Automatic Gain Tuning for Multirotors Using Differentiable Optimization

Achim von Prittwitz, Khaled Wahba, and Wolfgang Hönig

I. INTRODUCTION

Robust control in robotics applications such as quadrotors, humanoid, and quadrupedal systems, requires effective tuning of controller gains. Traditional manual tuning, though common, is inefficient and expert-dependent, motivating the development of automatic tuning techniques.

For legged robots, automatic gain tuning has shown great promise in improving control stability. For humanoid robots, momentum-based control strategies have been applied successfully to balancing and locomotion tasks, optimizing joint space dynamics [1]. Similarly, quadrupeds have benefited from Bayesian optimization techniques like [2], which tune controllers while ensuring safe exploration. This approach has been validated in locomotion tasks, balancing safety and performance, though computational demands increase with higher-dimensional spaces.

For aerial robots, methods like the work presented in [3] use gradient-based optimization to improve tracking performance in quadrotors, but they rely on informed initial guesses, limiting applicability to unknown systems. Safe Bayesian optimization has also been applied to uncrewed aerial vehicles (UAVs), ensuring safe parameter exploration under dynamic conditions, as shown in [4]. Additionally, gain scheduling has been explored for morphing UAVs, dynamically adjusting control parameters to maintain consistent performance across different configurations [5]. Multi-objective optimization (MOO) has been used for aerial manipulators to balance competing goals like minimizing error and enhancing stability [6].

Most prior works for multirotor gain tuning suffer from the curse of dimensionality [4] and can only tune a few gains, or require a good initial guess from an expert [3, 4]. In this work, we introduce a novel incremental trajectory length scheduling algorithm that helps stabilize differentiable optimization problems that reason over long time horizons. On the example of gain tuning for multirotors, we demonstrate that our approach is able to converge to stable control gains using uninformed initial guesses. Moreover, the resulting gains outperform the expertly tuned gains in terms of tracking error in many cases.

Algorithm 1: Incremental trajectory length scheduling for automatic gain tuning

$\mathcal{G} \leftarrow$ `InitializeGains()`
$T \leftarrow T_0$ ▷ *Initial length of trajectories*
$\mathcal{D} \leftarrow$ `SliceTrajectory(`T`)`
$\ell_{best} \leftarrow \infty$ ▷ *Current best loss*
$i_{dec} \leftarrow 0$ ▷ *Iterations since decrease*
for $n = 1, 2, \dots$ **do**
  $y_{pred} \leftarrow$ `SimulateMultirotorWithController(`$\mathcal{D}$`)`
  $\ell \leftarrow$ `ComputeLoss(`$y_{pred}, \mathcal{D}$`)`
  $\mathcal{G} \leftarrow$ `GradientStep(`$\mathcal{G}, \ell$`)`
  $\mathcal{G} \leftarrow$ `ClampParameters(`$\mathcal{G}$`)`
  if $\ell < \ell_{best}$ **then**
    $\ell_{best} \leftarrow \ell$
  else
    $i_{dec} \leftarrow i_{dec} + 1$
    if $i_{dec} > P$ **then**
      $T \leftarrow 2T$ ▷ *Double trajectory length*
      if $T > T_{max}$ **then**
        $T \leftarrow T_{max}$
      $\mathcal{D} \leftarrow$ `SliceTrajectory(`T`)`
      $\ell_{best} \leftarrow \infty$
      $i_{dec} \leftarrow 0$

II. APPROACH

The dynamics of a single multirotor is modeled as a 6 degrees-of-freedom floating rigid body with mass m and diagonal moment of inertia $\mathbf{J}$. The multirotor's state comprises of the global position $\mathbf{p} \in \mathbb{R}^3$, global velocity $\mathbf{v} \in \mathbb{R}^3$, attitude rotation matrix $\mathbf{R} \in SO(3)$ and body angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$. The dynamics can be expressed using Newton-Euler [7] equations of motion as follows

$$\dot{\mathbf{p}} = \mathbf{v}, \qquad m\dot{\mathbf{v}} = m\mathbf{g} + \mathbf{R}\mathbf{f}_u, \tag{1a}$$

$$\dot{\mathbf{R}} = \mathbf{R}\hat{\boldsymbol{\omega}}, \qquad \mathbf{J}\dot{\boldsymbol{\omega}} = \mathbf{J}\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}_u, \tag{1b}$$

where $\hat{\cdot}$ denotes a skew-symmetric mapping $\mathbb{R}^3 \to \mathfrak{so}(3)$; $\mathbf{g} = (0, 0, -g)^\top$ is the gravity vector; $\boldsymbol{f}_u = (0, 0, f)^\top$ and $\boldsymbol{\tau}_u = (\tau_x, \tau_y, \tau_z)^\top$ are the total thrust and body torques from the rotors, respectively. An exponentially stable geometric controller for a multirotor computes the desired force and torques as follows [8]:

