Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Distributed Model Predictive Control for Cooperative Landing

Robert Bereza*, **Linnea Persson***, **Bo Wahlberg***†

*Division of Decision and Control Systems, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
†Division of Automatic Control, Linköping University, 581 83 Linköping, Sweden
(e-mail: {robbj,laperss,bo}@kth.se)

Abstract: We design, implement and test two control algorithms for autonomously landing a drone on an autonomous boat. The first algorithm uses distributed model predictive control (DMPC), while the second combines a cascade controller with DMPC. The algorithms are implemented on a real drone, while the boat's motion is simulated, and their performance is compared to a centralized model predictive controller. Field experiments are performed, where all algorithms show an ability to land while avoiding violation of the safety constraints. The two distributed algorithms further show the ability to use longer prediction horizons than the centralized model predictive controller, especially in the cascade case, and also demonstrate improved robustness towards breaks in communication.

Keywords: Autonomous vehicles, Distributed control, Model-based control, Flight control, Autonomous landing, Cooperative control

1. INTRODUCTION

Efficient communication and distributed sensing capabilities of multi-agent systems allow groups of heterogeneous vehicles to cooperatively solve advanced control problems. Search-and-rescue at sea is one such challenging application where unmanned aerial vehicles (UAVs) cooperating with unmanned surface vehicles (USVs) can identify areas of interest or people in need of aid, as well as provide physical assistance when necessary. Because the rescue might take place far out at sea, the ability to autonomously land the UAV on a USV to e.g. recharge batteries is an important feature. In this paper, we consider the problem of landing a quadrotor UAV (Figure 1) on a cooperating USV, while avoiding collision with the USV's antennas and other protruding parts. The vehicles have heterogeneous and decoupled dynamics, but coupled objectives and constraints.

To address this problem, we consider the use of distributed model predictive control (DMPC). The use of MPC allows the vehicles to explicitly take safety and input constraints into account. The algorithms introduced in this paper are evaluated based on their computational efficiency, their ability to handle safety constraints, and their robustness towards breaks in communication. Efficiency is crucial because using sufficiently long prediction horizons is important for the stability of MPC controllers, see e.g. Alamir and Bornard (1995). Furthermore, because these algorithms are designed to work on a real system, they must have safe and reliable performance even if communication issues occur.

The main contributions of this paper are:

- Design and implementation of two distributed MPC algorithms on a UAV;
- Testing of the computational efficiency of the algorithms, compared to a centralized MPC, using hardware-in-the-loop (HIL) simulations;
- Testing the performance of the algorithms, compared to a centralized MPC, in outdoor field tests with loss of communication between the vehicles introduced.

The two distributed algorithms demonstrate improved robustness towards communication breaks and an ability to use longer predicted horizons compared to the centralized MPC algorithm.

Fig. 1. The DJI Matrice 100 drone used in the experiments.

*This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) and the Swedish Research Council

1.1 Related Work

Autonomous landings of UAVs on moving platforms have been considered in several projects over the last years. Many such projects consider non-cooperative landings where the UAV lands on a moving target, e.g. Herissé et al. (2012); Kim et al. (2017); Lee et al. (2012). A cooperative autonomous landing using proportional controllers with hybrid safety regions is presented in Muskardin et al. (2017). In Persson (2019), the landing problem is solved in a cooperative manner using an MPC algorithm which plans a safe trajectory and follows it. Though these algorithms are cooperative, they are centralized and all control is computed by the same computer. The use of *distributed* and cooperative control, where the computational load is divided among the cooperating agents, is common in e.g. formation control of groups of UAVs, see e.g. Keviczky et al. (2008); Huck et al. (2014); Liu et al. (2018); Zou et al. (2018). This approach is not yet commonly applied in autonomous landings algorithms. In this paper, we develop a distributed MPC for solving a spatially constrained cooperative landing problem. We specifically investigate the computational efficiency of distributed algorithms and their ability to function despite breaks in communication.

A thorough overview of different types of DMPC can be found in Negenborn and Maestre (2014) or Müller and Allgöwer (2017). A DMPC algorithm able to handle landing problems where the agent dynamics are decoupled, but both constraints and objectives are coupled, is described in Müller et al. (2012). The first algorithm designed in this paper is similar to it, but changes have been made that make the solutions less conservative at the cost of losing stability and feasibility guarantees. The second algorithm designed in this paper is an extension of the first and uses cascade DMPC controllers to achieve longer prediction horizons. The remainder of this paper is organized as follows. Section 2 describes the dynamical models used for the vehicles. Section 3 covers the algorithms and the approach used for constructing them, while experimental results using the algorithms are described in Section 4. Finally, future work and conclusions are discussed in Section 5.

