VIRIATO: Visual-Action Reinforcement Integrator for Actor-Critic with Temporal Observations

João Campanhã*†, Francisco Neves†, Benedita Malheiro*†, Andry Pinto†‡
*ISEP, Polytechnic of Porto, Porto, Portugal
†INESC TEC, Institute for Systems and Computer Engineering, Technology and Science, Porto, Portugal
‡FEUP, Engineering Faculty, University of Porto, Porto, Portugal

***Abstract*—The Visual-Action Reinforcement Integrator for Actor-Critic with Temporal Observations (VIRIATO) is a compact multi-input feature-extractor architecture designed to enable robust visual navigation of Unmanned Aerial Vehicles (UAVs) conducting close-range inspection of photovoltaic arrays. The target task of low-altitude flight over dynamic, visually variable surfaces without privileged information is inherently partially observable. VIRIATO augments stacked image observations with a short history of recent past actions, producing a richer latent state for the Soft Actor-Critic (SAC) agent. Training is performed with domain randomization to expose the policy to diverse lighting, backgrounds and panel layouts. In simulation, VIRIATO yields faster learning and improved sample efficiency compared to a standard image-only Convolutional Neural Network (CNN) feature extractor, achieving lower position and yaw errors and substantially better robustness under image perturbations while retaining high task completion rates. The architecture is intentionally simple and general: it improves temporal awareness without adding complex recurrence, and it could be adapted to other perception-driven robotic tasks. These results demonstrate that integrating historical action data with visual encoding, together with domain randomization, is an effective way to achieve reliable autonomous vision-based navigation.**

***Index Terms*—Reinforcement Learning (RL), Visual navigation, Unmanned Aerial Vehicles (UAVs), Domain Randomization, Photovoltaic (PV) array inspection, Simulation.**

I. Introduction

Photovoltaic (PV) power is expanding rapidly, with large-scale PV farms emerging as key components of a decarbonized electricity system. Global cumulative PV capacity has surpassed 2 TW for the first time [1] and is expected to sustain robust growth in the coming years.

At utility scale, Operation and Maintenance (O&M) is essential to maintain energy yield and long-term asset value. Traditional manual inspections are time-consuming, labor-intensive, and can require shutdowns, creating bottlenecks as PV capacity and plant sizes increase. Aerial inspection using UAVs has emerged as a practical alternative: industry reports and case studies show that UAV-based thermal and visual surveys can cover large areas orders of magnitude faster than ground-based manual inspections at significantly lower cost [2]. Reported case studies indicate substantial gains in inspection speed and cost when using UAVs for utility-scale PV monitoring [2, 3].

Although UAV-based inspections significantly reduce time and operational costs compared to traditional surveys, dependence on human pilots is inefficient for scaling routine operations [4]. Moreover, regulation in Europe currently requires visual Visual Line-Of-Sight (VLOS) for most operations, unless specific Beyond Visual Line-Of-Sight (BVLOS) approvals are granted [5], which restricts the continuous large-scale deployment of human-piloted missions.

Mission planning software, reliant on accurate Global Navigation Satellite Systems (GNSS) inputs, has been essential for covering large inspection areas, typically carried out at higher altitudes [6]. This approach can be effective in environments where accurate GNSS inputs are available. However, this is not the case for the PV farm in the Alqueva reservoir shown in Figure 1. The farm, Europe's largest floating solar plant in a reservoir, is subject to positional drift, and the main practical limitation is the moving-panel geometry. Using traditional mission planning software, these areas can only be inspected if the UAV operates at a higher altitude to cover the entire site, resulting in low detail images, possibly missing panel flaws that would otherwise be detected.

Fig. 1: The floating PV solar plant at the Alqueva reservoir.

In such cases, flying an UAV at low altitude would allow the capture of high-resolution images for effective fault detection. However, the camera's Field Of View (FOV) must be precisely aligned with the panels to ensure optimal data quality. In this scenario, alignment could be achieved through autonomous navigation guided by a visual camera.

II. Related Work

UAVs can be equipped with visual, Infrared (IR) and Electroluminescence (EL) sensors, each capturing complementary fault signatures in PV inspection [7]. Effective use of these

sensing modalities requires automation to reduce costs and maintain data quality [8].

Typically, flight planning is simplified by operating at higher altitudes using Boustrophedon paths[1], guided by predefined GNSS waypoints. This approach creates an orthomosaic by overlaying images, which can be examined later. Although this approach offers a comprehensive overview of the scene, high-altitude flight limits the resolution of individual images and is ineffective over empty zones [10].

Numerous studies confirm the value of high-resolution photos in PV plant inspection [8]. A flight campaign at $30\,\mathrm{m}$ and $80\,\mathrm{m}$ above ground conducted by Gallardo-Saavedra et al. [11] detected only $37\,\%$ and $24\,\%$ of anomalies respectively, compared with close-up manual inspections, underscoring the impact of resolution on fault detection. This need for high-resolution imaging drives the evolution of UAVs from manual or semi-automatic piloting toward fully autonomous close-range operations [4].

Several studies have proposed solutions to meet these requirements. Morando et al. [10] combine classical computer vision techniques for solar panel detection and segmentation with a traditional Proportional Integral Derivative (PID) controller to keep the UAV aligned with the center of the panel during flight. Their approach integrates waypoint navigation with real-time PV panel detection, using thermal and visual imaging, to achieve optimal inspection efficiency and data quality. Roggi et al. [12] explores visual servoing to inspect PV panel arrays with GNSS position errors, using the camera FOV along with classical computer vision techniques such as Canny Edge and Hough Lines. The results show fewer frames with incomplete PV coverage compared to a non-corrective approach.

