2024 IEEE 18th International Conference on Control & Automation (ICCA)
June 18-21, 2024. Reykjavík, Iceland

Gestelt: A framework for accelerating the sim-to-real transition for swarm UAVs

John Tan, Tianchen Sun, Feng Lin, Rodney Teo, Boo Cheong Khoo

Abstract— **Research in aerial swarms have gained traction in recent years and there appears to be a lack of user-friendly frameworks with a focus on bringing swarm UAVs from simulation to actual flight. Furthermore, spatial constraints and resource challenges hinder the validation of larger-scale swarm algorithms.**

To tackle these issues, we propose Gestelt, a relatively lightweight framework that accelerates the sim-to-real transition for swarm algorithms. First, the modular design of Gestelt is highlighted to illustrate it's generalization to multiple types of planning algorithms and paradigms. Next, we outline an approach to model any given quadrotor platform for use in our framework's simulation environment. Another unique feature of Gestelt is it's virtual-physical environment, which can simultaneously host both virtual and physical agents, thereby providing an intermediate platform for testing larger-scale swarm algorithms safely. Finally, we implement an asymptotically stable closed-loop control technique known as Robust Perfect Tracking (RPT) to track the reference trajectories of the swarm agents in the face of disturbances.

We demonstrate our framework through physical experiments featuring our custom swarm platform, the NUSwarm drone, where we show a swarm navigation scenario for 3 physical and 3 virtual drones. A video demonstrating the virtual-physical environment can be found at `https://youtu.be/FYlwz2yZxLE`

I. INTRODUCTION

Autonomous swarm flight has garnered much research interest in recent years as swarm UAVs are able to tackle various problems more efficiently than individual UAVs. These problems include search and rescue (SAR), industrial inspection, disaster monitoring, and agriculture [1] to name a few. One of the main research challenges our framework aims to tackle is the fast and safe navigation of multiple autonomous UAVs in dense obstacle environments.

We first compare existing frameworks developed for autonomous quadrotor flight. One single quadrotor framework is Agilicious [2], which focuses on high speed vision-based flight, being able to support both model-based and neural-network-based controllers. Another framework, KR Autonomous Flight [3], provides a full pipeline for single GPS-denied autonomous flight, with different components performing state estimation, mapping, planning and controls. Nevertheless, the aforementioned high performance pipelines focus on single UAV flight. There have been frameworks that seek to fill the gap in multi-UAV navigation, these include XTDrone [4], Aerostack 2 [5] and Crazyswarm [6]. One key similarity between these existing frameworks is that they only simulate predefined drone models and have no established approach to simulate custom drone models. Moreover, these frameworks are not intended for running on resource-constrained computers and hence require a heavier computational payload, ruling out smaller swarm drones. Additionally, there is a considerable gap in the sim-to-real transition and they lack an intermediate environment to safely test swarm algorithms before deployment in real flight. In most physical environments, these issues are coupled with the challenges of inadequate space and resources to simultaneously fly many physical drones safely.

Our paper is motivated by the above challenges in testing swarm algorithms and we propose the Gestelt framework to address these challenges and close the gap in the sim-to-real transition for swarm UAVs. The technical contributions are summarized as follows:

1) A well-defined approach to use experimental data from a custom quadrotor model for simulation and deployment as part of a swarm configuration (Fig. 1). We reduce the crucial leap from sim to real by providing a virtual-physical environment, where both physical and virtual drones navigate the same virtual obstacle environment, therefore enabling the safe testing of larger-scale swarm algorithms.
2) A modular architecture for Gestelt that generalizes to different types of planning algorithms and optimization approaches has been proposed. Moreover, integrated tools and modules such as a swarm bridge, simulated sensors, and swarm-to-obstacle collision checker improve the ease of testing arbitrary swarm configurations in both simulation and reality.
3) Robust Perfect Tracking (RPT) control technique with the rotor drag compensation is employed for designing a position controller such that the resulting closed-loop system is asymptotically stable and the controlled output almost perfectly tracks a given reference signal in the presence of any initial condition and external disturbances.

