

Fast Convex Model Predictive Trajectory Planning for Robot-Assisted Landings

Jannes Terlau

terlau@fh-aachen.de

FH Aachen

Vincent Konnow

FH Aachen

Philipp Hartmann

FH Aachen

Research Article

Keywords: Unmanned Aerial Vehicle, Model Predictive Control, Robot-Assisted Landing

Posted Date: May 19th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9428133/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at CEAS Aeronautical Journal on September 15th, 2026. See the published version at <https://doi.org/10.1007/s13272-026-01007-4>.

Fast Convex Model Predictive Trajectory Planning for Robot-Assisted Landings

Jannes Terlau^{1*}, Vincent Konnow¹ and Philipp Hartmann¹

^{1*}Faculty of Aerospace und Automotive Engineering, FH Aachen – University of Applied Sciences, Hohenstaufenallee, Aachen, 52064, North Rhine-Westphalia, Germany.

*Corresponding author(s). E-mail(s): terlau@fh-aachen.de;
Contributing authors: konnow@fh-aachen.de; p.hartmann@fh-aachen.de;

Abstract

Robot-assisted landings for aircrafts which are able to take off and land vertically offer the capability to speed up and automate landing and turnaround processes and are particularly advantageous for challenging landing conditions such as wind gusts or a moving landing platform. To perform an assisted landing, a precise and robust control approach is required, covering movement synchronization, catching and decelerating. In this paper a novel approach on model predictive trajectory planning for robot-assisted landings of unmanned aerial vehicles is designed, implemented and analyzed. The approach enables an unmanned aerial vehicle to be caught in hovering flight state using a 6-axis robotic arm with a positive-locking gripper. The trajectory planning utilizes a simplified robot model assuming an approximately linear mapping between Cartesian velocities and joint velocities, justified by a limited robot workspace. The aircraft's flight trajectory is predicted using a Kalman filter to overcome the robot's delay behavior. Catching tolerances between aircraft and gripper result in slackness for the synchronization process and are exploited during the trajectory planning. The suitability of the approach presented is analyzed and discussed in simulations, laboratory tests and several flight tests.

Keywords: Unmanned Aerial Vehicle, Model Predictive Control, Robot-Assisted Landing

Nomenclature

a	Translational acceleration	t	Time
a_x, a_y, a_z	Semi-major axes	T_r	Task space
d_{ac}	z -position offset	T_g	Workspace
f_{fg}	Command value filter	u	Input vector
J	Cost function	V	Variances of position estimation
N_c	Control horizon	x	State vector
N_d	Delay horizon	x, y, z	x -, y - and z -position
N_p	Prediction horizon	z	Optimization variable vector
p	Position	ψ, θ, ϕ	Yaw, pitch and roll angle
P	Pose	ω	Rotational velocity
q	Joint angle vector	$\square_{ac}$	Value $\square$ of the aircraft
Q	State weighting matrix	$\square_r$	Value $\square$ of the gripper
r_d	Scaling factor	b	Robot's base coordinate system
R	Input weighting matrix	f	Aircraft-fixed coordinate system
s	Decision variable	p	Optical measurement coordinate system
S	Decision criterion	UAV	Unmanned aerial vehicle
		QP	Quadratic problem
		SOCP	Second-order cone problem

1 Introduction

The approach of a robot-assisted landing for unmanned aerial vehicles (UAVs) which are able to take off and land vertically allows to speed up and automate landing and turnaround processes while reducing the required landing area [1, p. 2]. Especially for challenging landing conditions such as wind gusts or a moving landing platform like a ship deck, a robot-assisted landing is advantageous. To ensure safe landing while meeting catching tolerances and compensating disturbances, a precise and robust control approach is required. Previous work in the field of robot-assisted landings of UAVs has already investigated different approaches. In 2016, a landing platform as part of a DHL parcel station was developed [2]. The UAV could be moved on the parcel station’s landing platform with a sliding mechanism to replace the battery and payloads. The necessary landing area of such an UAV with no active catching mechanism was identified in [2] as $3\text{ m} \times 3\text{ m}$. In 2022, the company Quantum-Systems GmbH started developing a ground platform which is conceptually similar to the one used here. Quantum-Systems’ approach uses a tilt-rotor UAV and a robotic arm [3]. In 2025, the company Skydio, Inc. announced and demonstrated their product F10 which comprises a robotic arm for takeoff and landing [4]. For both Quantum-Systems’ and Skydio’s products, no technical details regarding the robot-assisted landing process are publicly available. In [5–8], Maier et al. investigated the capture of a hovering UAV using a robotic arm, with a particular focus on landing on ships under rough sea conditions. In contrast to a free capture process, the UAV was attached to a ball bearing. Therefore, only the last phase of the robot-assisted landing process was examined, in which a coordinated deceleration of the UAV and the robotic arm took place. The actual capture process of the UAV was not considered. In [9, 10], Xu et al. presented an unmanned surface vehicle (USV) equipped with a robotic manipulator built for handling UAVs in maritime domains. Although it employs a similar sensing and control strategy, a different aircraft and catching technique were used.

Within this paper, we propose an approach which employs a convex model predictive trajectory planner to catch a UAV with a positive-locking gripper mounted on a 6-axis robotic arm (see Fig. 1). The robotic arm synchronizes its motion with that of the UAV during the landing process as well as catches and decelerates the UAV, and additionally enables the alignment of the UAV against the direction of the wind for its take-off process. The robot’s workspace has been restricted intentionally, so that an approximately linear mapping between the gripper’s Cartesian velocities of and the robot’s joint velocities could be shown in previous work



Fig. 1: Flight test impression on the utilized ground platform

[1, p. 8]. This finding enables the utilization of a simplified robot model within the trajectory planner, while applying the robot’s computationally expensive inverse kinematics after optimization. The requirement of synchronization between the UAV’s flight trajectory and the robot gripper’s motion is defined as a quadratic cost function. To obtain a continual synchronization as well as to compensate for the robot’s delay and lag behavior, a prediction of the UAV’s flight trajectory is obtained, applying a Kalman filter. To ensure that the catching process can be accomplished within the robot’s workspace, the resulting optimization problem has to be constrained with respect to feasible gripper positions. To address this requirement, we evaluate a second-order cone program formulation, capable of considering non-linear constraints, against a quadratic program with linear constraints and a dedicated command value filter, for comparison. The convex formulation of both programs enables solving the underlying optimization problem deterministically. To model the entire landing sequence and to include catching tolerances between UAV and gripper, we use a finite state machine. The proposed approach is analyzed and discussed in simulations and laboratory tests as well as several field tests, physically catching the UAV.

The remainder of this paper is structured as follows: Section 2 provides a general description of the approach and prior investigations used to design the controller. Subsequently, the proposed trajectory planner is outlined in detail in Section 3. In Section 4 results based on both simulations and actual field tests are presented. A conclusion and outlook is given in Section 5.

2 Approach

2.1 Robot-Assisted Landing

The process of the robot-assisted landing will comprise three phases. Fig. 2 visualizes the characteristics of each phase. Starting with the position synchronization of gripper and UAV, the gripper will be located at the task space’s boundary to minimize the distance to the UAV. As soon as the UAV approaches, the gripper maintains a preset constant vertical distance to the aircraft. After the UAV has entered the task space and a sufficient synchronization was ensured, the catching process is triggered. To accomplish a successful catch, the gripper has to reduce the vertical offset while monitoring corresponding catch conditions. Once the vertical distance between gripper and UAV has been reduced and the gripper caught the UAV, a deceleration phase is invoked. The UAV-gripper system has to decelerate within adjustable constraints concerning accelerations or induced forces while not violating the robot’s workspace boundary.

