



# Bayesian Data Augmentation and Training for Perception DNN in Autonomous Aerial Vehicles

Ashik E Rasul<sup>\*†</sup>, Humaira Tasnim<sup>\*†</sup>, and Hyung-Jin Yoon<sup>‡</sup>  
*Tennessee Technological University, Cookeville, TN 38505 USA*

Ayoosh Bansal<sup>§</sup>  
*University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801*

Duo Wang<sup>¶</sup>  
*University of Nevada, Reno, NV 89557 USA*

Naira Hovakimyan<sup>\*\*</sup>, Lui Sha<sup>††</sup>  
*University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801*

Petros Voulgaris<sup>‡‡</sup>  
*University of Nevada, Reno, NV 89557 USA*

Learning-based solutions have enabled incredible capabilities for autonomous systems. Autonomous vehicles, both aerial and ground, rely on Deep Neural Networks (DNN) for various integral tasks, including perception. The efficacy of supervised learning solutions, such as the DNN used for perception tasks, hinges on the quality of the training data. Discrepancies between training data and operating conditions result in faults that can lead to catastrophic incidents. However, collecting and labeling vast amounts of context-sensitive data, with broad coverage of possible variations in the operating environment, is prohibitively difficult. To overcome this limitation, synthetic data generation techniques for DNN training emerged, allowing for the easy exploration of diverse scenarios. While significant synthetic data generation solutions exist for ground vehicles, for aerial vehicles such support is still lacking.

This work presents a data augmentation framework for aerial vehicle's perception training, leveraging photorealistic simulation seamlessly integrated with high-fidelity vehicle dynamics, control, and planning algorithms. Safe landing in urban environments is a crucial challenge in the development of autonomous air taxis, and therefore, landing maneuver is chosen as the focus of this work. With repeated simulations of landing maneuvers in scenarios with varying vehicle states, weather conditions and time of day, we assess the landing performance of the VTOL (Vertical Take off and Landing) type UAV and gather valuable data. The landing performance is used as the objective function to optimize the DNN through retraining. Given the high computational cost of DNN retraining, we incorporated Bayesian Optimization in our framework that systematically explores the data augmentation parameter space to retrain the best-performing models. The framework allowed us to identify high-performing data augmentation parameters that are consistently effective across different landing scenarios. Utilizing the capabilities of this data augmentation framework, we obtained a robust perception model. The model consistently improved the perception-based landing success rate by at least 20% under different lighting and weather conditions.

<sup>\*</sup>Equal contribution

<sup>†</sup>Graduate Student, Mechanical Engineering Department, 1 William L Jones Dr, Cookeville, TN 38505.

<sup>‡</sup>Assistant Professor, Mechanical Engineering Department, 1 William L Jones Dr, Cookeville, TN 38505.

<sup>§</sup>Postdoctoral Research Associate, Computer Science, 201 North Goodwin Avenue, Urbana, IL 61801, USA.

<sup>¶</sup>Postdoctoral Research Associate, Mechanical Engineering Department, 1664 N. Virginia Street, Reno, NV 89557

<sup>||</sup>Visiting Scholar at UIUC, Mechanical Science & Engineering Department, 105 S Mathews Ave, Urbana, IL 61801

<sup>\*\*</sup>Professor, Mechanical Engineering Department

<sup>††</sup>Donald B. Gillies Chair, Computer Science, 201 North Goodwin Avenue, Urbana, IL 61801, USA.

<sup>‡‡</sup>Professor, Mechanical Engineering Department, 1664 N. Virginia Street, Reno, NV 89557.

## I. Introduction

Perception solutions for autonomous vehicles are mostly based on *Deep Neural Networks (DNN)*. Numerous studies have demonstrated their superior performance compared to traditional computer vision methods [1]. Specific frameworks like *You Only Look Once (YOLO)* [2–5] showcase the power of *DNNs* in real-time object detection tasks. However, the success of *DNN* depends heavily on the quality and relevance of the training data. *DNNs* trained exclusively on general datasets often show limited performance when applied to specific applications [6]. This performance gap is rooted in the differences between the training data and the deployment environment [7]. Optimizing performance in complex environments requires fine-tuning the model’s hyperparameters by retraining it within the deployment environment.

Optimizing a *DNN* model for deployment in complex real-world environments comes with two significant challenges. First, such retraining requires large amounts of data collected in varying scenarios. For aerial vehicle perception, this would involve the use of expensive aerial vehicles over long periods of time to cover a comprehensive range of relevant scenarios. This limitation can be alleviated by using photorealistic simulators to freely explore various scenarios and collect task-specific data [7–9]. Second, retraining *DNN* models is computationally expensive, requiring substantial resources and time to refine them for real-world applications [10]. This high computational demand often becomes a bottleneck in the optimization process, necessitating improvements in the efficiency of the retraining process itself. Bayesian optimization has become a prominent method for this purpose, as it systematically searches the decision variable space, balancing exploration and exploitation to identify optimal decision variables without knowing the objective function explicitly [11]. This approach is particularly effective for deep neural network (*DNN*)-based perception systems, where robust performance depends on selecting hyperparameters that generalize well across diverse scenarios [12]. By leveraging feedback from the simulation outcomes, such as landing success rates, Bayesian optimization iteratively refines the model, ensuring it adapts effectively to varying vehicle and environmental states.

In this work, we address both challenges mentioned above in the context of perception systems for autonomous aerial vehicles, such as air taxis. Specifically, we focus on optimizing a *DNN*-based perception model for an automated landing system on a *Vertical Take-Off and Landing (VTOL)* aerial vehicle. The autonomous landing system employs a *DNN* object detector to identify the landing spot within a camera image frame. We propose an automated decision-making framework designed to enhance the utility of data samples collected for retraining, thereby improving the performance of the landing pad detector. This framework consists of two iterative phases. The first phase evaluates the overall performance of the vision-guided landing system and the uncertainties of the computer vision model given the operation scenarios in photorealistic simulation. In the second phase, the vehicle states and corresponding measure of cumulative landing success rate for the system are used in a Bayesian optimization loop to generate the next set of hyperparameter for retraining the model. Finally, the updated model is evaluated in the first phase. These phases close the loop in the data exploration process, as shown in Figure 1. To the best of our knowledge, this is the first use of such a framework for vision-based landing maneuvers in *VTOL*. Our key contribution can be summarized as follows:

- We develop a Bayesian retraining framework\* for the *YOLO* perception model [3] for vision-based autonomous landing within a photorealistic simulation environment based on the *CARLA* simulator [13], incorporating high-fidelity aerial vehicle dynamics from the *GUAM* vehicle simulation [14].

## II. Literature Review

Autonomous landing systems in *VTOL* vehicles depend on reliable landing pad detection. However, this detection is challenged by a wide range of variable environmental factors, such as weather conditions and lighting. Real-world datasets, particularly those for aerial vehicles, are unable to capture the full combinatorial complexity of all the factors that can impact the performance of landing pad detection. Synthetic data generation and data augmentation techniques have emerged as essential tools in the development of machine learning models to address this limitation of real-world datasets. The simulation-based methods enable the creation of diverse and customizable datasets, leading to improved resilience of models trained using these datasets. This section provides a review of related works on the components integrated into our proposed framework, including data augmentation, Bayesian optimization, and the use of *DNNs* in control applications.

- **Synthetic Data and Data Augmentation:** Synthetic data generation and data augmentation have become indispensable tools in enhancing the robustness of perception models for *VTOLs*. Docca and Torculas [7] highlight that synthetic data generated through simulation environments enables the creation of diverse image scenarios without the need for extensive manual data collection. These datasets enable model training on a wide variety of visual conditions. This helps the model generalize effectively in dynamic real-world environments. *Unity’s*