II. ARCHITECTURE OF GESTELT

A. Motivation

The architectural design of Gestelt is motivated by the modularity of state of the art decentralized multi-agent plan-

John Tan is with the Department of Mechanical Engineering, National University of Singapore johntgz@nus.edu.sg
Tianchen Sun is with the Department of Mechanical Engineering, National University of Singapore tianchen.sun@u.nus.edu
Feng Lin is with the Department of Electrical and Computer Engineering, National University of Singapore feng_lin@nus.edu.sg
Rodney Teo is with the National University of Singapore tsltshr@nus.edu.sg
Boo Cheong Khoo is with the Department of Mechanical Engineering, National University of Singapore mpekbc@nus.edu.sg

2024 IEEE 18th International Conference on Control & Automation (ICCA) | 979-8-3503-5440-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCA62789.2024.10591863

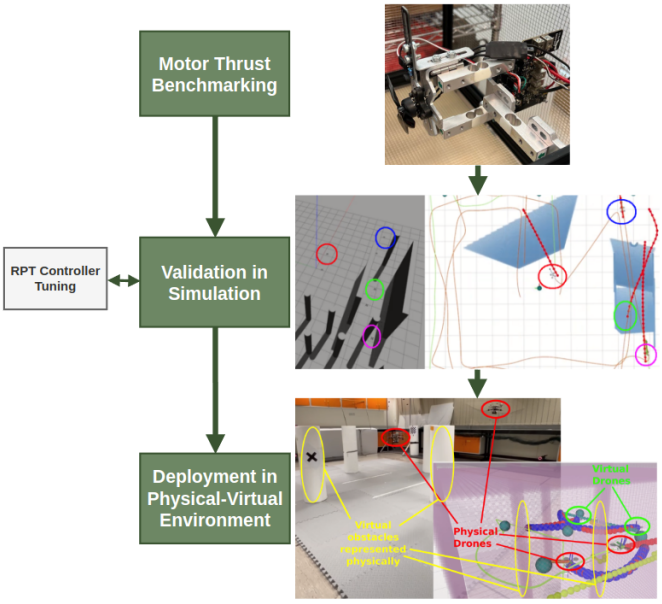


Fig. 1: Virtual-physical environment: Physical and virtual quadrotors navigate in the same obstacle map.

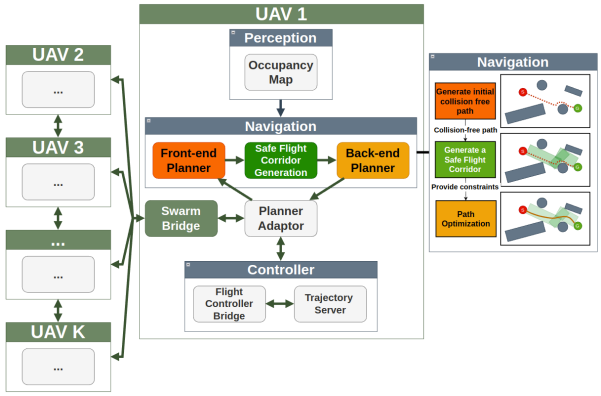


Fig. 2: High-level architecture of Gestelt

ners which split the planning stage into a front-end planner and back-end optimizer. The front-end planner outputs an initial collision-free path and/or safe flight corridor which is then fed to the back-end optimizer in the form of an initial guess and optimization constraint.

One of these planners, MADER [7], employs a front-end planner utilising the "Octopus Search" algorithm to obtain an initial path in the MINVO basis which serves as a constraint to a non-convex optimization problem. Similarly, EGOSwarm [8] makes use of a front-end local planner to obtain gradient information for the back-end optimizer, guiding convergence to a feasible and safe path. One other planner, DLSS [9] relies on a three stage planning pipeline that performs discrete planning, safe navigation corridor construction, and then trajectory optimization.

B. Overview and detail

The modular architecture of Gestelt renders it algorithm-agnostic, allowing for different planning algorithms through the replacement of modules in the planning pipeline (Fig. 2). For instance, one could change the initial front-end path generation from a search-based planner like A* [10] to a sampling-based planner like the rapidly-exploring random trees (RRT) [11] without the modification of other connected components within the pipeline. We proceed to elaborate on the 5 main modules of our framework, the **navigation module**, **perception module**, **controller module**, **planner adaptor** and **swarm bridge**. Lastly, we explain the purpose of the **virtual-physical environment**.

