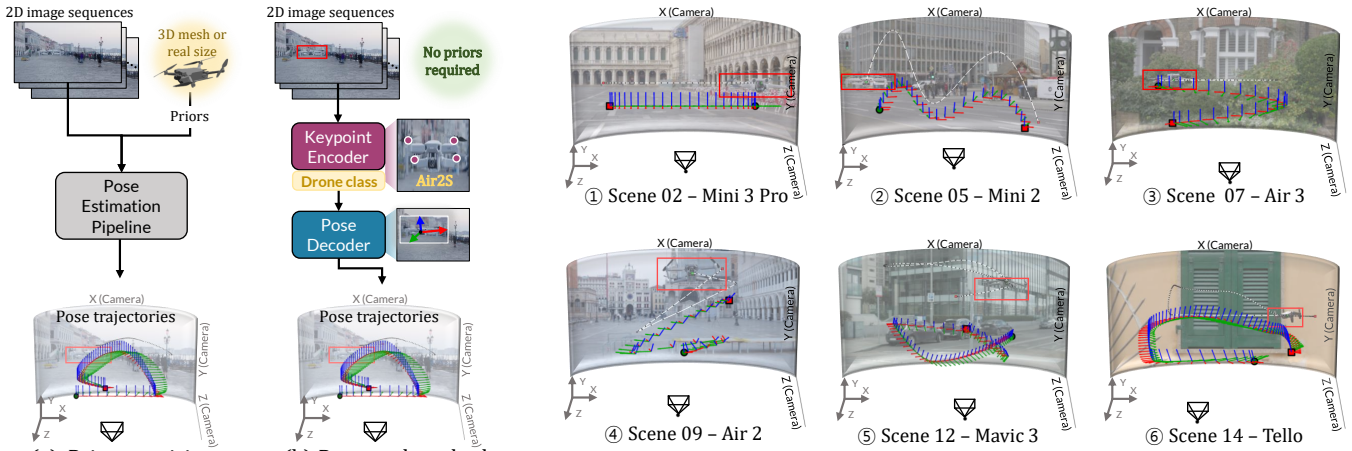
2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

DroneKey++: A Size Prior-Free Method and New Benchmark for Drone 3D Pose Estimation from Sequential Images

Seo-Bin Hwang[1] and Yeong-Jun Cho[1]*



(a) Prior-requiring methods [4, 5, 6]

(b) Proposed method DroneKey++ (Ours)

(c) Examples of 6DroneSyn dataset (Ours).

Fig. 1: **Method overview and our new dataset.** (a) Existing silhouette-based methods are limited to single drone types and need 3D mesh. (b) Our proposed DroneKey++ integrates keypoint encoder and 3D pose decoder, eliminating manual size input for various drone types. (c) Examples from our 6DroneSyn dataset across diverse scenes and drone types with corresponding 3D pose trajectories.

***Abstract*— Accurate 3D pose estimation of drones is essential for security and surveillance systems. However, existing methods often rely on prior drone information such as physical sizes or 3D meshes. At the same time, current datasets are small-scale, limited to single models, and collected under constrained environments, which makes reliable validation of generalization difficult. We present DroneKey++, a prior-free framework that jointly performs keypoint detection, drone classification, and 3D pose estimation. The framework employs a keypoint encoder for simultaneous keypoint detection and classification, and a pose decoder that estimates 3D pose using ray-based geometric reasoning and class embeddings. To address dataset limitations, we construct 6DroneSyn, a large-scale synthetic benchmark with over 50K images covering 7 drone models and 88 outdoor backgrounds, generated using 360-degree panoramic synthesis. Experiments show that DroneKey++ achieves MAE 17.34° and MedAE 17.1° for rotation, MAE 0.135 m and MedAE 0.242 m for translation, with inference speeds of 19.25 FPS (CPU) and 414.07 FPS (GPU), demonstrating both strong generalization across drone models and suitability for real-time applications. The dataset is available at [link].**

I. INTRODUCTION

Recently, illegal drone use is rapidly increasing and threatens national security and public safety. To address these threats, it is essential to develop anti-drone systems that can reliably detect and track unauthorized drones [1]. A key component of these systems is accurate 3D pose estimation, enabling the determination of a drone's line of sight and potential target regions. This provides critical information for assessing the threat level of drone activities. Existing drone 3D pose estimation studies mainly assume access to the drone's internal sensors [20] or rely on multi-camera settings [2]. However, accessing drones' internal sensor data is infeasible, and multi-camera setups are impractical in real-world settings. Given the widespread presence of CCTV in urban areas, single-camera approaches therefore represent the most practical and scalable solution for drone 3D pose estimation.

Various methods [3]–[6] have been proposed for drone 3D pose estimation using single-camera images. However, these approaches rely on prior information about drones, such as physical size or 3D shape priors, restricting their applicability in practice. For example, DronePose [5] relies on 3D drone mesh priors for training, as shown in Fig. 1a. Most drone pose estimation methods [4][6] adopt keypoint detection with a PnP solver [16]. However, these methods also rely on prior information about drones, such as physical size, which must be manually provided for each model and thus limits their applicability to unseen or diverse drones.

To solve these problems, we propose DroneKey++ as illustrated in Fig. 1b. This is a unified learning framework that integrates keypoint detection, drone classification, and 3D pose estimation. It eliminates the need for prior drone information such as physical sizes or 3D meshes. DroneKey++ consists of two main components: The keypoint encoder extends

[1]Department of AI Convergence, Chonnam National University, Korea. cnu.cvl.hsb@gmail.com and yj.cho@jnu.ac.kr
*Corresponding Author

TABLE I: **Comparison of drone datasets.** Each dataset is compared across task type (Task), dataset name (Dataset), number of images (Frames), image resolution (Resolution), number of drone classes (Classes), number and type of backgrounds (Background, in=indoor/out=outdoor), inclusion of synthetic data (Syn), inclusion of real-world data (Real), number of sequences (Seq), annotation types (Annotation), and public availability (Published). N/A is not available.

