Control of a Tail-sitter VTOL UAV Using Recurrent Neural Networks

Jinni Zhou[1], Hao Xu[1], Zexiang Li[1], Fu Zhang[2]

***Abstract*—Tail-sitter VTOL UAVs have the unique capability of both hovering and efficient level flight while retaining very simple mechanical structures. This paper presents a unified controller design for tail-sitter VTOL UAVs, using recurrent neural networks. A notable advantage of this control method is to treat the hover, transition, and level flight uniformly, regardless of the flight modes. The proposed controller consists of an outer-loop position controller and an inner-loop attitude controller. The outer-loop position controller consists of a PID linear controller, which computes the desired acceleration based on the position error, and a nonlinear solver, which computes the desired attitude based on the UAV dynamics and aerodynamics model. The inner-loop attitude controller consists of a proportional controller and a loop-shaping linear angular rate controller. To solve the long computation time problem of the nonlinear solver in the outer-loop position controller, a simple recurrent neural network (RNN) is utilized to approximate its behaviors. The proposed RNN, with proper dynamics excitation and data collection during the training, is able to run in real-time (i.e., 50 Hz) and approximate the nonlinear solver solution quite accurately. Moreover, the recurrent structure eliminates the sudden changes in control actions computed by the nonlinear solver, thereby mitigating the excitation of flexible modes of the UAV. System stability and robustness analyses are carried out, and the effectiveness of the proposed method is verified by comprehensive simulation and actual flight experiments.**

***Index Terms*—recurrent neural networks, nonlinear solver, flight control, unmanned aerial vehicle**

NOMENCLATURE

$\boldsymbol{p}$	Position vector represented in the Earth frame
$\boldsymbol{v}$	Aircraft speed vector represented in the Earth frame
m	Aircraft mass
$\boldsymbol{f}_t$	Force vector represented in the Earth frame
$\boldsymbol{R}$	Aircraft orientation w.r.t. the Earth frame
$\boldsymbol{\omega}$	Angular velocity represented in the body frame
$\boldsymbol{I}$	Inertia matrix of the aircraft
$\boldsymbol{\tau}$	Moment vector generated by the differential thrust
$\boldsymbol{m}_{aero}$	Aerodynamic moment represented in the body frame
$\boldsymbol{f}_{aero}$	Aerodynamic force represented in the body frame
$\boldsymbol{T}_i$	Thrust produced by the i-th propeller, i = 1, 2, 3, 4
$\boldsymbol{u}$	Airspeed vector represented in the body frame
$\boldsymbol{w}$	Wind speed vector represented in the Earth frame
U	Airspeed magnitude
α_x	Angle of attack
β	Sideslip angle
C_D	Drag coefficient
C_Y	Side force coefficient
C_L	Lift coefficient
C_l	Rolling coefficient
C_m	Pitching coefficient
C_n	Yawing coefficient
S	Reference area
a_{bx}	Aircraft acceleration along body X direction
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle

[1]Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (HKUST). email: {jinni.zhou, hao.xu, eezxli}@ust.hk.

[2]Department of Mechanical Engineering, University of Hong Kong. email:fuzhang@hku.hk.

I. INTRODUCTION

Nowadays, new designs and configurations of small-scale unmanned aerial vehicles (UAVs) are being developed to meet the higher requirements for professional fields such as parcel delivery, remote sensing, mapping, surveillance and dangerous missions. These applications need the UAVs to have the properties of both power efficiency and maneuverability. In many cases, suitable sites for taking off and landing are limited, so the UAV must have the ability to take off and land vertically. A simple way to design this kind of aircraft is to combine a quad-rotor and fixed-wing UAV, called a vertical takeoff and landing (VTOL) UAV. Among the mainstream VTOL UAVs [1], a tail-sitter VTOL UAV is the simplest because it does not need any additional actuators. Simple mechanisms are useful for small-scale UAVs because they can save manufacturing complexity and weight. Therefore, a large number of researchers have been attracted to study tail-sitter VTOL UAVs: in [2], a twin-engine design is presented; in [3], a structure with coaxial contra-rotating propellers is described; in [4], a quad-rotor configuration is developed, to name just a few. Generally, a tail-sitter VTOL UAV has two flight modes: hovering and level flight, and it switches between these two modes by tilting the pitch angle of the aircraft by 90 degrees. To avoid apparent altitude drop or gain, the control of transition flight should be stable and reliable. However, transition control is the most difficult part as it is usually aerodynamically unstable due to the well-known wing stalling phenomenon at a high angle of attack.

In this paper, a unified control method for a tail-sitter VTOL UAV is presented. Fig. 1 shows the tail-sitter aircraft used in this paper, which combines four rotors with a fixed-wing body. For details of the design and modeling process, readers can refer to [1] and [5]. The control method presented in this paper is unified, which means only one controller structure is used for all flight modes, regardless of whether it is hovering, transition or level flight. Compared to the other transition control methods, which will be discussed in the next section, the advantage of the control strategy presented in this paper is

Fig. 1: The tail-sitter VTOL UAV prototype used in this work

that it can control the aircraft's full flight dynamics during the transition process. As it does not distinguish between hover and level flight mode, the transition is continuous and smooth and eliminates significant altitude drop or gain.

The essential part of this control method is to solve the attitude and collective rotor acceleration problem to achieve the required acceleration provided by the position controller in real-time. For this purpose, a nonlinear optimization problem based on an accurate UAV dynamics and aerodynamics model is formulated and solved by a nonlinear sequential convex programming (SCP) solver. To address the problem of long computation time of the nonlinear solver, a novel idea is employed, inspired by the widely-used imitation learning techniques in recent years. The idea is to apply long term short term memory (LTSM) to learn the behavior of the SCP solver, and then generate a neural network called SCPNet to replace the SCP solver for real-time implementation. The computation time for SCPNet is quite short, within 5 ms on an Nvidia TX2 (an onboard computer used for real flights). Besides the low-computation cost, the proposed LSTM, with proper dynamics excitation and data collection during the training, can approximate the nonlinear solver solutions fairly accurately, as proved in the flight simulator [1]. After software-in-the-loop verification of SCPNet, we design several trajectories, including taking off, hovering, forward transition, level flight, backward transition, and landing, to demonstrate the effectiveness of this control method by real experiments in both indoor and outdoor environments.

This paper is organized as follows. In Section II, we discuss relevant literature. We introduce the system modeling of the tail-sitter UAV in Section III. The unified control method and imitation learning of the SCP solver is described in Section IV and V, respectively. Simulation and experiments results are analyzed in Section VI and VII, respectively. Finally, concluding remarks are provided in Section VIII.

II. Related Work

The first part of this section will present existing works related to flight control methods, and the second part of this section will focus on deep learning techniques for UAVs, and then propose a novel solution to solve the problem of long computation time using neural networks.

There is a variety of literature on the flight control of tail-sitter VTOL UAVs. In [6], a T-wing tail-sitter aircraft is developed, and an LQR controller is applied to control both the forward and vertical flight. The transition flight is controlled by the vertical controller, which receives the pitch angle commands. However, an open-loop scheme is used to specify the throttle, leading to an altitude drop or gain during the transition. In [7], although a databased off-line optimization algorithm is applied to minimize the transition time, it does not guarantee robustness and reliability when disturbances are present in the real operating environment.

Similarly, off-line optimization for transition control is presented in [8], [9], and in [10], Frank *et al*. use a set of trajectory points (relative to the pitch command) to realize the transition control; when the pitch angle is smaller than the set value, the hover controller will switch to the level flight controller. Three controllers are discussed on a Convair XFY-1 Pogo in [11]: a feedback linearization controller, vector-thrust model-based controller, and model reference adaptive controller (MRAC); however, all of these controllers are limited to two-dimensional flight dynamics, where only the longitudinal direction is controlled. More work on the tail-sitter aircraft's transition control can be found in [12] and [13], but most of the work is limited to two-dimensions, and only simulation results are presented.

In [12], the author presents two optimal problems for transition control, the objectives are minimum time and minimum energy, and the optimization problems are solved numerically and validated in simulation. Casau *et al*. [14] developed a hybrid control method for the transition of a model-scale fixed-wing aircraft, for which there is no hover flight mode. The contribution of this paper is to divide the flight envelope into four regions: hover, transition, recovery and level flight, and different control techniques are used for each region. The aerodynamic forces of a tail-sitter aircraft typically depend on the attitude (e.g., transition angle). To make the design of the controller easier, the author in [15] presents a transformation method to make the aerodynamic forces independent of the attitude. Changing the control input of thrust enables the computation of attitude and thrust separately, thus easing the control design and analysis of the transition process. Furthermore, the author in [13] investigates the stall effect and finds that the existence of control singularity and equilibrium bifurcation will cause some un-tractable maneuvers.

Despite the research, transition control of tail-sitter VTOL UAVs is still a challenge. Most of the existing control methods apply multiple control frameworks for different flight modes, such as using a hover controller for hovering, a level flight controller for level flight, and a transition controller for the transition between hovering and level flight. However, the switching between different flight modes requires much tuning and often leads to an unsatisfactory transient response. In this paper, we present a unified control framework to control a tail-sitter VTOL UAV, building on our previous work in [16]. In our prior work [16], the concept of unified control was proposed and proved in an ideal flight simulator. In this paper, we extend the results in [16] and prove the effectiveness of unified control method when all realistic flight characteristics,

such as motor delay, saturation, gyroscopic effects, etc, are considered. Stability and robustness analysis is carried out. Additionally, we propose a learning method to address the long computation time problem that occurred in our previous work.

Recent years have witnessed a rapidly growing trend of utilizing deep learning techniques for the control of UAVs. Travis *et al*. [17] present a nonlinear feedback controller for a quadrotor UAV using a neural network to learn the complete dynamics of the UAV in an online fashion. The effectiveness of this control scheme is demonstrated in the presence of unknown nonlinear dynamics and disturbances. In [18], an optimal controller design method using a neural network (NN) is proposed for a helicopter UAV. A single NN is utilized for the cost function approximation to achieve optimal trajectory tracking, and finally, the overall closed-loop stability and effectiveness of trajectory tracking are demonstrated. In [19], Giusti *et al*. train a deep network to determine actions that can keep a quadrotor on a trail by learning on single monocular images collected from the robot's perspective. Eight hours of video is captured using three GoPros mounted on the head of a hiker, with one pointing to the left, one to the right, and one directly forward. The optimal actions for the collected images can then be easily labeled; e.g., the quadrotor should turn right when facing an image collected from the left-facing camera. In [20], Jemin *et al*. present a method to control a quadrotor UAV with an NN trained using reinforcement learning techniques. The trained network can derive the mapping of states and actuator commands directly, making any predefined control structure obsolete for training. The performance of the trained policy is demonstrated both in simulation and experiments. Note the computation time of this NN is very short, only 7 μs per time step.

Although no prior works using a neural network to control tail-sitter VTOL UAVs have been found, the presented literature indicates the possibility of this application. Our task is to learn the behavior of the mentioned SCP solver, which can be formed as a typical imitation learning problem. Among the most widely-used imitation learning techniques, the task can be solved by behavior cloning. It solves problems in a supervised manner, by directly learning the mapping between the input observations and their corresponding actions, which are given by the expert policy. This formulation can give a satisfactory performance when there is sufficient training data [21].