2.2 System Overview

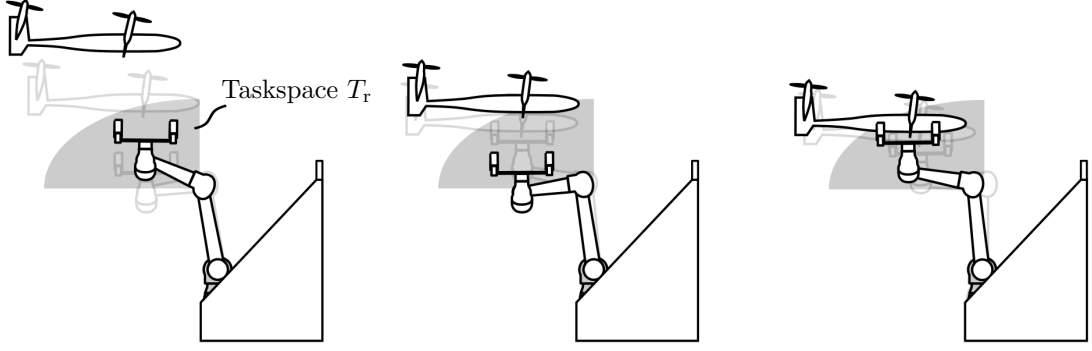
To command a set of robot joint angles such that gripper and UAV are sufficiently synchronized in each timestep, the model predictive trajectory planner, in the following also denoted as controller, requires information on the UAV’s flight trajectory. A sufficient synchronization will be evaluated based on catching tolerances between gripper and UAV, identified in a previous work [1, p. 9]. Fig. 3 depicts the ground platform control system. The controller maps an estimation of the UAV’s pose (position and attitude) $P_{ac} \in \mathbb{R}^{6 \times 1}$ to a set of robot joint angles $q_r \in \mathbb{R}^{6 \times 1}$, also denoted as robot configuration. The estimation of the UAV’s pose is obtained by a decentralized Kalman filter based on the measurements of a ground-based optical tracking system P_{ac}^p and on-board measurements of the UAV’s inertial measurement unit (IMU). The IMU provides measurements on translational accelerations a_{ac}^f and rotational velocities ω_{ac}^f , which are transferred in immediate vicinity to the ground platform via a low-latency telemetry link (as described in [1]). The Kalman filter can be utilized at a timestep k to predict the UAV’s flight trajectory for N_p timesteps, using a model of the UAV’s kinematics. N_p corresponds to the prediction horizon. The commanded joint angles, calculated by the controller, are sent to the robot, which executes them into an end effector movement $P_r(t)$ accordingly. To communicate with the robot’s controller an interface was created in a previous work [1, pp. 7-8], allowing direct setpoint inputs to the axes’ controller with a fixed

sample rate of 250 Hz. Together with the robot’s controller, the robot is position controlled. Furthermore, the robot is characterized by a dead time of approximately 40 ms and further behavior of a second order delay element in each axis. In [1, pp. 11-13], it was shown that a delay behavior like this impedes a sufficient synchronization for the catching process. The delay results in an unacceptable position deviation between UAV and gripper with respect to the given catch tolerances. Therefore, the prediction of the UAV’s flight trajectory is needed to overcome the robot’s inherent delay. To utilize the UAV’s predicted flight trajectory and concurrently consider constraints on the robot’s workspace and dynamic behavior, we employ model predictive control (MPC).

2.3 Model Predictive Control

Generally, MPC comprises a model of the controlled system to predict future states, the comparison of these predicted states with a reference trajectory and the conclusion for the current value of the input variable [11, pp. 1-3]. The term MPC describes a variety of different methods that use the system’s model to infer an input value for a system by optimizing a cost function. With that, constraints on state or input variables can be incorporated in the control method while performing the optimization in each time step. The main task when designing an MPC is choosing an appropriate type and complexity for the corresponding components, such as the system’s model, the cost function, the state or input variable constraints as well as the optimization problem formulation. For that, the available computational power often limits the optimization problem’s complexity.

Utilizing a MPC for a robot-assisted landing is advantageous for several reasons. Firstly, the robot is kinematically constrained, resulting in a feasible task space. Thus, the robot’s motion limits have to be considered while performing the trajectory planning. MPC is able to take constraints into account during the optimization. Secondly, with the predicted reference trajectory, the robot’s delay characteristics can be respected by extending the prediction horizon N_p and introducing a control horizon N_c . The control horizon is a subset of N_p and begins at the timestep $k + N_d$, with the robot’s delay N_d and $N_p = N_d + N_c$. The timestep k denotes the current timestep. The optimization is performed within the control horizon N_c . Thus, by shifting the control horizon, the robot’s delay N_d can be considered during control.



1. Synchronization

2. Catch

3. Deceleration

Fig. 2: Landing sequence

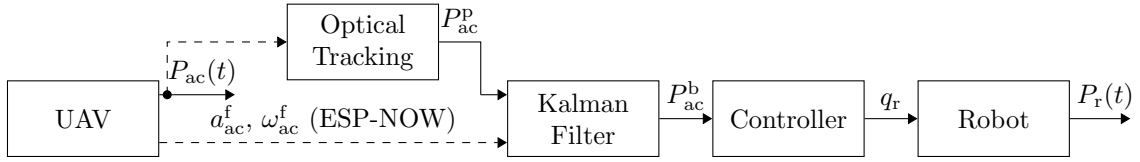


Fig. 3: Ground platform control system

2.4 Pre-Design Investigations

For a robot-assisted landing, the robot has to be capable of conducting fast linear traversal motions through its workspace. Since robot workspace singularities exist, which prevent straight linear movements, a sufficient task space T_r has to be defined. The task space is a subset of the robot's total workspace T_g and ensures certain characteristics demanded by the specific task. For the ground platform, the workspace T_g is reduced by potential front and side wall collisions. In a previous work, the robot's task space was analyzed based on the condition number of the Jacobian matrix [1, p. 8]. For a certain configuration type $\mathbb{P}$, which includes all joint angle vectors $q_r = (q_1, q_2, \dots, q_6)^T$ with

$$-\frac{\pi}{2} \leq q_1 \leq \frac{\pi}{2} \quad (1a)$$

$$-\frac{\pi}{4} \leq q_3 \leq \frac{\pi}{2} \quad (1b)$$

$$-\frac{\pi}{2} \leq q_4 \leq \frac{\pi}{2}, \quad (1c)$$

an approximately linear mapping between Cartesian velocities and joint velocities could be observed within T_r . This finding enables the possibility to neglect the robot's kinematics during the optimization process design, since an approximately linear mapping is given

after optimization. Fig. 4 visualizes the chosen task space. To ensure self-collision and singularity avoidance,

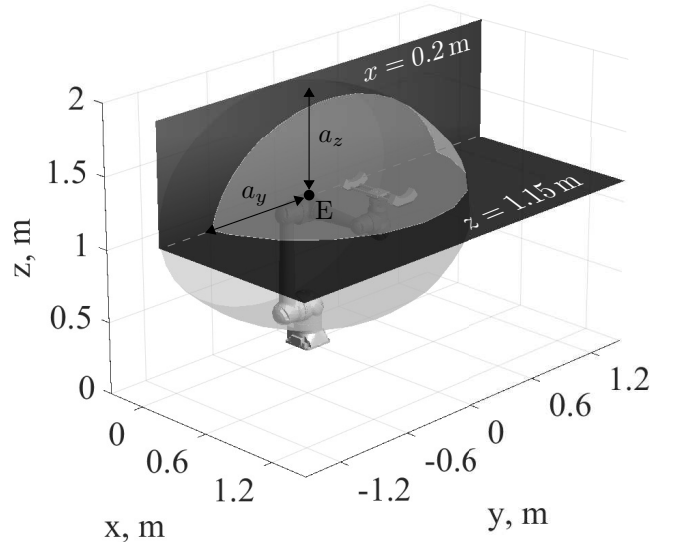


Fig. 4: Task space visualization

the task space is restricted by two planes. Another finding of [1] is that the designed gripper

enables certain tolerances for the catching process. Tab. 1 lists the identified tolerances for each degree of freedom. The tolerances are stated concerning the gripper’s coordinate system t , which is located in the robot’s tool center point (TCP). Due to the gripper’s and UAV’s geometry, a roll angle deviation of up to $\pm 20^\circ$ can be tolerated during the catching process. Based on the catching tolerances documented in Tab. 1, decision criteria for the execution of the catch can be formulated. In addition, the differently restrictive tolerances offer

Table 1: Tolerances of the UAV-gripper system

Tolerance	Value
Δx^t	$\pm 60 \text{ mm}$
Δy^t	$\pm 20 \text{ mm}$
Δz^t	$+ 30 \text{ mm}$
$\Delta \phi^t$	$\pm 20^\circ$
$\Delta \theta^t$	$\pm 5^\circ$
$\Delta \psi^t$	$\pm 10^\circ$

an individual approach for synchronization. For example, the roll angle tolerance $\Delta \phi^t$ allows a slackness for roll angle synchronization in contrast to the other two rotational degrees of freedom.

2.5 Planning Space Considerations

To utilize MPC for an industrial robot, it has to be chosen whether the state variables are represented in the Cartesian or the joint angle space. The UAV’s predicted pose is provided by the Kalman filter in a Cartesian space representation, meaning position and Euler angles of the UAV are given (Fig. 3). The robot, however, is controlled by its six individual joint angles, representing a joint angle space. Generally, the representation choice is connected to the specific robotic task [12, p. 166]. For a predetermined trajectory, the joint angle planning is recommended. However, for cases in which the robot’s end effector shall perform responsive trajectories based on a constantly changing reference, the Cartesian planning is recommended. The task of robot-assisted landing is part of the latter. Nevertheless, an advantage of joint space planning is given by the direct consideration of individual joint restrictions. In particular, joint angle and velocity constraints may be taken into account. This allows for the full utilization of the robot’s performance capabilities. The mapping of individual joint restrictions into Cartesian space is not trivial for a 6-axis serial kinematic robot. Individual joint constraints are interconnected and nonlinear

with respect to the corresponding Cartesian space representation. However, they do not need to be taken into account in the MPC, since the robot moves at a sufficient distance from joint velocity constraints during the robot-assisted landing (robot is capable of translational velocities of 2 ms^{-1} [1, p. 8], whereas the UAV moves at maximum translational velocities of approximately 1 ms^{-1} , [1, p. 4]). In addition to joint velocities, joint space planning offers the option of taking joint angle constraints into account. Yet, by maintaining a certain configuration type $\mathbb{P}$ within T_r it is already ensured that the robot joints do not reach their limits. Thus, with the given boundary conditions, joint space planning has no advantages over planning in Cartesian space for the task of a robot-assisted landing.