---

\*<https://github.com/arasul42/LCASL-TTU-vtol-bayes-sim>

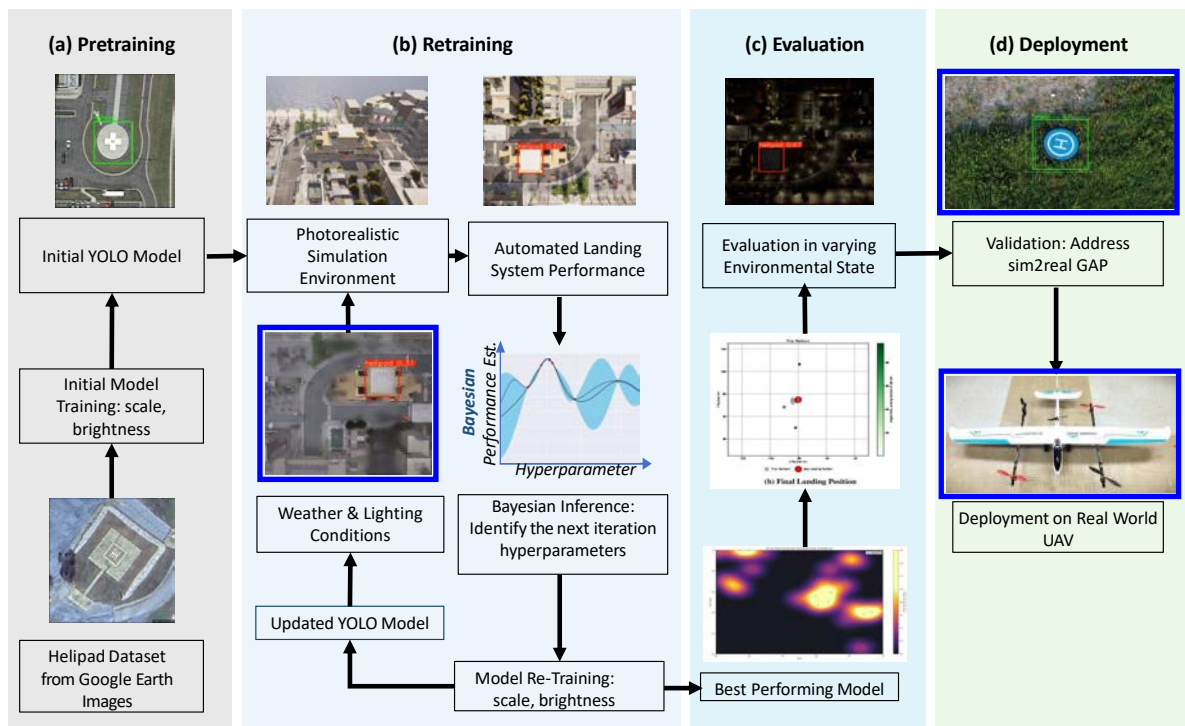


Fig. 1 Data Exploration Framework

simulated environments [9] contribute further by replicating complex landing scenarios and lighting conditions that are essential for robust VTOL perception. However, despite its flexibility, *Unity*'s simulated environments are not suitable for replicating the full complexity of real-world VTOL dynamics, as *Unity* is primarily designed for VR and game applications. Accurately modeling sensor noise, unpredictable environmental changes, and VTOL hardware limitations are the key difficulties that are crucial during operations like autonomous landings. While *Unity* excels in visual fidelity, the gap between simulated physics and real-world interactions necessitates further validation through real-world data or hybrid approaches combining simulations with digital twins [8].

- **Weather Augmentation:** Building on the strengths of synthetic data, weather augmentation adds a targeted approach to improving model robustness under adverse conditions. By generating realistic weather variations, such as rain, fog, and snow, during training, models become more robust and adaptable. Gupta et al. [6] highlight the importance of generating realistic weather variations, such as rain and fog, during training to enhance model robustness. By introducing weather-based transformations into datasets, models can generalize better to challenging real-world scenarios. Furthermore, techniques like adversarial augmentation add further complexity, which simulate edge cases that are difficult to replicate in physical experiments [15].
- **Digital Twins and Virtual Reality Environments:** While synthetic data focuses on creating individual variations, digital twins and Virtual Reality (VR) environments provide dynamic simulations of operational scenarios. Digital twins go beyond static synthetic data by simulating operational scenarios dynamically and integrating real-time environmental factors like wind and temperature [8]. Ensuring the fidelity of these twins, however, remains a challenge, especially for VTOLs operating in rapidly changing conditions. While synthetic data generates isolated variations, digital twins provide a comprehensive simulation of the VTOL's operational environment, incorporating variables such as lighting, weather, and terrain [8]. By leveraging these technologies, VTOL perception systems gain robustness, improving model reliability in dynamic environments. However, the fidelity of VR simulations can sometimes differ from real-world conditions, underscoring the need for iterative improvement [16].
- **Efficient Data Selection Techniques:** As the complexity of VTOL perception tasks grows, data selection becomes crucial in optimizing model training. Selecting the most informative data points ensures that the model focuses on samples that are difficult to learn, reducing both training time and computational load. Reducing the computational load of large-scale VTOL perception systems is essential. Techniques like *coreset* selection [17] and prioritized training [6] construct efficient datasets that balance size and representativeness. Mindermann et al. [18] introduce

a technique for prioritizing *learnable* and *informative* points, emphasizing that concentrating on challenging data can, in fact, enhance model accuracy. Similarly, Paul et al. [17] highlight *data pruning* techniques, which refine the training set by eliminating redundant examples, helping the model converge faster. Advanced data selection methods, such as those discussed by Guo et al. [19] and Sorscher et al. [20], use *coreset* selection to construct compact, representative datasets. These methods are especially valuable in VTOL perception, where reducing dataset size without compromising performance is crucial for model improvement. By prioritizing efficient data selection, VTOL models can maintain high performance while minimizing computational resource requirements.

- **Bayesian Inference and Optimization:** In complex VTOL perception systems, Bayesian inference offers a structured approach to quantify and incorporate uncertainty into model predictions, ultimately enhancing model robustness and adaptability. This approach is particularly beneficial in dynamic scenarios like VTOL landing, where conditions continuously evolve. Deng et al. [21] propose Bayesian data selection methods that leverage uncertainty estimates to refine training datasets, focusing on data points that improve model learning efficiency. Additionally, Shahriari et al. [22] provide a comprehensive review of Bayesian optimization techniques, underscoring its potential to iteratively adjust model parameters for better performance. For the VTOL landing tasks addressed in this work, we utilized Bayesian optimization [23] to fine-tune critical perception parameters, such as scale and brightness, achieving substantial improvements in detection accuracy across a wide range of environmental conditions.
- **Simulation Environments and Object Detection:** Advanced simulation environments like CARLA and object detection models like YOLO provide the infrastructure for rigorous testing of VTOL systems. CARLA offers detailed urban and rural scenarios with adjustable parameters such as lighting, weather, and environmental objects, enabling VTOLs to be tested in diverse landing contexts [13]. YOLO, known for its speed and accuracy in object detection, is particularly suited for real-time applications. Redmon and Farhadi's YOLO9000 model [3] and subsequent iterations have become widely used for VTOL applications, offering efficient and reliable detection capabilities in CARLA's complex simulated environments. By integrating YOLO models in CARLA, VTOL systems can gain real-world insights and refine detection accuracy under various conditions. This combination of CARLA's simulation environment and YOLO's detection capabilities represents a powerful toolkit for advancing safe VTOL landing technology.
- **Uncertainty Quantification and Adaptive Control:** Uncertainty quantification (UQ) enhances VTOL decision making by identifying high-confidence predictions, ensuring safety in dynamic conditions. UQ is critical in VTOL perception and control since it allows the system to estimate the reliability of its predictions in dynamic, often unpredictable environments. UQ helps identify high-confidence detections and ensures the VTOL only proceeds with safe, verified actions. Fernández Castaño et al. [24] demonstrate a UQ-based switching control approach that enables VTOLs to adapt their behavior based on environmental conditions, enhancing tracking accuracy. Adaptive control, together with UQ, offers a proactive response to uncertainty by adjusting control parameters in real time. For instance, Cao and Hovakimyan's  $\mathcal{L}_1$  adaptive control architecture [25] emphasizes rapid adaptation, ensuring stability and performance despite unpredicted changes. By combining UQ with adaptive control, VTOLs become better equipped for precise, reliable landing operations, even under uncertain conditions.
- **Neural Network-Based Control Methods:** Recent advancements in neural network-based control methods have shown promise in optimizing complex dynamical systems without the need for extensive retraining. Hose et al. [26] introduce a method for fine-tuning Approximate Model-Predictive Control (AMPC) using Bayesian Optimization (BO) to adapt neural network controllers efficiently. The proposed technique addresses a key challenge in AMPC: the need for retraining when system parameters change. By leveraging BO, the method allows for data-driven adaptation of AMPC, reducing manual tuning efforts and computational costs. The authors validated their approach on challenging control tasks, such as the cart-pole system and a balancing unicycle robot, demonstrating improved stability and reduced oscillations with minimal hardware experimentation. In a related approach, Ostafew et al. [27] applied a learning-based nonlinear Model-Predictive Control (MPC) to enhance vision-based path tracking for mobile robots in challenging outdoor environments. The study highlights the importance of integrating vision-based perception with advanced control strategies to handle unpredictable conditions. This technique utilized data-driven learning to adjust the controller in real-time based on visual feedback, significantly improving path-tracking performance. The insights from Ostafew et al.'s work are directly applicable to VTOL landing perception, where the integration of vision-based inputs and adaptive control methods is crucial for accurately detecting and landing on helipads in dynamic and uncertain environments.

