AIAA SciTech Forum
12-16 January 2026, Orlando, FL
AIAA SCITECH 2026 Forum

10.2514/6.2026-2867

Multidimensional Skill Training for Human Users via Shared Control

Hsin-Ying Lin*, Sooyung Byeon†, and Inseok Hwang‡
Purdue University, West Lafayette, IN, 47907

This paper proposes a multidimensional skill-conditioned shared control framework for training human users in complex dynamic control tasks. In human training, novices often learn through repeated practice and expert guidance. However, this expert-novice paradigm may not scale well in real-world settings due to the scarcity of expert instructors. The proposed framework serves as an expert, automatically evaluating user skill levels and providing an appropriate amount of assistance by fusing the autonomous control input with the user's input. Complex tasks naturally involve multiple distinct control behaviors that may develop at different learning rates. To accommodate this, we propose a skill identification method that decomposes a given task into multiple distinct sub-skills and develop a discriminator-based strategy for evaluating user proficiency at the sub-skill level. The proposed skill-conditioned shared control scheme dynamically adjusts the contribution of the autonomous input for each sub-skill based on the estimated proficiency of a human user. It increases the control authority of the autonomous input when proficiency is low to prevent user frustration, and reduces it as proficiency improves to prevent over-reliance on autonomy. We demonstrate the proposed framework on a set of multi-rotor Unmanned Aerial Vehicle (UAV) trajectory tracking tasks using simulated expert and novice agents. The proposed framework is compared against no shared control and a baseline single-dimensional shared control. Results show that the proposed framework can adaptively provide targeted assistance based on individual sub-skill levels and reduce the trajectory tracking error compared to these baselines.

I. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly used in both civilian and military applications. Despite advances in autonomy, human pilots are still required to take over control during brief but critical situations, such as emergency responses. Even for autonomous small UAV operations, the FAA (Federal Aviation Administration) guidance requires that a remote pilot in command retain the ability to immediately redirect or terminate the flight, ensuring human control during rare but safety-critical events [1–3]. This continued need for human involvement emphasizes the importance of training frameworks that can safely and efficiently improve pilot performance.

How can we efficiently train human pilots of UAVs? In human skill acquisition, humans often learn by practicing the skill repetitively. The expert-novice paradigm is commonly used to accelerate this process, where an expert demonstrates tasks and provides corrective assistance to a novice [4]. Since the ultimate goal is for the novice to learn the skill and perform the task independently, the expert should gradually reduce the assistance level as the novice improves to prevent over-reliance on the expert's support. This is done by decreasing the portion of control contributed by the expert. This model of learning is shown to be effective; however, it is difficult to scale due to the limited availability of expert instructors [5]. To address this, autonomous agents have been designed to act as synthetic experts to train novice human users [6, 7].

Accurate assessment of user skill proficiency is a fundamental aspect of human training. Traditional approaches often rely on simple metrics such as trajectory tracking error to quantify the deviation between novice and expert performance. However, these simple metrics fail to account for the inherent variability and stylistic adaptability of human behavior. To address this limitation, recent frameworks have adopted probabilistic methods that compare the statistical distribution of expert and novice trajectories [8]. This distributional approach is more robust to behavioral variance and better reflects underlying skill consistency. Alternatively, learning-based methods such as Inverse Reinforcement Learning (IRL) and Inverse Optimal Control (IOC) model human motor control as an optimization problem. The user

*Graduate Student, School of Aeronautics and Astronautics, lin1783@purdue.edu.
†Postdoctoral Fellow, School of Aeronautics and Astronautics, sbyeon@purdue.edu.
‡Paul Stanley Professor, School of Aeronautics and Astronautics, ihwang@purdue.edu, AIAA Associate Fellow.

skill level is then estimated by inferring the user's adherence to the expert's mission objective [9–11]. However, these methods typically yield a *single-dimensional* skill proficiency estimate, evaluating performance as a monolithic whole. In practice, complex tasks such as UAV navigation involve multiple distinct sub-skills that may develop at different learning rates. This limitation highlights the need for a training framework capable of *multidimensional* skill assessment and adaptive assistance.

Beyond assessing skill levels, an effective training framework must translate these assessments into adaptive guidance. To realize this, shared control provides a natural interface between human and autonomous agents, aligning with the expert–novice paradigm where the autonomous agent acts as the expert. Shared control is an architecture where the system is controlled by inputs from human and autonomous agents. It has been commonly used in domains such as robotic rehabilitation [9], driving assistance [12, 13], and multi-UAV teleoperation applications [14]. Prior work has proposed adaptive frameworks that adjust the level of assistance from the autonomous agent based on estimated user skill proficiency [8]. However, these approaches remain limited by their reliance on global, single-dimensional skill assessment. By providing single-dimensional assistance based on global performance evaluation, the current methods overlook the potential benefits of offering sub-skill-specific support. In this work, we design a training framework that can evaluate users' sub-skill proficiency and provide targeted assistance accordingly.

Providing targeted assistance requires a meaningful definition of what constitutes a "sub-skill". The meaning of sub-skill varies across research areas. In robotics, sub-skills are commonly referred to as primitive control behaviors learned independently by each end-effector, which can later be coordinated with other sub-skills to perform complex tasks [15]. In human skill acquisition, a complex skill is often decomposed into a set of sub-skills that can be trained and evaluated separately [16]. In this work, we define a sub-skill as a distinct control behavior required to perform a complex task. This definition provides a middle ground between actuator-level primitives in robotics and higher-level skill structures in human learning, while aligning with our framework by capturing recurring control behaviors. The number of sub-skills in a task may vary depending on the complexity of the system dynamics and whether the control inputs correspond to interpretable behaviors.

We propose a novel training framework that evaluates the user's proficiency in each sub-skill and provides sub-skill-specific assistance. Our work offers three main contributions. First, we systematically decompose the high-dimensional task of UAV navigation into multiple sub-skills. Second, we propose a framework that evaluates the proficiency of each sub-skill in real-time and provides sub-skill-specific assistance. Finally, we demonstrate the effectiveness of our proposed framework in a simulated UAV navigation task with various synthetic user profiles.

The remainder of the paper is organized as follows. Section II presents the proposed multidimensional shared control framework and the methods used for each component. Section III demonstrates and evaluates the proposed framework on a set of trajectory-tracking tasks, and Section IV reports the corresponding results and discussions. Finally, Section V provides the conclusions.

II. Methodology

A. Proposed Framework Overview

Figure 1 illustrates the proposed skill-conditioned shared control framework which consists of an offline training phase and an online execution phase. In the offline phase, we first decompose the expert demonstration into segments labeled by sub-skills via the skill identification process. These labeled samples are then used to train the skill classifier that maps state-control pairs to sub-skill labels. Using labeled expert and novice demonstrations, we train the skill evaluation module that compares novice behavior against expert behavior for each identified sub-skill. In the online phase, incoming novice state–control data are processed by the trained skill classifier to identify the active sub-skill. The skill evaluation module then estimates the user's proficiency level for that sub-skill. The resulting proficiency score determines the control mode of the skill-conditioned shared controller. Each mode corresponds to a distinct level of assistance: high, medium, or low. This discrete mode design is motivated by prior findings [17] showing that continuous variation of assistance level can induce mode confusion and degrade user experience. Based on the assigned mode, the framework blends the human and autonomous control inputs. The resulting shared input is then applied to the system.