On the other hand, the advantages of planning in Cartesian space are particularly given for the task of a robot-assisted landing. For example, task space boundaries, translational velocity restrictions or an imposed vertical offset to the aircraft can be taken into account directly in the MPC. Additionally, due to the aforementioned approximately linear mapping between Cartesian velocities and joint velocities within T_r , solving the inverse kinematics (IK) problem to get joint angles for a given pose do not need to be considered during optimization in the MPC. In order to command the planned Cartesian pose to the robot, the IK may be applied after the optimization.

3 Controller Design

3.1 Robot Model

The robot’s system dynamics can be considered in the MPC with the help of equality constraints. While having the real-time constraints of 250 Hz in mind, the model type should be as easy to solve as possible. In Section 2.5, the necessary complexity of the robot model was discussed. As a result, the nonlinear kinematics of the robot may be neglected during optimization and considered afterwards by the IK solver. Additionally, the robot’s dead time can be introduced inherently by using MPC. Thus, a linear model type (double integrator) was chosen to enable fast computation of model dynamics. The model’s state comprises the pose of the robot as well as the corresponding time derivatives excluding the roll angle ϕ . All states are expressed with respect to the robot’s base coordinate system b :

$$x_r = [x^b \ y^b \ z^b \ \psi^b \ \theta^b \ \dot{x}^b \ \dot{y}^b \ \dot{z}^b \ \dot{\psi}^b \ \dot{\theta}^b]^T \in \mathbb{R}^{10 \times 1}. \quad (2)$$

Considering Table 1, the pitch angle θ and yaw angle ψ have significantly lower tolerances compared to the roll

angle ϕ . Neglecting the roll angle during the landing process can be justified with typical roll angles of the UAV in hover flight of $\pm 10^\circ$ compared to a tolerance of $\pm 20^\circ$ due to the gripper's geometry [1, p. 4].

The corresponding input variables are given by the second derivative of the robot's pose with respect to time:

$$u_r = [\ddot{x}^b \ \ddot{y}^b \ \ddot{z}^b \ \ddot{\psi}^b \ \ddot{\theta}^b]^T \in \mathbb{R}^{5 \times 1} \quad (3)$$

For each time step k the state of the next time step $k+1$ may be computed by

$$x_{r,k+1} = A_r \cdot x_{r,k} + B_r \cdot u_{r,k}. \quad (4)$$

The robot's motion model, characterized by the system matrix

$$A_r = \begin{bmatrix} I & \Delta t \cdot I \\ 0 & I \end{bmatrix} \in \mathbb{R}^{10 \times 10} \quad (5)$$

and the input matrix B_r

$$B_r = \begin{bmatrix} \frac{1}{2} \Delta t^2 \cdot I \\ \Delta t \cdot I \end{bmatrix} \in \mathbb{R}^{10 \times 5} \quad (6)$$

is assumed to be time invariant.

3.2 Cost Function

In order to assess the reference trajectory tracking a quadratic cost function is applied:

$$J(x_{1:N_c}, u_{0:N_c-1}) = u_{r,0}^T R_0 u_{r,0} + \sum_{k=1}^{N_c-1} [(x_{r,k} - x_{ac,k})^T Q_k (x_{r,k} - x_{ac,k}) + u_{r,k}^T R_k u_{r,k}] + (x_{r,N_c} - x_{ac,N_c})^T Q_{N_c} (x_{r,N_c} - x_{ac,N_c}) \quad (7)$$

The input variables are weighted with weighting matrices R_k starting at the current timestep $k = 0$ up to the timestep $k = N_c - 1$ which portrays the last timestep effecting the state inside the control horizon. For the states' deviation, the weighting matrices are called Q_k . The corresponding evaluation horizon is shifted one timestep forwards in contrast to the input variables, including a terminal cost value. By applying a quadratic cost function like Eq. 7, a reference state for the UAV x_{ac} can be introduced which allows for synchronization evaluation of robot and UAV. However, only a basic structure is given with J . The tangible effect on the characteristics of the reference trajectory tracking is influenced by the choice of weighting values for R and Q only.

By choosing a diagonal matrix for Q , the states are weighted independent from each other. Absolute values of the weights do not matter, the relation between each other is decisive here. Since assessing the pose synchronization is the primal goal, the costs for the pose state deviation in comparison to the pose velocity state deviation have to be chosen higher:

$$Q_r = Q_k = \begin{bmatrix} 10^3 \cdot I & 0 \\ 0 & 10^0 \cdot I \end{bmatrix} \in \mathbb{R}^{10 \times 10} \ \forall k \in [1, N_c] \quad (8)$$

However, to ensure a smooth landing the velocity synchronization is declared as a secondary goal by choosing the weights accordingly. For simplicity, the weighting matrices do not change along the control horizon. For the input weights, there are no costs to be considered. However, since not weighting them at all would result in infinite utilization of input variables and hence to numerical instabilities a low weight of

$$R_r = R_k = [10^{-3} \cdot I] \in \mathbb{R}^{5 \times 5} \ \forall k \in [0, N_c - 1] \quad (9)$$

is applied.

3.3 Constraints

Since the optimization problem needs information about the robot's task space and actuator limitation, constraints are introduced. Generally, a subdivision in linear and nonlinear constraints can be made. Linear constraints can be formulated as

$$x_{r,\min} \leq x_r \leq x_{r,\max}, \quad (10)$$

$$u_{r,\min} \leq u_r \leq u_{r,\max}. \quad (11)$$

The state variables x_r and the input variables u_r are assigned to a lower and an upper boundary individually. Geometrically, linear constraints portray planes for a variable like the robot's position in a three dimensional Cartesian space. To consider velocity, acceleration or cubical-shaped spatial limitations, linear constraints are satisfactory. Since the robot's task space does not represent a cuboid but an ellipsoid, linear constraints only are not sufficient. In order to consider an ellipsoid, the robot's position $(x, y, z)^T$ has to be checked against

$$\underbrace{\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} \frac{1}{a_x} & 0 & 0 \\ 0 & \frac{1}{a_y} & 0 \\ 0 & 0 & \frac{1}{a_z} \end{bmatrix}}_E \begin{bmatrix} x \\ y \\ z \end{bmatrix} \leq 1. \quad (12)$$

Here the matrix E specifies the semi-major axes of the ellipsoid. Eq. 12 portrays a nonlinear equation with respect to the robot's position. Thus, for considering the

ellipsoidal task space in the MPC, non-linear constraints have to be formulated.

3.4 Optimization Problem

The content of the optimization problem is defined by the description in the previous sections. However, the problem formulation still needs to be discussed. As mentioned before, the controller will be executed in a real-time environment with a frequency of 250 Hz. Therefore, the optimization problem has to be solved efficiently and in a deterministic fashion. A convex optimization problem formulation [13, p. 7] allows for both and will be utilized in the following. Different convex optimization problem formulation yield different complexities. For example a quadratic problem (QP), defined as

$$\begin{aligned} \min \quad & z^T Q z + q^T z \\ \text{s.t.} \quad & F z = g \\ & G z \leq h. \end{aligned} \quad (13)$$

comes with a quadratic cost function and linear constraints [13, p. 152]. The QP is stated in matrix formulation for an optimization variable z . While allowing for a quadratic cost function and linear constraints, the QP formulation does not allow for the nonlinear ellipsoidal constraint for the robot's task space. Although the ellipsoidal task space constraint can't be considered inherently when utilizing a QP, a command value filter may be designed to allow for ellipsoidal constraints. The filter has to project the UAV position onto the ellipsoid if the UAV is located outside of the robot's task space. The corresponding idea is presented in Fig. 5. A pro-

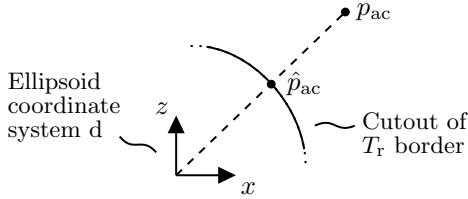


Fig. 5: Idea of a command value filter

jected UAV position $\hat{p}_{ac}^d$ is obtained based on the actual position p_{ac} . Therefore the logic

$$\hat{p}_{ac}^d = \begin{cases} r_d \cdot p_{ac}^d & \text{for } r_d < 1 \\ p_{ac}^b & \text{for } r_d \geq 1 \end{cases} \quad (14)$$

is applied. The scaling factor r_d is obtained by solving the ellipsoid equation as

$$r_d = \sqrt{\frac{1}{\frac{x^2}{a_x} + \frac{y^2}{a_y} + \frac{z^2}{a_z}}}. \quad (15)$$

The ellipsoid is characterized by the semi-major axes a_x , a_y and a_z . With this approach the robot may possess a remaining velocity component pointing outside T_r when reaching the border. Thus, the MPC would plan a trajectory which would leave T_r in this case. To mitigate this, a task space buffer needs to be applied which reduces the effective robot task space. A visualization of this phenomenon will be presented in Sec. 4.