The literature highlights the importance of a comprehensive approach that combines synthetic data generation, weather augmentation, advanced data selection techniques, simulation environments, and uncertainty-aware modeling



to enhance *VTOL* perception systems. In particular, weather augmentation bridges the gap between idealized training conditions and real-world variability, ensuring *VTOLs* can operate reliably in diverse and dynamic environments. By integrating these advancements, future *VTOL* systems will achieve increased robustness and operational safety, enabling dependable autonomous landings even in challenging conditions.

### III. Proposed Framework

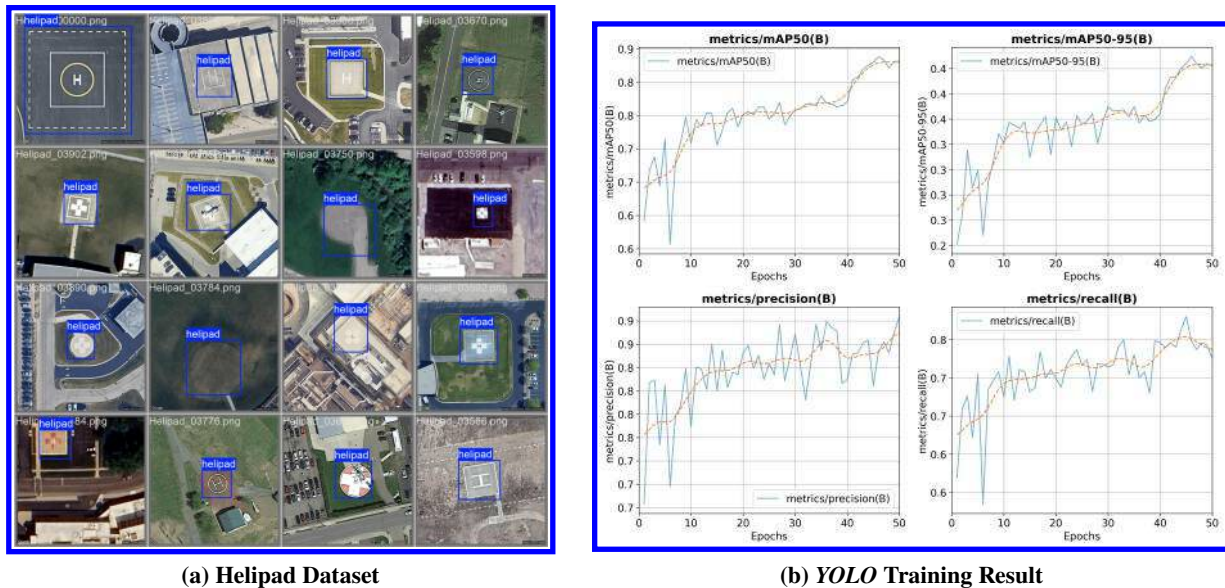
Developing perception systems for autonomous *VTOLs* demands a structured approach to address challenges such as data variability, hyperparameter optimization, and performance evaluation across diverse conditions. Our proposed framework addresses these challenges by integrating data augmentation, high-fidelity simulation environments, uncertainty quantification, and Bayesian optimization to enhance the performance of vision-based *VTOL* landing systems. This approach aims to iteratively adapt the model to new environments and enhance the landing success rate through systematic evaluation and retraining. The proposed framework, illustrated in Figure 1, comprises five main components: **Dataset Processing and Initial Model Training**, which involves preparing the dataset and training an initial *YOLO* model with reproducible results for helipad detection; **Uncertainty Estimation**, where prediction uncertainties are quantified to identify areas for model improvement and guide hyperparameter tuning; **Simulation Testing**, which evaluates the trained model in a high-fidelity, photorealistic simulation environment that incorporates vehicle dynamics; **Retraining Loop with Bayesian Optimization**, which iteratively refines the model by optimizing hyperparameters based on simulation feedback; and finally, **Validation Under Adverse Conditions**, where the model's robustness is tested under diverse weather and lighting scenarios to ensure reliable performance. The components are described below:

#### A. Dataset Processing and Initial Model Training

The importance of working with a high-quality dataset in training a model cannot be overstated, as the dataset fundamentally determines the model's learning and performance [7, 21]. Therefore, the first component of our framework focuses on data preprocessing and augmentation to train reproducible image detection *YOLO* models, specifically *YOLOv8* [28]. As dataset, we use *Google Earth* images of helipads [29]. *YOLO*'s fast inference time and efficiency make it well-suited for detecting objects in challenging real-time environments on mobile platforms [3]. *YOLO* models, such as *YOLOv5* and *YOLOv8*, employ random data augmentations during training to enhance model robustness and generalization. These augmentations include techniques like random resizing, flipping, cropping, color and brightness adjustments. By default, the randomness inherent in these augmentations leads to non-reproducible results across different training runs [30].

Reproducibility in empirical AI research is the ability of an independent research team to produce the same results using the same AI method based on the documentation made by the original research team [31]. Highly reproducible training ensures the same results are obtained over repeated iterations with the same environment, code and data [32]. To achieve reproducibility, we turn off the random augmentation of *YOLO* training, which gives us deterministic control over the image augmentation during the training cycle. This approach also allows us to isolate the effects of specific hyperparameters and observe the model's behavior on unseen data. However, this method of hyperparameter tuning has the inherent risk of overfitting the model that performs well only on that specific scenario, reducing its ability to generalize to unseen data. Our framework reduces the risk of overfitting by validating model performance under diverse lighting and weather conditions.

The training dataset [29] contains coordinates of helipads of different types, including their *Google Earth* URL, image size details, and ground truth bounding box annotations. The final dataset consists of approximately 4,000 images in JPEG format, each with a resolution of 640×640 pixels. The dataset labels are provided as the minimum and maximum coordinates of the bounding box (ground truth), which are not directly compatible with the open-source *YOLO* implementation used in this work [33]. *YOLO* requires annotations in a specific text file format, where bounding box coordinates are normalized relative to the image dimensions [3]. We transformed the label annotations to a *YOLO*-compatible format to prepare the dataset. Each annotation is stored in a plain text file corresponding to the image file, with rows representing individual objects and specifying their respective classes. Each annotation follows the format: <class\_id> <x\_center> <y\_center> <width> <height> where each value is normalized as per the *YOLO* format. Here, <class\_id> represents the object class, which is helipad in our case, <x\_center> and <y\_center> are the normalized coordinates of the bounding box centre relative to the image width and height and <width> and <height> are the normalized dimensions of the bounding box. After preprocessing, each image was paired with its corresponding annotation file in a directory, preparing the dataset for *YOLO* object detection training.



**Fig. 2 Initial Model Training**

From the dataset, 80% of the images were randomly selected for training, while the remaining 20% were reserved for validation. Figure 2a shows sample training images annotated with the *helipad* class bounding boxes, demonstrating the diversity and distribution of the dataset. The *YOLO* model outputs bounding box coordinates with confidence scores, indicating the likelihood of correct helipad detection. The model's performance was evaluated using standard object detection metrics, including mean Average Precision (mAP) at IoU (Intersection over Union) thresholds of 50% (mAP50) and a range from 50% to 95% (mAP50-95), as shown in Figure 2b. These metrics provide a comprehensive assessment of the model's detection accuracy. IoU measures the overlap between the predicted bounding box and the ground truth bounding box, serving as a criterion for correct detection [34]. Specifically, IoU is defined as the ratio of the intersection area to the union area of the two bounding boxes, with higher values indicating better alignment between the prediction and the ground truth [34]. Precision at a single IoU threshold (50%) is measured by mAP50, indicating how well the model detects objects at a basic level of overlap and mAP50-95 averages precision over multiple IoU thresholds [34]. This metric mAP50-95 provides a more rigorous evaluation of the model's performance, especially for scenarios requiring fine-grained object detection [34]. The curves in the top two charts illustrate that as training progresses over epochs, both mAP50 and mAP50-95 steadily increase, indicating improvements in the model's ability to detect helipads accurately. The precision and recall metrics, shown in the bottom two charts, provide further insight into the model's performance. Precision evaluates the proportion of correctly detected helipads out of all detections made, reflecting the model's ability to avoid false positives [34]. Recall measures the proportion of correctly detected helipads out of all actual helipads, indicating how well the model avoids false negatives [34]. These trends highlight the model's increasing generalization capability and robustness when applied to the validation set.