B. Skill Identification

To enable sub-skill-specific assistance for human users, it is essential to first decompose the UAV navigation task into distinct sub-skills. Existing approaches for skill identification generally fall into three categories: supervised

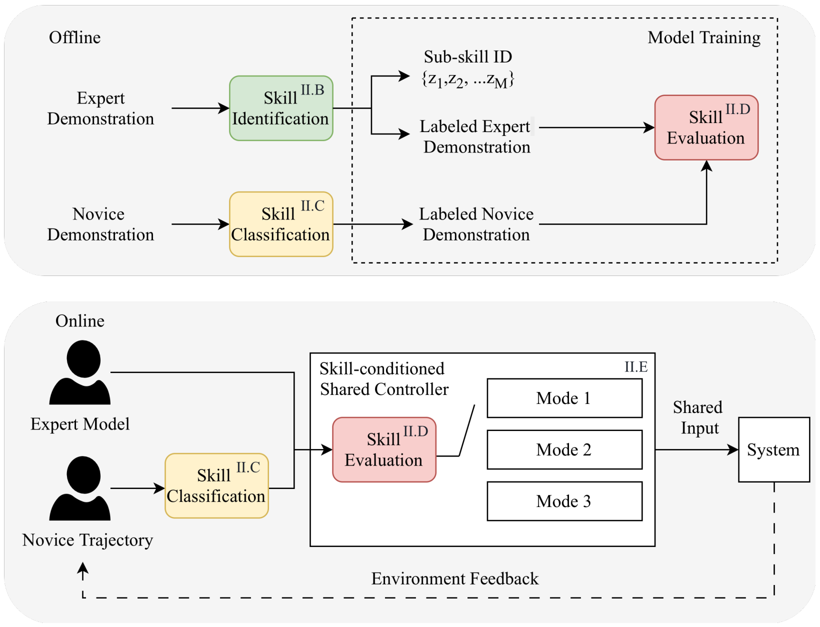

Offline
Model Training
Expert Demonstration
Skill Identification II.B
Sub-skill ID {z1,z2, ...zM}
Labeled Expert Demonstration
Skill Evaluation II.D
Novice Demonstration
Skill Classification II.C
Labeled Novice Demonstration
Online
Expert Model
Novice Trajectory
Skill Classification II.C
Skill-conditioned Shared Controller II.E
Skill Evaluation II.D
Mode 1
Mode 2
Mode 3
Shared Input
System
Environment Feedback

Fig. 1 The proposed framework for skill-conditioned shared control.

classification [18, 19], unsupervised learning [20, 21], and clustering-based segmentation [22, 23]. Supervised classification requires clean, pre-segmented skill trajectories, which in turn demands additional setup and expert-labeled data collection. On the other hand, unsupervised learning approaches typically require large amounts of data, which may be impractical in human-in-the-loop training settings. In this work, we employ the Hidden Semi-Markov Model (HSMM) [24] to segment expert demonstrations into interpretable sub-skills. Unlike conventional clustering algorithms such as Gaussian Mixture Models (GMMs), the HSMM explicitly models the temporal structure and duration of each sub-skill, enabling it to capture repeated and time-varying control patterns more accurately.

Let the expert demonstration be represented as $\mathcal{D}_E = \{(\mathbf{x}_k^E, \mathbf{u}_k^E)\}_{k=1}^{T^E}$ where $\mathbf{x}_k^E \in \mathbb{R}^n$ denotes the system state, $\mathbf{u}_k^E \in \mathbb{R}^m$ the control input at time step k, and T^E is the length of the expert demonstration. The superscript E indicates that the data are generated by an expert. For notational simplicity, we drop the superscript E and write $(\mathbf{x}_k, \mathbf{u}_k)$ when the context is clear. The sequence is modeled as a realization of the HSMM with a predefined number of discrete latent sub-skill state $z_k \in \mathcal{Z} = \{1, \ldots, M\}$ and a corresponding duration variable $d_k \in \mathbb{N}^+$. M denotes the number of sub-skills. We group consecutive time steps with the same sub-skill into segments and use s to index these segments. The sub-skill of segment s is denoted as $z_s \in \mathcal{Z}$, with duration $d_s \in \mathbb{N}^+$ and start time index τ_s. Segment s covers the indices $k = \tau_s, \ldots, \tau_s + d_s - 1$, and $z_k = z_s$ for all k in this range. Each sub-skill z_s persists for d_s time steps before transitioning to a subsequent sub-skill z_{s+1} according to a transition probability matrix $\mathcal{T} \in \mathbb{R}^{M \times M}$.

The generative process of the HSMM follows the finite Bayesian construction and can be written as:

$$\begin{aligned} z_1 &\sim \pi_0 \\ d_s \mid z_s &\sim p(d_s \mid z_s) \\ (\mathbf{x}_{\tau_s:\tau_s+d_s-1}, \mathbf{u}_{\tau_s:\tau_s+d_s-1}) \mid z_s &\sim \prod_{k=\tau_s}^{\tau_s+d_s-1} p(\mathbf{x}_k, \mathbf{u}_k \mid z_s) \\ z_{s+1} \mid z_s &\sim \pi_{z_s} \end{aligned} \tag{1}$$

where $p(\cdot \mid \cdot)$ denotes a probability mass function (pmf) or a probability density function (pdf). The first equation in Eq. (1) samples the initial sub-skill z_1 from the initial distribution π_0. The second equation samples the dwell time d_s for sub-skill z_s from the duration distribution $p(d_s \mid z_s)$. The third equation generates the observations $(\mathbf{x}_k, \mathbf{u}_k)$ in segment s using the emission density $p(\mathbf{x}_k, \mathbf{u}_k \mid z_s)$ (see Eq. (2)). The last equation samples the next sub-skill z_{s+1} from the transition distribution π_{z_s}, which is the z_s-th row of the transition matrix $\mathcal{T}$.

The emission model for each sub-skill is defined as a Gaussian distribution:

$$p(\mathbf{x}, \mathbf{u} \mid z_s) = \mathcal{N}([\mathbf{x}, \mathbf{u}]^\top \mid \mu_{z_s}, \Sigma_{z_s}) \tag{2}$$

where $\mu_{z_s} \in \mathbb{R}^{n+m}$ is the mean vector and $\Sigma_{z_s} \in \mathbb{R}^{(n+m)\times(n+m)}$ is the covariance matrix associated with sub-skill z_s. The duration distribution $p(d_s \mid z_s)$ is modeled as a Poisson distribution to capture variable dwell times [25]. The model parameters are denoted as $\Theta = \{\pi_0, \mathcal{T}, \{\mu_i, \Sigma_i, p(d_i)\}_{i=1}^{M}\}$, and are estimated using blocked Gibbs sampling [24]. Given the learned parameters, the most probable sub-skill sequence is obtained using the Viterbi algorithm [26]:

$$z^*_{1:T} = \arg\max_{z_{1:T}} p(z_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{u}_{1:T}, \Theta) \tag{3}$$

This produces a labeled expert dataset $\mathcal{D}'_E = \{(\mathbf{x}^E_k, \mathbf{u}^E_k, z^E_k)\}_{k=1}^{T^E}$, where each time step k is assigned a sub-skill label corresponding to an interpretable control behavior. Note that the HSMM is trained on expert demonstrations to learn the temporal and control structure of sub-skills. These learned parameters are then used to infer the sub-skill sequence for novice demonstrations, producing the labeled novice dataset $\mathcal{D}'_N = \{(\mathbf{x}^N_k, \mathbf{u}^N_k, z^N_k)\}_{k=1}^{T^N}$.

C. Skill Classification

The purpose of the skill classification module is to enable real-time identification of sub-skills from incoming novice data. While the HSMM described in Section II.B provides sub-skill labels for expert demonstrations offline, the HSMM inference is computationally expensive and unsuitable for online deployment. Therefore, we train a discriminative model that can efficiently map state–control pairs to sub-skill labels.

