance converges fast and after the drone gets within 20 cm of the ground, the distance does not exceed 0.5 m.

increase the accuracy of the relative positioning. This will be achieved by taking the correlation of GPS error sources into account, a method similar to the corrections done in Real-Time Kinematic GPS.

Finally, there will be more extensive experiments of the full system next year.

IX. Acknowledgments

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and the Swedish Science Foundation. We would like to thank the WARA project and in particular Saab Kockums for their support. Finally we would like to thank our colleagues Inês Lourenço and Martin Biel who on multiple occasions assisted in setting up and running the outdoor flight experiments.

References

[1] Xu, Z., Yang, J., Chaoyong, P., Wu, Y., Jiang, X., Li, R., Zheng, Y., Gao, Y., Liu, S., and Tian, B., "Development of an UAS for post-earthquake disaster surveying and its application in Ms7.0 Lushan Earthquake, Sichuan, China," *Computers & Geosciences*, Vol. 68, 2014. doi:10.1016/j.cageo.2014.04.001.

[2] Ezequiel, C., Cua, M., Libatique, N., Tangonan, G., Alampay, R., Labuguen, R., Favila, C., Honrado, J. L., Canos, V., Devaney, C., Loreto, A., Bacusmo, J., and Palma, B., "UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development," *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, 2014, pp. 274–283. doi:10.1109/ICUAS.2014.6842266.

[3] Wallenberg AI, Autonomous Systems and Software Program, "WARA-PS (Public Safety)," `http://wasp-sweden.org/demonstrators/wara-ps-public-safety/`, 2018. Accessed: 2018-12-04.

[4] Araar, O., Aouf, N., and Vitanov, I., "Vision Based Autonomous Landing of Multirotor UAV on Moving Platform," *Journal of Intelligent & Robotic Systems*, Vol. 85, No. 2, 2017, pp. 369–384. doi:10.1007/s10846-016-0399-z, URL `https://doi.org/10.1007/s10846-016-0399-z`.

[5] Wenzel, K. E., Masselli, A., and Zell, A., "Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle," *Journal of Intelligent & Robotic Systems*, Vol. 61, No. 1, 2011, pp. 221–238. doi:10.1007/s10846-010-9473-0, URL `https://doi.org/10.1007/s10846-010-9473-0`.

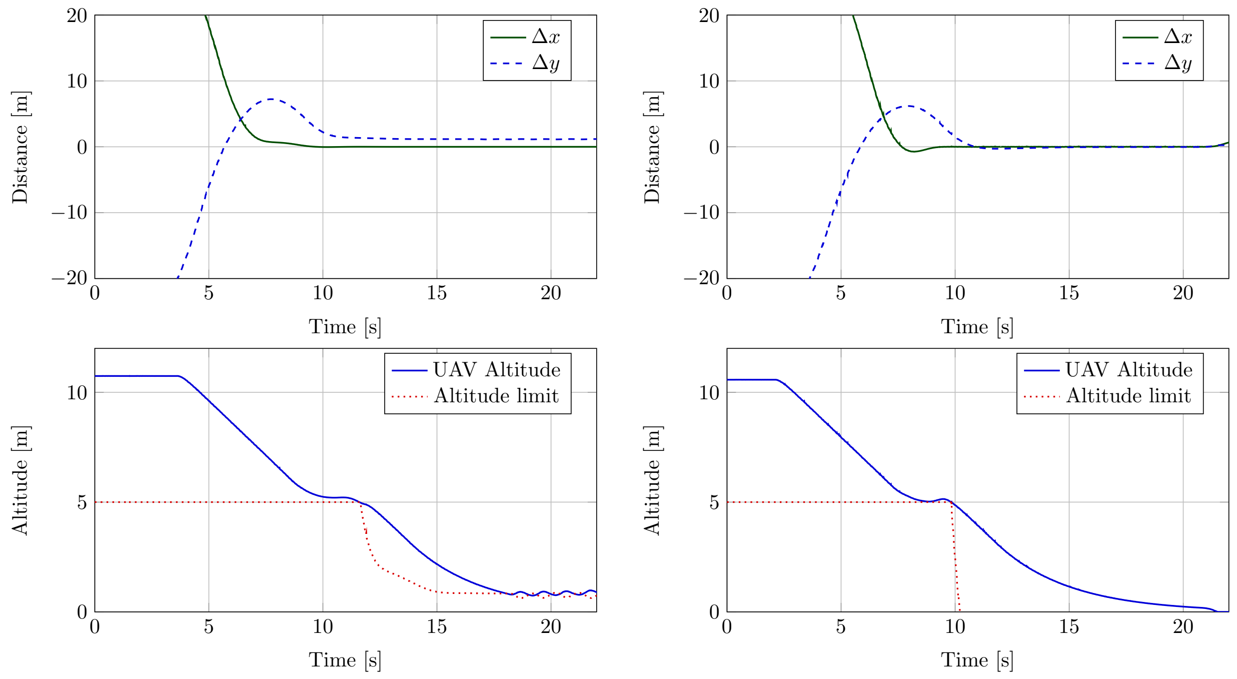


Fig. 9 By estimating the disturbance and feeding it into the optimization problem, offset-free tracking is achieved.