$$f = -(-\mathbf{K_p}\mathbf{e_p} - \mathbf{K_v}\mathbf{e_v} - m\mathbf{g} + m\dot{\mathbf{v}}_\mathbf{d}) \cdot \mathbf{R}\mathbf{e_3}, \tag{2a}$$

$$\begin{aligned} \boldsymbol{\tau}_u = &-\mathbf{K_R}\mathbf{e_R} - \mathbf{K}_\omega\mathbf{e}_\omega + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \\ &- \boldsymbol{J}(\hat{\boldsymbol{\omega}}\mathbf{R}^\top\mathbf{R_d}\boldsymbol{\omega_d} - \mathbf{R}^\top\mathbf{R_d}\dot{\boldsymbol{\omega}}_{\boldsymbol{d}}), \end{aligned} \tag{2b}$$

where $\mathbf{e_3} = (0, 0, 1)^\top$, the subscript d refers to the desired reference trajectory, $\mathbf{e_p}$, $\mathbf{e_v}$, $\mathbf{e_R}$, $\mathbf{e}_\omega \in \mathbb{R}^3$ are errors with respect to this reference (mathematically defined in [8]), and $\mathbf{K_p}$, $\mathbf{K_v}$, $\mathbf{K_R}$, $\mathbf{K}_\omega \in \mathbb{R}^{3\times 3}$ are diagonal positive gain matrices that need to be tuned. The twelve non-zero elements of the gain matrices are the parameter set $\mathcal{G}$ that we optimize.

All authors are with the Faculty of Electrical Engineering and Computer Science, Technical University of Berlin, Berlin, Germany.

The research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 448549715.

TABLE I

GAINS AFTER AUTOMATIC TUNING ON FIGURE-8 TRAJECTORY.

Gain	$\mathbf{K_p}$			$\mathbf{K_v}$			$\mathbf{K_R}$			$\mathbf{K}_\omega$		
axis	x	y	z	x	y	z	x	y	z	x	y	z
initial	1	1	1	1	1	1	1	1	1	1	1	1
auto-tuned	2.5613	2.4822	1.0091	9.5109	11.2808	1.0125	0.0265	0.0738	0.0001	0.0026	0.0020	9.706e-6
hand-tuned	9	9	9	7	7	7	0.0055	0.0055	0.0055	0.0013	0.0013	0.0013

TABLE II

TRACKING ERRORS FOR TEST TRAJECTORIES. LOWER (BOLD) VALUES ARE BETTER. THE FIGURE-8 TRAJECTORY WAS USED FOR TRAINING.

avg. Errors	$\mathcal{L}_p$		$\mathcal{L}_v$		$\mathcal{L}_R$		$\mathcal{L}_\omega$	
	hand-tuned	auto-tuned	hand-tuned	auto-tuned	hand-tuned	auto-tuned	hand-tuned	auto-tuned
Figure-8	1.345e-5	**1.023e-5**	2.430e-4	**8.286e-6**	6.801e-3	**5.600e-3**	1.741e-3	**9.902e-4**
Circle	**2.269e-6**	9.658e-6	**1.470e-5**	1.589e-5	1.072e-8	**7.402e-9**	2.802e-4	**2.005e-4**
Helix	**4.886e-6**	8.888e-6	7.360e-5	**4.368e-6**	**1.737e-3**	1.851e-3	**1.453e-3**	1.550e-3
Random Waypoints	**2.206e-6**	8.777e-6	1.735e-5	**6.273e-6**	**1.318e-3**	1.336e-3	6.332e-5	**4.836e-5**

We can simulate over a time horizon T by composing the simulator and controller, e.g., $\mathbf{p}_{t+2} = sim(ctrl(sim(ctrl(\mathbf{p}_t, \mathbf{p_d}_t)), \mathbf{p_d}_{t+1}))$ if simplifying for the positional part for brevity. Here sim is the simulator implementing (1) and $ctrl$ is the controller implementing (2). Since all operations are differentiable, we use PyTorch [9] to automatically compute gradients. With untuned controller gains the tracking is highly unstable and catastrophic divergence already occurs after a few steps. To prevent this divergence and consequently numeric overflows and exploding gradients, we use a scheduled approach, see Algorithm 1. First we tune the gains using only slices of T_0 steps were T_0 is a hyperparameter (Line 2). The model is trained to convergence i.e. until the loss does not decrease over a number of P epochs (Line 7 to Line 15). After the model converges, the length of the trajectory slices T is doubled (Line 16). This procedure is repeated until the optimization is done for the whole trajectory or a certain number of steps T_{max} to prevent the model from getting too deep. To prevent the procedure from generating unreasonable gains, we clamp the gains to be not smaller than 10^{-8} (Line 10). This is necessary as negative gains violate the stability assumptions of the controller. In addition to clamping the gains, we also clip the gradients to be within a 10% range of the current parameter values (not shown in the pseudo code). This approach allows for stable training in the beginning and the capture of long term behavior in the end. As a loss function we used the mean squared error for the position, velocity, rotation and attitude rate tracking errors:

