2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Verification of Cooperative Maneuvers in FlightGear using MPC and Backwards Reachable Sets

Linnea Persson[1] and Bo Wahlberg[2]

Abstract— **In this paper we develop a simulation setup for testing and analyzing cooperative maneuvers and corresponding control algorithms. We also find feasible initial sets using backwards reachable set computations for the cooperative control problem, which we then test using the simulation setup. The particular example considered is a cooperative rendezvous between a fixed-wing unmanned aerial vehicle and a unmanned ground vehicle. The open-source software FlightGear and JSBSim are used for the vehicle dynamics, enabling testing of algorithms in a realistic environment. The aircraft models include nonlinear, state-dependent dynamics, making it possible to capture complex behaviors like stall and spin. Moreover, environmental effects such as wind gusts and turbulence are directly integrated into the simulations. From the simulations we get a comprehensive understanding of the controller performance and feasibility when tested in a real-time scenario. Results from several landing simulations are presented, and demonstrate that the MPC solution for the cooperative rendezvous problem is a promising method also for use in complex, safety-critical systems.**

I. Introduction

Following advances in wireless communication technologies, cooperative control is increasingly being applied to systems to allow separated agents to exchange information and operate together. Such systems have a wide range of application areas in e.g. aerospace, robotics and automotive industry, where collaboration can be utilized to achieve mission objectives that are not possible with a single agent. In this paper, we develop an experimental setup for simulation and testing of cooperative aerial systems.

Realistic simulations and corresponding visualizations are a vital part of the control design for any complex and/or safety critical system. It can be used to obtain a comprehensive overview of the performance of the system under the presence of modeling errors, communication delays, measurement errors and disturbances. Simulation can also be used for computing suitable control parameter values prior to testing the controller on a real system.

The cooperative maneuver simulation setup developed in this paper, uses FlightGear and JSBSim for the visualization and aircraft dynamical models, with simulations being monitored and controlled using Python. Figure 1 shows a snapshot of the visualization in FlightGear.

The specific cooperative maneuver considered in this paper consists of a fixed-wing Unmanned Aerial Vehicle (UAV) and a Unmanned Ground Vehicle (UGV) that cooperate to perform a landing of the UAV on top of the UGV. This maneuver is very safety-critical and requires high precision. The agents must coordinate their motions while moving at high speeds. Disturbances from wind, as well as the nonlinear dynamic, dependent on factors such as speed and angle of attack, adds even more complexity to the maneuver.

We have previously [1] for this maneuver suggested the use of Model Predictive Control (MPC) to generate optimal trajectories and control inputs that forces the system to stay within some safe sets . For this control system to be efficiently implemented on a real system, we need to consider from what states the maneuver is feasible within a certain amount of time. In this paper, the maneuver feasibility and time to rendezvous is estimated using backwards reachable sets. These sets are used in the controller to drive the system to a feasible state for the initiation of the maneuver.

The contributions of this paper include

- Development of an experimental setup using open-source software FlightGear, JSBSim and Python for simulation of cooperative maneuvers.
- Reachable set computations resulting in sets of states from which the maneuver can be initiated. This information assists in deciding whether or not to start the maneuver, and as a result maneuver cancellations can be avoided.
- The implementation of a MPC to cooperatively control a fixed-wing UAV and a UGV to rendezvous from estimated vehicle models.
- Realistic simulation tests with visualization of the UAV/UGV cooperative landing maneuver using the

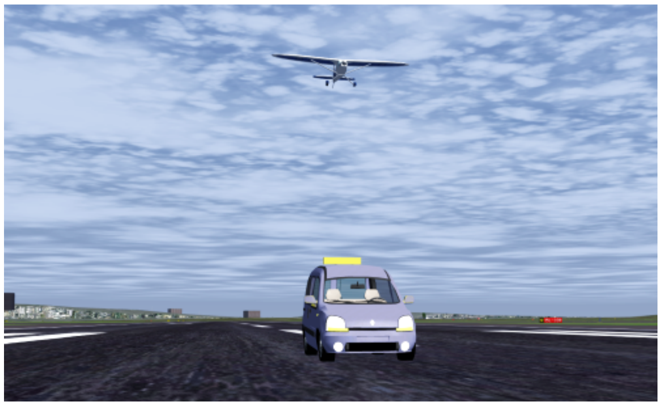
Fig. 1: A cooperative landing maneuver visualized in FlightGear, as a UAV is approaching a UGV for landing.