Increasing the complexity of the convex optimization problem, the second order cone problem (SOCP), defined as

$$\min \quad f^T z \quad (16a)$$

$$\text{s.t.} \quad F z = g \quad (16b)$$

$$\|A_i z + b_i\|_2 \leq c_i^T z + d_i, \quad i = 1, \dots, m. \quad (16c)$$

can be formulated [13, p. 156]. The SOCP comprises a linear cost function, a linear equality constraint and a norm inequality constraint. Utilizing a SOCP, Eq. 16c allows for the nonlinear ellipsoidal constraint for the robot's task space. A quadratic cost function, like defined in Section 3.2, can be considered by introducing an additional optimization variable and utilizing the norm inequality constraint [14, p. 197]. Both, the QP and the SOCP provide different advantages for the task of robot-assisted landing and will be analyzed in the following.

For solving convex optimization problems like QP or SOCP, a variety of different algorithms exists [13, p. 455]. By exploiting task-related characteristics of the optimization problem such as matrix sparsity, highly specialized solvers for both, QP and SOCP will be evaluated in the following.

3.5 Controller Architecture

The controller obtains the UAV pose from the Kalman filter and calculates joint angles for the robot in each time step. Fig. 6 visualizes the controller architecture as a block diagram for a timestep $k = 0$, utilizing a QP optimization problem formulation. For the implementation of different phases of the landing process, like depicted in Fig. 2, the command value has to be modifiable to realize for example a constant vertical offset to the actual UAV position. The Kalman filter provides the controller with the UAV state vector for the optimization horizon $x_{ac, N_d+1:N_p}$ with an adjustable offset

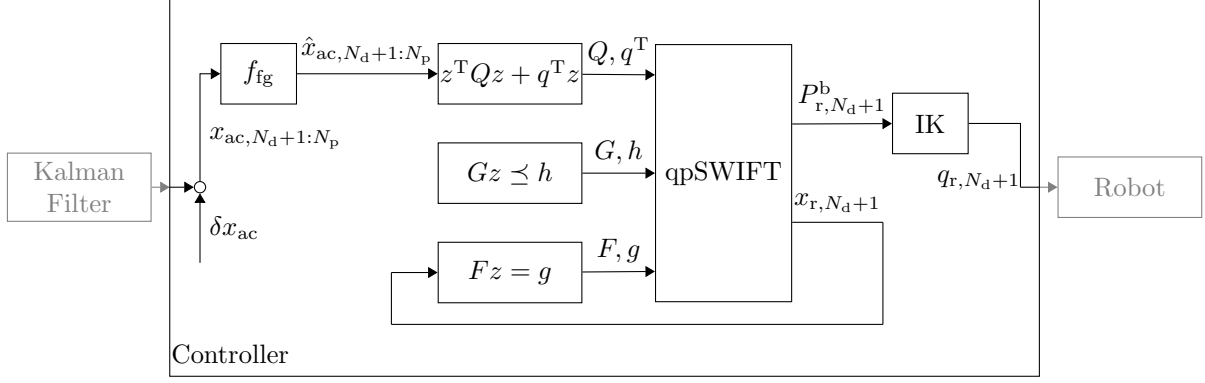


Fig. 6: Block diagram of QP controller

of δx_{ac} . To achieve a phase-defined landing process, the vertical distance and the pitch angle synchronization will be adjusted with

$$\delta x_{ac} = [0 \ 0 \ d_{ac} \ 0 \ \zeta_{ac} \ 0 \ 0 \ 0 \ 0]^T \in \mathbb{R}^{10 \times 1} \quad (17)$$

only. For the QP controller, the command value filter f_{fg} is applied to restrict the UAV state to the robot's ellipsoidal task space before transferring the altered UAV state $\hat{x}_{ac}$ to the optimization problem. Additionally, inequality and equality constraints are defined and transferred. To ensure a smooth trajectory, the last computed robot state is fed back as an initial condition. For solving the QP, the open-source solver qpSWIFT [15] is utilized. qpSWIFT was developed explicitly for robotic applications with sparse matrices. The solver is able to perform an optimization variable initialization and applies a modified interior point method for solving the optimization problem. The optimization ends as soon as defined absolute or relative tolerances of the Karush-Kuhn-Tucker (KKT) conditions are met. After optimization, the pose for the timestep $N_d + 1$ is extracted and transferred to the inverse kinematics (IK) solver. Since the expected changes of joint angles are small, the iterative IK solver TRAC-IK [16] is utilized to compute the small iterative change in robot joint angles, opposed to more computationally expensive types of IK solvers. After TRAC-IK has finished, the joint angles q_{r, N_d+1} are verified to be part of the desired configuration set $\mathbb{P}$. If this verification fails, the joint angles of the last timestep are commanded.

For the SOCP controller, the architecture resembles the one of the QP controller. However, no command value filter is needed, since the task space constraint may be handled inherently by the optimization problem. For solving the optimization problem the open-source solver ECOS [17] is utilized. The execution time of the SOCP solver is expected to be higher, given the more complex

optimization problem formulation. The results of this comparison are presented in Section 4.

3.6 State Machine

For execution, the designed MPC is embedded in a state machine. The state machine is based on the three phases of the robot-assisted landing, visualized in Fig. 2. To achieve conditional state transitions inside the state machine, decision criteria have to be specified. For that, a decision variable

$$s_k = [q_{r,k} \ V_{pos,k} \ P_{ac,k+1:k+N_p}^b \ \dot{p}_{r,k}^b]^T \quad (18)$$

is defined for each timestep k with a length of $N_s = 6 + 3 + (6 \cdot N_p) + 3$. Here, s_k comprises the current robot configuration $q_{r,k} \in \mathbb{R}^{6 \times 1}$, the current variances of the UAV position estimate of the Kalman filter $V_{pos,k} \in \mathbb{R}^{3 \times 1}$, the predicted poses of the UAV $P_{ac,k+1:k+N_p}^b \in \mathbb{R}^{(6 \cdot N_p) \times 1}$ as well as the current time derivative of the robot position $\dot{p}_{r,k}^b \in \mathbb{R}^{3 \times 1}$ in the base coordinate system b . The decision criteria are formulated as sets in order to perform a check whether the decision variable s_k is an element of the corresponding decision criteria set. If s_k is an element of the corresponding set, a state transition occurs. State transitions occur deterministically defined by the underlying finite state machine. The flowchart of the designed state machine is pictured in Fig. 7. The state machine starts with an initialization of parameters and variables followed by a first synchronization between the UAV and the robot with a chosen vertical distance of $d_{ac} = -0.3$ m and no pitch angle synchronization. The state machine ends with the robot's deceleration and homing after a successful catch. In between, state transitions occur based on membership examinations regarding s_k . For example the set S_L , titled as "synchronization condition (coarse)" is defined