## B. Uncertainty Estimation

The initial model training process was repeated five times using dataset *Subsampling*, similar to the approach used in random forests [35]. *Subsampling* is a method where subsets of data are drawn randomly, without replacement, from the original dataset. This method allows the creation of multiple training datasets smaller than the original, effectively simulating the variability of data distributions. It is beneficial while working with small datasets, as it provides a way to estimate the model's robustness and reduces over-fitting by introducing diversity into training subsets. First, we separate the validation dataset (20% of the total dataset). Second, we randomly distribute the remaining 80% of the dataset to create five subsets of training data. For each *YOLO* model training, 4 out of 5 subsets are combined and used as the training dataset, as shown in Figure 3a. However, the validation set is kept the same to ensure consistency in evaluation. For each training step, the *YOLO* model was trained to detect helipads, producing five bounding boxes corresponding to the class *helipad*. The standard deviation of the five predicted bounding box coordinates provides a measure of uncertainty, as illustrated in Figure 3b. This uncertainty represents the epistemic uncertainty [36] of the system, which

is observed to be significant under lighting and weather conditions different from those in the training dataset.

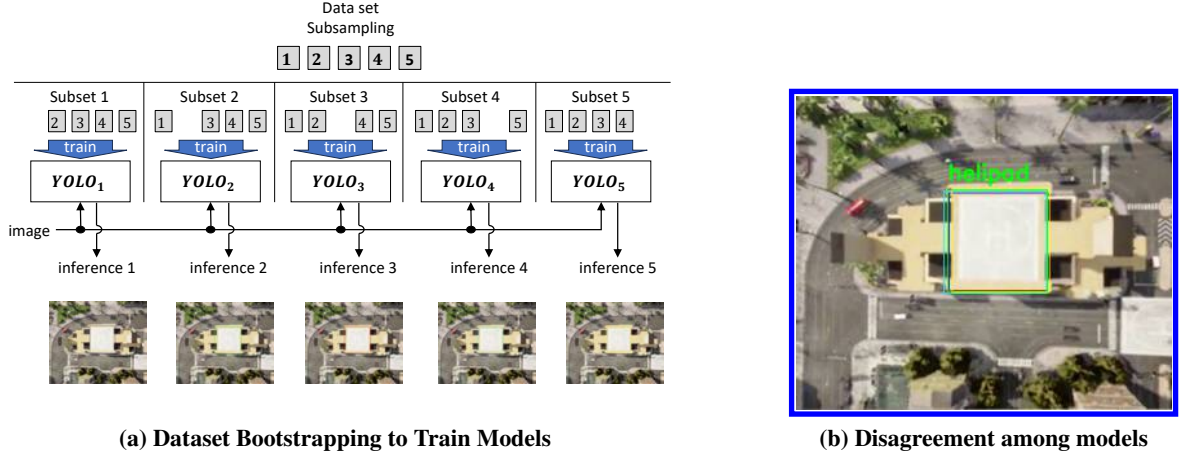


Fig. 3 Uncertainty Estimation in The System

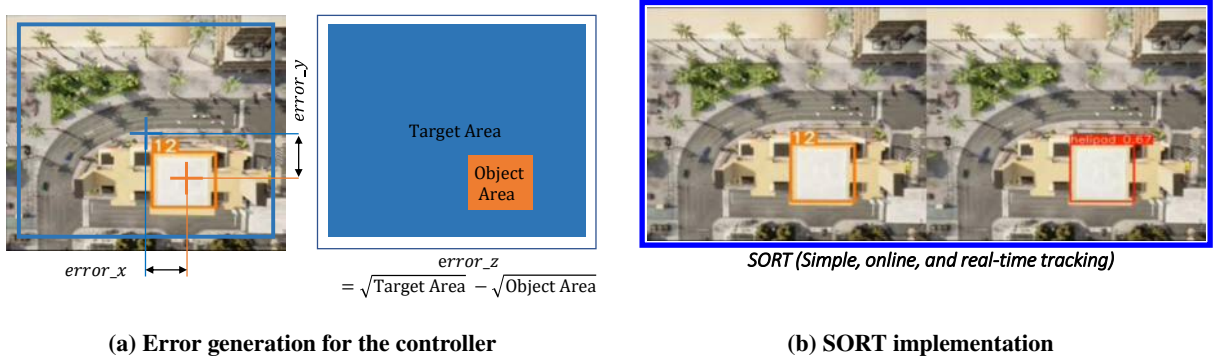
### C. Simulation Testing

The trained *YOLO* model is integrated into the high-fidelity photorealistic simulation environment *CARLA*, to evaluate its performance in controlled yet dynamic conditions. *CARLA*, built using the *Unreal Engine*, provides a photorealistic simulation of urban and rural environments, making it an ideal platform for testing autonomous systems [13, 37]. *CARLA* also lets us define many environmental conditions, which in turn lets us generate realistic-looking challenging conditions [13]. This flexibility facilitates a safe and repeatable testing environment without the risks associated with real-world *VTOL* operations, especially during the early stages of development [13]. The simulation specifically assessed the model's ability to detect helipads. The photorealistic rendering closely mimicked real-world challenges, providing a comprehensive testbed to evaluate the *YOLO* model's robustness. The next step involves simulating vehicle dynamics.

*Generic Urban Air Mobility (GUAM)*, developed by NASA, is a simulator designed for common rigid body 6-DOF (Degree of Freedom) frames of reference. It includes a wide range of aerospace signals and quantities of interest. The simulator's modular architecture readily supports the integration and replacement of aircraft models, sensors, actuator models, and control algorithms [14]. *GUAM* simulation is migrated from *MATLAB Simulink* to a *Python* environment using *Google's JAX* framework in [38]. *JAX* leverages GPU parallel computing and provides automatic differentiation, similar to *PyTorch* [39], enabling faster simulation speeds of up to 200 Hz. To integrate *CARLA* with *JAX-GUAM* and the *YOLO* detection model, we utilized *ROS* [40] as middleware. Once the systems are connected, the next step involves implementing a control algorithm. In *CARLA*, we mounted a downward-facing camera at the bottom of the aerial vehicle. The images captured by this camera serve as input for our landing system.

The bottom camera of the vehicle captures image frames that are streamed into the *YOLO* model to generate bounding boxes for the detected helipad. The position and size of the detected bounding box are compared to the camera's center bounding box (the target area), generating an error vector, as shown in Figure 4a. The camera center bounding box is represented by the blue box in the figure. To handle multi-object detection, we implemented the *SORT* (*Simple Online and Real-time Tracking*) method [41], as illustrated in Figure 4b. *SORT* is a multiple-object tracking algorithm specifically designed for video sequences. It ensures consistent tracking of the correct helipad bounding box in dynamic environments, even when the target is partially occluded or temporarily lost. With accurate object detection and *SORT*, the *VTOL* can continuously produce the necessary error signal for the control algorithm.

We define the landing control error as follows:  $\mathbf{e} = \mathbf{b}_c - \mathbf{b}_o$  where  $\mathbf{b}_c$  is the camera center bounding box and  $\mathbf{b}_o$  is the detected bounding box. Both the  $\mathbf{x}$  and  $\mathbf{y}$  components of the error can be found from the bounding box center deviation in the respective axis. For the  $\mathbf{z}$  component of the error, we take the size difference of the object compared to the target area (Camera Center). This error is high when *VTOL* is at a high altitude and low at the landing position. This method works if the landing pad size is less than the camera center bounding box at the landing position, which is 50 ft<sup>2</sup> in our



**Fig. 4 Object Detection, Tracking and Landing Controller**

case. This error is processed by a PID controller [42] to compute the corrective velocity vector,  $\mathbf{v}_{\text{corr}} = [v_x, v_y, v_z]^T$ :

$$v_i = K_p e_i + K_i \int e_i dt + K_d \frac{de_i}{dt}, \quad i \in \{x, y, z\},$$

where  $v_i$  is the corrective velocity along axis  $i$ ,  $e_i$  is the error along axis  $i$ , and  $K_p$ ,  $K_i$ ,  $K_d$  are the proportional, integral, and derivative gains for the respective axis [43]. The calculated velocity commands are constrained to prevent unmanageable overshoot and then fed into *JAX-GUAM*, which updates the vehicle's state, guiding it closer to the ideal landing position based on the corrective velocity vector  $v_i$ .

The landing performance evaluation consists of ten trials with random initial positions. A landing is considered successful if the vehicle is within 4 meters of the ideal center on the horizontal plane and within 1 meter on the  $z$ -axis. The success rate is calculated based on the number of successful landings out of 10 attempts. For example, if six trials are successful, the success rate is 60%.

