Rotorcraft guidance with sampling based model predictive control

Alexej Dikarew
Research Engineer
German Aerospace Center (DLR)
Braunschweig, Germany

Tobias Winkler
Research Engineer
German Aerospace Center (DLR)
Braunschweig, Germany

ABSTRACT

A key technology in automation of rotorcraft flight is collision-free guidance. Especially in operations close to the ground, detection and automatic avoidance of ground-based obstacles and aircraft is a demanding task. This paper presents a sampling-based model predictive approach for collision-free guidance of rotorcraft. The method calculates control inputs for the flight controller by predicting the closed-loop dynamics of the rotorcraft for a short time horizon and evaluating the predictions with a cost function, which can take an arbitrary form. The approach implements a simple algorithm, mitigating the need for iterative optimization and allowing for deterministic execution time. The cost function is set to ensure collision-free maneuvering while following a desired path, as well as considering constraints of the rotorcraft states and control inputs. The path following performance is tested in closed-loop simulations with a non-linear helicopter model. The algorithm is implemented on a graphics processing unit for parallel execution, strongly decreasing the computation time.

NOTATION

$\boldsymbol{A}$	Stability derivative matrix
$\boldsymbol{B}$	Control derivative matrix
c	Scaling factor for control input sampling
d	Distance (m)
d_f	Collision cost fade-in distance (m)
d_s	Safety margin (m)
h	Integration step size
i	Control input sample index
J	Cost function value
k	Prediction step index
M	Prediction model function
n_p	Prediction horizon
$\boldsymbol{P}_{obs}$	Obstacle position (m)
p	Roll rate in body frame (deg/s)
q	Pitch rate in body frame (deg/s)
r	Yaw rate in body frame (deg/s)
S	Control input sampling function
t	Time (s)
$\boldsymbol{u}$	Upper mode command vector
u_f	Velocity along body frame x-axis (m/s)
u_v	Velocity along vertical frame x-axis (m/s)
v_f	Velocity along body frame y-axis (m/s)
v_v	Velocity along vertical frame y-axis (m/s)
W	Cost function weighting factor
w_f	Velocity along body frame z-axis (m/s)
w_v	Velocity along vertical frame z-axis (m/s)
$\boldsymbol{x}$	State vector
$\boldsymbol{x}_{cmd}$	Commanded state vector
$\boldsymbol{x}_m$	Measured state vector
$\boldsymbol{x}_{mo}$	Model output state vector
$\boldsymbol{x}_{tc}$	State vector correction for turn coordination
$\boldsymbol{x}_{trim}$	State vector at trimmed flight state
x	Position in local NED frame x-axis (m)
y	Position in local NED frame y-axis (m)
z	Position in local NED frame z-axis (m)
$\boldsymbol{\delta}$	Control input vector (%)
δ	Control input in one control axis (%)
δ_c	Current value of control input (%)
$\boldsymbol{\delta}_s$	Set of sampled control input values
ϕ	Roll angle (deg)
θ	Pitch angle (deg)
ψ	Yaw angle (deg)

Presented at the Vertical Flight Society's 80th Annual Forum & Technology Display, Montréal, Québec, Canada, May 7–9, 2024.

INTRODUCTION

Rotorcraft are used in a broad field of applications due to their ability of hovering flight. Often, the missions are carried out close to the ground and include approaches and landing on unprepared and unknown sites, resulting in high workloads for the crew. This puts the aircraft at high risk of collision with ground-based obstacles and other aircraft. Recent safety reports of the european aviation safety agency (EASA) show that inadequate separation to other aircraft and inadequate obstacle clearance are a major source of accidents and serious incidents in helicopter operations (Refs. 1–3). Especially, the number of near-miss incidents and collisions with small UAVs strongly increased in the past decade. These UAVs are not covered by air traffic control and pose a potential threat to aircraft operating in low altitude, for example helicopter emergency medical services (HEMS). One focus of rotorcraft research is therefore to increase the pilot's situational awareness by detecting obstacles in the vicinity of the rotorcraft with appropriate sensors, e.g. cameras, radar and light detection and ranging (LiDAR). As more information to the pilot tends to increase the workload, another trend in research is the development of advanced automatic flight control systems that support the pilot in high workload situation, e.g. in obstacle avoidance.

Detection and automatic avoidance of obstacles for rotorcraft is being investigated by various authors. Scherer et. al. proposed a system for autonomous flight of a small unmanned helicopter in an unknown environment (Ref. 4). The system consists of a laser scanner for obstacle detection, a path planning algorithm and a reactive collision avoidance method. The control commands are calculated geometrically by evaluating the angles and distances to obstacles and the goal point, respectively. Flight tests with a full-scale helicopter showed the capability of avoiding static obstacles at lower flight speed (Ref. 5).

Further significant research in the field of automatic helicopter guidance has been carried out by Goerzen and his colleagues. They presented a system for obstacle field navigation that provides helicopter guidance in low altitude flight in an unknown environment based on sensor data (Ref. 6). The capabilities of the system have been demonstrated in extensive flight tests on helicopters with both full authority (Ref. 7) and partial authority (Ref. 8) flight control systems. In their approach, a path to the goal location is generated by creating a risk-based map of the terrain and obstacles in the vicinity of the rotorcraft and applying a navigation function that provides a minimum-risk path. In order to command the helicopter to the goal, a velocity command controller is used that follows the navigation function. A simple model based on fixed acceleration limits is utilized to generate feasible velocity commands. In recent work (Ref. 9), the authors presented an improved method to generate the velocity commands. For this approach, a set of reachable flight states is estimated by calculating constant turn-radius paths for different lateral accelerations. The trajectory with the lowest risk is chosen and the corresponding command is applied.

An approach to explicitly consider system dynamics and constraints in trajectory generation is given by model predictive control (MPC). MPC, or receding horizon control, is an optimal control method where the dynamics of the system are included as a prediction model in an iterative optimization to find control inputs that minimize a cost function. Depending on the complexity of the prediction model, the number of constraints and the length of the time horizon, MPC in general introduces high computational effort to evaluate the control law. Nonlinear MPC has been used for collision-free guidance of an unmanned helicopter (Ref. 10), however a simple nonlinear translational model has been used in order to reduce the computational cost.

To overcome the disadvantages of MPC, model predictive path integral (MPPI) has been proposed by Williams et. al. (Ref. 11). For this method, a finite number of trajectories is being sampled by generating random deviations of the control input and applying them to the model over a short prediction horizon. The optimal trajectory is chosen by evaluating a cost function and the control sequence corresponding to the optimal trajectory is set as input to the system. The trajectory sampling reduces optimization iterations and allows to execute a large part of the computations in parallel, e.g. on graphics processing units (GPUs). Recently, the MPPI method has been adopted in different fields of aerospace research. Comandur et. al. use the approach for optimizing trajectories in automatic helicopter ship deck landings in order to compensate for random ship movement (Refs. 12–14). In (Ref. 15), the method is applied for controlling an aircraft on a racecourse, while (Ref. 16) proposes MPPI for collision avoidance of a fixed-wing aircraft.

Another sampling-based model predictive approach is given by sampling a set of predicted trajectories based on constant control inputs instead of using random deviations. For that, a set of values is sampled from the allowed range and fed into

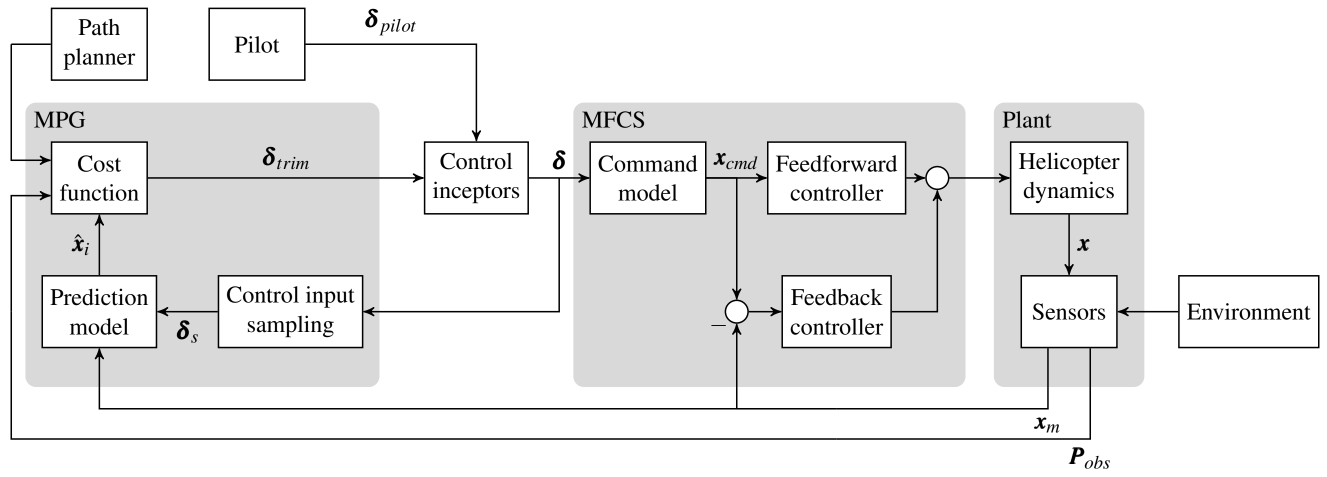


Figure 1: Architecture of sampling-based model predictive guidance system

the prediction model, being held constant for each trajectory over the prediction horizon. This method has been proposed for collision avoidance in automotive applications (Refs. 17, 18).