*This work was partially supported by the Wallenberg Autonomous Systems and Software Program (WASP).

The authors are with the Department of Automatic Control and ACCESS, School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden, [1]laperss@kth.se, [2]bo.wahlberg@ee.kth.se

MPC approach.

The code presented in this paper has been published online and can be used to further analyze this or other cooperative maneuvers. The code has been tested with several different vehicle models and types, and can be adapted to do similar experiments with e.g. multiple UAVs. All software that has been used is open-source and freely available online.

The structure of the paper is as follows. Previous and related work is described in Section II. In Section III, the notation and the setup of the problem is explained, including kinematic models and system constraints. This section also includes a summary of the MPC derived in [1]. The implementation, including the autopilot, the flight dynamics model and the visualization, is treated in Section IV. Simulations are demonstrated in Section V. Finally, Section VI presents conclusions further work.

II. Background

Several projects have in recent years evaluated or performed experiments on systems where a fixed-wing UAV cooperates with a vehicle on ground or in water [2], [3], [4]. One motivation behind doing an aerial/ground cooperative maneuver can be to extend the capabilities of the ground vehicle, as in the case of search and rescue missions, cooperative air/ground surveillance, or autonomous delivery of consumer products. Another reason for doing this could be to assist the aerial vehicle in e.g. an emergency landing or refueling.

In the experiments described in [2], landings were performed using a fixed-wing UAV and a semi-autonomous ground vehicle. The experiments resulted in several successful landings, proving feasibility of the maneuver using a PID controller in the lateral direction, and a safety-based hybrid controller for the vertical path. However, for the landing to be guaranteed safe and feasible, we need to keep the descent of the aerial vehicle controlled with respect to the distance between the vehicles. This means that the aerial vehicle should only descend if the future inputs that will be applied keeps it safe. This motivates the need for a model based approach that takes the system dynamics and constraints into account and computes safe trajectories for the vehicles to follow.

We have in recent work [1] suggested the use of a MPC in two parts to solve the UAV/UGV rendezvous problem. The paper derives a controller which forces the UAV to stay within some safe sets, and demonstrates how the controller offers robustness towards disturbances. The use of MPC to solve finite maneuver problems such as this has several advantages. MPC takes physical limitations of the system, as well as constraints imposed by the environment, into consideration already in the computation of the control inputs. Long-term control effects are incorporated into the computations through the prediction of the state dynamics. The prediction of the system state evolution results in an estimate of the future trajectory. Terminal constraints can be imposed on the system to always assure that the rendezvous point is within the horizon of the optimization. Imposing requirements on the terminal state is one of many methods for dealing with stability in MPC [5]. For a finite maneuver, a natural choice of terminal set is the target set of the maneuver. Many MPC stability proofs depend on the terminal state being control invariant [6], but methods exists also for maneuvers when this is not the case [7]. When dealing with finite maneuvers, and as such with finite horizon optimization problems, the feasibility has to be considered, since unfeasible states lead to undefined control inputs. The vehicles have to belong to a certain initial set for the optimization to be feasible. The set from which an optimization problem is feasible within a certain number of time steps can be computed using backwards reachable sets [8]. We use results from backwards reachable sets to compute feasible initial regions for the maneuver, which are then used in the simulations to estimate when the maneuver should be initiated.