1.2 Preliminaries

States and control inputs are denoted by $\boldsymbol{x}$ and $\boldsymbol{u}$, respectively. Discrete-time variables are denoted with square brackets $\boldsymbol{x}[k]$. These variables are sub-indexed by h, v and b, corresponding to the *horizontal* UAV problem, *vertical* UAV problem, or horizontal USV problem. Values for variables over an entire prediction horizon are denoted using a bar, e.g. $\bar{\boldsymbol{x}} = \left[\boldsymbol{x}^T[0], ..., \boldsymbol{x}^T[N]\right]^T$. Quadratic stage costs are written as $l(\boldsymbol{y}_1, ..., \boldsymbol{y}_n) = \boldsymbol{y}_1^T Q_1 \boldsymbol{y}_1 + ... + \boldsymbol{y}_n^T Q_n \boldsymbol{y}_n$, where Q_i are the cost matrices. The letter j similarly denotes quadratic terminal costs. Finally, the superscript $*$ denotes optimal values of variables, obtained from the solution of an optimization problem.

2. PROBLEM STATEMENT

In this work, we consider the problem of autonomously landing a UAV on a USV in a spatially constrained environment. In particular, a DMPC controller is used to make the algorithms both more efficient and less dependent on uninterrupted communication.

2.1 Dynamics and Spatial Safety Constraints

The state of the UAV consists of its position (p_x, p_y, p_z) and velocity (u, v, w) along the x-, y-, and z-axes respectively, as well as its attitude ϕ (roll), θ (pitch), and ψ (yaw). These dynamics are based on the model from (Persson, 2019, pp. 66–68). The attitude is controlled by the inputs ϕ_{cmd}, θ_{cmd}, and ψ_{cmd} respectively, while the vertical velocity is controlled by the input w_{cmd}. For simplicity, it is assumed that ψ will be kept at 0 at all times. Further, it is assumed that there exists a sufficiently fast lower-level control for the thrust and attitude of the UAV, such that the responses from the control inputs are approximately first-order systems. We now choose the following roll and pitch commands

$$\theta_{cmd} = \arctan \frac{a_x^*}{g + \frac{1}{\tau_w}(k_w w_{cmd}^* - w)},$$
$$\phi_{cmd} = \arctan \frac{-a_y^*}{g + \frac{1}{\tau_w}(k_w w_{cmd}^* - w)}$$

where a^* is the desired acceleration and τ_w and k_w are model parameters. This allows us to approximate the forward dynamics linearly as $\dot{u} = -k_{dx} u + a_x^*$, where k_{dx} represents drag, and corresponding for side motion. It is now possible to separate the UAV motion into two sets of equations, one for the vertical motion and one for the horizontal motion

$$\boldsymbol{x}_h[k+1] = f_h(\boldsymbol{x}_h[k], \boldsymbol{u}_h[k])$$
$$\boldsymbol{x}_v[k+1] = f_v(\boldsymbol{x}_v[k], \boldsymbol{u}_v[k])$$

where $\boldsymbol{x}_h[k] = [p_x[k], p_y[k], u[k], v[k]]^T$ is the horizontal state, $\boldsymbol{x}_v[k] = [p_z[k], w[k]]^T$ is the vertical state and both f_h and f_v are linear functions.

We consider a simple linear dynamical model for the boat. Only the horizontal dynamics are considered, with the control inputs being the horizontal acceleration. More advanced variants of dynamical models for USVs can be found in e.g. Fossen (2011).

When landing, we require the UAV to descend in a way such that it avoids antennas and other protruding parts on the USV. This constraint is captured by forcing the planned path to stay inside a safe region, as is illustrated in Figure 2. This region can be described by two linear

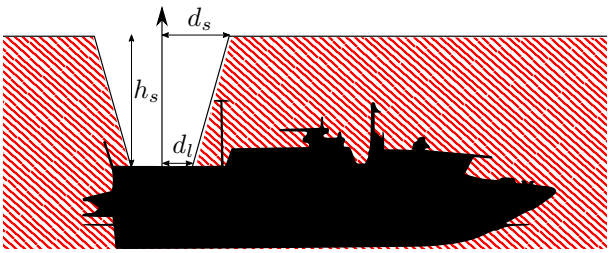


Fig. 2. The striped region is considered a dangerous area, and should be avoided by the controller.

inequalities. Details on the formulation of such a safety constraint are given in Persson et al. (2017). Note that d denotes the distance between the vehicles in the horizontal plane, while h is the altitude of the UAV. A binary variable b will also be used, and it will be set to 1 only when $d < d_s$.