The authors of the paper at hand adapted this method to short-term collision avoidance for rotorcraft (Refs. 19, 20). The approach is characterized by a simple algorithm, which the authors believe is beneficial in aircraft applications. Also, the method provides deterministic computation time as no iterative optimization is utilized, which is favorable for the execution under real-time constraints. Compared to MPPI, fewer trajectories need to be predicted, reducing the computational effort. While the authors' previous publications on the topic focused on collision avoidance, the paper at hand describes the status of work on extending the method to provide guidance of the helicopter along planned flight paths, which are generated by a separate path planning algorithm.

The remaining paper will be structured as follows: First, the proposed system for collision-free guidance and its components are described in detail with the extensions and adjustments made compared to the initial publications. Then, the performance of the system for path following is evaluated in closed-loop simulations. The paper closes with a conclusion of the findings and an outlook to future work.

SAMPLING-BASED MODEL PREDICTIVE GUIDANCE APPROACH

The proposed model predictive guidance (MPG) system is intended to guide a helicopter along a desired path while avoiding obstacles. The approach builds upon an underlying flight controller, that provides stabilization and axis decoupling. The implementation presented in this paper utilizes the model following controller of german aerospace center's (DLR's) research helicopter active control technology/flying helicopter simulator (ACT/FHS) (Refs. 21–23) as the underlying flight controller, but any flight control system that accounts for stabilization and axis decoupling is suitable for the method.

The overall MPG system architecture is depicted in Fig. 1. The algorithm calculates inputs for the flight controller that provide collision-free guidance. To achieve this, an approximation of the closed-loop helicopter flight dynamics is predicted from the current flight state over a short time horizon for a set of constant control inputs. The resulting set of predicted trajectories is evaluated by applying a suitable cost function. The control input corresponding to the trajectory with lowest cost is fed to the underlying flight controller for the current time step. In the next time step, the initial flight state for prediction is updated with the currently measured flight state of the helicopter and the algorithm is executed again. By applying this feedback, a closed-loop control system for collision-free guidance is established. A programmatic overview of the method is given in algorithm 1.

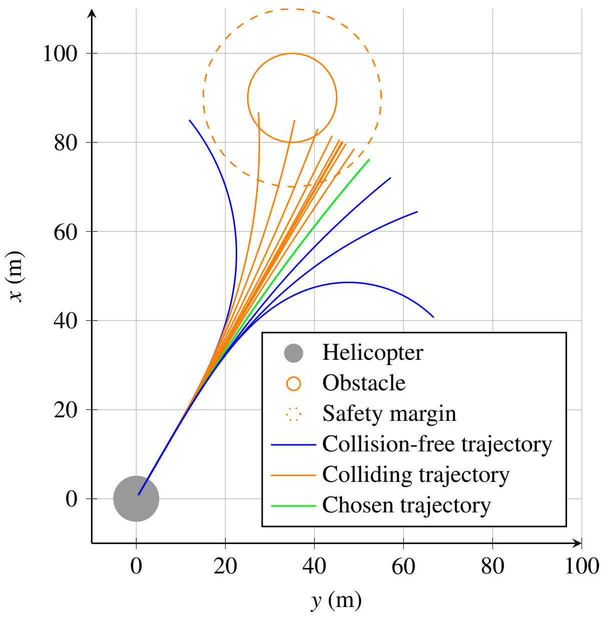
The trajectory prediction and selection is illustrated in figures 2a and 2b. Figure 2a provides a top-down view of a situation with an obstacle in front of the helicopter. The helicopter

Algorithm 1 Model predictive guidance algorithm

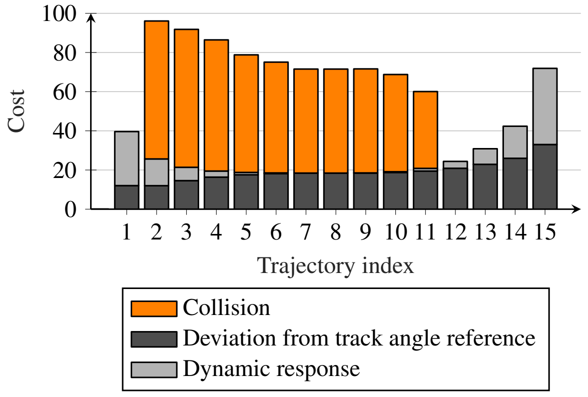
```
 1: Given:
 2: $J$:                                         ▷ cost function value
 3: $\delta_c$:                                  ▷ current control input value
 4: $S$:                                         ▷ control input sampling function
 5: $n_\delta$:                                  ▷ number of control input samples
 6: $\boldsymbol{x}_m$:                          ▷ currently measured state
 7: $n_p \leftarrow t_p/h$:                      ▷ number of predicted time steps
 8: $\hat{\boldsymbol{x}}_k$:                    ▷ predicted state
 9: $t_p$:                                       ▷ prediction horizon
10: $h$:                                         ▷ prediction step size
11: $M$:                                         ▷ prediction model
12: $\delta_{opt}$:                              ▷ optimal control input value
13:
14: repeat
15:     $J_{min} \leftarrow$ inf
16:     $\boldsymbol{\delta}_s \leftarrow S(\delta_c)$          ▷ generate control input set
17:     for $i \leftarrow 1$ to $n_\delta$ do                   ▷ for each trajectory
18:         $\boldsymbol{x}_1 \leftarrow \boldsymbol{x}_m$
19:         for $k \leftarrow 1$ to $n_p$ do                    ▷ for each prediction step
20:             $\hat{\boldsymbol{x}}_{k+1} \leftarrow M(\hat{\boldsymbol{x}}_k, \delta_{s,i}, h)$   ▷ predict state
21:             $J_i \leftarrow J_i + J(\hat{\boldsymbol{x}}_{k+1})$                 ▷ evaluate costs
22:         end for
23:         if $J_i < J_{min}$ then
24:             $\delta_{opt} \leftarrow \boldsymbol{\delta}_s(i)$      ▷ update best control
25:             $J_{min} \leftarrow J_i$
26:         end if
27:     end for
28:     send $\delta_{opt}$
29: update $\boldsymbol{x}_m$                                  ▷ update measured state
```

dynamics are predicted for different input values in the lateral cyclic control axis. The colors of the resulting trajectories indicate if a collision with the obstacle or the safety margin is predicted. In figure 2b, the corresponding cost function values are displayed. The trajectory index is plotted on the x-axis, with index 1 corresponding to the the leftmost trajectory and index 15 corresponding the rightmost trajectory. The cost is plotted on the y-axis. In this example, the cost function holds three components. First, colliding with the obstacle is penalized (orange). Second, a deviation from a reference track angle of zero, i.e. along the x-axis, generates costs (dark grey) and third, the dynamic response of the helicopter shall be minimized (light grey). By summing up the cost components for each tracjectory, the final cost function value is found. In this example, the trajectory passing the obstacle on the right side (index 12) has lowest costs and is selected by the algorithm.