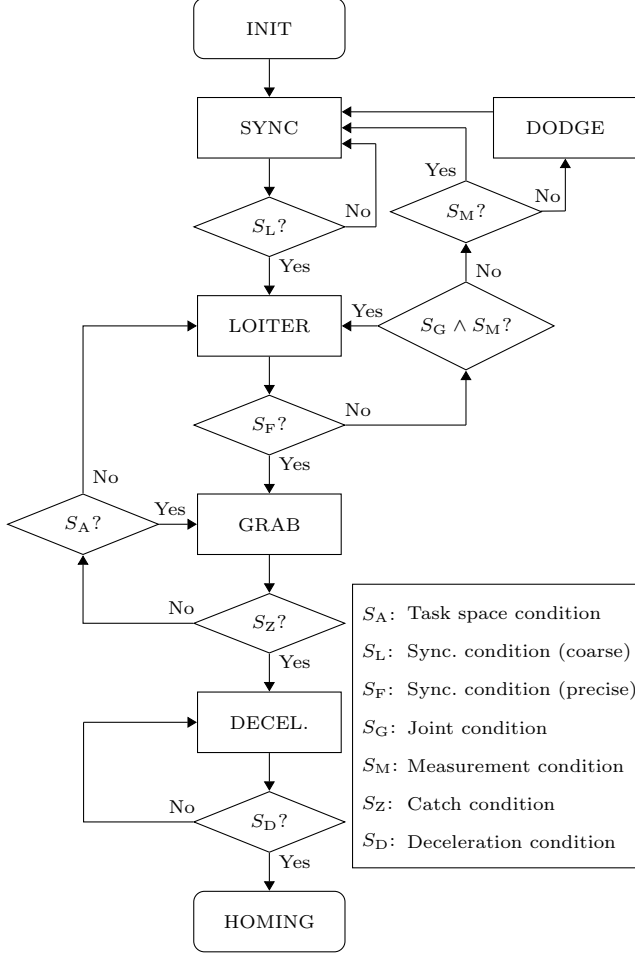


Fig. 7: Flowchart of the state machine

as

$$S_L = \left\{ s_k \mid s_k \in (S_M \cap S_A \cap S_G \cap S_I) \right\}. \quad (19)$$

Thus, the decision variable s_k has to be an element of the measurement criteria set

$$S_M = \left\{ s_k \mid V_{\text{pos},k} \preceq 1 \right\}, \quad (20)$$

the task space criteria set

$$S_A = \left\{ s_k \mid P_{\text{ac},k+1:k+N_p}^b \in T_r \right\}, \quad (21)$$

the joint criteria set

$$S_G = \left\{ s_k \mid -160^\circ \leq q_{6,k} \leq 160^\circ \right\} \quad (22)$$

and the coarse synchronization criteria set

$$S_I = \left\{ s_k \mid \begin{aligned} &-0.1 \text{ m} \leq x_{\text{ac},k+1:k+N_p}^t \leq 0.1 \text{ m}, \\ &-0.1 \text{ m} \leq y_{\text{ac},k+1:k+N_p}^t \leq 0.1 \text{ m}, \\ &0 \text{ m} \leq z_{\text{ac},k+1:k+N_p}^t \leq 0.1 \text{ m}, \\ &-30^\circ \leq \psi_{\text{ac},k+1:k+N_p}^t \leq 30^\circ, \\ &-10^\circ \leq \theta_{\text{ac},k+1:k+N_p}^b \leq 10^\circ \end{aligned} \right\} \quad (23)$$

for the state transition to be triggered. Each set describes a task-related condition. The quality of the pose estimate is evaluated with the measurement criteria set S_M . The task space criteria set S_A allows to assess whether the robot task space will be left by the UAV within the prediction horizon. Since the sixth joint angle may encounter the hardware angle limitation of $\pm 210^\circ$, a hysteresis of $\pm 180^\circ$ is implemented, the joint criteria set S_G ensures a sufficient angle distance to the hysteresis' threshold. Finally, the coarse synchronization criteria set S_I comprises all s_k for which the gripper pose (excluding the roll angle) is "coarsely" synchronized with the UAV.

If s_k is an element of S_L , the state transition into LOITER is invoked, reducing the vertical distance to the UAV to $d_{\text{ac}} = -0.1 \text{ m}$ and starting to synchronize the pitch angle. Tab. 2 lists the corresponding position offsets for different states. As soon as s_k is an ele-

Table 2: Vertical position offset of gripper to the UAV

State	d_{ac}
SYNC	-0.3 m
LOITER	-0.1 m
GRAB	-0.01 m

ment of S_F , the state transition into GRAB occurs and the vertical distance to the UAV is further reduced to $d_{\text{ac}} = -0.01 \text{ m}$. For that, S_F is identical to S_L , only that the gripper pose (excluding the roll angle) has to be "precisely" synchronized with the UAV, meaning the corresponding catch tolerances (Tab. 1) have to be met. If s_k is not an element of S_F the system either remains in LOITER if s_k is an element of S_G and S_M or returns to SYNC. If the pose estimate quality is low and s_k is not an element of S_M , the state DODGE will ensure an appropriate vertical distance to the UAV even without a pose estimate of the UAV before returning to SYNC. In GRAB the catch condition set S_Z gets checked, which

solely purpose is the assessment of a sufficient synchronization between UAV and gripper by considering the current vertical position offset of $d_{ac} = -0.01$ m. Concurrently, the gripper is closed based on the current vertical distance to the UAV. As long as s_k is not an element of S_Z but an element of S_A the system remains in GRAB.

After a successful catch and a state transition into DECELERATE, the system is decelerated within predefined acceleration constraints. If s_k is an element of S_D , the system halted and a state transition in HOMING occurs. Consequently, the system moves to a known pose with predefined acceleration and velocity constraints. The robot-assisted landing is finished.

4 Results and Discussion

4.1 Simulation of Trajectory Planning

The simulations were conducted on a Intel Core i5-8265U utilizing the programming language MATLAB to verify the fundamental working principle of the trajectory planning and to assess different scenarios for the QP and the SOCP controller concerning their control characteristics. For MATLAB, the solver qpSWIFT and ECOS provide corresponding interfaces. For the simulation, KKT condition tolerances were set to 10^{-8} and the maximum number of iterations to 15.

Fig. 8 shows the differences between the QP and the SOCP controller for an artificial example. The UAV

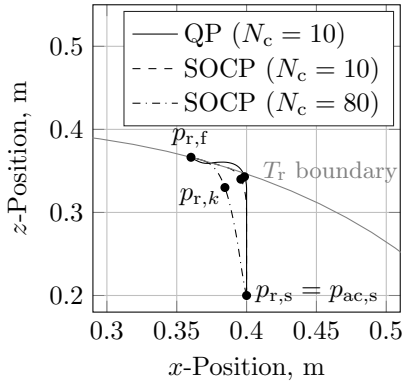


Fig. 8: Trajectory planning for QP and SOCP

leaves the robot's task space T_r (drawn in gray) on a fast linear trajectory starting at $x = 0.4$ m and $z = 0.2$ m and stopping at $x = 0.4$ m $z = 0.5$ m. The depicted robot trajectories starting at $p_{r,s}$ and ending in $p_{r,f}$ are generated with a QP controller and two SOCP controllers with varying optimization horizons N_c respectively. At timestep k , the UAV leaves T_r . For the QP controller, the UAV position p_{ac} gets projected

onto the task space boundary via the command value filter f_{fg} . As described in Sec. 3.4, the finite deceleration of the robot results in an overshoot of the task space boundary. Thus, a reduction of the task space $T_r^* \subset T_r$ for the QP controller is necessary to ensure that the robot remains within T_r .

If, on the other hand, the SOCP controller with identical optimization horizon N_c is applied, no overshoot appears. Here, the task space boundary is part of the optimization problem and hence gets considered during the trajectory planning. However, Fig 9 emphasizes the increased computational time for solving a more complex optimization problem. For $N_c = 10$, the time for

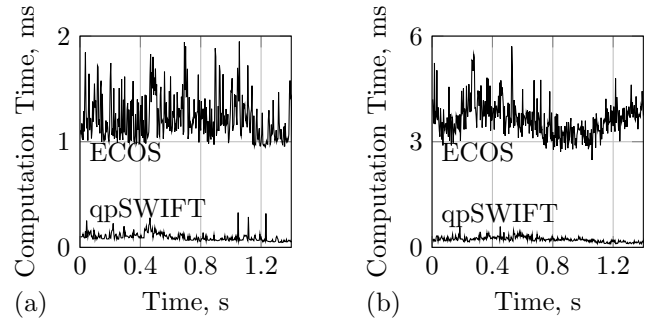


Fig. 9: Computation time for QP and SOCP controllers for (a) $N_c = 5$ and (b) $N_c = 10$

solving the SOCP varies around 4 ms, whereas the solution for the QP is obtained in less than 1 ms. To keep the computation time for the SOCP within the real-time constraints, the optimization horizon N_c has to be reduced. For $N_c = 5$, the SOCP is solved in less than 2 ms. Note, the computation time for ECOS increases quadratically, whereas for qpSWIFT only linearly. This is because the number of equations to be solved for the SOCP increases proportionally to the square of the optimization horizon ($\propto N_c^2$), whereas the number of equations to be solved for the QP increases only linearly with the increase of the optimization horizon ($\propto N_c$). By increasing the optimization horizon to $N_c = 80$, the robot's trajectory will adjust as soon as the UAV leaves the task space within the optimization horizon, resulting in a shorter trajectory to the final position p_r compared to the other two controllers (Fig. 8). For an ideal prediction of the UAV's position, an increased optimization horizon would be desirable. However, the prediction obtained by the Kalman filter is not ideal and hence shows an increased variance for longer prediction horizons. Thus, the choice for N_c is based on a consideration between controller responsiveness and prediction quality.