III. System Modeling

A. Flight dynamics

In this paper, the coordinate systems of conventional fixed-wing aircraft are applied [22], [23]. Figure 2 shows the Earth frame $F_e(O_e, X_e, Y_e, Z_e)$ in the convention of north, east and down, and the body frame $F_b(CG, X_b, Y_b, Z_b)$ in terms of the front, right and down direction of the aircraft and $\boldsymbol{R} \in SO(3)$ is defined as the orientation of the body frame w.r.t. the Earth frame. Applying Newton's equations of motion to the airframe derives the following dynamic model:

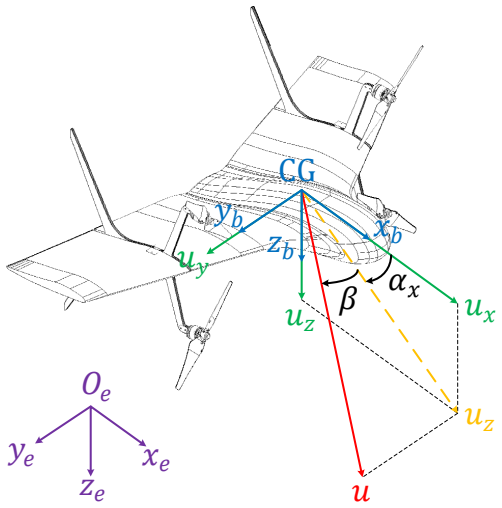

Fig. 2: Earth frame versus body frame

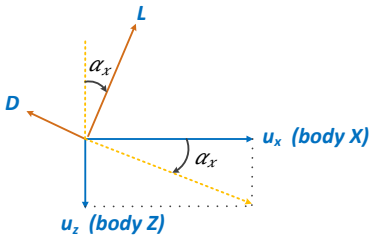

Fig. 3: Lift and drag

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{1a}$$
$$m\dot{\boldsymbol{v}} = \boldsymbol{f}_t \tag{1b}$$
$$\dot{\boldsymbol{R}} = \boldsymbol{R}\widehat{\boldsymbol{\omega}} \tag{1c}$$
$$\boldsymbol{I}\dot{\boldsymbol{\omega}} = -\boldsymbol{\omega} \times (\boldsymbol{I}\boldsymbol{\omega}) + \boldsymbol{\tau} + \boldsymbol{m}_{aero}, \tag{1d}$$

where $\boldsymbol{p} = [x, y, z]^T$ is the position of the aircraft's mass center expressed in the Earth frame, $\boldsymbol{v} = [v_x, v_y, v_z]^T$ is the aircraft's linear velocity expressed in the Earth frame, $\boldsymbol{f}_t$ is the total force applied to the aircraft, $\boldsymbol{\omega}$ denotes the angular rate of the aircraft expressed in the body frame, m is the total mass of the aircraft, $\boldsymbol{I}$ is the aircraft's inertia matrix, $\boldsymbol{\tau}$ denotes the moment vector generated by the differential thrust, and $\boldsymbol{m}_{aero}$ is the aerodynamic moment expressed in the body frame. $\widehat{\boldsymbol{\omega}}$ represents the skew-symmetric matrix, and $\widehat{\boldsymbol{\omega}}x = \boldsymbol{\omega} \times \boldsymbol{x}$ satisfies for any vector $\boldsymbol{x} \in \mathbb{R}^3$.

Forces applied on the vehicle $\boldsymbol{f}_t$ include gravity, propeller thrust and aerodynamic force. Denoting T_i as the thrust produced by i-th propeller, $i = 1, 2, 3, 4$, $\boldsymbol{f}_t$ can be expressed as

$$\boldsymbol{f}_t = \boldsymbol{R}\left(\sum \boldsymbol{R}_m^{(i)} \begin{bmatrix} T_i \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{f}_{aero}\right) + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}, \tag{2}$$

where $\boldsymbol{f}_{aero}$ is the aerodynamic force represented in the body frame, while $\boldsymbol{f}_t$ is in the earth frame. And $\boldsymbol{R}_m^{(i)}$ is the

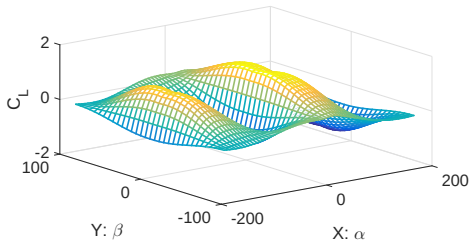

Fig. 4: Full envelope lift coefficient C_L

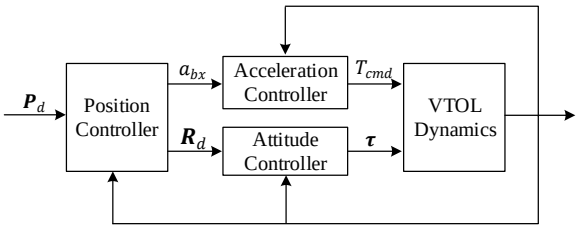

Fig. 5: Control structure

orientation of the i-th motor frame w.r.t. the body frame, where $i = 1, 2, 3, 4$.

B. Aerodynamic modeling

Beginning with the analysis of a tail-sitter aircraft's aerodynamics, the airspeed $\boldsymbol{u}$ should be defined as:

$$\boldsymbol{u} = \boldsymbol{R}^T(\boldsymbol{v} - \boldsymbol{w}), \tag{3}$$

where $\boldsymbol{v}$ denotes the linear velocity of the aircraft in Eq. (1a) and $\boldsymbol{w}$ refers to the wind speed, which can be estimated by wind estimation algorithms [24] or measured by sensors such as a pitot tube. The magnitude of airspeed $\boldsymbol{u}$, angle of attack α_x and angle of sideslip β are respectively expressed as

$$U = \sqrt{u_x^2 + u_y^2 + u_z^2} \tag{4a}$$

$$\alpha_x = \tan^{-1}\left(\frac{u_z}{u_x}\right) \tag{4b}$$

$$\beta = \sin^{-1}\left(\frac{u_y}{U}\right). \tag{4c}$$

The aerodynamic forces are expressed as $\boldsymbol{f}_{aero} = [X, Y, Z]^T$ and the aerodynamic moments are expressed as $\boldsymbol{m}_{aero} = [l, m, n]^T$. Generally, the aerodynamic forces are discussed in the stability axes frame [25] where lift L, drag D and side force Y are included. Fig. 3 shows that the lift is perpendicular to the airspeed, and drag is in the opposite direction of the airspeed. The right-handed frame of the drag, side force and lift follow the definitions

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -\cos\alpha_x & 0 & \sin\alpha_x \\ 0 & 1 & 0 \\ -\sin\alpha_x & 0 & -\cos\alpha_x \end{bmatrix} \begin{bmatrix} D \\ Y \\ L \end{bmatrix}, \tag{5}$$

and the lift L, drag D, side force Y and moments are

$$\begin{aligned} L &= C_L QS; \quad D = C_D QS; \quad Y = C_Y QS \\ l &= C_l QS\bar{c}; \quad m = C_m QS\bar{c}; \quad n = C_n QS\bar{c}, \end{aligned}$$

where C_L, C_D and C_Y denotes the coefficient of the lift, drag and side force, respectively, C_l, C_m and C_n represents the coefficient of the rolling, pitching, and yawing moment, respectively, $Q = \frac{1}{2}\rho U^2$ denotes the dynamic pressure, S denotes the area, and $\bar{c}$ denotes the mean aerodynamic chord (MAC). For a tail-sitter aircraft whose operating speed is far off the speed of sound, the coefficients depend on the attack angle α and angle of sideslip β, which can be derived from wind tunnel experiments. For compactness, we only show the full envelope of the lift in Fig. 4. The drag, side force coefficients, and the detailed wind tunnel experiments can be found in [1].

After the modeling of flight dynamics and aerodynamics, the controller design procedures will be shown in the next section.

IV. Unified Control Method

The control structure is shown in Fig. 5, which consists of an outer loop position controller providing the desired attitude and body X acceleration, and an inner attitude controller and acceleration controller to track the desired attitude and acceleration, respectively. Based on the dual-loop control structure presented in [16], an acceleration controller is added to generate the thrust command for the tail-sitter aircraft. This acceleration controller is added because the thrust cannot be controlled directly in a real system; while the four rotors generate the thrust, the value is proportional to the magnitude of the body X acceleration, and the direction is the same as the body X direction. Therefore, we include an acceleration controller to compute the thrust command, where the command value is in the range of 0 - 1, related to the position of the throttle.

In this section, the unified control framework will be introduced in detail. Here, the inner attitude controller will not be described, it can be referred to in our previous work, [16] and [26].

A. Underlying acceleration control

In this control structure, an acceleration control loop is introduced to track the body X acceleration a_{bx} computed by the position controller. Generally, there is an IMU in the avionics system of an aircraft, and the accelerometer in the IMU can directly measure the value of a_{bx}; therefore, the acceleration loop is reasonable for real implementation. Here, a simple PID controller is applied to track the body X acceleration command a_{bx} generated by the position controller, shown as the following equation:

$$T_{cmd} = K_{p1}a_e + K_{i1}\int a_e dt + K_{d1}\frac{da_e}{dt}, \tag{6}$$

where T_{cmd} is the total thrust command sent to the actuators, a_e is the tracking error of the body X acceleration, and K_{p1}, K_{i1} and K_{d1} is the proportional, integral and derivative gain, respectively, for the PID controller in the body X acceleration loop.

B. Unified position control based on the SCP solver

Combining Eq. (1a), Eq. (1b) and Eq. (2), the translational motion of the aircraft can be rewritten as

$$\begin{aligned} \dot{\boldsymbol{p}} &= \boldsymbol{v} && \text{(7a)} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e}_3 + \boldsymbol{R}\left(T\boldsymbol{e}_1 + \boldsymbol{f}_{aero}\right), && \text{(7b)} \end{aligned}$$

where T is the total thrust generated by the 4 propellers, $\boldsymbol{e}_1 = [1,0,0]^T$, and $\boldsymbol{e}_3 = [0,0,1]^T$. For a pre-set trajectory $\boldsymbol{p}_d(t)$, the position error can be computed as $\boldsymbol{e}_p = \boldsymbol{p}_d(t) - \boldsymbol{p}(t)$. Then, a PID controller is applied to compute the desired acceleration, with a term of the feed-forward acceleration:

$$\boldsymbol{a}_d = \ddot{\boldsymbol{p}}_d(t) + K_p\boldsymbol{e}_p + K_i\int \boldsymbol{e}_p dt + K_d\frac{d\boldsymbol{e}_p}{dt}, \tag{8}$$

where a_d is the desired acceleration computed by the PID controller, K_p, K_i and K_d is the proportional, integral and derivative gain, respectively, for the PID controller in the position loop.