#### D. Bayesian Optimization for Iterative Retraining of Perception DNN

The Bayesian optimization in this work aims to improve the landing success rate by iteratively retraining the object detection DNN (YOLO). In each iteration, the retraining hyperparameters are treated as decision variables in the optimization process. Unlike typical optimization problems that assume a known objective function or gradient, Bayesian optimization handles an unknown objective function, mapping hyperparameters to the landing controller's performance. In the Bayesian optimization, we estimate the unknown objective function, which evaluates the landing success rate, as a posterior distribution based on data samples of the hyperparameters and their corresponding objective function values (i.e., landing success rates). The hyperparameters considered include the scale factor and brightness value used in data augmentation during training data processing for each retraining iteration. As described in Algorithm 1, we use the acquisition function to identify the most promising data point to explore. The acquisition function balances the trade-offs between exploration and exploitation. This approach is particularly useful for optimizing black-box objective functions, making it highly effective for identifying the optimal combination of hyperparameters and augmentation strategies for an unknown objective function  $f(\mathbf{x})$  [22]. In our case,  $\mathbf{x}$  represents the hyperparameters and the augmentation parameters, such as scale and brightness.

The optimization aims to find the performance maximizer as follows:

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}),$$

where  $\mathcal{X}$  is the space of possible hyperparameters and augmentation combinations and  $f(\mathbf{x})$  is the landing success rate obtained from the photorealistic simulation environment.

We use the Gaussian Process (GP) to determine the posterior distribution given the data points obtained in data exploration. To use GP to model the unknown function  $f(\mathbf{x})$ , we assume  $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ . Here  $m(\mathbf{x})$  is the mean function which is defined as  $m(x) = \mathbb{E}[f(\mathbf{x})]$ .  $\mathbb{E}[f(\mathbf{x})]$  is the mean value that the function  $f(\mathbf{x})$  is assumed to take, given the prior distribution or as inferred from the data. If no prior information is available, the mean function



$m(\mathbf{x})$  is often set to zero for simplicity. And,  $k(\mathbf{x}, \mathbf{x}')$  is the covariance function (kernel), which defines the relationship between different combinations of hyperparameters [22].

After we set the upper and lower bounds for scaling and brightness hyperparameters. The next sample point is chosen based on an acquisition function  $a(\mathbf{x})$ , which balances exploration (sampling uncertain regions) and exploitation (sampling regions likely to yield high rewards):

$$\mathbf{x}_{n+1} = \arg \max_{\mathbf{x} \in \mathcal{X}} a(\mathbf{x}; \mathcal{D}_n)$$

where  $\mathcal{D}_n = \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^n$  is the set of previous observations [23]. One of the most widely used and simplest acquisition functions is the mean function plus the upper confidence bound (UCB). Where,

$$UCB(\mathbf{x}) = \mu(\mathbf{x}) + \kappa \cdot \sigma(\mathbf{x})$$

where  $\mu(\mathbf{x})$  is the posterior predictive mean,  $\sigma(\mathbf{x})$  is the posterior predictive standard deviation, and  $\kappa$  controls the exploration/exploitation trade-off. In our case, we used  $\kappa = 2.567$ , which encourages exploration by optimistically selecting new data points with high uncertainty, while also prioritizing areas with a high probability of finding the optimal solution. However, the value of  $\kappa$  can be adjusted to make the algorithm more exploratory or exploitative, depending on the desired balance.

After evaluating  $f(\mathbf{x}_{n+1})$ , the GP is updated to incorporate the new observation [44] into the data set:

$$\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{(\mathbf{x}_{n+1}, f(\mathbf{x}_{n+1}))\}.$$

Note that the GP is a non-parametric model; therefore, updating the dataset corresponds to updating the GP itself. The process is repeated until a stopping condition is met as:

$$\text{Stop if: } \max a(\mathbf{x}; \mathcal{D}_n) < \epsilon,$$

where  $\epsilon$  is a predefined threshold [23].

The Gaussian Process (GP) and acquisition function in Algorithm 1 were implemented using the *Python* package for Bayesian Optimization [45].

---

**Algorithm 1** Bayesian Optimization for Scale and Brightness

---

- 1: **Given:** Parameter space  $\mathcal{X} = \{(S, B) : S, B \in [0, 1]^2\}$
  - 2: **Given:** Success rate function  $f(S, B) = \text{EvaluateModel}(\text{TrainModel}(S, B))$
  - 3: **Given:** Surrogate function  $\hat{f}(S, B)$  modeled as  $\mathcal{GP}(m(x), k(x, x'))$
  - 4: **Given:** Acquisition function  $\alpha(S, B)$  (e.g., Upper Confidence Bound)
  - 5: **Given:** Initial samples  $\mathcal{X}_0 \leftarrow \text{RandomSample}(\mathcal{X}, K)$
  - 6: **Given:** Initial evaluations  $\mathcal{Y}_0 \leftarrow \{f(S, B) : (S, B) \in \mathcal{X}_0\}$
  - 7: Initialize dataset  $\mathcal{D}_0 \leftarrow \{(\mathcal{X}_0, \mathcal{Y}_0)\}$
  - 8: **for**  $n = 1, \dots, N$  **do**
  - 9:   Update surrogate function  $\hat{f}(S, B)$  using  $\mathcal{D}_n$
  - 10:    $(S_{\text{next}}, B_{\text{next}}) \leftarrow \arg \max \alpha(S, B | \mathcal{D}_n)$
  - 11:   Train model  $M \leftarrow \text{TrainModel}(S_{\text{next}}, B_{\text{next}})$
  - 12:   Evaluate success rate  $y_{\text{next}} \leftarrow \text{EvaluateModel}(M)$
  - 13:   Update  $\mathcal{D}_{n+1} \leftarrow \mathcal{D}_n \cup \{(S_{\text{next}}, B_{\text{next}}, y_{\text{next}})\}$
  - 14:   **if**  $\max \alpha(x; \mathcal{D}_n) < \epsilon$  **then**
  - 15:     **break** (Stopping condition met: no promising regions remain to explore)
  - 16:   **end if**
  - 17:   **if**  $y_{\text{next}} \geq \text{Success Threshold}$  **then**
  - 18:     **return** Optimal parameters  $(S_{\text{next}}, B_{\text{next}})$ , Success rate  $y_{\text{next}}$
  - 19:   **end if**
  - 20: **end for**
  - 21: **Output:** Best parameters  $(S^*, B^*) = \arg \max \mathcal{Y}$ , Best success rate  $y^* = \max \mathcal{Y}$
-

### E. Validation Under Weather and Lighting Conditions

After optimization, we evaluate the best object detection model under varying lighting conditions (e.g., night) and weather conditions (e.g., heavy rain). We measure the landing success rate from 10 random initial vehicle positions that range from 0-40 m offset on the horizontal plane and 0-120 m in the vertical direction. For evaluating the confidence score of object detection we consider the score from the extreme offset position (40 m, 40 m and 120m in  $x$ ,  $y$  and  $z$  direction respectively). We imposed the same criteria for successful landing described in the simulation testing section. The landing success rate is then incorporated into Algorithm 1 to identify the best-performing model for these specific environmental states. These optimizations generate hyperparameter spaces mapped to the landing success rate for each environmental condition. We obtained models that consistently perform with more than 70% landing success rates for each condition. In addition to the best-performing model for each environmental condition, we identify a high-performing parameter space shared across different landing scenarios. This common parameter space enables the training of a robust model that demonstrates improved performance across all evaluated lighting and weather conditions. The final evaluation shows a 70% landing success rate in clear day and night conditions and a 50% landing success rate at night with rain conditions. An illustrative video of the final landing evaluation is available here.<sup>†</sup>

## IV. Experiments and Result

### A. Photorealistic simulator

The photorealistic simulator in Figure 5a is developed using the existing autonomous car simulation application, *CARLA* [13]. Using the *Unreal Engine* [37], 3D rendering of an urban air mobility (UAM) vehicle is added, as shown in Figure 5a. The simulator integrates vehicle dynamics, controller, planning, and perception. The data-exploration decision-making is implemented through Simulation Control, as shown in Figure 5b. The simulation control chooses different weather conditions and lighting conditions, as shown in Figure 6.

