Quadrotor Control Parameters Optimization Using Gradient Descent Method

Ekaterina Kosareva[0000−0002−5388−4597], Artemii Zenkin[0000−0002−8871−3835], Ivan Kirilenko[0000−0001−9416−8645], and Aleksandr Kapitonov[0000−0001−5517−3038]

ITMO University, 49 Kronverksky Pr., St. Petersburg 197101, Russia
ksrve@mail.ru,a.zenkin@itmo.ru, ivan009ki@gmail.com, kapitonov.aleksandr@itmo.ru

Abstract. Proportional-Integral-Derivative (PID) controller is widely used for quadrotors due to its usability and robustness over many years. The PID controller is easy to understand and implement, but the quality of its parameters directly determines the control effect. Manual adjustment of three PID parameters using trial-and-error method is a time-consuming process that can hardly reach a good result. In recent years, numerical optimization methods have been developed and expanded. In this paper, a proposed optimization by the gradient descent (GD) method is employed to tune PID controller for a quadrotor. First, the initial parameters of the GD-based PID controller are chosen, including learning rate, number of epochs and values of the parameter vector. Then the gradient-based algorithm is employed to minimize the cost function by adjusting the parameters. Both the classical and the self-adjusting PID controller were implemented to track defined trajectory with quadrotor. Unlike conventional PID design, this method is capable of reaching the desired control performance, such as minimum overshoot. Another unique feature is that the optimized PID controller is applicable to a wide variety of time-varying and nonlinear systems. The analysis of the experiment results demonstrates that algorithm can quickly and accurately find the optimal parameters solutions. Comparison of simulations shows that this method is robust and adaptable and can also improve control precision.

Keywords: UAV · Quadrotor · Parameters tuning · Gradient optimization · PID controller.

1 Introduction

In recent years, interest in research on small unmanned aerial vehicles (UAVs) has increased dramatically. These UAVs have become one of the major and

critical areas [1, 2]. They are widely used for military and civilian purposes such as remote sensing, aerial surveillance, data collection, fire detection and logistics. Other advantages with using unmanned aerial vehicle include being superior over a manned aircraft in terms of loss of life and absence of limitations such as human-related fatigue or operating hours as with a manned vehicle.

One of the most preferable unmanned aerial vehicles is four–rotor drones, also known as quadcopters or quadrotors. At a small size, quadcopters are cheaper and more durable than conventional helicopters due to their mechanical simplicity. This aircraft has the ability to take off and land vertically (VTOL) and hover in a stable air condition. However, it has a number of limitations, such as complex and difficult control and limited capacity of energy and load.

There are many controller designs and applications in the literature to control unmanned aerial vehicles, e.g., fuzzy controller [3], sliding mode controller [4], neuro-fuzzy controller [5] and image-based controller [6], but conventional PID controller is widely applied in UAV for its practicability and robustness in the past decades.

A proportional–integral–derivative controller is a control loop mechanism that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value $e(t)$ as the difference between a desired setpoint and a measured process variable and applies a correction based on proportional, integral, and derivative terms.

The PID controller is easy to understand and implement, but the quality of its parameters directly determines the control effect. Manual adjustment of three PID parameters using trial-and-error method is a time-consuming process that can hardly reach a good result. In recent years, numerical optimization methods have been developed and expanded.

Many approaches have been proposed for improving the setting performance of PID parameters using heuristic optimization methods such as Particle Swarm Optimization (PSO) [7, 8] and Genetic Algorithm (GA) [9].

Gradient Descent (GD) – a first-order iterative optimization algorithm for finding the local minimum of a differentiable function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. If, instead, one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function. Several works have been proposed for tuning the PID parameters using the gradient descent methods such as [10] and [11]. Our approach is based on the use of improved methods of classical or "vanilla" gradient descent.

In the given work the control system of the quadcopter, realizing the flight on control points in an automatic mode, is considered. Tuning of coefficients is performed during the flight using the method of numerical optimization – the gradient descent.

The paper organized as follows. In next section, a brief description about the AR.Drone 2.0 as a controlled system has given. After that the gradient descent

algorithm for control parameters optimization is described. Then quadrotor control results has been demonstrated. In the last section, the main outcomes of this work are summarized.

2 Controlled system

The AR.Drone 2.0 quadcopter from the French company Parrot was chosen as the control object. A short description of the quadrotor is presented in the following section of this report.