Navigation module The navigation module (Fig. 2) comprises the front-end planner, safe flight corridor generation and back-end planner. The front-end planner utilises either a search-based or sampling-based algorithm to plan an initial collision-free path from a given starting point to the goal position. Then, the front-end path acts as a reference for safe flight corridor generation, similar to what has been done in [12]. Finally, the front-end path and the safe flight corridor are given as an initial guess and constraint respectively to the back-end optimizer.

Perception module The perception module provides a three-dimensional voxel grid map that is queried by the navigation module for collision checking. Map construction is performed using the probabilistic mapping method with point cloud input from either a simulated or actual sensor.

Controller module The controller module is a state machine controlling the flow between different low-level flight states such as taking off, landing, emergency stop and mission mode.

Planner adaptor The last module, planner adaptor, acts as an abstraction layer between the controller and navigation module. This is useful because existing available planners can be integrated without any heavy modification. The only requirement is to establish a set of ROS message interfaces between the planner and planner adaptor, thereby treating the planner module as a black box.

Swarm Bridge The swarm bridge serves as a message relay between multiple agents, where crucial information such as flight trajectories, agent state, and collective goals are compressed and then shared between the agents to improve the reliability of communications especially in outdoor environments. Our current implementation relies on ROS1(TCP/UDP) [13]. We will improve on the communication robustness of the framework by transiting to a decentralized middleware such as ROS2(DDS) [14].

Virtual-Physical Environment The virtual physical environment serves as a testbed for validating larger-scale swarm algorithms. Assuming that ground truth is available through an external localization system (e.g. motion capture setup), this approach enables us to solely focus on testing swarm algorithms without concerns about sensing or localization errors due to noise from hardware sensors. Another benefit of this approach is that it can validate planning safety on physical drones while reducing risks of collisions with physical obstacles or agents which are replaced by their virtual counterparts.

Symbol	Description
$P_{\mathrm{w}} = [x,\ y,\ z]^{\mathsf{T}}$	Position vector in wold frame
V_{w}	Velocity vector in world frame
V_{b}	Velocity vector in body frame
$\Omega = (p, q, r)^{\mathsf{T}}$	Angular velocity in body frame
$\boldsymbol{q} = (q_0, q_1, q_2, q_3)$	Attitude of the quadrotor in quaternion
${}^{\mathrm{w}}R_{\mathrm{b}}$	Rotation matrix from body to world frame
m	Mass of the quadrotor
$J = \mathrm{diag}(J_{xx}, J_{yy}, J_{zz})$	Inertia matrix of the quadrotor

TABLE I: Definition of model parameters

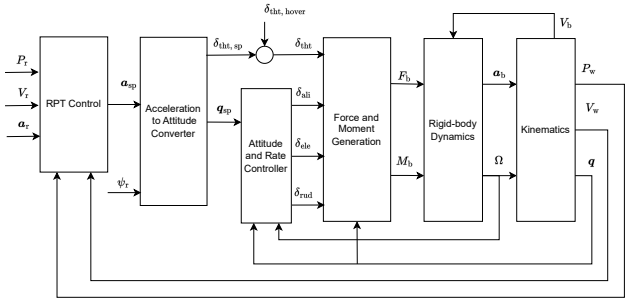


Fig. 3: RPT control structure

III. Modelling and Simulation

A. Quadrotor Dynamics and Control

We start by describing the dynamic model of the quadrotor model in this section. Two coordinate systems are defined: world frame w and body frame b. For convenience, common symbols and variables to be used in the derivation of the model are listed in Table I.

The dynamic model of the quadrotor is given below:

$$\dot{P}_{\mathrm{w}} = V_{\mathrm{w}}, \qquad \dot{V}_{\mathrm{w}} = m^{-1}\,{}^{\mathrm{w}}R_{\mathrm{b}}F_b \tag{1}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}Q(\boldsymbol{q})\Omega, \quad \dot{\Omega} = J^{-1}\left[-\Omega \times J\Omega + M_{\mathrm{b}}\right] \tag{2}$$

where

$$Q(\boldsymbol{q}) = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \tag{3}$$

F_{b} and M_{b} represent the forces and moments exerted on the quadrotor, elaborated upon extensively in Section III-C.