3. CONTROL STRATEGY

In this section, two different DMPC algorithms are introduced, together with the centralized MPC algorithm used for comparison.

With decoupled horizontal and vertical dynamics, the trajectory planning can be decoupled as well, adapted from Persson (2019):

(1) Solve horizontal MPC to obtain a_x^* and a_y^*
(2) Compute predicted distances $\bar{d}$
(3) Based on the predicted distances, compute binary variables $\bar{b}$ representing the controllers' ability to land at a certain point on the trajectory
(4) Solve vertical MPC using $\bar{d}$ and $\bar{b}$ to obtain w_{cmd}^*
(5) Using a_x^*, a_y^*, and measured w, find ϕ_{cmd} and θ_{cmd}
(6) Apply w_{cmd}^* as well as appropriate ϕ_{cmd} and θ_{cmd}

3.1 Centralized Control Problem

In the centralized algorithm, the control inputs for both vehicles are computed by the same process. Because the dynamics of the UAV are faster than the dynamics of the USV, the computations are done on the drone computer. The horizontal optimization problem is given by:

$$\begin{aligned}\underset{\bar{\boldsymbol{x}}_h,\bar{\boldsymbol{x}}_b,\bar{\boldsymbol{u}}_h,\bar{\boldsymbol{u}}_b}{\text{minimize}}\ & \textstyle\sum_{k=0}^{N-1} l_c(\boldsymbol{x}_h[k]-\boldsymbol{x}_b[k],\boldsymbol{u}_h[k],\boldsymbol{u}_b[k]) \\ & + m_b(\boldsymbol{x}_b[k]) + j_c(\boldsymbol{x}_h[N]-\boldsymbol{x}_b[N]) \qquad (1)\\ \text{subject to}\ & \left.\begin{aligned}\boldsymbol{x}_h[k+1] &= f_h(\boldsymbol{x}_h[k],\boldsymbol{u}_h[k])\\ \boldsymbol{x}_b[k+1] &= f_b(\boldsymbol{x}_b[k],\boldsymbol{u}_b[k])\end{aligned}\right\} k=0,...,N-1\\ & \left.\begin{aligned}\boldsymbol{x}_h[k] &\in \mathcal{X}_h, \quad \boldsymbol{x}_b[k] \in \mathcal{X}_b\\ \boldsymbol{u}_h[k] &\in \mathcal{U}_h, \quad \boldsymbol{u}_b[k] \in \mathcal{U}_b.\end{aligned}\right\} k=0,...,N\end{aligned}$$

where the quadratic cost $m_b(\boldsymbol{x}_b[k])$ penalizes deviations from some specified USV velocity, letting the USV follow multiple objectives. The index c denotes the *centralized* problem. The state constraints include the safety constraints from Section 2.1, as well as velocity constraints and touchdown constraints, restricting the UAV descend velocity. The vertical problem is similarly given by:

$$\begin{aligned}\underset{\bar{\boldsymbol{x}}_v,\bar{\boldsymbol{u}}_v}{\text{minimize}}\ & \textstyle\sum_{k=0}^{N-1} l_v(b[k]\boldsymbol{x}_v[k],\boldsymbol{u}_v[k]) + b[N] j_v(\boldsymbol{x}_v[N])\\ \text{subject to}\ & \boldsymbol{x}_v[k+1] = f_v(\boldsymbol{x}_v[k],\boldsymbol{u}_v[k]) \qquad (2)\\ & \boldsymbol{x}_v[k] \in \mathcal{X}_v(d[k]),\ \boldsymbol{u}_v[k] \in \mathcal{U}_v, k=0,...,N.\end{aligned}$$

Note that the binary variables b are precomputed using the predicted distance $\bar{d}$ as described above.

3.2 Distributed Control Problem

The first new algorithm uses a straightforward distributed MPC approach, and it will be referred to simply as the *distributed algorithm*. In this algorithm, the computation of the horizontal problem is distributed between the vehicles.

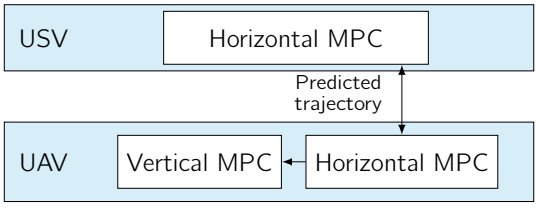


Fig. 3. In the distributed controller, the horizontal problem is distributed between the UAV and USV, and the vehicles share their predicted trajectories.