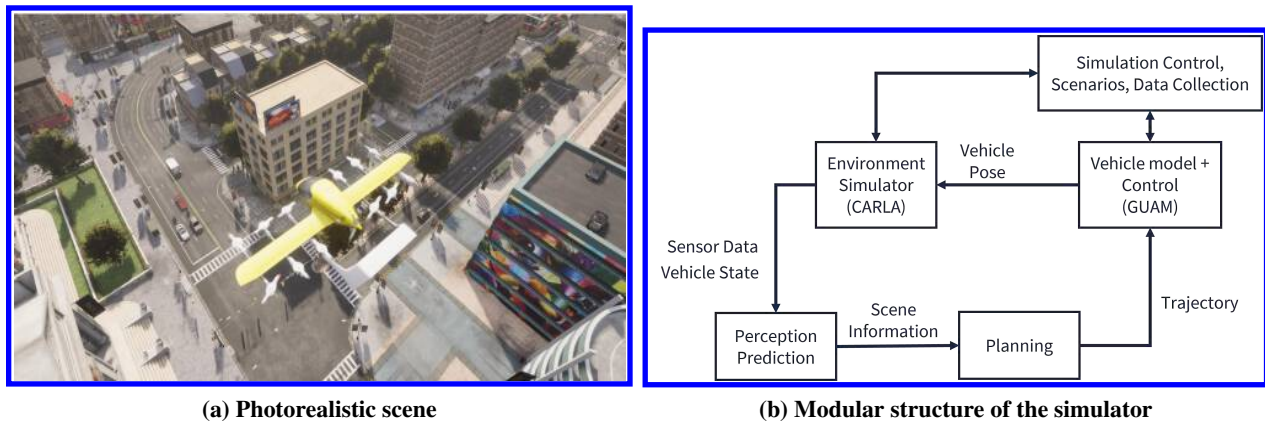
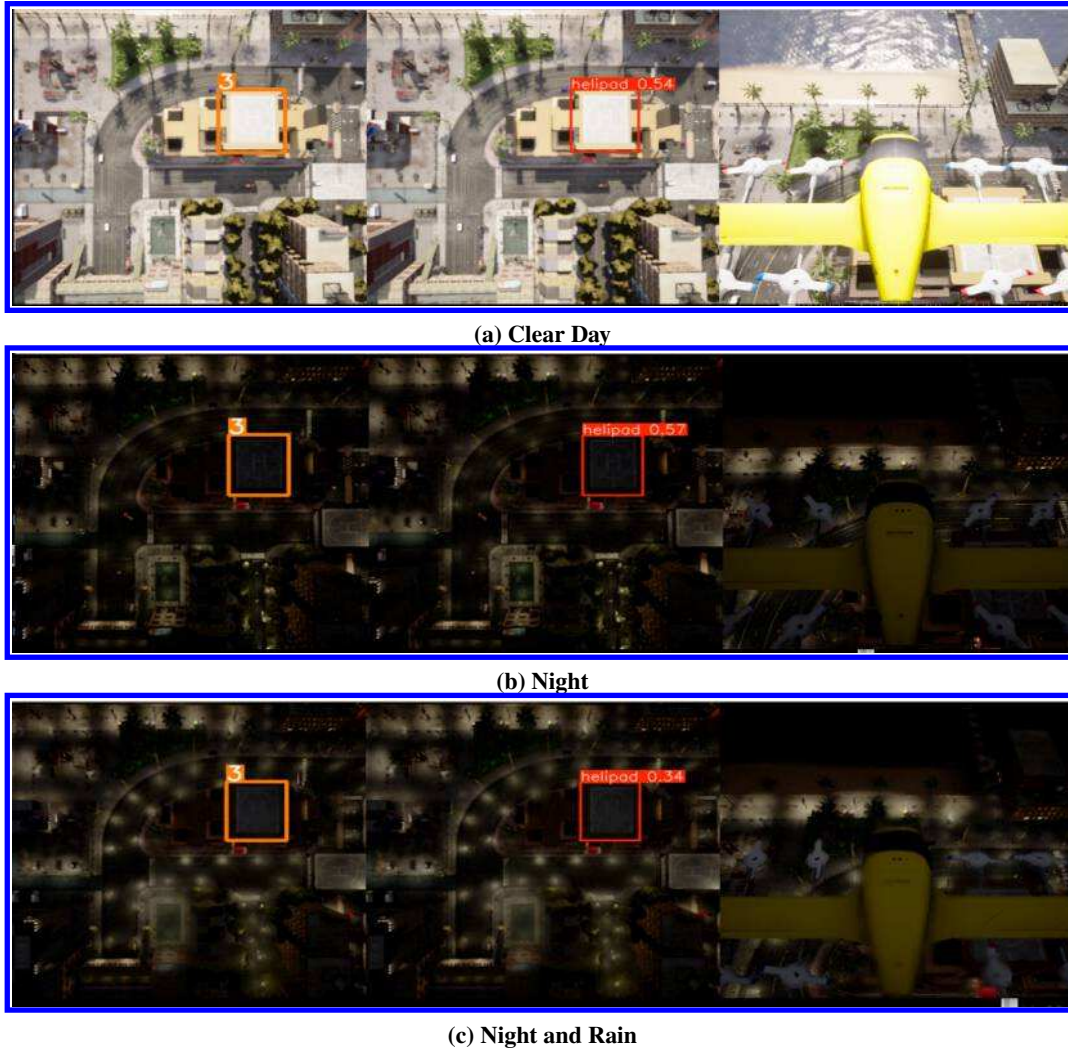


Fig. 5 Integration of perception, planning, and control with the simulator

### B. Result

The proposed framework was evaluated using key performance metrics, including mean average precision (mAP), confidence scores, and landing success rates, to assess its effectiveness in detecting helipads and guiding safe VTOL landings. The initial YOLOv8 model was trained using randomly selected augmentations (scale and brightness). During the evaluation, the model achieved an mAP (50-95) of 43%, which is close to the benchmark mAP (50-95) for YOLOv8 [33], typically reported as 44.9% for large-scale datasets, such as the COCO dataset [34]. The pre-trained model was integrated into CARLA to assess the landing success rate from 10 randomly generated initial positions, as shown in Figure 7a. The initial model achieved a success rate of 50%, as illustrated in Figure 7b, under clear, bright weather conditions in CARLA. This is a baseline evaluation of the model's performance. The Bayesian optimization algorithm 1 uses the evaluation outcome, which consists of a pair of data augmentation hyperparameters (scale and

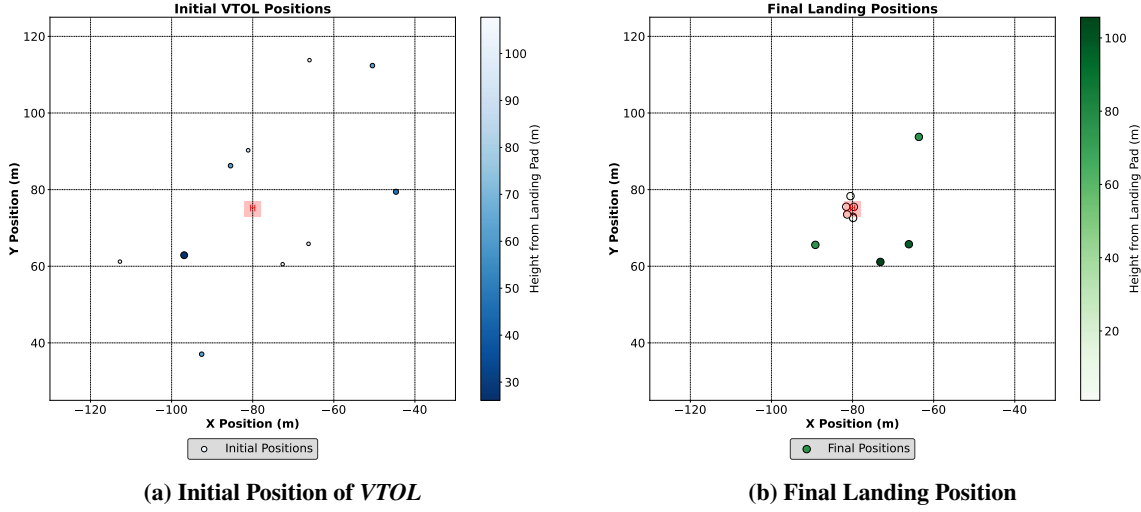
<sup>†</sup>[https://www.youtube.com/watch?v=WAGicg\\_xGj8](https://www.youtube.com/watch?v=WAGicg_xGj8)



