2024 21st International Conference on Ubiquitous Robots (UR)
June 24 - 27, 2024, New York University, USA

Tuning PID Controller for Quadrotor Using Particle Swarm Optimization

Eric X. Rodriguez[1] and Qi Lu[2]

Abstract— **Energy expenditure for quadrotor control has a likelihood of being costly given parameter-dependent controllers that are less than optimal. The cost can grow proportionally when applied to multiple quadrotors for tracking and collaborative navigation tasks. This research aims to establish a basic approach to tuning PID (Proportional-Integral-Derivative) parameters for a simulated quadrotor drone. A PID controller for autonomy provides a straightforward method for correcting robotic movement based on its current state. However, applying a PID system to a flight controller poses challenges with an inherently under-actuated system, which includes the likelihood of large overshoots and lengthy adjustment times. To address this, we utilize PSO (Particle Swarm Optimization) for optimizing PID parameters in a simulated quadrotor. The PSO is employed to find optimal PID values for thrust, yaw, and translational movement on x- and y-positions by identifying converging values across randomly created particles. We conducted a set of experiments and compared it to the default PID controller. The experiments demonstrate converging properties for particles that achieve minimal fitness scores, particularly in reducing overshoot. The results indicate that the optimized PID controller outperforms the default PID controller without optimization. Using optimized PID controllers can decrease the amount of positional error during flight and when adjusting position with collaborative navigation and collision avoidance algorithms.**

The PID controller is a simple and effective method employed for system monitoring and control [1]. However, a notable drawback of the PID control scheme is utilizing disproportional gains that can lead to system failure. For example, the goal of a PID controller is to minimize overshooting a desired setpoint while reducing adjustment time. A common approach to addressing this is manually tuning each PID parameter for every assigned response system. Unfortunately, this method can be time-consuming and may still leave the system vulnerable to inadequate flight responses. To overcome this, we propose the use of Particle Swarm Optimization (PSO) to optimally tune PID parameters for a simulated quadrotor.

PSO demonstrates rapid convergence speed in obtaining optimal parameters, requiring adjustment of only a few setting parameters for convergence, and exhibits high calculation efficiency [2]. Moreover, it often achieves high-quality solutions in a shorter time compared to other stochastic methods [3]. This article aims to identify optimal PID parameters for a simulated DJI Mavic2Pro in the Webots robot simulator using PSO. We evaluate the efficiency of the tuned PID against the PID with default parameters, analyzing performance by comparing flight trajectories with a reference trajectory over predetermined waypoints. The results indicate that the PID parameters tuned by PSO outperform those with default settings. With our findings, we intend to apply an equivalent PSO tuning implementation for a similar simulated quadrotor and transfer found PID gains to a physical quadrotor controller. We believe that validating tuning parameters for a physical quadrotor can be replicated in multiple physical drones. To caveat, we hope this will assist with future efforts in exploring collaborative navigation. This paper is organized with Section I exploring related works. Section II will provide a detailed background in the PID control scheme and PSO. Section III will describe the experimental setup with results depicted in Section IV. Finally, Section V will provide the conclusion.

I. Related work

Numerous studies have explored the stability and control performance of quadcopter systems, often employing the linear PD control method. In a notable example [4], researchers developed a control strategy for quadrotor-type aerial robots using the PD loop to stabilize and control the quadrotor's position during disturbances. Another study [5] focused on elucidating the dynamical model of a quadrotor, implementing the PD control algorithm to regulate orientation and trajectory tracking at slow speeds.

Additionally, an innovative approach was taken in [6], where an extended Kalman filter (EKF)-based smart self-tuning fuzzy-PID (SSTF-PID) controller was introduced for posture control of the quadcopter. Authors in [7] implemented an adaptive pole placement using a self-tuning-PID (ST-PID) controller to stabilize the Euler angle of the quadcopter. To enhance stability and tune PID parameters, researchers applied a gain-scheduled PID controller and a fuzzy logic control (FLC) in [8]. The study also included a comparison between the behaviors of conventional PID and gain-scheduled PID controllers.

The application of Particle Swarm Optimization (PSO) for tuning PID parameters is a widely employed technique, often compared with other optimization methods. In a study by [9], the PSO search algorithm was modified, introducing a new algorithm called modified-PSO for the twin-rotor system. Abdelghany et al. conducted a comprehensive investigation [10], benchmarking quadrotor stabilization using a nonlinear quadrotor model. They compared three control techniques, including a PD controller tuned based on two heuristic algorithms: a genetic algorithm (GA) and PSO.

[1]Eric X. Rodriguez is with the Department of Computer Science, The University of Texas at Rio Grande Valley, Edinburg, USA eric.rodriguez09@utrgv.edu

[2]Qi Lu is with the Department of Computer Science, The University of Texas at Rio Grande Valley, Edinburg, TX, USA qi.lu@utrgv.edu

2024 21st International Conference on Ubiquitous Robots (UR) | 979-8-3503-6107-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/UR61395.2024.10597517

Notably, GA proved to be the better algorithm, with slight contrasts observed in comparison to their modified PSO implementation.