2.1 Description

The AR.Drone 2.0 is an electrically powered quadcopter intended for augmented reality games [12]. It consists of a carbon-fiber support structure, plastic body, four high-efficiency brushless motors, sensor and control board, two cameras and indoor and outdoor removable hulls. The control board not only ensures safety by instantly locking the propellers in case of a foreign body contact, but also assists the user with difficult maneuvers such as takeoff and landing.

The AR.Drone 2.0 is a not open source UAV introduced by parrot company as a toy in 2010. However this quadcopter has increased its popularity in the academic and developmental applications, a numerous of application has been developed for manipulate the drone from a smart phone, tablet or a computer with wi–fi connection, there are also numerous publications, softwares and videos with useful information for the users.

Takeoff and land functions are controlled for the embedded electronic system. Landing sequence is automatically activated in case of low battery situation. AR.Drone have also an emergency function which cut off the power on the motors and can be activated during a flight if any object comes in contact with the propellers or the emergency function is sent by code from the pilot device.

In work project, the AR.Drone 2.0 was controlled from a computer using Ubuntu 16.04 with Robot Operating System (ROS) Kinetic Kame [13] and ROS packages, tum_simulator [14] and ardrone_autonomy [15].

2.2 System identification

AR.Drone is mainly controlled considering four degree of freedom (4DoF) and the control parameters to the on board controller can be given as float values in between [-1,1]. The four inputs represent by (u_x, u_y, u_z,u_ψ), are linear velocities on longitudinal axes, transversal axes, vertical speed and yaw angular speed respectively. The four outputs (x, y, z, ψ), are position in longitudinal axes, position in transversal axes, position in vertical axes and yaw angle respectively.

3 Control system

In this section we describe control system for quadcopter.

3.1 Control loop

Despite technological progress, the PID controller remains the most popular control loop mechanism and is used in many areas. This is due to the fact that the PID controller is easy to understand and implement. The controller forms a control signal, which is the sum of three parts: proportional, integral and derivative components.

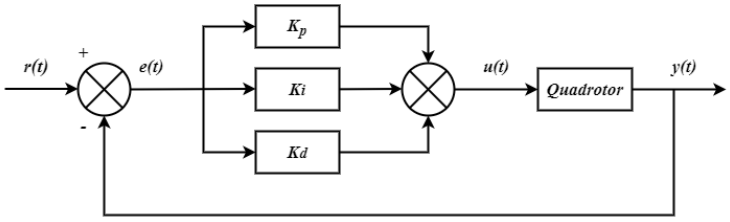

Fig. 1: The block diagram of PID controller.

The general control function can be mathematically expressed as:

$$u(t) = P + I + D = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t). \quad (1)$$

where K_p, K_i, K_d are proportional, integral and derivative gains, respectively; e - is the error caused by the difference between the reference and response of the system.

3.2 Setting the parameters of the controller

The success of the PID controller depends on the quality of the gain choice. The PID controller in the literature can mainly be divided into two categories.

In the first category, the PID controller parameters remain the same during the control. At this point the best known method is to set the PID controller using the Ziegler-Nichols formula [16]. This method includes setting the D and I gains to zero value. The parameters which are monitored and manipulated in this method are Proportional Gain (Kp), Ultimate Period (Pu) and Oscillation Period (Tu). The P, I and D gains are now set against oscillation period and ultimate gain so that desired output is achieved. This method is very simple and gives not very good results. However, it is still often used in practice, although many more precise methods have emerged since then.

3.3 Gradient descent method

In this paper, the gradient descent method was considered, which is described in details in [17] and [18]. Gradient descent is a method of optimizing the finding of

the minimum value of the error function, which is widely used to train artificial neural networks. The purpose of the algorithm is to find the model parameters (in our case, the parameters of the PID controller), which minimize the model error in the set of training data. This is achieved by making changes to the model, which moves it along the gradient or the error slope down to the minimum error value.

A gradient is usually counted as the sum of the gradients caused by each learning element. The vector of parameters changes in the direction of the anti–gradient with a specified step. Therefore, a standard (batch) gradient descent requires a single pass through all learning data before it can change parameters.

In the case of the stochastic gradient descent, instead of cycling through each example of learning, we use only one randomly selected learning element. Thus, the model parameters change after each learning object. For large data arrays, a stochastic gradient descent can give a significant speed advantage over a standard gradient descent.

The compromise between the two methods is mini-batch. In this case, a small set of data is processed instead of calculating the gradient from 1 sample (SGD) or all n training samples (GD), we calculate the gradient from k training samples.

