applied sciences

MDPI

Article

Iterative Learning Method for In-Flight Auto-Tuning of UAV Controllers Based on Basic Sensory Information

Wojciech Giernacki

Institute of Control, Robotics and Information Engineering, Electrical Department, Poznan University of Technology, Piotrowo 3a Street, 60-965 Poznan, Poland; wojciech.giernacki@put.poznan.pl; Tel.: +48-61-665-23-77

Received: 31 December 2018; Accepted: 30 January 2019; Published: 14 February 2019

Abstract: With an increasing number of multirotor unmanned aerial vehicles (UAVs), solutions supporting the improvement in their precision of operation and safety of autonomous flights are gaining importance. They are particularly crucial in transportation tasks, where control systems are required to provide a stable and controllable flight in various environmental conditions, especially after changing the total mass of the UAV (by adding extra load). In the paper, the problem of using only available basic sensory information for fast, locally best, iterative real-time auto-tuning of parameters of fixed-gain altitude controllers is considered. The machine learning method proposed for this purpose is based on a modified zero-order optimization algorithm (golden-search algorithm) and bootstrapping technique. It has been validated in numerous simulations and real-world experiments in terms of its effectiveness in such aspects as: the impact of environmental disturbances (wind gusts); flight with change in mass; and change of sensory information sources in the auto-tuning procedure. The main advantage of the proposed method is that for the trajectory primitives repeatedly followed by an UAV (for programmed controller gains), the method effectively minimizes the selected performance index (cost function). Such a performance index might, e.g., express indirect requirements about tracking quality and energy expenditure. In the paper, a comprehensive description of the method, as well as a wide discussion of the results obtained from experiments conducted in the AeroLab for a low-cost UAV (Bebop 2), are included. The results have confirmed high efficiency of the method at the expected, low computational complexity.

Keywords: UAV; auto-tuning; machine learning; iterative learning; extremum-seeking; altitude controller

1. Introduction

1.1. Auto-tuning of UAV Controllers—Context and Novelty

Common availability of low-cost, computationally efficient embedded systems and small size sensors directly influence the development of the construction of unmanned aerial vehicles and their applications, the number of which has been increasing in recent years [1–4]. In every UAV prototype, the need to ensure reliability and flight precision, both in manual and autonomous mode, are key aspects and depend directly on the selection of sensors [5], estimation methods [6], and the quality of position and orientation by controllers resulting from the applied control architecture [7,8]. In addition to the advanced control systems that often require precise models of UAV dynamics [9–12], due to their simplicity and versality, fixed-value controllers with a small number of parameters, are commonly and successfully used [13–16]. They determine the safety of operation, maximum flight duration and the UAV's in-flight behavior. That is why it is so important to learn and systematize the mechanisms of optimal self-tuning of their parameters for various environmental disturbances and for a radical change in the dynamics of the UAV itself due to a change in its total mass. Due to the attractive field of applications of such solutions in many areas (transportation and manipulation tasks performed by one or several UAV units [17–19], precision agriculture [20,21], missions requiring the sensory equipment to be re-armed, rescue operations [22], etc.), one is looking for fast solutions with low computational complexity that work in real-time mode.

While the state-of-the-art analysis shows several computationally complex approaches (requiring numerous repetitions and the use of the UAV model) to batch, optimal auto-tuning of controllers (via heuristic bio-inspired [23,24] and deterministic methods [25]), there has been no method reported to optimize the gains of fixed-value UAV controllers so far. No method has also been reported to do the latter in flight, iteratively and exclusively on the basis of available, periodic, basic sensory information (without using the UAV model)—to indirectly increase the flight duration by minimizing the energy expenditure through shaping a smooth flight characteristic. This issue has been selected as the core of the conducted research. The obtained result in the form of effective machine learning method for auto-tuning of gains of UAV controllers is a novelty presented in this article and thoroughly expands the concept of the method presented in [26] (using the weighted sum of the control error and control signal in predefining expectations for time courses and as a measure of tracking quality in the optimization algorithm). In addition, the most important added value also became:

- assessment and systematization (by means of simulation and experimental studies) of the influence of several environmental factors on the process of auto-tuning of UAV controllers during the flight by the proposed extremum-seeking method. The key issue here is the analysis of results in terms of assessing the quality of work of tuned controllers and the work of the optimization mechanism itself in the following test areas: presence of disturbances (wind gusts), UAV mass change, different sensory sources, flight dynamics/optimized performance index,
- outlining the rules for conducting the auto-tuning process of controllers, so that the automatic exploration of the gain space for individual controllers can be as safe as possible (one needs to keep in mind that the proposed method is not based on any stability criterion, which is its main limitation compared with numerous batch solutions based on models).

1.2. Motivation

In previous research [26], the author has drawn inspiration from the demanding problems of mobile robotics, which the world research centers have been coping with. Examples of such problems can be found in particular challenges of the Mohamed Bin Zayed International Robotics Challenge (http://www.mbzirc.com) [27], where the common denominator are tasks requiring the use of one or a group of UAV units to conduct autonomous flights with high precision in varied conditions (outdoor and indoor) and varied UAV mass. In preparation for the MBZIRC'2020 edition, it turned out that the only currently available auto-tuning algorithm on commercial auto-pilots (as Pixhawk, Naze32, Open Pilot, CC3D), named `AutoTune`, *"(...) uses changes in flight attitude input by the pilot to learn the key values for roll and pitch tuning. (...) While flying the pilot needs to input as many sharp attitude changes as possible so that the autotune code can learn how the aircraft responds"* [28]. Unfortunately, this solution is problematic due to the tuning safety (especially in prototyping UAV constructions) and control goal set: to provide the most smooth, feasible flight trajectories, which will reduce the control effort to reasonable level, and as a result will be maximally energy efficient. Therefore, in the method considered in this work, a gain tuning of UAV controllers based on dynamic behavior was replaced by more energy-efficient and automatic machine learning technique.

1.3. Related Work

Among numerous approaches to machine learning, and apart techniques using neural networks, which require many learning data sets, the mechanisms based on reward and punishment (as in the case of reinforcement learning approaches) are becoming increasingly common. In [29], Rodriguez-Ramos et al. have taught the control system to land autonomously on a moving vehicle, and in [30] Koch et al. trained a flight controller attitude control of a quadrotor through reinforcement learning. Despite the obviously large number of classic approaches to tuning of fixed-value controllers (Panda presents a whole array of such approaches, of which several dozen are practice-oriented [31]), the optimal techniques of iterative learning are invariably gaining on importance [32–34]. Iterative learning techniques have three desirable attributes, namely: automated tuning, low computational complexity (in optimization algorithms, a decision is made only on the basis of current, cyclic information from the selected performance index—cost function), and fast tuning speed [26,35] (in contrast to reinforcement learning approaches, which requires numerous experiments during the learning that makes it unpractical).

While the methods approximating the gradient of the cost function (first- and second-order optimization algorithms) presented in [25,36] can be quite problematic for UAV auto-tuning from noisy measurements (an aspect for careful comparisons in subsequent author's research), the zero-order optimization methods works efficiently because of the speed of calculations. However, it should be remembered and accepted that the obtained solution may be a local (there is no guarantee to obtain global solutions) or a value near it (depending on the declared level of expected accuracy of calculations ϵ).

Among the zero-order optimization methods presented by Chong & Zak in [25], such as Fibonacci-search, golden-search, equal division, and dichotomy algorithms, especially the first two of region elimination methods—developed by Kiefer [37] are effective in optimal control problems [38]. A broad description of the method based on Fibonacci numbers which was used for UAV altitude controller tuning can be found in the mentioned publication [26] of the author—especially mathematical basics and proofs for the region elimination mechanisms. Therefore, for undisturbed presentation of the proposed new method based on the modified golden-search algorithm used in the auto-tuning of the altitude controller during the UAV flight, only necessary mathematical description has been presented in the remaining part of the paper. Instead, the author paid more attention to the application aspects of the method (by placing the necessary pseudocodes) and a wide analysis of the results obtained from the conducted research experiments.

The paper is structured as follows: in Section 2 the UAV description as a control object and measurement system, as well as considered control system, is presented. Therein, the control purpose is highlighted, and the optimization problem is outlined. In the same Section, the proposed auto-tuning method is introduced, and its mathematical basics are explained. Furthermore, the experimental platform is shown. The comprehensive description of simulation and real-world experiments results with discussion are provided in Section 3. Finally, Section 4 presents conclusions and further work plans.

For a better understanding of the presented content, the most important symbols used in the paper are described in Table 1.

Table 1. Symbols used in this article.

Symbol of Variable	Explanation
α, β	weights (in cost function J)
θ, ϕ, ψ	*roll*, *pitch*, *yaw* angles
ρ	golden-search reduction factor
ϵ	expected accuracy in GLD method
$\mathcal{BF}$	body frame of reference
$\mathscr{D}^{(k)}$	considered range for the optimized parameter at k-th iteration
$\mathcal{EF}$	Earth frame of reference
$e(t)$	tracking error (in time domain)
$f(\cdot)$	cost function (in GLD method)
J	performance index (cost function in GLD procedure)

Table 1. *Cont.*

Symbol of Variable	Explanation
k_P, k_I, k_D	proportional/integral/derivative gains
N	minimal number of iterations required to ensure accuracy ϵ
N_b	number of the predefined bootstrap cycles
N_c	number of sampling periods necessary to calculate J at l-th iteration of the GLD
N_{max}	number of sampling periods related to the length of the tuning procedure
$\underline{p}_d$	vector of desired UAV position
$\underline{p}_m$	vector of measured UAV position
t_a	time of gathering information for calculation of J in GLD methods
t_h	flight time horizon
T_f	time constant of a low-pass filter of transfer function
T_p	sampling period for calculation of J
T_s	sampling period in low-pass filter
$u(t)$	control signal (in time domain)
$x^{(k^-)}$	lower bound for optimized parameter at k-th iteration
$x^{(k^+)}$	upper bound for optimized parameter at k-th iteration
$\hat{x}$	candidate point in the optimization procedure
$\hat{x}^*$	iterative estimate of the optimal solution
x_b, y_b, z_b	axes of the $\mathcal{BF}$
x_d, y_d, z_d	desired position coordinates
x_e, y_e, z_e	axes of the $\mathcal{EF}$
x_m, y_m, z_m	measured position of the UAV

2. Materials and Methods

2.1. Multirotor UAV as a Control Object and Its Measurement System

The multirotor UAV can be considered as a multidimensional control plant, being underactuated, strongly non-linear, and highly dynamic with (in general) non-stationary parameters. These features result from its physical structure—especially the use of several propulsion units mounted at the ends of the frame. In addition, measuring, processing, and communication systems are also attached to the middle of this frame—suited for a particular UAV construction. From the perspective of control, the appropriate selection of propulsion units (composed of: brushless direct current motors, electronic speed controllers and propellers) is a key aspect to ensure the expected flight dynamics expressed via thrust (T) and torque ($\underline{M}$) generated by the rotational movement of propellers [39]. By changing the rotational speed, it is possible to obtain the expected position and orientation of the UAV in 3D space, i.e., control of its 6 degrees of freedom (DOFs). The obtained control precision also depends on the quality of sensory information to a large extent. Presently, even in the simplest, low-cost UAVs (Figure 1), in order to determine current position and orientation estimates during the flight (e.g., based on more or less advanced modifications of Kalman filters [6]), the sensory data fusion is used (from 3-axes accelerometer, 3-axes gyroscope, 3-axes magnetometer, pressure sensor, optical-flow sensor, GPS, ultrasound sensor, etc.).

In the paper, two sources of measurements are used in the proposed auto-tuning procedure: on-board UAV avionics (for *roll* and *pitch* angles measurements) and external motion capture system (`OptiTrack`) (X, Y, Z position, and *yaw* angle). In the UAV autonomous control, to ensure unambiguous description of the UAV's position and orientation in 3D space, the North-East-Down (NED) configuration of the reference system is used, since the on-board measurements are expressed in local coordinate system ($\mathcal{BF}$—Body Frame), and the position control, as well as motion capture measurements are defined in the global one ($\mathcal{EF}$—Earth Frame). In the paper of Xia et al. [40], one may find a better known, basic information about the mechanisms of conversions, e.g., how the posture of the multirotor (its rotational and translational motion) can be described by the relative orientation between the $\mathcal{BF}$ and the $\mathcal{EF}$ with the use of the rotation matrix $R \in SO(3)$.