However, these approaches lack robustness to variations in lighting and environmental conditions, a common limitation of conventional methods [13]. At the time of writing, no studies apply end-to-end deep reinforcement learning on UAVs to integrate perception and control for robust navigation during PV panel array inspections. This gap highlights the relevance of the present contribution.

III. Visual-Action Reinforcement Integrator for Actor-Critic with Temporal Observations

A. Problem Statement

The UAV, represented by the body frame $\mathcal{F}_b = \{x_b, y_b, z_b\}$, will be controlled using high-level velocity commands in the body frame, specifically ω_z and v_y, instead of directly controlling the rotor speeds, abstracting the low-level control problem. The UAV will maintain a fixed altitude z and a constant forward velocity v_x. This design isolates the lateral/yaw alignment behavior that dominates close-range panel following in the target dam scenario, reducing confounding factors and improving training stability. Therefore, the autonomy addressed in this work is scoped to visual lateral centering and yaw correction along a predefined inspection sweep. Although controlling v_x and z could support active perception, this was intentionally left out to keep the benchmark focused and to enable controlled comparison across methods. It will navigate relative to the world frame $\mathcal{F}_w = \{x_w, y_w, z_w\}$ and will be equipped with a gimbal, allowing configuration of the position of the camera frame $\mathcal{F}_c = \{x_c, y_c, z_c\}$, as shown in Figure 2.

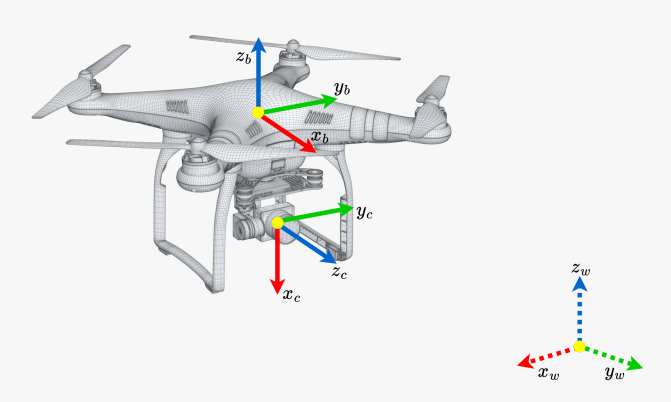


Fig. 2: Camera and body frames of the UAV, shown with respect to the world frame.

In this context, the UAV position relative to the PV panels is estimated from the visual camera, which serves as the primary observation source for the agent state. Because sensor readings only partially observe the full system state, the problem is formulated as a Partially Observable Markov Decision Process (POMDP) using a model-free RL agent. As model-free methods do not require knowledge of the transition dynamics $P(s'|s,a)$, the problem can be represented by the tuple $(\mathcal{S}, \mathcal{A}, R, \mathcal{O}, Z)$, where:

- $\mathcal{S}$: Set of true state space, where $S_t = (e_d, e_\psi)$ is defined by the displacement magnitude error and the angular error, respectively. The agent will be trained to operate within an interval range $e_{d,t} \in [0, e_{d_{\max}}]$ and $e_{\psi,t} \in [0, e_{\psi_{\max}}]$. This is illustrated in Figure 3.
- $\mathcal{A}$: Set of action space of high-level body velocity commands, which is defined by $A_t = (v_{y,t}, \omega_{z,t})$, with $v_{y,t} \in [v_{y_{\min}}, v_{y_{\max}}]$ and $\omega_{z,t} \in [\omega_{\min}, \omega_{\max}]$. The action v_x, will not be included, since it will be fixed, with a value of $0.25\,\mathrm{m\,s^{-1}}$, which is an adequate velocity for the inspection.
- R: The reward function $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$, with $\mathcal{S}$ and $\mathcal{A}$ corresponding to the state and action spaces, depends primarily on the current state S_t and action A_t. This is formally defined in Equation 3.
- O: The observation space of VIRIATO is an approximation of the agent's history at time t, $O_t = (I_t, A_t) \approx H_t$, where:
  - $I_t \in [0,1]^{H \times W \times F}$ is a stack of normalized grayscale pixel images, H and W are the height and width of each frame and F is the number of stacked frames.

[1] Alternating-direction movement pattern used for coverage tasks in robotics such as vacuuming, mapping, area scanning, and inspection [9].

- $A_t \in [-1,1]^{K\times 2}$ is a stack of the previous K normalized action vectors and each action consists of linear and angular velocities (v_y, ω_z) scaled to the range $[-1,1]$.

- Z: The observation mapping Z uses a feature extractor f to map raw observations O_t to latent embeddings $z_t := f(O_t)$. These embeddings approximate the agent's belief by accounting for environment dynamics rather than relying solely on the previous action.

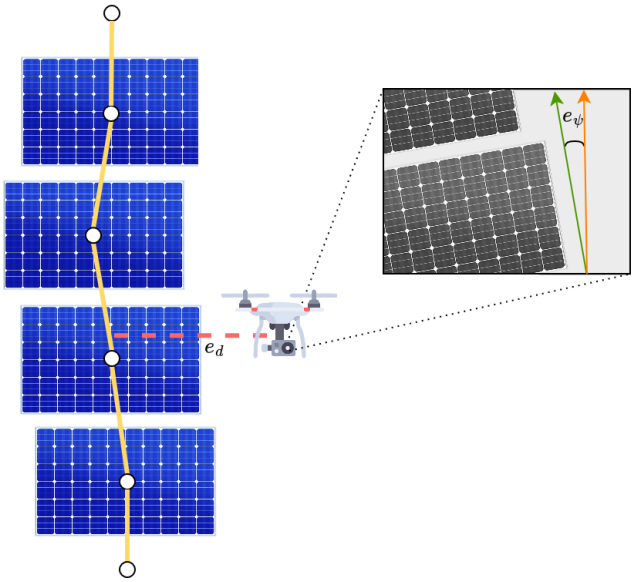


Fig. 3: Visualization of the true state errors e_d and e_ψ.