To achieve the desired acceleration $\boldsymbol{a}_d$, the attitude $\boldsymbol{R}$ and the total thrust T in Eq. (7b) should satisfy the following equation, which is derived by dividing both sides of Eq. (7b) by m, and substituting $\boldsymbol{a}_d$ for $\dot{\boldsymbol{v}}$:

$$g\boldsymbol{e}_3 + \frac{1}{m}\boldsymbol{R}\left(T\boldsymbol{e}_1 + \boldsymbol{f}_{aero}\right) = \boldsymbol{a}_d. \tag{9}$$

As the thrust cannot be directly controlled, here, the body X acceleration a_{bx} is used to substitute for Eq. (9), and a new equation is derived as follows:

$$g\boldsymbol{e}_3 + \boldsymbol{R}\boldsymbol{e_1}a_{bx} + \frac{1}{m}\boldsymbol{R}\boldsymbol{f}_{aero} = \boldsymbol{a}_d. \tag{10}$$

In the controller design of conventional quad-rotors, the aerodynamic force $\boldsymbol{f}_{aero}$ is usually ignored, and the analytical solution of the attitude and body X acceleration in Eq. (10) can be derived [25] easily. However, a tail-sitter aircraft uses the aerodynamic force generated to provide lift during a forward flight to improve efficiency. Thus, the aerodynamic forces cannot be ignored when computing the attitude and body X acceleration.

As discussed in Section III, the aerodynamic forces are described in Eq. (5), which change with the angle of attack α, which is calculated from the airspeed $\boldsymbol{u}$. Therefore, the aerodynamic forces can be described as a general nonlinear function of the airspeed $\boldsymbol{u}$. Substituting $\boldsymbol{u}$ with Eq. (3), we derived the following equation:

$$\boldsymbol{f}_{aero} = f(\boldsymbol{R}^T(\boldsymbol{v} - \boldsymbol{w})). \tag{11}$$

Because of the nonlinearity of the aerodynamic force and attitude, it is too complicated to derive an analytical solution for Eq. (9). Therefore, in this work, numerical methods are applied to derive the attitude and body X acceleration. An optimization problem is formulated to derive proper attitude and body X acceleration for a given inertial acceleration. The optimization problem consists of the three equality constraints provided by Eq. (9), forming an objective function. The objective function attempts to align the vehicle's heading with the direction of velocity $\boldsymbol{v}$, which is consistent with the aerodynamic requirements and the actual flight experience. As a result, the optimization problem can be constructed as

$$\begin{aligned} \min_{[\boldsymbol{\xi},T]\in\mathbb{R}^4} \quad & |v_{by}| \\ \text{s.t.} \quad & g\boldsymbol{e}_3 + \boldsymbol{R}\boldsymbol{e_1}a_{bx} + \frac{1}{m}\boldsymbol{R}f(\boldsymbol{R}^T(\boldsymbol{v}-\boldsymbol{w})) - \boldsymbol{a}_d = 0 \\ & \boldsymbol{R} = \boldsymbol{R}_z\boldsymbol{R}_x\boldsymbol{R}_y \\ & a_{bx\min} \le a_{bx} \le a_{bx\max}, \end{aligned} \tag{12}$$

where v_{by} is the body velocity in the Y direction. In the objective function, v_{by} is formulated to be minimized as we want the velocity along the Y direction to be as small as possible, which is reasonable to flying operation experience. In this optimization problem, we parameterize the attitude $\boldsymbol{R}$ by the ZXY Euler angle representation as the singular point is rarely reached for tail-sitter aircraft: more specific reasons were mentioned previously in the modeling section. The desired body X acceleration a_{bx} should be bounded by its limits $a_{bx\min}$ and $a_{bx\max}$, which depend on the maximum and minimum thrust generated by the 4 motors. In the controller design, saturation conditions can also be considered. When a_{bx} is outside the range of $a_{bx\min}$ and $a_{bx\max}$, the integrator parameter will be set to zero.

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & g_0(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \le 0 \ \forall i \\ & h_j(\boldsymbol{x}) = 0 \ \forall j, \end{aligned} \tag{13a}$$

$$\min_{\boldsymbol{x}} \ g_0(\boldsymbol{x}) + \mu\sum_i |g_i(\boldsymbol{x})|^+ + \mu\sum_j |h_j(\boldsymbol{x})| = \min_{\boldsymbol{x}} \ f_\mu(\boldsymbol{x}). \tag{13b}$$

In this paper, an SCP algorithm is applied to solve the non-convex optimization problem in Eq. (12). It is an iterative method that solves non-convex optimization problems by iteratively solving a sequence of convex sub-problems, shown in Algorithm 1.

To solve a general non-convex optimization problem (i.e., Eq. (13a)), we first construct an unconstrained merit function (i.e., Eq. (13b)) from Eq. (13a), and set the values of the current point $\bar{\boldsymbol{x}}$ and 5 parameters, where μ represents the initial penalty coefficient, ϵ_0 represents the initial trust region size, α_1 represents the threshold of the improving ratio, β_1 represents the shrink ratio of the trust region, and t represents the coefficient increasing ratio. There are 3 loops in the algorithm: (1) in loop 1, we keep increasing the penalty coefficient for constraints and re-initializing the trust-region size until the absolute sum of the constraints is less than δ, or the penalty coefficient is bigger than μ_{max} (the solver fails because of too many iterations); (2) in loop 2, we use the first Taylor series expansion to generate a sequence of approximate convex sub-problems, and find the optimal value for f_μ; (3) in loop 3, we first call the convex program to get the next point $\bar{\boldsymbol{x}}_{next}$, then update the current point $\bar{\boldsymbol{x}}$ and the trust region by comparing the improve ratio with α_1.

This SCP algorithm is implemented using a MATLAB package, CVX, and the parameters are appropriately tuned for Eq. (12). The attitude and body X acceleration solved from

Algorithm 1 Sequential Convex Programming

Inputs: $\bar{\boldsymbol{x}}, \mu = 1, \epsilon_0, \alpha_1 \in (0.5, 1), \beta_1 \in (0, 1), t \in (1, \infty)$
while ($\sum_i |g_i \bar{\boldsymbol{x}}|^+ + \sum_j |h_j \bar{\boldsymbol{x}}| \geq \delta$ AND $\mu < \mu_{MAX}$) **do**
  $\mu \leftarrow t\mu, \epsilon \leftarrow \epsilon_0$
  while (1) **do**
    compute terms of first-order approximations:
    $g_0(\bar{\boldsymbol{x}}), \nabla_{\boldsymbol{x}} g_0(\bar{\boldsymbol{x}}), g_i(\bar{\boldsymbol{x}}), \nabla_{\boldsymbol{x}} g_i(\bar{\boldsymbol{x}}), h_j(\bar{\boldsymbol{x}}), \nabla_{\boldsymbol{x}} h_j(\bar{\boldsymbol{x}}), \forall i, j$
    while (1) **do**
      call convex program solver to solve:

$$(\bar{f_u}(\bar{\boldsymbol{x}}_{next}), \bar{\boldsymbol{x}}_{next}) = \min_{\boldsymbol{x}} \; g_0(\bar{\boldsymbol{x}}) + \nabla_{\boldsymbol{x}} g_0(\bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}}) + \mu \sum_i |g_i(\bar{\boldsymbol{x}}) + \nabla_{\boldsymbol{x}} g_i(\bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})|^+ + \mu \sum_j |h_j(\bar{\boldsymbol{x}}) + \nabla_{\boldsymbol{x}} h_j(\bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})|$$

$$\text{s.t.} \quad ||\boldsymbol{x} - \bar{\boldsymbol{x}}||_2 \leq \epsilon$$

      if $\frac{f_\mu(\bar{\boldsymbol{x}}) - f_u(\bar{\boldsymbol{x}}_{next})}{f_\mu(\bar{\boldsymbol{x}}) - \bar{f_u}(\bar{\boldsymbol{x}}_{next})} \geq \alpha 1$ **then**
        shrink trust region: $\epsilon \leftarrow \epsilon\beta_1$
      else
        update $\bar{\boldsymbol{x}} \leftarrow \bar{\boldsymbol{x}}_{next}$,
        grow trust region: $\epsilon \leftarrow \epsilon\beta_1$,
        and break out of while [3]
      end if
      if ϵ below some threshold **then**
        break out of while [3] and while [2]
      end if
    end while
  end while
end while

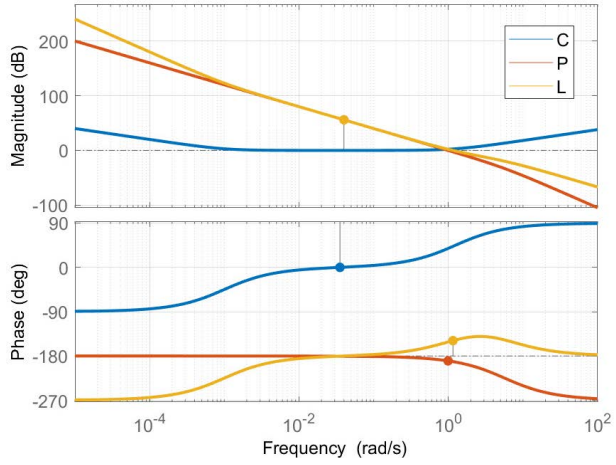

Fig. 6: Bode diagram of the aircraft's longitudinal loop with the SCP solver implemented

Eq. (12) will achieve the required acceleration a_d to eliminate the position error. Therefore, the vehicle can be uniformly controlled in hovering, transition and level flight, regardless of the flight modes.

C. Stability and robustness analysis

In the previous section, we assumed that the attitude command computed by the SCP solver can be achieved by the real plant instantaneously. However, there exists a coupling phenomenon between the translation and rotation behaviors of the aircraft.

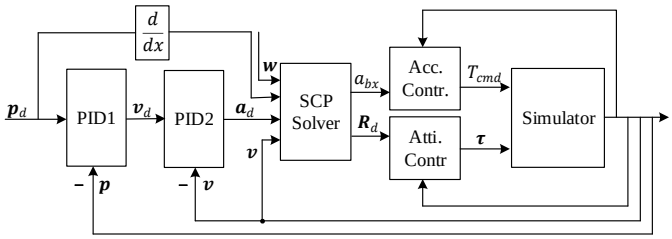

Fig. 7: Simulation structure

Firstly, the influence of translational dynamics on rotational dynamics is analyzed. The aircraft's linear velocity v affects the motor thrust, and at the same time, the differential motor thrust generates the moment τ. When the velocity v increases, the thrust of each motor decreases because of the higher advanced ratio. Fortunately, the differential motor thrust is independent of v due to the symmetrical allocation of the motors, as the same decrement will be caused by v for the thrust of each motor.

Secondly, the influence of rotational dynamics on translational dynamics is analyzed. For the design of position controllers, an assumption is usually made that the attitude command can be achieved instantaneously. However, there exists a transient response to the actual attitude because of the attitude controller's limited bandwidth and the inertia of the vehicle. To analyze the influence of the transient response, a bode plot of the aircraft's longitudinal direction is shown in Fig. 6, where P denotes the aircraft attitude dynamics, and C denotes the position controller. The plant P can be modeled as a first-order delay part with a delay time of 176 ms (derived by analyzing the attitude response), multiplied by a second-order integrator representing that the pitch angle causes an acceleration, and L denotes the open loop transfer function of the system and is defined as $L = PC$. To make a smaller impact of the transient response of the attitude loop on the position controller, the bandwidth of L should be set lower than the bandwidth of the attitude delay time (i.e., 5.7 rad/s). In this work, the bandwidth of L is set to 1.15 rad/s, so the influence of the attitude loop's transient response can be reasonably ignored.

In addition to the above analysis, the bode diagram in Fig. 6 can also be used to indicate the robustness of the position controller, where the lower and upper gain margin is -55.9 dB and $+\infty$, respectively, and the phase margin is 31.2 deg. This implies that the amount of gain and phase we can add in the loop before it goes unstable is (-55.9 dB, $+\infty$) and 31.2 deg, respectively.