Robust Perfect Tracking Control A controller offering high tracking performance and ease of tuning is essential for facilitating the deployment and experimentation of swarm algorithms. The default PX4 cascaded PID structure often falls short of meeting these requirements, particularly in terms of convenience in parameter tuning. In Gestelt, Robust Perfect Tracking (RPT) control [15] has been implemented to expedite hardware deployment. With minimal modifications to the PX4 controller source code, this approach simplifies controller tuning with straightforward system dynamics requirements. The RPT controller structure is shown in Fig. 3. Given that rotor drag effects are non-negligible in high speed flight, another improvement we made is to implement the rotor drag compensation from [16].

1) Robust Perfect Control Theory: RPT position control simplifies the dynamics of the inner control loop as a universal six-DoF particle, whose dynamics are expressed as:

$$\begin{bmatrix} \dot{P}_{\mathrm{w}} \\ \dot{V}_{\mathrm{w}} \end{bmatrix} = \begin{bmatrix} 0 & \mathrm{I} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P_{\mathrm{w}} \\ V_{\mathrm{w}} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{a}_{\mathrm{sp}} \tag{4}$$

where the acceleration set-point $\boldsymbol{a}_{\mathrm{sp}} = [a_{\mathrm{x,sp}}, a_{\mathrm{y,sp}}, a_{\mathrm{z,sp}}]^{\mathsf{T}}$ is the output of the RPT position controller.

The Robust Perfect Tracking control law [15] takes in desired position P_{r}, velocity V_{r}, rotor drag compensation term $\boldsymbol{a}_{\mathrm{drag}}$, and acceleration $\boldsymbol{a}_{\mathrm{r}}$ as control feed-forward. It is formulated as:

$$\begin{aligned} \boldsymbol{a}_{\mathrm{sp}} = {} & F_{\mathrm{I}}\int(P_{\mathrm{r}} - P_{\mathrm{w}})\,dt + F_{\mathrm{P}}\,(P_{\mathrm{r}} - P_{\mathrm{w}}) \\ & + F_{\mathrm{v}}\,(V_{\mathrm{r}} - V_{\mathrm{w}}) + \boldsymbol{a}_{\mathrm{r}} - \boldsymbol{a}_{\mathrm{drag}} \end{aligned} \tag{5}$$

where:

$$F_{\mathrm{I}} = \left[\frac{k_{\mathrm{i,xy}}\omega_{\mathrm{n,xy}}^2}{\epsilon_{\mathrm{xy}}^3},\ \frac{k_{\mathrm{i,xy}}\omega_{\mathrm{n,xy}}^2}{\epsilon_{\mathrm{xy}}^3},\ \frac{k_{\mathrm{i,z}}\omega_{\mathrm{n,z}}^2}{\epsilon_{\mathrm{z}}^3}\right]^{\mathsf{T}}$$

$$F_{\mathrm{V}} = \left[\frac{2\zeta_{\mathrm{xy}}\omega_{\mathrm{n,xy}} + k_{\mathrm{i,xy}}}{\epsilon_{\mathrm{xy}}},\ \frac{2\zeta_{\mathrm{xy}}\omega_{\mathrm{n,xy}} + k_{\mathrm{i,xy}}}{\epsilon_{\mathrm{xy}}},\ \frac{2\zeta_{\mathrm{z}}\omega_{\mathrm{nz}} + k_{\mathrm{i,z}}}{\epsilon_{\mathrm{z}}}\right]^{\mathsf{T}}$$

$$F_{\mathrm{P}} = \left[\frac{\omega_{\mathrm{n,xy}}^2 + 2\zeta_{\mathrm{xy}}\omega_{\mathrm{n,xy}}k_{\mathrm{i,xy}}}{\epsilon_{\mathrm{xy}}^2}, \frac{\omega_{\mathrm{n,xy}}^2 + 2\zeta_{\mathrm{xy}}\omega_{\mathrm{n,xy}}k_{\mathrm{i,xy}}}{\epsilon_{\mathrm{xy}}^2}, \frac{\omega_{\mathrm{n,z}}^2 + 2\zeta_{\mathrm{z}}\omega_{\mathrm{n,z}}k_{\mathrm{i,z}}}{\epsilon_{\mathrm{z}}^2}\right]^{\mathsf{T}}$$