B. Reward Function

The RL reward function was defined to encourage a reduction of the true state–space errors at each time step t ($e_{d,t}$, $e_{\psi,t}$), while simultaneously increasing the individual rewards $r_{\text{pos},t}$ and $r_{\text{yaw},t}$, respectively. As shown in Equation 1, these reward components are computed using a zero-mean Gaussian function with a task-specific standard deviation σ, chosen according to the scale of the corresponding error, such that $\exp(\cdot) \in [0,1]$.

$$\begin{aligned} r_{\text{pos},t} &= \exp\!\left(-\frac{e_{d,t}^2}{2\,\sigma_d^2}\right), \\ r_{\text{yaw},t} &= \exp\!\left(-\frac{e_{\psi,t}^2}{2\,\sigma_\psi^2}\right), \end{aligned} \tag{1}$$

The smoothness reward component is computed differently. First, the quadratic difference between the current action a_t and the previous action a_{t-1} is calculated, yielding $\Delta a_t^2 = (a_t - a_{t-1})^2$. Next, this difference is normalized and inverted over the full action range, producing an error vector $e_{\text{diff\_norm}}$ whose elements lie in $[0,1]$, where 0 indicates an abrupt change and 1 indicates no change in the action. These normalized values are then weighted by a diagonal importance matrix $A = \text{diag}(k_{\text{pos}}, k_{\text{yaw}})$, composed of constants k_{pos} and k_{yaw} satisfying $\sum_i k_i = 1$, and summed to yield the smoothness reward r_{smooth}, as shown in Equation 2.

$$\begin{aligned} \Delta a_t^2 &= (a_t - a_{t-1})^2 \\ e_{\text{diff\_norm}} &= 1 - \frac{\Delta a_t^2}{m_{\text{diff}}} \\ A &= \text{diag}(k_{pos}, k_{yaw}) \\ r_{\text{smooth}} &= \sum_{i=1}^{n} (A_{ii} \cdot (e_{\text{diff\_norm}})_i) \end{aligned} \tag{2}$$

If the agent reaches the end of the panel array within the predefined bounds, it receives a large bonus β. Conversely, if it reaches the state space limit or fails to complete within the bounds, it incurs a large penalty λ. These sparse rewards are triggered by the respective Boolean signals $r_{\text{goal},t}$ and $r_{\text{boundary},t}$. Finally, these dense and sparse rewards are combined into R_t, the overall reward at time t. It is computed using non-negative weights summing to unity to scale the dense rewards, as shown in Equation 3.

$$\begin{aligned} R_t &= w_{\text{yaw}}\, r_{\text{yaw},t} + w_{\text{pos}}\, r_{\text{pos},t} + w_{\text{smooth}}\, r_{\text{smooth},t} \\ &\quad + \beta\, r_{\text{goal},t} - \lambda\, r_{\text{boundary},t} \\ &\text{subject to } \begin{cases} w_{\text{yaw}} + w_{\text{pos}} + w_{\text{smooth}} = 1, \\ w_i \geq 0, \quad \forall i \in \{\text{yaw, pos, smooth}\}. \end{cases} \end{aligned} \tag{3}$$

The reward constants were selected empirically in preliminary tuning runs and then kept fixed for all reported experiments: $\sigma_{\text{yaw}} = 0.10$, $w_{\text{yaw}} = 0.35$, $\sigma_{\text{pos}} = 0.25$, $w_{\text{pos}} = 0.40$, $w_{\text{smooth}} = 0.15$, $(k_{\text{pos}}, k_{\text{yaw}}) = (0.5, 0.5)$, $(\Delta a_{\max,\text{pos}}, \Delta a_{\max,\text{yaw}}) = (2.0, 0.6)$, $\beta = 10$, $(e_{d,\text{th}}, e_{\psi,\text{th}}) = (0.25\,\text{m}, 3^\circ)$, and $\lambda = -10$.

C. Domain Randomization

To ensure consistent learning under varying conditions, the agent was trained using domain randomization. This approach makes each episode unique yet sufficiently similar for the agent to understand the task. The randomization process, partially illustrated in Figure 4, follows these principles:

- **Background changes**: Random selection from a finite set of realistic terrains, represented by $\text{rand}(\text{terrain}_1, \ldots, \text{terrain}_n)$, on which the PV panels are mounted. Although the target use case is a floating PV plant, at the operating altitude and camera pitch used for close-range panel following, the FOV is dominated by panel rows, making explicit water texture less influential than panel geometry, lighting, and reflections.
- **Panel Offsets**: Application of positional offsets, relative to the base panel positions, to all panels in the x and y directions:

$$\left([x_{\text{panel}_{\text{base}}},\, y_{\text{panel}_{\text{base}}}]\right) + \text{rand}\left([x_{\text{panel}_{\text{offset}}},\, y_{\text{panel}_{\text{offset}}}]\right).$$

- **Panel layouts and UAV base positions**: The panels are arranged in three side-by-side arrays, i_0, i_1, and i_2, corresponding to the left, middle, and right positions,

respectively. Each array offers a distinct aerial perspective of the observation conditions, allowing the agent to occasionally capture neighboring panels within the camera's FOV. The three arrays are always present in the environment, and the agent starts in the array closest to its randomly selected base position:

$$\text{rand}(i_0, i_1, i_2) \implies \big([x_{\text{uav}_i}, y_{\text{uav}_i}, \psi_{\text{uav}_i} = 0]\big).$$

Here, $\psi_{\text{uav}i} = 0$ indicates that the UAV's yaw orientation is perfectly aligned with the panel. The positions x_{uav_i} and y_{uav_i} correspond to the location directly in front of the first PV panel.