MPC design is for a real system a relatively challenging task, requiring both the estimation of system dynamic models and the tuning of optimization parameters. As such, the control design process benefits from verification using realistic simulations. This paper uses FlightGear[1] for simulation, with JSBSim[2] as the underlying flight dynamics model (FDM). FlightGear is an open-source flight simulator, used in academic and industrial settings, as well as by hobbyists and for pilot training. In academia, FlightGear is used in a variety of applications including hardware-in-the-loop simulations and development of aircraft and flight control systems. JSBSim is a standalone, open-source flight-dynamics model, which defines and computes the movement of the aircraft. JSBSim includes configurable flight control systems, aerodynamics, and actuator definitions using XML-based text format [9].

III. Problem statement and proposed controller

This section introduces the example used for the simulations in this paper. First, the goal of the simulations is stated. Then, kinematic constraints, actuator constraints, and safety constraints are defined. Finally the proposed controller is given.

A. Objective

The cooperative maneuver that we consider is the rendezvous between a UAV and a UGV. The vehicles cooperate to compute a sequence of control actions to:

- Align the vehicles in the x-y-plane,
- Make the UAV descend to land safely on top of the UGV in a finite distance

In addition to these hard objectives, it is preferable if:

- The UAV stays at a high altitude until the landing is feasible,
- The landing is completed as fast as possible.

[1]http://www.flightgear.org/
[2]http://jsbsim.sourceforge.net/

B. Vehicle models

For the purpose of the control design, the the agents are modeled using linear difference equations, where T is the time step. The state of the UAV at time t is given by:

$$\begin{aligned} x_t^{UAV} &= [x_a(t), y_a(t), h_a(t), v_a(t), a_a(t), \gamma_a(t), \psi_a(t)]^T \\ u_t^{UAV} &= [a_r(t),\ \psi_r(t),\ \gamma_r(t)]^T, \end{aligned} \tag{1}$$

with x and y denoting the position forward and sideways, h the altitude, v velocity, a acceleration, γ flight path angle and ψ heading angle respectively. Similarly, UGV is represented by

$$\begin{aligned} x_t^{UGV} &= [x_a(t),\ y_a(t),\ v_a(t),\ a_a(t),\ \psi_a(t)]^T \\ u_t^{UGV} &= [a_r(t),\ \psi_r(t)]^T. \end{aligned} \tag{2}$$

The state of the vehicle k time steps in the future is written x_{t+k}. For each agent $i \in \{UAV, UGV\}$, the difference equation governing the vehicle motion is a function of the previous states and inputs

$$x_{t+1}^i = f^i(x_t^i, x_{t-1}^i, \ldots, x_{t-k}^i, u_t^i, \ldots, u_{t-l}^i).$$

C. Safety constraints

In order to safely land the UAV on top of the UGV, we require that some additional constraints are fulfilled. The complete derivation of the constraints can be found in [1]. The constraints are summarized as

- The landing should be completed with the UAV approaching the UGV approximately directly from above, as illustrated in Figure 2. This constraint is expressed as a logic constraint, making the system Mixed Logic Dynamical [10] system, formally expressed as

$$A \begin{bmatrix} d(i) \\ h(i) \end{bmatrix} \geq d - M \begin{bmatrix} b_1(i) \\ b_2(i) \end{bmatrix} \tag{3}$$

$$b_1(i) + b_2(i) \leq 1 \tag{4}$$

  where $A \in \mathbb{R}^{2\times 2}$, $d \in \mathbb{R}^2$ and $b_1, b_2 \in \{0,1\}^N$ are binary decision variables.
- The vertical component of the UAV velocity at landing should not be too large at touchdown, meaning that the flight path angle should be above a certain value γ_{td} at $h_a(t) = 0$, where $\gamma_{td} < 0$. This constraint is expanded to

$$\gamma_a(t) \geq \frac{\gamma_a^{min} - \gamma_{td}}{h_f} h_a(t) + \gamma_{td}, \quad \text{if } h_a(t) < h_f, \tag{5}$$

  where h_f is some altitude where the UAV should start to increase the lower limit of the constraint on γ_a.