Each vehicle computes its own control inputs, based on a predicted state trajectory of the other vehicle. The vehicles exchange their predicted trajectory and adjust their own trajectory to better adapt themselves to each other. The schematic of this algorithm is illustrated in Figure 3. One advantage of this algorithm is that it allows the vehicles to act more independently from one another, in case of e.g. a short communication loss. This stands in contrast to the centralized case, where the USV cannot act without control inputs from the UAV. Dividing an optimization problem into two smaller ones also decreases the solution time. However, this is at the cost of performance since neither of the algorithms has full knowledge of the entire system. The vertical problem in the distributed case is only solved on the UAV and is therefore identical to (2). The horizontal problem of both vehicles is as follows:

$$\begin{aligned}\underset{\bar{\boldsymbol{x}}_{self},\bar{\boldsymbol{u}}_{self}}{\text{minimize}}\ & \textstyle\sum_{k=0}^{N-1} l_{self}(\boldsymbol{x}_{self}[k]-\hat{\boldsymbol{x}}_{other}[k],\boldsymbol{u}_{self}[k])\\ & + m_{self}(\boldsymbol{x}_{self}[k])\\ & + j_{self}(\boldsymbol{x}_{self}[N]-\hat{\boldsymbol{x}}_{other}[N])\\ \text{subject to}\ & \boldsymbol{x}_{self}[k+1] = f_{self}(\boldsymbol{x}_{self}[k],\boldsymbol{u}_{self}[k]) \qquad (3)\\ & \left.\begin{aligned}\boldsymbol{x}_{self}[k] &\in \mathcal{X}_{self}\\ \boldsymbol{u}_{self}[k] &\in \mathcal{U}_{self}\end{aligned}\right\} k=0,...,N\end{aligned}$$

where $\hat{\boldsymbol{x}}_{other}$ is the predicted state of the other vehicle and $(self, other) \in \{(h,b),(b,h)\}$ for the UAV and USV respectively. We define $m_h(\boldsymbol{x}_h[k]) = 0$, since we do not want the UAV to pursue any objectives other than landing.

3.3 Cascade Control Problem

The second new algorithm is called the *cascade distributed algorithm*, which, on top of the distributed architecture from the distributed algorithm uses a cascade controller for each problem (see Figure 4). To control the vehicle, it has both an outer controller generating a trajectory and an inner controller which computes the inputs to follow the planned trajectory. This algorithm requires the use of parallel processes to solve the outer and inner problems simultaneously. The horizontal and vertical outer problems are identical to (3) and (2) respectively. To allow for longer prediction horizons, the outer controllers in this algorithm update at a frequency lower than the sampling frequency. The inner controllers instead solve smaller problems at the sampling frequency. This algorithm allows for the use of longer prediction horizons in the outer controller, at the cost of using older measurements in the outer optimization. The inner optimization problems are given by

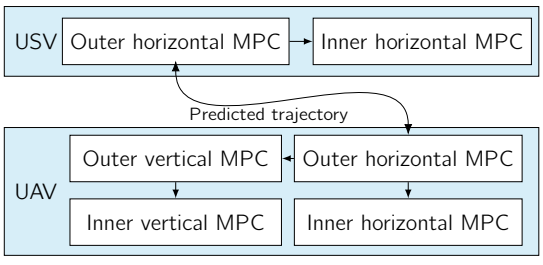


Fig. 4. In the cascade controller, all subsystems have an outer and an inner controller. The outer runs at a lower frequency and plans the trajectory, while the inner is responsible for trajectory following.

$$\begin{aligned}\underset{\bar{\boldsymbol{x}}_\sigma, \bar{\boldsymbol{u}}_\sigma}{\text{minimize}} \ & \sum_{k=0}^{N_{in}-1} l_{\sigma,in}(\boldsymbol{x}_\sigma[k]-\boldsymbol{x}_\sigma^{ref}[k], \boldsymbol{u}_\sigma[k]-\boldsymbol{u}_\sigma^{ref}[k]) \\ & + j_{\sigma,in}(\boldsymbol{x}[N_{in}] - \boldsymbol{x}^{ref}[N_{in}]) \\ \text{subject to } & \boldsymbol{x}_\sigma[k+1] = f_\sigma(\boldsymbol{x}_\sigma[k], \boldsymbol{u}_\sigma[k]) \qquad (4) \\ & k = 0, ..., N_{in} - 1.\end{aligned}$$

where $\sigma \in \{h, b, v\}$, such that all subsystems solve equivalent inner optimization problems. In all cases, $\boldsymbol{x}_\sigma^{ref}[k]$ and $\boldsymbol{u}_\sigma^{ref}[k]$ are the desired state and desired control input at time step k respectively, obtained from the outer problem. While the outer controllers aim to minimize the state difference between the vehicles, the inner controllers instead minimize the difference between the actual trajectories and the trajectories predicted by the outer controllers. To speed up solution times, these optimizations do not explicitly take all constraints into account. These constraints will instead be satisfied as long as the vehicles follow the planned trajectory closely enough.