- **UAV pose offset**: A randomized offset is added to the previously selected base UAV position. This offset affects the x, y, and ψ (yaw) components of the UAV base pose as follows:

$$\big([x_{\text{uav}_i}, y_{\text{uav}_i}, \psi_{\text{uav}_i}]\big) + \text{rand}\big([x_{\text{uav}_{\text{offset}}}, y_{\text{uav}_{\text{offset}}}, \psi_{\text{uav}_{\text{offset}}}]\big).$$

- **Sunlight position**: The position of the Sun is randomized by varying the azimuth and zenith angles to produce viable daylight positions.

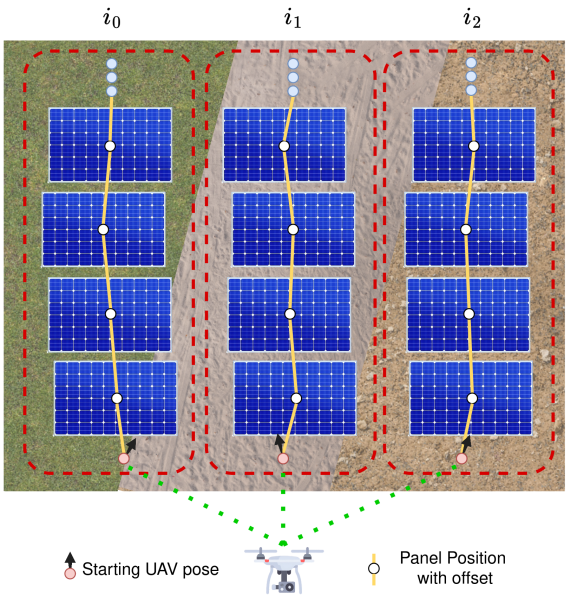


Fig. 4: Domain randomization of the training environment.

D. Data and Architecture

The Soft Actor-Critic (SAC) algorithm [14] was chosen to learn the optimal policy for the environment and task at hand. It supports continuous actions and maintains stochasticity, which can be beneficial in a POMDP [15]. As an off-policy algorithm, SAC enables efficient reuse of past experiences, while its model-free nature removes the need for an explicit system model, simplifying deployment across UAV platforms.

A new feature extractor architecture was developed on top of the CNN introduced by Mnih et al. [16]. In this design, the original architecture was extended to also process a set of past actions through a Multilayer Perceptron (MLP). The embeddings produced by the CNN and MLP are then concatenated to form the hidden state that the SAC processes. To keep the comparison attributable to the proposed multi-input design, the visual branch preserves the same core convolutional scale as Mnih et al. [16] (three convolutional layers with 32/64/64 filters and equivalent kernel/stride pattern), with only task-driven adaptations such as input resolution and normalization layers. Frame/action stacking is prioritised over recurrent modules (e.g., Long Short-Term Memory) to retain implementation simplicity and off-policy sample efficiency while still injecting short-term temporal context into the latent state.

However, this CNN does not receive the usual 84×84 images. Instead, the visual stream preserves the original 4:3 camera aspect ratio (1080×1440), resized to 96×76. Preserving the aspect ratio is crucial for accurately observing the state, as it provides a more precise perception of the camera's alignment relative to the panel.

Due to the high frame rate relative to the UAV motion, a frame-skipping of 4 is applied. With the image buffer organised as a 20-frame First In, First Out (FIFO) queue, this results in 5 distinct frames representing different time instances, as shown in Figure 5.

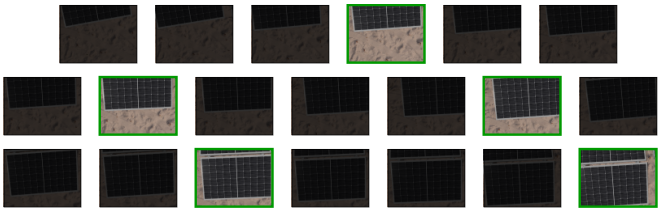


Fig. 5: Frame-skipping observations.

In addition to resizing, the original Red-Green-Blue (RGB) observations are converted to grayscale. This reduces input dimensionality and computational cost, and encourages color-agnostic features that are less sensitive to illumination and white-balance shifts introduced by domain randomization. The operational scope in this work is therefore monocular daytime visual navigation (RGB-derived grayscale), while thermal and EL sensing are outside the evaluated system scope. A slight (5×5) blur is applied after resizing to bias the CNN toward broader image patterns rather than fine textures, improving generalisation.

The CNN block processing the stacked images comprises three convolutional layers:

- Conv1: 32 filters, 8×8 kernel, stride 4;
- Conv2: 64 filters, 4×4 kernel, stride 2;
- Conv3: 64 filters, 3×3 kernel, stride 1.