Task	Dataset	Frames	Resolution	Classes	Background	Syn	Real	Seq	Annotation*	Published
2D Drone Detection	`Drone Image Dataset` [9]	8,001	Diverse	1	Diverse		✓	N/A	2D(B)	✓
	`Drone Dataset` [10]	1,359	Diverse	1	Diverse		✓	N/A	2D(B)	✓
	Hwang et al. [11]	12,541	Diverse	4	Diverse	✓	✓	N/A	2D(B)	✓
3D Trajectory Reconstruction	Li. et al. [2]	N/A	N/A	1	2 (in+out)		✓	2	2D(B), 3D(t)	
	Hwang et al. [11]	2,850	640x480	4	2 (out)	✓	✓	12	2D(B), 3D(t)	✓
3D Pose Estimation	Jin et al. [4]	15,620	N/A	2	5 (in+out)		✓	6	2D(B), 3D(R+t)	
	Albanis et al. [5]	55,988	320x240	1	90 (in)	✓		90	2D(M+B), 3D(R+t)	✓
	You et al. [12]	27,351	1024x1024	4+	3 (out)		✓	3	2D(K+B), 3D(R+t)	
	Hwang et al. [6]	11,200	1920x1080	2	4 (out)	✓		13	2D(K+B), 3D(R+t)	✓
	`6DroneSyn` (Ours)	52,920	1920x1080	7	88 (out)	✓		91	2D(K+B), 3D(R+t)	Will be

* In annotations, B=bounding box, K=keypoint, M=mask, R=rotation, and t=translation.

DroneKey's transformer [7] structure with class tokens [13] to simultaneously perform keypoint detection and drone type classification. This enables the generation of class embeddings that automatically encode physical size information. The pose decoder estimates rays from keypoints, combines them with class embedding features of the drone to estimate 3D points, and then predicts the drone's pose based on this. For training, we use a combined loss function that includes both keypoint and 3D pose errors. Our method achieves MAE 17.34° (rotation), MAE 0.135m (translation), and 414.07 FPS (GPU, 19.25 FPS on CPU), outperforming existing approaches.

Along with these methodological contributions, we also constructed a large-scale benchmark dataset. Since drones are relatively recent objects of study, existing datasets are small-scale and often restricted to a single drone model, which severely limits generalization and the reliability of methodological validation. To overcome these limitations, we introduce `6DroneSyn`, a synthetic dataset with over 50K images covering 7 drone models and 88 diverse outdoor backgrounds (Fig. 1c). The dataset provides rich annotations, including 2D and 3D keypoints, bounding boxes, camera information, and full 3D poses (rotation and translation). By addressing both scale and diversity, `6DroneSyn` enables robust evaluation of pose estimation methods for real-world anti-drone applications. Furthermore, through our 360-degree panorama-based synthesis approach, we generate realistic imagery, with feature distribution analysis in Sec. V-D confirming effective domain gap reduction.

The main contributions of this work are as follows:

- **Prior-free framework.** We integrate keypoint detection, drone classification, and 3D pose estimation into a unified end-to-end framework. Unlike prior methods that rely on PnP solvers with manual size inputs, our learned pose decoder leverages class embeddings to implicitly encode scale information, eliminating the need for external priors.
- **New benchmark.** We present `6DroneSyn`, a high-quality synthetic dataset with over 50K images and comprehensive annotations, specifically designed to minimize the domain gap with real-world environments.
- **Performance.** Our method achieves MAE 17.34° and MedAE 17.1° for rotation, MAE 0.135m and MedAE 0.242m for translation, with inference speeds of 19.25 FPS (CPU) and 414.07 FPS (GPU), demonstrating suitability for real-time applications.

II. RELATED WORKS

A. Drone Datasets

Despite rapid expansion across military, commercial, and civilian applications, drones are not included in large-scale general datasets like COCO [8]. In recent years, specialized drone datasets have been developed for research areas such as drone detection, 3D trajectory reconstruction, and 3D pose estimation, as summarized in Tab.I.

• **2D drone detection.** It was the first area to develop drone datasets, with early works including `Drone Image Dataset` [9] and `Drone Dataset` [10]. These datasets contain 8,001 and 1,359 images, respectively, and have been primarily used for training 2D drone detection models. In addition, `2Drone (on+Aug)` [11] was introduced, generating 12,541 large-scale training samples by synthesizing 4 DJI drone models with diverse backgrounds through data augmentation.

• **3D trajectory reconstruction.** Unlike detection datasets that provide only 2D annotations, 3D trajectory reconstruction datasets include both 2D images and 3D drone positions for each frame. Li et al. [2] proposed a dataset captured in multi-camera environments, but it is not publicly available. Hwang et al. [11] proposed `Syn3Drone`, which provides 2,850 drone frames based on 4+ drone models and 12 scenarios, supporting trajectory estimation and reconstruction research. However, these data only provide drone positions without rotation, limiting their use in security applications where orientation and viewing direction are critical.

• **3D pose estimation.** Several datasets, including `UAVA` [5], `UAV-ADT` [12], and `3DKey+3DPose` [6] have been proposed for drone 3D pose estimation. `UAVA` [5] has substantial scale (55,988 images) but is limited to a single drone model and low resolution (320×240), while `UAV-ADT` [12] includes real data but has limited scale (27,351 images). `3DKey+3DPose` [6] provides high-resolution data but is restricted to only 2 drone models. Consequently, all existing datasets have limitations in drone model diversity and real-world generalization.

Since collecting precise 3D drone pose information in real-world environments is nearly impossible, most research relies on synthetic data. In fact, all publicly available drone pose estimation datasets are synthetic, as obtaining accurate ground truth in real environments is extremely challenging due to