3.4 Algorithms

1 **while** *Has not landed* **do**
2 Receive state $\hat{\boldsymbol{x}}_b$ from USV;
3 Solve centralized problem (1);
4 Compute $\bar{d}$ and $\bar{b}$ based on $\bar{\boldsymbol{x}}_h^*$ and $\bar{\boldsymbol{x}}_b^*$;
5 Solve vertical problem (2);
6 Apply control signals $\boldsymbol{u}_h^*[0]$ and $w_{cmd}^*[0]$;
7 Transfer $\boldsymbol{u}_b^*[0]$ to USV;
8 **end**

Algorithm 1: Centralized algorithm

The centralized algorithm is given by Algorithm 1. This algorithm runs on the UAV, and the USV applies the control inputs received from the UAV. The distributed algorithm, on the other hand, consists of two separate algorithms, where the first algorithm runs on the UAV and the second on the USV. A description of the first can be seen in Algorithm 2. The algorithm running on the USV is nearly identical, but with indices h and b reversed, the abbreviations *USV* and *UAV* exchanged, as well as lines 4 and 5 removed. Notable differences between these algorithms and the centralized algorithm are in lines 2 and 7, where different information is exchanged. It is worth noting that $\hat{\bar{\boldsymbol{x}}}_b$ is usually the most recent predicted trajectory from the other vehicle, but in the case that no new message arrives on time, the predicted trajectory is approximated by shifting the previous predicted trajectory by a time step. The formula for an n-step shift is given by

$$\begin{aligned}\boldsymbol{x}_{new}[k] &= \boldsymbol{x}_{old}[k+n], \quad \forall\ 0 \le k \le N-n \\ \boldsymbol{x}_{new}[k] &= \boldsymbol{x}_{old}[N], \quad \forall\ N-n < k \le N.\end{aligned}$$

The cascade distributed problem also consists of two algorithms, where the UAV algorithm is shown in Algorithm 3. The USV algorithm can be recreated by removing lines 8, 9 and 12. Notable changes between this algorithm and

1 **while** *Has not landed* **do**
2 Obtain most recent $\hat{\bar{\boldsymbol{x}}}_b$;
3 Solve distributed UAV problem (3);
4 Compute $\bar{d}$ and $\bar{b}$ based on $\bar{\boldsymbol{x}}_h^*$ and $\hat{\bar{\boldsymbol{x}}}_b$;
5 Solve vertical problem (2) ;
6 Apply control signals $\boldsymbol{u}_h^*[0]$ and $w_{cmd}^*[0]$;
7 Transfer $\bar{\boldsymbol{x}}_h^*$ to USV;
8 **end**

Algorithm 2: UAV distributed algorithm

Algorithm 2 are the if-statements starting at lines 4 and 10 in Algorithm 3. This means that every c iterations, a new solution to the outer problem should be solved in a parallel process, while the inner controller on the main process computes the control inputs that should be used. The outer problem has to be solved within c iterations. This limits how long the outer horizon can be since the computational time is dependent on the horizon. Because the outer problem takes c iterations to solve, the optimizer needs to choose the initial state as $\boldsymbol{x}_{h,in}[c]$, that is, the predicted state c steps into the future.

1 Set $i = 0$;
2 **while** *Has not landed* **do**
3 Obtain most recent $\hat{\bar{\boldsymbol{x}}}_b$;
4 **if** *modulo(i,c) is 0* **then**
5 Update $\bar{\boldsymbol{x}}_h^{ref}$ and $\bar{\boldsymbol{u}}_h^{ref}$;
6 Transfer $\bar{\boldsymbol{x}}_h^{ref}$ to USV;
7 **end**
8 Solve inner UAV and vertical problem (4);
9 Compute $\bar{d}$ and $\bar{b}$ based on $\bar{\boldsymbol{x}}_h^{ref}$ and $\hat{\bar{\boldsymbol{x}}}_b$;
10 **if** *modulo(i,c) is 0* **then**
11 Start solving outer UAV problem (3);
12 Start solving outer vertical problem (2);
13 **end**
14 Apply control $\boldsymbol{u}_{h,in}^*[0]$ and $w_{cmd,in}^*[0]$;
15 Increment i;
16 **end**

Algorithm 3: UAV cascade distributed algorithm

4. EXPERIMENTAL RESULTS

Two computers were used in the experiments. The first is the ground laptop, a Dell Latitude E6230 running Ubuntu 16.04. The second is the drone onboard computer, which is a NUC 7i7BNB, also running Ubuntu 16.04. This computer was concurrently running the low-level drone control. The machines communicated with each other using ROS (Robot Operating System) Kinetic Kane distribution, described in Quigley et al. (2009). To solve the optimization problems, the solver OSQP introduced by Stellato et al. (2017) was used through its Python-interface. The code is available on GitHub [1] . Two different kinds of tests were performed: tests of the computational performance that investigated how long prediction horizons would be feasible to use, and field experiments investigating the landing performance of the algorithms.