Given the labeled expert and novice datasets $\mathcal{D}'_E$ and $\mathcal{D}'_N$, respectively, we construct finite sequences of length L using a sliding window as: $\mathbf{w}_k = \{(\mathbf{x}_{k-l}, \mathbf{u}_{k-l})\}_{l=0}^{L-1}$. We define a classifier parameterized by Ψ that models the categorical distribution over sub-skill labels given the recent window as:

$$C_\Psi : \mathbf{w}_k \to p_\Psi(z \mid \mathbf{w}_k) \tag{4}$$

where $p_\Psi(z \mid \mathbf{w}_k)$ is a probability distribution over all sub-skill labels $z \in \mathcal{Z}$, and $z_k \in \mathcal{Z}$ is the HSMM-derived sub-skill label at time step k used as the training target for window $\mathbf{w}_k$. The classifier C_Ψ is implemented using a Long Short-Term Memory (LSTM) network [27], which captures temporal dependencies over the input window. The classifier parameters Ψ are optimized by minimizing the categorical cross-entropy loss over the HSMM-labeled dataset. During online inference, we maintain a sliding window of the most recent L state–control pairs. For each window $\mathbf{w}_k$, the classifier outputs the probability vector $p_\Psi(z \mid \mathbf{w}_k)$ over all sub-skill labels, and selects the most probable label as the current sub-skill estimate:

$$\hat{z}_k = \arg\max_{z\in\mathcal{Z}} p_\Psi(z \mid \mathbf{w}_k) \tag{5}$$

providing real-time sub-skill estimates for the subsequent skill evaluation and shared control modules.

D. Skill Evaluation

Building on the expert–novice learning paradigm, novice skill evaluation can be formulated as a comparison between novice behavior and expert demonstrations. Given the labeled expert and novice datasets $\mathcal{D}'_E$ and $\mathcal{D}'_N$, we define the skill evaluation function $\mathcal{F}$ that maps the current novice state–control pair and its sub-skill label to a normalized score:

$$\mathcal{F} : (\mathbf{x}^N_k, \mathbf{u}^N_k, z_k) \to s_k \tag{6}$$

where the superscript N denotes novice behavior, and $z_k \in \mathcal{Z}$ is the sub-skill at time step k. The score $s_k \in [0, 1]$ measures how closely the novice behavior aligns with the expert performance under sub-skill z_k, with higher values indicating better alignment. Inspired by adversarial imitation learning [28], we formulate the skill evaluation function $\mathcal{F}$ as a discriminator D_Φ. In robotics, the discriminator's role is to distinguish between real samples and generated samples. Similarly, the objective of the skill evaluation module is to distinguish human expert and novice behaviors within the context of a specific sub-skill z.

This can be formulated as minimizing the following adversarial objective, based on binary cross-entropy:

$$\mathcal{L}_I(\Phi) = -\mathbb{E}_{(\mathbf{x},\mathbf{u},z)\sim\mathcal{D}'_E}\left[\log D_\Phi(\mathbf{x}, \mathbf{u} \mid z)\right] - \mathbb{E}_{(\mathbf{x},\mathbf{u},z)\sim\mathcal{D}'_N}\left[\log\left(1 - D_\Phi(\mathbf{x}, \mathbf{u} \mid z)\right)\right] \tag{7}$$

Here $(\mathbf{x}, \mathbf{u}, z) \sim \mathcal{D}'_E$ and $(\mathbf{x}, \mathbf{u}, z) \sim \mathcal{D}'_N$ denote sampling state–control–skill tuple from the expert and novice datasets, respectively. The first term is the expected negative log-likelihood of the discriminator output on expert samples, and it penalizes assigning low scores to expert data. The second term is the expected negative log-likelihood of one minus the discriminator output on novice samples, and it penalizes assigning high scores to novice data. To make the discriminator skill-aware and to prevent the network from ignoring the condition, we include the Condition-Aware Loss ($\mathcal{L}_{CA}$) [29]. This is achieved by penalizing the discriminator for misclassifying mismatched pairs, where an expert sample is paired with an incorrect skill label z_{mis}:

$$\mathcal{L}_{CA}(\Phi) = -\mathbb{E}_{(\mathbf{x},\mathbf{u})\sim\mathcal{D}'_E,\ z_{\text{mis}}\sim\mathcal{Z}}\left[\log\left(1 - D_\Phi(\mathbf{x}, \mathbf{u} \mid z_{\text{mis}})\right)\right] \tag{8}$$

Here $(\mathbf{x}, \mathbf{u}) \sim \mathcal{D}'_E$ means the state–control pair is sampled from the expert dataset, and $z_{\text{mis}} \sim \mathcal{Z}$ means the label is drawn from $\mathcal{Z}$ but does not match the true sub-skill. This term is the expected negative log-likelihood of one minus the discriminator output on such mismatched pairs. It penalizes the discriminator for assigning high scores to mismatched tuples and forces the model to learn the conditioning variable z. To prevent the discriminator from overfitting on the limited expert data and encourage it to learn more general features of each skill, we introduce a weight decay regularization term $\mathcal{L}_{WD}$, which penalizes the squared ℓ_2-norm of the discriminator's weights:

$$\mathcal{L}_{WD}(\Phi) = \|\Phi\|_2^2 \tag{9}$$

Finally, to stabilize adversarial training, we incorporate a gradient penalty term $\mathcal{L}_{GP}$, which discourages overly sharp decision boundaries by penalizing gradients with respect to the expert inputs:

$$\mathcal{L}_{GP}(\Phi) = \mathbb{E}_{(\mathbf{x},\mathbf{u},z)\sim\mathcal{D}'_E}\left[\left(\|\nabla_{(\mathbf{x},\mathbf{u})} D_\Phi(\mathbf{x}, \mathbf{u} \mid z)\|_2 - 1\right)^2\right] \tag{10}$$

where $\nabla_{(\mathbf{x},\mathbf{u})} D_\Phi(\mathbf{x}, \mathbf{u} \mid z)$ is the gradient of the discriminator output with respect to its input. The final objective for the discriminator is a weighted combination of these losses, with ω balancing the contribution of each term:

$$\mathcal{L}_D(\Phi) = \omega_i \mathcal{L}_I(\Phi) + \omega_{ca}\mathcal{L}_{CA}(\Phi) + \omega_{wd}\mathcal{L}_{WD}(\Phi) + \omega_{gp}\mathcal{L}_{GP}(\Phi) \tag{11}$$

Once trained by minimizing this objective, the discriminator's output $s_k = D_\Phi(\mathbf{x}_k, \mathbf{u}_k \mid z_k)$ serves as the normalized skill evaluation score.

E. Shared Control for Assistance

We consider a 6 degree-of-freedom multirotor UAV modeled as a general nonlinear discrete-time system, with its full dynamics detailed in Appendix. At time step k, the full state is $\mathbf{x}_k = [\mathbf{p}_k^\top, \dot{\mathbf{p}}_k^\top, \boldsymbol{\phi}_k^\top, \boldsymbol{\omega}_k^\top]^\top$ with position, velocity, Euler angle, and angular velocity. The control input is $\mathbf{u}_k = [\phi_{d,k}, \theta_{d,k}, \dot{\psi}_{d,k}, T_k]^\top$, corresponding to commanded roll, pitch, yaw rate, and thrust. We model the system control input at time step k as a convex combination of inputs from the human user and the autonomous agent (i.e., UAV controller):

$$\mathbf{u}_k = \alpha_k \cdot \mathbf{u}_k^h + (1 - \alpha_k) \cdot \mathbf{u}_k^a \tag{12}$$

where $\mathbf{u}_k^h$ and $\mathbf{u}_k^a$ are the control inputs from the human and the autonomous agent, respectively. The scalar $\alpha_k \in [0, 1]$ denotes the control authority, with $\alpha_k = 1$ corresponding to full manual control and $\alpha_k = 0$ to full autonomy.