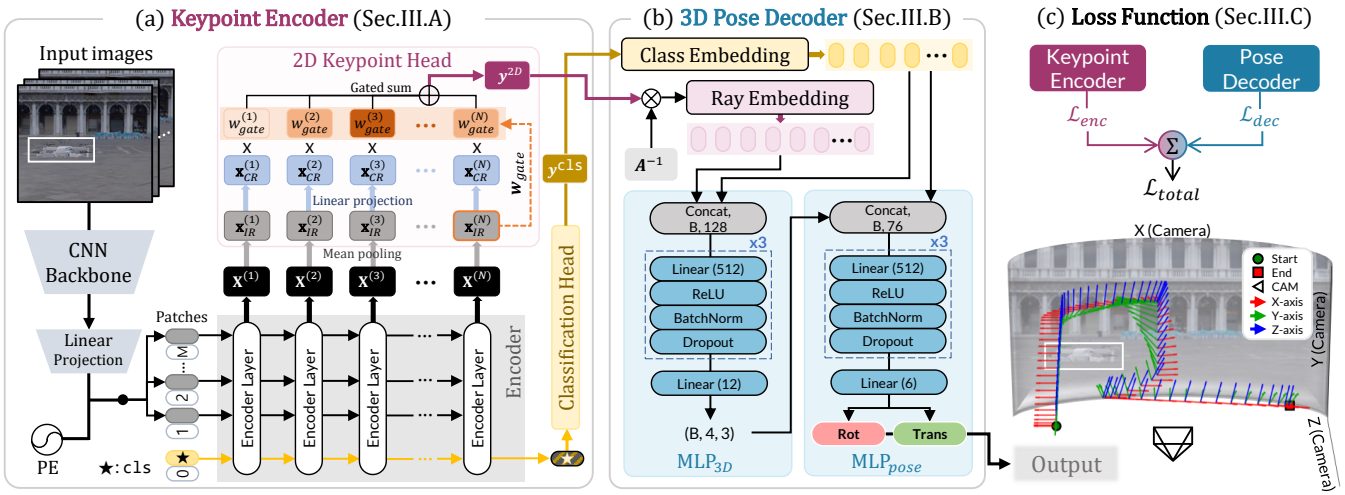


Fig. 2: **Overall framework for DroneKey++.** The proposed end-to-end pipeline consists of three main components: (a) keypoint encoder, which extracts 2D keypoints and drone class features from the input image sequence; (b) 3D pose decoder, which integrates class embedding and ray embedding to estimate the drone's 3D rotation and translation; and (c) loss function, which combines encoder and decoder supervision into the total loss. This unified architecture enables simultaneous keypoint detection, class prediction, and accurate 3D pose estimation.

safety regulations, equipment limitations, and cost constraints. However, existing synthetic data suffer from domain gap issues due to differences from real environments and lack of drone model diversity. Therefore, generating synthetic data similar to real environments and building comprehensive benchmarks covering diverse drone models is essential.

B. Drone 3D Pose Estimation Methods

Drone 3D pose estimation is becoming increasingly important for anti-drone systems and surveillance, yet it remains underexplored. Existing studies can mainly be categorized into two types: those that depend on 3D mesh priors and those that depend on physical size priors. For example, DronePose [5] adopts synthetic training with a silhouette loss, which requires drone 3D mesh priors to render silhouette masks. However, its reliance on a single drone model and 3D mesh priors hinders generalization to drones of different shapes. Jin et al. [4] leveraged relational graph networks to enhance accuracy, while DroneKey [6] proposed gated key-representations. Nevertheless, both approaches depend on prior knowledge of the drone's physical size to establish 2D–3D correspondences via PnP solvers [16].

In summary, both 3D mesh priors and the size priors restrict scalability and prevent immediate deployment to unknown drones in practical environments. This highlights the need for *prior-free* methods that can automatically learn class and scale cues directly from images. Such methods would enable robust 3D pose estimation across diverse and unseen drone types.

III. PROPOSED METHODS

As shown in Fig. 2, we propose DroneKey++, an end-to-end framework that integrates keypoint detection, drone classification, and 3D pose estimation. It consists of two main networks: a **keypoint encoder** and a **3D pose decoder**. The keypoint encoder detects 2D keypoints and the drone's class from input images. The 3D pose decoder then estimates the drone's 3D pose, including translation and rotation, based on the extracted keypoints and class features. Unlike existing approaches such as DroneKey [6] that rely on a two-stage pipeline with a PnP solver [16] requiring manually provided physical sizes, our method replaces this with a learned pose decoder, enabling fully end-to-end prior-free pose estimation. Specifically, the network first infers the drone's class from the input images, and this inferred class is then exploited as class embeddings within the decoder network to implicitly encode scale and shape cues, guiding the learning of 3D pose representations. Through this design, the network can estimate the drone's full 3D pose directly from images alone, without requiring any external priors.

A. Keypoint Encoder

For 3D drone pose estimation, keypoints are essential to capture the object's geometric structure. Propellers are commonly chosen as keypoints because they are present across most drone models and provide geometric cues. However, traditional CNN-based methods [21][22] face challenges in reliable keypoint detection, since propellers appear highly similar across a drone and also require consistent ordering to recover the 3D geometry. To address this issue, prior work such as DroneKey [6] proposed a transformer-based encoder [7], which leverages self-attention to learn spatial arrangements among keypoints.

Unlike prior methods that rely on explicit priors such as the drone's physical size for 2D–3D matching, we extend the DroneKey [6] keypoint encoder with a classification head. By leveraging the ViT [13] `[cls]` token, the encoder predicts drone class information, guiding the learning of 3D geometry for prior-free pose estimation. As a result, the proposed encoder jointly performs spatial keypoint extraction and drone classification.

To achieve this, we employ ResNet [19] as the CNN backbone to extract feature maps $\mathbf{X}$ from input images for transformer input. The feature map is then tokenized

into patches, converted to embeddings, and combined with positional encoding to form $\mathbf{X}^{(0)}$, which represents the initial input with spatial position. $\mathbf{x}_i^{(0)} \in \mathbb{R}^d$ represents the i-th token of dimension d, and the complete set is composed of $\mathbf{X}^{(0)} = \{\mathbf{x}_1^{(0)}, \ldots, \mathbf{x}_M^{(0)}\} \in \mathbb{R}^{d \times M}$. A learnable [cls] token and the token set are jointly fed into N self-attention layers:

$$\left[\mathbf{x}_{\texttt{cls}}^{(l)},\ \mathbf{X}^{(l)}\right] = \text{selfAttn}\left(Q,K,V = \left[\mathbf{x}_{\texttt{cls}}^{(l-1)},\ \mathbf{X}^{(l-1)}\right]\right), \quad (1)$$

where, $l = 1,2,...,N$ represents the index of transformer encoder layers and $\mathbf{X}^{(l)}$ is the set of feature representations after passing through the l-th layer. Q, K and V for self-attention operations, represent query, key, and value.

To effectively integrate features across encoder layers, we adopt a gated summation mechanism, following DroneKey [6]. For the output of each layer $\mathbf{X}^{(l)}$, we perform max pooling, yielding an intermediate representation $\mathbf{X}_{IR}^{(l)} \in \mathbb{R}^d$. We then apply a linear projection to map $\mathbf{X}_{IR}^{(l)}$ into the keypoint space,

$$\mathbf{X}_{CR}^{(l)} = \mathbf{W} \cdot \mathbf{X}_{IR}^{(l)} \in \mathbb{R}^{4\times 2}, \quad (2)$$

where of the four rows corresponds to four keypoints and the two columns represent its (x,y) coordinates. The resulting compact representations $\mathbf{X}_{CR}^{(l)}$ are subsequently integrated across layers through gated summation. To this end, we learn a weight vector of dimension N from the final layer's intermediate representation $\mathbf{X}_{IR}^{(N)}$ as follows:

$$\mathbf{w}_{gate} = \text{softmax}\left(\mathbf{W}_g \cdot \mathbf{X}_{IR}^{(N)}\right) = \left[w_{gate}^{(1)}, w_{gate}^{(2)}, \ldots, w_{gate}^{(N)}\right], \quad (3)$$

where $\mathbf{W}_g \in \mathbb{R}^{N\times d}$ is a projection matrix. Final keypoint coordinates are predicted as follows:

$$\mathbf{y}^{2D} = \text{ReLU}\left(\sum_{l=1}^{N} w_{gate}^{(l)} \mathbf{X}_{CR}^{(l)}\right). \quad (4)$$

The Rectified Linear Unit (ReLU) prevents unrealistic negative predictions and enables stable representation learning.