D. Proposed controller

The controller is separated into two separate model predictive controllers, as is illustrated in Figure 3. This structure significantly simplifies the optimization, as two smaller convex problems are solved instead of one large, nonconvex problem [1]. The first MPC computes the lateral paths for both vehicles. The altitude and flight path angle are ignored at this stage. The objective of this first stage is to align the vehicles in the xy-plane to make the landing feasible. This MPC results in an optimal lateral trajectory $X_\star^{lat}$ and corresponding inputs $U_\star^{lat}$.

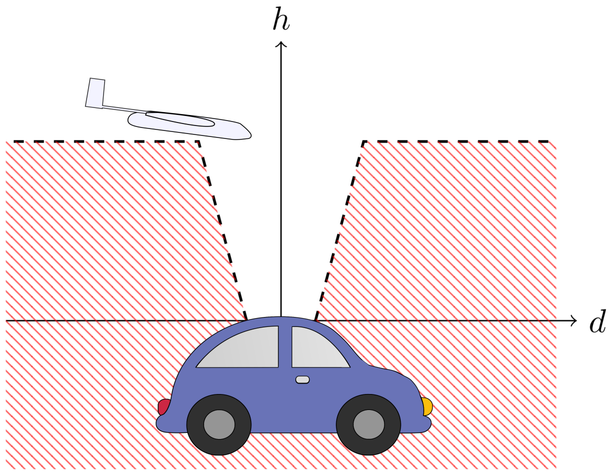

Fig. 2: The UAV should not enter the danger zone during landing. It must either stay above the safe altitude or approach from within a safe descent zone.

The second MPC computes a vertical trajectory for the UAV. The binary variables b_1 and b_2 from the safety constraint (4) can now be precomputed from the future lateral trajectory $X_\star^{lat}$ as

$$\left.\begin{aligned} d(i) \geq d_s &\implies b_1(i) = 0 \\ d(i) \leq d_s &\implies b_2(i) = 0 \end{aligned}\right\} \text{ for } \ i \in \{1, \ldots, N\}.$$

where $d(i)$ is the distance at time i and d_s denotes the safe landing distance. The second MPC results in an optimal vertical trajectory and a reference input for the flight path angle γ_r.

IV. Implementation

This section explains the implementation of the simulation and visualization in detail, starting by describing the structure of the implementation. After this, the autopilots and MPC are derived. Finally, the feasibility of the maneuver and the MPC is considered by looking at reachable sets.

A. Simulation Structure

There are two simultaneous instances of FlightGear and JSBSim running during the landing simulation (see Figure 3). The communication between Python and JSBSim/FlightGear is done over telnet for individual and occasional commands, and over TCP for continuous commands such as the reference trajectories.

The two vehicles are connected using the FlightGear MultiPlay mode, which makes it possible for the vehicles to interact and be seen in the same visualization window. The two instances communicate over TCP at a rate of 180 Hz. Still, there is a delay in the exchange of positions, causing a