D. Simulation results of the unified control method

In this section, several typical trajectories are tested to verify the unified control method. Fig. 7 is the simulation structure, where R_d and a_{bx} are solved by the SCP solver with given acceleration a_d. The simulator has considered a lot of realistic issues, such as the rotor gyroscopic effect, torque caused by

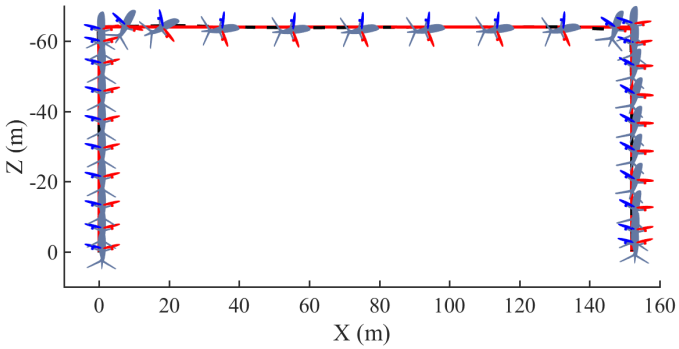

Fig. 8: Takeoff, transition, forward flight and landing

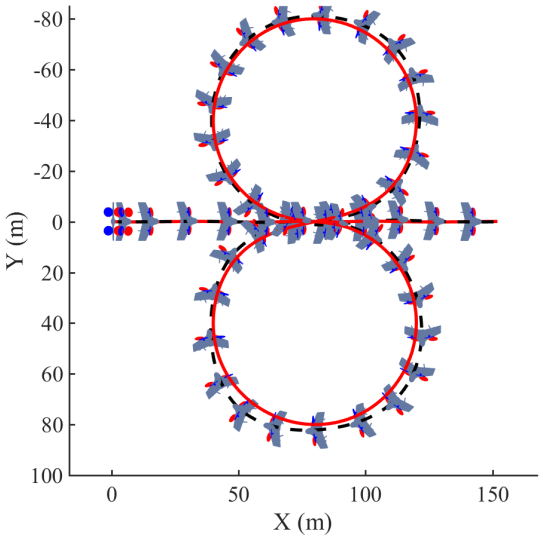

Fig. 9: Flight of "figure 8" trajectory in the XY plane

motor acceleration and deceleration, delay of motor response, saturation and propeller profile, etc., to make the simulation performance close to a real system. The detailed development of the simulator can be found in [1].

Due to the symmetrical structure of the tail-sitter aircraft, it is hard to distinguish the vehicle's back and front, so the color of the two rotors mounted on the vehicle's front is red, and the two rotors mounted in the vehicle's back are marked in blue in Fig. 8 and Fig. 9. Fig. 8 is an entire whole trajectory including takeoff, transition, forward flight and landing. The aircraft first takes off vertically and then transits its body to forward flight at a speed of 12 m/s, it then transits it is body back to hover attitude and lands vertically. In Fig. 8, the gray dashed line is the actual flight trajectory, and the solid red line is the desired trajectory. The results show good tracking performance, even for the transition process; the altitude drop or gain is less than 0.4 m, showing significant advantage over other control methods (mentioned in the introduction section). In [13], the author claims that because of the existence of equilibrium bifurcation caused by the big change of pitch angles in the stall region, the ideal transition from hover to forward flight is not reachable. However, no significant altitude gain/drop is observed in the transition process of our simulation, because we have restricted the vibration of the pitch angles to 9 degrees, and the acceleration in Eq. (8) could be non-zero in our algorithm, which is changed by the position controller in an online fashion. The above two reasons can avoid the equilibrium bifurcation described in [13]. Another interesting point is that during the landing process, the aircraft is tilted slightly. The reason behind this is that the lifting line of the aircraft is not precisely in the -180 degrees direction of the body frame.

In Fig. 9, a flight of a "figure 8" trajectory in the XY plane is shown. The aircraft first transits from hover to level flight at a speed of 12 m/s, then maintains the velocity to fly a 40 m radius circle trajectory in the upper half of the XY plane. It then flies a symmetrical circle in the lower half of the XY plane. After the completed "figure 8" trajectory, the aircraft changes back to level flight and then stops. From the figure, tracking errors are observed during the circling process, and this is because the PID parameters of the position controller are not well-tuned. This can be solved by other more systematic controller design methods.

E. Comparison with traditional control methods

As introduced in Section II, most of the existing control methods for VTOL aircraft apply a hierarchical control framework (i.e., multiple control frameworks) for different flight modes such as rotary-wing mode, fixed-wing mode, and transition mode. In our research work, we also applied this method on our tail-sitter platform, the details for the design can be found in [27]. Although we achieved the autonomous flight mission, which included takeoff, hovering, transition, level flight, and landing, the transition process was not sufficiently smooth and the tracking accuracy was not satisfactory. The main drawbacks for this hierarchical framework lie in the transition process. (1) For a forward transition from rotary-wing mode to fixed-wing mode, the aircraft executes the transition by sending a linearly decreasing pitch angle command to the altitude holding controller, while the roll and yaw angles are set to zero. Once the prescribed pitch angle and airspeed are both reached, the fixed-wing mode will be triggered. (2) Similarly, the vehicle in fixed-wing mode will execute the backward transition by calling the same altitude holding controller if a transition command is given. During the transition process, the aircraft only controls the altitude, leaving the X and Y directions uncontrolled. Even though the transition process only takes several seconds, it significantly affects the trajectory tracking accuracy when a disturbance (such as a gust wind) occurs. Therefore, we proposed a unified control framework in this paper to improve the aircraft's performance in terms of higher tracking accuracy, smoother transition process, and better disturbance resistance.

Here, we focus on the forward transition process to compare the performance between the hierarchical control framework and the unified control framework, and the results are shown in Table I. Table I shows that: (1) the transition process takes about 3 seconds for both the unified and hierarchical frameworks with a small effect caused by the disturbance; and (2) the performance of each of these two frameworks is

TABLE I: Transition process from hovering to level flight (i.e., pitch angle from 90 to 25 degrees)

Conditions	Metrics	Unified framework	Hierarchical framework
no disturbance	x distance (m)	15.46	17.12
	y max err (m)	0.32	0.61
	z max err (m)	0.01	0.03
	time consumed (s)	3.32	3.21
with disturbance (mean value: 0 variance: 5)	x distance (m)	15.25	20.74
	y max err (m)	1.4	7.12
	z max err (m)	0.57	1.39
	time consumed (s)	3.4	3.29

quite similar (the hierarchical framework has a slightly larger tracking error) with the condition of no disturbance; when there exists disturbance, the gap becomes very large, especially for the tracking error in the Y direction: the maximum error for the unified framework is 1.4 m, while that of the hierarchical framework is 7.12 m.

F. Conclusion

In this section, a robust unified controller method was presented for a tail-sitter aircraft, but the computation time of the SCP solver is 1 s on average and the maximum computation time will exceed 10 s in extreme states when there are large disturbances to the velocity. The position controller, on the other hand, is run on the frequency of 50 HZ and requires the solver converge within 0.02 s.

V. Imitation Learning of the SCP Solver

This section focuses on the pipeline to build SCPNet, where a neural network model is used to mimic the behavior of an SCP solver through imitation learning. We first briefly introduce the concept and algorithms of imitation learning, and later, the details of data collection, network structure, training, and testing for our experimental implementation are introduced.

A. Imitation learning

Unlike reinforcement learning where the policy is searched through a hand-designed reward, imitation learning is an alternative to learning control policies by providing the learning agent with experts' demonstrations [28]. It offers a paradigm for agents to learn successful policies in fields where people can easily demonstrate the desired behavior but find it difficult to hand program or hardcode the correct cost or reward function. This is especially useful for humanoid robots or manipulators with high degrees of freedom. For our SCP solver, the behavior of outputs according to different inputs can be easily demonstrated via the simulator. When the learning accuracy reaches a high level, the newly-generated SCPNet learning from the SCP solver can substitute the SCP solver for real-time implementation.

In this paper, we use a neural network model to learn the behavior of the SCP solver. Besides the current inputs, the solver also needs to consider historical states to compute the appropriate outputs. Therefore, we apply an RNN structure for learning, which targets sequential prediction problems [29]. These structures are widely used to tackle supervised problems involving learning a mapping from an input sequence to an output sequence.

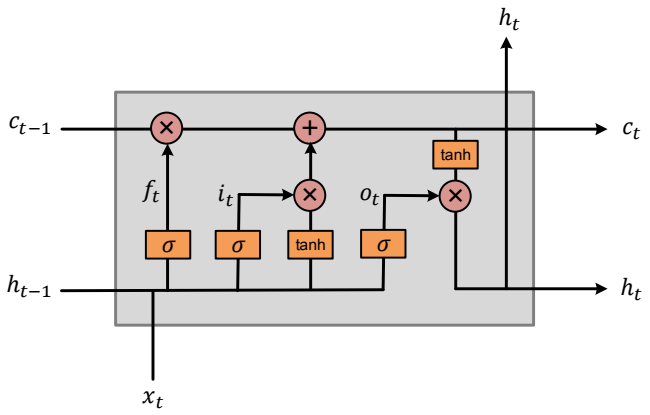

Fig. 10: Long short-term memory cell

Hochreiter & Schmidhuber [30] proposed a special type of RNNs called long short-term memory (LSTM), which applies purpose-built memory cells to store information, thus mitigating the long-term dependence problem of conventional RNNs. A single LSTM memory cell is shown in Fig. 10. In this paper, we use the standard version of LSTM [31], which is expressed by the following function:

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) && \text{(14a)}\\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) && \text{(14b)}\\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) && \text{(14c)}\\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) && \text{(14d)}\\
h_t &= o_t \tanh\left(c_t\right), && \text{(14e)}
\end{aligned}
$$

where σ is the sigmoid function, i, f, o and c is the *input gate*, *forget gate*, *output gate* and *cell* activation vectors, respectively, and all four vectors have the same length as the hidden vector h, and the W terms represent weight matrices (e.g., W_{ci} denotes the cell-gate weight matrix).

In our design, we use a sliding-window-like LSTM structure, where the current LSTM cell only depends on the last 5 historic cell outputs.

B. Data collection

To excite the possible states of the aircraft near specified trajectories such that the trained SCPNet is robust to disturbances, we add Gaussian white noise disturbance to the inputs of the SCP solver when collecting data for training. The noise is added to the actual inertial velocity v_i and the desired acceleration a_d, shown in Fig. 12. The input and output data of the SCP solver are generated through the predefined control framework in the closed-loop simulator, shown in Fig. 7, without the requirement of manual label work. It can be regarded as a self-supervised learning approach.