Another article delves into an alternative cost function, assessing adjustment time and overshoots concerning the desired pitch angle within a cascading PI_PID control scheme [2]. The study concluded that the PSO algorithm is suitable for performing preliminary tuning on parameters for their cascade-designed controller. Lastly, the article [11] applies a PD system for trajectory control and a PID system for attitude control. The authors utilized PSO for tuning both systems and compared the performance between traditionally, "hand-tuned" gains with PSO-tuned gains. Their method of comparing parameters involved collecting a mean-square error (MSE) flight performance which resulted in tuned PSO parameters having the least error between performance and reference trajectories. We used a similar approach for testing parameter gains in this research. Specifically, our first experiment aims to collect PSO parameters and verify them with basic flight movements that will be used in our second experiment, a comprehensive error observation over predetermined paths.

This project aims to apply PSO to tune PID gains for simulated quadrotor flight controllers and prove that optimal gain parameters can be found with PSO using combined fitness functions. Many of the articles mentioned above did not make use of combined fitness functions to tune PID parameters, but we included this within our experiments.

II. Background

A. PID Control Scheme

The PID control scheme relies on proportional gains K_p, K_i, and K_d to generate corrective responses for the system. The error state, denoted as $e(t)$, represents the disparity between the measured state and the system's desired setpoint and is comprised of three components: the current instantaneous-time error state, the integrated error state accumulated over a sample of time, and the differentiated error state from a previous instantaneous time to the current. These terms are summed to produce a desired error correction response value, $u(t)$, portrayed in Equation 1 below and explained in [12]. The PID control scheme block diagram is illustrated in Figure 1 and features the output response feedback.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt} \tag{1}$$

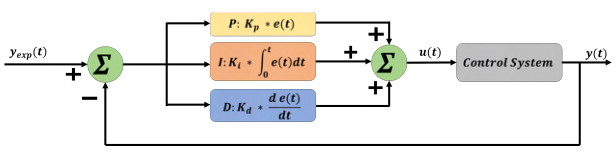


Fig. 1: PID Control Scheme and output response feedback loop.

To address potential drawbacks, we have opted for PSO to fine-tune our parameters. Our goal is to showcase the efficacy of this optimization technique by conducting a performance score comparison. This involves evaluating a simulated quadrotor's flight trajectory against a given reference trajectory within a virtual robot simulator.

B. Particle Swarm Optimization (PSO)

PSO is an efficient optimization technique that stems from the biological phenomena of a flock of birds [13], [14]. The formation represents quality opportunities for the flock to find food, roosting areas, best flight patterns, etc. PSO works in this way and, in a sense, is a stochastic algorithm like gradient descent [15].

PSO uses M-dimensional features with N-particles, where N can be considered the size of the swarm population. Each particle, $i = 1, \ldots, N$ in PSO is correlated to a position $x = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})$ in the search space. Along with the position of the particle, the current state of the particle is influenced by the given velocity-v of the particle ($v_{i,1}$, $v_{i,2}$, …, $v_{i,m}$). The x^{th} – position and the v^{th} – velocity updates are dependent on the best-position ($p_{i,1}$, $p_{i,2}$, …, $p_{i,m}$) found in the search space by the particle so far, including the overall best position ($p_{g,1}$, $p_{g,2}$, …, $p_{g,m}$) across the entire swarm.

The position for N-particles is randomized across a preset range of search values at the initial state of PSO. In our case, we will use a set of parameters from one of the assigned PIDs to control either attitude, altitude, or forward translational movement. For example, Equation 2 exhibits what we would consider a particle position for a single set of parameters. Due to the design of our experiment, we explore PSO convergence with particle parameters by sequentially tuning for Thrust, ψ, and translational (x, y) positions PID parameters.

$$X(t) = \{K_p, K_i, K_d\} \tag{2}$$

Velocity, notated in Equation 3, is applied in the particle's position update term in Equation 4 and is also initially random. The velocity update includes dependencies on fitness scores of the individual particle itself ($p_{i,m}$) and the overall best fitness from the group ($p_{g,m}$). The weighted factor (w) is considered as the inertial weight of the current velocity. In relation, the local-cognition weight (c_1) and the social-cognition weight (c_2) are respective terms for the influence with either cognition computation. The local-cognition weight relates to the individual particle's parameters associated with the best fitness documented through iterations of the PSO algorithm and is also known as p_{best}. The social-cognition weight is related to the best fitness score observed throughout the entirety of the swarm population and is also known as g_{best}. Lastly, randomized values r_1 and r_2 are used to provide additional randomness to the update of a particle's velocity but were not considered during tuning.

$$\begin{aligned} v_{i,m}(t+1) = \; & w * v_{i,m}(t) + c_1 * r_1(p_{i,m} - x_{i,m}(t)) \\ & + c_2 * r_2 * (p_{g,m} - x_{i,d}(t)) \end{aligned} \tag{3}$$

$$x_{i,m}(t+1) = x_{i,m}(t) + v_{i,m}(t+1) \quad (4)$$

After updating particle positions for the next iteration, PSO proceeds to gather the fitness score for the n^{th} particle. The obtained fitness score is then utilized to update p_{best} and g_{best} variables in Equation 3, leading to adjustments in particle parameter values for the upcoming iteration. Throughout the loop, PSO continuously compares individual and group fitness scores until a predefined minimum threshold is met. The optimization loop persists until a desired fitness threshold is attained, and the PID parameters converge within a range noticeably smaller than the standard deviation from the initially randomized particles.