The algorithm provides control inputs, i.e. stick deflections, that correspond to the optimal predicted trajectory. The stick deflections are fed to the control inceptors as trim commands. When the system operates autonomously, i.e. when the pilot is not providing stick inputs, the control inceptor positions will correspond to the trim commands, hence the optimal control input calculated by the algorithm is fed to the flight controller. While the pilot is hands-on, he will receive haptic feedback of the algorithm's chosen control input via the stick deflection. In addition, the pilot may override the MPG output and fly



(a) Predicted trajectories in example situation



(b) Cost function value per trajectory

Figure 2: Trajectory prediction and selection

manually by moving the control inceptors out of the trim position. The algorithm will continue to provide trim commands while the pilot is hands-on, hence the pilot can detect the direction and amplitude of the algorithm's control input through the trim forces. When the pilot releases the control inceptors, their position will again correspond with the trim commands provided by the MPG and the system will continue with automatic guidance of the rotorcraft.
This design shall allow for the pilot to interact with the automation without the need for disabling and re-enabling the automatic guidance system. In addition, it uses haptic cueing to provide awareness of the automation's state. The MPG system aims at contribtung to DLR's ongiong research in the field of pilot interaction with highly automated rotorcraft in multimodal cueing environments (Ref. 24).

The following sections will give a detailed descriptions of the further development of the three main components of the algorithm: control input sampling, prediction model and cost function.

Control input sampling

To calculate a set of predicted trajectories, a suitable set of corresponding control inputs needs to be generated. In the proposed approach, this is done by sampling a set of discrete values out of the continuous range of possible control inputs δ. The sampled set $\boldsymbol{\delta_s}$ shall cover the full range of control authority and therefore generate predictions which represent the full bandwidth of possible maneuvers. Also, the distance between neighboring values should be small to achieve a high resolution. To reduce the total number of control samples and hence the number of predicted trajectories and computational cost, the control inputs are sampled with a non-linear distribution that adapts to the current input value δ_c. A cubic function is applied to provide high resolution in the vicinity of the current control input value δ_c while also covering the full range of δ. The sampling method is based on (Ref. 18) and described in Refs. 19, 20. The previously used sample distribution would, however, place half of the samples left of δ_c and the other half right of it. When δ_c is close to the minimum value δ_{min} or maximum value δ_{max} of δ, the overall resolution would decrease. Therefore, the sampling function was slightly adapted and now reads as

$$\boldsymbol{\delta}_s = \begin{cases} \dfrac{\delta_{min}-\delta_c}{(1-i_c)^3}\cdot(i-i_c)^3+\delta_c & \text{for } i<i_c \\ \dfrac{\delta_{max}-\delta_c}{(n_\delta-i_c)^3}\cdot(i-i_c)^3+\delta_c & \text{for } i>i_c \\ 0 & \text{for } i=i_c\ , \end{cases} \tag{1}$$

with n_δ being the number of control input samples and i_c being the index that the current control input δ_c is mapped to. It is defined by

$$i_c = \begin{cases} \lfloor c \rfloor & \text{for } c > \left\lfloor \dfrac{n_\delta}{2} \right\rceil \\ \lceil c \rceil & \text{for } c \le \left\lfloor \dfrac{n_\delta}{2} \right\rceil\ , \end{cases} \tag{2}$$

with

$$c = \frac{\dfrac{n_\delta - 1}{2}}{\left(\dfrac{\delta_{max}-\delta_{min}}{2}\right)^3}\cdot\left(\delta_c - \frac{\delta_{max}-\delta_{min}}{2} - \delta_{min}\right)^3 + \left\lfloor \frac{n_\delta}{2} \right\rceil\ . \tag{3}$$

Equations 2 and 3 basically adapt the position of i_c inside the range of n_δ. Figure 3 shows an example of the applied sampling method. The plot displays the value distributions for $\delta_c = 0\%$ and $\delta_c = 40\%$ respectively, with $\delta_{min} = -50\%$ and $\delta_{max} = 50\%$. The number of samples is set to $n_\delta = 15$. As can be seen, the area with highest density (highlighted areas in the plot) is located around the respective current value δ_c. For $\delta_c = 40\%$, the index of δ_c is shifted to $i_c = 11$, to increase the resolution of samples for $\boldsymbol{\delta}_s < 40\,\%$. The control input sampling method is executed for each control axis at every time step of the MPG algorithm. To account for the dynamics of the control inceptors (see figure 1), the set of control inputs is fed into a model of the control inceptors that represents the

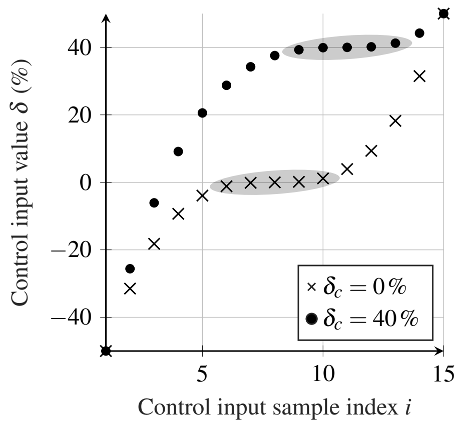


Figure 3: Control input sampling

stick dynamics as a second order system with specific frequency and damping. The set of responses of the inceptor model for the input $\boldsymbol{\delta}_s$ is then fed as input to the prediction model to generate a set of predicted trajectories.

Prediction model validation

The proposed model predictive guidance algorithm strongly relies on a mathematical model that provides predictions of the aircraft dynamics based on given inputs. As the algorithm is used for short-term collision-free guidance, the predictions need to be as accurate as possible. On the other hand, the model needs to be computationally inexpensive as the algorithm needs to run under real-time constraints and the model is repeatedly evaluated for each predicted trajectory.
The prediction model needs to represent the closed-loop dynamics of the rotorcraft and its flight control system. For flight control design, models of the bare airframe dynamics are typically made available by physical modeling or system identification. One would need to combine these models with the dynamics of the flight controller or generate a closed-loop approximation to gain a suitable prediction model for the proposed guidance algorithm.

A simplification in the synthesis of the prediction model arises if the flight controller utilizes a model following control scheme. Provided that the controller tracks the reference model with sufficient performance, the closed-loop response of the aircraft corresponds with the controller's reference model. Therefore, the reference model may be used as the prediction model.
The flight control system of the ACT/FHS research helicopter utilizes such a model following control scheme. Here, the pilot commands a stable helicopter model that provides axis decoupling and dynamics with desired handling qualities according to the detail specification handling qualities for military rotorcraft (MIL-DTL-32742) (Ref. 25), which is the successor of the aeronautical design standard 33 performance specification handling qualities requirements for military rotorcraft (ADS-33E-PRF). This command model (CM) calculates reference states x_{cmd} based on the pilot input that are being tracked by both feedforward and feedback controllers, see figure 1. A detailed description of the CM used in the flight controller of the ACT/FHS can be found in Refs. 21 and 22.

For the implementation in this paper, the CM is being used as the algorithm's prediction model. For the implementation of the prediction model, several modifications and simplifications compared to the original CM have been applied, a complete description of the prediction model can be found in Refs. 19 and 20. The resulting prediction model is a nonlinear six degree-of-freedom model accounting for rigid-body dynamics. The state-space equations are given by

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + f\left(\boldsymbol{x}+\boldsymbol{x}_{tc}+\boldsymbol{x}_{trim}\right) - f(\boldsymbol{x}_{trim}) + \boldsymbol{B}\boldsymbol{u}\ ,\\ \boldsymbol{x}_{mo} &= \boldsymbol{x}+\boldsymbol{x}_{tc}+\boldsymbol{x}_{trim}\ ,\end{aligned} \tag{4}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ denote the stability derivative and control derivative matrices, $\boldsymbol{x} = \left[u_f\ v_f\ w_f\ p\ q\ r\right]^T$ denotes the state vector in body frame of reference with the velocities u_f, v_f, w_f and angular rates p, q, r. The trim velocities are based on recorded trim states of the ACT/FHS research helicopter and are incorporated by $\boldsymbol{x}_{trim}$, while $f(\boldsymbol{x}+\boldsymbol{x}_{tc}+\boldsymbol{x}_{trim})$ and $f(\boldsymbol{x}_{trim})$ account for the gravitational forces at the flight states $\boldsymbol{x}+\boldsymbol{x}_{tc}+\boldsymbol{x}_{trim}$ and $\boldsymbol{x}_{trim}$, respectively. Coordinated turns with $\dot{v}_f = 0$ are achieved by applying a correction of angular rates $\boldsymbol{x}_{tc}$. The model output state vector accounting for trim angles and turn coordination is given by $\boldsymbol{x}_{mo}$. The vector $\boldsymbol{u}$ incorporates the control inputs and is being generated by the control response modes, which calculate velocity and angular rate commands based on the control input $\boldsymbol{\delta}$. For cyclic inputs, the attitude command attitude hold (ACAH) mode has been implemented. It commands pitch angles proportional to the longitudinal cyclic stick deflection and roll angles proportional to the lateral cyclic stick deflection, respectively. The yaw axis incorporates a rate command direction hold (RCDH) mode for low speed flight. For high speed flight, the model provides turn coordination (TC) when no pedal input is given. For the heave axis, the rate command height hold (RCHH) mode is implemented, which commands a vertical velocity proportional to the collective stick deflection and tracks the current altitude when no input is given.
To reduce the computational effort in solving the model's differential equations, the forward Euler method is used. Applying the solver to equation 4 yields