In order to obtain the autonomous control input for the shared controller, we need an assistance policy that produces expert-like inputs. We represent this policy as a state-feedback controller. Given the expert trajectory distribution $\mathbf{p}_k^E \sim \mathcal{N}(\mu_k^E, \Sigma_k^E)$, the controller minimizes the deviation between the current state and the expert trajectory while penalizing control effort. The trajectory tracking cost is formulated as:

$$c_k = (\mathbf{p}_k - \mu_k^E)^\top (\Sigma_k^E)^{-1} (\mathbf{p}_k - \mu_k^E) + \mathbf{U}_k^\top R \mathbf{U}_k \tag{13}$$

where $\mathbf{U}_k = \ddot{\mathbf{p}}_k$ is the translational acceleration. Note that the control input command $\mathbf{u}_k$ can be mapped to the translational acceleration $\mathbf{U}_k$ through the nonlinear dynamic mapping function $\mathcal{U}(\cdot)$ defined in Eq. (24) from Appendix. The first term measures the Mahalanobis distance to the expert mean, accounting for variability in expert demonstrations. The second term penalizes the control effort with weighting matrix $R = \rho I$, $\rho > 0$. To ensure physically feasible and safe control commands, we also impose translational acceleration constraints $A^u \mathbf{U}_k \leq b^u$, where A^u represents the inequality acceleration constraints and b^u the corresponding bounds.

The optimization problem is solved using a finite-horizon *Model Predictive Control* (MPC) framework [30]. We use the translational state $\boldsymbol{\zeta}_k = [\mathbf{p}_k^\top, \dot{\mathbf{p}}_k^\top]^\top$, and the system dynamics is modeled as:

$$\boldsymbol{\zeta}_{k+1} = A\boldsymbol{\zeta}_k + B\mathbf{U}_k \tag{14}$$

where the system matrices $A = \begin{bmatrix} I & I\Delta t \\ 0 & I \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ I\Delta t \end{bmatrix}$, with I denotes the identity matrix and Δt the sampling time. Let the concatenated state vector over the prediction horizon N_p be defined as $\boldsymbol{\zeta}_{[1:N_p]} = \left[\boldsymbol{\zeta}_1^\top, \boldsymbol{\zeta}_2^\top, \ldots, \boldsymbol{\zeta}_{N_p}^\top\right]^\top$, the future states over the prediction horizon are obtained by propagating the system dynamics:

$$\boldsymbol{\zeta}_{[1:N_p]} = S^x\boldsymbol{\zeta}_1 + S^u\tilde{\mathbf{U}} \tag{15}$$

where the matrices S^x and S^u are constructed from the discrete-time system matrices A and B as [31]:

$$S^x = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{N_p-1} \end{bmatrix}, \quad S^u = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-2}B & A^{N_p-3}B & \cdots & B \end{bmatrix}, \quad \tilde{\mathbf{U}} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \mathbf{U}_3 \\ \vdots \\ \mathbf{U}_{N_p} \end{bmatrix} \tag{16}$$

The cost function over N_p can be written as:

$$C = (\tilde{\mu}_{N_p}^E - S^x\boldsymbol{\zeta}_1 - S^u\tilde{\mathbf{U}})^\top(\tilde{\Sigma}^E)^{-1}(\tilde{\mu}_{N_p}^E - S^x\boldsymbol{\zeta}_1 - S^u\tilde{\mathbf{U}}) + \tilde{\mathbf{U}}^\top\tilde{R}\tilde{\mathbf{U}} \tag{17}$$

where

$$\tilde{\mu}^E = \left[(\mu_1^E)^\top, (\mu_2^E)^\top, \ldots, (\mu_{N_p}^E)^\top\right]^\top, \quad \tilde{\Sigma}^E = \mathrm{diag}(\Sigma_1^E, \ldots, \Sigma_{N_p}^E), \quad \tilde{R} = \mathrm{diag}(R, \ldots, R) \tag{18}$$

Substituting the predicted states into the tracking cost yields a quadratic objective over the control sequence, which can be expressed as a standard *Quadratic Programming (QP)* problem [8, 32]:

$$\begin{aligned} \min_{\tilde{\mathbf{U}}} \quad & \tfrac{1}{2}\tilde{\mathbf{U}}^\top H\tilde{\mathbf{U}} + g^\top\tilde{\mathbf{U}} \\ \text{s.t.} \quad & \zeta_{k+1} = A\zeta_k + B\mathbf{U}_k \\ & A^u\tilde{\mathbf{U}} \leq b^u \end{aligned} \tag{19}$$

where

$$\begin{aligned} &H = 2(W^\top W + \tilde{R}), \quad W = LS^u, \quad L = \mathrm{diag}(L_1, \cdots, L_l), \\ &g = -2\nu^\top W, \quad \nu = L(\tilde{\mu}_{N_p}^E - S^x\zeta_1), \quad (\hat{\Sigma}_k^E)^{-1} = L_k^\top L_k \end{aligned} \tag{20}$$

The resulting optimal acceleration $\mathbf{U}_k^a$ is mapped to the UAV's physical control input through the inverse of the dynamic mapping function as:

$$\mathbf{u}_k^a = \mathcal{U}^{-1}(\mathbf{U}_k^a \mid \phi_k) \tag{21}$$

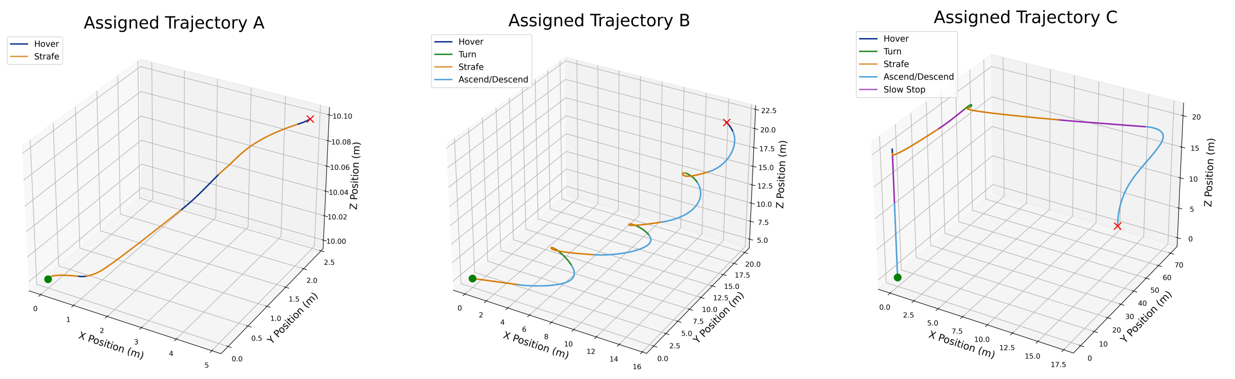
where ϕ_k represents the UAV's attitude and relevant dynamic parameters. This mapping ensures that the assistive command remains directly executable by the UAV.