III. Experimental Setup and Design

A. Proposed Experimental PID-Controller

We used two experiments in this research. Our first experiment is to identify optimal parameters for a flight controller and test for performance. Here, we will use log_{10}-SSE to measure for error with tuned parameters across a predetermined waypoint course. The initial experiment seeks to identify optimal PID parameters for controlling the thrust, yaw, and translational(x, y) position of a simulated drone in Webots. We proposed tuning these parameters consecutively starting with thrust, or what we are labeling as $throttlePID$. Figure 2 portrays the state diagram of the quadrotor for this experiment.

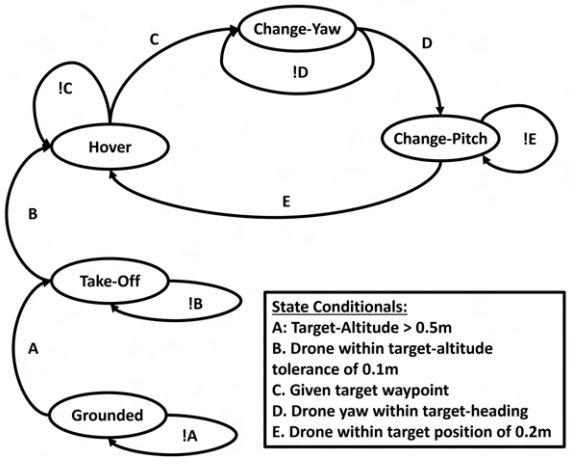


Fig. 2: The quadrotor state diagram visualizes transitions between states and their respective conditionals. This also shows the determined method by which PID systems were tuned sequentially

The drone begins in *grounded* state and has a starting location at a global waypoint [0, 0, 0.1]. Upon receiving a target altitude exceeding 0.5m, the drone transitions into a takeoff state. Once the quadrotor reaches the target altitude, it enters *hover* state. The time interval from *grounded* to *hover* serves as the period during which we observe flight behavior, capturing overshoot (θ) and adjustment time (t_s) for thrust. These measurements are used to collect the fitness score, as employed in [11]. In our context, fitness is computed by summing two log_{10} functions, applied to the ratio of overshoot and expected overshoot (θ_{exp}), along with the ratio of adjustment time and expected adjustment time (t_{exp}).

The same fitness collection method was employed when the drone transitioned into *change-yaw* state, changing the orientation of its initial heading towards a target heading. This ensures the drone progresses forward in the direction of the waypoint. The PID control system responsible for *change-yaw* is labeled as $YawPID$. The quadrotor moves into the *change-pitch* state when both the drone's heading aligns with the target heading.

While in *change-pitch*, the fitness for PSO tuning is represented as the log_{10} of the sum of square errors (SSE) between the performed flight and reference trajectories. This fitness score is calculated by collecting the error between trajectories from the takeoff position to the target point. Tuning parameters for translational (x, y) position control is then performed, labeled as $XposPID$ and $YposPID$, respectively.

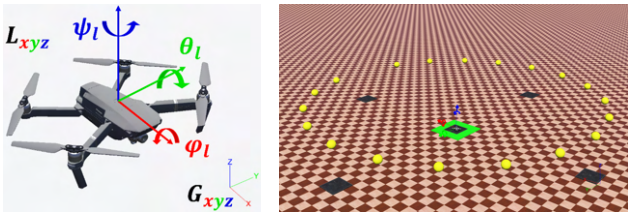
We applied PID control systems for φ and θ without tuning, maintaining parameters at their default values. While these parameters influence motor inputs for the quadrotor, we labeled them as $rollPID$ and $pitchPID$. The second experiment involved comparing the natural logarithm of the sum of square errors (SSE) between flight trajectories of two PID controllers: one with default manually tuned parameters and another with PSO-tuned parameters.

B. Quadcopter Modeling and Simulation

As many of the articles discussed in Section I model their quadcopters in MATLAB, we decided to avoid building similar complex models. Instead, we were more determined to focus on a quadrotor model that was simple to implement and fault-deterrent with our exploration into PSO using Python.

Another reason why Webots was a primary choice for our experiments was the straightforward interface and class hierarchy. Webots offers a Supervisor class that facilitates direct interaction with the simulation, commonly utilized for optimization techniques such as PSO. The quadrotor model and the experimental environment are demonstrated in Figure 3, along with local and global reference frames. For this expermient we utilzied a ENU global refernece and mapped an XYZ local system for the quadrotor.

For this experiment, we assigned 6 PID control schemes for attitude control and positional control. For attitude control, we assigned PID systems to control roll (φ), pitch (θ), and yaw (ψ). For positional control, we assigned one set of PID parameters to control thrust and two sets of PID parameters to control the x- and y-position of the quadrotor. In total, we used 18 different parameters to control our quadrotor system. We adopted this approach from [11] which utilized classical PD and PID approaches for trajectory, attitude, and altitude control for a cross-shaped quadrotor.