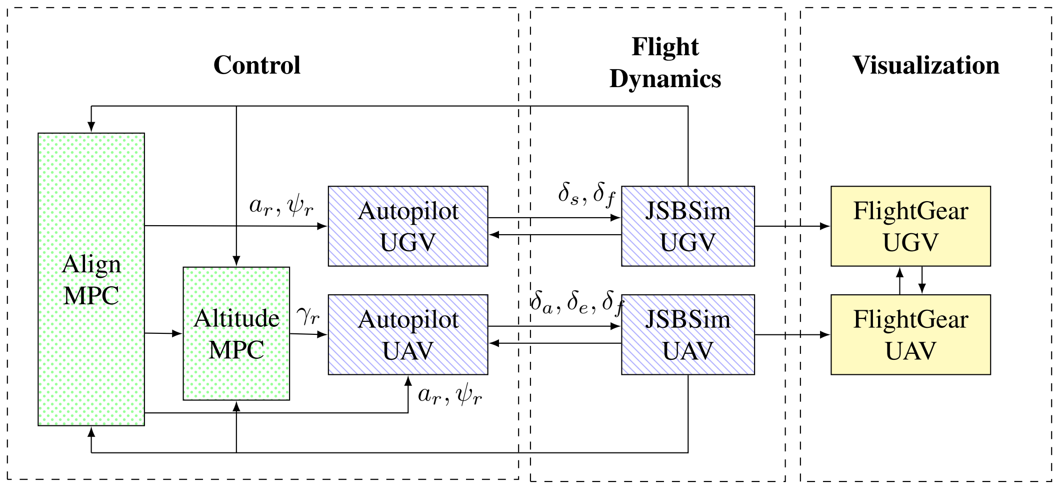

Fig. 3: The structure of the implementation. The green blocks are implemented using Python and CVXPY [11], the blue blocks are implemented using JSBSim. The two yellow blocks are the FlightGear multiplay instances.

lag in the visualization. To counteract this, there is a patch for predicting the movement of the other vehicle based on its velocities and accelerations. This corrects the problem to a large degree. After this correction, there still remains small fluctuations in the data stream from JSBSim to Python. To counteract this, the data has been time-stamped, and is corrected for before the data is sent to the controllers.

B. Autopilot

The autopilots are written in the JSBSim XML-based file format. Each loop is a simple, cascaded PID controller. The UAV autopilot take as input:

- Velocity or Acceleration,
- Altitude or Flight Path Angle,
- Heading or Wings-Level,

and from this computes the elevator, aileron, and throttle commands, denoted δ_e, δ_a and δ_t respectively. The acceleration a and the flight path angle γ of the aerial vehicle is controlled using a Total Energy Control System (TECS) [12]. With TECS, the use of throttle and elevator are coordinated by taking the total kinetic energy required and the desired energy distribution into account. This has the advantage that the autopilot more precisely can control velocity and flight path simultaneously. TECS computes a pitch reference θ_r and a thrust reference T_r, from which the elevator and throttle commands are computed. In parallel, the UAV has a control system for following the reference heading ψ_r by computing aileron commands.

The UGV control system takes as input:

- Velocity or Acceleration,
- Heading,

and computes a throttle command δ_f and a steering command δ_s. s Both autopilots have been tuned experimentally to give satisfiable responses.

C. Model predictive control

At each sampling time, the MPC computes a rendezvous trajectory for each vehicle based on the current states and inputs. The model predictive controller has here been implemented using Python and CVXPY [11]. An alternative, nonlinear, controller was implemented for the alignment MPC, using the ACADO Code Generation tool [13]. The MPC is called every Δt seconds, and in case no solutions are found (e.g. because the states are outside of the feasible set), the vehicles would be sent commands to cancel the maneuver and reach a safe state.

D. Backwards reachable sets

To decide if the maneuver is feasible, and if not, from where it will be feasible, we precompute feasibility sets for the maneuver. This is important since unfeasible states lead to undefined control inputs when using MPC, and also because we want to avoid attempting to land at times when it is not possible.

When computing the reachability sets, we only consider the lateral reachability. This is done under the assumption that the lateral and the vertical movement of the UAV are independent. This is approximately true for small vertical movements. The constraints of the reachability sets are the same as those of the alignment MPC. Given the control invariant terminal set $\mathcal{X}_e$, we can iteratively compute feasible sets by propagating the system backwards. A recursive formula for this is

$$\begin{aligned} &\mathcal{X}_0 = \mathcal{X}_f \\ &\mathcal{X}_i = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} \text{ s.t. } Ax + Bu \in \mathcal{X}_{i-1}\} \end{aligned} \quad (6)$$

Every such set is expressed as a polytope on the form

$$\mathcal{X}_i = \{x \in \mathbb{R}^n \mid \alpha_i x - \beta_i < 0\} \quad (7)$$

where $\alpha_i \in \mathbb{R}^{n \times n}$ and $\beta_i \in \mathbb{R}^n$. The reachable sets are computed using the MPT3 Toolbox [14].