A key element of any shared control scheme is determining the control authority α_k at each time step k. To ensure intuitive human interaction, our framework uses a fixed set of control authority values $\alpha_k \in \mathcal{A} = \{0.1, 0.5, 1.0\}$. This design choice is supported by prior studies [17], which have shown that continuously varying control authority can cause mode confusion in the human operator. We categorize user sub-skill proficiency into three categories: low, medium, and high, based on the output of the skill evaluation function $\mathcal{F}$. Specifically, we define low skill as $s_k \in [0, 0.3]$, medium skill as $s_k \in (0.3, 0.7]$, and high skill as $s_k \in (0.7, 1.0]$. These thresholds were set to capture meaningful distinctions in user proficiency and were chosen based on the distribution of trajectory tracking performance and corresponding skill scores. The control authority for each sub-skill is determined by mapping the sub-skill proficiency to a value in the set $\mathcal{A}$: low skill $\rightarrow \alpha_k = 0.1$, medium skill $\rightarrow \alpha_k = 0.5$, and high skill $\rightarrow \alpha_k = 1.0$. For example, a pilot scoring 0.9 would be classified as highly skilled. In this case, the proposed framework assigns control authority $\alpha_k = 1.0$, granting full control authority to the user.

III. Numerical Simulation

A. Design

The objective of the numerical simulation is to demonstrate that the proposed framework can provide sub-skill-specific assistance by adjusting support based on the user's estimated proficiency. The proposed framework is demonstrated using a simulated multi-rotor UAV training tasked with tracking three predefined 3D trajectories, as illustrated in Fig. 2. Each trajectory is designed to include segments that correspond to different sub-skills, with the green dot and the red cross representing the start and end points, respectively. To evaluate performance across varying levels of difficulty, the trajectories follow a complexity progression. Trajectory A (Fig. 2a) represents a simple *Obstacle Avoidance* task requiring only basic stability. Trajectory B (Fig. 2b) is a *Helical Flight* task that serves as an intermediate task introducing multi-axis coordination. Trajectory C (Fig. 2c) represents a complex *Aerial Imaging* task that challenges the user to execute precise stops and maintain stability for image capture. This trajectory integrates all five sub-skills to ensure a comprehensive evaluation of skill acquisition. For the demonstration, we define five representative sub-skills



(a) A simple course requiring two sub-skills: Hover and Strafe.

(b) An intermediate-level course combining four sub-skills: Hover, Turn, Strafe, and Ascend/Descend.

(c) A complex course integrating all five sub-skills to execute smooth deceleration and stable hovering for imaging tasks.

Fig. 2 Assigned Trajectories with increasing complexity for the numerical simulations.

based on the domain knowledge [33, 34]. Table 1 summarizes these sub-skills and their descriptions. The selected sub-skills cover the most common UAV maneuver types observed in the assigned trajectories, though they do not encompass all possible maneuvers. Two numerical simulation scenarios are designed to validate the adaptability of the proposed framework:

- Progressive skill improvement: Simulates gradual pilot learning over time to validate that the proposed method can adaptively adjust the assistance level as the novice's performance improves over time.
- Uneven skill proficiency: Simulates nonuniform proficiency across sub-skills to demonstrate that the proposed framework can provide sub-skill-specific assistance.

Table 1 Representative UAV sub-skills and their descriptions.

Sub-skill	**Description**
Hover	Maintain position and heading with minimal drift.
Turn	Rotate around the vertical axis to a new heading while maintaining a fixed position.
Strafe	Move along one of the body-frame axes (forward, backward, left, or right) while maintaining a constant heading and altitude.
Ascend/Descend	Adjust altitude while maintaining horizontal position and heading.
Slow Stop	Gradually decelerate to a stable hover from forward motion.

B. Implementation Details

All numerical simulations are conducted in a Python-based simulation environment implementing the multi-rotor UAV dynamics described in Appendix. The simulator operates at a 50 Hz frequency with a prediction horizon of 40 steps for the MPC controller. The *skill identification* module employs the HSMM, and is implemented using the `pyhsmm` library [24]. The maximum of twenty latent components is used, and model parameters are learned offline through blocked Gibbs sampling until the log-likelihood of the expert demonstration data under the HSMM converges. The hyperparameters of the Poisson duration model were selected through a constrained model-selection process. A set of candidate configurations was evaluated on expert demonstration data based on classification accuracy.

The *skill classification* module is implemented as a unidirectional LSTM network with two stacked layers of 128 hidden units each and a time window length of $L = 150$. Input sequences consist of 16-dimensional concatenated state–control vectors normalized using dataset statistics. The network is trained offline for 60 epochs using the Adam optimizer with a learning rate of 10^{-4} and the batch size of 256. During online operation, the trained classifier model runs in real-time using a sliding buffer of the most recent 150 samples to predict the active sub-skill.

The *skill evaluation* module adopts a conditional discriminator implemented as a three-layer feedforward neural network with 256, 128, and 64 hidden units. The discriminator takes 16-dimensional state–action inputs and is conditioned on one of five sub-skill labels. The model is trained for 200 epochs with the batch size of 256 using the Adam optimizer (learning rate 10^{-4}) and the combined objective in Eq. (11), with the loss weights $(\omega_i, \omega_{ca}, \omega_{gp}) = (10.0, 1.0, 5.0)$. During runtime, the trained model provides real-time sub-skill-specific evaluation scores for the shared control module.

C. Data

Expert and novice datasets are synthetically generated by perturbing the control inputs of the assigned trajectories with additive Gaussian noise [35, 36] and time delay [37, 38]. This design maintains consistent trajectory structures while introducing variability representative of human control performance. To simulate realistic pilot diversity, six expert and six novice pilot profiles are created. Each pilot is assigned distinct noise levels and time-delay parameters for different sub-skills, capturing individual differences in control precision and reaction time. Every pilot performs ten runs for each of the three trajectory tasks (Trajectories A–C), producing a total of 60 demonstrations per group. The resulting expert and novice datasets are used to train the skill identification, skill classification, and skill evaluation modules under identical conditions. For testing, two evaluation datasets are generated: (i) Progressive Skill Improvement, comprising ten novice profiles with ten runs each, simulating the users with gradually improving control performance; and (ii) Uneven Skill Proficiency, consisting of three novice profiles with six runs each, simulating the users with nonuniform skill levels across sub-skills. These datasets are used exclusively for evaluation to ensure that the framework's adaptive behavior is tested on unseen proficiency conditions.

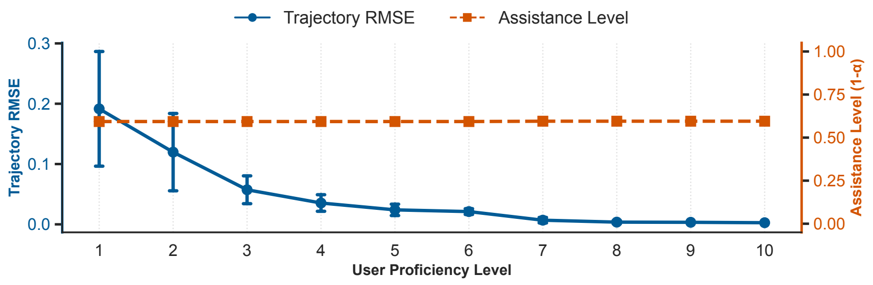
D. Evaluation

The proposed Multidimensional Shared Control framework (MD-SC) is evaluated against two baseline conditions: (i) *No Shared Control (NSC)*, where the UAV is fully operated by the novice agent without any autonomous assistance, and (ii) *Single-Dimensional Shared Control (SD-SC)*, which follows the same shared control structure but removes sub-skill conditioning. In the latter case, the discriminator is trained without the sub-skill label as input, producing a global proficiency estimate for each trajectory window. These baselines provide reference points for assessing whether sub-skill-specific assistance leads to measurable improvements in task performance and training efficiency.