In this paper, we collect training data for the following three types of trajectories: takeoff, hovering and forward. Here the

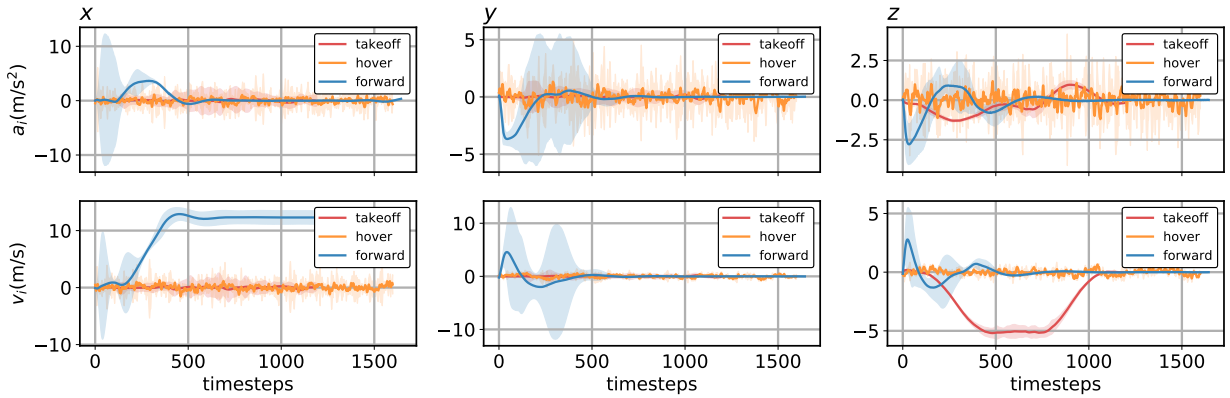

Fig. 11: Training data for the neural network: inputs of inertial acceleration (above) and inertial velocity (below)

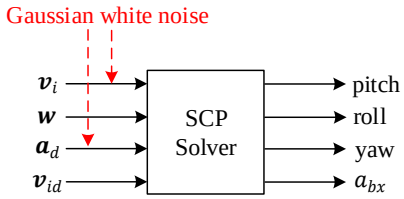

Fig. 12: SCP solver structure

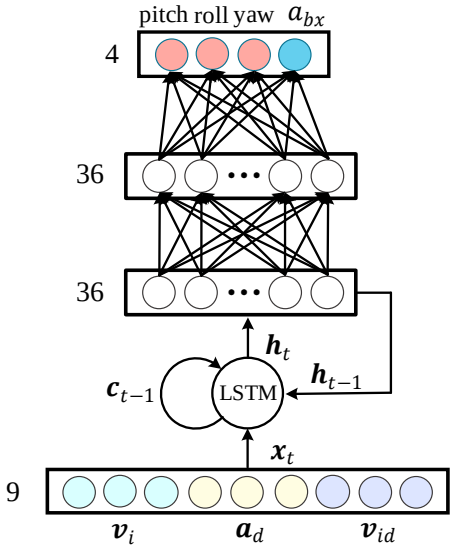

Fig. 13: SCPNet structure

forward trajectories include the transition process and forward flight. The training data has 400 sequences for each kind of trajectory, and the number of timesteps of the sequences for takeoff, hovering and forward trajectory is 1200, 1600, and 1650, respectively. Fig. 11 shows parts of the training data. The three figures in the top row show the distribution of the inertial accelerations along each of the axis X, Y, and Z, and the figures in the bottom row show the distribution of inertial velocities. Each color represents one kind of trajectory, the lines represent the mean values, and the color-filled areas represent the variance fluctuations near the mean. From the figures, we can find the inertial acceleration and velocity ranges for each direction, which are reasonable for real flights, e.g., the mean value for forward flight is 12 m/s, which is the optimal cruising speed for this VTOL aircraft. Additionally, the data for other inputs and outputs are also vibrated in some ranges, as the data collection process is conducted in a closed-loop way.

Finally, all the 1200 sequences are randomly divided into two parts: training sequences and test sequences, with a ratio of 9:1. As a result, a total number of 1080 sequences are used for training and 120 sequences are used for testing. The takeoff, hovering, and forward trajectories are randomly distributed in both the training and testing sequences.

C. Network structure

The network structure of SCPNet is shown in Figure 13. The inputs of this network have 9 dimensions, which are the current velocity, desired acceleration, and the desired velocity. They are directly fed to the LSTM layer, which outputs a vector of 36 dimensions, then, the LSTM outputs are fed to a fully connected (FC) network. The FC network has one ReLU hidden layer with a dimension of 36 and a regression output layer to predict the attitude (i.e., pitch, roll, and yaw) and the body X acceleration.

D. Network training

To train the SCPNet in Fig. 13, we use the mean squared error (MSE) between the ground truth outputs of the SCP solver and predictions of SCPNet as the cost function, and apply RMSprop [32] as the optimizer. For each iteration, the parameters for weight w and bias b are updated by the following equations:

$$w = w - \alpha \frac{g_w}{\sqrt{s_{g_w} + \epsilon}} \tag{15a}$$

$$b = b - \alpha \frac{g_b}{\sqrt{s_{g_b} + \epsilon}}, \tag{15b}$$

where α denotes learning rate, g_w and g_b is the gradient of training loss with respect to w and b, respectively. As s_{g_w} may be a small number close to 0, ϵ is added to ensure numerical

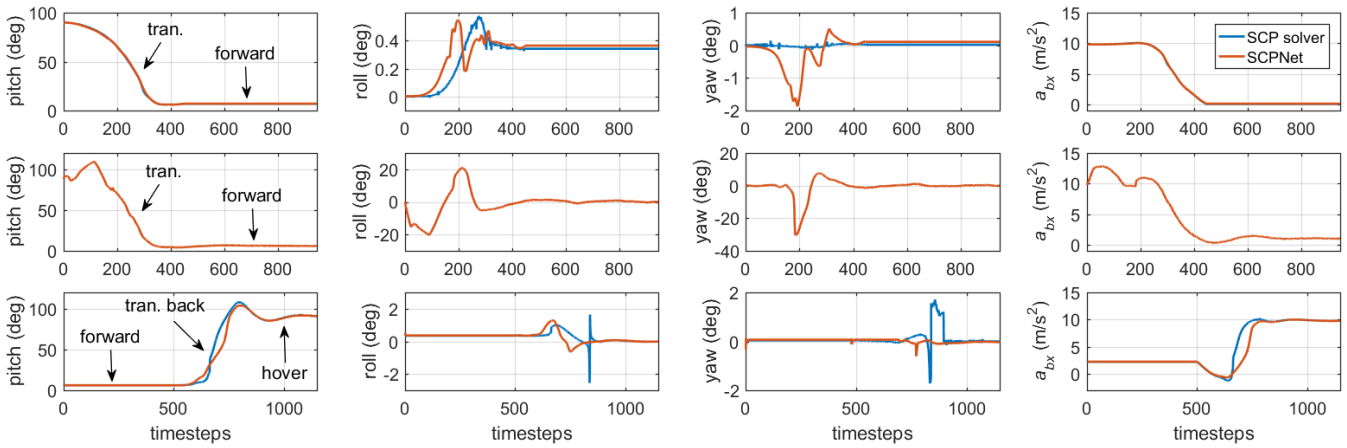

Fig. 14: Test results for SCPNet: transition and forward flight with no disturbance (above); transition and forward flight with disturbance (center); forward flight and transition back with no disturbance (below)

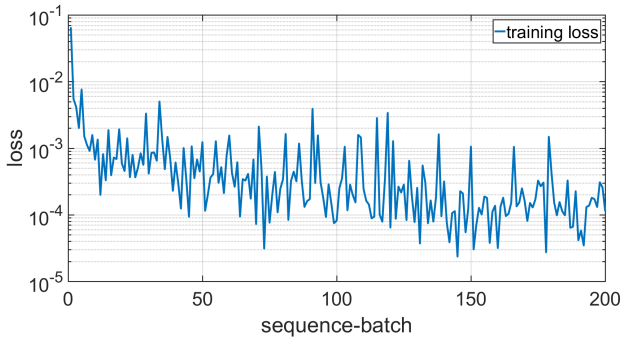

Fig. 15: Training loss

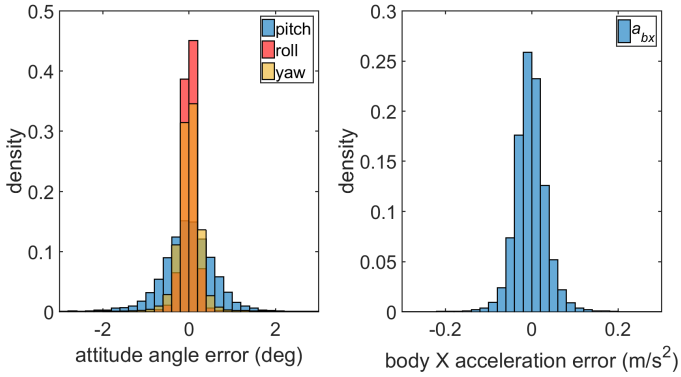

Fig. 16: Error distribution of attitude and body X acceleration

stability; generally it will be set to $1e^{-8}$. Additionally, s_{g_w} and s_{g_b} can be calculated by Eq. 16b and Eq. 16b:

$$s_{g_w} = \beta s_{g_w} + (1-\beta) g_w^2 \quad (16a)$$

$$s_{g_b} = \beta s_{g_b} + (1-\beta) g_b^2, \quad (16b)$$

where β is the value of the momentum, which takes the past gradients into account to smooth out the steps of the gradient descent. In [32], it is recommended to set β to 0.9, as it works very well for most applications, including our network (we tried different values of β between 0.9 to 0.99, and the results are basically consistent).

TABLE II: Computation time comparison between SCPNet and SCP solver (100,000 sets of testing data)

Metrics	SCPNet	SCP Solver
avg time (s)		
max time (s)		
time $>$ 0.02s (%)	0	

During the network training process, we first set the initial model weights following the XAVIER method [33]. We then deploy the training process by randomly sampling one from all the 1080 training sequences. For each iteration in the sampled sequence, we backpropagate the total loss of 30 timesteps of data and we update the model weights as in (15) and (16). As a result, a sequence consisting of 1200-1650 timesteps leads to 40-55 iterations. For every 10 sequences (a sequence-batch), the learning rate is updated by decreasing linearly from $1e^{-4}$ to $1e^{-8}$. This procedure repeats until the loss is sufficiently small, as shown in Fig. 16. The final training loss is quite small, at the level of $1e^{-4}$. We also notice that the training time is quite short, at only 2-3 hours.

E. Network testing

To evaluate the performance of the trained SCPNet, the testing data is applied to SCPNet. Fig. 16 shows the error distribution of the outputs, where the learning errors of pitch, roll, and yaw are mostly less than 2 degrees, and the errors of body X acceleration are less than 0.2 m/s^2. Table II shows the comparison of the computation time between SCPNet and SCP solver for the same 100,000 sets of testing data. It shows that ??

Additionally, we test the network using a variety of trajectories such as hovering, takeoff, transition, forward flight, transition back and landing. Although data of transition back and landing are not included in the training data, testing results for these two parts are quite satisfactory. These results indicate the trained SCPNet has gained sufficient robustness to be generalized to un-seen scenarios. To streamline the manuscript,

only the results for transition, forward flight and the transition back part will be presented.

Fig. 14 shows several testing results for SCPNet: the blue lines represent the outputs of the SCP solver, and the orange lines represent the outputs of SCPNet with the same inputs. The figures in the top row show the comparison of outputs between the SCP solver and SCPNet for transition and forward flight with no disturbance. We find the behavior of SCPNet is similar to that of the SCP solver; however, there always exist disturbances during real flight, such as wind. Therefore, the figures in the middle row present testing results for the same trajectory with disturbance. The behavior of the SCPNet outputs is almost the same as that of the SCP solver. To better verify the ability of SCPNet, we test some un-seen scenarios during the training, such as transition back from forward flight, and the landing process. As the landing process is quite simple, here we only illustrate the transition back process, shown in the figures in the bottom row of Fig. 14.