(a) (b)

Fig. 3: Illustrations of a simulated DJI Mavic2Pro quadrotor and the experimental environment in Webots. (a) The DJI Mavic2Pro quadrotor; (b) The experimental environment for the quadrotor and a circular trajectory. The trajectory is illustrated by a set of balls in yellow. The quadrotor is located on a squared plane in the center.

The advantage of using a simulated quadrotor provides us the opportunity to explore a performance mapping between PID responses and motor inputs. Using this inertial sensor provides a representation of the quadrotor's attitude in both local and global reference frames. This also provided a scheme to relate the quadrotor's position using a rotational matrix influenced by the local yaw rotation as expressed in Equation 5. We then used this rotational matrix to translate the quadrotor's position over the global reference frame as expressed in Equation 6 where we use the rotational matrix to translate the local position with the global target position.

$$R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} x_G \\ y_G \\ z_G \end{bmatrix} = R(\psi) \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix} \tag{6}$$

We also incorporated a GPS sensor which returns the quadrotor position in the global reference frame. To clarify, this provides less intra-system calculations and in general, removes the probability of accounting for accumulated error. Instead, we proposed to use global-oriented sensor data as input into our PID control systems and for PSO fitness.

C. Modeling PID-Controller for Quadcopter

We labeled attitude stability influenced inputs as vertical u_1, yaw u_2, pitch u_3, and roll u_4. These are used in combination as input into setting the motor's rotational velocity. u_1 is influenced by the error between the current altitude and target altitude, and u_2 is influenced by the current heading and a desired target heading provided by the next target waypoint.

The pitch u_3 and roll u_4 inputs are influenced by collected response outputs from their respective PID control systems and also influenced by two translational PIDs which include assigned PID control systems for the x and y positions. We set the desired pitch and roll angles to be set with the initial waypoint. Given the first target waypoint of our experiment as [0, 0, 3], there is no translational adjustment in the (x, y) directions. This keeps the desired angle for pitch and rolls relatively small.

The incorporated translational PID control systems are the mechanisms that influence the quadrotor to adjust its roll and pitch angle to achieve forward flight towards a target point. This is when we applied the rotational matrix described in Equation 5 to transform the GPS received global coordinates into a directional movement to global target waypoints. We utilized this approach for trajectory control and re-interpreted our attitude and altitude control inputs with the equations described in Equations 7, 8, 9, and 10. The equations below describe the individual inputs that affect drone flight. K_φ and K_θ are separate proportional constants, unrelated to our PID parameters, that ensure the adequate input response for φ and θ.

$$u_1 = ThrottlePID(z_G) \tag{7}$$

$$u_2 = YawPID(\psi_L) \tag{8}$$

$$\begin{aligned} u_3 = K_\theta * clamp(\theta_L, -1, 1) + \dot{\theta}_L + pitchPID(\theta_L) \\ + clamp(mod - XposPID(x_G), -1, 1) \end{aligned} \tag{9}$$

$$\begin{aligned} u_4 = K_\varphi * clamp(\varphi_L, -1, 1) + \dot{\varphi}_L + rollPID(\varphi_L) \\ + clamp(mod - YposPID(y_G), -1, 1) \end{aligned} \tag{10}$$

$mod - XposPID$ and $mod - YposPID$ are values retrieved from the cross-product of our rotation matrix in Equation 6 with given yaw values from the onboard IMU, which provides pitch and roll influences to move the drone from one waypoint to another. This is described below in Equation 11.

$$\begin{bmatrix} mod - XposPID \\ mod - YposPID \end{bmatrix} = R(\psi) \begin{bmatrix} XposPID(x_G) \\ YposPID(y_G) \end{bmatrix} \tag{11}$$

The attitude and altitude inputs are finally included in the motor inputs where their respective sums are values to set for the motor's rotational velocity, as seen in Equations 12, 13, 14, and 15. Like the preset proportional constants used in pitch and roll control, the K_T proportional thrust constant is a preset variable used in the foundational controller. With the PIDs recognized within motor inputs, the next part of this research explored constructing our PSO implementation.

$$m1 = K_T + u_1 - u_2 + u_3 - u_4 \tag{12}$$

$$m2 = K_T + u_1 + u_2 + u_3 + u_4 \tag{13}$$

$$m3 = K_T + u_1 + u_2 - u_3 - u_4 \tag{14}$$

$$m4 = K_T + u_1 - u_2 - u_3 + u_4 \tag{15}$$

We set to tune for twelve separate PID parameters consecutively starting with $ThrottlePID$'s K_p, K_i, and K_d, then moving to tune for $YawPID$'s parameters, and finally tuning for forward-travel with tuning $XposPID$'s and $YposPID$'s parameters. In total, this research explores tuning a total of 12 different parameters across four assigned PIDs, not including $rollPID$ and $pitchPID$. These control systems were kept as default for the second experiment as well.

IV. Experimental results

We present training results of tuned parameters for throttle, yaw, forward, and translational movements. All parameters were tuned for 70 iterations with the average fitness across three experimental trials used as the measured fitness score for the particle.