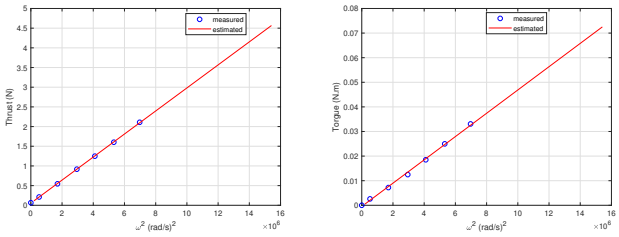
where $(\omega_{\mathrm{n,xy}}, \omega_{\mathrm{n,z}})$ and $(\zeta_{\mathrm{xy}}, \zeta_{\mathrm{z}})$ are the desired system's nominal natural frequency and damping ratio respectively. $(k_{\mathrm{i,xy}}, k_{\mathrm{i,z}})$ is the absolute location of the closed-loop system third pole that lies on the real axis, in the horizontal and vertical directions respectively. These three parameters explicitly determine the close-loop system characteristics. The parameter $(\epsilon_{\mathrm{xy}}, \epsilon_{\mathrm{z}})$ indicates the time constant of the convergence of tracking error. In Chapter 9.4 of Robust and H Control [17], it is proven that as ϵ approaches zero, the controller's tracking error will also converge toward zero. According to [16], $\boldsymbol{a}_{\mathrm{drag}} = -R_{\mathrm{r}}DR_{\mathrm{r}}^{\mathsf{T}}V_{\mathrm{r}}$ is the acceleration induced by the rotor drag, where R_{r} is the reference pose calculated by the feed-forward acceleration $\boldsymbol{a}_{\mathrm{r}}$. $D = \mathrm{diag}(d_x, d_y, d_z)$ is a constant diagonal rotor drag coefficient matrix, normalized by the drone mass and only (d_x, d_y) is considered for the compensation.

With these precise controller performance specifications, achieving an accurate and robust controller becomes feasible with minimal tuning. The tracking results are presented in section IV-B.

2) Implementation within PX4 autopilot: Referring to Eq. 5, the controller feedback and feed-forward quantities coincide with the PX4 original cascaded position controller. As a result, the RPT controller could be integrated into the PX4 controller source code, specifically the `mc_position_control` file, as an additional function to replace both the `_positionControl` and `_velocityControl` functions.

B. Collection of physical parameters

In order to determine the thrust and torque coefficients experimentally, we perform thrust benchmarking on the



(a) Motor thrust T (b) Motor torque Q

Fig. 4: Experimental data of thrust (left) and torque (right) against angular velocity ω

motor and obtain experimental data shown in Fig. 5. Note that for Eq. (6) which uses a first order approximation, we follow the convention in Gazebo which relates the `motorConstant` to thrust as in Eq. 8, and the `momentConstant` as a ratio of the moment to thrust coefficient in Eq. 9. ω_i is defined as the angular velocity of the i-th propeller. The `motorConstant` is therefore computed as 2.9265e^{-7} kg m and the `motorConstant` as 0.016178 m.

$$\begin{aligned} \texttt{motorConstant} &= \mathrm{k_t} = \frac{T}{{\omega_i}^2}, \\ \texttt{momentConstant} &= \mathrm{k_m} = \frac{T\ {\omega_i}^2}{Q\ {\omega_i}^2} \end{aligned} \tag{6}$$

Next, we can approximate the mass moment of inertia $J = \mathrm{diag}(J_{xx}, J_{yy}, J_{zz})$ from a Computer Aided Design (CAD) model, where the assumption is made that the mass moment of inertia is symmetrical about the X, Y and Z axes of the quadrotor body. We therefore obtain the $J_{xx} = 3.9195\text{e}^{-4}$, $J_{yy} = 4.0515\text{e}^{-4}$ and $J_{zz} = 6.3890\text{e}^{-4}$.