The increased computational time of the SOCP controller violates the system's real-time constraints already for small control horizons N_c . This observation was also made when utilizing the host computer's CPU. Therefore and since advantages over the QP controller during the robot-assisted landing become recognizable only for significantly longer prediction horizons, concurrently introducing a comparably high variance in computation time, the QP controller was chosen for implementation on the target hardware.

4.2 Laboratory Experiments

After the trajectory planning has been analyzed based on simulations, laboratory experiments on the target hardware were conducted to tune parameters empirically and validate the trajectory planning with the help of a test campaign and a mock-up UAV. Therefore, the controller is executed as a C++ runtime library on the real-time host computer. The host computer is equipped with a Intel Core i7-8700T CPU.

This section first describes the experimental setup with regard to the mounting system of the UAV mock-up in the laboratory. In particular, the differences to the behavior of the flying UAV are analyzed. This is followed by an estimation of the dead time to be taken into account for the MPC. The estimated dead time should include both the dead time of the robot and its further second-order deceleration behavior. Finally, a test campaign for validation of the model predictive trajectory planning is described.

4.2.1 Rope system

For the laboratory experiments, a rope system and a UAV mock-up were utilized (Fig. 10). The rope system

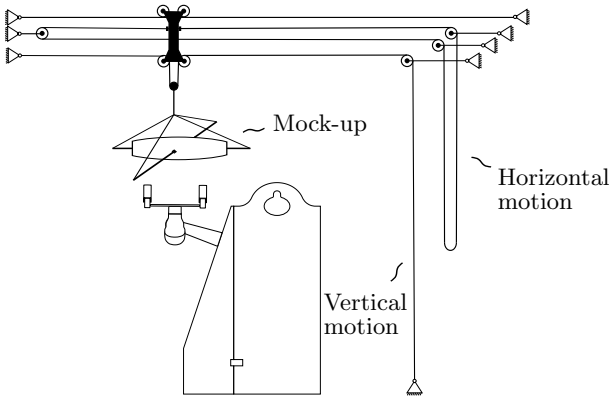


Fig. 10: Rope system

allows for horizontal and vertical motion of the mock-up via a pulley. Thus, the mock-up was able to leave and approach the robot's task space for validation of the trajectory planning's functionality, examining pose detection outages and controller initialization aspects. The attitude of the mock-up is adjustable with pitch and roll angles of approximately $\pm 45^\circ$. Furthermore, the UAV mock-up replicated the flying UAV by sending rotational velocities and translational accelerations to the robot via the ESP-NOW telemetry link, measured by an IMU. Additionally, the moment of inertia and the aircraft weight (mass of ≈ 7.5 kg and wingspan of ≈ 1.8 m) were aligned by mounting weights on the mock-up.

The rope system, however, induces a pendulum motion on the mock-up's movements due to the single-point suspension. Considering that the laboratory experiments were utilized for validation and testing, the pendulum motion had to be minimized, since the flying UAV will not show these characteristics. Fig. 11 visualizes the pendulum motion. To mitigate the pendulum

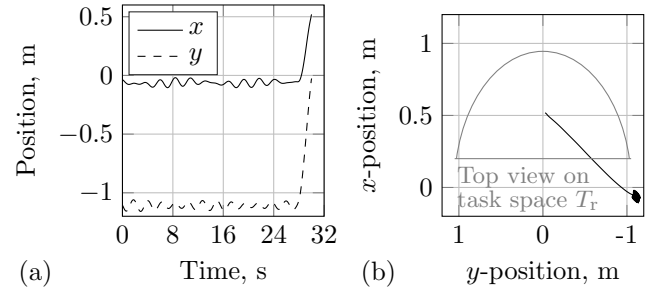


Fig. 11: Movement of UAV mock-up into task space T_r shown as (a) positional values over time and (b) a top view on task space T_r

motion's effect on the flight trajectory, the mock-up was moved into the task space T_r at $t \approx 28$ s with a constant translational velocity, resulting in a horizontal motion into T_r . While the mock-up was moving, the catching process was performed.

4.2.2 Dead time estimation

As described in Sec. 2, the robot possesses a dead time of approximately 40 ms and further behavior of a second order delay element. The robot model of the MPC, however, does not incorporate the delay behavior, which would demand a nonlinear transformation from joint space into the Cartesian space via forward kinematics during the optimization to represent the axis-wise delay. Here, the more pragmatic approach has been taken, where a total robot delay N_d was estimated, which

includes both the robot's dead time and the second order delay element characteristics.

For the experimental investigation, a constant optimization horizon of $N_c = 10$ was set and different values for N_d were assessed, starting at $T_d = 40$ ms. The investigation is based on the assumption that the delay behavior of the robot occurs alike in the corresponding dimensions of Cartesian space. This applies without restriction to the dead time of the robot, since no directional dependency exists. When considering the PT_2 behavior, this is a simplified assumption. Since the parameter space of N_d is discretized in steps of 4 ms, a full parameter space search for N_d up to $T_d = 120$ ms could be performed easily. For a visualization, Fig. 12 shows the current robot's x -position $x_{\text{pos},r}$, the current solver's output $x_{\text{pos},\text{sol}}$ and the current UAV's x -position $x_{\text{pos},\text{ac}}$ as the reference trajectory in the robot's base coordinate system. The goal is to align $x_{\text{pos},r}$ and $x_{\text{pos},\text{ac}}$.

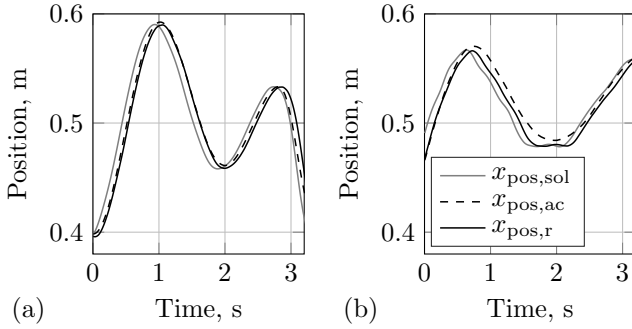


Fig. 12: Analysis of the robot's delay behavior for (a) $T_d = 80$ ms and (b) $T_d = 120$ ms

For $T_d = 80$ ms (Fig. 12a), an acceptable tracking of the UAV's position is recognizable. At approximately 3 s, however, the dead time compensation is still insufficient. For $T_d = 120$ ms (Fig. 12b), the robot precedes the UAV's movement. Additionally, with a prediction time totaling $T_p = T_d + T_r = 130$ ms, it is recognizable that the quality of the Kalman filter's linear pose prediction decreases. The commanded robot position is visibly influenced by the erratic pose prediction of the Kalman filter, which possesses a high variance for a prediction horizon of $T_p = 130$ ms, compared to the prior example with $T_p = 90$ ms. Nevertheless, the filtering property of the MPC for the commanded trajectory with respect to the predicted reference trajectory (not shown) can be recognized. A satisfactory result was achieved with $T_d = 88$ ms.

4.2.3 Test campaign

For the validation of the robot-assisted landing a test campaign was designed. The test campaign resulted in 44 successful catching processes with covering the full state space of the state machine as well as covering the UAV's entry into the heterogeneous robot task space from different Cartesian directions. Additionally, tuning parameters such as the catching criteria, the timing for the gripper's catch command and the control horizon length were determined. The prediction horizon N_p was chosen to $N_p = N_d + N_c = 22 + 10 = 32$ ($T_p = 128$ ms). The landing processes were conducted while meeting the real-time constraints of 250 Hz. Fig. 13 depicts nine exemplary catching trajectories of the test campaign in the x - y -plane. The robot's position in the x - y -plane p_r is shown for 400 ms prior the state transition into GRAB, indicated by the cross marker. The current error in position to the UAV's flight trajectory e_{sync} in the x - y -plane is represented by vectors. Generally, the synchronization error decreases approaching the transition into GRAB. However, e.g. in Fig. 13d, an increase in e_{sync} can be recognized (to satisfy conditions, such that e.g. $s_k \in S_F$). As long as e_{sync} does not exceed the catching tolerances (see Tab. 1), a safe catching process can be conducted. Further analysis of the landing process is shown in the following section using a field test as an example.

4.3 Field Tests

A total of 5 field tests were conducted with the flying UAV. For a detailed analysis and discussion of the robot-assisted landing, one of them will serve as a representative example. An impression on the chosen field test is shown in Fig. 1. During the field test, the errors between the position components of the projected UAV and the gripper were within the range of ± 6 mm, and hence well within the identified tolerances of the UAV-gripper system (see Tab. 1).