A. Tuned Throttle PID Parameters

Figures 4, 5, and 6 demonstrate the behavior of PSO as it explores all three parameters for $ThrottlePID$. The fitness used here will equate how well the quadrotor achieved *hover* state with a target altitude of 3m. Figures 4 and 6 display convergence properties between iterations 20 and 30 while Figure 5 shows the parameter does not have a significant difference through all iterations.

B. Tuned Yaw PID Parameters

The same fitness function was used for tuning $yawPID$ parameters. Once the quadrotor reaches the desired target altitude, its state will then switch to *yaw-change*, at which point it will change its heading towards the first target waypoint. Tuned results are shown in Figures 7, 8, and 9. From this exploration, we found similar convergence patterns as we observed for $ThrottlePID$. Figures 7 and 9 show convergence early in the exploration and trend towards values with common respective fitness.

C. Tuned Positional and Translational Parameters

Once the quadrotor's heading aligns with the target heading, it will begin traveling forward from its current pose and toward the target waypoint. To find the optimal parameters for forward travel, we decided to tune parameters for $xposPID$ and $yposPID$. To equate the fitness for both control systems, we utilized two fitness functions and measured the overshoot and adjustment to the waypoint including the error with flight and reference trajectories.

Figures 10, 11, and 12 portray the tuning results for $xposPID$. Figures 10 and 12 display a convergence for both K_p and K_d gains at around 20-30 iterations. Figure 11 shows that the algorithm begins to explore the search space, but after iteration 30 returns the swarm to values closer to the initial.

Figures 13, 14, and 15 demonstrate the particle position average across 70 iterations for $yposPID$. Particle K_p, in Figure 13, has a small converging range shortly before reaching 70 iterations, which we used for our tuned parameters. Parameter K_d, in Figure 14, shows convergence shortly after 20 iterations. The algorithm continues to find a relative converging area through remaining iterations for gain K_d. Lastly, a similar converging behavior is displayed in Figure 14 for the K_i parameter as was displayed in Figure 11 with $xposPID$'s K_i parameter.

TABLE I: Final Particle Disparity for applied PID controls

	Tuned-avg	**Global-Best**	**Default**
ThrottlePID-K_p	1.52 +/-0.13	1.789	1.2
ThrottlePID-K_i	0.22 +/-0.06	0.256	0.1
ThrottlePID-K_d	0.57 +/-0.14	0.754	0.5
YawPID-K_p	0.84 +/-0.17	1.148	0.232
YawPID-K_i	0.13 +/-0.05	0.062	0.1
YawPID-K_d	0.28 +/-0.18	0.465	0.3722
xposPID-K_p	0.88 +/-0.22	1.238	0.232
xposPID-K_i	0.02 +/-0.03	0.001	0.01
xposPID-K_d	0.83 +/-0.38	0.857	0.6756
yposPID-K_p	0.75 +/-0.02	1.275	0.232
yposPID-K_i	0.02 +/-0.02	0.001	0.01
yposPID-K_d	0.99 +/-0.38	0.757	0.6756

D. Final PSO-tuned PID parameters

Table I displays the average parameter value over the entire swarm at the final tuning iteration. We labeled these as "tuned" parameters throughout. The last column of the table also provides the global best parameters based on the best-measured fitness score across the swarm through all iterations.

E. Comparison with MSE from Reference Trajectory Course

After acquiring tuned parameters for our assigned PID controls, we observed overshoot and adjustment time from *grounded* state to *hover* state at which point the quadrotor hovers at 3m. This provided us with a visualization for comparing default, tuned, and global-best parameters and how well each performed with transitioning the quadrotor to *hover* state. Figure 16 displays a comparative visual for thrust applied by default, tuned, and global-best parameters. The figure shows evidence that global-best parameters provided the least amount of overshoot and adjustment time compared to our tuned and default parameters.

The next comparison involved measuring performance for default, tuned, and global-best parameters for $YawPID$. We compiled this by visualizing overshoot and adjustment from the initial *hover* heading and the change over to the target heading in Figure 17. Again, the best performers are global-best found parameters with tuned parameters having exceptional performance over default.

Our last comparison between our tested parameters consists of visualizing the quadrotor's position as it travels from the initial "takeoff" state waypoint to the target waypoint at [7.6, -2.5, 3]. Figures 18 and 19 portray the results of the quadrotor's movement with a given set of parameters. Once again, the best performers were tuned and global-best parameters. With default parameters, the quadrotor took much longer to get into the state where it could travel to the target waypoint. It also demonstrated a large amount of overshoot and little improvement with adjustment time.