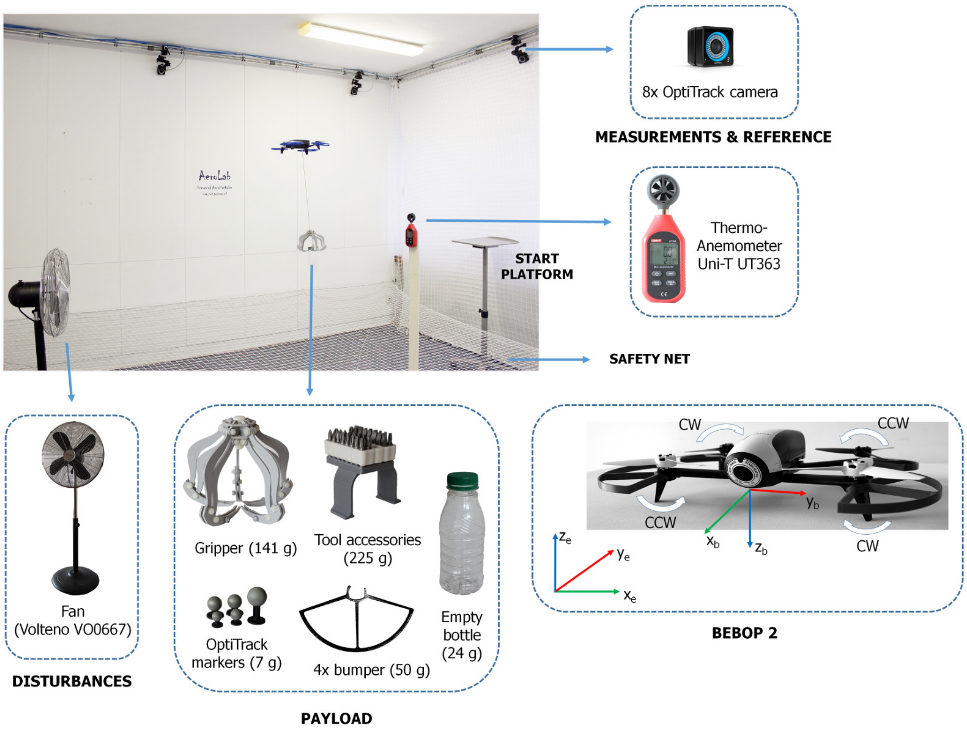

Figure 1. The *Bebop 2* quadrotor (and its coordinate system) during one of the initial experiments with the carrying of payload conducted in *AeroLab* of *Poznan University of Technology*.

2.2. Considered Control System and Control Purpose (Formulation of Optimization Problem)

The control system of multirotor UAV from Figure 2 considered here is based on cascaded control loops. There is control of angles *roll* (θ) and *pitch* (ϕ) around the x_b and y_b axes, according to the set (desired) position in the x_e and y_e axes in faster, internal control loops. Their control is performed in slower external loops. The control of θ and ϕ angles occurs indirectly in the realization of autonomous flight trajectory expressed using the vector of desired position trajectory $\underline{p}_d = (x_d, y_d, z_d)^T$ and desired angle of rotation *yaw* (ψ_d) around the z_e axis. The purpose of the autonomous control is then to ensure the smallest tracking errors $e(t)$ during the UAV flight, i.e., the difference in the values of the reference signals (desired) and output signals (actual/measured) [41]:

$$\underline{e}_p = \underline{p}_d - \underline{p}_m, \tag{1}$$

$$e_\psi = \psi_d - \psi_m, \tag{2}$$

where the m index refers to the measured values.

Bearing in mind that in UAVs the current tracking error information from (1) and (2) is used as the input of a given fixed-value controllers, in the commonly used proportional-derivative (PD) controller structure or proportional-integral-derivative (PID), it is proposed to use this information (as well as information from the output of a given type of controller with control signal $u(t)$) to formulate a measure of the tracking quality during UAV flight, i.e., the cost function/performance index $J(t)$ (see Figure 3), defined as follows:

$$J\left(t\right) = \int_0^{t_a} \left(\alpha \left|e\left(t\right)\right| + \beta \left|u\left(t\right)\right|\right) dt, \tag{3}$$

where t_a is the time of gathering information (to calculate new controller gains) in the optimization procedure. By introducing the penalty for excessive energy expenditure (expressed in the cost function through actual values of the control signal $u(t)$), it is possible to shape expectations towards transients and the controller's dynamics profile (providing smooth or dynamic flight trajectories). At small values of the β, the controller works aggressively, using more energy, often at the expense of the appearance of overshoot, which is undesirable in missions and tasks requiring high flight precision.

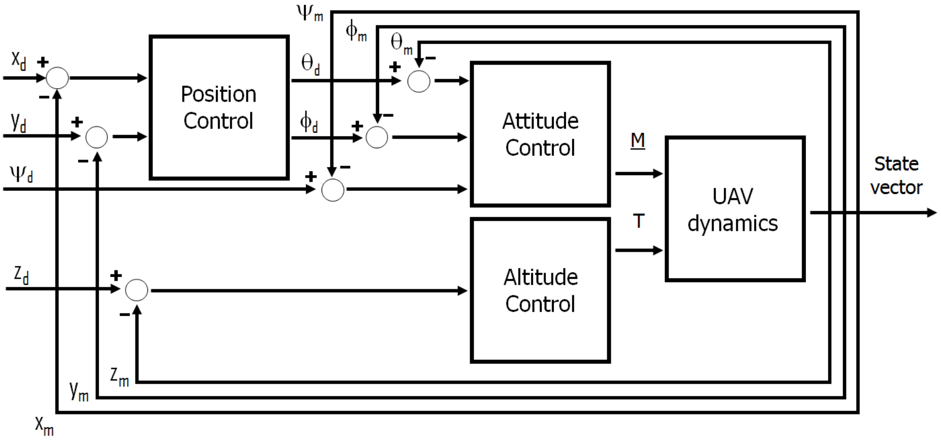

Figure 2. Diagram of considered control system.

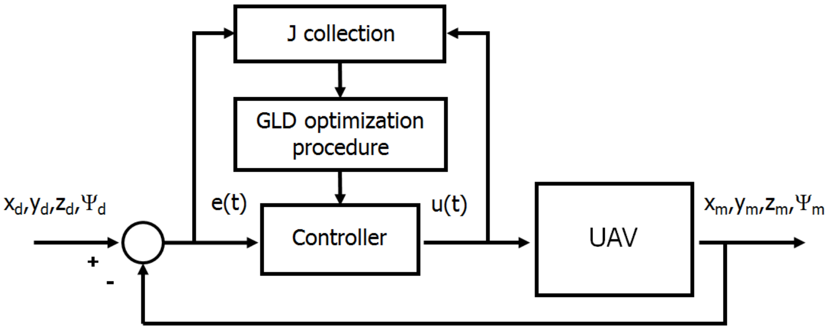

Figure 3. General block diagram of the control system with optimization.

Unconstrained control signal $u(t)$ is calculated from the controller's equation, which in the case of PID structure it is given by

$$u(t) = k_P e(t) + k_I \int_0^{t_h} e\,(t)\,dt + k_D \frac{d}{dt} e(t), \tag{4}$$

where t_h is a flight time horizon, k_P is the proportional gain, k_I represents the integral gain and k_D the derivative gain, respectively. Gains k_P and k_D are expected to be found using the proposed iterative learning method.

Remark 1. *In the article, when there is a reference to the PID controller, it should be remembered that only the k_P and k_D gains are tuned automatically, whereas the value k_I (used to eliminate the steady-state error) is selected in a manual manner. The proposed auto-tuning method can be used to optimize the gains of any type of controller with three (or even more) parameters; however, this will result in a longer tuning time. Therefore, from the application point of view, it is better to use the procedure presented further in the article.*

Recalling (4), this work deals with the search for the controller gains k_P and k_D, to minimize the cost function (3). That is, the current controller design procedure can be posed as an optimization problem where the solution to the following problem is sought:

$$\min_{k_1,k_2,\ldots,k_N} \quad J(t) = \int_0^{t_a} \left(\alpha\,|e(t)| + \beta\,|u(t)|\right) dt, \qquad \text{s.t.} \quad \begin{array}{c} 0 \le k_1 \le k_1^{max} \\ 0 \le k_2 \le k_2^{max} \\ \ldots \\ 0 \le k_N \le k_N^{max} \end{array} \tag{5}$$

where k_1^{max}, k_2^{max}, ..., k_N^{max} are upper bounds of the predefined ranges of exploration in the optimization procedure of N controller parameters.

Remark 2. *In the numerical implementation of optimization problem from (5), to quantify the tracking quality by using the cost function (3), its discrete-time version is used (the integration operation is replaced with the sum of samples). Then the cost function is built from the weighted sum of the absolute values of the tracking error samples and the absolute values of the control signal samples (for a given sampling period T_p).*

2.3. Procedure for Tuning of Controllers

To increase the safety in the process of tuning UAV controller parameters during the flight, it is proposed to use the procedure from the flowchart (Figure 4), corresponding to the pyramid of subsequent expectations for the work of control system (Figure 5).

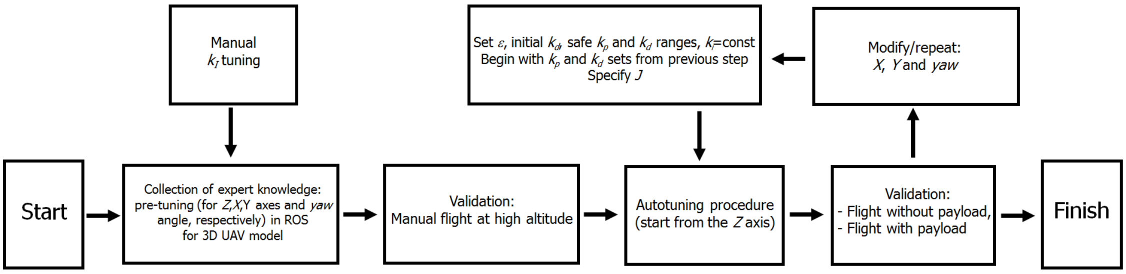
Manual
k_I tuning
Set ε, initial k_d, safe k_p and k_d ranges, k_I=const
Begin with k_p and k_d sets from previous step
Specify J
Modify/repeat:
X, Y and yaw
Start
Collection of expert knowledge: pre-tuning (for Z,X,Y axes and yaw angle, respectively) in ROS for 3D UAV model
Validation:
Manual flight at high altitude
Autotuning procedure (start from the Z axis)
Validation:
- Flight without payload,
- Flight with payload
Finish

Figure 4. Flowchart for the proposed tuning strategy of the UAV controllers.

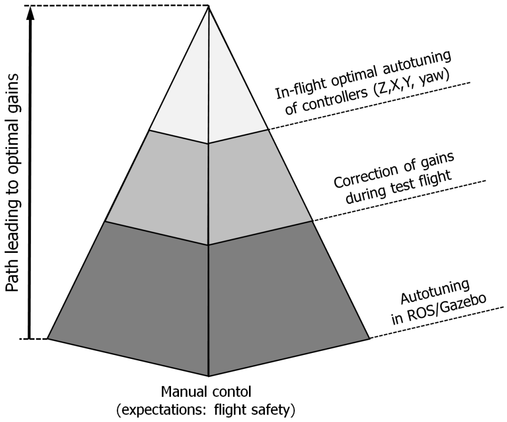
Path leading to optimal gains
In-flight optimal autotuning of controllers (Z,X,Y, yaw)
Correction of gains during test flight
Autotuning in ROS/Gazebo
Manual contol
(expectations: flight safety)

Figure 5. Following steps to obtain optimal gains of controllers.

Remark 3. *Manual tuning of UAV control system prototype is out of scope of this work (to focus on auto-tuning mechanisms). Some useful information regarding UAV controllers prototyping can be found at well-recognized by the UAV community webpages [42–44].*

2.4. Iterative Learning Method for In-Flight Tuning of UAV Controllers—General Idea

Bearing in mind that the search space for the J_{min} for all combinations of controller gains $\underline{k} = (k_1, k_2, \ldots, k_N)^T$ in predefined intervals (ranges) of gains, in the problem outlined in (5) is huge, one needs a fast, effective mechanism for search space exploration. It should be characterized by low computational complexity, and after checking the value of $J(t)$ in a maximum of dozen or several dozen of gain combinations, should be able to provide the value of J_{min} (locally best variant) or a significant improvement compared to the controller's original gains (expressed using e.g., the expected accuracy of the ϵ).