4.3.1 Landing process overview

Fig. 14 shows the time-shifted visualization of the synchronization of the robot's position with the UAV's position in the robot's base coordinate system, meaning that $t = 0$ s is shifted to the state transition from LOITER to GRAB. The UAV's approach is mainly noticeable while looking at the z -position. At $t \approx 0$ s, the robot reduces the vertical distance to the UAV and the catch is performed. Concurrently, the x and y -positions as well as the yaw angle ψ (not shown) are synchronized. After the state transition from SYNC to LOITER, the pitch angle is synchronized as well.

At the start of the synchronization around -12 s an increase in x -position deviation due to the robot's

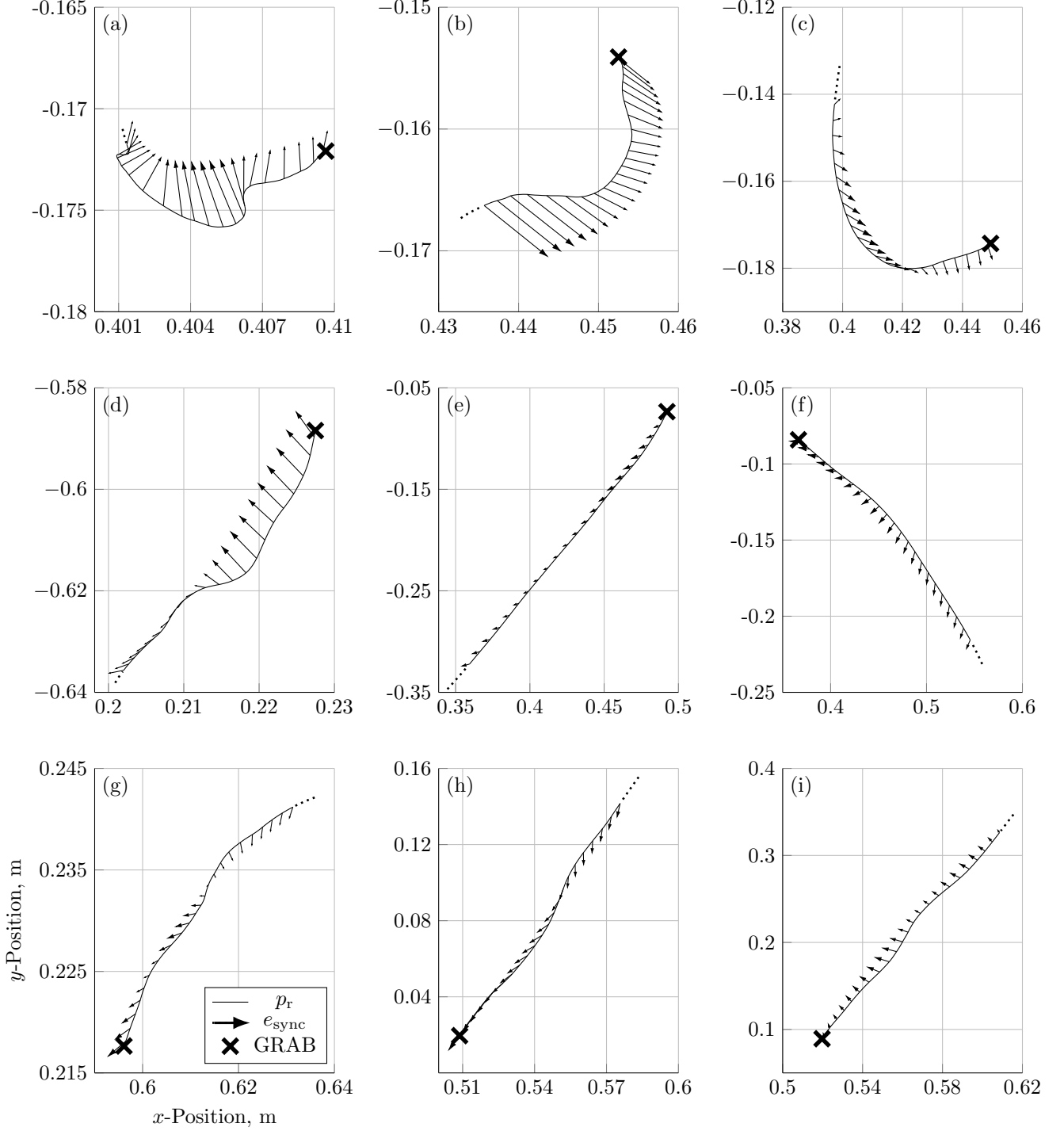


Fig. 13: Exemplary catching trajectories during the test campaign

motion restriction is recognizable. Since the UAV is not within the robot's task space T_r , the command value filter projects the UAV's position on the boundary of T_r . In this example, the chosen projection method generates a projected x -position which deviates from the current UAV's x -position. As the UAV approaches,

the difference between projected and actual x -position is reduced. During synchronization with the projected UAV's position, the robot's x -position is limited to 0.2m at around -11 s for robot singularity avoidance. After a successful catching process, the robot moves back to its neutral position at $(x, y, z)^T =$

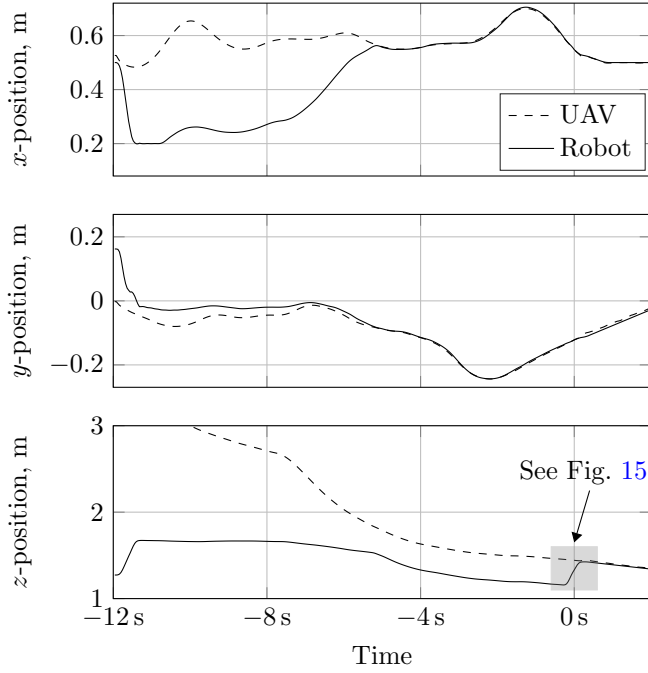


Fig. 14: Catching process

$(0.5 \text{ m}, 0.162 \text{ m}, 1.27 \text{ m})^T$ with the UAV firmly gripped in neutral orientation. For that, acceleration and velocity constraints ensure a slow and smooth trajectory for a safe UAV movement.

4.3.2 Catching process

The catching process, indicated by the gray area in Fig. 14, is visualized in Fig. 15 in detail. For this purpose, the respective states of the state machine, important z -positions in the base coordinate system as well as velocities and accelerations of the robot are visualized. The different z -positions include the position of the UAV $z_{\text{pos,ac}}$, the reference position of the MPC N_p time steps in the future $z_{\text{pos,ref}}$, the commanded position of the robot $z_{\text{pos,sol}}$, and the position of the robot $z_{\text{pos,r}}$. The vertical guide lines mark the points in time at which state transitions occur.

At the start, the robot is in the state SYNC. The transition to LOITER is made using the coarse synchronization condition S_L . The vertical distance offset is adjusted to $d_{\text{ac}} = -0.1 \text{ m}$, and the robot moves toward the UAV at maximum acceleration and speed. At approximately -120 ms reaching the corresponding vertical distance already enters into the optimization horizon of the MPC. Since the vertical distance of 0.1 m should be maintained as long as the fine synchronization condition ($s_k \in S_F$) is not satisfied, a deceleration command is issued. As soon as the distance between $z_{\text{pos,ref}}$ and $z_{\text{pos,sol}}$ reaches the tolerance value of 0.03 m and the precise synchronization condition S_F is met, the

transition to GRAB is induced. The vertical distance offset is changed to $d_{\text{ac}} = -0.01 \text{ m}$ and the robot moves toward the UAV again at maximum acceleration and the gripper is closed. In GRAB, the catch condition S_Z is continuously checked. As soon as the catch condition S_Z is met, the robot switches to DECELERATE and decelerates its own movement and that of the UAV. It is no longer possible to return to previous states of the state machine.

In DECELERATE, the acceleration limits of the x and y accelerations are adjusted to $\pm 5 \text{ m s}^{-2}$ in order to achieve a smooth deceleration of the UAV. The acceleration limit of the z -acceleration is not adjusted, as the vertical velocity in the direction of the UAV must be reduced. With additional limitation of the z -acceleration of the robot, there would be a risk of collision.

Once the deceleration condition S_D is met, the transition to HOMING takes place. In HOMING, the acceleration limits of the MPC are again adjusted to $\pm 0.5 \text{ m s}^{-2}$ and the velocity limits of the MPC are additionally adjusted to $\pm 0.05 \text{ m s}^{-1}$ in order to achieve a slow movement towards the neutral pose. The robot-assisted landing is finished.