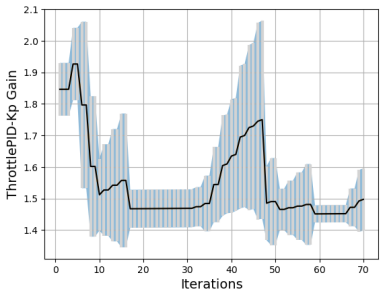


Fig. 4: Gain for throttle PID K_p

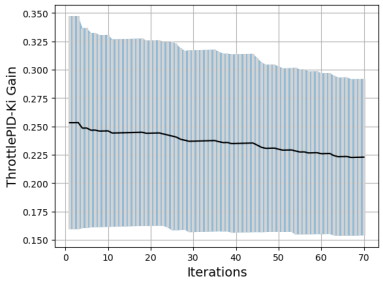


Fig. 5: Gain for throttle PID K_i

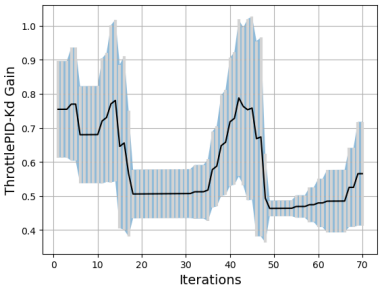


Fig. 6: Gain for throttle PID K_d

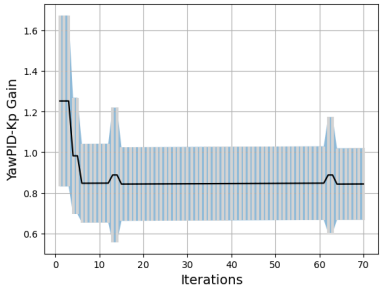


Fig. 7: Gain for yaw PID K_p

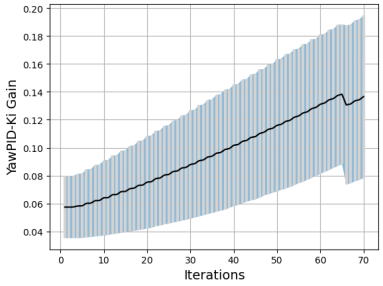


Fig. 8: Gain for yaw PID K_i

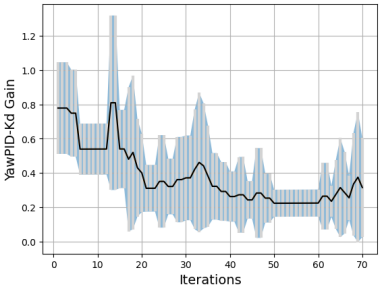


Fig. 9: Gain for yaw PID K_d

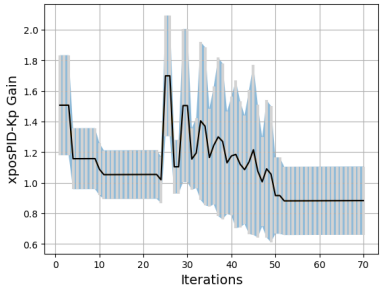


Fig. 10: Gain for x position PID K_p

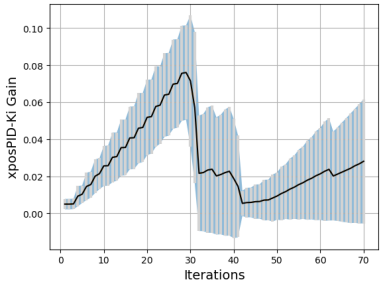


Fig. 11: Gain for x position PID K_i

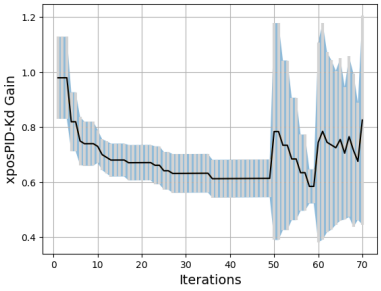


Fig. 12: Gain for x position PID K_d

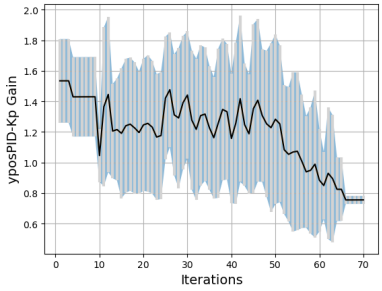


Fig. 13: Gain for y position PID K_p

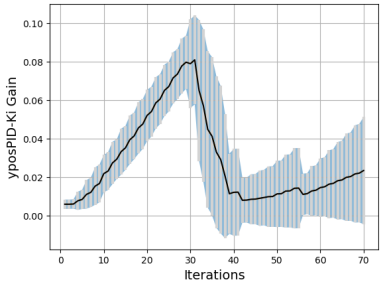


Fig. 14: Gain for y position PID K_i

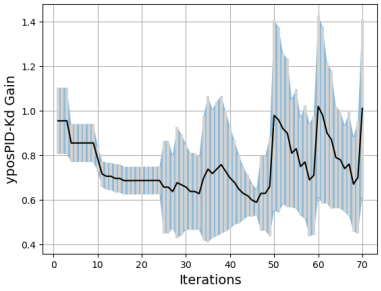


Fig. 15: Gain for y position PID K_d

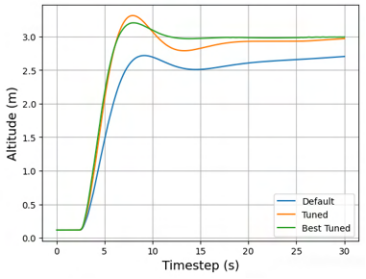


Fig. 16: The performance of throttle parameter

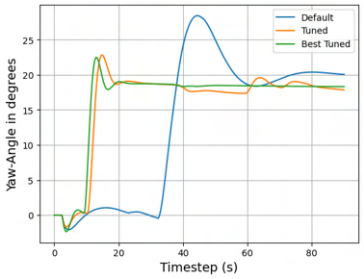


Fig. 17: The performance of Yaw parameter

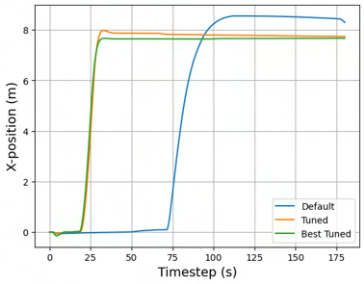


Fig. 18: The performance of XposPID parameter

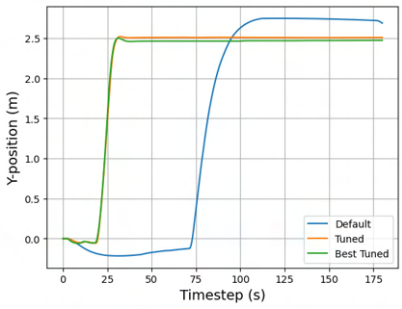


Fig. 19: The performance of YposPID parameter

To assess the quality of performance with our tuned parameters we created a second experiment involving collecting errors between flight and reference trajectories across three different paths: a 20-waypoint "Pac-Man" course, a 28-waypoint "infinity-loop" course, and an 8-waypoint "triangle" course. All courses are conducted at an altitude of 3m.

Each course has its respective purpose for testing the parameter's performance. The "PacMan" course is designed to validate continuous flight with found parameters. The "infinity-loop" is designed to test performance as the quadrotor rotates yaw clockwise and counter-clockwise while traveling. The final test course involves traversing two equilateral triangle paths, both with different distances at 4m and 12m. We used this course to test how well it performs when traversing short distances and long distances.

Figures 20, 21 and 22 display performance trajectories between all three parameter sets for each course. In addition, Tables II, III and IV provide supplemental respective results that include $log_{10}SSE$ measurements. Not only do global-best parameters allow the quadrotor to finish each course in record time but it also obtained the least error. Once again, the best-performing set of tuned gains can be found with global parameters.