Each convolution is followed by a Rectified Linear Unit (ReLU) activation and a Group Normalization layer (GroupNorm, $G = 8$). After the final ReLU, the output is flattened and fed into the projection block. This block includes a linear layer followed by Layer Normalization and a ReLU that projects the image features to an embedding of size $(N, 512)$.

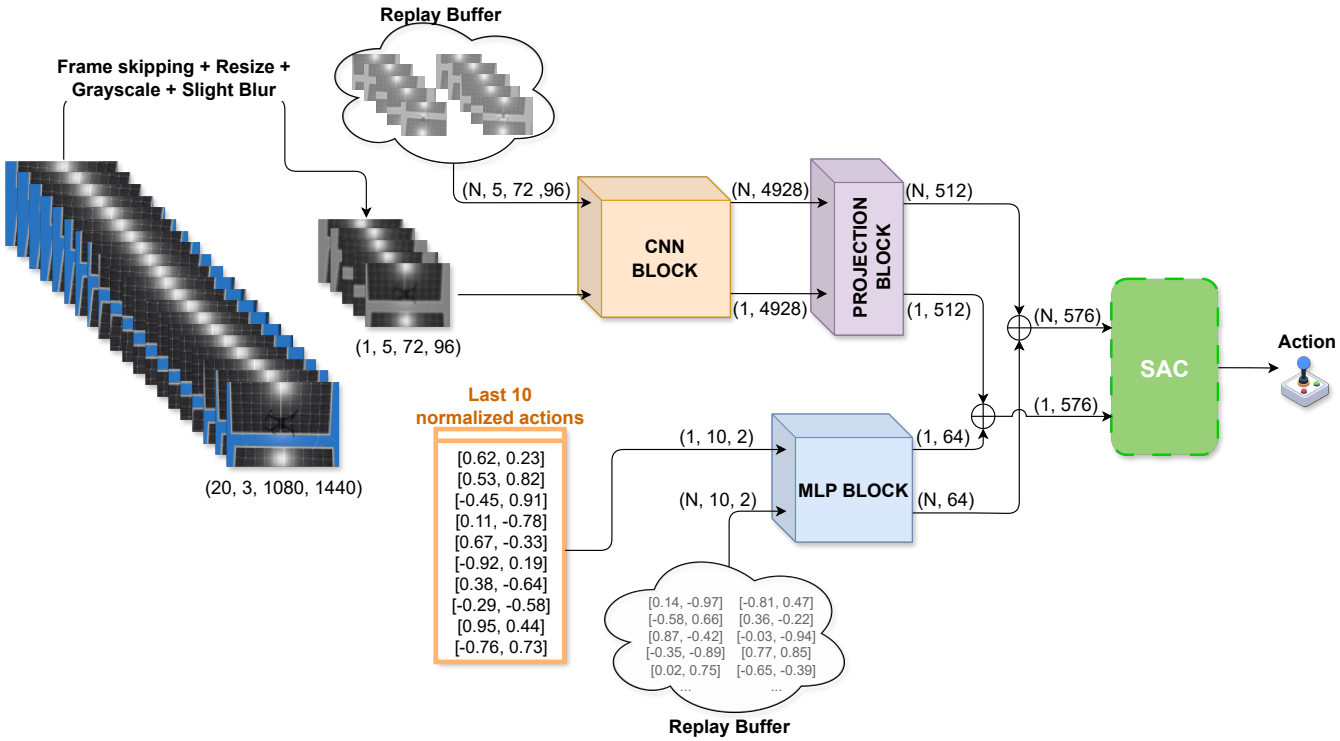


Fig. 6: Feature extractor architecture with normalized actions for the SAC algorithm.

The action history is stored in a FIFO queue of length $K = 10$ (stacking actions $a_t, \ldots, a_{t-9}$) at approximately 10 Hz, representing about one second of past actions. These normalized actions form a tensor of shape $(N, 10, 2)$, which is flattened to $(N, 10 \cdot 2)$ before entering the MLP block. The MLP (velocity/action branch) consists of a hidden layer with 128 neurons and an output layer with 64 neurons; Layer Normalization and ReLU activations follow each linear layer. This produces an action/velocity embedding of size $(N, 64)$.

Finally, the image embedding $(N, 512)$ and the action/velocity embedding $(N, 64)$ are concatenated to form the final embedding of size $(N, 576)$. All experience (observations, actions, etc.) is stored in the replay buffer and sampled in mini-batches of size N for training with SAC.

Figure 6 depicts the overall architecture. The SAC block is simplified for the context. Both the actor and critic networks use this architecture to process the observations. However, there is no single shared feature extractor model. The weights are neither shared nor combined between the actor and critic networks to avoid degrading performance due to conflicting updates. This promotes better learning, since they are optimized for different tasks.

IV. Experimental Testing

A. Simulation Setup

All experiments were carried out on a computer whose hardware and software specifications are summarized in Table I. This information is included to ensure the reproducibility of the experiments and clarify the computational resources used.

TABLE I: Hardware and software specification.

Component	Specification
CPU	Intel Core i7-14700 K
RAM	32 GB
GPU	NVIDIA RTX 4060 Ti
VRAM	16 GB
OS	Ubuntu 22.04
Python	3.10.12
ROS	ROS 2 Humble
Gazebo	Harmonic
Arducopter	4.7.0-dev
StableBaselines3	2.6.0
Pytorch	2.5.1
OpenCV	4.10.0
Albumentations	2.0.8

To allow for the low-level control of the UAV, the Gazebo simulation platform was integrated with the Robot Operating System (ROS) 2 middleware and the Software-in-the-loop (SITL) Ardupilot autopilot. A realistic 3D model of a PV panel with reflective properties and different Physically Based Rendering (PBR) textures were used to create a rich simulation environment, as shown in Figure 7. The domain randomization details mentioned previously were applied to the simulation. In total, five random backgrounds were used during the training process: Blue Plastic, Dirt, Grass, Gravel, and Sand.