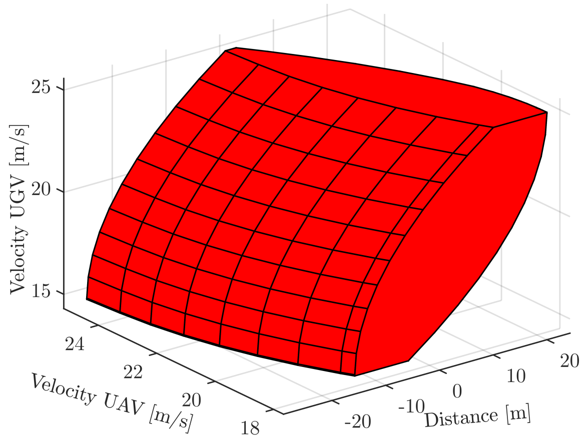

Fig. 4: Backwards reachable set for 5 seconds. The set has been projected to show only the d, v^{UAV} and v^{UGV} dimensions. For the landing to occur within 5 seconds, the vehicles should not be more than approximately 25 meters apart.

The optimization problem is feasible if the system state lies within one of these sets (7) for $i \leq N$. In the simulations, this is captured with the switching condition

$$\begin{array}{ll} \text{if } \alpha_N x < \beta_N : & \text{Solve MPC} \\ \text{else:} & \text{Reach } \mathcal{X}_N \end{array}$$

Any standard control method could be used to reach the final maneuver initiation set. In this implementation, a PD controller is used.

An example of a reachable set projected on three dimensions is shown in Figure 4. In this example, the set is computed for a horizon of 5 seconds. The state from which the optimization can be solved depends on the distance between the vehicles, but also on the velocities that the vehicles have.

V. Simulations

Simulations have been performed on a Intel Core i7 CPU 3.40 GHz×8 computer running Ubuntu 16.04, using FlightGear 2017.2.1 and Python 3.6. The code is publicly available on GitHub[3]. The sampling rate of the MPC is 5.5 Hz.

A. Evaluation of the reachability computations

The objective of the reachability computations was to approximate from where a certain terminal set could be reached, and thus where the final stage of the landing should be initiated. Figure 5 shows the evolution of the system after it has entered the backwards reachable set. Under perfect conditions, the system should reach the terminal set within 5 seconds. The figure shows that this is approximately true. The difference is likely due to wind disturbances and approximations made in the model dynamics.

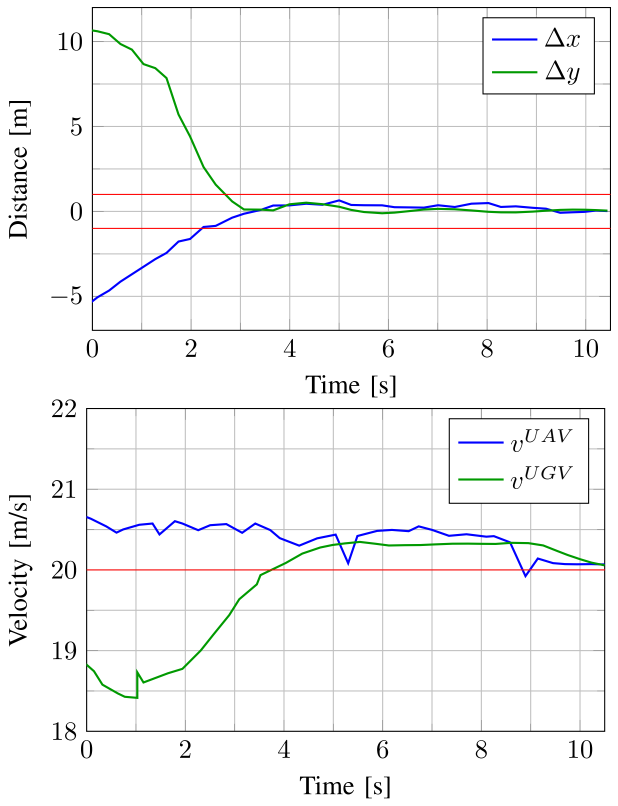

Fig. 5: The evolution of the system under slightly windy conditions after reaching the "reachability set" corresponding to −5 seconds. This implies that the system should converge to the terminal set within 5 seconds. The distance should stay within the range indicated by red lines in the topmost picture, and the velocities should have reached 20 m/s. This is approximately achieved.