Figure 20 demonstrates that there is little to no major error correction when traversing the course. This can be seen with the noted overshooting trajectories from default parameters and tuned parameters. The same can be said for Figure 21 where default and tuned parameters spend more time adjusting to waypoints than compared global-best parameters. Figure 22 has noticeable trajectory error with traveling greater distances, including global-best parameters having the least error. Overall, we believe that using PSO to find optimal gains for a PID controller is a sufficient and encouraged approach to achieving flawless autonomous flight.

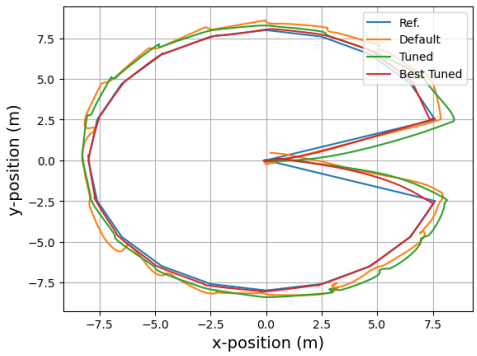


Fig. 20: First Trajectory Comparison

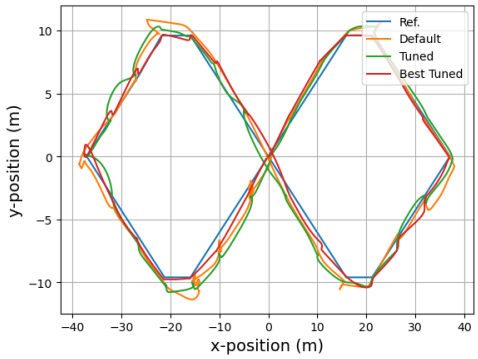


Fig. 21: Second Trajectory Comparison

V. Conclusion

This research sheds light on the advantages of utilizing Particle Swarm Optimization (PSO) for tuning PID parameters in a quadrotor's flight controller. In our first experiment, we demonstrated that applying PSO could identify optimal PID parameters, resulting in minimal overshoot and adjustment time. PSO not only found parameters that surpassed default PID settings but also achieved the global

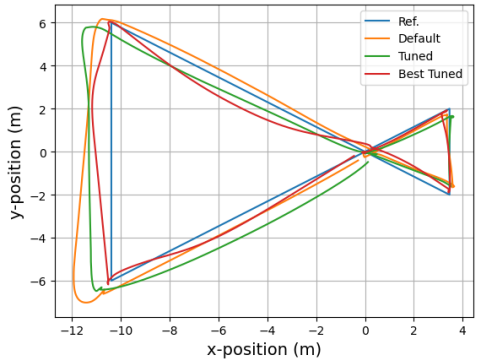


Fig. 22: Third Trajectory Comparison

TABLE II: $log_{10}SSE$ comparison between parameter sets for "Pac-Man" course

	Course-Error	Course Times
Default	5.2249	9m 55s
PSO-Tuned	4.4850	3m 50s
PSO-Global-Best	4.3598	2m 33s

best parameters, offering optimal overall performance. This was further validated in our second experiment, where a comparison of flight trajectories revealed that the global best parameters were the most efficient, completing the course four times faster than default parameters. Although PSO-tuned parameters exhibited more accumulated error over the trajectory, they still outperformed default parameters in completion time. Additionally, we confirmed that employing combined fitness functions for tuning PID parameters can serve as a supplementary method to validate fitness scores. We utilized one fitness function to measure overshoot and adjustment time, while another measured $log_{10}SSE$ between flight and reference trajectories for tuning translational PID control systems.