To estimate drone classes alongside keypoint detection, we use the final [cls] token output $\mathbf{x}_{\texttt{cls}}^{(N)}$, which encodes global image information. This representation is then passed to a separate classification head to predict drone types as follows:

$$\mathbf{y}^{\texttt{cls}} = \text{softmax}\left(\mathbf{W}_{\texttt{cls}} \cdot \mathbf{x}_{\texttt{cls}}^{(N)}\right), \quad (5)$$

where, $\mathbf{W}_{\texttt{cls}}$ is the linear transformation matrix for classification and $\mathbf{y}^{\texttt{cls}}$ represents the probability distribution over drone classes.

B. 3D Pose Decoder

Traditional approaches compute drone 3D pose by combining keypoints with explicit priors, e.g., the drone's physical size. In contrast, we propose a novel decoder that allows the network to learn this estimation process directly, without relying on priors. By leveraging the class probability distribution $\mathbf{y}^{\texttt{cls}}$ from the encoder, the decoder learns drone class-specific 3D pose estimation that is more robust to errors than conventional PnP-based [16] 2D–3D matching. The overall process consists of class embedding, ray extraction, feature fusion, and prediction of 3D keypoints and pose.

First, we compute normalized ray direction vectors from predicted 2D keypoints and camera intrinsic parameters by

$$\mathbf{v}_k = \mathbf{A}^{-1}[\mathbf{y}_k^{2D}, 1]^\top, \quad (6)$$

where $\mathbf{A} \in \mathbb{R}^{3\times 3}$ is the camera intrinsic matrix, $\mathbf{y}_k^{2D}$ is the predicted 2D coordinate of the k-th keypoint, and $\mathbf{v}_k$ is the corresponding ray direction. The computed ray direction $\mathbf{v}$ is embedded into $\mathbf{e}_{ray} \in \mathbb{R}^{64}$, transforming raw coordinates into a structured feature space that can be effectively exploited by the network. The drone class predicted by the keypoint encoder is also embedded into $\mathbf{e}_{\texttt{cls}} \in \mathbb{R}^{64}$, which provides cues about the drone's physical size. For these embeddings, we learn simple linear projection layers and then concatenate $\mathbf{e}_{ray}$ and $\mathbf{e}_{\texttt{cls}}$ to form the fused feature $\mathbf{f}_{fused}$.

Next, an MLP-based 3D keypoint head estimates the relative 3D keypoint coordinates of the drone from the fused features as follows:

$$\mathbf{y}^{3D} = \text{MLP}_{3D}\left(\mathbf{f}_{fused}\right). \quad (7)$$

It reconstructs 3D positions from 2D observations and learned class information. Finally, we concatenate the fused features with the predicted 3D keypoints and feed them into an MLP-based pose head as follows:

$$\mathbf{y}^{pose} = \text{MLP}_{pose}\left(\left[\mathbf{f}_{fused},\ \mathbf{y}^{3D}\right]\right). \quad (8)$$

The details of the MLPs (MLP_{3D} and MLP_{pose}) are illustrated in Fig. 2b. The output 6-dimensional vector $\mathbf{y}^{pose} \in \mathbb{R}^6$ is split into rotation and translation components. The first three elements are processed through sigmoid activation to obtain normalized rotation angles $\mathbf{r}^{pred}$, which provides more stable and efficient convergence during training [27]. The last three elements directly represent the translation vector $\mathbf{t}^{pred}$ in the camera coordinate system.

C. Loss Functions

Our framework is trained end-to-end, where the keypoint encoder and 3D pose decoder are optimized jointly. Since both modules address closely related tasks, we design a unified loss that combines their objectives, ensuring consistent learning across stages. The keypoint encoder is formulated as a multi-task structure that performs 2D keypoint detection and drone class classification simultaneously. For 2D keypoint detection, we use mean squared error (MSE) loss $\mathscr{L}_{\text{2D}}$ between predicted keypoint coordinates $\mathbf{y}^{2D}$ and ground truth. For drone class classification, we use cross-entropy loss $\mathscr{L}_{\texttt{cls}}$ between predicted probability distribution $\mathbf{y}^{\texttt{cls}}$ and actual class labels. The loss function for the keypoint encoder is defined as follows:

$$\mathscr{L}_{\text{enc}} = \mathscr{L}_{\text{2D}} + \mathscr{L}_{\texttt{cls}}. \quad (9)$$

The 3D pose decoder is trained through regression tasks to predict 3D keypoints, rotation, and translation, and is optimized with corresponding loss functions. The loss for 3D keypoint prediction is denoted as $\mathscr{L}_{\text{3D}}$, while the loss for translation is denoted as $\mathscr{L}_{\text{trans}}$, both computed as mean squared error (MSE) between predictions and ground truth. Here, the 3D keypoints correspond to the coordinates of the four drone propellers, whereas the translation represents

the 3D coordinates of the drone's center. For rotation, we design a loss function that accounts for the periodic nature of rotation representations. Both the predicted rotation $\mathbf{r}^{pred}$ and the ground-truth rotation $\mathbf{r}^{gt}$ are normalized to the $[0,1]$ range, and the circular loss is then defined as:

$$\mathscr{L}_{\text{rot}} = \frac{1}{3} \sum_{j \in \{x,y,z\}} \left[\min \left(|\mathbf{r}_j^{pred} - \mathbf{r}_j^{gt}|, 1 - |\mathbf{r}_j^{pred} - \mathbf{r}_j^{gt}| \right) \right]^2. \quad (10)$$

This formula directly computes the shortest angular distance on the unit circle and reflects that 0 and 1 represent the same angular position. The 3D pose decoder loss function is defined by combining the losses as follows:

$$\mathscr{L}_{\text{dec}} = \mathscr{L}_{\text{3D}} + \mathscr{L}_{\text{rot}} + \mathscr{L}_{\text{trans}}. \quad (11)$$

Finally, the overall training objective is obtained by summing the encoder and decoder losses:

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{enc}} + \mathscr{L}_{\text{dec}}. \quad (12)$$

In this work, we experimentally investigated several weighting strategies for different loss terms and confirmed that the simple summation scheme consistently yielded stable and effective performance. A detailed comparison and validation of these weighting strategies is provided in Sec. V-C.3.