Recalling the publications cited in Section 1.3, iterative learning algorithms are characterized by fast convergence towards the minimum value—especially the region elimination methods (REMs). To be able to use them, one needs to refer to the general idea of iterative learning approaches (proposed by Arimoto et al. in [45]), i.e., minimization of the norm of error (here: cost function) in order to tune particular controller using the periodical repetitiveness of the trials (here: repetitions of the same, predefined trajectory primitives—see Figure 6). Then, to find locally best gains of a particular controller based on a given reference of x_d, y_d, z_d or ψ_d primitive, and corresponding measured value, the performance index $J(t)$ calculated during the flight with given safe ranges of controller parameters, enables the minimum-seeking procedure to find controller gains with respect to the preferred dynamics, and for a given tolerance of the solution.

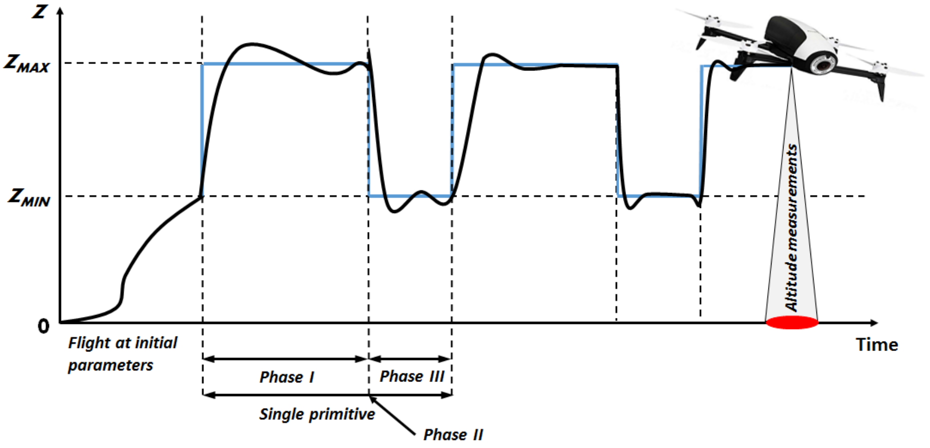

Figure 6. Following steps in iterative learning mechanism for altitude controller tuning.

Remark 4. *Since the method is based solely on the cyclic collection of measurement data to determine $J(t)$, the need to use the UAV model is reduced. However, its knowledge is advantageous when in the simulation conditions it is possible to roughly estimate/determine the maximum values of the elements of the $\underline{k}$ vector, for which the UAV does not lose its stability.*

For every single primitive being used in the optimization procedure, three phases can be distinguished (see Figure 6):

1. Acquisition of measurement data (current, sampled values: x_m, y_m, z_m or ψ_m) for set controller gains with the assumed T_p and the assumed form of the $J(t)$ function,
2. Determination of new controller gains based on the estimated value of the cost function from the phase no. 1,
3. Adjusting the controller according to iteratively corrected gains and waiting for the time necessary to stop the transient processes caused by it.

Determining the sequence of controller gains is possible by systematically narrowing the search space. For this purpose, the use of region elimination method based on the zero-order deterministic optimization algorithm (GLD), is proposed.

2.5. Region Elimination Method Based on GLD Algorithm

Let us consider the problem of iterative searching for a particular controller's gain as a problem of reducing the set range of this gain, in which the criterion of stopping the algorithm is the proximity of following solutions, i.e., the value of the cost function in subsequent solutions (for subsequent controller gains), and the convergence of the algorithm ensures the use of the mechanism based on golden-section search from [25] used in REMs.

Principles and assumptions in the GLD method are similar to those in the modified Fibonacci-search method (FIB) proposed in [26]. The most important are: the unimodality assumption of optimized cost function $f(\cdot)$, lack of knowledge about the global minimum (which gave rise to formulation of stopping criteria in the iterative tuning algorithm, e.g., given tolerance to find the minimizer), successively narrowing the range of values inside which the extremum is known to exist according to the definition (2.1) of the fundamental rule for REMs.

Definition 1. *Let us consider an optimization problem of a one-argument unimodal cost function* $f : \mathscr{R} \rightarrow \mathscr{R}$ *within in the predefined range* $[x^{(0^-)}$,$x^{(0^+)}]$ *in initial* 0*th iteration, where* $x^{(0^-)}$<$x^{(0^+)}$ *of a unimodal function* f*. The argument* x *of this function can be interpreted as a gain of controller (here:* k_P *or* k_D*), and the value of* f *can be understood as the* J *value (within some horizon) corresponded to it.*

Now, for a pair of two arguments $x^{(1^-)}$ *i* $x^{(1^+)}$*, which lie in the range* $[x^{(0^-)}$,$x^{(0^+)}]$*, and which satisfy* $x^{(0^-)}$<$x^{(1^-)}$<$x^{(1^+)}$<$x^{(0^+)}$*, it is true that:*

- *If* $x^{(1^-)} > x^{(1^+)}$*, then the minimum* $\hat{x}^*$ *does not lie in* $(x^{(0^-)}, x^{(1^-)})$*,*
- *If* $x^{(1^-)} < x^{(1^+)}$*, then the minimum* $\hat{x}^*$ *does not lie in* $(x^{(1^+)}, x^{(0^+)})$*,*
- *If* $x^{(1^-)} = x^{(1^+)}$*, then the minimum* $\hat{x}^*$ *does not lie in* $(x^{(0^-)}, x^{(1^-)})$ *and* $(x^{(1^+)}, x^{(0^+)})$*.*

A region elimination fundamental rule is used to find the $\hat{x}^*$ with the minimum value of f within predefined range, based on repeatedly selection of two arguments from the current range according to symmetrically reduction the range of possible arguments:

$$x^{(1^-)} - x^{(0^-)} = x^{(1^+)} - x^{(0^+)} = \rho(x^{(0^+)} - x^{(0^-)}), \tag{6}$$

where $\rho = \frac{3-\sqrt{5}}{2} = 0.381966$ is a golden-search reduction factor.

Remark 5. *An advantage of using the golden-search reduction factor (according to Algorithm 1) is the fast exploration of the interval, because following values of* x *(controller gains) are selected to use one of the values of the cost function calculated in the previous iteration. For this purpose, the interval is divided regarding the golden ratio. As a result, of the use of the golden-search reduction factor for a given interval, two new sub-intervals are obtained. For the new intervals, the ratio of the longer length to the shorter length is equal to the ratio of the length of the divided interval to the length of the longer interval.*

Due to this mechanism, and by using the golden-search reduction factor, the time of range exploration is shortened (through the reduction the number of points for which f *function needs to be evaluate) or alternatively the* f *function values can be averaged for the same* x *in following iterations, i.e.,* $x^{((k+1)^+)}$ *and* $x^{(k^-)}$*, which is useful in order to reduce the impact of measurement disturbances during the UAV outdoor flight.*

Based on predefined initial range $x \in \mathscr{D}^{(0)} = \left[x^{(0^-)}, x^{(0^+)}\right]$, the golden-search algorithm can be implemented according to the pseudo-code presented below (Algorithm 1).

Algorithm 1 Golden-search algorithm.

Step 1. Evaluate the minimal number N of iterations required to provide the sufficient (predefined) value of the ϵ:

$$|x^* - \hat{x}^*| \leq \epsilon(x^{(0^+)} - x^{(0^-)}), \tag{7}$$

where $|x^* - \hat{x}^*|$ is the absolute value of the difference between the true (unknown minimum x^*) and iterative solution $\hat{x}^*$ (which is assumed to be in the center of $\mathscr{D}^{(N)}$).

Step 2. For iteration $k = 1, \ldots, N$,

1) select a pair of intermediate points $\hat{x}^{(k^-)}$ and $\hat{x}^{(k^+)}$ ($\hat{x}^{(k^-)} < \hat{x}^{(k^+)}$, $\left\{\hat{x}^{(k^-)}, \hat{x}^{(k^+)}\right\} \in \mathscr{D}^{(k-1)}$),
2) reduce the range to $\mathscr{D}^{(k)}$ based on REM fundamental rule:
    a) $x^{(k+1)} \in \mathscr{D}^{(k)} = \left[x^{(k-1^-)}, \hat{x}^{(k^+)}\right]$ for $f(\hat{x}^{(k^-)}) < f(\hat{x}^{(k^+)})$,
    b) $x^{(k+1)} \in \mathscr{D}^{(k)} = \left[\hat{x}^{(k^-)}, x^{(k-1^+)}\right]$ for $f(\hat{x}^{(k^-)}) \geq f(\hat{x}^{(k^+)})$,
    c) start next iteration $k := k + 1$.

Step 3. Stop the algorithm; put $\hat{x}^* = \frac{1}{2}\left(x^{(N^+)} + x^{(N^-)}\right)$.

For the given value of ϵ, the minimum number N of iteration in the GLD algorithm can be calculated according to:

$$(1 - \rho)^N \leq \epsilon, \tag{8}$$

and for $k = 1, \ldots, N$ one may find the pair of intermediate points using

$$\hat{x}^{(k^-)} = x^{(k-1^-)} + \rho(x^{(k-1^+)} - x^{(k-1^-)})\,, \tag{9}$$

$$\hat{x}^{(k^+)} = x^{(k-1^-)} + (1 - \rho)(x^{(k-1^+)} - x^{(k-1^-)})\,. \tag{10}$$

2.6. Optimal Gain Tuning of a Two-Parameter Controller Based on Bootstrapping Mechanism

In a two-dimensional space of parameters, the vector of parameters $\underline{x} = \begin{bmatrix} x_1, & x_2 \end{bmatrix}^T$ for the cost function $f(\underline{x})$ (calculated from in-flight measurements) can be interpreted as controller gains (here: k_P and k_D). For fast exploration of this space and to give a global character the GLD extremum-seeking procedure, Algorithm 2 is proposed. It is based on the bootstrapping mechanism (see Table 2), for the predefined bootstrap cycles N_b. In considered two-parameter controller tuning, in every single bootstrap, two launch of GLD algorithm (for each of controller gains) are executed to obtain expected value of the ϵ. Firstly, the gain no.1 is tuned (while the gain no. 2 is fixed), and then, the gain no. 2 (for fixed value of the no. 1).

Algorithm 2 Two-parameter controller tuning.

Step 0. Put the bootstrap cycles counter to $l = 0$; for initial $\mathscr{D}_i^{(l)}$ $(i = 1, 2)$ define ϵ, N_b, and initial value of the second parameter $x_2^{(l)}$ (take $\hat{x}_2^{(l)^*} = x_2^{(l)}$), set $l := l + 1$.

Step 1. Find the optimal $\hat{x}_1^{(l)^*}$ using the GLD algorithm, with the second parameter fixed at $\hat{x}_2^{(l-1)^*}$.

Step 2. Calculate the optimal $\hat{x}_2^{(l)^*}$ analogously to the method from the Step 1, keeping the first parameter fixed at $\hat{x}_1^{(l)^*}$.

Step 3. If $l < N_b$, increase the bootstrap cycles counter $l := l + 1$, and proceed to Step 1, otherwise stop the algorithm—the optimal solution $\hat{\underline{x}}^* = \begin{bmatrix} \hat{x}_1^{(l)^*}, & \hat{x}_2^{(l)^*} \end{bmatrix}^T$ has been obtained after N_b bootstrap cycles, as desired.

Table 2. Steps in the bootstrapping mechanism.