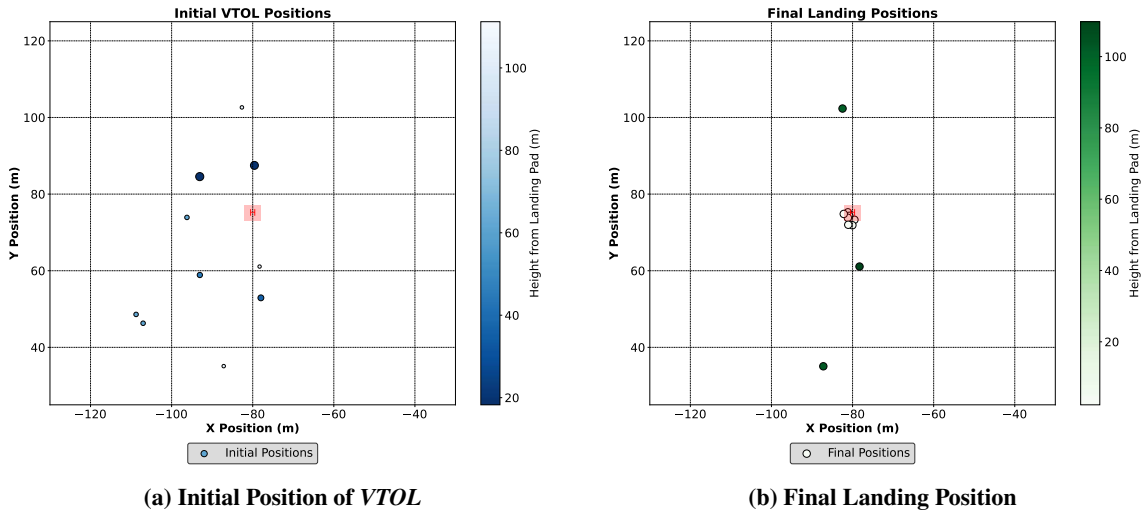
**Fig. 6 Simulation for different landing scenarios.**

brightness) and the corresponding success rate. The optimization process begins by collecting the evaluation outcomes, forming a closed loop of evaluation and optimization steps in each iteration. After conducting 30 iterations, the landing success rate improved from the baseline of 50% to 70%, as shown in Figure 8. This experimental result demonstrates that the proposed Bayesian data augmentation approach can improve the landing success rate by identifying more suitable data augmentation hyperparameters.

To better understand the selection of the hyperparameters in the Bayesian optimization, we plotted a contour plot showing the estimated landing success rate for each pair of scale and brightness (estimated by the GP in Bayesian optimization), as shown in Figure 9. In Figure 9, we present the contours for different weather conditions: (a) clear day; (b) clear night; (c) night and rain. As shown in Figure 9a, there are several pairs of scale and brightness augmentations that achieve the same landing success rate. Next, we selected one of these pairs and tested it under different lighting conditions (clear night). However, we achieved a zero landing success rate under this condition with the hyperparameters that worked well in the other conditions. This example demonstrates that Bayesian optimization, when performed under favorable conditions, may carry the risk of failure, in contrast to optimization conducted under more challenging or adverse conditions. Next, we applied Bayesian optimization under two adverse conditions. First, we ran Algorithm 1 under clear night conditions, achieving a landing success rate of 70% and identifying the corresponding best-performing parameters, as shown in Figure 9b. Then, we ran the algorithm under rain and night conditions to obtain the relevant set of hyperparameters, as shown in Figure 9c. As shown in Figure 9, the hyperparameters  $\text{scale}=0.77$  and  $\text{brightness}=0.66$  are jointly shared across all three environmental states as high-performing parameters. As a result, the model achieves a



**Fig. 7 Landing Performance without Optimization**



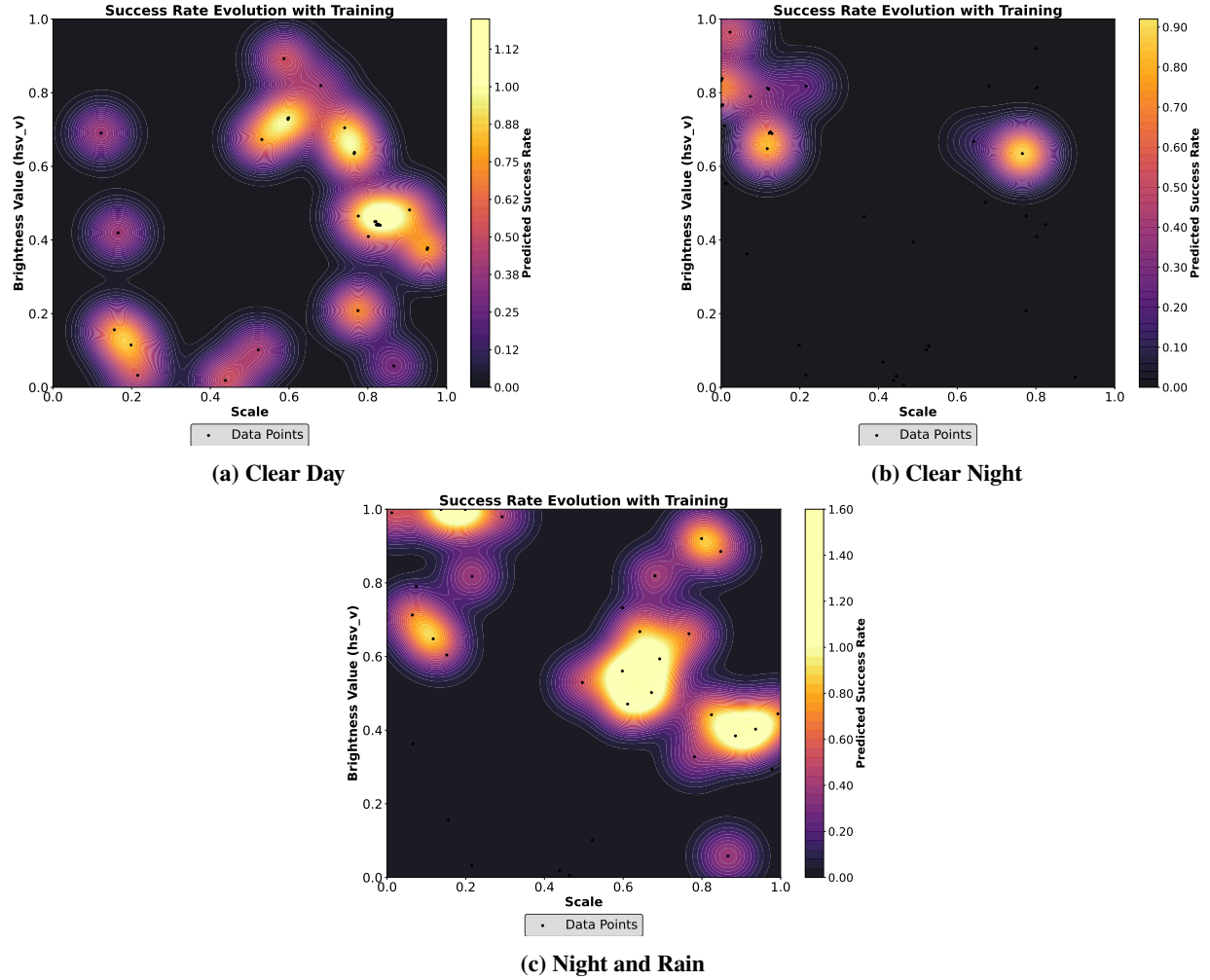
**Fig. 8 Landing Performance after Optimization (20% Improvement in Landing Success Rate)**

70% landing success rate in both day and night conditions, and a 50% success rate in night with rain conditions. This suggests the potential of using adverse conditions to improve the robustness of the perception model. Furthermore, during the final validation of the perception model, we have observed the confidence scores for bounding boxes also showed enhanced reliability, even under varying lighting and weather conditions. The confidence score improved from 30% to 50% in clear day and from 0% to 50% in clear night landing scenario while evaluated from extreme offset position. These results emphasize the effectiveness of the proposed framework in improving VTOL-specific perception and control tasks.

## V. Conclusion and Future Work

This work introduces a novel framework for enhancing VTOL perception systems that focuses on vision-guided landing performance. By integrating reproducible training of DNN model, uncertainty estimation, simulation testing and Bayesian optimization, the framework addresses challenges related to dataset variability, environmental unpredictability and model refinement. The Bayesian data augmentation framework improved the landing success rate from the baseline of 50% to 70%, demonstrating its ability to adapt and enhance model performance under dynamic conditions. This framework highlights how domain-specific optimizations can bridge the gap between general-purpose object detection





**Fig. 9 Best Performing Parameters for Landing**

and specialized *VTOL* tasks, outperforming typical *YOLO* object detection benchmark models.

Future work will focus on expanding the dataset, testing under additional environmental conditions, and transitioning to real-world *VTOL* operations. By addressing these aspects, the framework has the potential to significantly advance the development of reliable, autonomous *VTOL* systems capable of safe and efficient landings in complex environments. Building on these promising simulation results, our next step is to test the model in real-flight scenarios using the platform currently under development, as shown in Figure 10.



**Fig. 10 Prototype Design for Real Flight Experiment**

### Acknowledgments

This material is based upon work supported by the National Aeronautics and Space Administration (NASA) under the cooperative agreement 80NSSC20M0229 and University Leadership Initiative grant no. 80NSSC22M0070, and the National Science Foundation (NSF) under grant no. CNS 1932529 and ECCS 2311085.

Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

### References

- [1] Grigorescu, S., Trasnea, B., Cocias, T., and Macesanu, G., "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, Vol. 37, No. 3, 2020, pp. 362–386.
- [2] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [3] Redmon, J., and Farhadi, A., "YOLO9000: better, faster, stronger," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
- [4] Ultralytics, "YOLOv5: You Only Look Once version 5," <https://github.com/ultralytics/yolov5>, 2020.
- [5] Ultralytics, "YOLOv8: You Only Look Once version 8," <https://github.com/ultralytics/ultralytics>, 2023.
- [6] Gupta, H., Kotlyar, O., Andreasson, H., and Lilienthal, A. J., "Robust Object Detection in Challenging Weather Conditions," *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 7523–7532.
- [7] Docca, A., and Torculas, M., "How to Train a Defect Detection Model Using Synthetic Data with NVIDIA Omniverse Replicator," <https://developer.nvidia.com/blog/how-to-train-a-defect-detection-model-using-synthetic-data-with-nvidia-omniverse-replicator>, 2023. Accessed: 2024-05-22.
- [8] Dallel, M., Havard, V., Dupuis, Y., and Baudry, D., "Digital twin of an industrial workstation: A novel method of an auto-labeled data generator using virtual reality for human action recognition in the context of human–robot collaboration," *Engineering applications of artificial intelligence*, Vol. 118, 2023, p. 105655.

- [9] Technologies, U., “Efficient Development of Simulated Environments for Autonomous Vehicle Training,” <https://unity.com/how-to/simulated-environments-for-autonomous-vehicle-training>, 2023. Accessed: 2023-08-22.
- [10] Thompson, N. C., Greenewald, K., Lee, K., and Manso, G. F., “The computational limits of deep learning,” *arXiv preprint arXiv:2007.05558*, Vol. 10, 2020.
- [11] Frazier, P. I., “A tutorial on Bayesian optimization,” *arXiv preprint arXiv:1807.02811*, 2018.
- [12] Victoria, A. H., and Maragatham, G., “Automatic Tuning of Hyperparameters Using Bayesian Optimization,” *Evolving Systems*, Vol. 12, 2021, pp. 217–223.
- [13] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V., “CARLA: An Open Urban Driving Simulator,” *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
- [14] NASA, “Generic Urban Air Mobility (GUAM),” <https://github.com/nasa/Generic-Urban-Air-Mobility-GUAM>, 2024. Accessed: November 23, 2024.
- [15] Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A., and Le, Q. V., “ADA: Adversarial Data Augmentation for Object Detection,” *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, 2019, pp. 7364–7373.
- [16] Mishra, N., Sieb, M., Abbeel, P., and Chen, X., “Closing the Visual Sim-to-Real Gap with Object-Composable NeRFs,” *arXiv preprint arXiv:2403.04114*, 2024. URL <https://arxiv.org/abs/2403.04114>.
- [17] Paul, M., Ganguli, S., and Dziugaite, G. K., “Deep learning on a data diet: Finding important examples early in training,” *Advances in Neural Information Processing Systems*, Vol. 34, 2021, pp. 20596–20607.
- [18] Mindermann, S., Brauner, J. M., Razzak, M. T., Sharma, M., Kirsch, A., Xu, W., Hölting, B., Gomez, A. N., Morisot, A., Farquhar, S., et al., “Prioritized training on points that are learnable, worth learning, and not yet learnt,” *International Conference on Machine Learning*, PMLR, 2022, pp. 15630–15649.
- [19] Guo, C., Zhao, B., and Bai, Y., “Deepcore: A comprehensive library for coreset selection in deep learning,” *International Conference on Database and Expert Systems Applications*, Springer, 2022, pp. 181–195.
- [20] Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., and Morcos, A., “Beyond neural scaling laws: beating power law scaling via data pruning,” *Advances in Neural Information Processing Systems*, Vol. 35, 2022, pp. 19523–19536.
- [21] Deng, Z., Cui, P., and Zhu, J., “Towards Accelerated Model Training via Bayesian Data Selection,” *Advances in Neural Information Processing Systems*, Vol. 36, 2024.
- [22] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N., “Taking the human out of the loop: A review of Bayesian optimization,” *Proceedings of the IEEE*, Vol. 104, No. 1, 2015, pp. 148–175.
- [23] Snoek, J., Larochelle, H., and Adams, R. P., “Practical bayesian optimization of machine learning algorithms,” *Advances in neural information processing systems*, Vol. 25, 2012.
- [24] Fernández Castaño, A., Patton, C., Yoon, H. J., and Voulgaris, P., “Uncertainty Quantification-Based Switching Control Method for Vision-Based Object Tracking in Unmanned Aerial Vehicles,” *AIAA SCITECH 2024 Forum*, 2024, p. 0516.
- [25] Cao, C., and Hovakimyan, N., “Design and analysis of a novel  $\mathcal{L}_1$  adaptive control architecture with guaranteed transient performance,” *IEEE Transactions on Automatic Control*, Vol. 53, No. 2, 2008, pp. 586–591.
- [26] Hose, H., Brunzema, P., von Rohr, A., Gräfe, A., Schoellig, A. P., and Trimpe, S., “Fine-Tuning of Neural Network Approximate MPC without Retraining via Bayesian Optimization,” *CoRL Workshop on Safe and Robust Robot Learning for Operation in the Real World*, 2024. URL <https://openreview.net/forum?id=1Sah6an1Ar>.
- [27] Ostafew, C. J., Schoellig, A. P., and Barfoot, T. D., “Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments,” *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4029–4036. <https://doi.org/10.1109/ICRA.2014.6907444>.
- [28] Terven, J., and Cordova-Esparza, D., “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” *arXiv preprint arXiv:2304.00501*, 2023. URL <https://arxiv.org/abs/2304.00501>.
- [29] Bitoun, J., and Winkler, S., “HelipadCat: Categorised Helipad Image Dataset and Detection Method,” *Proceedings of the IEEE Region 10 Conference (TENCON)*, IEEE, 2020, pp. 685–689.

- [30] Ultralytics, *Ultralytics YOLO Documentation: Configuration*, 2024. URL <https://docs.ultralytics.com/usage/cfg/>, accessed: 2024-12-01.
- [31] Tatman, R., Vanderplas, J., and Dane, K., “State of the Art: Reproducibility in Artificial Intelligence,” *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 1–10.
- [32] Tatman, R., VanderPlas, J., and Dane, S., “A practical taxonomy of reproducibility for machine learning research,” , 2018.
- [33] Ultralytics, “YOLOv8,” <https://github.com/ultralytics/ultralytics>, 2023.
- [34] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., “Microsoft coco: Common objects in context,” *European conference on computer vision*, Springer, 2014, pp. 740–755.
- [35] Breiman, L., “Random forests,” *Machine Learning*, Vol. 45, No. 1, 2001, pp. 5–32.
- [36] Hüllermeier, E., and Waegeman, W., “Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods,” *arXiv preprint arXiv:1910.09457*, 2019. URL <https://arxiv.org/abs/1910.09457>.
- [37] “Unreal Engine,” , 2021. URL <https://www.unrealengine.com/>, accessed: 2021-12-01.
- [38] So, O., “JAX-GUAM: General Uncertainty Approximation Models,” [https://github.com/oswinso/jax\\_guam](https://github.com/oswinso/jax_guam), 2023.
- [39] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” *Advances in Neural Information Processing Systems* 32, Curran Associates, Inc., 2019, pp. 8024–8035. URL <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [40] Stanford Artificial Intelligence Laboratory et al., “Robotic Operating System,” , 2018. URL <https://www.ros.org>.
- [41] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B., “Simple online and realtime tracking,” *2016 IEEE international conference on image processing (ICIP)*, IEEE, 2016, pp. 3464–3468.
- [42] Åström, K. J., and Murray, R., *Feedback systems: an introduction for scientists and engineers*, Princeton university press, 2021.
- [43] Şumnu, A., Güzelbey, İ. H., and Çakir, M. V., “Simulation and PID control of a Stewart platform with linear motor,” *Journal of mechanical science and technology*, Vol. 31, 2017, pp. 345–356.
- [44] Rasmussen, C. E., and Nickisch, H., “Gaussian processes for machine learning (GPML) toolbox,” *The Journal of Machine Learning Research*, Vol. 11, 2010, pp. 3011–3015.
- [45] Nogueira, F., “Bayesian Optimization: Open source constrained global optimization tool for Python,” , 2014–. URL <https://github.com/bayesian-optimization/BayesianOptimization>.