3.4 Extensions and variants

The learning rate is the most important hyper-parameter for gradient-descent based methods [19]. Based on how the learning rate is set, two types of GD-based optimization methods can be categorized. The first type indicates fixed learning rate methods such as SGD [20], SGD Momentum, and Nesterovs Momentum [21], and the second type includes auto learning rate methods, such as AdaGrad [22], RMSProp [23], and Adam [24]. In this paper, we discuss methods based on fixed learning rate.

Stochastic Gradient Descent (SGD) is a widely used optimization algorithm for machine learning. SGD usually uses a fixed learning rate. This is because the SGD gradient estimator introduces a source of noise (the random sampling of m training examples), and that noise does not vanish even when the loss arrives at a minimum. A hyper-parameter $\alpha \in (0, 1)$ is the step of the method (or learning rate). Stochastic gradient descent updates the parameters for each observation, so tuning happens online while flying.

The parameter update rule of SGD from time (i.e., iteration) t to time $t + 1$ is given by:

$$\theta_{t+1} = \theta_t - \alpha \partial L_t / \partial \theta_t \tag{2}$$

SGD Momentum is designed to accelerate learning, especially in the case of small and consistent gradients. The momentum algorithm accumulates an exponentially decayed moving average of past gradients and continues to move in the consistent direction. The name momentum derives from a physical analogy, in which the negative gradient is a force moving a particle through parameter

space. A hyper-parameter $\gamma \in (0, 1)$ determines how much the past gradients to the current update of the weights. The momentum term γ is usually set to 0.9 or a similar value.

Its parameter update rule:

$$\begin{cases} V_{t+1} = \gamma V_t - \alpha \partial L_t / \partial \theta_t \\ \theta_{t+1} = \theta_t + V_{t+1} \end{cases} \tag{3}$$

Nesterov's Momentum is a simple change to normal momentum. Here the gradient term is not computed from the current position θ_t in parameter space but instead from a position $\theta(t + 1) = \theta(t) + \gamma V_t$. This helps because while the gradient term always points in the right direction, the momentum term may not. If the momentum term points in the wrong direction or overshoots, the gradient can still "go back" and correct it in the same update step.

The Nesterov's Momentum update rule is given by:

$$\begin{cases} V_{t+1} = \gamma V_t - \alpha \partial L_t / \partial \theta_t (\theta_t + \gamma V_t) \\ \theta_{t+1} = \theta_t + V_{t+1} \end{cases} \tag{4}$$

3.5 GD-based controller approach

Let's suppose, gradient-descent method will adjust the PID controller parameters by minimizing the squared controller error, which defined:

$$e = r - y \tag{5}$$

where r - the set point and y - the control output.

Let's define the objective function, which should be minimized:

$$L = \frac{1}{2} e^2 \implies \frac{1}{2}(r - y)^2 \tag{6}$$

where $\theta = (K_p, K_i, K_d)$ is the vector of parameters.

From the definition of the parameter vector θ:

$$\frac{\partial L}{\partial \theta} = \left(\frac{\partial L(\theta)}{\partial K_p}, \frac{\partial L(\theta)}{\partial K_i}, \frac{\partial L(\theta)}{\partial K_d}\right) \tag{7}$$

When used to minimize the above function, the main challenge remains behind the calculation of gradient $\partial L_t / \partial \theta_t$ in 2. In this case we must apply a chain rule to composites of more than two functions:

$$\frac{\partial L}{\partial K_{p/i/d}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial K_{p/i/d}} \tag{8}$$

where $K_{p/i/d}$ - one of the PID controller parameters.

To measure the rate of change of L in any direction that is represented by a unit vector u, in multivariate calculus, we define the directional derivative of L at θ in the direction of u as:

$$\frac{\partial y}{\partial u} = \frac{y_{t+1} - y_t}{u_{t+1} - u_t} \tag{9}$$

where y_t - the actual output of the plant and u_t - the control signal (output of the controller)

Therefore, the update rules for PID control parameters for the gradient-descent are expressed as:

$$\begin{cases} K_p = K_p - \alpha \cdot \frac{\partial L}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial K_p} = K_p + \alpha \cdot e_t \frac{\partial y}{\partial u} e_t \\ K_i = K_i - \alpha \cdot \frac{\partial L}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial K_i} = K_i + \alpha \cdot e_t \frac{\partial y}{\partial u} \int_0^t e_t dt \\ K_d = K_d - \alpha \cdot \frac{\partial L}{\partial y}\frac{\partial y}{\partial u}\frac{\partial u}{\partial K_d} = K_d + \alpha \cdot e_t \frac{\partial y}{\partial u}\frac{d}{dt} e_t \end{cases} \tag{10}$$

The GD-based PID controller is shown in Figure 2. The inputs of algorithm are created by proportion, integration and derivation of error e.