IV. Dataset - 6DroneSyn

We present 6DroneSyn, a comprehensive benchmark dataset for drone 3D pose estimation. The dataset contains 52,920 high-resolution (1920×1080) synthetic images. The entire dataset comprises 120GB+, representing a substantial scale compared to existing drone pose estimation datasets. It contains 7 DJI drone models (Mini3 Pro, Mini2, Air3, Air2, Mavic 2 Pro, Mavic3, Tello) and 88 diverse background environments to reflect the variety of real surveillance scenarios. The synthetic data comprises 13 scenes, with each scene generating independent sequences for each drone model, totaling 91 sequences. These cover various flight patterns, from simple linear movements to complex 3D trajectories, encompassing all types of drone motions in real operations. The summary and overall composition of 6DroneSyn are presented in Tab. II.

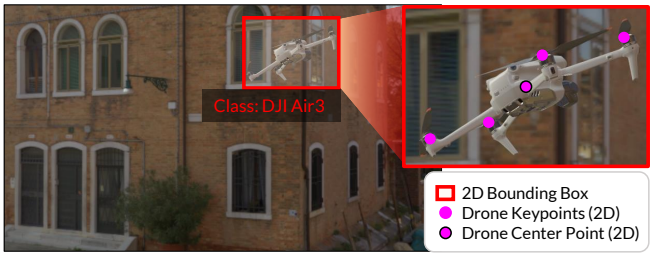
For synthetic data generation, we utilized 360-degree background images collected from 22 real environments. Each environment was rotated 90 degrees around the Z-axis to create a total of 88 background variations. Several examples of the backgrounds are shown in Fig. 1c. We used 7 DJI 3D drone models covering a wide range from small to professional types. Each frame contains the following precise annotations, as illustrated in Fig. 3:

- 2D keypoint coordinates of drone propellers and corresponding 3D coordinates.
- Drone 3D pose (rotation **R** and translation **t**).
- Camera intrinsic and extrinsic parameters.
- 2D bounding boxes of drone regions in full-frame videos.

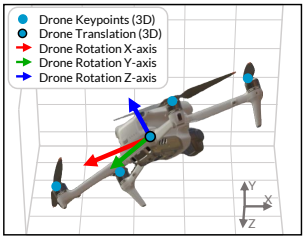
All annotations are automatically generated in the rendering pipeline, completely eliminating human labeling errors. More details of our dataset are provided in the supplementary video.

TABLE II: **6DroneSyn dataset composition.** The dataset consists of 13 synthetic scenes, each featuring diverse drone motion patterns (linear and non-linear) and 22 background variations. Each scene is further divided into 21 subsequences, created from 7 drone types and 3 backgrounds.

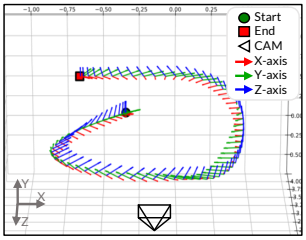
Scene	Motion	# of Subseq.	FPS	Duration	Total frames
#01–#03	Linear	21	30	4s	7,560
#04–#06	Non-linear	21	30	4s	7,560
#07–#09	Linear	21	30	4s	7,560
#10–#13	Non-linear	21	30	12s	30,240



(a) 2D annotations and an example frame from the sequence



(b) 3D annotations



(c) 3D trajectory

Fig. 3: **6DroneSyn Dataset Annotations.** Examples of annotations: (a) 2D bounding boxes and 2D keypoints, (b) 3D keypoints with translation and rotation vectors, and (c) full 3D trajectories with rotation and translation from sequential images.

V. Experimental Results

A. Experimental Settings

Datasets. All experiments were conducted on the proposed 6DroneSyn dataset. The dataset size is about 120GB, significantly larger than existing drone pose datasets. It was split scenario-wise into train (01-02, 04-05, 08, 10-12), validation (03, 09, 13), and test (06-07) sets. The validation set was used to select the best model checkpoints during training, and the selected models were subsequently evaluated on the test set. This experimental setup follows the common practice adopted in most related works.

Evaluation metrics. 3D pose estimation evaluates both rotation and translation accuracy. To assess these aspects, common metrics such as MAE, MedAE, and RMSE are employed [25][26]. However, RMSE is often sensitive to outliers and instability [23][24]; therefore, we adopt MAE and MedAE to better capture both overall accuracy and stable, representative performance. The rotation error is defined as the normalized angular difference between the predicted and ground-truth rotation matrices as follows:

$$\text{MAE}_{\mathbf{r}} = \frac{1}{S} \sum_{s=1}^{S} \frac{1}{\pi} \arccos \left(\frac{\text{tr}\left((\mathbf{r}_s^{pred})^{\top} \mathbf{r}_s^{gt} \right) - 1}{2} \right), \quad (13)$$

TABLE III: **Performance comparison with drone 3D pose estimation approaches.** Results on scene #06 and scene #07 show that our method achieves the best accuracy in both rotation and translation estimation, outperforming DronePose [5] and DroneKey [6].