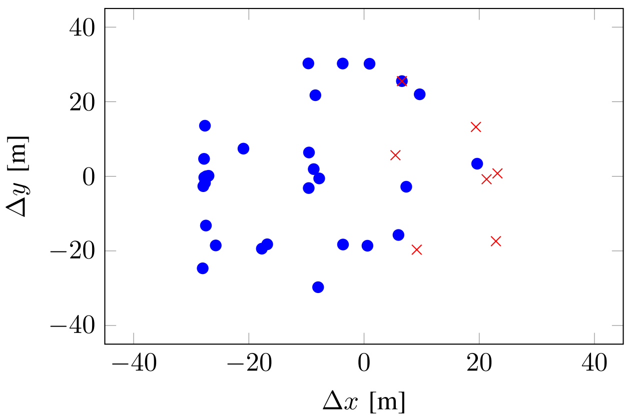

Fig. 6: The relative distance between the vehicles at the initiation of the maneuver. The blue dots show the positions where the landing succeeded while satisfying the safety constraints, and the red crosses are the cases when the safety constraints were broken at some point during the landing. The initial difference in altitude was 20 m for all simulations, but the initial velocities and accelerations are different.

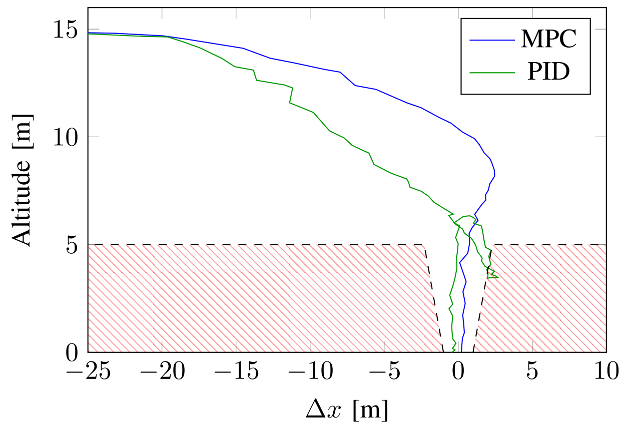

Fig. 7: Landing profiles comparison for two controllers starting from the same initial values, under mild wind conditions. The blue curve shows the MPC described in Section IV-C, and the green curve shows a simple PD controller.

Figure 6 illustrates how the landing success dependeds on the initial conditions in 37 different simulations. The simulations were run with a nonlinear MPC in the lateral direction and the vertical MPC from Section III. The simulations all have different initial velocities and accelerations, as well as initial relative distance, but only the relative distance is shown in the figure. The figure shows that not all of the landings succeeded on the first try, even though all initial positions are within the feasible set. This likely due to the simplifications made in the reachability computations 7, where the flight path angle is not into account. As a result, when the UAV starts too far in front of the UGV, the safety constraints (Eq. 3–4) is not guaranteed to be satisfiable for all future iterations.

B. Comparison between MPC and PID approach

For comparison, a simple PD controller was implemented for the same maneuver. The PD controller separately controls Δx and Δy to reach 0, and has a switching law in the h direction for avoiding the danger zone. Both controllers have the same constraints on input and state values. Figure 7 shows a comparison between the landing outcome when using the MPC strategy presented in this paper and the PD controller. The vehicles were disturbed by mild wind disturbances. The simulation shows that the MPC is able to predict the overshoot, and adapts its descent angle to this information. This way it avoids reaching the danger zone, and the maneuver is never canceled.