The learning accuracy is quite high, and even though no transition back data is involved in the training data set, the trained network has gained the main behavior of the SCP solver. Note that the behavior for SCPNet is smoother than for the SCP solver, especially for the timesteps between the transition back process and hovering. This is because the SCP solver only considers the current states and the last states for calculation of the outputs, while this neural network considers more historical states (the last states for five timesteps), thus improving the overall performance.

VI. Software-in-the-loop Verification

Before the real implementation of this control method, we first analyze the stability and robustness of this control framework, and then verify the effectiveness of this unified control method via a realistic simulator; the simulation structure is similar to that in Fig. 7. The main difference is that there are no wind inputs to SCPNet, because the accuracy of the onboard pitot tube is low. Therefore, we omit the wind inputs to SCPNet and regard the wind as a disturbance when in simulation and real flight experiments.

A. Stability and robustness analysis

To analyze the stability and robustness, we derive the linear model of the aircraft control system. The general way to derive the linear model is not to assemble the complete equations of the system, but to collect the longitudinal and lateral directional equations separately, as the longitudinal and lateral directional equations are proved to be decoupled [34].

Firstly, we analyze the aircraft's stability and robustness in the longitudinal direction. The simulation structure is shown in Fig. 18, where x_d and z_d are the position points of the desired trajectory, v_{xd} and v_{zd} are the desired inertial velocities calculated by the position controller, a_{xd} and a_{zd} are the desired accelerations calculated by the velocity controller, x and z are the actual position points, v_x and v_z are the actual inertial velocities, θ_d is the desired pitch angle of the aircraft, and τ_θ is the angular momentum of the pitch angle. Notice that the x and y in the above variables represent the X direction and the Y direction, respectively, in the Earth frame.

This simulator includes the following modules: (1) position controller and velocity controller, (2) acceleration controller and attitude controller, (3) SCPNet, and (4) plant. The details of each module are described below.

(1) & (2) The PID method is applied in the position controller, velocity controller, acceleration controller, and attitude controller, and the parameters of these controllers are consistent with those of controllers applied in real flights. For the details of the attitude controller design, readers can refer to our previous work [16].

(3) As mentioned in the previous section, SCPNet has a similar performance to the SCP solver, which can be represented by a combination of SCP solver and Gaussian noise, as shown in Fig. 18. To analyze the stability and robustness of the open loop, we linearize the SCP solver along the way points of a whole trajectory, which includes takeoff, transition, level flight and landing. Compared with Fig. 7, wind $\boldsymbol{w}$ and desired velocity $\boldsymbol{v}_d$ are omitted in the inputs to SCP solver. Here, wind is omitted and is regarded as disturbance as with in real flights because it cannot be measured accurately by the onboard pitot tube. The role of the desired trajectory is to determine whether the aircraft is in hovering mode or not, and if is, the initial yaw angle will be set to 0 to prevent the aircraft from misdirection. For a tail-sitter aircraft, the aerodynamic force can be ignored in hovering mode as it performs like a quadrotor, whose stability and robustness are well proved in extensive literature. Therefore, we omit the desired velocity for simplification, and express SCP solver for longitudinal direction as $g(v_x, v_z, a_{xd}, a_{zd})$, and the outputs are (a_{bx}, θ_d). Then the linearized SCP solver can be described by the following equation:

$$\begin{aligned} g_l((v_x, v_z, a_{xd}, a_{zd})) = g(v_{x0}, v_{z0}, a_{xd0}, a_{zd0}) + \\ \frac{\partial g(v_x, v_z, a_{xd}, a_{zd})}{\partial v_x}\Delta v_x + \frac{\partial g(v_x, v_z, a_{xd}, a_{zd})}{\partial v_z}\Delta v_z \\ + \frac{\partial g(v_x, v_z, a_{xd}, a_{zd})}{\partial a_{xd}}\Delta a_{xd} + \frac{\partial g(v_x, v_z, a_{xd}, a_{zd})}{\partial a_{zd}}\Delta a_{zd}. \end{aligned} \tag{17}$$

After the linearized SCP solver g_l is derived, we replace it with the original SCP solver, and it works well for the whole trajectory with small tracking errors. As SCPNet is a combination of SCP solver and Gaussian noise, and Gaussian noise only affects the errors of the outputs, we can directly apply the linearized SCP solver in the open-loop for stability and robustness analysis.

Additionally, to evaluate the influence of noise on the control accuracy, we first obtain the transfer functions from (a_{bx}, θ_d) to (x, z), in turn, which are shown in the following equations:

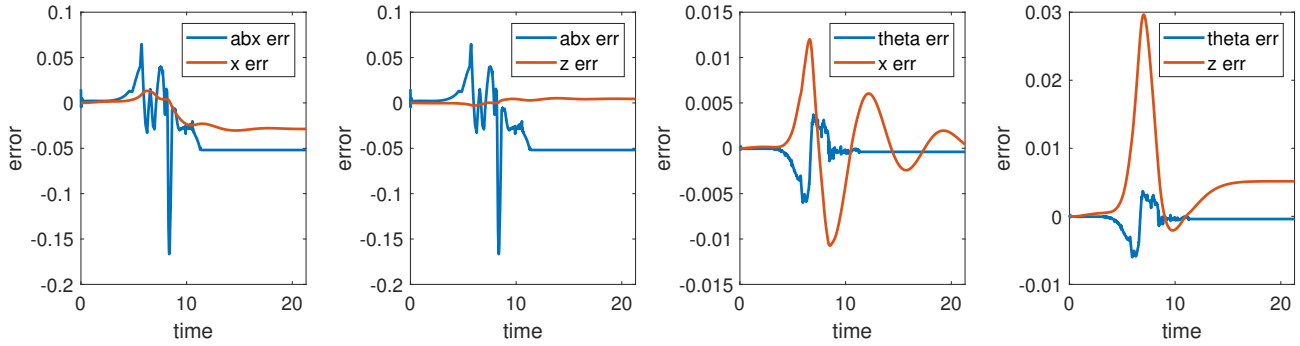

Fig. 17: Tracking error caused by noise

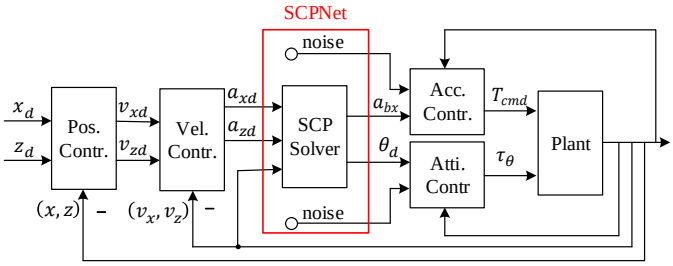

Fig. 18: Simulation structure in the longitudinal direction

$$T_{abx2x} = \frac{0.04489s + 0.6051}{s^2 + 0.6812s + 1.102} \tag{18a}$$

$$T_{abx2z} = \frac{-0.04589s - 0.1274}{s^2 + 0.6215s + 1.48} \tag{18b}$$

$$T_{theta2x} = \frac{-2.277s - 0.9678}{s^2 + 0.3733s + 0.8233} \tag{18c}$$

$$T_{theta2z} = \frac{0.4618s - 9.939}{s^2 + 1.602s + 0.7606}. \tag{18d}$$

Then, we derive the output errors (i.e., noise in Fig. 18) between SCPNet and the SCP solver for the same input points of a whole trajectory, and get the position tracking error caused by the noise, which is shown in Fig. 17. From the figure, we find the maximum error is 0.03m in the Z direction, caused by the noise on theta θ_d. And these errors are very small and therefore can be ignored in the actual flights.

(4) To derive the model of the plant, we conduct extensive model identification and wind tunnel experiments to collect data. Here, we divide it into two sub-models: angular momentum of the pitch angle τ_θ to pitch angle θ, and thrust command T_{cmd} to a_{bx}.

Firstly, the data collected at cruising speeds of 1 m/s - 10 m/s are used for frequency model identification from τ_θ to pitch angle rate q. (The details of the model identification method used in this paper can be found to in [26]). Fig. 19 shows the bode diagrams of the simulation model, and the dynamic results show consistency at different cruising speeds. The reason is that the wings of this tail-sitter have a small angular velocity aerodynamic damping coefficient C_{mq} .

As the normal flight speed is around 10 m/s, we select the frequency domain data at this speed for transfer function fitting, shown as the following equation:

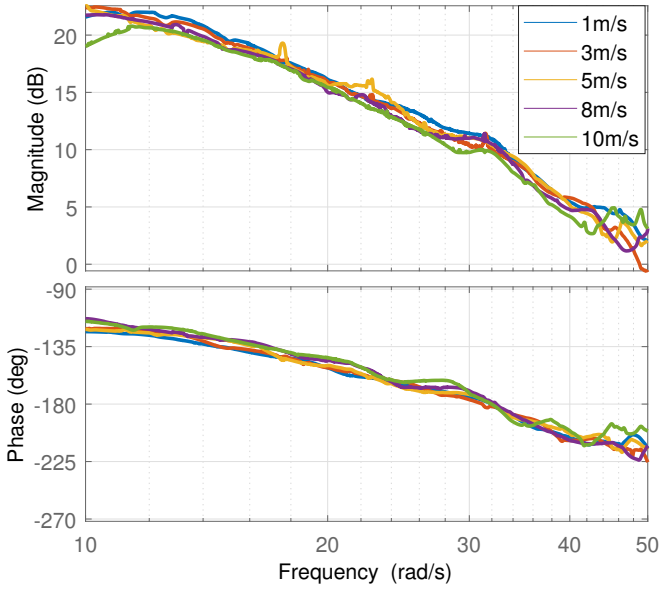

Fig. 19: Bode diagram of simulation model: τ_θ to q

$$tf_{tau} = \frac{-67.38s + 3105}{s^2 + 22.25s + 49.24}, \tag{19}$$

which is a second-order transfer function. Then we can get the model of τ_θ θ by adding an integrator.

Similarly, we derive the transfer function for thrust dynamics (i.e., thrust command T_{cmd} to thrust T) using the frequency data collected around the aircraft hovering state.

$$tf_{thr} = \frac{1}{0.076s + 1}. \tag{20}$$

Then, we model the aerodynamics of the aircraft based on the wind tunnel experimental data to obtain the force along the X direction [1], and finally obtained the body X acceleration a_{bx}.

Based on the above analysis and calculation, we build the aircraft simulator for the longitudinal direction in Simulink, and apply the frequency response estimation function to obtain the position x and z open-loop bode diagrams, shown in Fig. 20, this figure shows that both the x and z control loops are stable, and the gain margins are 45.9 dB and 44.3 dB, and the phase margins are 65.3 deg and 164 deg, respectively.