C. Modelling and simulation in Gazebo

The forces acting on a quadcopter compose of the gravitational force, gyroscopic effects, aerodynamic drag and rotor effects [18] presented as Eq. 7. To simulate the drone dynamics, we make use of the PX4 Autopilot Gazebo plugin [19]. Equations 8-13 model the aerodynamic effects of the rotating propeller. The other forces and moments such as gravity and gyroscopic effects resulting from quadcopter motion are implicitly handled by Gazebo's physics engine.

$$\begin{bmatrix} F_\mathrm{b} \\ M_\mathrm{b} \end{bmatrix} = \begin{bmatrix} F_\mathrm{gravity} \\ 0 \end{bmatrix} + \begin{bmatrix} F_\mathrm{rotor} \\ M_\mathrm{rotor} \end{bmatrix} + \begin{bmatrix} F_\mathrm{drag} \\ M_\mathrm{drag} \end{bmatrix} + \begin{bmatrix} 0 \\ M_\mathrm{gyro} \end{bmatrix} \tag{7}$$

Rotor thrust and torque The thrust and torque generated by the rotors are defined as:

$$F_\mathrm{rotor} = [0,\ 0,\ f]^\intercal \tag{8}$$

$$M_\mathrm{rotor} = [\tau_x,\ \tau_y,\ \tau_z]^\intercal \tag{9}$$

where

$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} \mathrm{k_t} & \mathrm{k_t} & \mathrm{k_t} & \mathrm{k_t} \\ 0 & \mathrm{k_t}L & 0 & -\mathrm{k_t}L \\ -\mathrm{k_t}L & 0 & \mathrm{k_t}L & 0 \\ \mathrm{k_m k_t} & -\mathrm{k_m k_t} & \mathrm{k_m k_t} & -\mathrm{k_m k_t} \end{bmatrix} \begin{bmatrix} {\omega_1}^2 \\ {\omega_2}^2 \\ {\omega_3}^2 \\ {\omega_4}^2 \end{bmatrix} \tag{10}$$

and $\mathrm{k_t}$ is the `motorConstant` and $\mathrm{k_m}$ is the `momentConstant`. L is distance between the z-axis of the quadrotor body center and the rotor's rotational axis.

Rotor drag The relative wind velocity acting at the propeller's geometric center is defined as V_rw, and it's perpendicular and parallel component as $V_{\perp Z_\mathrm{mr}}$ and $V_{\parallel Z_\mathrm{mr}}$ respectively:

$$\begin{aligned} V_\mathrm{rw} &= V_\mathrm{prop} - V_\mathrm{wind} \\ V_{\parallel Z_\mathrm{mr}} &= (V_\mathrm{rw} \cdot Z_\mathrm{mr})\ Z_\mathrm{mr} \\ V_{\perp Z_\mathrm{mr}} &= V_\mathrm{rw} - V_{\parallel Z_\mathrm{mr}} \end{aligned} \tag{11}$$

where V_prop and V_wind are the linear velocities of the propeller center and wind respectively, and Z_mr the propeller rotational axis.

The drag force F_drag is the sum of it's parallel and perpendicular components with respect to Z_mr, as described in Eq. (12):

$$\begin{aligned} F_\mathrm{drag} &= F_{\perp\mathrm{drag}} + F_{\parallel\mathrm{drag}} \\ F_{\perp\mathrm{drag}} &= \mathrm{k_{drag}}\ V_{\perp Z_\mathrm{mr}}\ \omega_i \\ F_{\parallel\mathrm{drag}} &= -\frac{\|V_{\parallel Z_\mathrm{mr}}\|}{\alpha}\ F_\mathrm{rotor} \end{aligned} \tag{12}$$

where the constant $\mathrm{k_{drag}}$ is the drag coefficient in forward flight, constant α is the maximum $V_{\parallel Z_\mathrm{mr}}$ at which the propeller produces zero thrust, and ω_i is the i-th propeller's angular velocity. α is set as 25 m/s in the plugin. The assumption made is that the drag force scales linearly with $V_{\parallel Z_\mathrm{mr}}$.

For the rotor drag coefficient $\mathrm{k_{drag}}$, currently we use an estimated value that minimizes the gap in performance from simulation to actual flight. Further experiments using a wind tunnel to measure more accurate values will be considered.