Methods		Metric		Test scenes													Average
		3D Pose	Metric	Scene #06							Scene #07						
				Mini3	Mini2	Air3	Air2	Mav2	Mav3	Tello	Mini3	Mini2	Air3	Air2	Mav2	Mav3	Tello	
Multi-stage	Keypoint Detector [28] + PnP [16]	**R** (°)	MAE	90.55	69.89	90.28	90.45	78.51	91.39	69.52	99.79	64.31	60.17	74.37	90.85	74.44	76.92	79.25
			MedAE	98.98	84.75	90.26	96.51	76.33	108.96	65.03	106.11	78.11	87.03	86.94	96.92	62.46	70.06	87.55
		t (m)	MAE	0.656	0.295	0.121	0.343	0.450	0.426	0.203	0.521	0.313	0.184	0.448	0.670	0.710	0.199	0.391
			MedAE	0.280	**0.088**	**0.027**	0.095	**0.123**	**0.131**	0.060	**0.155**	**0.094**	0.031	0.120	0.154	**0.128**	0.056	**0.105**
	DroneKey [6]	**R** (°)	MAE	33.84	**14.61**	**11.51**	24.78	22.25	31.85	32.13	25.32	**4.660**	**8.290**	**7.350**	**21.95**	22.29	21.49	20.17
			MedAE	31.96	14.06	**1.840**	24.35	20.61	33.48	30.70	26.98	**1.580**	**8.540**	**6.400**	**25.15**	23.20	21.17	21.28
		t (m)	MAE	0.668	1.094	1.005	0.211	0.462	0.208	**0.056**	0.484	0.952	0.911	**0.017**	**0.168**	0.648	0.577	0.533
			MedAE	0.656	0.830	1.002	**0.032**	0.235	0.187	**0.056**	0.478	0.787	0.950	**0.017**	**0.112**	0.221	0.048	0.370
End-to-end	DronePose [5]	**R** (°)	MAE	80.33	23.67	135.5	45.49	86.16	113.9	88.17	79.42	15.27	121.5	85.72	82.72	105.0	49.45	79.06
			MedAE	81.56	**8.400**	148.4	32.47	100.1	135.5	103.3	100.4	9.650	126.0	88.95	69.28	109.3	43.17	82.63
		t (m)	MAE	0.148	0.161	0.108	0.122	0.176	0.180	0.163	**0.221**	0.179	**0.132**	**0.134**	**0.173**	**0.153**	0.236	0.163
			MedAE	0.280	0.322	0.169	0.219	0.289	0.353	0.301	0.497	0.385	**0.254**	**0.246**	**0.328**	0.238	0.536	0.317
	DroneKey++ (Ours)	**R** (°)	MAE	**9.910**	19.91	19.74	**18.27**	**19.36**	**12.09**	**14.42**	**16.61**	22.35	15.53	17.08	26.90	**12.49**	**18.32**	**17.34**
			MedAE	**8.580**	20.38	18.68	**16.82**	**17.20**	**11.76**	**14.01**	**16.85**	23.24	15.13	16.44	27.27	**12.38**	**19.73**	**17.10**
		t (m)	MAE	**0.121**	**0.142**	**0.067**	**0.100**	**0.140**	**0.114**	0.075	0.268	**0.149**	0.135	0.080	0.234	0.187	**0.083**	**0.135**
			MedAE	**0.224**	0.175	0.129	0.181	0.248	0.219	0.158	0.561	0.187	0.293	0.150	0.580	0.391	**0.171**	0.242

$$\text{MedAE}_{\mathbf{r}} = \text{median}_s \left\{ \frac{1}{\pi} \arccos \left(\frac{\text{tr}\left((\mathbf{r}_s^{pred})^{\top} \mathbf{r}_s^{gt} \right) - 1}{2} \right) \right\}, \quad (14)$$

where s indexes each of the S samples, and $\text{tr}(\cdot)$ denotes the trace, i.e., the sum of the diagonal elements. $(\mathbf{r}_s^{pred})^{\top}\mathbf{r}_s^{gt}$ represents the relative rotation. The trace formula gives the minimal angle difference, always in 0°–180°. Dividing by π normalizes the error to the range $[0, 1]$, where a value of 1 corresponds to a 180° rotation error. Translation error is measured as the Euclidean distance between the predicted and ground-truth translation vectors, evaluated using MAE and MedAE.

Implementation details. Training and evaluation were performed on an NVIDIA A100 GPU, with real-time tests additionally on an Intel Xeon Gold 6440 CPU. We used the Adam optimizer (initial lr = 1e-5) with a cosine annealing scheduler, batch size 32, and 100 training epochs. The backbone was initialized with ImageNet [18] pre-trained ResNet [19], and no data augmentation was applied. All comparisons were performed under identical settings.

B. Performance Comparison of Drone 3D Pose Estimation

We evaluate the performance of the proposed DroneKey++ against existing drone pose estimation methods. Experiments were conducted on test scenes #06 and #07 of the `6DroneSyn` dataset. Since prior research on drone 3D pose estimation is relatively limited, we designed the comparisons to be as fair and comprehensive as possible. Specifically, we considered the following categories of baselines:

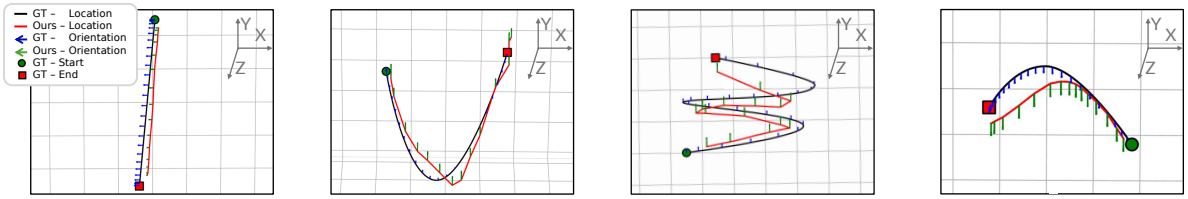
- **Keypoint Detector + PnP**: Although originally developed for common objects such as humans or vehicles rather than drones, existing keypoint detection [28] methods were combined with a PnP solver [16] to infer 3D drone poses.
- **Drone-specific approaches**: We also compare against drone-focused 3D pose estimation methods, such as DroneKey [6] and DronePose [5][1], which share a similar goal to ours.

[1]For DronePose [5], the original 3D mesh-based prior could not be reproduced due to unavailable mesh and dataset. Instead, we reimplemented it using MSE loss on rotation and translation, resulting in a prior-free setting consistent with our evaluation.

Quantitative results are summarized in Table III. Since DronePose [5] was re-implemented in a prior-free setting, it shows large rotation errors, while translation can be relatively competitive in some sequences. On the other hand, DroneKey [6] achieves high rotation accuracy but exhibits unstable translation. This is because PnP solvers inherently suffer from scale ambiguity and require the actual physical size of each drone to resolve absolute translation. The Keypoint Detector [28] + PnP [16] baseline shows competitive $\text{MedAE}_{\mathbf{t}}$ errors but suffers from severe outliers and high variance due to frequent keypoint detection failures on small drone targets.

Our DroneKey++ achieves the lowest rotation errors, with MAE of 17.34° and MedAE of 17.1°, representing improvements of 78% over DronePose and 14% over DroneKey. For translation, it yields MAE of 0.135 m and MedAE of 0.242 m, achieving improvements of 17% over DronePose and 75% over DroneKey. This is because the class embeddings implicitly encode size and shape cues, while the ray features enhance pose estimation robustness, yielding consistent performance across both metrics. We note that DroneKey [6] achieves lower rotation errors on some individual sequences (e.g., Mini2, Air3, Air2 in Scene #07), as it directly uses ground-truth physical sizes via the PnP solver. However, this advantage comes at the cost of unstable translation and the need for manual size input per drone model, which limits practical applicability.