For testing the computational performance of the algorithms, HIL-simulations were performed where the simulated dynamics matched the modeled dynamics exactly. The centralized and the UAV algorithms were run on the UAV onboard computer, while the USV was simulated and controlled from the ground laptop. The goal was to run all algorithms with an (inner) update frequency of 20 Hz, so the maximum feasible horizon was chosen as the one where both the mean and median iteration times over 100 consecutive trials were below 0.05 s. The centralized and distributed algorithms could then achieve horizons of at most 193 and 244 time steps (9.65 s and 12.2 s) respectively, while the cascade distributed algorithm could have an outer horizon and inner horizon of 340 and 211 time steps (17 s and 10.55 s) respectively when using $c = 5$.

[1] `https://github.com/robbj-git/Distributed_Rendezvous`

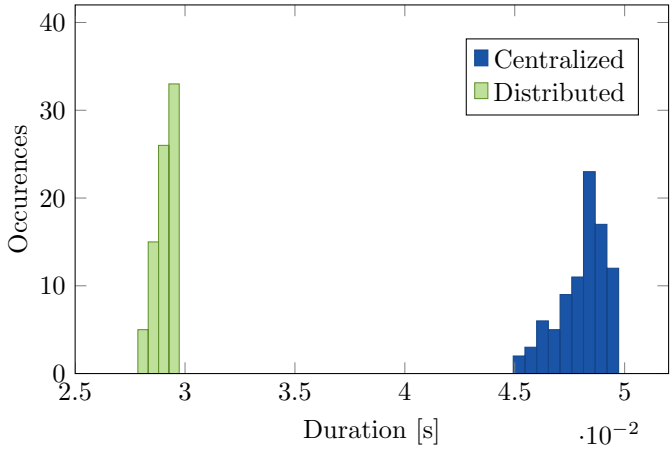


Fig. 5. Distribution of the iteration computational times for Algorithms 1 and 2, both with $N = 193$ time steps.

To better illustrate the performance difference between the centralized and distributed algorithm, 100 trials with a prediction horizon of 193 time steps were also performed with the distributed controller, and the distribution of iterations times for both it and the centralized algorithm is shown in Figure 5. The ability of the algorithms to successfully land the UAV was tested in flight experiments performed outdoors. To ensure that every iteration would finish within the desired time, a prediction horizon of 100 time steps (5 s) was used for the centralized and distributed algorithms. The cascade distributed algorithm used an outer horizon of 170 steps (8.5 s) and an inner horizon of 60 steps (3 s). The altitude of the UAV during a landing experiment, together with the horizontal distance between the vehicles and the altitude safety constraints, is shown for all three algorithms in Figure 6. All algorithms successfully performed the landing without violating any safety constraints. The parameters describing the safety region were $h_s = 4\,\text{m}$, $d_s = 2\,\text{m}$, and $d_l = 1\,\text{m}$. Note that the USV was in constant motion during these experiments, so the UAV was able to land while following the USV motion. All algorithms landed in a similar amount of time, with the cascade distributed algorithm taking a couple of seconds longer than the other algorithms. This delay was caused by the outer controller planning to slow down the descent to avoid violating constraints put on the vertical velocity. The outer controller could not recover from this slowdown as fast as the other algorithms due to its lower update rate.

To compare the ability of the algorithms to handle short communication losses, a set of experiments where the communication was broken for about 2.5 s were performed. The communication break was introduced around 4 s after the start of the landing, and during that time no messages were sent from either of the vehicles. Figure 7 shows the landing attempts of each algorithm in these experiments. Both distributed algorithms successfully satisfied the safety constraints at all times, which the centralized algorithm, on the other hand, could not. In the centralized case, once the USV stops receiving instructions from the UAV, it will stop applying control inputs, making the predictions made by the UAV inaccurate. Therefore, the UAV will continue moving, expecting the USV to move along with it, until it eventually fails to satisfy the safety constraints. In the distributed cases, once the UAV and USV stop receiving updated plans from each other, they still have old predictions to follow. By assuming that the other vehicle follows the last received plan, both vehicles can continue moving approximately as agreed, until new plans arrive.