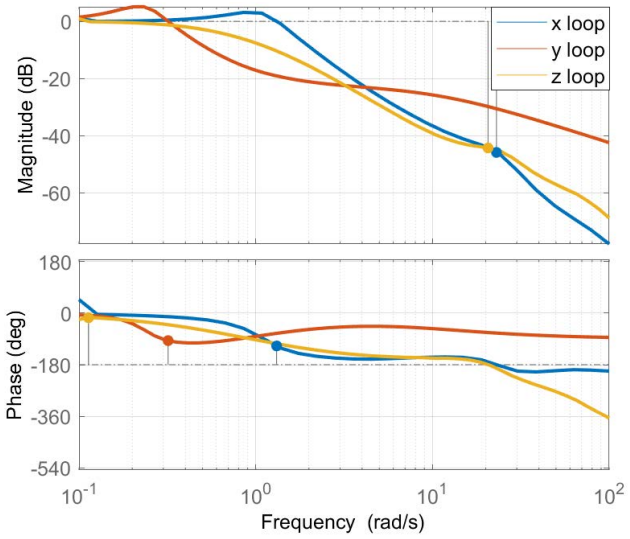

Fig. 20: Bode diagram of the aircraft's x, y, and z open loop with SCPNet implemented

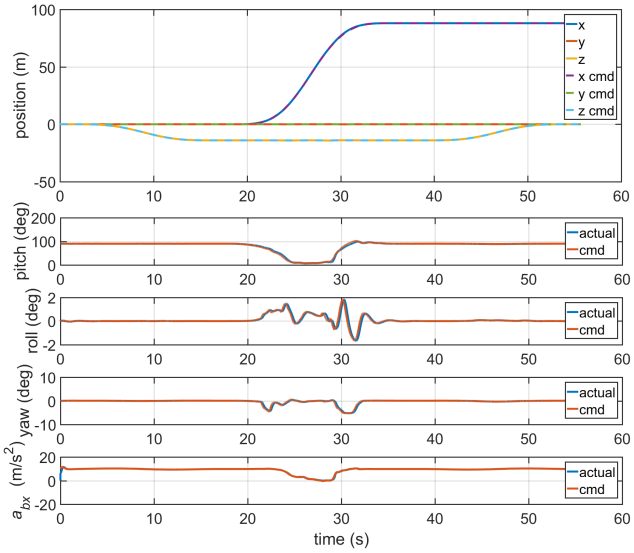

Fig. 21: Position (top) and attitude (bottom) response without disturbance for a whole trajectory consisting of taking off, hovering, forward transition, level flight, backward transition and landing

Similarly, We also establish the aircraft simulator for the lateral direction in Simulink, and analyze the stability and robustness via the Bode plot, shown in Fig. 20. It shows the aircraft's lateral direction is stable, the gain margin is $(-\infty, +\infty)$ dB, and the phase margin is 83.9 deg.

B. Verification for typical trajectories

After the stability and robustness are proved to meet the requirements, we also design several typical trajectories to analyze the performance both in no disturbance scenarios and in zero disturbance scenarios via the realistic simulator.

1) Trajectories with no disturbance: For the desired trajectory p_d including takeoff, transition, forward flight, transition

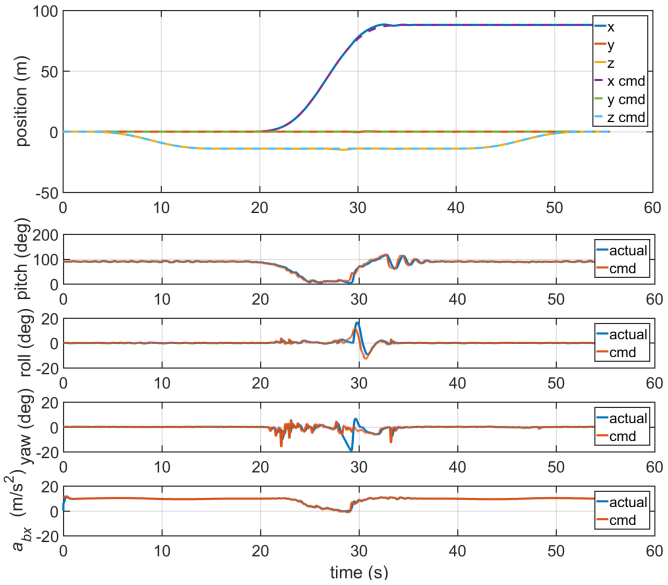

Fig. 22: Position (top) and attitude (bottom) response with disturbance for a whole trajectory consisting of taking off, hovering, forward transition, level flight, backward transition and landing

back and landing process, in Fig. 21, the top figure shows the position response, the solid line represents the actual position, and the dashed line represents the desired position. In this figure, the aircraft first takes off to the height of 14 m, transits to forward flight, transits back and stops at 88 m along the X direction, and finally, lands on the ground. The position is followed accurately as the curves representing the actual position and the desired position almost coincide with each other.

Additionally, the bottom figure shows the commands generated by SCPNet and the command execution conditions. At first, the command for pitch, roll, and yaw is 90, 0, and 0 degrees, respectively, and the command for body X acceleration (represented as a_{bx}) is 9.8 m/s^2, which represents the hover condition. During the takeoff stage, there are minor changes to the commands. For the transition stage, the pitch angle smoothly changes from 90 to 5 degrees, and a_{bx} command changes from 9.8 to almost 0 m/s^2. For the forward flight stage, the commands maintain stable values. Then comes the transition back and landing stage, which are the opposite form of transition and takeoff. During the whole process, the commands are reasonable and smooth with a vibration amplitude of less than 2 degrees, and the commands are executed well in the simulator.

2) Trajectories with disturbance: In the previous part, we verified the control method with no disturbance, in this part, we will analyze the performance when disturbances are presented. The verification with disturbance is important before implementation as wind usually exists and is regarded as a disturbance in real flights.

For the same entire whole trajectory analyzed in the previous subsection, we add Gaussian white noise as the disturbance to the input of $\boldsymbol{v}_i$ in Fig. 7, and the amplitude of this

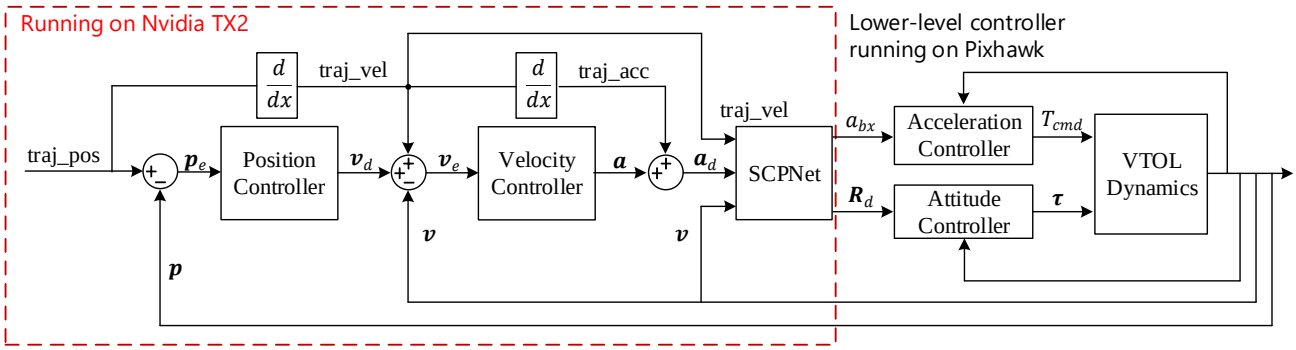

Fig. 23: Experiment structure

TABLE III: Position errors for typical scenarios (SCPNet in the simulation loop)

Trajectory	Metrics	Err	RMSE
whole trajectory* with no disturbance	x (m)	[-0.12, 0.45]	0.08
	y (m)	[-0.05, 0.03]	0.01
	z (m)	[-0.16, 0.11]	0.03
whole trajectory* with disturbance	x (m)	[-0.45, 1.78]	0.37
	y (m)	[-0.37, 0.32]	0.05
	z (m)	[-0.86, 0.15]	0.12
hovering with disturbance	x (m)	[-1.17, 1.87]	0.64
	y (m)	[-0.98, 1.97]	0.62
	z (m)	[-0.99, 1.86]	0.58

* includes taking off, hovering, forward transition, level flight, backward transition and landing

disturbance is from -3 to 3, which is a normal wind speed in the real experiment environment. Fig. 22 shows the position and attitude response for this scenario. Compared with Fig. 21, we find there exists a much larger attitude vibration, but the scale is within a controllable rage (within 20 degrees). As a result, the position commands are followed well with the presented disturbances. Table III shows the error ranges (Err) and root mean square error (RMSE) for each position in the x, y, and z directions. Here, three typical trajectories are analyzed. The position errors for the whole trajectory with disturbance are larger than those of the no disturbance trajectory, but the largest error is less than 1.78 m, with an RMSE of less than 0.37, which indicates the system has good performance with normal wind disturbance.

Moreover, we also verify the hovering scenario with disturbance. In this test, we use 100,000 timesteps of hovering trajectory points with a large random disturbance (amplitude from -6 to 6) added to $\boldsymbol{v}_i$. For this large disturbance, the position varies within a small range, less than 2 m, the details of which are shown in the last three columns of Table III.

VII. Experiment Results

After scientific verification of the control method with SCPNet, we conduct extensive experiments to demonstrate the strength of this framework. The control structure for the experiments is shown in Fig. 23. The position controller and SCPNet are run on an onboard computer, TX2, the outputs of the SCPNet are packaged as MAVlink messages, then the

Fig. 24: Environment of indoor experiments

messages are sent to the pixhawk and received by the attitude controller and body X acceleration controller separately. The frequency of the position controller is 50 HZ and attitude controller is 250 HZ.

This section is divided into two parts, indoor experiments, and outdoor experiments, a video of these experiments can be found at https://youtu.be/xm8CZagg6V8.

A. Indoor Experiments

For safety consideration, we first performed some indoor tests before conducting outdoor flight testing. Figure 24 shows the environment of the indoor experiments. Here, a motion capture system is used to provide the position information of the aircraft, and four industrial fans are utilized to generate an un-uniform wind field imitating wind in the outdoor environment. A handheld anemometer measures the wind speed in the hovering position of the aircraft, and the value is between 3.5 - 4.1 m/s, which is a common wind speed in Hong Kong.

Figure 25 shows the position and attitude performance of these hovering experiments. From the top figure, we find the original position of the aircraft is [0 0 0], then the aircraft starts to drift when the fans are turned on. After the fans are turned off, the aircraft returns to the original point. The bottom figure shows the attitude and body X acceleration performance of this hovering experiment where there exist some sudden changes

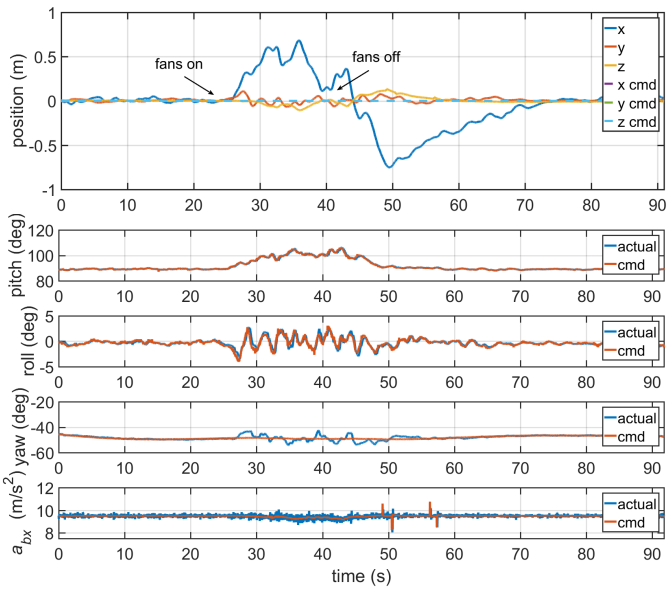

Fig. 25: Position and attitude response for a hovering trajectory with disturbance

in attitude due to the change of the wind speed, but it can transfer back to a stable attitude after the wind is stopped.