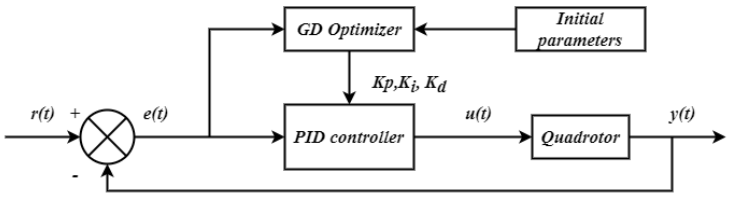

Fig. 2: Architecture of GD-based PID controller

4 Results and analysis

The main objective of this study is to enable the aircraft to follow a defined trajectory with algorithm of an improved controller. When the aircraft takes off, the initial position and orientation of the aircraft is set to zero. Then the aircraft is informed about the reference waypoints. The waypoints information which are necessary for trajectory tracking includes four data. These are X_{ref}, Y_{ref}, h_{ref} and ψ_{ref}. The relative position between vehicle and target is calculated in PC. And the controllers provide to track desired parameters. The aircraft's position and velocity is measured by its inertial sensors.

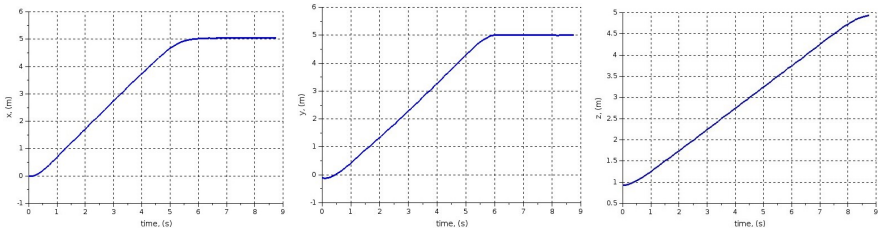

Fig. 3: The movement of the quadcopter to point (5.0; 5.0; 5.0) along X axis, Y axis and Z axis respectively.

4.1 Simulation Results

The simulation has done using Gazebo. The results of the performance of SGD-Momentum method is given (see Figure 3). In the following simulations, the proposed algorithm SGD-Momentum used a set of parameters as: the learning rate $\alpha = 0.0001$, momentum parameter $\gamma = 0.9$, the number of iterations $n = 100$. In Figure 5, the simulation gains has been shown using GD-based autotuning.

4.2 Realtime Results

In this section, the experimental results has shown. In Figure 4, the real experimental results has plotted. From this figure it is clear that, AR.Drone successfully had gone to the point with coordinates (2.0; 2.0; 2.0).

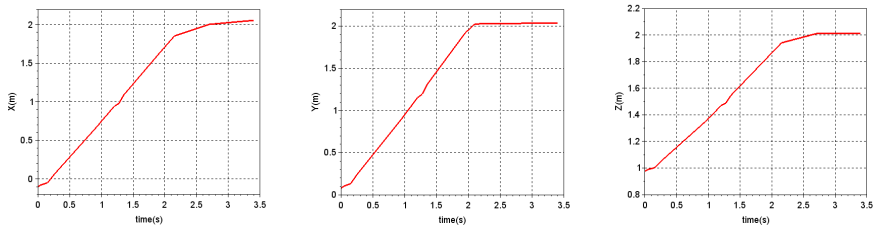

Fig. 4: The realtime movement of the quadcopter to point (2.0; 2.0; 2.0) along X axis, Y axis and Z axis respectively.

We can see that the parameters are adjusted faster than in the approach with classical gradient descent. The optimal coefficients have been selected, thus we reaching the desired control performance, such as minimum overshoot (see Figure 5).

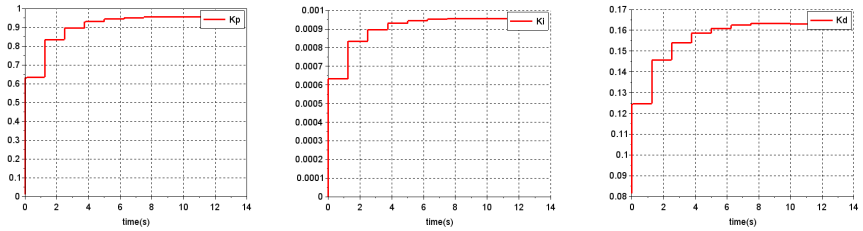

Fig. 5: Autotuning of PID gains along the X axis

5 Conclusions

In this paper, the control system of the quadcopter based on PID controller with optimization of parameters using the gradient descent method was considered.