VI. Conclusions

In this paper we have developed an experimental setup for verification of cooperative maneuvers. The setup, with flight dynamics models from FlightGear and JSBSim, offers a straightforward way to perform additional tests on cooperative maneuvers, with the possibility to test in a variety of different conditions. The simulation setup can be used to analyze how disturbances, delays in data transfer, inaccuracies in the assumed model dynamics etc. affects the overall system, something that is difficult to do analytically for a complex system.

For the future, there are plans on testing the described cooperative landing algorithm out on a real system. The setup provided in this paper can then be used both for verification and future development.

The model predictive controller which was implemented and tested provided us with a better understanding of the control system and its performance limitations. The simulations showed that the method developed in [1] can successfully be used on a realistic system under the influence of disturbances, delays and unsynchronized data. Precomputed backwards reachable sets assisted in deciding when to initiate the maneuver.

Although the backwards reachable sets were reliable for a majority of the initial conditions, they overestimated the reachability in the forward direction. This knowledge will be taken into account when further developing the system. If we want the reachability analysis to be valid also from initial conditions where the UAV starts in front of the UGV, the backwards reachable sets need to be computed with respect to the altitude and the vertical dynamics as well. This would likely also improve the time to rendezvous for the maneuver since the UAV would could have a lower altitude when starting the final stage of the maneuver.

[3]https://github.com/laperss/fg-cc-sim/

References

[1] L. Persson, T. Muskardin, and B. Wahlberg, “Cooperative rendezvous of ground vehicle and aerial vehicle using model predictive control,” in *2017 IEEE Conference on Decision and Control (CDC)*, December 2017.

[2] T. Muskardin, G. Balmer, L. Persson, S. Wlach, M. Laiacker, A. Ollero, and K. Kondak, “A novel landing system to increase payload capacity and operational availability of high altitude long endurance UAVs,” *Journal of Intelligent & Robotic Systems*, pp. 1–22, 2017.

[3] P. Wu, M. Voskuijl, M. J. L. van Tooren, and L. L. M. Veldhuis, “Take-off and landing using ground-based power-simulation of critical landing load cases using multibody dynamics,” *Journal of Aerospace Engineering*, vol. 29, no. 3, 2015.

[4] K. Klausen, J. B. Moe, J. C. van den Hoorn, A. Gomola, T. I. Fossen, and T. A. Johansen, “Coordinated control concept for recovery of a fixed-wing UAV on a ship using a net carried by multirotor UAVs,” in *International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016.

[5] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, “Constrained model predictive control: Stability and optimality,” *Automatica*, vol. 36, no. 6, pp. 789 – 814, 2000.

[6] F. Blanchini, “Set invariance in control,” *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[7] A. Richards and J. P. How, “Robust variable horizon model predictive control for vehicle maneuvering,” *International Journal of Robust and Nonlinear Control*, vol. 16, no. 7, pp. 333–351, 2006.

[8] F. Borrelli, A. Bemporad, and M. Morari, “Predictive control for linear and hybrid systems,” *Cambridge, 2011*, 2011.

[9] J. S. Berndt *et al.*, *JSBSim - An open source, platform-independent, flight dynamics model in C++*, June 2011.

[10] A. Bemporad and M. Morari, “Control of systems integrating logic, dynamics, and constraints,” *Automatica*, vol. 35, pp. 407–427, Mar. 1999.

[11] S. Diamond and S. Boyd, “CVXPY: A Python-embedded modeling language for convex optimization,” *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[12] A. A. Lambregts, “Vertical flight path and speed control autopilot design using total energy principles,” *Guidance and Control Conference*, 1983.

[13] H. Ferreau, T. Kraus, M. Vukov, W. Saeys, and M. Diehl, “High-speed moving horizon estimation based on automatic code generation,” in *Proceedings of the 51th IEEE Conference on Decision and Control (CDC 2012)*, 2012.

[14] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, “Multi-Parametric Toolbox 3.0,” in *Proc. of the European Control Conference*, (Zürich, Switzerland), pp. 502–510, July 17–19 2013. http://control.ee.ethz.ch/~mpt.