Bootstrap No.	Gain No.1	Gain No.2
1	Tuning (according to the GLD REM)	Kept constant
1	Kept constant	Tuning
2	Tuning	Kept constant
2	Kept constant	Tuning
...	...	...
N_b	Tuning	Kept constant
N_b	Kept constant	Tuning

To ensure high effectiveness of the proposed method of auto-tuning, one should remember about several important aspects (in configuration and implementation):

- The proposed method requires predefining the initial, admissible ranges for $\underline{x}$, i.e., $\mathscr{D}_i^{(0)} = \left[x_i^{(0^-)}, \quad x_i^{(0^+)}\right]$ for $i = 1, 2$. It is a crucial choice from the perspective of ensuring the safety of autonomous flight. If there is a such a possibility, it is strongly recommended to use the expert knowledge about the controller gains (from initial flights on the base of analysis of a rise time and the maximum overshoot, prototyping in virtual environment, default settings of on-board controller, detailed analysis of the UAV feedback control system, etc.),
- For the expected tolerance ϵ, the number N is calculated. $2N$ calculations of f are needed in the tuning of a pair of controller parameters of a single bootstrap,
- The algorithm's execution time depends on: N_b, N, and the time of a single reference primitive, which must be correlated with the expected UAV dynamics and its natural inertia,
- Recalling the most important principles of the zero-optimization method from [26], one needs to have in mind that the proposed method *"(...) is iterative-based and collects information about the performance index (on incremental cost function value) at sampling time instants, equally spaced every T_p seconds"* during the tuning experiments. Thus, for sampling period T_p, a single evaluation of f value according to Step 2 of Algorithm 1 with a change of a single parameter of controller is performed using Procedure 1 (for symbols from the Table 1).
- The performance index is calculated as

$$\Delta J^{(n)} = J^{(n+1)} + \Delta J^{(n)}, \tag{11}$$

where $\Delta J^{(n)}$ can be obtained from the discrete-time version of Equation (3), which for n-th sample (tracking error and control signal) at time $t = nT_p$ is given by

$$\Delta J^{(n)} = \alpha \left| e_n \right| + \beta \left| u_n \right| . \tag{12}$$

Algorithm 3 Evaluation of performance index (with single change of controller parameter) [26]

Recalling defined N_c, N_{max}, and n for $f(\cdot)$. Then:

- for $n = 1, \ldots, N_c - 1$ with the controller parameters are updated in the previous iteration, the performance index is evaluated using (11) by adding (12); set $J^{(0)} = 0$;
- for $n = N_c$ a single iteration of GLD algorithm is initialized, cost function is stored, and if possible—reduce the range for controller parameters or perform the bootstrap; it results in a transient behavior of the dynamical signal;
- for $n = N_c + 1, \ldots, N_{max}$ tuning is not performed; the controller parameters have been updated; no performance index is collected; transient behavior should decay.

2.7. Signals Acquisition and Their Filtration in the Proposed Method

Bearing in mind that in general to determine the performance index, sensory information is used from sources with different precision of estimation of the position and orientation of an UAV, therefore in the auto-tuning procedure it is proposed to use:

- the signals from the UAV odometry—processed using commonly used Kalman filtration. Thanks to that, it is possible to fuse data from several standard UAV on-board sensors,
- low-pass filtration (presented and tested primarily in [26]), expressed by a transfer function of first-order inertia type

  $$G(s) = \frac{k}{1 + T_f s}, \tag{13}$$

  where k is its gain, and T_f is a chosen time constant (here: $k = 1$, $T_f = 0.1$ sec.

  For the implementation of the GLD method, the discretized, recursive version of the low-pass filter (13) for the chosen sampling period T_s, is used:

  $$y\,(n) = a\,(n-1) + (1-a)\,u\,(n-1)\,, \tag{14}$$

  where

  $$a = \exp\left(-T_s/T_f\right), \tag{15}$$

  and $y(n)$ and $u(n)$ are filtered and pure errors at sample n, respectively.
- (optional) measurement information from an external high-precision measurement system—for example, the motion capture system (for indoor flights) or GNSS (outdoor), treated as the ground truth in estimating the difference to UAV avionics measurements.

2.8. Experimental Platform

In the real-world experiments, the low-cost, micro quadrotor Bebop 2 from Parrot company, was used (see Figure 1 and [46]). Since it is equipped in P7 dual-core CPU Cortex 9 processor, 1 GB RAM memory, and 8 GB of flash memory, it is possible to perform on-board state estimation of the UAV using Extended Kalman Filter (EKF) for the data gathered from its on-board sensors listed in Table 3. The Bebop 2 uses the Busybox Linux operating system. Compact sizes of the UAV (33 × 38 × 3.6 cm with hull) and efficient propulsion units (4 × 1280 KV BLDC Motor, 7500-12000 rpm), in combination with 2700 mAh battery provide maximum flight time up to 25 minutes and maximum load capacity up to 550 g (which gives a maximum takeoff mass equal to 1050 g, since the UAV weighs 500 g).

Table 3. General characteristic of Bebop 2 sensors.

Parameter	Value
accelerometer & gyroscope	3-axes MPU 6050
pressure sensor (barometer)	MS5607 (analyses the flight altitude beyond 4.9 m)
ultrasound sensor	analyses the flight altitude up to 8 m
magnetometer	3-axes AKM 8963
geolocalization	Furuno GN-87F GNSS module (GPS+GLONASS+Galileo)
Wi-Fi Aerials	2.4 and 5 GHz dual dipole
vertical stabilization camera	photo every 16 ms
camera	14 Mpx 3-axis Full HD 1080p with Sunny 180 fish-eye lens: 1/2.3″

All experimental studies discussed in the article were carried out in `AeroLab` [47], the research space created at the *Institute of Control, Robotics and Information Engineering of Poznan University of Technology* for testing solutions in the field of UAVs flight autonomy, where ground truth is the OptiTrack motion capture system equipped with 8 Prime 13W cameras (with markers placed on the UAV), and a processing unit (PC) equipped with `Motive`—OptiTrack's unified motion capture software platform. The measurement program (Robot Operating System (ROS) node) is executed with the frequency of 100 Hz, control actions with 30 Hz, whereas the tuning methods with 5 Hz. The system is connected to the ground station (Figure 7) to which information about the current position and orientation of the UAV (from motion capture system and UAV) are transmitted. The ground station is the Lenovo Legion Y520 notebook, equipped with Intel Core i7-7700HQ (2.8 GHz frequency), 32 GB DDR4 RAM memory, SSD hard drive and GeForce GTX 1050 2048 MB under Linux Kinetic 16.04 LTS operating system. Such a powerful computer was proposed for the autonomous control of the Bebop 2 UAV, to conduct all necessary calculations at the ground station, including: path planning, data (measurements) processing, autonomous control, auto-tuning of controllers, safety control, etc.

The ground station was also used for tests of the proposed GLD auto-tuning method in simulation environment. These tests were carried out under the control of the ROS, using the open-source flight simulator `Sphinx` [48] and `bebop_autonomy` library [49] extended by models of cascade control system enabling simulation of autonomous flights in x_e and y_e axes(flight for the given coordinates). In the external position control loops, the PID-type controllers have been used.

During the flights, to ensure the safety, Bebop 2 was equipped with 4 bumpers (12.5 g each, made in 3D printing technology) protecting propellers, and in `AeroLab` an additional horizontal safety net was installed to protect it against hard crashes to the ground level. In addition, for security reasons, the priority over the autonomous flight of the drone was allocated to the operator equipped with SkyController 2, enabling manual flight control. Furthermore, a safety button was introduced to cut off the UAV power supply in a situation of imminent danger. It supported initial experiments, where additional safety rope was used.

In experiments on variable mass flights, the UAV was also equipped with a plastic bottle and a gripper (made in 3D printing technology), or alternatively with tool accessories mounted directly on the Bebop (see Figure 1). Additionally, in studies on the influence of environmental disturbances on the auto-tuning process, the UT363 thermo-anometer from Uni-T company was used to measure the air flow speed generated from the Volteno VO0667 fan.

For the simulation and experimental results presented in the next section of the article, a movie clips (available at the webpage http:www.uav.put.poznan.pl), were prepared.

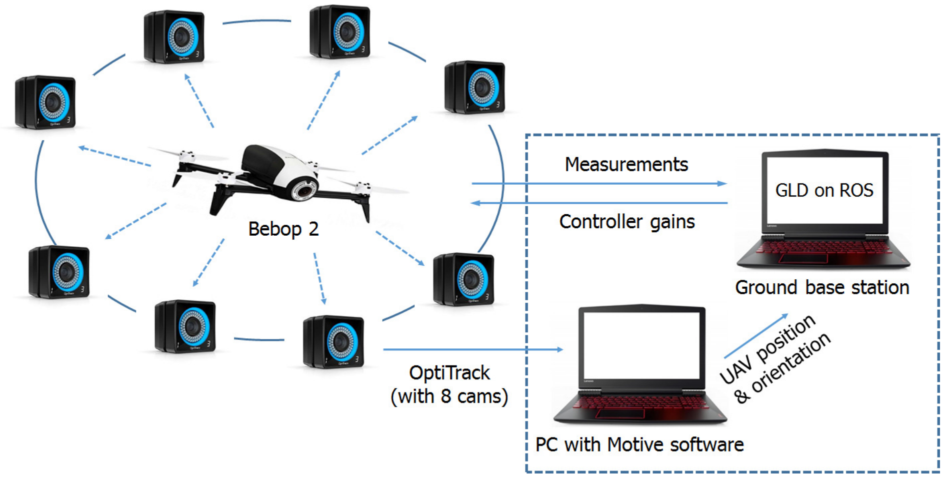

Figure 7. Simplified block diagram of measurement and control signal architecture used during the experiments with in-flight tuning of controllers.

3. Results and Discussion

3.1. Simulation Experiments

Let us consider the problem of searching locally best gains of the altitude PID-type controller of Bebop 2 unmanned aerial vehicle. Default gains are not made available by the Parrot company, hence the problem of finding the best gains (summarized in Table 4) has been treated at the prototyping stage. After development of the 3D model of this UAV (with bumpers) in the Blender software, it was implemented in the ROS/Gazebo environment, giving the physical dimensions, mass and moments of inertia from the real flying robot to its virtual counterpart embedded in the virtual `Aerolab` scenery. This enabled reliable preliminary experiments to be conducted in the simulator.

The research purposes were set as:

- recognizing the nature of optimized function $J = f(k_P, k_D)$ for its various structures (α = var, β = var),
- validation if given gain ranges of k_P and k_D (for a constant, very small value of $k_I = 0.0003$) are safe (i.e., if the closed-loop control system is stable),
- comparative analysis of the effectiveness of GLD and FIB methods.

Table 4. Gains of Bebop's controllers used in experiments.

	***X*-axis**	***Y*-axis**	***Z*-axis**	$\boldsymbol{\theta}$	$\boldsymbol{\phi}$	$\boldsymbol{\psi}$
k_P	0.69	0.69	1.32	default	default	0.07
k_I	0.00015	0.00015	0.0003	default	default	0.00001
k_D	50	50	10.2	default	default	0.9

In the first phase of the research, more than 33 hours of simulation tests were conducted. The results are presented in Figure 8. The same dynamics of the desired reference signal was set as in [26] for the FIB method. Every 12 seconds the UAV changed periodically the flight altitude (1.2→1.9→1.2 m). The value of J was being recorded for 10 sec repeatedly. For each combination of k_P and k_D gains, the J value was averaged from 5 trials. The results of 400 combinations of (k_P, k_D) were recorded for three various J functions. In none of the 2000 trials, the UAV model showed dangerous behavior, and as expected: higher values of k_P correspond to a better quality of reference signal tracking (lower values of J).

In the second phase of the research, the effectiveness of the GLD and FIB methods was compared for three initial values of the k_D and three J function structures. The very promising results are presented in Figure 8 and Table 5. Both methods effectively explore the gains space (k_P, k_D) in search for smaller values of J, avoiding the local minima (they do not "get stuck" in there)—see Figure 8 (right column). Depending on the set k_{Dinit} gain value, both the methods yield in similar k_P values, but various k_D, slowing down the expected tracking dynamics respectively (for larger values of β). It is particularly noteworthy to compare the signals for subsequent set values of β (Figure 9). Bearing in mind the diversity of UAV applications, it is possible to shape the "energy policy", i.e., through an introduction of larger values of β, one obtain a smooth, slower trajectory of the altitude signal, with a smaller control signal amplitudes (for which the β is punishing), which is conducive to extend the flight time.

In relation to the FIB method, an additional time of 96 sec (corresponding to 8 iterations of the auto-tuning algorithm), allows the GLD method in subsequent iterations only to slightly improve the value of the J performance index (respectively by 1.62%, 0.78%, and 2.10%). The introduction of the second bootstrap is justified in the FIB method (improvement by respectively: 11.92%, 4.95%, 1.40%), while in the case of the GLD method, just only one bootstrap provides similar results. The listings from the altitude controller auto-tuning process are available for both methods in the supplementary materials at the `AeroLab` webpage.

Table 5. Results of simulation experiments.