Table IV shows the error range (Err) and root mean square error (RMSE) for each of the position, attitude and body X acceleration: (1) the position errors are less than 0.75 m, and RMSE is less than 0.27 m; (2) the max attitude errors are less than 0.11 deg, and RMSE is less than 0.02 deg; and (3) the body X acceleration errors are less than 1.43 m/s^2, and RMSE is less than 0.13 m/s^2.

From this indoor experiment, we conclude that the position drift and attitude vibration are within a safe range when the wind speed is less than 4.1 m/s.

B. Outdoor Experiments

Finally, we conduct outdoor experiments when the wind speed is less than 4.1 m/s. This section will show the results of a whole trajectory including hovering, takeoff, forward flight, transition back and landing.

The aircraft first takes off and climbs to the height of 14 m. The speed first changes from 0 m/s to 2 m/s, then from 2 m/s to 0 m/s in the Z direction. After takeoff, the aircraft hovers for 2 seconds and then starts to transit from hovering to level flight. After it reaches the speed of 12 m/s, it maintains this speed for 4 seconds, then transits back to hovering, before at last landing on the ground.

Fig. 26 shows the position and attitude response for the whole trajectory experiment. The performance is satisfactory with small drifting errors. From the figure, we find the position commands are executed well and the attitude performance is smooth. The initial yaw performance looks strange, and this is because the aircraft is not precisely facing the North direction, while the initial yaw command is set to 0 degrees in the controller. Therefore, it has a process to turn the yaw angle back to 0 degrees.

TABLE IV: Position, attitude and body X acceleration errors for a hovering trajectory with disturbance

Metrics	Err	RMSE
x (m)	[-0.75, 0.68]	0.27
y (m)	[-0.06, 0.11]	0.02
z (m)	[-0.10, 0.13]	0.03
pitch (deg)	[-0.05, 0.03]	0.01
roll (deg)	[-0.03, 0.03]	0.01
yaw (deg)	[-0.09, 0.11]	0.02
$a_{bx}(m/s^2)$	[-1.39, 1.43]	0.13

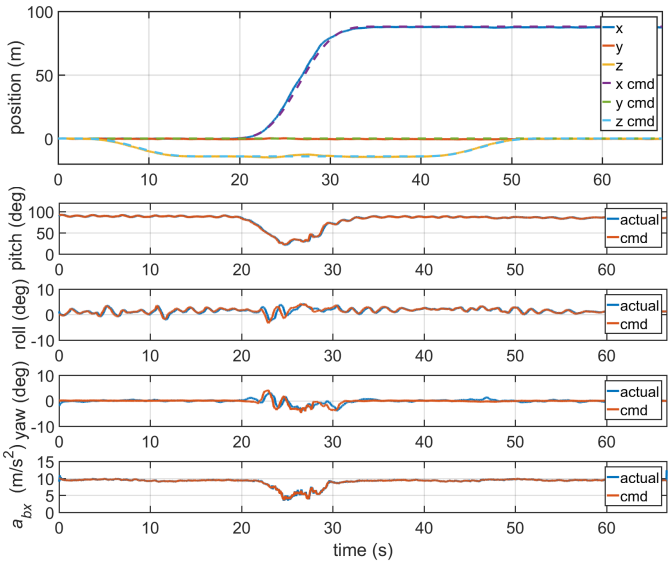

Fig. 26: Position and attitude response for a whole trajectory

Overall, the performance is satisfactory with small and reasonable tracking errors due to the presence of wind disturbance. The details are shown in Table V: (1) the position errors are less than 3.28 m, and RMSE is less than 0.81 m; (2) the max attitude errors are less than 0.07 deg, and RMSE is less than 0.02 deg; (3) the body X acceleration errors are less than 3.02 m/s^2, and RMSE is less than 0.17 m/s^2.

We find the trajectory tracking errors in real flight are larger than in the simulation, this is mainly caused by wind disturbance and the difference between the aircraft in simulator and the real aircraft. The performance can be improved by applying a more advanced controller method, such as the disturbance observer (DOB) design methodology [35].

TABLE V: Position, attitude and body X acceleration errors for a whole trajectory with disturbance

Metrics	Err	RMSE
x (m)	[-0.98, 3.28]	0.81
y (m)	[-0.68, 0.20]	0.37
z (m)	[-0.86, 1.40]	0.37
pitch (deg)	[-0.16, 0.07]	0.02
roll (deg)	[-0.06, 0.07]	0.01
yaw (deg)	[-0.05, 0.07]	0.01
$a_{bx}(m/s^2)$	[-1.03, 3.02]	0.17

VIII. Conclusions

A unified controller method is presented for controlling a quadrotor tail-sitter VTOL UAV. The controller consists of an attitude controller, a body X acceleration controller, and an optimization-based position controller. The position controller is capable of uniformly treating all the flight modes including taking off, hovering, forward transition, level flight, backward transition, and landing. It also significantly enlarges the flight envelope such that the aircraft can fly at different pitch angles and different forward speeds. Additionally, RNN LSTM is applied to solve the computation time problem of the SCP solver in the position controller. Using machine learning techniques to mimic the performance of an optimization problem is a novel method to minimize the gap between simulation and implementation; it provides a practical method for the real implementation of complex algorithms that have computation problems.

Acknowledgment

The authors would like to thank Lei Tai for his valuable discussions on neural-network-related techniques, also want to thank Xiaoyu Cai and Haowei Gu for their help with the experiments.

References

[1] F. Zhang, X. Lyu, Y. Wang, H. Gu, and Z. Li, "Modeling and flight control simulation of a quad rotor tail-sitter VTOL UAV," in *AIAA Modeling and Simulation Technologies Conference*, 2017, p. 1561.

[2] R. Stone and G. Clarke, "The t-wing: a VTOL UAV for defense and civilian applications," *University of Sydney*, 2001.

[3] D. Chu, J. Sprinkle, R. Randall, and S. Shkarayev, "Automatic control of vtol micro air vehicle during transition maneuver," in *AIAA Guidance, Navigation and Control Conference*, 2009, pp. 10–13.

[4] A. Oosedo, S. Abiko, A. Konno, T. Koizumi, T. Furui, and M. Uchiyama, "Development of a quad rotor tail-sitter VTOL UAV without control surfaces and experimental verification," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 317–322.

[5] X. Lyu, H. Gu, Y. Wang, Z. Li, S. Shen, and F. Zhang, "Design and implementation of a quadrotor tail-sitter VTOL UAV," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3924–3930.

[6] R. H. Stone, "Control architecture for a tail-sitter unmanned air vehicle," in *5th Asian Control Conference*, vol. 2. IEEE, 2004, pp. 736–744.

[7] R. H. Stone and G. Clarke, "Optimization of transition maneuvers for a tail-sitter unmanned air vehicle (UAV)," in *Australian International Aerospace Congress*, vol. 4. Citeseer, 2001.

[8] K. Kita, A. Konno, and M. Uchiyama, "Transition between level flight and hovering of a tail-sitter vertical takeoff and landing aerial robot," *Advanced Robotics*, vol. 24, no. 5-6, pp. 763–781, 2010.

[9] A. Oosedo, S. Abiko, A. Konno, and M. Uchiyama, "Optimal transition from hovering to level-flight of a quadrotor tail-sitter UAV," *Autonomous Robots*, vol. 41, no. 5, pp. 1143–1159, 2017.

[10] A. Frank, J. McGrew, M. Valenti, D. Levine, and J. How, "Hover, transition, and level flight control design for a single-propeller indoor airplane," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6318.

[11] N. Knoebel, S. Osborne, D. Snyder, T. Mclain, R. Beard, and A. Eldredge, "Preliminary modeling, control, and trajectory design for miniature autonomous tailsitters," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006, p. 6713.

[12] R. Naldi and L. Marconi, "Optimal transition maneuvers for a class of V/STOL aircraft," *Automatica*, vol. 47, no. 5, pp. 870–879, 2011.

[13] D. Pucci, "Flight dynamics and control in relation to stall," in *American Control Conference (ACC)*. IEEE, 2012, pp. 118–124.

[14] P. Casau, D. Cabecinhas, and C. Silvestre, "Hybrid control strategy for the autonomous transition flight of a fixed-wing aircraft," *IEEE Transactions on control systems technology*, vol. 21, no. 6, pp. 2194–2211, 2013.

[15] D. Pucci, T. Hamel, P. Morin, and C. Samson, "Nonlinear control of pvtol vehicles subjected to drag and lift," in *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*. IEEE, 2011, pp. 6177–6183.

[16] J. Zhou, X. Lyu, Z. Li, S. Shen, and F. Zhang, "A unified control method for quadrotor tail-sitter UAVs in all flight modes: Hover, transition, and level flight," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[17] T. Dierks and S. Jagannathan, "Output feedback control of a quadrotor UAV using neural networks," *IEEE transactions on neural networks*, vol. 21, no. 1, pp. 50–66, 2010.

[18] D. Nodland, H. Zargarzadeh, S. Jagannathan, *et al.*, "Neural network-based optimal adaptive output feedback control of a helicopter UAV," *IEEE transactions on neural networks and learning systems*, vol. 24, no. 7, pp. 1061–1073, 2013.

[19] A. Giusti, J. Guzzi, D. C. Ciresan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots." *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2016.

[20] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[21] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 2759–2764.

[22] R. C. Nelson, *Flight stability and automatic control*. WCB/McGraw Hill New York, 1998, vol. 2.

[23] B. Etkin and L. D. Reid, *Dynamics of flight*. Wiley New York, 1959.

[24] A. Cho, J. Kim, S. Lee, and C. Kee, "Wind estimation and airspeed calibration using a UAV with a single-antenna GPS receiver and pitot tube," *IEEE transactions on aerospace and electronic systems*, vol. 47, no. 1, pp. 109–117, 2011.

[25] T. Hamel, R. Mahony, R. Lozano, and J. Ostrowski, "Dynamic modelling and configuration stabilization for an x4-flyer." *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 217–222, 2002.

[26] J. Zhou, X. Lyu, X. Cai, Z. Li, S. Shen, and F. Zhang, "Frequency domain model identification and loop-shaping controller design for quadrotor tail-sitter VTOL UAVs," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018 (submitted).

[27] X. Lyu, H. Gu, J. Zhou, Z. Li, S. Shen, and F. Zhang, "A hierarchical control approach for a quadrotor tail-sitter VTOL UAV and experimental verification," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[28] J. A. Bagnell, "An invitation to imitation," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 2015.

[29] K. Nguyen, "Imitation learning with recurrent neural networks," *arXiv preprint arXiv:1607.05241*, 2016.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[32] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[34] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[35] X. Lyu, J. Zhou, H. Gu, Z. Li, S. Shen, and F. Zhang, "Disturbance observer based hovering control of quadrotor tail-sitter VTOL UAVs using H infinity synthesis," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2910–2917, 2018.