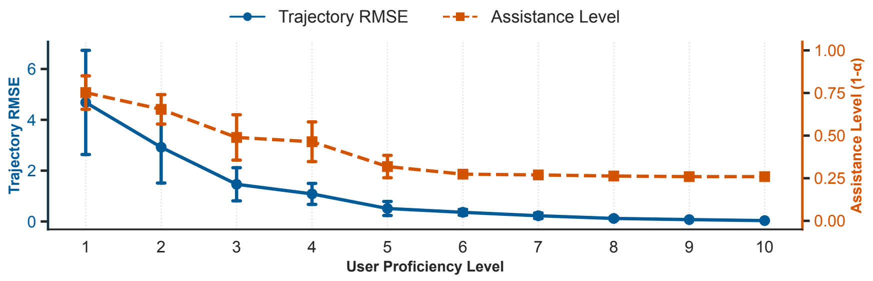
Evaluation is conducted using three key metrics commonly applied in human–machine interaction (HMI) research. The sub-skill-wise score captures how well the user performs on individual sub-skills, as estimated by the discriminator [28]. Trajectory tracking accuracy measures how closely the UAV follows the full reference trajectory throughout the task [39]. Finally, we measure the average assistance level provided by the autonomous agent for each tracking task.

IV. Results

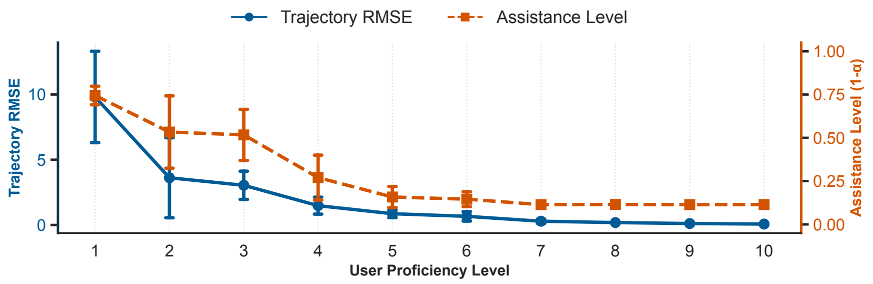
In the first numerical simulation scenario, we test the proposed framework's ability to adapt the assistance level as the user's proficiency improves over time. Figure 3 presents the mean trajectory and the mean assistance levels across different user proficiency levels, where level 1 represents the lowest skill proficiency and level 10 corresponds to expert-like performance. The error bars denote one standard deviation across simulated pilot runs from the stochastic Gaussian noise added to the control inputs. Note that the trajectory tracking error is computed using the raw user state without autonomous correction to reflect the underlying user proficiency rather than the assisted system performance. In



(a) Trajectory A



(b) Trajectory B



(c) Trajectory C

Fig. 3 Progression of trajectory tracking error and assistance level relative to pilot proficiency.

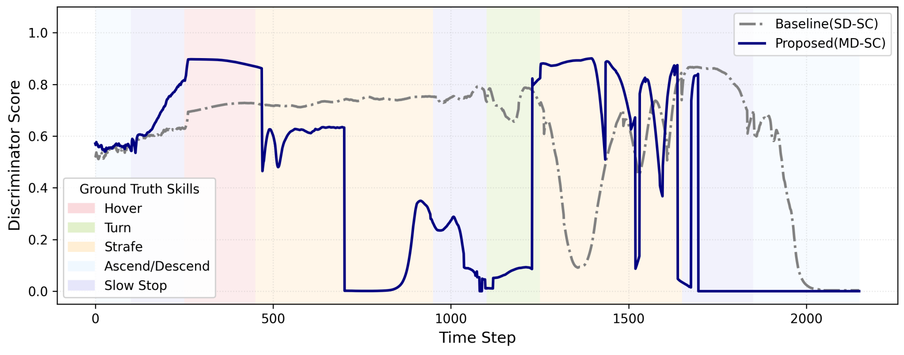
the figures, the blue solid lines indicate the root mean square trajectory tracking error (RMSE). The error decreases as proficiency increases, showing the simulated learning progression. The orange dashed lines represent the assistance level provided by the system.

In Fig. 3a, the assistance level remains nearly constant across proficiency levels for Trajectory A. This is because *Hover* and *Strafe* are relatively simple sub-skills, making the expert and novice demonstrations highly similar. As a result, the discriminator cannot reliably distinguish between them and outputs a neutral score of 0.5. The shorter duration of Trajectory A further limits the model's ability to capture fine-grained differences. In contrast, Trajectories B and C exhibit the expected trend. The assistance level decreases as the user's proficiency improves, gradually transferring control authority back to the human pilot. This result confirms that the proposed framework adaptively adjusts the assistance level according to the user's skill, within the limits of discriminator sensitivity and data variability.

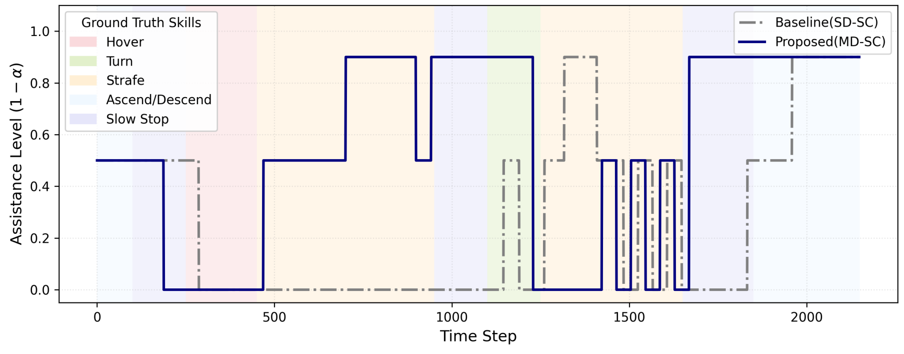
In the second numerical simulation scenario, we evaluate the proposed framework against the two baselines: No Shared Control (NSC) and Single-Dimensional Shared Control (SD-SC). Three pilot profiles are simulated with uneven proficiency across sub-skills to examine whether the proposed method can deliver skill-specific assistance that leads to better trajectory tracking performance. Table 2 summarizes the quantitative results. Each entry reports the mean ± standard deviation across six randomized runs over three pilot profiles. The *Trajectory Error* column shows the trajectory tracking RMSE computed from the assisted system state to evaluate final task performance, with values

Table 2 Quantitative comparison of performance across all sub-skills.

Skill	Framework	Trajectory Error	Assistance Level
Hover	NSC	1.72 ± 2.13	0.00 ± 0.00
	SD-SC	1.53 ± 2.37 (11.2%)	0.40 ± 0.38
	MD-SC	**0.98** ± **1.46** (**43.0%**)	0.40 ± 0.43
Turn	NSC	1.84 ± 1.37	0.00 ± 0.00
	SD-SC	1.01 ± 0.61 (44.9%)	0.56 ± 0.31
	MD-SC	**0.71** ± **0.63** (**61.2%**)	0.76 ± 0.28
Strafe	NSC	1.70 ± 1.67	0.00 ± 0.00
	SD-SC	0.51 ± 0.64 (69.8%)	0.39 ± 0.23
	MD-SC	**0.43** ± **0.55** (**74.8%**)	0.56 ± 0.28
Ascend/Descend	NSC	3.11 ± 3.38	0.00 ± 0.00
	SD-SC	3.12 ± 3.92	0.48 ± 0.27
	MD-SC	**1.92** ± **2.33** (**38.2%**)	0.67 ± 0.29
Slow Stop	NSC	2.59 ± 2.64	0.00 ± 0.00
	SD-SC	1.47 ± 1.69 (43.5%)	0.31 ± 0.27
	MD-SC	**1.12** ± **1.33** (**56.7%**)	0.49 ± 0.36



(a) Score over Time



(b) Assistance Level over Time

Fig. 4 Demonstration of an online run (Trajectory C, Pilot 2, Run 2).

in parentheses indicating percentage improvement relative to NSC. The *Assistance level* column reports the average assistance provided by the autonomous agent, computed at each time step as $1 - \alpha_k$, where α_k denotes the user's control authority.