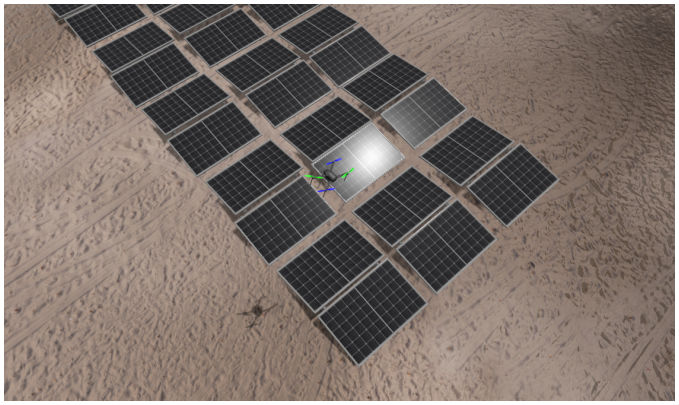


Fig. 7: Simulation world with sand background.

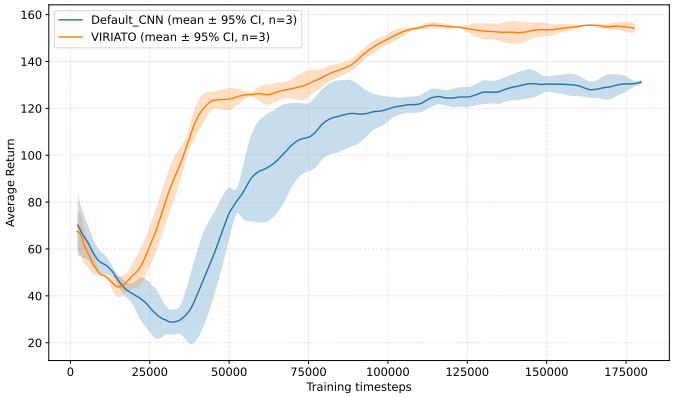
To assess the performance and viability of the VIRIATO architecture, it was compared to the well-known Deep Q-Learning (DQN) feature-extraction CNN from Mnih et al. [16], which served as the inspiration for this design. All models were trained with SAC, in a total of 180 000 simulation steps, with 3 different seeds; thus, training was budgeted by steps rather than by a fixed number of episodes. The reward was compared as well as the accuracy metrics of displacement/position (e_d) and angular/yaw (e_ψ) errors. Both architectures were trained using the set of hyperparameters in Table II. The replay buffer size was set to the maximum capacity allowed by the available RAM. Wind disturbances were not explicitly modeled, since this study targets high-level visual guidance while low-level stabilization is handled by the flight controller. To isolate the effect of action-history integration, both models also used the same preprocessing pipeline (grayscale conversion, blur and frame-skipping), domain randomization settings and training budget; the additional action-history branch is the distinguishing component of VIRIATO.

TABLE II: Hyperparameters used in the training of the models.

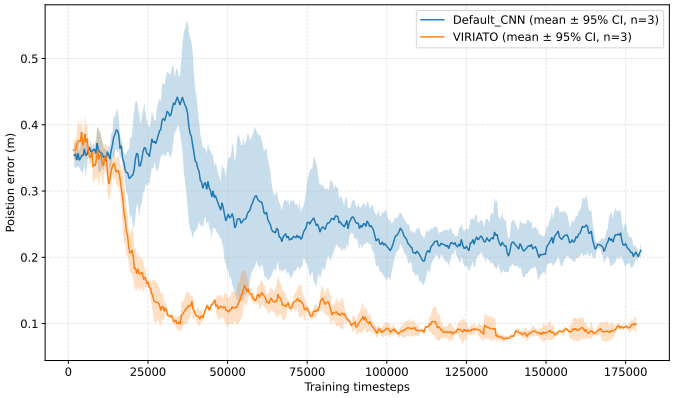
Parameter	**Symbol**	**Value**
Learning rate	lr	1×10^{-4}
Replay buffer size	B	80^5
Learning starts	N_{start}	5000
Batch size	N	256
Soft update coefficient	τ	0.01
Discount factor	γ	0.99
Train frequency	f_{train}	1
Gradient steps	G	1
Initial entropy coefficient	α	`1.0`
Target entropy	H_{target}	$-\dim(\mathcal{A})$
Target update interval	T_{target}	1

B. Training Performance

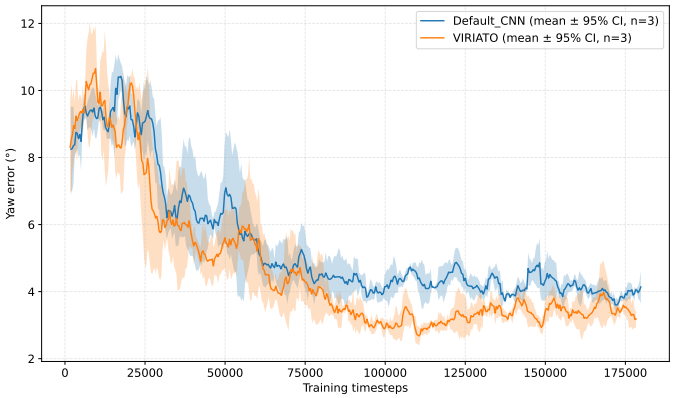
Figure 8a shows that VIRIATO reaches higher cumulative reward faster than the image-only CNN baseline, indicating better sample efficiency and learning stability across seeds. This improvement is consistent with the lower position and yaw errors in Figures 8b and 8c, supporting the claim that action-history encoding helps maintain a more accurate belief state for alignment control.