Qualitative results in Fig. 4 demonstrate that our method provides more stable and accurate 3D poses across diverse sequences. The method works robustly even with cluttered backgrounds and varying drone types. Additional qualitative results are provided in the supplementary video, including validation on real drone scenarios to verify the generalization capability. These results indicate that DroneKey++ not only outperforms existing methods in quantitative metrics but also delivers consistent performance across diverse scenarios. However, we observe that performance degrades under extreme rotation angles, where the drone's appearance changes drastically and keypoint detection becomes less reliable. Addressing such cases through rotation-aware augmentation or keypoint confidence estimation remains a direction for future work.



(a) Scene #03 (Valid) - Mini3 (b) Scene #09 (Valid) - Mini2 (c) Scene #06 (Test) - Mavic3 (d) Scene #07 (Test) - Air2

Fig. 4: **Qualitative results of the drone 3D pose estimation based on the proposed method.** The results show pose estimation across different validation and test scenes and drone types after applying Gaussian smoothing post-processing.

TABLE IV: **Effect of the keypoint encoder.** The encoder jointly predicts 2D keypoints and class labels in an end-to-end manner (✓), while the baseline directly uses ground-truth annotations (blank).

Encoder	R (°)		t (m)	
	MAE	MedAE	MAE	MedAE
	59.12	44.28	0.159	0.254
✓	**17.34** (-41.78)	**17.10** (-27.18)	**0.135** (-0.024)	**0.242** (-0.012)

TABLE V: **Ablation study of the 3D pose decoder with different component configurations.**

Decoder Settings	R (°)		t (m)	
	MAE	MedAE	MAE	MedAE
(1) Only MLP_{pose}	17.40	17.19	0.219	0.372
(2) RayEmbed + MLP_{pose}	17.76	17.86	0.153	0.248
(3) RayEmbed + MLP_{pose} + MLP_{3D}	19.71	19.43	0.158	0.296
(4) Full decoder (Ours)	**17.34**	**17.10**	**0.135**	**0.242**

C. Ablation Studies

1) ***Keypoint encoder:*** Table IV presents the effect of the proposed keypoint encoder, which jointly predicts 2D keypoints and class labels. When ground-truth annotations are directly provided without the encoder, rotation estimation suffers from large errors. By contrast, enabling the encoder significantly reduces both MAE and MedAE in rotation (59.12° →17.34° and 44.28° →17.21°, respectively), demonstrating that end-to-end prediction and contextual feature propagation are crucial for robust 3D pose estimation. Translation performance is also improved, with MAE reduced from 0.159m to 0.135m, indicating that the keypoint encoder benefits both rotation and translation estimation.

2) ***3D pose decoder:*** As illustrated in Fig. 2b, the proposed 3D pose decoder consists of two MLPs (MLP_{3D}, MLP_{pose}) and two embedding layers (ray embedding and class embedding). To analyze the contribution of each component, we conducted an ablation study on four decoder configurations: **(1)** Only MLP_{pose}: directly regresses 3D pose from encoder-predicted 2D keypoints; **(2)** RayEmbed + MLP_{pose}: converts 2D keypoints into rays using the camera intrinsic matrix, which are then fed into MLP_{pose}; **(3)** RayEmbed + MLP_{pose} + MLP_{3D}: adds MLP_{3D} to infer 3D keypoints before final pose estimation; **(4)** Full decoder (Ours): integrates all components, including the class embedding, for end-to-end pose estimation.

The results in Table V highlight the contribution of each decoder component. Using only MLP_{pose} yields reasonable rotation (17.4°) but large translation error (0.219m), as 2D observations alone cannot resolve scale ambiguity. Adding ray embedding improves translation to 0.153m by providing geometric depth cues, though absolute scale remains unresolved. Introducing MLP_{3D} degrades both metrics, likely due to accumulated errors from intermediate 3D estimation. The full decoder with class embedding achieves the best performance (0.135m translation), demonstrating that class embeddings implicitly encode physical size and shape cues, enabling automatic scale inference without external priors.

TABLE VI: **Comparison of training strategies with different loss weighting schemes.**

Training settings	R (°)		t (m)	
	MAE	MedAE	MAE	MedAE
Tahn-weighted	27.30	20.40	0.192	0.303
Smoothly-shifted	19.37	19.24	0.154	0.269
3D-biased	18.38	18.62	0.165	0.293
Equal Loss	**17.34**	**17.10**	**0.135**	**0.242**

3) ***Training strategy:*** Table VI compares different loss weighting strategies for the encoder loss $\mathscr{L}_{\text{enc}}$ and the decoder loss $\mathscr{L}_{\text{dec}}$. The tanh-weighted scheme emphasizes the encoder loss during the first half of training and gradually shifts the weight to the decoder loss. In addition to the tanh-weighted scheme, the smoothly-shifted loss follows a similar idea but transitions gradually rather than abruptly. Meanwhile, the 3D-biased loss places stronger emphasis on the 3D terms (e.g., assigning a weight five times larger to the 3D pose decoder). We evaluated these strategies to better understand how loss weighting influences the balance between encoder and decoder.

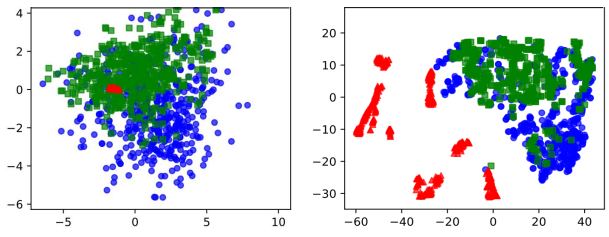
However, the results show that the proposed weighting schemes lead to higher errors in both rotation and translation. In contrast, the Equal Loss strategy, where all loss terms are weighted equally, achieves the best performance across all metrics. This indicates that enforcing complex weight schedules can destabilize optimization, whereas a simple equal weighting naturally balances the contributions of 2D and 3D objectives. These findings highlight that balanced multi-task supervision is more effective than overemphasizing a single objective, making equal weighting the most suitable strategy for our architecture.

D. Evaluating Realism and Diversity of Our Dataset

To evaluate the realism and diversity of our dataset, we compared it against two baselines:

- Real drone image frames collected from the web.
- Existing synthetic 2D drone dataset generated with 3D drone models and virtual backgrounds [11].
- Our dataset created by compositing 3D drone models with 360-degree background panoramas.