Rolling moment The rolling moment is an aerodynamic effect produced during forward flight, caused by the advancing propeller blade producing more lift than the retreating blade. It is calculated using Eq. 13 and is relatively small compared to the other forces and moments acting on the quadrotor.

$$M_\mathrm{rolling} = -\epsilon\ \mathrm{k_{rolling}}\ V_{\perp Z_\mathrm{mr}}\ {\omega_i}^2 \tag{13}$$

Gravitational effects Gravitational effects act universally on objects in Gazebo, resulting in the gravitational force F_gravity.

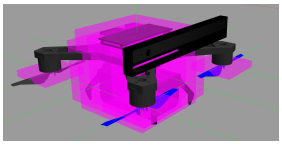
Gyroscopic effects The Gazebo's physics engine supports implicit solving for gyroscopic forces and provides a gyroscopic moment M_gyro on the quadrotor.

Rotor drag torque The rotor drag torque M_drag is related to the mass moment of inertia of the propeller and it's angular velocity, being implicitly acted on by the physics engine.

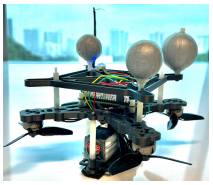
Through thrust bench testing, we obtain experimental data in table II. We also relate the Gazebo parameters to their corresponding constants in Eq. 8-13. The `maxRotVelocity` is set as the maximum angular velocity corresponding to the maximum continuous current allowable by the motor ESCs, which is 2700 rad/s at 5A continuous current.

Symbol	Parameter Name (Gazebo)	Value	Unit
k_t	motorConstant	$2.9265e^{-7}$	kg m
k_m	momentConstant	0.016178	m
k_{drag}	rotorDragCoefficient	$2.4193e^{-5}$	kg
$k_{rolling}$	rollingMomentCoefficient	$1.0e^{-6}$	kg m
$\omega_{i,max}$	maxRotVelocity	2700	rad/s
J_{xx}	ixx	$3.9195e^{-4}$	kg m^2
J_{yy}	iyy	$4.0515e^{-4}$	kg m^2
J_{zz}	izz	$6.3890e^{-4}$	kg m^2

TABLE II: Gazebo parameters derived from experimental data



(a) Moment of Inertia as visualized in Gazebo



(b) The NUSwarm quadrotor

Fig. 5: The NUSwarm quadcopter

IV. RESULTS AND DISCUSSION

A. Swarm navigation in the virtual-physical environment

We go on to show how the physical-virtual environment aids the transition of swarm-based algorithms from simulation to actual flight.

First, we run the swarm algorithm (EGOSwarm [8]) in the simulation environment with our approximate dynamic model (Fig. 1), where it is shown navigating a complex obstacle environment at speeds of up to 3 $\mathrm{m/s}$ with the NUSwarm's simulated flight dynamics. Having validated the swarm algorithm in simulation, we transition to testing in the virtual-physical environment (Fig. 6) with 6 drones (3 physical and 3 virtual) at 1.5 $\mathrm{m/s}$. Another test for the same algorithm pipeline with four drones (two physical and two virtual) is shown in Fig. 1. The physical cylinders placed in the flight arena corresponds to the virtual obstacles.

B. RPT Controller's tracking performance

The RPT controller performance has been validated using a circular trajectory that provides position, velocity, and acceleration as references. The circle radius is 1.5 m and the maximum acceleration is 13.5 $\mathrm{m/s^2}$. The tracking performance of this controller is indicated in Fig. 7, with a maximum tracking error of 0.16 m.

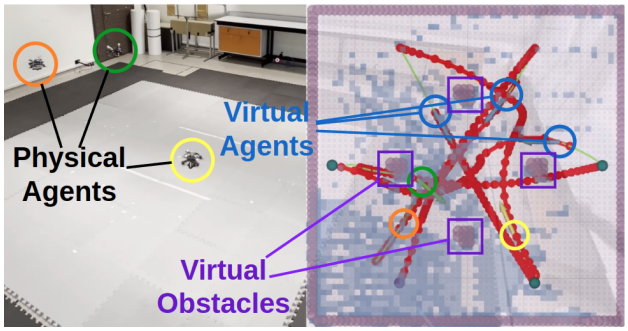


Fig. 6: Physical-Virtual Environment with 3 physical and 3 virtual drones.