(a) Average cumulative reward for each model during training.



(b) Average position error (e_d) for each model.



(c) Average yaw error (e_ψ) for each model.

Fig. 8: State-error performance for the VIRIATO and the baseline model. The VIRIATO model achieves lower position and yaw errors, indicating improved UAV to panel alignment.

C. Evaluation and Generalization

The best-performing version of VIRIATO, selected as the one whose weights achieved the highest reward on the validation set, is evaluated across four scenarios designed to test dif-

ferent aspects of generalization. A new "Rocky" background was included in all tests.

*1) **Default Test**:* Larger PV arrays than during training are used to evaluate the agent's ability to navigate plants of different dimensions. The model achieved $100\,\%$ task completion with $72\,\%$ of episodes perfectly aligned. Perfectly aligned means that the agent reached the end with an error of less than $0.25\,\mathrm{m}$ and $3°$. The mean position error was $\bar{e}_d = 0.09\,\mathrm{m}$ and the mean yaw error was $\bar{e}_\psi = 3.57°$, with no significant degradation across terrains and arrays (Table III).

TABLE III: Average Yaw and Position Errors with Standard Deviations for all the planes and arrays in the default test.

Surface/Array	$\bar{e}_\psi$ (°)	σ_{e_ψ} (°)	$\bar{e}_d$ (m)	σ_{e_d} (m)
By Terrain				
Blue Plastic	3.38	1.77	0.08	0.01
Dirt	4.51	2.60	0.09	0.04
Grass	3.18	0.94	0.10	0.05
Gravel	3.61	0.96	0.11	0.04
Rocky	3.69	0.95	0.09	0.02
Sand	3.30	1.03	0.10	0.03
By Array				
Left	3.04	0.76	0.09	0.02
Middle	3.77	1.49	0.09	0.04
Right	3.75	1.87	0.10	0.04

*2) **Aligned Test**:* The PV panels are perfectly aligned to see how the agent performs without misalignment cues. In this test, the agent reached the end in $99\,\%$ of episodes and $71\,\%$ of them aligned within the acceptance bounds. The failed episode can be explained by a rare randomized sunlight condition (outlier). As expected, errors were reduced compared to the default test (Table IV), with the largest values on blue plastic, coinciding with the sunlight outlier.

TABLE IV: Average Yaw and Position Errors with Standard Deviations for all the terrains and arrays in the aligned test.

Surface/Array	$\bar{e}_\psi$ (°)	σ_{e_ψ} (°)	$\bar{e}_d$ (m)	σ_{e_d} (m)
By Terrain				
Blue Plastic	4.29	2.13	0.10	0.04
Dirt	2.96	1.18	0.08	0.01
Grass	3.45	2.15	0.09	0.02
Gravel	3.47	0.95	0.08	0.02
Rocky	3.56	1.64	0.09	0.03
Sand	3.73	1.10	0.09	0.03
By Array				
Left	3.39	1.20	0.08	0.03
Middle	3.58	1.58	0.08	0.03
Right	3.68	1.97	0.09	0.03

*3) **Altitude Test**:* The agent flies at previously unseen altitudes ($2.5\,\mathrm{m}$, $2.7\,\mathrm{m}$ and $2.9\,\mathrm{m}$), which alters the perceived size of the PV panels; during training, altitude was fixed at $2.4\,\mathrm{m}$. The agent reached the end in $100\,\%$ of episodes, with $62\,\%$ of episodes falling within the alignment bounds. The overall mean position error was $\bar{e}_d = 0.077\,\mathrm{m}$ and the overall mean yaw error was $\bar{e}_\psi = 3.51°$. These results indicate that the same trained model generalized across different flight altitudes, with slightly smaller errors at the highest altitude (Table V).

TABLE V: Average Yaw and Position Errors with Standard Deviations for all the terrains, arrays and different altitudes.

Surface/Array/Altitude	$\bar{e}_\psi$ (°)	σ_{e_ψ} (°)	$\bar{e}_d$ (m)	σ_{e_d} (m)
By Terrain				
Blue Plastic	3.66	1.11	0.08	0.04
Dirt	3.42	1.50	0.08	0.03
Grass	2.83	0.96	0.06	0.01
Gravel	4.75	3.48	0.07	0.03
Rocky	4.16	2.94	0.09	0.07
Sand	2.64	0.77	0.06	0.01
By Array				
Left	3.51	1.84	0.07	0.02
Middle	3.66	2.41	0.08	0.03
Right	3.36	2.19	0.09	0.07
By Altitude				
$2.5\,\mathrm{m}$	3.71	2.40	0.07	0.02
$2.7\,\mathrm{m}$	3.42	2.06	0.08	0.04
$2.9\,\mathrm{m}$	3.38	1.94	0.08	0.06

*4) **Image Perturbation Test**:* Brightness variations, noise, flares, and blobs are applied to the camera feed to test visual robustness. These perturbations were applied only at evaluation time (not during training-time domain randomization). The agent completed the task $98\,\%$ of the episodes with $40\,\%$ of panel alignment. The overall mean position error was $\bar{e}_d = 0.11\,\mathrm{m}$ and the overall mean yaw error was $\bar{e}_\psi = 5.35°$. The perturbations had a visible but limited impact, with most episodes completed successfully (Table VI).