$$\begin{aligned}\hat{\boldsymbol{x}}(k+1) &= \hat{\boldsymbol{x}}(k)\\ &\quad + h\cdot\Bigg(\boldsymbol{A}\hat{\boldsymbol{x}}(k) + f\Big(\hat{\boldsymbol{x}}(k)+\hat{\boldsymbol{x}}_{tc}(k)+\hat{\boldsymbol{x}}_{trim}(k)\Big)\\ &\quad - f\left(\hat{\boldsymbol{x}}_{trim}(k)\right) + \boldsymbol{B}\hat{\boldsymbol{u}}(k)\Bigg)\ ,\\ \hat{\boldsymbol{x}}_{mo}(k) &= \hat{\boldsymbol{x}}(k)+\hat{\boldsymbol{x}}_{tc}(k)+\hat{\boldsymbol{x}}_{trim}(k)\ ,\end{aligned} \tag{5}$$

with h denoting the integration step size, k denoting the prediction step at time t_k and $k+1$ denoting the subsequent prediction step at $t_{k+1} = t_k + h$. The notation with hat symbol denotes quantities calculated with the solver, i.e. predictions. Besides the body frame velocity and angular rates, additional states are calculated at each prediction step. This includes Euler angles, the aircraft position in the north-east-down (NED) frame of reference and additional velocities in the vertical frame of reference, for which the z-axis is aligned with the local NED z-axis and the x-axis is horizontally aligned with the body x-axis. The overall predicted state vector at prediction step k reads as $\hat{\boldsymbol{x}}(k) = \left[\hat{u}_f\ \hat{v}_f\ \hat{w}_f\ \hat{p}\ \hat{q}\ \hat{r}\ \hat{x}\ \hat{y}\ \hat{z}\ \hat{\phi}\ \hat{\theta}\ \hat{\psi}\ \hat{u}_v\ \hat{v}_v\ \hat{w}_v\right]^T$. With equation 5, the prediction model dynamics are being computed from $k = 1$, with $\hat{\boldsymbol{x}}(k = 1) = \boldsymbol{x}_m$ holding the measured state vector, to $k = n_p$ with $n_p = t_p/h$ holding the prediction horizon. Performing this calculations for the set of sampled control input values $\boldsymbol{\delta}_s$ provides the set of predicted trajectories $\hat{\boldsymbol{x}}$.

The MPG system relies on the prediction model to closely resemble the closed-loop system response of the controlled helicopter. However, there are different sources that introduce error to the model's output.
First, the model assumes that the flight controller perfectly tracks the commanded states. Controller tracking errors will therefore result in errors of the prediction model.
Second, the prediction model introduces simplifications of the baseline reference model of the controller and therefore introduces modeling errors and errors due to the numerical solver.
Third, external disturbances, e.g. the influence of wind, will cause the actual response of the helicopter to diverge from the predictions.
The control input calculated by the algorithm is updated at each cycle by executing the algorithm with the currently measured flight state of the rotorcraft. This feedback loop reduces the influence of prediction model errors to the overall system performance. Nonetheless, a validation of the prediction model was carried out to quantify the error introduced by modeling and controller tracking errors.
To assess the quality of the model's predictions, its output was compared to reference data. The reference data is generated by running simulations of a non-linear physics-based model of the helicopter in closed-loop with the flight controller. The same model is used for real-time simulation of the research helicopter ACT/FHS at DLR's full-flight air vehicle simulator (AVES). The closed-loop response was recorded for various flight states: the initial flight speed was varied in the range of $30\,\mathrm{m/s}$ to $45\,\mathrm{m/s}$, the initial roll angle was varied from $-30°$ to $30°$ and the initial rate of climb was varied from $-3\,\mathrm{m/s}$ to $3\,\mathrm{m/s}$. For each flight condition, control inputs with different amplitude are fed to the controller in the longitudinal cyclic, lateral cyclic and heave axes. This results in approximately 600 test points for validation.
To compare the validation data to the prediction model output, the prediction horizon needed to be defined. For the application as an collision avoidance system, the prediction horizon needs to cover the time that is needed to decelerate the rotorcraft from its current air speed to hover, in case that avoiding a collision through lateral or vertical maneuvers is not reasonable. The validation data shows that the controller is capable of decelerating the helicopter from $45\,\mathrm{m/s}$ air speed to hover in about $9\,\mathrm{s}$. To cover this maneuver, the prediction horizon for validation was defined as $t_p = 10\,\mathrm{s}$.
Next, the integration step size for the solver was defined. Therefore, its output was calculated for the validation test points for different step sizes and the root mean square error (RMSE) to the validation data was calculated. Figure 4 shows a box plot of the prediction model RMSE of the pitch rate $\hat{q}$ over all validation test points for integration step sizes $8\,\mathrm{ms} < h < 160\,\mathrm{ms}$. The pitch rate was chosen as its RMSE was found to be most sensitive to variations of h compared to the other states. While the plot shows an increase in RMSE

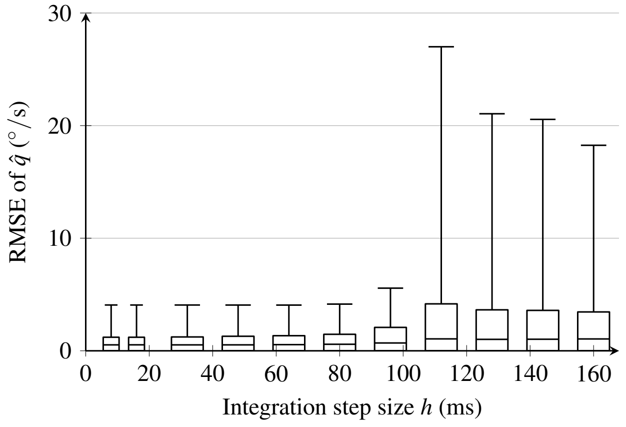


Figure 4: Influence of integration step size h on RMSE

between $h = 80\,\mathrm{ms}$ and $h = 120\,\mathrm{ms}$, the error does not monotonically increase as one could expect. Also, the prediction model output shows strong oscillations due to undersampling for higher integration step sizes, which are not visible in the RMSE. Therefore, the model output was investigated in the frequency domain as well in order to identify the oscillations. Figure 5 shows the frequency spectrum of the prediction model's pitch rate $\hat{q}$ for different integration step sizes, normalized to the sampling frequency $f_s = 1/h$. The plot shows

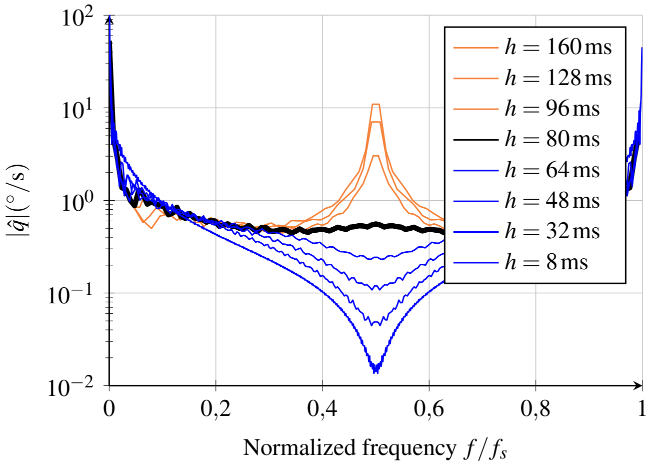


Figure 5: Spectrum of prediction model output for $\hat{q}$

a peak at $f/2$ for $h > 80\,\mathrm{ms}$, indicating oscillations due to undersampling.
As the RMSE is not significantly lower for smaller values of h, and larger values introduce undersampling effects, an integration step size $h = 80\,\mathrm{ms}$ is used for the prediction model.

With the prediction horizon t_p and the integration step size h set, the prediction model's output was compared to the validation data. It was found, that in order to reduce the controller's tracking error, the range of allowed control input needs to be limited. After limiting the control input amplitude, the system is able to provide inputs that allow for pitch angles of $-12° < \theta < 35°$, roll angles of $-45° < \phi < 45°$ and rates of descent of $-10\,\mathrm{m/s} < w_v < 13\,\mathrm{m/s}$.