For a fair comparison, 500 samples were randomly selected from each dataset. We then applied two dimensionality reduction techniques for visualization. Principal component analysis



(a) PCA (b) t-SNE

Fig. 5: **Comparison of feature distributions across drone image datasets.** Dimensionality reduction visualization comparing real-world drone images (●), existing synthetic dataset (▲), and our 360-camera-based synthetic dataset (■) using (a) PCA with the first two principal components and (b) t-SNE with 2D embedding components.

(PCA) [14] projects high-dimensional features into a lower-dimensional space while preserving major variance, thereby revealing overall distribution patterns. In contrast, t-distributed stochastic neighbor embedding (t-SNE) [15] preserves local similarity, highlighting cluster structures and domain differences. Before analysis, all features were standardized to zero mean and unit variance using a standard scaler [17].

As shown in Fig. 5, existing synthetic data (▲) is concentrated in a narrow region, indicating limited diversity. In contrast, real data (●) and our synthetic data (■) are broadly distributed. In the t-SNE space, our synthetic data overlaps with real data clusters, highlighting the realism and diversity achieved by our 360-degree camera-based synthesis strategy.

VI. Conclusions and Future Works

In this work, we presented DroneKey++, a prior-free end-to-end framework that integrates keypoint detection, drone classification, and 3D pose estimation. By leveraging class embeddings and ray-based geometric reasoning, our method estimates 3D poses accurately without requiring prior size information. It achieves rotation MAE 17.34° and translation MAE 0.135 m, significantly outperforming existing methods, while running in real time at 414.07 FPS on GPU and 19.25 FPS on CPU. In addition, we introduced the `6DroneSyn` dataset with 52,920 images across 7 drone models and 88 backgrounds, generated via 360-degree panorama-based synthesis. This dataset provides high-quality annotations and reduced domain gap, offering a valuable benchmark for advancing future research in drone pose estimation.

This work identifies three main directions for future research. First, we will explore strategies to further balance dataset realism and learnability, ensuring both reduced domain gap and effective training. Second, we will develop keypoint confidence mechanisms and integrate them into the 3D pose decoder to improve robustness under occlusion. Third, we plan to expand our dataset with more diverse drone models and environments, and conduct real-world validation to complete our methodology. In particular, we will investigate generalization to unseen drone types not included in training, addressing the current limitation that class embeddings are learned only from known models. We expect that these extensions will further strengthen the utility of our dataset and framework for future drone-related research.

Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) — the Artificial Intelligence Convergence Innovation Human Resources Development (No. IITP-2023-RS-2023-00256629) and the Innovative Human Resource Development for Local Intellectualization (No. IITP-2026-RS-2022-00156287), and the Korea Institute of Planning and Evaluation for Technology in Food, Agriculture and Forestry (IPET) through the Agriculture and Food Convergence Technologies Program for Research Manpower Development funded by the Ministry of Agriculture, Food and Rural Affairs (MAFRA) (No. RS-2024-00397026). We appreciate the high-performance GPU computing support of HPC-AI Open Infrastructure via GIST SCENT.

References

[1] S. Park *et al.*, "Survey on anti-drone systems: Components, designs, and challenges," *IEEE Access*, vol. 9, pp. 42635–42659, 2021.
[2] J. Li *et al.*, "Reconstruction of 3D flight trajectories from ad-hoc camera networks," in *IROS*, 2020, pp. 8637–8644.
[3] Q. Fu, Q. Quan, and K.-Y. Cai, "Robust pose estimation for multirotor UAVs using off-board monocular vision," *IEEE Trans. Ind. Electron.*, vol. 64, no. 10, pp. 7942–7951, Oct. 2017.
[4] R. Jin *et al.*, "Drone detection and pose estimation using relational graph networks," *Sensors*, vol. 19, no. 6, p. 1479, Mar. 2019.
[5] G. Albanis *et al.*, "Dronepose: Photorealistic UAV-assistant dataset synthesis for 3D pose estimation via a smooth silhouette loss," in *ECCVW*, 2020, pp. 588–604.
[6] S.-B. Hwang and Y.-J. Cho, "DroneKey: Drone 3D pose estimation in image sequences using gated key-representation and pose-adaptive learning," in *IROS*, Oct. 2025, pp. 708–715.
[7] A. Vaswani *et al.*, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.
[8] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *ECCV*, 2014, pp. 740–755.
[9] B. Kartheek, "Drone type classification dataset," Kaggle, 2021. [Online]. Available: `https://www.kaggle.com/datasets/balajikartheek/drone-type-classification`
[10] D. Mehdi, "Drone dataset UAV," Kaggle, 2023. [Online]. Available: `https://www.kaggle.com/datasets/dasmehdixtr/drone-dataset-uav`
[11] S.-B. Hwang and Y.-J. Cho, "Single-camera-based 3D drone trajectory reconstruction for surveillance systems," *IEEE Access*, vol. 13, pp. 56413–56427, 2025.
[12] J. You *et al.*, "UAV-Pose: A dual capture network algorithm for low altitude UAV attitude detection and tracking," *IEEE Access*, vol. 11, pp. 129144–129155, 2023.
[13] A. Dosovitskiy *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," in *ICLR*, 2021.
[14] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philos. Mag.*, vol. 2, no. 11, pp. 559–572, 1901.
[15] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[17] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[18] J. Deng *et al.*, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.
[19] K. He *et al.*, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[20] G. Chi *et al.*, "Wi-drone: Wi-Fi-based 6-DoF tracking for indoor drone flight control," in *MobiSys*, 2022, pp. 56–68.
[21] K. He *et al.*, "Mask R-CNN," in *ICCV*, 2017, pp. 2961–2969.
[22] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.
[23] S. H. Lee and J. Civera, "What's wrong with the absolute trajectory error?," in *ECCVW*, 2024, pp. 108–123.
[24] S. H. Lee *et al.*, "Alignment scores: Robust metrics for multiview pose accuracy evaluation," in *ICCVW*, 2025, pp. 200–209.
[25] T. O. Hodson *et al.*, "Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not," *Geosci. Model Dev.*, vol. 15, no. 14, pp. 5481–5487, 2022.
[26] O. Stamm *et al.*, "Accuracy of monocular two-dimensional pose estimation," *Front. Sports Active Living*, vol. 2, art. 624968, 2020.
[27] Y. LeCun *et al.*, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, 2002, pp. 9–50.
[28] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: `https://github.com/ultralytics/ultralytics`