The proposed Multidimensional Shared Control (MD-SC) method consistently achieves lower trajectory error and higher improvement rates compared to both baselines. The proposed method performs better because the skill evaluation module is conditioned on the active sub-skill, enabling a more fine-grained comparison between expert and novice behaviors. This allows the discriminator to detect subtle differences in control quality across behaviors, resulting in more precise assistance. As reflected in the assistance column, the proposed framework allocates higher assistance, leading to more context-aware support during training.

To further illustrate the proposed framework in the uneven proficiency scenario, Figure 4 presents a representative online run from the same simulation setting. Figure 4a shows the discriminator score over time, while Figure 4b presents the corresponding assistance level. The background is color-coded, with each color representing one ground-truth sub-skill segment. In this demonstration, the discriminator score for the proposed method fluctuates more than the baseline, indicating that it responds more sensitively to variations in user behavior. This finer granularity arises from conditioning the discriminator on sub-skills, which allows it to differentiate expert and novice control patterns in greater detail. The pilot in this run performs well in *Hover*, *Strafe*, and *Turn*, but struggles with *Ascend/Descend* and *Slow Stop*. The score trajectory of the proposed method reflects this pattern, showing higher scores in proficient sub-skills and lower scores in weaker ones, with a slight delay. The delay results from the time window $\mathbf{w}_k$ introduced in Sections II.C and II.D, which allows the classifier and discriminator to aggregate recent temporal information before updating their prediction. This temporal smoothing improves robustness but introduces a short response latency. In Fig. 4b, we observe that the assistance level follows the score-to-control mapping, providing more assistance when the discriminator score is low. This demonstration shows that the proposed framework can evaluate pilot performance at a finer granularity and adjust the assistance level in real-time according to the active sub-skill proficiency.

V. Conclusion

In this paper, we proposed a multidimensional skill-conditioned shared control framework for training human UAV pilots. The framework first decomposes a complex navigation task into sub-skills using a Hidden Semi-Markov Model (HSMM)-based skill identification module. Using the resulting labels, a skill classifier provides real-time sub-skill recognition, and a skill-evaluation module produces sub-skill-specific proficiency scores by comparing novice behavior against expert demonstrations. A skill-conditioned shared controller then maps these scores to discrete assistance modes, providing more support when proficiency is low. We demonstrated the performance of the proposed framework via a numerical simulation of multi-rotor UAV trajectory tracking with synthetic expert and novice profiles. In a progressive skill improvement scenario, the proposed framework decreases assistance as simulated proficiency increases, returning control authority to the pilot while tracking underlying performance. In an uneven proficiency scenario, the proposed framework consistently allocates assistance to weaker sub-skills and reduces trajectory tracking error.

Appendix

Dynamic System Modeling

We consider a multi-rotor UAV with 6 degree-of-freedom (DOF). The system is modeled as a general nonlinear system in discrete time:

$$\begin{bmatrix} \mathbf{p}_{k+1} & \dot{\mathbf{p}}_{k+1} & \boldsymbol{\phi}_{k+1} & \omega_{k+1} \end{bmatrix} = f(\mathbf{p}_k, \dot{\mathbf{p}}_k, \boldsymbol{\phi}_k, \omega_k, \mathbf{u}_k) \tag{22}$$

where $\mathbf{p}_k \in \mathbb{R}^3$ is the position, $\dot{\mathbf{p}}_k \in \mathbb{R}^3$ is the velocity, $\boldsymbol{\phi}_k \in \mathbb{R}^3$ is the Euler angle, and $\omega_k \in \mathbb{R}^3$ is the angular velocity of the system at time step k. The control input at time step k is defined as $\mathbf{u}_k = [\phi_{d,k},\ \theta_{d,k},\ \dot{\psi}_{d,k},\ T_k]^\top$, where $\phi_{d,k}$ and $\theta_{d,k}$ are the commanded roll and pitch angles, $\dot{\psi}_{d,k}$ is the desired yaw rate, and T_k is the total thrust. We can reformulate Eq. (22) as:

$$\begin{bmatrix} \mathbf{p}_{k+1} \\ \dot{\mathbf{p}}_{k+1} \end{bmatrix} = \begin{bmatrix} I & I\Delta t \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \dot{\mathbf{p}}_k \end{bmatrix} + \begin{bmatrix} 0 \\ I\Delta t \end{bmatrix} \mathcal{U}(\mathbf{p}_k, \dot{\mathbf{p}}_k, \boldsymbol{\phi}_k, \omega_k, \mathbf{u}_k) \tag{23}$$

where I is the identity matrix, Δt is the time step, and $\mathcal{U}$ is the mapping from control input command to translational acceleration. The mapping $\mathcal{U}$ can be expressed as [40]:

$$\ddot{\mathbf{p}}_k = \mathcal{U}(\mathbf{p}_k, \dot{\mathbf{p}}_k, \boldsymbol{\phi}_k, \omega_k, \mathbf{u}_k) = -\frac{1}{m_u}\mathcal{R}(\psi_k)\begin{bmatrix}\cos\phi_k \sin\theta_k \\ -\sin\phi_k \\ \cos\phi_k \cos\theta_k\end{bmatrix} T_k + \mathbf{g} \tag{24}$$

where $\mathcal{R}(\psi_k) \in \mathbb{R}^{3\times3}$ is the rotation matrix about the yaw axis, and m_u and $\mathbf{g}$ represent the mass of the multi-rotor UAV and the gravity vector, respectively.

References

[1] Federal Aviation Administration, "14 CFR Part 107 – Small Unmanned Aircraft Systems," Code of Federal Regulations, Title 14, Part 107, 2025. URL https://www.ecfr.gov/current/title-14/chapter-I/subchapter-F/part-107, accessed: 25 November 2025.

[2] Federal Aviation Administration, "Advisory Circular 107-2A: Small Unmanned Aircraft System (Small UAS)," , Feb. 2021. URL https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC_107-2A.pdf, accessed: 25 November 2025.

[3] Federal Aviation Administration, "Urban Air Mobility (UAM) Concept of Operations 2.0," , 2023. URL https://www.faa.gov/sites/faa.gov/files/Urban%20Air%20Mobility%20(UAM)%20Concept%20of%20Operations%202.0_1.pdf, accessed: 30 November 2025.

[4] Marcano, M., Diaz, S., Perez, J., and Irigoyen, E., "A Review of Shared Control for Automated Vehicles: Theory and Applications," *IEEE Transactions on Human-Machine Systems*, Vol. 50, No. 6, 2020, pp. 475–491. https://doi.org/10.1109/THMS.2020.3017748.

[5] Federal Aviation Administration, "FAA Aerospace Forecast: Fiscal Years 2024–2044," , 2024. URL https://www.faa.gov/dataresearch/aviation/aerospaceforecasts/faa-aerospace-forecast-fy-2024-2044, accessed: 30 November 2025.

[6] Fitzsimons, K., Kalinowska, A., Dewald, J. P., and Murphey, T. D., "Task-Based Hybrid Shared Control for Training through Forceful Interaction," *The International Journal of Robotics Research*, Vol. 39, No. 9, 2020, pp. 1138–1154.

[7] Li, Y., Huegel, J. C., Patoglu, V., and O'Malley, M. K., "Progressive Shared Control for Training in Virtual Environments," *World Haptics 2009-Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, IEEE, 2009, pp. 332–337.

[8] Byeon, S., Choi, J., Zhang, Y., and Hwang, I., "Stochastic-Skill-Level-Based Shared Control for Human Training in Urban Air Mobility Scenario," *ACM Transactions on Human-Robot Interaction*, Vol. 13, No. 3, 2024, pp. 1–25. https://doi.org/10.1145/3603194.