5 Conclusion

In this paper, a novel approach of a model predictive trajectory planning for robotic assisted landing of UAVs was designed, implemented and analyzed. The trajectory planning allows a UAV to be caught in hovering flight using a 6-axis robotic arm, assuming an approximately linear mapping between Cartesian velocities of the gripper and robot's joint velocities. Concerning the formulation of the optimization problem a second-order cone problem (SOCP) and a quadratic problem (QP) were compared. Based on a required real-time constraint of 250 Hz , the QP was selected for higher computationally efficiency. The suitability of the method presented was shown in 44 successful laboratory experiments as well as in 5 successful field tests with the flying UAV.

For future investigations, the optimization problem formulated as a second-order cone program (SOCP) offers an interesting approach. Here, the use of such a formulation was ruled out due to increased computing times with respect to the real-time requirements of the ground platform control system. However, the replacement of the solver used, an increase in available computing power or further investigations regarding the required length the optimization horizon could provide a remedy. In particular, this could enable cost-optimized consideration of the ellipsoidal workspace. If the accelerations and velocities of the UAV exceed the limits of the robot's motion dynamics in future landing scenarios,

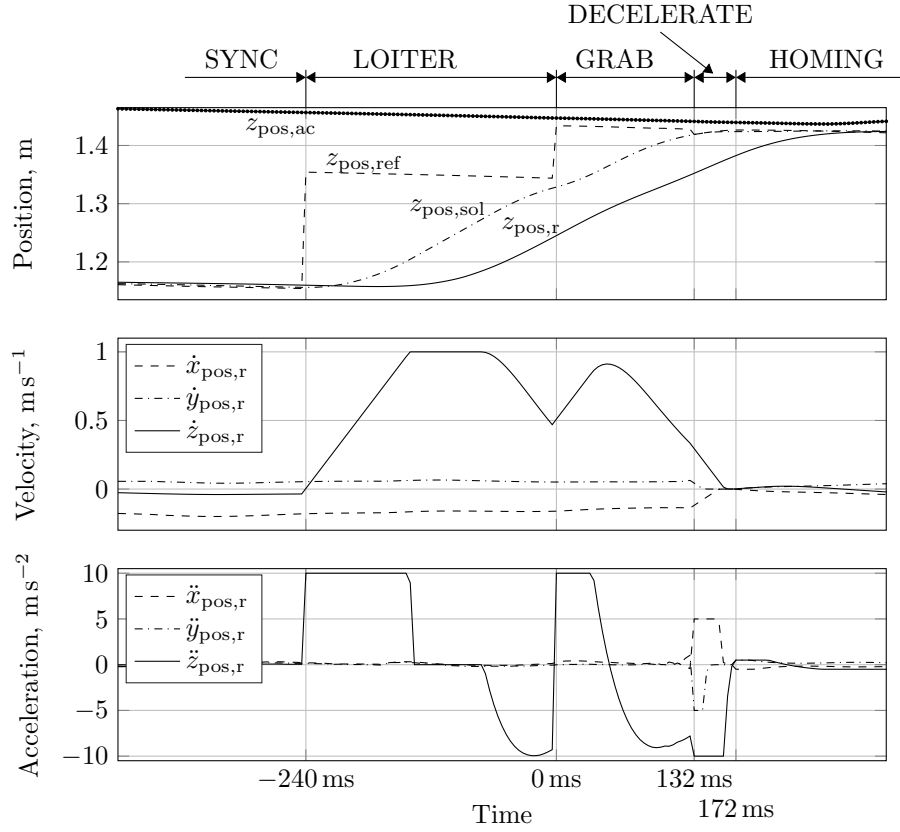


Fig. 15: Catching process in detail

it could be valuable to consider the joint space restrictions directly in the trajectory planning. This requires an alternative formulation of the optimization problem, since the nonlinear kinematics of the robot prevent the convex formulation of a quadratic program.

Acknowledgements

Our work has been conducted within the EULE project. This project is funded by the Federal Ministry for Digital and Transport (BMDV) within the mFund program. We thank all project partners for their work and contributions to the EULE project.

Gefördert durch:



aufgrund eines Beschlusses
des Deutschen Bundestages



References

- [1] Konnow, V., Terlau, J., Hildebrand, L., Hartmann, P.: Robot assisted landing process of small uavs using decentralized kalman filter. In: Proceedings of the 2024 CEAS EuroGNC Conference, Bristol, UK (2024). doi: <https://doi.org/10.82124/CEAS-GNC-2024-084>
- [2] Dobrev, Y., Schütt, M., Hartmann, P., Moormann, D.: Entwurf und Validierung eines Präzisionslandesystems für unbemannte Tiltwing-Fluggeräte. In: Luft- und Raumfahrt - Hochtechnologie aus Deutschland : Deutscher Luft- und Raumfahrtkongress 2016 : 13.-15. September 2016, Braunschweig / DGLR. Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., Bonn (2016)
- [3] Systems, Q.: FreeRail MFUND Projekt- Gleisin-spektion Mittels UAS. Youtube. Retrieved 2025-10-28. https://www.youtube.com/watch?v=iMkL_P5Za_A&ab_channel=QuantumSystems

- [4] Skydio: Skydio F10. Skydio, Inc.. Retrieved 2025-10-28. <https://www.skydio.com/f10>
- [5] Maier, M., Kondak, K.: Landing of vtol uavs using a stationary robot manipulator: A new approach for coordinated control. In: 2015 54th IEEE Conference on Decision and Control (CDC), pp. 1497–1502. IEEE, Osaka (2015). doi: <https://doi.org/10.1109/cdc.2015.7402422>
- [6] Maier, M., Oeschger, A., Kondak, K.: Robot-assisted landing of vtol uavs: Design and comparison of coupled and decoupling linear state-space control approaches. IEEE Robotics and Automation Letters **1**, 114–121 (2016). doi: <https://doi.org/10.1109/LRA.2015.2502920>
- [7] Maier, M., Kondak, K.: Robot assisted landing of vtol uavs on ships: A simulation case study of the touch-down phase. In: 2017 IEEE Conference on Control Technology and Applications (CCTA), pp. 2094–2101 (2017). doi: <https://doi.org/10.1109/CCTA.2017.8062762>
- [8] Maier, M., Kondak, K., Ott, C.: Enabling robot assisted landing of heavy uav rotorcraft via combined control and workload sharing. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 5128–5134 (2017). doi: <https://doi.org/10.1109/IROS.2017.8206399>
- [9] Xu, R., Ji, X., Hou, J., Liu, H., Qian, H.: A predictive control method for stabilizing a manipulator-based uav landing platform on fluctuating marine surface*. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 8625–8632 (2021). doi: <https://doi.org/10.1109/IROS51168.2021.9636055>
- [10] Xu, R., Jiang, Z., Liu, B., Wang, Y., Qian, H.: Confidence-aware object capture for a manipulator subject to floating-base disturbances. IEEE Transactions on Robotics, 4396–4413 (2024). doi: <https://doi.org/10.1109/TRO.2024.3463476>
- [11] Grüne, L., Pannek, J.: Nonlinear Model Predictive Control: Theory and Algorithms, 2nd edn. Springer, Switzerland (2017). isbn: 978-3-319-46023-9
- [12] Siciliano, B., Khatib, O. (eds.): Springer Handbook of Robotics, 2nd edn. Springer, Switzerland (2016). isbn: 978-3-319-32550-7
- [13] Boyd, S., Vandenberghe, L.: Convex Optimization, 7th edn. Cambridge University Press, Cambridge (2009). isbn: 978-0-521-83378-3
- [14] Lobo, M.S., Vandenberghe, L., Boyd, S., Lebre, H.: Applications of second-order cone programming. Linear Algebra and its Applications **284**, 193–228 (2024). doi: [https://doi.org/10.1016/S0024-3795\(98\)10032-0](https://doi.org/10.1016/S0024-3795(98)10032-0)
- [15] Pandala, A.G., Ding, Y., Park, H.-W.: qpSWIFT: A real-time sparse quadratic program solver for robotic applications. IEEE Robotics and Automation Letters **4**, 3355–3362 (2019). doi: <https://doi.org/10.1109/LRA.2019.2926664>
- [16] Beeson, P., Ames, B.: TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In: IEEE RAS Humanoids Conference, Seoul, Korea (2015). doi: <https://doi.org/10.1109/HUMANOIDS.2015.7363472>
- [17] Alexander Domahidi, S.B. Eric Chu: ECOS: An socp solver for embedded systems. In: 2013 European Control Conference (ECC), pp. 3071–3076 (2013). doi: <https://doi.org/10.23919/ECC.2013.6669541>