5. FUTURE WORK AND CONCLUSIONS

Two distributed algorithms for autonomously solving a cooperative landing problem between a UAV and USV were designed and compared to a centralized MPC algorithm. The first designed algorithm used a distributed MPC approach, while the second combined distributed MPC with a cascade controller architecture. The computational performance of the algorithms was tested in a simulated environment, while the landing performance was tested in field experiments. In comparison to a centralized MPC algorithm, the distributed algorithms could achieve longer prediction horizons and improved robustness towards communication breaks. In particular, the cascade distributed algorithm could have significantly longer prediction horizons in its outer controller. All three algorithms showed similar performance in landing and satisfying safety constraints when no communication breaks were present.

An interesting consequence of using the distributed or cascaded controller is that computational resources that would otherwise have been used in solving the larger centralized problem are now freed. In this paper, the extra computational power has been put into increasing the prediction horizon of the MPC problems. However, there are many other ways that this extra computational power can be exploited, e.g. to have more advanced models of the vehicles in the optimization problem. By using nonlinear instead of linear models, the vehicle attitudes could be less constrained, allowing for better utilization of their mobility. Especially by using a more accurate model of the vertical motion, it could be possible to operate closer to constraint bounds and thus be able to descend faster.

REFERENCES

Alamir, M. and Bornard, G. (1995). Stability of a truncated infinite constrained receding horizon scheme: the general discrete nonlinear case. *Automatica*, 31(9), 1353 – 1356. doi:10.1016/0005-1098(95)00042-U.

Fossen, T. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley.

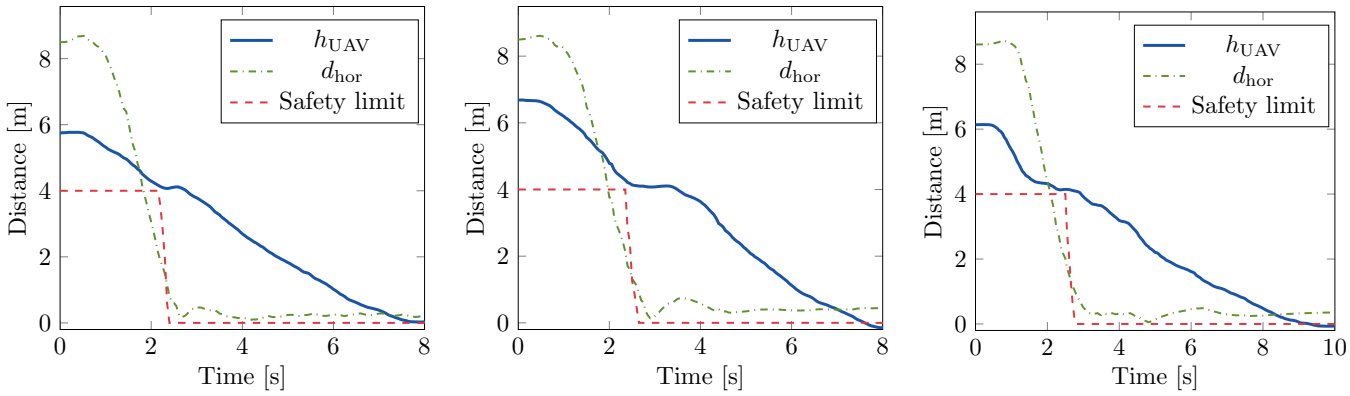
Herissé, B., Hamel, T., Mahony, R., and Russotto, F. (2012). Landing a VTOL unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on Robotics*, 28(1). doi:10.1109/TRO.2011.2163435.

Huck, S.M., Rueppel, M., Summers, T.H., and Lygeros, J. (2014). RCopterX - experimental validation of a distributed leader-follower MPC approach on a miniature helicopter test bed. In *2014 European Control Conference (ECC)*, 802–807. doi:10.1109/ECC.2014.6862458.

Keviczky, T., Borrelli, F., Fregene, K., Godbole, D., and Balas, G.J. (2008). Decentralized receding horizon control and coordination of autonomous vehicle formations. *IEEE Transactions on Control Systems Technology*, 16(1), 19–33. doi:10.1109/TCST.2007.903066.