[9] Pezeshki, L., Sadeghian, H., Keshmiri, M., Chen, X., and Haddadin, S., "Cooperative Assist-As-Needed Control for Robotic Rehabilitation: A Two-Player Game Approach," *IEEE Robotics and Automation Letters*, Vol. 8, No. 5, 2023, pp. 2852–2859. https://doi.org/10.1109/LRA.2023.3261750.

[10] Ng, A. Y., and Russell, S., "Algorithms for Inverse Reinforcement Learning," *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, Morgan Kaufmann, Stanford, CA, 2000, pp. 663–670.

[11] Inga, J., Köpf, F., Flad, M., and Hohmann, S., "Individual Human Behavior Identification Using an Inverse Reinforcement Learning Method," *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 99–104. https://doi.org/10.1109/SMC.2017.8122585.

[12] Li, W., Li, Q., Li, S. E., Li, R., Ren, Y., and Wang, W., "Indirect Shared Control Through Non-Zero Sum Differential Game for Cooperative Automated Driving," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 9, 2022, pp. 15980–15992. https://doi.org/10.1109/TITS.2022.3146895.

[13] Li, M., Song, X., Cao, H., Wang, J., Huang, Y., Hu, C., and Wang, H., "Shared Control with a Novel Dynamic Authority Allocation Strategy Based on Game Theory and Driving Safety Field," *Mechanical Systems and Signal Processing*, Vol. 124, 2019, pp. 199–216. https://doi.org/10.1016/j.ymssp.2019.01.040.

[14] Lee, D., Franchi, A., Son, H. I., Ha, C., Bülthoff, H. H., and Giordano, P. R., "Semiautonomous Haptic Teleoperation Control Architecture of Multiple Unmanned Aerial Vehicles," *IEEE/ASME Transactions on Mechatronics*, Vol. 18, No. 4, 2013, pp. 1334–1345. https://doi.org/10.1109/TMECH.2013.2263963.

[15] Lee, Y., Yang, J., and Lim, J. J., "Learning to Coordinate Manipulation Skills via Skill Behavior Diversification," *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxB2lBtvH.

[16] Yu, C., Xu, Y., Li, L., and Hsu, D., "COACH: Cooperative Robot Teaching," *6th Annual Conference on Robot Learning*, 2022.

[17] Byeon, S., Sun, D., and Hwang, I., "Skill-Level-Based Hybrid Shared Control for Human-Automation Systems," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, Melbourne, Australia, 2021, pp. 1507–1512. https://doi.org/10.1109/SMC52423.2021.9658994.

[18] Ijspeert, A. J., Nakanishi, J., and Schaal, S., "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, Vol. 2, IEEE, 2002, pp. 1398–1403.

[19] Park, D.-H., Hoffmann, H., Pastor, P., and Schaal, S., "Movement Reproduction and Obstacle Avoidance with Dynamic Movement Primitives and Potential Fields," *Humanoids 2008-8th IEEE-RAS international conference on humanoid robots*, IEEE, 2008, pp. 91–98.

[20] Murali, A., Garg, A., Krishnan, S., Pokorny, F. T., Abbeel, P., Darrell, T., and Goldberg, K., "TSC-DL: Unsupervised Trajectory Segmentation of Multi-Modal Surgical Demonstrations with Deep Learning," *2016 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2016, pp. 4150–4157.

[21] Shao, Z., Zhao, H., Xie, J., Qu, Y., Guan, Y., and Tan, J., "Unsupervised Trajectory Segmentation and Promoting of Multi-Modal Surgical Demonstrations," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 777–782.

[22] Yamada, Y., Colan, J., Davila, A., and Hasegawa, Y., "Task Segmentation Based on Transition State Clustering for Surgical Robot Assistance," *2023 8th International Conference on Control and Robotics Engineering (ICCRE)*, 2023, pp. 260–264. https://doi.org/10.1109/ICCRE57112.2023.10155581.

[23] Lee, S. H., Suh, I. H., Calinon, S., and Johansson, R., "Autonomous Framework for Segmenting Robot Trajectories of Manipulation Task," *Autonomous robots*, Vol. 38, No. 2, 2015, pp. 107–141.

[24] Johnson, M. J., and Willsky, A. S., "Bayesian Nonparametric Hidden Semi-Markov models," *Journal of Machine Learning Research*, Vol. 14, 2012, pp. 673–701.

[25] Johnson, M., and Willsky, A., "The Hierarchical Dirichlet Process Hidden Semi-Markov Model," *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, UAI 2010*, 2010, pp. 252–259.

[26] Forney, G. D., "The Viterbi Algorithm," *Proceedings of the IEEE*, Vol. 61, No. 3, 2005, pp. 268–278.

[27] Hochreiter, S., and Schmidhuber, J., "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[28] Ho, J., and Ermon, S., "Generative Adversarial Imitation Learning," *Proceedings of the 30th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, 2016, p. 4572–4580.

[29] Liao, H., Li, Z., Meng, Z., Song, R., Li, Y., and Zhang, W., "Integrating Controllable Motion Skills from Demonstrations," *arXiv preprint*, 2024. https://doi.org/10.48550/arXiv.2408.03018.

[30] Borrelli, F., Bemporad, A., and Morari, M., *Predictive Control for Linear and Hybrid Systems*, Cambridge University Press, 2017.

[31] Calinon, S., "A Tutorial on Task-Parameterized Movement Learning and Retrieval," *Intelligent service robotics*, Vol. 9, No. 1, 2016, pp. 1–29.

[32] Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge university press, 2004.

[33] Cao, S., Wang, X., Zhang, R., Yu, H., and Shen, L., "From Demonstration to Flight: Realization of Autonomous Aerobatic Maneuvers for Fast, Miniature Fixed-Wing UAVs," *IEEE Robotics and Automation Letters*, Vol. 7, No. 2, 2022, pp. 5771–5778.

[34] Kim, K., and Hwang, I., "Intent-Based Detection and Characterization of Aircraft Maneuvers in En Route Airspace," *Journal of Aerospace Information Systems*, Vol. 15, No. 2, 2018, pp. 72–91.

[35] Zaal, P., and Sweet, B., "Identification of Time-Varying Pilot Control Behavior in Multi-Axis Control Tasks," *AIAA Modeling and Simulation Technologies Conference*, 2012, p. 4793.

[36] Sternad, D., “It’s Not (Only) the Mean that Matters: Variability, Noise and Exploration in Skill Learning,” *Current opinion in behavioral sciences*, Vol. 20, 2018, pp. 183–195.

[37] Schriver, A. T., Morrow, D. G., Wickens, C. D., and Talleur, D. A., “Expertise Differences in Attentional Strategies Related to Pilot Decision Making,” *Human factors*, Vol. 50, No. 6, 2008, pp. 864–878.

[38] Toader, A., and Ursu, I., “Pilot Modeling Based on Time-Delay Synthesis,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 228, No. 5, 2014, pp. 740–754.

[39] Steinfeld, A., Fong, T., Kaber, D., Lewis, M., Scholtz, J., Schultz, A., and Goodrich, M., “Common Metrics for Human-Robot Interaction,” *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, ACM, Salt Lake City, Utah, USA, 2006, pp. 33–40. https://doi.org/10.1145/1121241.1121249.

[40] Lee, S. J., Kim, S. H., and Kim, H. J., “Robust Translational Force Control of Multi-Rotor UAV for Precise Acceleration Tracking,” *IEEE Transactions on Automation Science and Engineering*, Vol. 17, No. 2, 2020, pp. 562–573. https://doi.org/10.1109/TASE.2019.2935792.