To assess the prediction model's accuracy, a point of reference for the RMSE is needed. For validation of rotorcraft flight mechanics models gathered from system identification, an acceptable model fit is achieved if RMSE < 2 (Ref. 26). Although the prediction model is not used for flight mechanics analysis, the value is used as guidance to estimate the quality of the predictions. The RMSE values of the body velocities $\hat{u}_f$, $\hat{v}_f$, $\hat{w}_f$ and angular rates $\hat{p}$, $\hat{q}$, $\hat{r}$ for all test points together with the guidance value are displayed in figure 6. Velocities are measured in knots and angular rates are measured in deg/s to match the units of measurement proposed in (Ref. 26). The

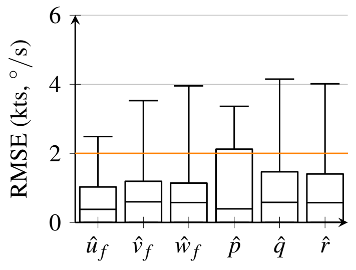


Figure 6: Prediction model RMSE

plot shows that the RMSE values of the prediction model are within the order of magnitude of the guidance threshold for all test points. In addition, the 75 % quartile of all body velocities and angular rates except the roll rate $\hat{p}$ are well within the threshold, indicating good agreement of the prediction model's output with the validation data for the majority of test points.

In addition to the velocities and angular rates, the RMSE values of the flight mechanic quantities used in the algorithm's cost function are analyzed. Figure 7 shows the RMSE values over all test points for the angular rates $\hat{p}$, $\hat{q}$, $\hat{r}$, the Euler angles $\hat{\phi}$, $\hat{\theta}$, $\hat{\psi}$, the forward and vertical velocities in the vertical frame of reference $\hat{u}_v$, $\hat{w}_v$ and the position in space $\hat{P}$, which is computed from the states $\hat{x}$, $\hat{y}$, $\hat{z}$. In contrast to figure 6, the velocities are measured in m/s. The maximum RMSE for all states besides the heading angle $\hat{\psi}$ and the position $\hat{P}$ is within a value of 5 in the respective unit of measurement. For $\hat{\psi}$ and $\hat{P}$ the maximum RMSE is within 9 deg and 9 m, respectively. The larger error in the heading and position error is caused by the utilized flight controller modes. By using the ACAH and RCHH control response types, no tracking of the heading angle and position is established, resulting in larger error compared to the other states. Nonetheless, the results show satisfactory agreement of the prediction model's output with closed-loop responses of the helicopter for the relevant states.

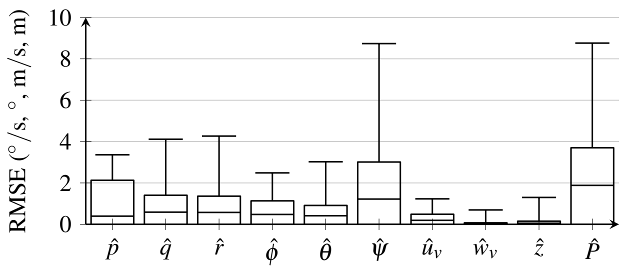


Figure 7: Prediction model RMSE

Therefore, the prediction model is considered to provide sufficiently reliable predictions to be used in the MPG system.

Cost function design

For collision-free guidance of the rotorcraft, at each time step the most suitable trajectory in the set of predictions needs to be identified. This is realized by evaluating a cost function for the prediction data. The trajectory with the smallest cost is considered to be optimal and the corresponding control input δ_{opt} is send to the control inceptors. The original cost functions from Refs. 19 and 20 have been adapted to, besides avoiding collisions and following reference states, account for state and control input constraints. The resulting cost function reads as

$$J = \sum_{k=1}^{n_p} \left(W_1 J_{1,k} + \boldsymbol{W}_2 \cdot \boldsymbol{J}_{2,k} + \boldsymbol{W}_3 \cdot \boldsymbol{J}_{3,k} + \boldsymbol{W}_4 \cdot \boldsymbol{J}_{4,k} \right) . \quad (6)$$

The cost function value is calculated for each prediction step k and summed up over the prediction horizon n_p to obtain the overall cost for each trajectory respectively. $J_1, \boldsymbol{J}_2, \boldsymbol{J}_3, \boldsymbol{J}_4$ hold the cost function components, which will be explained in the following. $W_1, \boldsymbol{W}_2, W_3, \boldsymbol{W}_4$ hold the weights for each corresponding component.

The first component J_1 accounts for avoiding collisions. Therefore, at each prediction step the distance $d_{obs,k}$ between the rotorcraft's predicted position $\hat{\boldsymbol{P}}_k = \left[\hat{x}_k \; \hat{y}_k \; \hat{z}_k\right]$ and the obstacle's position $\hat{\boldsymbol{P}}_{obs,k}$ is calculated. Depending on the distance towards an obstacle, the collision avoidance component is given by

$$J_1 = \begin{cases} 2 - \dfrac{1}{{d_s}^2} {d_{obs,k}}^2 & \text{if } d_{obs,k} \leq d_s \; , \\ \dfrac{\left(d_{obs,k} - (d_s + d_f)\right)^2}{{d_f}^2} & \text{if } d_s < d_{obs,k} < d_s + d_f \; , \\ 0 & \text{if } d_{obs,k} \geq d_s + d_f \; , \end{cases} \quad (7)$$

and is plotted in figure 8. If the predicted distance to the obstacle $d_{obs,k}$ is less than the minimum safety distance d_s, a collision is predicted and the cost value is set to a value greater than 1. A quadratic function is applied that increases the cost value with decreasing distance. This ensures that trajectories with higher distance to the obstacle are chosen when all predicted trajectories collide with the safety margin. If the predicted position of the helicopter is not colliding but is within a certain range to the obstacle $d_s < d_{obs,k} < d_s + d_f$, another quadratic function is applied that generates rising cost with decreasing distance. The range for this fade-in of the cost has been set to $d_f = 5$ m. This measure removes sharp discontinuities of the cost value in the vicinity of the obstacle, which caused undesired behavior in early experiments, i.e. oscillations of the control inputs. Finally, if the rotorcraft's position is outside the fading distance $d_{obs,k} \geq d_s + d_f$, the cost is set to $J_1 = 0$.

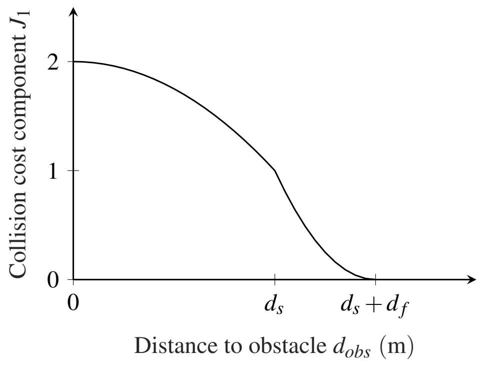


Figure 8: Collision cost dependent on distance to obstacle

To safely avoid collisions, the error in the rotorcraft's predicted position needs to be considered in the cost function. In figure 7 the RMSE of the position prediction is analyzed, for collision avoidance however, the maximum error needs to be taken into account. Figure 9 shows the maximum error in x, y, z and the resulting position P for all validation test points as a function of the prediction time t_p. The error in position increases approximately linearly with increasing time. Therefore, a linear scheduling of the minimum safety distance d_s is added to the calculation of J_1 to account for the position error.