The results verify that the proposed algorithm is an effective method for optimal tuning of PID parameters in term of no overshoot, zero steady state error and extremely short setting time and rise time. The proposed approach is highly robust and effective.

This proposed tuning method may create some benefit in real-life and industrial applications since it saves time for none expert control engineering to find optimal controller parameters to enhance the performance quality. For future work, gradient based method will be applied to tune PID controller of PX4 quadrotor.

6 Acknowledgments

This work was financially supported by ITMO University: grant num. 419331 "Development of a multifunctional autonomous landing station for multi–copter with an autopilot system".

References

1. Santamarina-Campos, V., Segarra-Oña, M.: Drones and the Creative Industry. Springer (2018)
2. Hiltner, P. J.: The drones are coming: Use of unmanned aerial vehicles for police surveillance and its fourth amendment implications. Wake Forest JL & Pol'y, **3**, pp. 397 (2013)
3. Sabo, C., Cohen, K.: Fuzzy logic unmanned air vehicle motion planning. Advances in Fuzzy Systems(2012)
4. Xiong, J. J., Zhang, G.: Sliding mode control for a quadrotor UAV with parameter uncertainties. In: 2016 2nd International Conference on Control, Automation and Robotics (ICCAR), pp. 207-212. IEEE (2016)
5. Farid, A. M.: UAV controller based on adaptive neuro-fuzzy inference system and PID. IAES International Journal of Robotics and Automation, **2**(2), 73 (2013)

6. Asl, H. J., Yoon, J.: Robust image-based control of the quadrotor unmanned aerial vehicle. Nonlinear Dynamics, **85**(3), 2035-2048 (2016)
7. Liu, X., Zhao, D., Wu, Y.: Application of improved PSO in PID parameter optimization of quadrotor. In: 2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), pp. 443-447. IEEE (2015)
8. Mac, T. T., Copot, C., Duc, T. T., De Keyser, R.: AR. Drone UAV control parameters tuning based on particle swarm optimization algorithm. In: 2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), pp. 1-6. IEEE (2016)
9. Chen, Q., Chen, T., Zhang, Y.: Research of GA-based PID for AUV motion control. In: 2009 International Conference on Mechatronics and Automation, pp. 4446-4451. IEEE (2009)
10. Babu, V. M., Das, K., Kumar, S.: Designing of self tuning PID controller for AR drone quadrotor. In: 2017 18th international conference on advanced robotics (ICAR), pp. 167-172. IEEE (2017)
11. Zhu, J., Liu, E., Guo, S., Xu, C.: A gradient optimization based PID tuning approach on quadrotor. In: The 27th Chinese Control and Decision Conference (2015 CCDC), pp. 1588-1593. IEEE (2015)
12. Krajník, T., Vonásek, V., Fišer, D., Faigl, J.: AR–drone as a platform for robotic research and education. In: International conference on research and education in robotics, pp. 172–186. Springer, Berlin, Heidelberg (2011)
13. Robot Operating System, https://www.ros.org/
14. Tum_simulator homepage, http://wiki.ros.org/tum_simulator
15. Ardrone homepage, https://ardrone-autonomy.readthedocs.io/en/latest/
16. Ziegler, J. G., Nichols, N. B.: Optimum settings for automatic controllers. Trans. ASME, **64**(11) (1942).
17. Saad, D.: Online algorithms and stochastic approximations. Online Learning. Cambridge University Press, 5, 6–3 (1998)
18. An, W., Wang, H., Sun, Q., Xu, J., Dai, Q., Zhang, L.: A pid controller approach for stochastic optimization of deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8522–8531 (2018)
19. Zulkifli, H.: Understanding learning rates and how it improves performance in deep learning. Towards Data Science, 21, 23 (2018)
20. Bottou, L.: Online learning and stochastic approximations. On-line learning in neural networks, **17**(9), 142 (1998)
21. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International conference on machine learning, pp. 1139-1147. (2013)
22. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 2121-2159 (2011)
23. Hinton, G., Srivastava, N., Swersky, K.: Neural networks for machine learning lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude (2012).
24. Kingma, D. and Ba, J.: Adam: A method for stochastic optimization. In: International Conference for Learning Representations (ICLR) (2014)