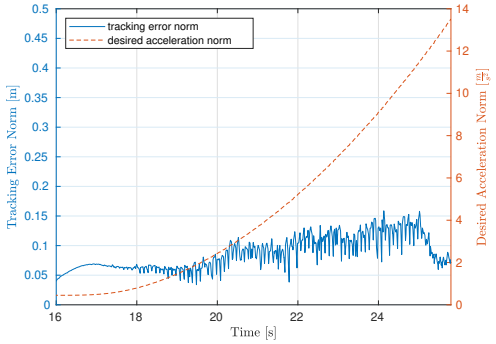


Fig. 7: Tracking performance of the NUSwarm drone with RPT control.

C. Hardware

As a lightweight framework, Gestelt is intended for resource-constrained systems, occupying less than 30% of all 4 CPU cores on the Radxa Zero single board computer, which features a quad-core 64-bit ARM processor and 4GB of memory. The NUSwarm drone (Fig. 5b) uses a 3D-printed frame, being optimized for cost and ease of maintenance, therefore serving as an ideal platform for swarm experiments. The total vehicle mass with batteries is under 200 g and it is capable of producing continuous accelerations of up to 8 Newtons with an average flight time of 7 minutes.

V. CONCLUSION AND FUTURE WORK

We have demonstrated the efficacy of our systematic approach in simulating a custom quadrotor platform given empirically determined physical parameters and deploying it for swarm navigation in a physical-virtual environment, which we have shown for a total of 3 physical and 3 virtual drones.

Additionally, tracking performance is crucial for swarm flight and we improve upon the default PX4 cascaded PID structure by implementing Robust Perfect Tracking (RPT) control.

In our work, we made use of simplified flight dynamics during simulations and ignored complex secondary aerodynamic effects such as blade flapping. Future work will focus on development of a high-fidelity simulator that generalizes well to high speed flight and scalability to higher number of simulated agents.

We anticipate that Gestelt will help to accelerate the deployment of swarm algorithms on custom quadrotor platforms through it's virtual environment approach and modular architecture.

VI. ACKNOWLEDGMENTS

We would like to thank Matthew Woo, Zheng Tian Ma and Derek Neo for their crucial role in the design and development of the NUSwarm drone.

REFERENCES

[1] Y. Zhou, B. Rao, and W. Wang, "Uav swarm intelligence: Recent advances and future trends," *IEEE Access*, vol. 8, pp. 183 856–183 878, 2020.

[2] P. Foehn, E. Kaufmann, A. Romero, R. Pěnička, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," 07 2023.

[3] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. Taylor, and V. Kumar, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, 12 2017.

[4] K. Xiao, S. Tan, G. Wang, X. An, X. Wang, and X. Wang, "Xtdrone: A customizable multi-rotor uavs simulation platform," 2020.

[5] M. Fernandez-Cortizas, M. Molina, P. Arias-Perez, R. Perez-Segui, D. Perez-Saura, and P. Campoy, "Aerostack2: A software framework for developing multi-robot aerial systems," 2023.

[6] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304.

[7] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multi-agent and dynamic environments," 2021.

[8] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, 05 2022.

[9] B. Şenbaşlar, W. Hoenig, and N. Ayanian, "Rlss: real-time, decentralized, cooperative, networkless multi-robot trajectory planning using linear spatial separations," *Autonomous Robots*, vol. 47, pp. 1–26, 05 2023.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, pp. 100–107, 1968. [Online]. Available: https://api.semanticscholar.org/CorpusID:206799161

[11] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:14744621

[12] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009.

[14] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abm6074

[15] B. Chen, Z. Lin, and K. Liu, "Robust and perfect tracking of discrete time systems," in *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, vol. 4, 2001, pp. 2594–2599 vol.4.

[16] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[17] B. M. Chen, *Robust and H_ Control*. Springer Science & Business Media, 2013.

[18] S. K. Phang, C. Cai, B. M. Chen, and T. H. Lee, "Design and mathematical modeling of a 4-standard-propeller (4sp) quadrotor," in *Proceedings of the 10th World Congress on Intelligent Control and Automation*, 2012, pp. 3270–3275.

[19] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *RotorS – A Modular Gazebo MAV Simulator Framework*, 01 2016, vol. 625, pp. 595–625.