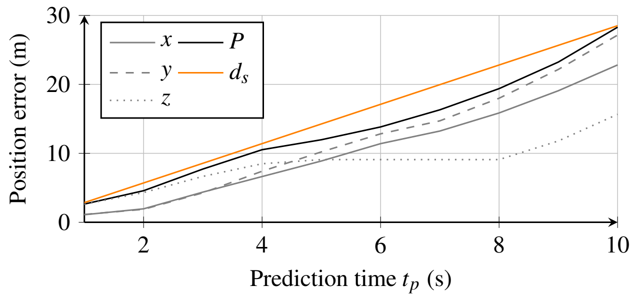


Figure 9: Scaling of safety distance d_s

The second cost function component introduces soft constraints on the predicted states and the control input rates to limit the dynamic response of the aircraft. The constraint function is implemented as

$$f_c(s, s_{min}, s_{max}) = max\{s_{min} - x, 0, s - s_{max}\} , \quad (8)$$

with s_{min} and s_{max} representing lower and upper bounds of the state or control input rate s. The constraint function is applied to the state predictions of $\hat{p}$, $\hat{\phi}$ and $\hat{\theta}$ and the control input rates $\dot{\delta}_x$, $\dot{\delta}_y$, $\dot{\delta}_0$. Therefore J_3 reads as

$$\boldsymbol{J}_2 = \left[f_c(\hat{p}) \; f_c(\hat{\phi}) \; f_c(\hat{\theta}) \; f_c(\dot{\delta}_x) \; f_c(\dot{\delta}_y) \; f_c(\dot{\delta}_0)\right] . \quad (9)$$

The third cost function component accounts for tracking reference states that are needed to guide the rotorcraft along a path. The planned path is represented by a list of waypoints, which hold information of position and desired velocity at each point. The waypoint information is transformed into reference values for the airspeed $u_{v,ref}$, track angle χ_{ref} and altitude z_{ref}. The reference track angle is calculated based on the position of the helicopter related to the position of the waypoint. To proceed to the upcoming waypoint when the current one is reached, the distances to both the current and the upcoming waypoint are calculated. When the distance to the current waypoint increases while the distance to the upcoming waypoint decreases, the helicopter is considered to has passed the current waypoint and the reference states are updated with information from the upcoming waypoint. The cost function component $\boldsymbol{J}_3$ penalizes deviations of the predictions from the reference states and reads

$$\boldsymbol{J}_3 = \begin{cases} \left[|\hat{u}_v - u_{v,ref}| \; |\hat{\psi} - \chi_{ref}| \; |\hat{z} - z_{ref}|\right] & \text{if } t_p \leq t_{ref} , \\ \left[0\;0\;0\right] & \text{if } t_p > t_{ref} , \end{cases} \quad (10)$$

with t_{ref} being a tunable parameter that allows to set the maximum prediction horizon that is considered for reference state tracking.

In order to smooth the control input history generated by the algorithm, the cost function component $\boldsymbol{J}_4$ penalizes changes of the control input, i.e. the control input rates. It reads as

$$\boldsymbol{J}_4 = \left[\dot{\delta}_x \; \dot{\delta}_y \; \dot{\delta}_0\right] . \quad (11)$$

To achieve desired output, the individual cost function components need to be weighted against each other. The different aspects can intuitively be ordered by their prioritization: Avoiding collisions, keeping constraints, track references, minimize control effort. Therefore, the weights can be set accordingly $W_1 >> \boldsymbol{W}_2 >> \boldsymbol{W}_3 >> \boldsymbol{W}_4$, with $\boldsymbol{W}_i >> \boldsymbol{W}_j$ expressing that every element of $\boldsymbol{W}_i$ is greater than any element of $\boldsymbol{W}_j$.

IMPLEMENTATION

The collision-free guidance algorithm was initially implemented for simulations in DLR's full-flight simulator AVES (Ref. 27). The simulator implements a replication of the flight control hardware of the ACT/FHS research helicopter for execution of flight control algorithms. The MPG algorithm was implemented in MATLAB/Simulink and executed on the flight control computer in a single thread. This means, all calculations were carried out sequentially. Refs. 19 and 20 show that the proposed short-term collision avoidance method could be executed only for one axis with this hardware setup. The computational effort of the algorithm strongly depends on the number of predicted trajectories. It was found that 15 trajectories per control axis yield satisfactory results for collision-free guidance. For operation in the longitudinal cyclic, lateral cyclic and collective control axes, the calculation of $15^3 = 3375$ predicted trajectories is necessary. For future investigations, the trajectory sampling method may be adapted to include additional maneuvers, further increasing the total number of predictions and hence the computational effort. However, the calculations for the predicted trajectories are independent of each other, such that the algorithm may benefit from parallel execution.

Amdahl's Law states that the maximum speedup of any algorithm is constrained by the percentage of sequential operations (Ref. 28). Due to the small fraction of the MPG algorithm that needs to execute sequentially, it is well-suited for extensive parallelization. This characteristic makes it particularly effective on parallel computing architectures, such as GPUs. These architectures are highly concurrent systems equipped with multiple cores, referred to as streaming multiprocessors (SMs) by NVIDIA and programmed using the Compute Unified Device Architecture (CUDA). SMs are specifically designed to execute numerous threads in parallel, facilitating efficient parallel processing.
The MPG system described in the previous section and depicted in algorithm 1 was re-implemented utilizing parallelization techniques with the CUDA framework. The new implementation can be divided in four different phases: *fetch*, *update*, *prediction* and *evaluation*. Their concurrent execution timing is displayed in figure 10.

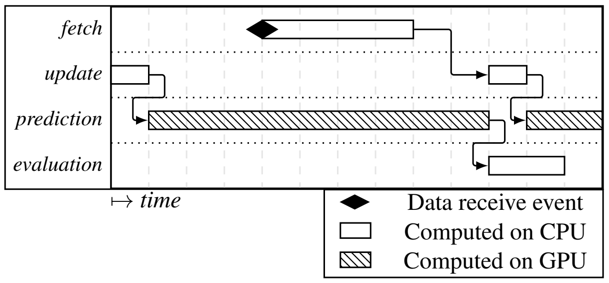


Figure 10: Concurrent execution timing of different phases of the parallel implementation

The *fetch* phase gathers data like the helicopter flight state, the waypoint list from the path planner and data holding information on terrain and obstacle positions. This raw data has to be constantly processed and updated. For example, the terrain data is fused with readings from a LiDAR sensor and stored in an occupancy grid.

The *update* phase copies the data prepared at the *fetch* phase to the GPU for further processing. This phase has to be synchronized with the *prediction* phase to not cause any read-after-write data hazards during the *prediction* phase.

The *prediction* phase takes the currently measured rotorcraft state $\boldsymbol{x}_m$ as input and predicts a set of trajectories based on the set of control inputs $\boldsymbol{\delta}_s$ and the model equation 5. At every prediction step, the cost function (equation 6) is evaluated to calculate the cost value for each individual trajectory.

The *evaluation* phase receives the individual cost function values calculated in the *prediction* phase and identifies the minimum cost trajectory. The corresponding control input value is set as output of the algorithm.

Considerations for efficient implementation

The *fetch* phase is executed concurrently with the *prediction* phase to reduce the overall sequential computation time and to meet real-time requirements. The phase is triggered by data receive events and the most recent processed data is send to the *update* phase after completion of the *prediction* phase in order to minimize latency.

The data processed in the *fetch* phase can be pre-stored in GPU memory during the *prediction* phase, significantly reducing the execution time for the *update* phase. The *update* routine then only needs to update memory references to the new data. To achieve this, a double buffer is employed — a computer programming concept utilizing two buffers to alternate between reading and writing data, allowing simultaneous reading from one buffer while writing to the other. If the *fetch* phase takes longer than a single *prediction* phase, a double buffer is sufficient for the speedup. If two *fetch* phase executions for any given data stream may occur during a single *prediction* phase, data hazards can still arise. For these data streams a triple buffer guarantees the non blocking execution of *fetch* phases and the prevention of data hazards during the *prediction* phase whilst guaranteeing that the most recent data will always be used in the *prediction* phase.

Efficient implementation of the prediction model calculations was attained by storing the state vector $\boldsymbol{x}$ of the model entirely in the registers of each SM. This reduces the delay caused by memory accesses, enhancing computational throughput and minimizing latency and hence reducing the computation time of the *prediction* phase.

Performance evaluation

To assess the performance gain resulting from executing the *prediction* phase in parallel on multiple SMs of a GPU, the execution times of a parallel implementation and a sequential implementation running in a single thread on a central processing unit (CPU) were measured and compared. The assessment was conducted using a NVIDIA Jetson AGX Orin 32GB Development Kit (Ref. 29), which is a board for embedded computing equipped with a NVIDIA GPU. The AGX Orin lacks support for CUDA `concurrentManagedAccess` (Ref. 30), resulting in the necessity to copy data from the *fetch* phase only after the completion of the *prediction* phase. This limitation prevents writing to the GPU memory during the *prediction* phase, leading to an extended execution time for the *update* phase compared to GPUs supporting `concurrentManagedAccess`.

The execution time was assessed by varying the number of predicted trajectories. The results are plotted in figure 11. The GPU exhibits superior speed when computing more than

50 trajectories. For the calculation of 3375 trajectories, which are targeted for the current implementation, the GPU performs the computation in approximately 30 ms, while the CPU requires 2 s for the computation. Leveraging GPUs for the sampling based guidance approach is strongly advised, as their enhanced performance contributes to reduced execution time. Additionally, substantial power savings can be realized, given the GPUs proficiency in handling the required floating-point calculations (Ref. 32).