TABLE VI: Average Yaw and Position Errors with Standard Deviations for all the terrains and arrays with image perturbations.

Surface/Array	$\bar{e}_\psi$ (°)	σ_{e_ψ} (°)	$\bar{e}_d$ (m)	σ_{e_d} (m)
By Terrain				
Blue Plastic	5.81	1.66	0.11	0.02
Dirt	5.08	1.18	0.11	0.02
Grass	5.18	1.54	0.10	0.02
Gravel	5.67	1.00	0.10	0.02
Rocky	5.34	1.12	0.11	0.02
Sand	5.32	1.18	0.11	0.02
By Array				
Left	5.26	1.38	0.11	0.02
Middle	5.27	1.16	0.11	0.02
Right	5.59	1.33	0.11	0.02

V. Conclusion

VIRIATO delivers robust RL-based visual navigation for UAVs inspecting PV panel arrays. By integrating visual observations with recent past actions, it constructs a richer state representation and improves control in partially observable environments. Compared to a baseline CNN feature extractor, VIRIATO achieved higher cumulative reward during training and lower positional/yaw errors.

Extensive evaluation across panel alignment, altitude, and image perturbations confirmed the model's generalization.

Although perturbations slightly increased error, VIRIATO achieved near-complete task success, showing robustness. These results highlight that multi-input feature integration and domain randomization improve UAV autonomy for lateral centering and orientation correction during predefined sweeps, and suggest that the architecture could be readily adapted to other perception-driven tasks.

At this stage, validation remains simulation-only; wind-related disturbances are primarily handled by the low-level flight controller, while real-world end-to-end robustness of the proposed high-level policy remains to be verified.

Future work involves real-world deployment, integration with other RL algorithms for further adaptability and efficiency, extension to IR/EL sensing for non-daylight inspection scenarios, and full component-wise ablations (e.g., domain-randomization on/off and recurrent baselines).

REFERENCES

[1] G. Masson, A. V. Rechem, M. de l'Epine, and A. Jäger-Waldau, "Snapshot 2025," IEA PVPS, Tech. Rep., Apr. 2025. [Online]. Available: https://iea-pvps.org/wp-content/uploads/2025/04/Snapshot-of-Global-PV-Markets_2025.pdf

[2] A. Fischer, "Aerial inspection of solar plants goes mainstream," https://pv-magazine-usa.com/2022/06/29/aerial-inspections-of-solar-plants-goes-mainstream/, Jun. 2022.

[3] EagleNXT, "Increasing solar inspection efficiency with drones," https://eaglenxt.com/use-cases/case-study-increasing-solar-inspection-efficiency-with-drones/, Oct. 2018.

[4] O. I. Olayiwola and F. Camara, "Challenges and Opportunities for Autonomous UAV Inspection in Solar Photovoltaics: International Conference on Renewable Energy and Environment Engineering," in *E3S Web Conferences*, vol. 572. EDP Sciences, Sep. 2024.

[5] European Union Aviation Safety Agency, "Is beyond visual line of sight (BVLOS) operation possible for flying drones with goggles (first-person view (FPV)) operation?" https://www.easa.europa.eu/en/faq/140038, 2024.

[6] DJI, "Photovoltaic Power Plant - Renewables - Inspection - DJI Enterprise," https://enterprise.dji.com/inspection/photovoltaic-power-plant, 2025.

[7] M. Meribout, V. Kumar Tiwari, J. Pablo Peña Herrera, and A. Najeeb Mahfoudh Awadh Baobaid, "Solar panel inspection techniques and prospects," *Measurement*, vol. 209, p. 112466, 2023.

[8] A. Niccolai, F. Grimaccia, S. Leva, and P. Eleftheriadis, "Photovoltaic Plant Inspection by means of UAV: Current practices and future perspectives," in *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, Sep. 2021, pp. 1–6.

[9] H. Choset and P. Pignon, "Coverage Path Planning: The Boustrophedon Cellular Decomposition," in *Field and Service Robotics*, A. Zelinsky, Ed. London: Springer, 1998, pp. 203–209.

[10] L. Morando, C. T. Recchiuto, J. Calla, P. Scuteri, and A. Sgorbissa, "Thermal and Visual Tracking of Photovoltaic Plants for Autonomous UAV Inspection," *Drones*, vol. 6, no. 11, p. 347, Nov. 2022.

[11] S. Gallardo-Saavedra, L. Hernández-Callejo, and O. Duque-Perez, "Image Resolution Influence in Aerial Thermographic Inspections of Photovoltaic Plants," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 12, pp. 5678–5686, Dec. 2018.

[12] G. Roggi, A. Niccolai, F. Grimaccia, and M. Lovera, "A Computer Vision Line-Tracking Algorithm for Automatic UAV Photovoltaic Plants Monitoring Applications," *Energies*, vol. 13, no. 4, p. 838, Jan. 2020.

[13] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep Learning vs. Traditional Computer Vision," in *Advances in Computer Vision*, K. Arai and S. Kapoor, Eds. Cham: Springer International Publishing, 2020, pp. 128–144.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," Aug. 2018.

[15] J. Williams and S. Singh, "Experimental Results on Learning Stochastic Memoryless Policies for Partially Observable Markov Decision Processes," in *Advances in Neural Information Processing Systems*, vol. 11. MIT Press, 1998.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.