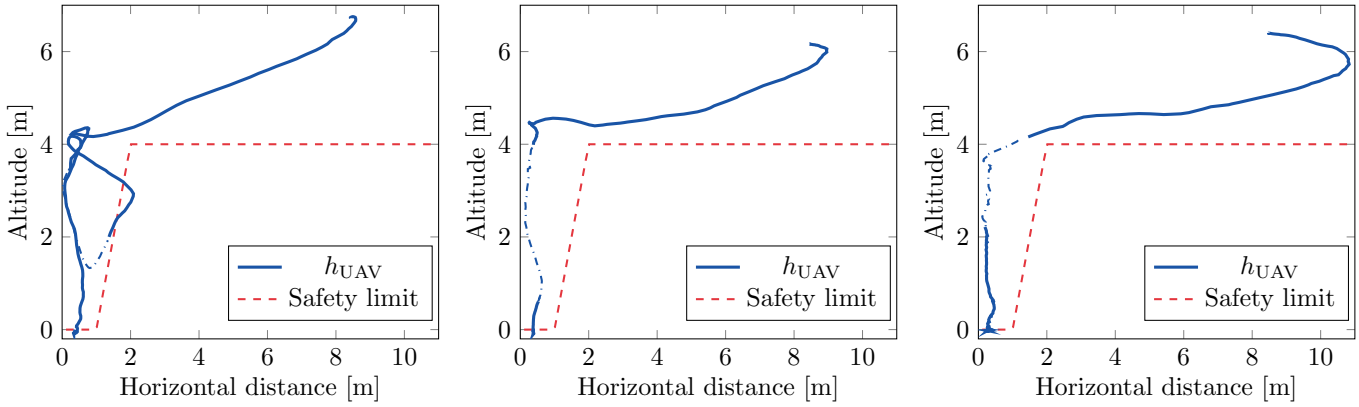
Kim, J., Woo, S., and Kim, J. (2017). Lidar-guided autonomous landing of an aerial vehicle on a ground vehicle. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 228–231. doi:10.1109/URAI.2017.7992719.

Lee, D., Ryan, T., and Kim, H.J. (2012). Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing. In *2012 IEEE International Conference on Robotics and Automation*, 971–976. doi:10.1109/ICRA.2012.6224828.



(a) Landing using Algorithm 1. (b) Landing using Algorithm 2. (c) Landing using Algorithm 3.

Fig. 6. Landing attempts using Algorithms 1–3 in an outdoor field experiment. All algorithms successfully avoided crossing the safety limit. A slight deceleration can be observed as a result of the vertical velocity constraints. The UAV altitude is denoted by h_{UAV}, and the distance (in the horizontal plane) between the vehicles is denoted d_{hor}.



(a) Landing using Algorithm 1. (b) Landing using Algorithm 2. (c) Landing using Algorithm 3.

Fig. 7. Landing attempts with a 2.5 s break in communication. The motion during the time of communication loss is represented by a dash-dotted line. For the centralized controller, the break in communication made the USV fall behind the UAV, and despite attempts to abort and go to a safe altitude, safety constraints were violated. The distributed algorithms satisfied the safety constraints throughout the landing.

Liu, Y., Montenbruck, J.M., Zelazo, D., Odelga, M., Rajappa, S., Bülthoff, H.H., Allgöwer, F., and Zell, A. (2018). A distributed control approach to formation balancing and maneuvering of multiple multirotor UAVs. *IEEE Transactions on Robotics*, 34(4), 870–882. doi: 10.1109/TRO.2018.2853606.

Müller, M.A. and Allgöwer, F. (2017). Economic and distributed model predictive control: Recent developments in optimization-based control. *SICE Journal of Control, Measurement, and System Integration*, 10(2), 39–52. doi: 10.9746/jcmsi.10.39.

Müller, M.A., Reble, M., and Allgöwer, F. (2012). Cooperative control of dynamically decoupled systems via distributed model predictive control. *International Journal of Robust and Nonlinear Control*, 22(12), 1376–1397. doi:10.1002/rnc.2826.

Muskardin, T., Balmer, G., Persson, L., Wlach, S., Laiacker, M., Ollero, A., and Kondak, K. (2017). A novel landing system to increase payload capacity and operational availability of high altitude long endurance UAVs. *Journal of Intelligent & Robotic Systems*, 88(2), 597–618. doi:10.1007/s10846-017-0475-z.

Negenborn, R.R. and Maestre, J.M. (2014). Distributed model predictive control: An overview and roadmap of future research opportunities. *IEEE Control Systems Magazine*, 34(4). doi:10.1109/MCS.2014.2320397.

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

Persson, L. (2019). *Autonomous and Cooperative Landings Using Model Predictive Control*. Licentiate thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.

Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. (2009). ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2017). OSQP: An operator splitting solver for quadratic programs. *ArXiv e-prints*.

Zou, Y., Zhou, Z., Dong, X., and Meng, Z. (2018). Distributed formation control for multiple vertical takeoff and landing UAVs with switching topologies. *IEEE/ASME Transactions on Mechatronics*, 23(4), 1750–1761. doi:10.1109/TMECH.2018.2844306.