	FIB	**GLD**	**FIB**	**GLD**	**FIB**	**GLD**
α	1.0	1.0	0.9	0.9	0.8	0.8
β	0.0	0.0	0.1	0.1	0.2	0.2
k_{Dinit}	2.0	2.0	10.0	10.0	18.0	18.0
k_P range	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]
k_D range	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]
No. of bootstrap cycles	2	2	2	2	2	2
No. of main iterations	48	56	48	56	48	56
Tuning time [sec]	576	672	576	672	576	672
Low-pass filtration	yes	yes	yes	yes	yes	yes
ϵ	0.05	0.05	0.05	0.05	0.05	0.05
Best k_P and k_D values	3.56/8.70	3.70/8.18	4.63/5.99	4.49/8.58	4.23/10.51	4.39/15.44
J_1 (after the 1st bootstrap)	6.8235	5.4575	5.7596	5.4865	5.9641	5.8197
J_{48} (after 48 iter.)	6.0969	5.5024	5.4882	5.4492	5.8815	5.9059
J_{end} (after the tuning proc.)	6.0969	5.4149	5.4882	5.4068	5.8815	6.0298
J_{avg} (average for tuning proc.)	6.7264	5.4754	5.7730	5.5643	6.0345	5.9376

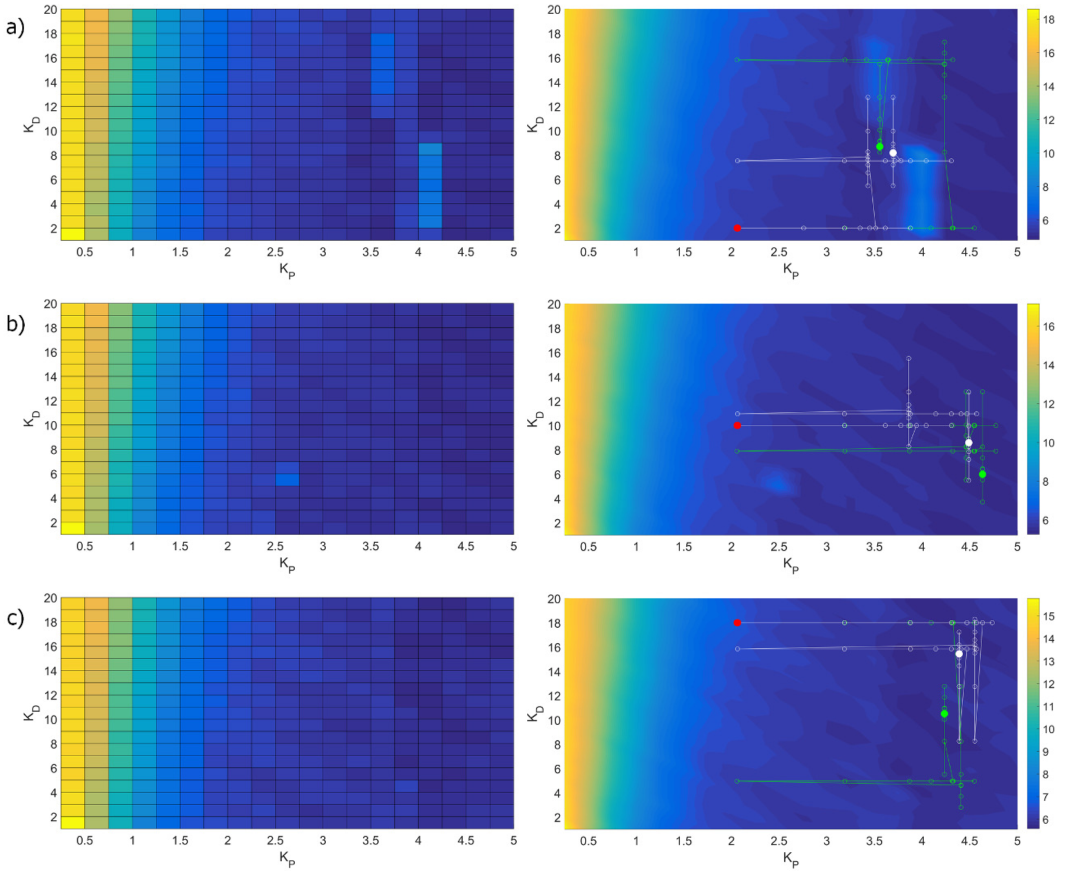

Figure 8. Obtained values of the J performance index for k_P and k_D combinations (left column) and $J = f(k_P, k_D)$ approximations (right column) for: (**a**) $\alpha = 1.0$, $\beta = 0.0$, (**b**) $\alpha = 0.9$, $\beta = 0.1$, (**c**) $\alpha = 0.8$, $\beta = 0.2$. FIB (green) and GLD (white) tuning results for: (**a**) $k_{Dinit} = 2$, (**b**) $k_{Dinit} = 10$, (**c**) $k_{Dinit} = 18$ (marked in red).

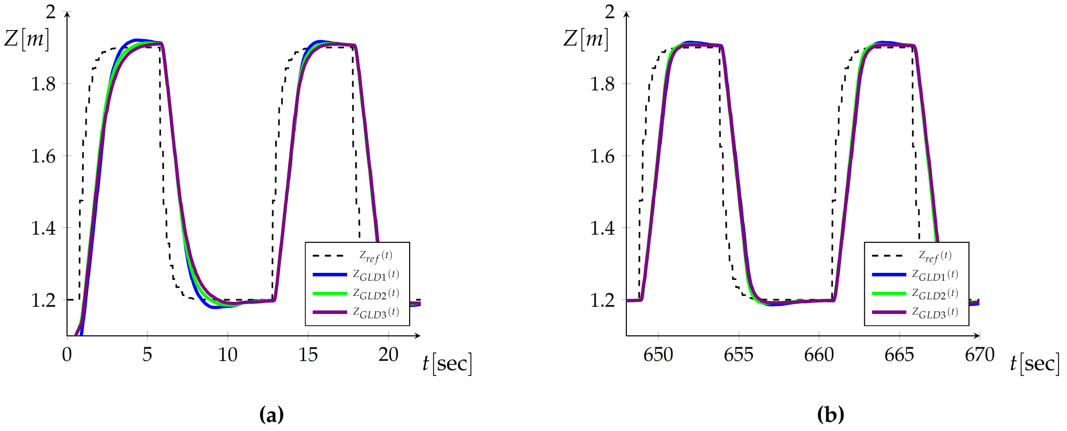

Figure 9. Time courses for: (**a**) first two iterations of the GLD method (mistuned gains), (**b**) last two iterations (well-tuned gains) for Z_{GLD1} (α = 1.0, β = 0.0), Z_{GLD2} (α = 0.9, β = 0.1), Z_{GLD3} (α = 0.8, β = 0.2).

3.2. *Experiments in Flight Conditions*

The GLD method was verified in real-world experiments on the same UAV and for the same parameter configuration as in simulation tests. The method was tested with great attention paid to the efficiency of obtaining altitude controller gains and the tracking quality. From variety of conducted experiments, the author decided to present and discussed, a few, which are the most representative. Supplementary materials (video and listings) are available at: http://www.uav.put.poznan.pl.

3.2.1. Uncertainty of Altitude Measurements. Change of Sensory Information Sources

The aim of the experiment was to verify how imprecise and non-stationary the altitude measurements of the UAV flight are in the building, based on its basic on-board avionics only. The motion capture system was used as a ground truth. The results are shown in Figure 10. The task for the UAV was to fly to a fixed altitude of 1 m and hover in the air.

As the average error from registered trials is only 0.80%, the range of actual/instantaneous values ranges from 0.85 m to 1.14 m and increases with the passage of time. Such a dispersion of measurements is a problem and major difficulty in the proposed machine learning procedure used in real-world conditions for altitude controller tuning. Therefore, the motion capture system was used for further estimation of the UAV flight altitude. This eliminates the measurement error as a source of additional errors during the J calculation.

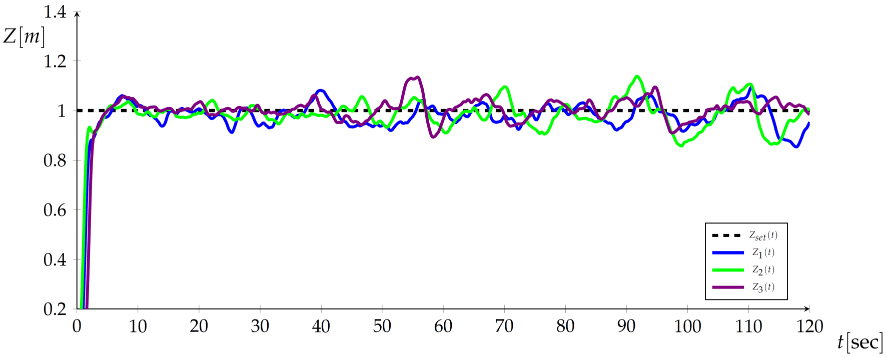

Figure 10. Tracking of the reference altitude Z_{set} by the Bebop 2 UAV in three trials (Z_1–Z_3).

3.2.2. Comparison of the Tuning Effectiveness: FIB vs. GLD Method Used in Real-World Conditions

In the Figure 11, the altitude controller gains during auto-tuning procedure using GLD and FIB methods, are presented. Based on simulation results it was decided to terminate both methods after 48 iterations. Final and average values of J (see Figure 12), are lower for the GLD method: 43.97% and

3.39%, for which the tracking quality is better (Figure 13), e.g., lower overshoots were recorded during the tuning time.

Furthermore, it is worth mentioning that both methods here shown convergence in the vicinity of the two local minima of the $J = f(k_P, k_D)$ function, which were estimated based on the preliminary simulation experiments.

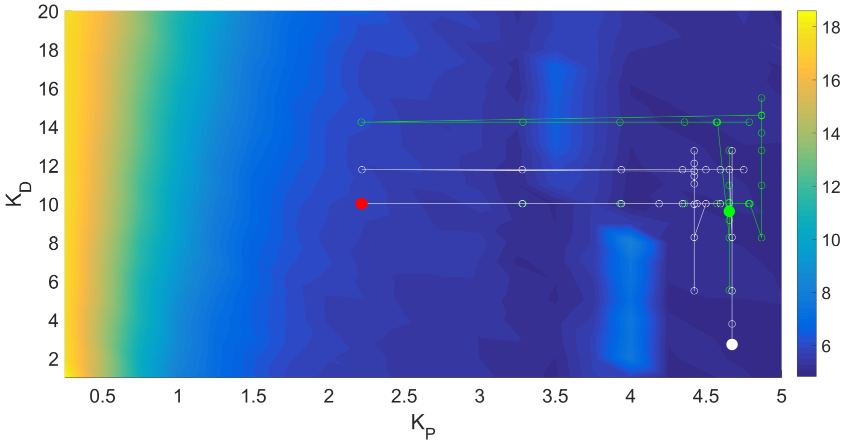

Figure 11. The altitude controller gains and $J = f(k_P, k_D)$ values during auto-tuning process using GLD (white color) and FIB (green) methods; k_{Dinit} = 10 (marked in red).

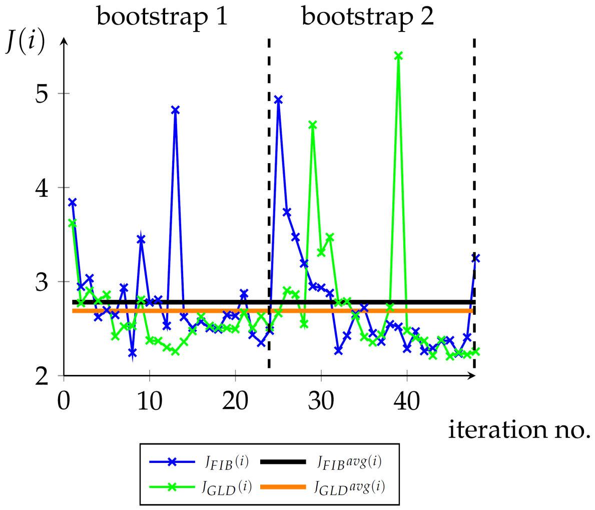

Figure 12. Time courses for the GLD and FIB tuning process in real-world conditions.