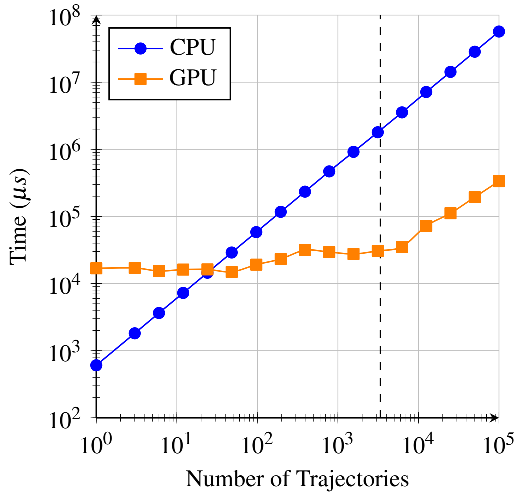


Figure 11: CPU and GPU performance comparison for the *prediction* phase

PATH FOLLOWING PERFORMANCE EVALUATION

In the previous sections, the design and implementation of the sampling-based model predictive guidance algorithm have been described. In this section, the algorithm's capability for path following shall be evaluated. Therefore, software-in-the-loop simulations have been conducted. The simulation is setup according to figure 1. The helicopter flight dynamics are represented by a physics-based nonlinear model of the ACT/FHS research helicopter. It is the same model which was used for the validation of the prediction model. To evaluate the algorithm's path following performance for turns and climb/descent maneuvers, a test path was generated, which is based on (Ref. 8). The maneuver starts with a short straight segment (1) followed by a 180° left turn with 20° roll angle (2), and a climb with 7° slope (3). After the climb, the aircraft shall take a 180° right turn with 20° roll angle (4) to retake the initial heading. After the turn, the aircraft shall descend with a slope of 7° (5) to the initial altitude. The maneuver is concluded by a combination of a right turn of 180° (6) and a left turn of 180° (7), both at 20° roll angle, taking the aircraft back to the position where the maneuver started. After another straight segment, the maneuver is repeated with bank angles of 30° and climb/descent slopes of 9°. The path is represented by points in space. The distance between consecutive points is approximately 40 m. The helicopter shall maintain a speed of 41 m/s during the maneuver, therefore it shall pass a waypoint approximately each second. The complete reference path and the path flown by the helicopter in simulation are plotted in figure 12. To assess the path tracking performance, the distance of the aircraft to the reference path was measured. Figure 13 shows the tracking error for different settings of the reference horizon t_{ref}. A setting of $t_{ref} = 3\,\text{s}$ indicates that only the first 3 s of the predicted trajectories are considered for calculating the reference deviation cost $\boldsymbol{J}_3$, see equation 10. The boxplot shows that the tracking error increases with increasing reference horizon t_{ref}. The reference tracking cost function $\boldsymbol{J}_3$ behaves like a moving average filter: with increasing reference horizon the number of considered waypoints increase and hence the window of the averaging filter. Therefore, increasing the reference horizon decreases the bandwidth of change of the reference states, resulting in larger tracking error. On the other hand, the bandwidth of the control input is decreased as well, resulting in smoother time histories. The time histories for the lateral control input at the start of the first right turn of the maneuver is plotted in figure 14. The plot shows a smoother time history for $t_{ref} = 5\,\text{s}$ and $t_{ref} = 7\,\text{s}$ compared to $t_{ref} = 3\,\text{s}$. For $t_{ref} = 3\,\text{s}$, there is a oscillation visible at 1/s, which coincides with the helicopter passing waypoints and the reference states getting updated. Therefore, choosing a reference time horizon $t_{ref} > 3\,\text{s}$ seems advisable. However, to mitigate the oscillations for shorter horizons as well and provide smoother time histories for the reference states, an interpolation or fading of the reference states may be added.

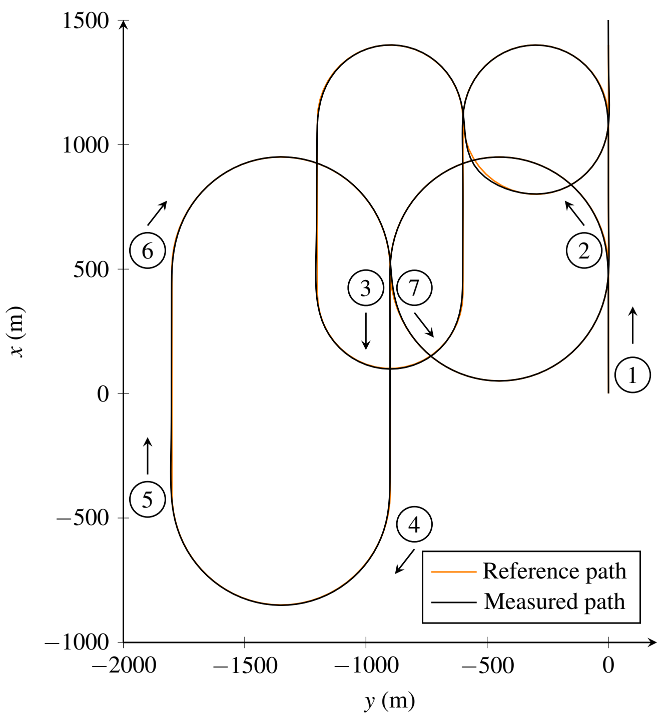


Figure 12: Path for testing path following controller

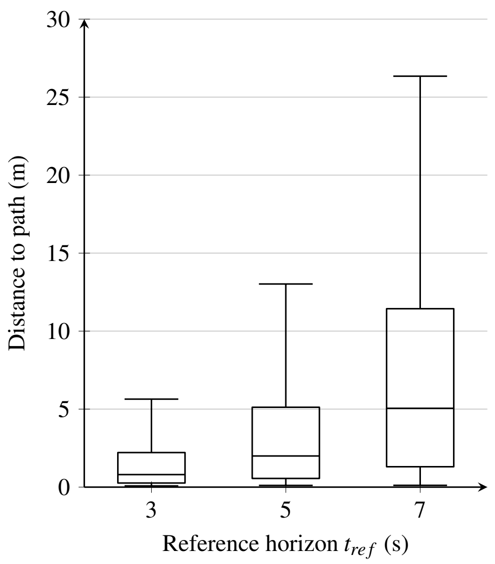


Figure 13: Distance to path for different reference horizon settings

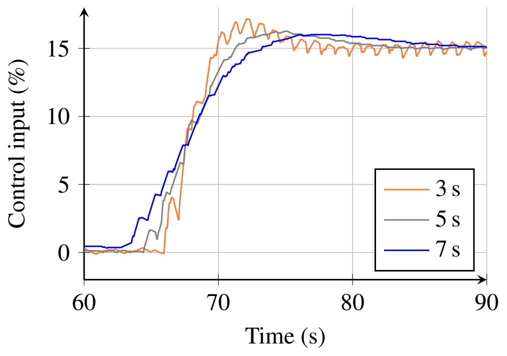


Figure 14: Lateral cyclic control input for path following

CONCLUDING REMARKS

The paper at hand proposes a sampling-based model predictive approach for automatic collision-free guidance of rotorcraft. The method calculates control inputs for the flight controller by predicting the closed-loop dynamics of a stabilized helicopter for a short time horizon. The predictions are evaluated by a cost function, that can take arbitrary forms. It accounts for collision-free maneuvering while following a desired path, as well as considering constraints on the helicopter states and control inputs. The inputs are fed to the helicopter in parallel to the pilot's input by utilizing the control inceptor trim actuators.

The prediction model used for the approach was derived from the reference model of the underlying model following flight controller, assuming that it would represent the closed-loop dynamics of the helicopter if the flight controller's performance is sufficient. The prediction model was validated against simulations of a high fidelity, physics-based helicopter model in closed-loop with the flight controller. The model showed acceptable agreement with the validation data over a broad range of the control input bandwidth, indicating that the approach of utilizing the flight controllers reference model is valid.

The method, previously used for collision avoidance only, was extended to be utilized for path following. The cost function was modified to allow for following reference states which are calculated from a list of waypoints that represents the planned path. The path following performance was evaluated in closed-loop simulation with a test path that includes typical maneuvers. The method was capable of following the path with sufficient performance, keeping the distance to track within 5 m for the majority of time. However, deficiencies in generating the reference states were identified, leading to oscillations in the control inputs.

The algorithm was re-implemented for execution on GPU hardware, extensively utilizing parallel execution of the trajectory prediction. Optimizations related to the parallel execution were introduced, reducing the computation time of the algorithm. The performance of parallel execution on a GPU has been compared to the sequential execution on a CPU, clearly indicating the performance increase by utilizing GPUs.