$$\min_{\mathbf{K_p},\mathbf{K_v},\mathbf{K_R},\mathbf{K}_\omega} \lambda_p\mathcal{L}_p + \lambda_v\mathcal{L}_v + \lambda_R\mathcal{L}_R + \lambda_\omega\mathcal{L}_\omega, \tag{3a}$$

$$\mathcal{L}_p = \sum_{t=1}^{T} \mathbf{e_p}_t^2, \qquad \mathcal{L}_v = \sum_{t=1}^{T} \mathbf{e_v}_t^2, \tag{3b}$$

$$\mathcal{L}_R = \sum_{t=1}^{T} \mathbf{e_R}_t^2, \qquad \mathcal{L}_\omega = \sum_{t=1}^{T} \mathbf{e}_{\omega t}^2, \tag{3c}$$

where $\lambda_{\{p,v,R,\omega\}}$ are weighting parameters.

III. RESULTS

The controller is tuned on a figure-8 trajectory in the xy-plane with a control frequency of $100\,\mathrm{Hz}$. The total duration of the trajectory is 7.28 seconds. Although the poor initial guesses of the gains (see Table I) do not allow for the simulation of the whole trajectory due to numerical instability, we obtain gains which enable stable control after running the optimization loop for 500 epochs. The initial length of the trajectory slices is $T_0 = 3$ and as maximum trajectory length we chose $T_{max} = 400$. At a sampling rate of $100\,\mathrm{Hz}$ this is equivalent to simulating 4 seconds of flight. The patience for the scheduling process is set to $P = 5$. Smaller P values lead to too fast scheduling whereas larger P values lead to slow scheduling wasting computing resources on small values of T. For our loss function, we weigh all the components equally, i.e. $\lambda_p = \lambda_v = \lambda_R = \lambda_\omega = 1$. The tuning took 40 minutes on a Laptop computer with a Intel i7-10510U CPU.

The resulting gains, shown in Table I, achieve better tracking on the training trajectory and stable behavior on the test trajectories, see Table II. Note that we use an initial guess of 1.0 for all gains to simulate an uninformed guess. This is in contrast to prior work which used initial values which already allowed for stable control [3]. The gains for the z components are much smaller than the gains for the x and y components. We believe that this indicates that the chosen training trajectory did not sufficiently excite the z axis.

IV. CONCLUSION

The idea of scheduling the optimization time horizon to balance between long-term behavior and stability shows promising results on the example of multirotor gain tuning. Further investigation on how the training trajectory and the initial guesses are influencing the tuned gains is needed. Moreover, we believe that our scheduling approach's stability will enable tackling of higher-dimensional problems. For example, jointly optimizing unknown system identification parameters, gains for a state estimator, and control gains.

References

[1] D. Pucci, G. Nava, and F. Nori, *Automatic gain tuning of a momentum based balancing controller for humanoid robots, in 2016 ieee-ras 16th international conference on humanoid robots (humanoids)*, 2016.

[2] D. Widmer, D. Kang, B. Sukhija, J. Hübotter, A. Krause, and S. Coros, "Tuning legged locomotion controllers via safe bayesian optimization," in *Conference on Robot Learning*, PMLR, 2023, pp. 2444–2464.

[3] S. Cheng, M. Kim, L. Song, C. Yang, Y. Jin, S. Wang, and N. Hovakimyan, "Difftune: Auto-tuning through auto-differentiation," *IEEE Transactions on Robotics*, 2024.

[4] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *Machine Learning*, vol. 112, no. 10, pp. 3713–3747, Oct. 1, 2023, ISSN: 1573-0565.

[5] Y. Aboudorra, A. Saini, and A. Franchi, "Gain scheduling position control for fully-actuated morphing multi-rotor uavs," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2024, pp. 15–22.

[6] X. Zhou and X. Zhang, "Multi-objective-optimization-based control parameters auto-tuning for aerial manipulators," *International Journal of Advanced Robotic Systems*, vol. 16, no. 1, p. 1 729 881 419 828 071, 2019.

[7] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.

[8] T. Lee, K. Sreenath, and V. Kumar, "Geometric control of cooperating multiple quadrotor uavs with a suspended payload," in *IEEE Conference on Decision and Control (CDC)*, 2013, pp. 5510–5515.

[9] J. Ansel *et al.*, "Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024*, R. Gupta, N. B. Abu-Ghazaleh, M. Musuvathi, and D. Tsafrir, Eds., ACM, 2024, pp. 929–947.