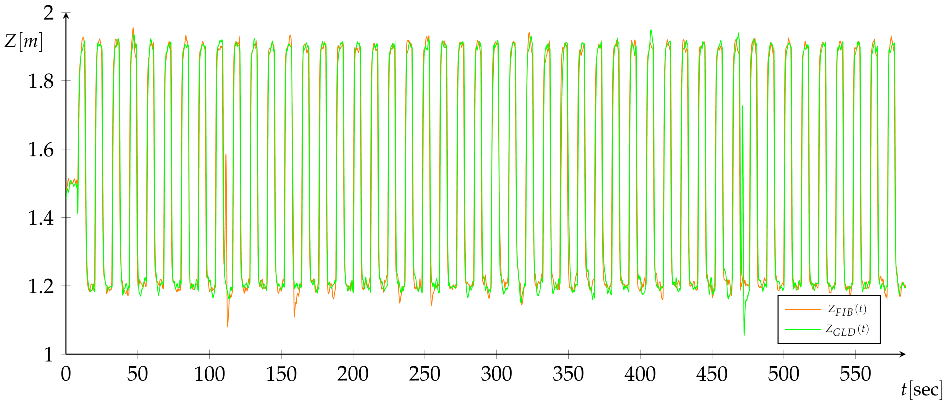

Figure 13. Values of $J(i)$ in consecutive steps (i) of GLD and FIB tuning.

3.2.3. Analysis of the Impact of Environmental Disturbances (Wind Gusts) on the Auto-Tuning Procedure

Usually in scientific world literature presented results of research on UAV flights under wind gust conditions concern the case when the air stream is directed towards the UAV frontally. In real flight conditions, this direction is usually random and variable in time. Thus, it was decided to verify the effectiveness of the GLD auto-tuning method with a low-pass filtration, during the UAV flight, in the stream of the air generated from the rotating fan (1.2 m high), at a distance of 1.8 m, behind the UAV on the left, as in Figure 14.

In the auto-tuning procedure, the disturbances were introduced twice (see Figure 15). In the first phase, the maximum air flow speed was 2.7 m/s, in the second—3.7 m/s. It is a severe disturbance referring to the ratio of physical dimensions to the small weight of the UAV. A complete 56-iterative tuning cycle was conducted. The results are summarized in Figure 16 and Table A1 (see Appendix A), and compared with the results of the auto-tuning from the previous Subsection. Very similar, promising final values of the J performance index were obtained—even only slightly smaller for the case of impact of a wind gust during the GLD procedure.

Determinism of the method is illustrated by the results of 10 first iterations in both trials and iterations no. 29-38, where for different values of J, the calculated k_P gain values are identical. Similar behavior can be observed in the presence of wind gusts (iterations no. 15–20, and 43–48). In the future research, it is worth considering an approach in which two or several UAV units (agents) could be used to parallel measurements and averaging computations during the auto-tuning procedure, resulting in better tuning precision.

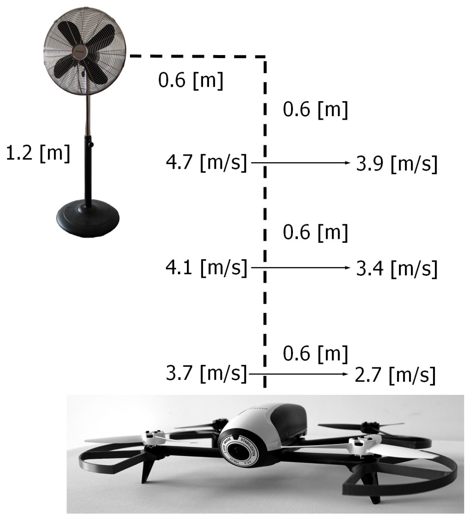

Figure 14. Test bed for research on wind gusts impact on the GLD method.

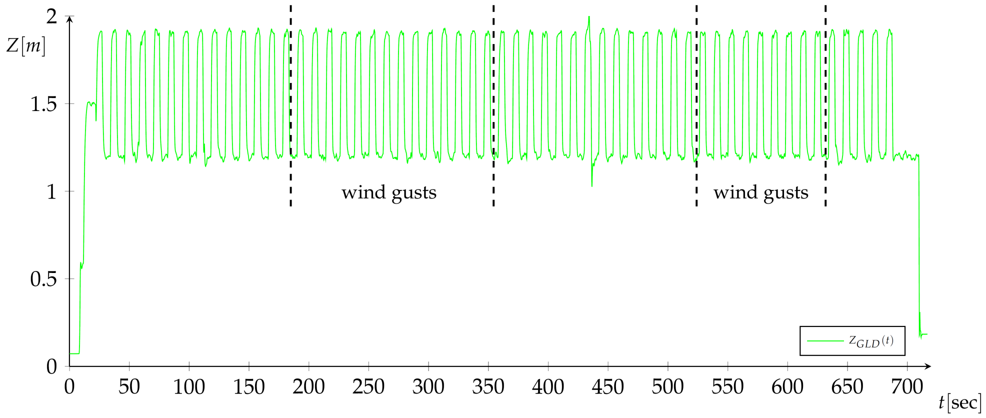

Figure 15. Time course for the GLD tuning process in the presence of wind gusts.

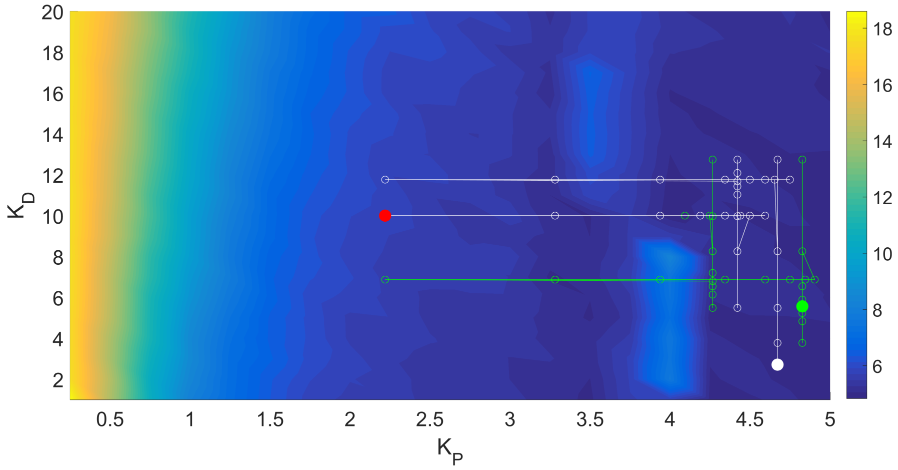

Figure 16. The altitude controller gains and $J = f(k_P, k_D)$ values during auto-tuning process using GLD method for nominal case (white color) and at the presence of wind gusts (green); k_{Dinit} = 10 (marked in red).

3.2.4. Flights and Auto-Tuning in UAV Mass Change Conditions

The last interesting aspect of the conducted research was to provide knowledge about the quality of the obtained gains in the context of transport tasks and the use of the GLD method to tune the gains of the altitude controller after changing the total takeoff weight of the UAV. A series of experimental studies was conducted for this purpose.

In the simulation experiments, the efficiency of tuning of the UAV altitude controller using the GLD method was verified in conditions of lifting of the additional payload (jar on the gripper and tool accessories attached to the UAV). The gains of the other controllers (for X and Y axes, and for *yaw* angle control), were adopted from Table 4. In subsequent simulations, the values α and β of the J function were changed. The results are presented in Table 6, and the search process for the controller's gains is illustrated in the attached video material. Based on the obtained results, it can be noticed that in the case of both payloads tested, the values of k_P were smaller than in the nominal case (flight without payload), and k_D values were larger. Increased starting mass of the UAV forces the use of more thrust to lift the UAV and at the same time—to provide its effective balance, so as not to cause any overshoots (exceeding the given/reference altitude). In the qualitative evaluation of the results of the auto-tuning procedure, the obtained controller using a similar gain value of the proportional part, compensates with a larger gain of k_D the nervous behavior of the UAV (which for particular J function tries to match the dynamics to higher UAV inertia).

Table 6. Results of simulation experiments—flying with: gripper & jar (GRIP), and tool accessories (TOOL); for 2 bootstrap cycles, 56 iterations of the GLD method with low-pass filtration and ϵ = 0.05.

	GRIP	TOOL	GRIP	TOOL	GRIP	TOOL
α	1.0	1.0	0.9	0.9	0.8	0.8
β	0.0	0.0	0.1	0.1	0.2	0.2
k_{Dinit}	10.0	10.0	10.0	10.0	10.0	10.0
k_P range	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]
k_D range	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]
Best k_P and k_D values	2.30/15.11	2.39/18.94	2.20/17.88	1.08/18.94	0.82/15.77	0.67/13.40
J_1 (after the 1st bootstrap)	7.5143	7.9741	7.6119	7.6654	7.5773	7.5454
J_{end} (after the tuning proc.)	7.5150	7.8334	7.5473	7.6591	7.7385	7.4198
J_{avg} (average for tuning proc.)	7.4306	7.9191	7.7877	7.5485	7.5906	7.5454

In the first real-world experiment (Figure 17), the task of the UAV was to start the autonomous flight from a platform with a plastic bottle attached; then, to fly to the point where the GLD auto-tuning procedure begins; finally, to perform 56 iterations of the algorithm in the presence of wind gusts. The drone, using its on-board avionics (including the optical-flow and ultrasonic sensors) moved vertically after stabilizing the position of the gripper, since it recognized its position as altitude equal to 0, and in effect moved upwards, which created a danger. The same behavior was observed in the second experiment, where the UAV task was to compensate its position in the X, Y, and Z axis (refer to supplementary video material). A decision was made to change the type and manner of payload attachment as shown in Figure 18, which played its role, both in the GLD auto-tuning experiments with additional mass, as well as in transportation tasks at designated nominal gains (see Figure 19). In every conducted trial (Table 7), for subsequent J functions, similar behavior was observed as in the case of simulation tests. For example, let us consider the results obtained for $\alpha = 1.0$ (Figure 20). It can be noticed that the time courses with large overshoots (when the controller forces too hard the UAV, wanting to overcome its increased inertia), result in an increase in the value of J and are effectively rejected in the procedure of seeking the smallest value of this performance index. In addition, by analyzing the subsequent values of this index (Figure 21), it can be seen that the selection of the gain value k_D directly implies the UAV vertical flight dynamics profile. This is particularly seen in the first bootstrap (marked in Figure 20).

Auto-tuning in UAV mass change conditions will be the subject of a separate article, while it is worth stressing that the second problem encountered—mentioned at the beginning of the article—i.e., lack of stability criterion based on which it would be possible to estimate safe gains ranges of k_P and k_D for their exploration in the GLD method. Despite its high efficiency and safe operation in tuning of controllers of UAVs with nominal mass or with low extra mass, in case of large payloads (see Figure 22 for the case of 282 g) one can find examples of unstable flights. Then it is strongly recommended to use preliminary simulation tests based on the model. The introduction of the stability criterion into the proposed GLD method is in the area of further research interest of the author [50].

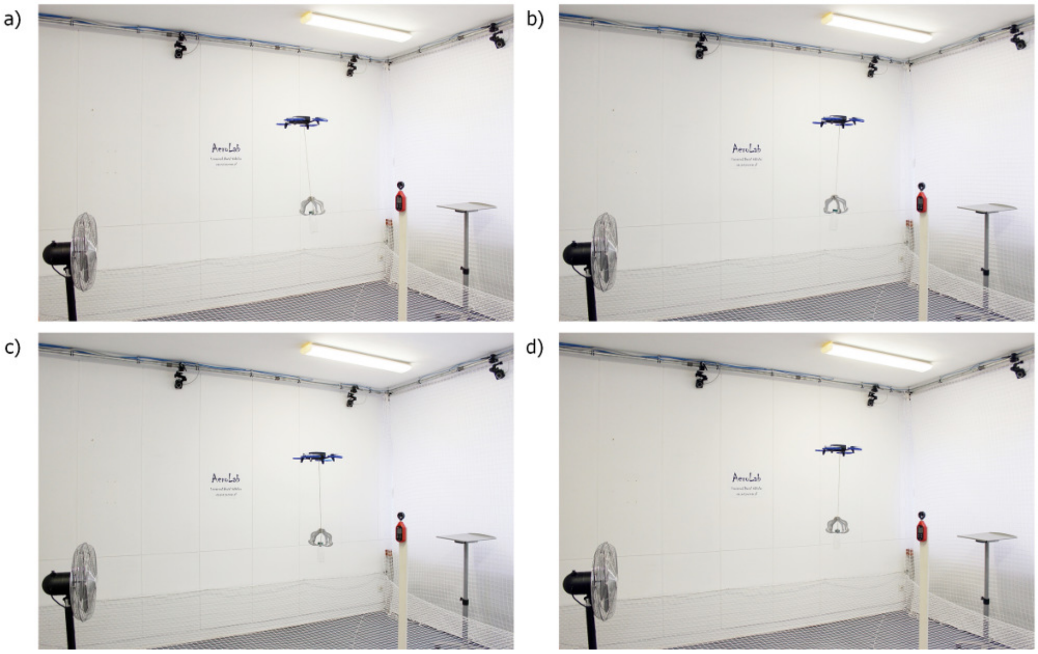

Figure 17. Snapshots from one of the initial research experiments with the auto-tuning of the altitude controller during the flight in the presence of wind gusts and with the mass attached to the UAV on a flexible joint.