Future work will focus on assessing the system performance for a broader range of maneuvers and extending the scope of application to include maneuvering at low speed and hover. Furthermore, the processing of terrain and obstacles data shall be fused to generate a comprehensive representation of the rotorcraft's environment. Another topic is the consideration of competing pilot input on the control inceptors. This information may be used to detect the pilot's intention and react to it and hence decrease pilot workload and improve safety.

Author contact:
Alexej Dikarew alexej.dikarew@dlr.de
Tobias Winkler tobias.winkler@dlr.de

REFERENCES

1. European Aviation Safety Agency. *Annual safety review 2020.* Publications Office, LU, 2020.

2. European Union Aviation Safety Agency. *Annual safety review 2021.* Publications Office, LU, 2021.

3. European Union Aviation Safety Agency. *Annual safety review 2022.* Publications Office, LU, 2022.

4. Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Srikanth Saripalli. Flying Fast and Low Among Obstacles. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2023–2029, Rome, Italy, April 2007. IEEE. ISSN: 1050-4729.

5. Sebastian Scherer, Lyle Chamberlain, and Sanjiv Singh. First results in autonomous landing and obstacle avoidance by a full-scale helicopter. In *2012 IEEE International Conference on Robotics and Automation*, pages 951–956, St Paul, MN, USA, May 2012. IEEE.

6. Chad Goerzen and Matthew Whalley. Minimal risk motion planning: a new planner for autonomous UAVs in

uncertain environments. In *AHS International Specialists' Meeting on Unmanned Rotorcraft*, page 21, Tempe, Arizona, USA, January 2011.

7. Matthew S. Whalley, Marc D. Takahashi, Jay W. Fletcher, Ernesto Moralez, LTC Carl R. Ott, LTC Michael G. Olmstead, James C. Savage, Chad L. Goerzen, Gregory J. Schulein, Hoyt N. Burns, and Bill Conrad. Autonomous Black Hawk in Flight: Obstacle Field Navigation and Landing-site Selection on the RASCAL JUH-60A: Autonomous Black Hawk in Flight. *Journal of Field Robotics*, 31(4):591–616, July 2014.

8. Marc D. Takahashi, Brian T. Fujizawa, Jeffery A. Lusardi, Chad L. Goerzen, Mark J. Cleary, James P. Carr, and David W. Waldman. Comparison of Autonomous Flight Control Performance Between Partial- and Full-Authority Helicopters. *Journal of Guidance, Control, and Dynamics*, 45(5):885–901, May 2022.

9. Chad Goerzen, Marc Takahashi, Matt Whalley, MAJ Mark Cleary, and Jeffrey Cox. Kinematic Velocity Commander for Obstacle Field Navigation. In *Vertical Flight Society Autonomous VTOL Technical Meeting and Electric VTOL Symposium*, page 12, Mesa, Arizona, USA, January 2019.

10. Satoshi Suzuki, Takahiro Ishii, Yoshihiko Aida, Yohei Fujisawa, Kojiro Iizuka, and Takashi Kawamura. Collision-Free Guidance Control of Small Unmanned Helicopter Using Nonlinear Model Predictive Control. *SICE Journal of Control, Measurement, and System Integration*, 7(6):347–355, November 2014.

11. Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model Predictive Path Integral Control: From Theory to Parallel Computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, February 2017.

12. J V R Prasad, Vinodhini Comandur, Robert Walters, and David Guerrero. Model Predictive Path Integral Approach for Trajectory Guidance of Rotorcraft Shipboard Landing. In *AHS International 74th Annual Forum*, page 16, Phoenix, Arizona, USA, May 2018.

13. Vinodhini Comandur and JVR Prasad. Rotorcraft Shipboard Landing Guidance using MPPI Trajectory Optimization. In *44th European Rotorcraft Forum*, page 14, Delft, The Netherlands, September 2018.

14. Vinodhini Comandur, Robert Walters, and J V R Prasad. MPPI Parallel Trajectory Optimization for Guidance in Rotorcraft Shipboard Landing. In *AHS International 75th Annual Forum*, page 17, May 2019.

15. Jintasit Pravitra, Evangelos Theodorou, and Eric N. Johnson. Flying Complex Maneuvers with Model Predictive Path Integral Control. In *AIAA Scitech 2021 Forum*, VIRTUAL EVENT, January 2021. American Institute of Aeronautics and Astronautics.

16. Yuji Shimizu. Nonlinear Model Predictive Real-time Control of Aircraft in Collision Avoidance. In *AIAA Scitech 2020 Forum*, Orlando, FL, January 2020. American Institute of Aeronautics and Astronautics.

17. Martin Keller, Carsten Hass, Alois Seewald, and Torsten Bertram. A Model Predictive Approach to Emergency Maneuvers in Critical Traffic Situations. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 369–374, Gran Canaria, Spain, September 2015. IEEE.

18. Andreas Homann, Markus Buss, Martin Keller, Karl-Heinz Glander, and Torsten Bertram. Multi Stage Model Predictive Trajectory Set Approach for Collision Avoidance. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 945–950, Maui, HI, November 2018. IEEE.

19. Alexej Dikarew. Model Predictive Approach for Short-Term Collision Avoidance. In *Proceedings of the Vertical Flight Society 77th Annual Forum*, pages 1–10, Virtual Conference, May 2021. The Vertical Flight Society.

20. Alexej Dikarew. Helicopter short-term collision avoidance with sampling-based model predictive control. *CEAS Aeronautical Journal*, March 2024.

21. Benjamin Fragnière, Johannes Wartmann, and Steffen Greiser. Preliminary Results Towards Fidelity Evaluation of an In-flight Simulator. In *AIAA Scitech 2019 Forum*, San Diego, California, January 2019. American Institute of Aeronautics and Astronautics.

22. Steffen Greiser, Robin Lantzsch, Jens Wolfram, Johannes Wartmann, Mario Müllhäuser, Thomas Lüken, Hans-Ullrich Döhler, and Niklas Peinecke. Results of the pilot assistance system "Assisted Low-Level Flight and Landing on Unprepared Landing Sites" obtained with the ACT/FHS research rotorcraft. *Aerospace Science and Technology*, 45:215–227, September 2015.

23. Jürgen Kaletka, Hermann Kurscheid, and Ulrich Butter. FHS, the new research helicopter: Ready for service. *Aerospace Science and Technology*, 9(5):456–467, July 2005.

24. Sven Schmerwitz, Alexej Dikarew, Tim Laudien, Johannes M. Ernst, and Mario R. Müllhäuser. Investigating multimodal cueing concept for human-machine shared control under automatic trajectory following low-level operation. In Paul L. Muench, Hoa G. Nguyen, and Brian K. Skibba, editors, *Unmanned Systems Technology XXV*, page 19, Orlando, United States, June 2023. SPIE.

25. U.S. Department of Defense. MIL-DTL-32742, Detail Specification Handling Qualities for Military Rotorcraft, March 2023.

26. Mark B. Tischler and Robert K. Remple. *Aircraft and Rotorcraft System Identification: Engineering Methods with Flight-Test Examples*. American Institute of Aeronautics and Astronautics, Inc., 2012.

27. Holger Duda, Sunjoo K. Advani, and Mario Potter. Design of the DLR AVES Research Flight Simulator. In *AIAA Modeling and Simulation Technologies (MST) Conference*, Boston, MA, August 2013. American Institute of Aeronautics and Astronautics.

28. Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities, reprinted from the afips conference proceedings, vol. 30 (atlantic city, n.j., apr. 18–20), afips press, reston, va., 1967, pp. 483–485, when dr. amdahl was at international business machines corporation, sunnyvale, california. *IEEE Solid-State Circuits Society Newsletter*, 12(3):19–20, 2007.

29. NVIDIA. Jetson AGX Orin Series Data Sheet. https://developer.download.nvidia.com/assets/embedded/secure/jetson/agx_orin/Jetson_AGX_Orin_Series_Data_Sheet_DS-10662-001_v1.5.pdf, 2023. 2024-02-20.

30. NVIDIA. Cuda C++ Programming Guide. https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf, 1 2024. 2024-01-08.

31. Alexander Shapiro and Tito Homem-de Mello. On the rate of convergence of optimal solutions of monte carlo approximations of stochastic programs. *SIAM Journal on Optimization*, 11(1):70–86, 2000.

32. Maurizio Capra, Beatrice Bussolino, Alberto Marchisio, Guido Masera, Maurizio Martina, and Muhammad Shafique. Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead. 12 2020.