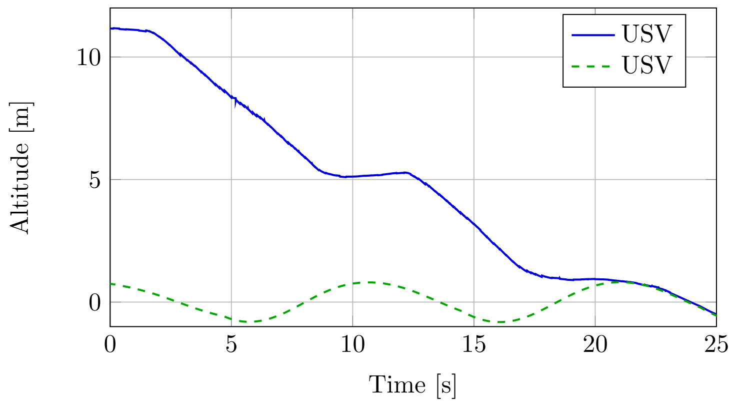


Fig. 10 A landing with simulated waves causing a cyclic up and down motion of the deck of the boat. Using the heave estimation presented in Section IV. The drone adapts its vertical velocity to avoid touching down with a too large vertical velocity.

[6] Muskardin, T., Balmer, G., Persson, L., Wlach, S., Laiacker, M., Ollero, A., and Kondak, K., "A Novel Landing System to Increase Payload Capacity and Operational Availability of High Altitude Long Endurance UAVs," *Journal of Intelligent & Robotic Systems*, Vol. 88, No. 2, 2017, pp. 597–618. doi:10.1007/s10846-017-0475-z, URL `https://doi.org/10.1007/s10846-017-0475-z`.

[7] Persson, L., Muskardin, T., and Wahlberg, B., "Cooperative rendezvous of ground vehicle and aerial vehicle using model predictive control," *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 2819–2824. doi:10.1109/CDC.2017.8264069, URL `https://doi.org/10.1109/CDC.2017.8264069`.

[8] Venugopalan, T. K., Taher, T., and Barbastathis, G., "Autonomous landing of an Unmanned Aerial Vehicle on an autonomous marine vehicle," *2012 Oceans*, 2012, pp. 1–9. doi:10.1109/OCEANS.2012.6404893, URL `https://doi.org/10.1109/OCEANS.2012.6404893`.

[9] Rawlings, J. B., "Tutorial overview of model predictive control," *IEEE Control Systems*, Vol. 20, No. 3, 2000, pp. 38–52.

[10] Bemporad, A., and Morari, M., "Robust model predictive control: A survey," *Robustness in identification and control*, Springer, 1999, pp. 207–226.

[11] Maeder, U., Borrelli, F., and Morari, M., "Linear offset-free Model Predictive Control," *Automatica*, Vol. 45, No. 10, 2009, pp. 2214 – 2222. doi:https://doi.org/10.1016/j.automatica.2009.06.005, URL `http://www.sciencedirect.com/science/article/pii/S0005109809002969`.

[12] Morari, M., and Maeder, U., "Nonlinear offset-free model predictive control," *Automatica*, Vol. 48, No. 9, 2012, pp. 2059 – 2067. doi:https://doi.org/10.1016/j.automatica.2012.06.038, URL `http://www.sciencedirect.com/science/article/pii/S0005109812002932`.

[13] Kamel, M., Burri, M., and Siegwart, R., "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles," *IFAC-PapersOnLine*, Vol. 50, No. 1, 2017, pp. 3463 – 3469. doi:https://doi.org/10.1016/j.ifacol.2017.08.849, URL `http://www.sciencedirect.com/science/article/pii/S2405896317313083`, 20th IFAC World Congress.

[14] Alessio, A., and Bemporad, A., *A Survey on Explicit Model Predictive Control*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 345–369. doi:10.1007/978-3-642-01094-1_29, URL `https://doi.org/10.1007/978-3-642-01094-1_29`.

[15] Wang, Y., and Boyd, S., "Fast Model Predictive Control Using Online Optimization," *IEEE Transactions on Control Systems Technology*, Vol. 18, No. 2, 2010, pp. 267–278. doi:10.1109/TCST.2009.2017934.

[16] Moore, T., and Stouch, D., "A generalized extended kalman filter implementation for the robot operating system," *Intelligent Autonomous Systems 13*, Springer, 2016, pp. 335–348.

[17] Houska, B., Ferreau, H., and Diehl, M., "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, Vol. 32, No. 3, 2011, pp. 298–312.

[18] Mattingley, J., and Boyd, S., "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, Vol. 13, 2012. doi:10.1007/s11081-011-9176-9.