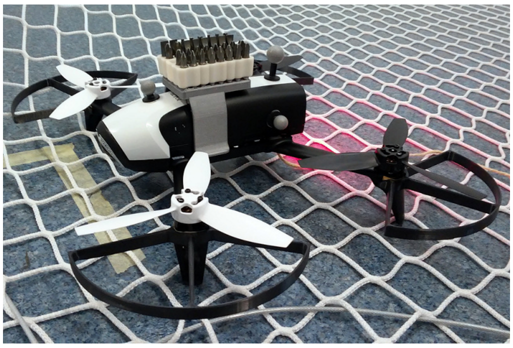

Figure 18. The Bebop 2 with additional payload used for in-flight auto-tuning experiments.

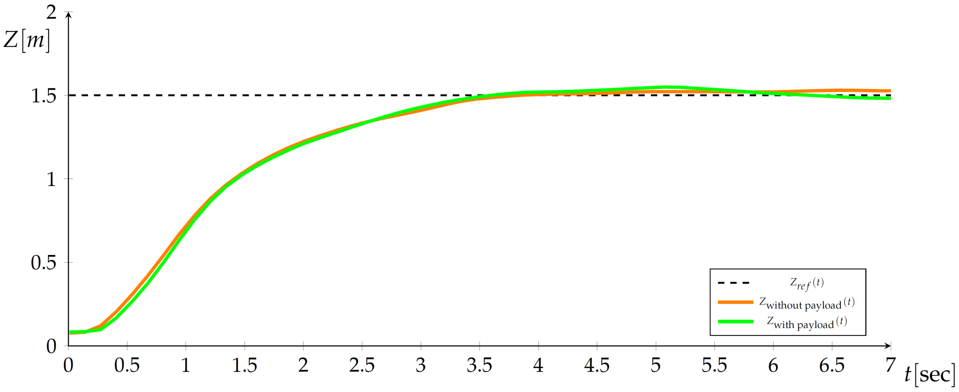

Figure 19. Step responses for the Bebop 2 UAV tuned with GLD method—variants: with and without addition mass (225 g tool accessories).

Table 7. Results of real-world experiments—flying with the payload (tool accessories); for 2 bootstrap cycles, 56 iterations of the GLD method with low-pass filtration and ϵ = 0.05.

	Exp.1	**Exp.2**	**Exp.3**
α	1.0	0.9	0.8
β	0.0	0.1	0.2
k_{Dinit}	10.0	10.0	10.0
k_P range	[0.5,5.0]	[0.5,5.0]	[0.5,5.0]
k_D range	[1.0,20.0]	[1.0,20.0]	[1.0,20.0]
Best k_P and k_D values	3.92/7.85	4.02/9.57	3.20/11.68
J_1 (after the 1st bootstrap)	2.5070	2.4992	2.6096
J_{end} (after the tuning proc.)	1.7583	2.5779	2.5255
J_{avg} (average for tuning proc.)	2.8091	2.4566	2.6553

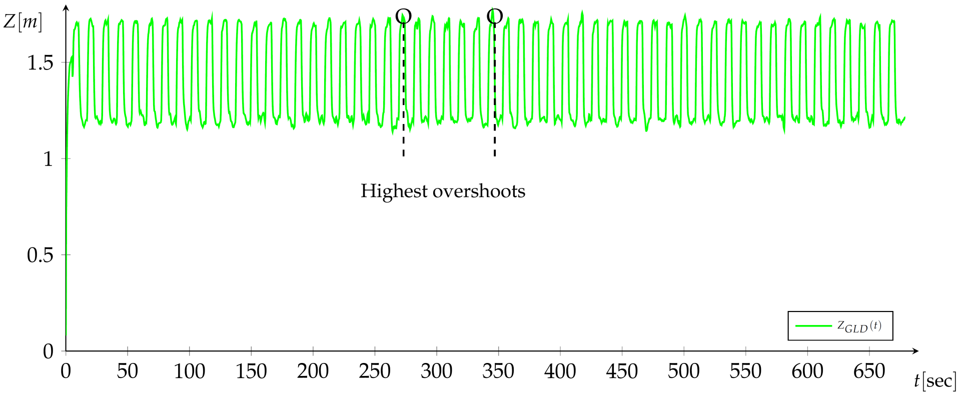

Figure 20. Time course from the tuning experiment via GLD method—variant: tuning of the altitude controller during the UAV flight with an additional (heavy) mass (225 g); the experiment interrupted due to the loss of stability.

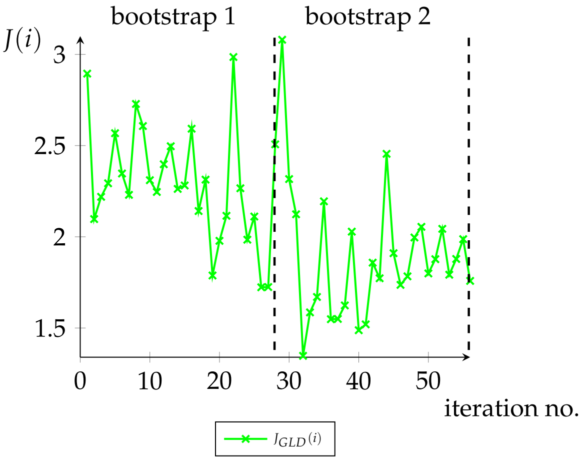

Figure 21. Values of $J(i)$ in consecutive steps (i) of the GLD method (Exp. no. 1)—flying with the payload.

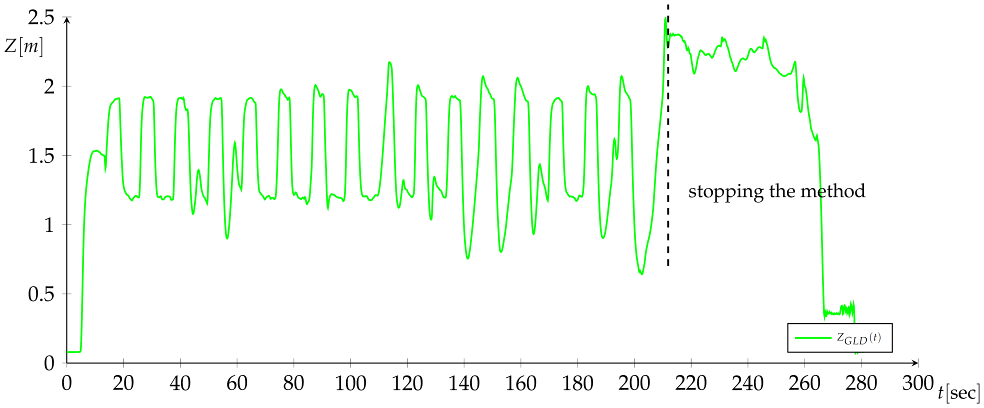

Figure 22. Time course from the real-world experiment (Exp. no. 1): tuning of the altitude controller during the UAV flight with an additional (heavy) mass (225 g).

4. Conclusions and Further Work

In the paper, a new and efficient real-time auto-tuning method for fixed-parameters controllers based on the modified golden-search (zero-order) optimization algorithm and bootstrapping technique, has been presented. The method ensures fast, iterative behavior, and as a result—returns in the worst case the locally best gains of controller, in the best case—globally optimal. The GLD method is fully automated, and uses a low-pass filtration while working in a stochastic environment. It is a model-free approach, but as it has been articulated in the paper, it is good to combine its advantages with initial model-based prototyping, since the method does not use any stability criterion. The author is interested and looking for the mathematical solutions, i.e., in the area of stochastic analysis and probability, which can be easily adapted into the proposed GLD procedure—without increase of its computational complexity. It will be useful in a context of solving mentioned transportation tasks and problems (especially when flying near to the lifting capacity of the UAV).

Supplementary Materials: Recorded videos and data from ROS bags are available online at http://uav.put.poznan.pl.

Funding: This research was financially supported as a statutory work of Poznan University of Technology (04/45/DSPB/0196).

Acknowledgments: The author would like to thank Bartłomiej Kulecki for his help with software configuration.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BF	Body Frame
CCW	Counter-Clockwise
CW	Clockwise
DOF	Degrees of Freedom
EF	Earth Frame
FIB	Fibonacci-search Method
GLD	Golden-search Method
GNSS	Global navigation satellite system
GPS	Global Positioning System
MBZIRC	Mohamed Bin Zayed International Robotics Challenge
NED	North-East-Down
PD	Proportional-Derivative Controller
PID	Proportional-Integral-Derivative Controller
REM	Region Elimination Method
ROS	Robot Operating System
UAV	Unmanned Aerial Vehicle

Appendix A

Table A1. Comparison of the results of auto-tuning of the UAV's altitude controller using the GLD method—variants: nominal and at the presence of wind gusts.

	Nominal	**Disturbed**	**Nominal**	**Disturbed**	**Nominal**	**Disturbed**
No. of Iter.	k_P	k_P	k_D	k_D	J	J
1	2.2190	2.2190	10.0000	10.0000	3.6232	3.9781
2	3.2810	3.2810	10.0000	10.0000	2.7718	2.9733
3	3.2813	3.2813	10.0000	10.0000	2.9025	3.0650
4	3.9377	3.9377	10.0000	10.0000	2.7974	2.9231
5	3.9379	3.9379	10.0000	10.0000	2.8608	2.9099
6	4.3435	4.3435	10.0000	10.0000	2.4185	2.6717
7	4.3436	4.3436	10.0000	10.0000	2.5228	2.7708
8	4.5943	4.5943	10.0000	10.0000	2.5281	3.2745
9	4.1886	4.1886	10.0000	10.0000	2.8080	2.6899
10	4.3435	4.3435	10.0000	10.0000	2.3742	2.8161
11	4.3436	4.0928	10.0000	10.0000	2.3683	3.0077
12	4.4393	4.1886	10.0000	10.0000	2.3004	2.6560
13	4.4393	4.1886	10.0000	10.0000	2.2584	2.7008
14	4.4985	4.2478	10.0000	10.0000	2.3600	2.4583
15	4.4210	4.2661	8.2580	8.2580	2.4742	2.4305
16	4.4210	4.2661	12.7420	12.7420	2.6279	2.5428
17	4.4210	4.2661	5.4854	5.4854	2.5291	2.8694
18	4.4210	4.2661	8.2566	8.2566	2.5097	2.3611
19	4.4210	4.2661	8.2574	8.2574	2.5057	2.3465
20	4.4210	4.2661	9.9700	9.9700	2.4944	2.7500
21	4.4210	4.2661	9.9705	7.1985	2.6712	2.4564
22	4.4210	4.2661	11.0289	8.2569	2.5066	2.5012
23	4.4210	4.2661	11.0292	6.5441	2.6319	2.5666
24	4.4210	4.2661	11.6833	7.1982	2.5153	2.7464
25	4.4210	4.2661	11.6835	6.1397	2.6646	2.6194
26	4.4210	4.2661	12.0877	6.5439	2.9049	2.4165
27	4.4210	4.2661	11.4336	6.5441	2.8642	2.4360
28	4.4210	4.2661	11.6834	6.7939	2.5492	2.3116
29	2.2190	2.2190	11.7607	6.8711	4.6672	4.4838
30	3.2810	3.2810	11.7607	6.8711	3.3072	2.7008