VI. Future Works

This research aims to stimulate further exploration into applying this tuning method to multiple quadrotors using a tuned PID control system in support of collaborative navigation. We believe using tuned PID control systems can decrease the amount of positional-error during flight and when adjusting position with collaborative navigation and collision avoidance algorithms. Additionally, we desire to explore comparisons between tuned PID controllers and other controllers that provide predictive position without relying on GPS and IMU sensors in Webots; and instead, utilize local measurement sensors such as accelerometers and gyroscopes.

TABLE III: $log_{10}SSE$ comparison between parameter sets for "Infinity-Loop" course

	Course-Error	Course Times
Default	6.8215	16m 46s
PSO-Tuned	6.7335	14m 07s
PSO-Global-Best	6.2770	5m 22s

TABLE IV: $log_{10}SSE$ comparison between parameter sets for "Triangle" course

	Course-Error	Course Times
Default	5.8365	4m 42s
PSO-Tuned	5.7286	3m 38s
PSO-Global-Best	4.9862	1m 23s

ACKNOWLEDGMENT

This work is supported by the SLA (Scientific Leadership Award) program through DHS (Department of Homeland Security) Award No. 21STSLA00009-01-00. The authors would also like to acknowledge the partial funding provided by the CREST MECIS program through NSF (National Science Foundation) Award No. 2112650 and the MSI program through NSF Award No. 2318682.

References

[1] N. Minorsky, "Directional stability of automatically steered bodies," *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, p. 280–309, 1922.

[2] X. D. Jing and X. F. Wang, "PSO algorithm tuning PI_ PID controller parameters of quad-rotor UAV," in *Journal of Physics Conference Series*, vol. 2228, Mar. 2022, pp. 12–17.

[3] J. A. Cárdenas, U. E. Carrero, E. C. Camacho, and J. M. Calderón, "Optimal pid ø axis control for uav quadrotor based on multi-objective pso," *11th IFAC Symposium on Intelligent Autonomous Vehicles IAV*, vol. 55, no. 14, pp. 101–106, 2022.

[4] N. S. Özbek, M. Önkol, and M. Önder Efe, "Feedback control strategies for quadrotor-type aerial robots: a survey," *Transactions of the Institute of Measurement and Control*, vol. 38, no. 5, pp. 529–554, 2016.

[5] J. Li and Y. Li, "Dynamic analysis and pid control for a quadrotor," in *2011 IEEE International Conference on Mechatronics and Automation*, 2011, pp. 573–578.

[6] D. Gautam and C. Ha, "Control of a quadrotor using a smart self-tuning fuzzy pid controller," *International Journal of Advanced Robotic Systems*, vol. 10, no. 11, p. 380, 2013.

[7] J. Yang, Z. Cai, Q. Lin, and Y. Wang, "Self-tuning pid control design for quadrotor uav based on adaptive pole placement control," in *2013 Chinese Automation Congress*, 2013, pp. 233–237.

[8] M. H. Amoozgar, A. Chamseddine, and Y. Zhang, "Fault-tolerant fuzzy gain-scheduled pid for a quadrotor helicopter testbed in the presence of actuator faults," *IFAC Proceedings Volumes*, vol. 45, no. 3, pp. 282–287, 2012, 2nd IFAC Conference on Advances in PID Control.

[9] M. Moness and A. M. Moustafa, "Tuning a digital multivariable controller for a lab-scale helicopter system via simulated annealing and evolutionary algorithms," *Transactions of the Institute of Measurement and Control*, vol. 37, no. 10, pp. 1254–1273, 2015.

[10] M. B. Abdelghany, A. M. Moustafa, and M. Moness, "Benchmarking tracking autopilots for quadrotor aerial robotic system using heuristic nonlinear controllers," *Drones*, vol. 6, no. 12, pp. 1–29, 2022.

[11] S. Madruga, A. De Holanda Barreto Martins Tavares, G. Basso, D. Nascimento, T. P., and A. Brito, "A pso-based tuning algorithm for quadcopter controllers," in *Proceedings XXII Congresso Brasileiro de Automática*, vol. 1, no. 1, 2018.

[12] M. S. Hasanoglu and M. Dolen, "Multi-objective feasibility enhanced particle swarm optimization," *Engineering Optimization*, vol. 50, no. 12, pp. 2013–2037, 2018.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[14] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 1998, pp. 69–73.

[15] C.-F. Wang and K. Liu, "A novel particle swarm optimization algorithm for global optimization," *Intell. Neuroscience*, vol. 2016, jan 2016.