Table A1. *Cont.*

	Nominal	Disturbed	Nominal	Disturbed	Nominal	Disturbed
No. of Iter.	k_P	k_P	k_D	k_D	J	J
31	3.2813	3.2813	11.7607	6.8711	3.4729	2.9624
32	3.9377	3.9377	11.7607	6.8711	2.7758	2.4287
33	3.9379	3.9379	11.7607	6.8711	2.7904	2.5296
34	4.3435	4.3435	11.7607	6.8711	2.6357	2.3066
35	4.3436	4.3436	11.7607	6.8711	2.4105	4.4358
36	4.5943	4.5943	11.7607	6.8711	2.3535	2.6047
37	4.5943	4.5943	11.7607	6.8711	2.4362	2.5906
38	4.7493	4.7493	11.7607	6.8711	2.7252	2.4460
39	4.4986	4.7493	11.7607	6.8711	5.4026	2.5969
40	4.5943	4.8450	11.7607	6.8711	2.4769	2.3667
41	4.5943	4.8451	11.7607	6.8711	2.4032	2.3662
42	4.6535	4.9042	11.7607	6.8711	2.3701	2.9292
43	4.6718	4.8268	8.2580	8.2580	2.2139	2.3028
44	4.6718	4.8268	12.7420	12.7420	2.3797	2.5986
45	4.6718	4.8268	5.4854	5.4854	2.2050	2.1790
46	4.6718	4.8268	8.2566	8.2566	2.2468	2.3559
47	4.6718	4.8268	3.7720	3.7720	2.2244	2.4715
48	4.6718	4.8268	5.4846	5.4846	2.2563	2.1976
49	...	4.8268	...	5.4851	...	2.1532
50	...	4.8268	...	6.5435	...	2.5722
51	...	4.8268	...	4.8307	...	2.3696
52	...	4.8268	...	5.4848	...	2.3328
53	...	4.8268	...	5.4850	...	2.2445
54	...	4.8268	...	5.8892	...	2.7051
55	...	4.8268	...	5.2350	...	2.2805
56	...	4.8268	...	5.4848	...	2.2393

References

1. Valavanis, K.; Vachtsevanos, G.J. (Eds.) *Handbook of Unmanned Aerial Vehicles*; Springer: Dordrecht, The Netherlands, 2015.
2. Jordan, S.; Moore, J.; Hovet, S.; Box, J.; Perry, J.; Kirsche, K.; Lewis, D.; Tsz Ho Tse, Z. State-of-the-art technologies for UAV inspections. *IET Radar Sonar Navig.* **2018**, *12*, 151–164. [CrossRef]
3. Hinas, A.; Roberts, J.M.; Gonzalez, F. Vision-Based Target Finding and Inspection of a Ground Target Using a Multirotor UAV System. *Sensors* **2017**, *17*, 2929. [CrossRef] [PubMed]
4. Sandino, J.; Gonzalez, F.; Mengersen, K.; Gaston, K.J. UAVs and Machine Learning Revolutionising Invasive Grass and Vegetation Surveys in Remote Arid Lands. *Sensors* **2018**, *18*, 605. [CrossRef] [PubMed]
5. Dziuban, P.J.; Wojnar, A.; Zolich, A.; Cisek, K.; Szumiński, W. Solid State Sensors—Practical Implementation in Unmanned Aerial Vehicles (UAVs). *Procedia Eng.* **2012**, *47*, 1386–1389. [CrossRef]
6. Gośliński, J.; Giernacki, W.; Królikowski, A. A nonlinear Filter for Efficient Attitude Estimation of Unmanned Aerial Vehicle (UAV). *J. Intell. Robot. Syst.* **2018**. [CrossRef]
7. Urbański, K. Control of the Quadcopter Position Using Visual Feedback. In Proceedings of the 18th International Conference on Mechatronics (Mechatronika), Brno, Czech Republic, 5–7 December 2018; pp. 1–5.
8. Ebeid, E.; Skriver, M.; Terkildsen, K.H.; Jensen, K.; Schultz, U.P. A survey of Open-Source UAV flight controllers and flight simulators. *Microprocess. Microsyst.* **2018**, *61*, 11–20. [CrossRef]
9. Lozano, R. (Ed.) *Unmanned Aerial Vehicles: Embedded Control*; John Wiley & Sons: New York, NY, USA, 2010.
10. Santoso, F.; Garratt, M.A.; Anavatti, S.G. State-of-the-Art Intelligent Flight Control Systems in Unmanned Aerial Vehicles. *IEEE Trans. Autom. Sci. Eng.* **2018**, *15*, 613–627. [CrossRef]
11. Mahony, R.; Kumar, V.; Corke, P. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robot. Autom. Mag.* **2012**, *19*, 20–32. [CrossRef]

12. Ren, B.; Ge, S.; Chen, C.; Fua, C.; Lee, T. *Modeling, Control and Coordination of Helicopter Systems*; Springer: New York, NY, USA, 2012. [CrossRef]
13. Pounds, P.; Bersak, D.R.; Dollar, A.M. Stability of small-scale UAV helicopters and quadrotors with added payload mass under PID control. *Auton. Robots* **2012**, *33*, 129–142. [CrossRef]
14. Li, J.; Li, Y. Dynamic Analysis and PID Control for a Quadrotor. In Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation (ICMA), Beijing, China, 7–10 August 2011; pp. 573–578. [CrossRef]
15. Espinoza, T.; Dzul, A.; Llama, M. Linear and nonlinear controllers applied to fixed-wing UAV. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 1–10. [CrossRef]
16. Lee, K.U.; Kim, H.S.; Park, J.-B.; Choi, Y.-H. Hovering Control of a Quadrotor. In Proceedings of the 2012 12th International Conference on Control, Automation and Systems (ICCAS), JeJu Island, South Korea, 17–21 October 2012; pp. 162–167.
17. Pounds, P.E.; Dollar, A.M. Aerial Grasping from a Helicopter UAV Platform, Experimental Robotics. *Springer Tracts Adv. Robot.* **2014**, *79*, 269–283. [CrossRef]
18. Kohout, P. A System for Autonomous Grasping and Carrying of Objects by a Pair of Helicopters, Master's Thesis, Czech Technical University in Prague, Prague, Czech Republic, 2017.
19. Spica, R.; Franchi, A.; Oriolo, G.; Bülthoff, H.H.; Giordano, P.R. Aerial grasping of a moving target with a quadrotor UAV. In the Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 October 2012; pp. 4985–4992. [CrossRef]
20. Yang, F.; Xue, X.; Cai, C.; Sun, Z.; Zhou, Q. Numerical Simulation and Analysis on Spray Drift Movement of Multirotor Plant Protection Unmanned Aerial Vehicle. *Energies* **2018**, *11*, 2399. [CrossRef]
21. Rao Mogili, U.M.; Deepak, B.B.V.L. Review on Application of Drone Systems in Precision Agriculture. *Procedia Comput. Sci.* **2018**, *133*, 502–509. [CrossRef]
22. Imdoukh, A.; Shaker, A.; Al-Toukhy, A.; Kablaoui, D., El-Abd, M. Semi-autonomous indoor firefighting UAV. In Proceedings of the 2017 18th International Conference on Advanced Robotics (ICAR), Hong Kong, China, 10–12 July 2017; pp. 310-315. [CrossRef]
23. Duan, H.; Li, P. *Bio-inspired Computation in Unmanned Aerial Vehicles*; Springer: Berlin, Germany, 2014. [CrossRef]
24. Giernacki, W.; Espinoza Fraire, T.; Kozierski, P. Cuttlesh Optimization Algorithm in Autotuning of Altitude Controller of Unmanned Aerial Vehicle (UAV). In Proceedings of the Third Iberian Robotics Conference (ROBOT 2017), Seville, Spain, 22–24 November 2017; pp. 841–852. [CrossRef]
25. Chong, E.K.P.; Zak, S.H. *An Introduction to Optimization*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2001.
26. Giernacki, W.; Horla, D.; Báča, T.; Saska, M. Real-time model-free optimal autotuning method for unmanned aerial vehicle controllers based on Fibonacci-search algorithm. *Sensors* **2018**, *19*, 312. [CrossRef]
27. Spurný, V.; Báča, T.; Saska, M.; Pěnička, R.; Krajník, T.; Loianno, G.; Thomas, J.; Thakur, D.; Kumar, V. Cooperative Autonomous Search, Grasping and Delivering in Treasure Hunt Scenario by a Team of UAVs. *J. Field Robot.* **2018**, 1–24. [CrossRef]
28. Automatic Tuning with AUTOTUNE. Ardupilot.org. Available online: http://ardupilot.org/plane/docs/automatic-tuning-with-autotune.html (accessed on 12 November 2018).
29. Rodriguez-Ramos, A.; Sampedro, C.; Bavle, H.; de la Puente, P.; Campoy, P. A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform. *J. Intell. Robot. Syst.* **2018**, 1–16. [CrossRef]
30. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement Learning for UAV Attitude Control. Available online: https://arxiv.org/abs/1804.04154 (accessed on 12 November 2018).
31. Panda, R.C. *Introduction to PID Controllers—Theory, Tuning and Application to Frontier Areas*; In-Tech: Rijeka, Croatia, 2012. [CrossRef]
32. Rios, L.; Sahinidis, N. Derivative-free optimization: A review of algorithms and comparison of software implementations. *J. Glob. Optim.* **2013**, *56*, 1247–1293. [CrossRef]
33. Spall, J.C. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*; Wiley: New York, NY, USA, 2003. [CrossRef]
34. Hjalmarsson, H.; Gevers, M.; Gunnarsson, S.; Lequin, O. Iterative feedback tuning: Theory and applications. *IEEE Control Syst. Mag.* **1998**, *18*, 26–41. [CrossRef]

35. Reza-Alikhani, H. PID type iterative learning control with optimal variable coefficients. In the Proceedings of the 2010 5th IEEE International Conference Intelligent Systems, London, UK, 7–9 July 2010; pp. 1–6. [CrossRef]
36. Ghadimi, S.; Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.* **2013**, *23*, 2341–2368. [CrossRef]
37. Kiefer, J. Sequential minimax search for a maximum. *Proc. Am. Math. Soc.* **1953**, *4*, 502–506. [CrossRef]
38. Brasch, T.; Byström, J.; Lystad, L.P. *Optimal Control and the Fibonacci Sequence*; Statistics Norway, Research Department: Oslo, Norway, 2012; pp. 1–33. Available online: https://www.ssb.no/a/publikasjoner/pdf/DP/dp674.pdf (accessed on 28 December 2018).
39. Theys, B.; Dimitriadis, G.; Hendrick, P.; De Schutter, J. Influence of propeller configuration on propulsion system efficiency of multi-rotor Unmanned Aerial Vehicles. In Proceedings of the 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, TX, USA, 7–10 June 2016; pp. 195–201. [CrossRef]
40. Xia, D.; Cheng, L.; Yao, Y. A Robust Inner and Outer Loop Control Method for Trajectory Tracking of a Quadrotor. *Sensors* **2017**, *17*, 2147. [CrossRef] [PubMed]
41. Wang, Y.; Gao, F.; Doyle, F. Survey on iterative learning control, repetitive control, and run-to-run control. *J. Process Control* **2009**, *10*, 1589–1600. [CrossRef]
42. Multicopter PID Tuning Guide. Available online: https://docs.px4.io/en/config_mc/pid_tuning_guide_multicopter.html (accessed on 19 November 2018).
43. How to Tune PID I-Term on a Quadcopter. Available online: https://quadmeup.com/how-to-tune-pid-i-term-on-a-quadcopter/ (accessed on 18 November 2018).
44. Quadcopter PID Explained. Available online: https://oscarliang.com/quadcopter-pid-explained-tuning/ (accessed on 18 November 2018).
45. Arimoto, S.; Kawamura, S.; Miyazaki, F. Bettering operation of robots by learning. *J. Robot. Syst.* **1984**, *1*, 123–140. [CrossRef]
46. Parrot BEBOP 2. The Lightweight, Compact HD Video Drone. Available online: https://www.parrot.com/us/drones/parrot-bebop-2 (accessed on 26 November 2018).
47. AeroLab Poznan University of Technology Drone Laboratory Webpage. Available online: http://uav.put.poznan.pl/AeroLab (accessed on 2 December 2018).
48. What is Sphinx. Available online: https://developer.parrot.com/docs/sphinx/whatissphinx.html (accessed on 18 November 2018).
49. bebop_autonomy—ROS Driver for Parrot Bebop Drone (quadrocopter) 1.0 & 2.0. Available online: https://bebop-autonomy.readthedocs.io/en/latest/ (accessed on 18 November 2018).
50. Giernacki, W.; Horla, D.; Sadalla, T.; Espinoza Fraire, T. Optimal Tuning of Non-integer Order Controllers for Rotational Speed Control of UAV's Propulsion Unit Based on an Iterative Batch Method. *J. Control Eng. Appl. Inform.* **2018**, *24*, 22–31.