

EVLM: Intent-Driven Edge Vision Language Model for UAV-Based Power Line Inspection

Reza Farahani*, Zoha Azimi[†], Ilir Murturi[‡], Arda Goknil[§], Sagar Sen[§], Christian Timmerer[†], Schahram Dustdar*

*Distributed Systems Group (DSG), Technical University of Vienna, Austria

[†] Department of Information Technology (ITEC), University of Klagenfurt, Austria

[‡] Department of Mechatronics, University of Prishtina, Kosovo

[§] SINTEF, Norway

Abstract—Inspection of critical infrastructure, such as power lines, is increasingly conducted using unmanned aerial vehicles (UAVs) that capture aerial video for subsequent human review. Although recent edge-based approaches deploy onboard object detectors to identify predefined defect classes, these pipelines remain closed-set, task-specific, and largely decoupled from operator intent and edge resource constraints. This paper introduces EVLM, an intent-driven vision-language framework for onboard UAV-based power line inspection. Given a high-level operator intent, EVLM (i) leverages lightweight histogram-based frame filtering to extract salient key frames under bounded compute budgets, (ii) executes a domain-adapted vision language model (VLM) directly on the UAV for intent-conditioned multimodal reasoning, and (iii) synthesizes structured inspection reports together with a minimal set of evidence frames, replacing continuous raw video transmission with compact semantic outputs. To align the VLM with infrastructure inspection semantics while preserving edge efficiency, we perform parameter-efficient fine-tuning using Low-Rank Adaptation (LoRA), enabling domain specialization without updating the full model parameters. We implement and fully deploy EVLM on an NVIDIA Jetson device representative of UAV-class onboard hardware and evaluate it using 20 publicly released power line inspection video sequences spanning 8 heterogeneous environments and 5 operational intent categories. Experimental results show a data reduction of 94.8%, with transmitted data decreasing from 485 kB to 25 kB per 4 s segment, corresponding to 72.75 MB versus 3.75 MB over a 10 min inspection mission. EVLM operates feasibly on embedded hardware, maintaining moderate CPU/GPU utilization and bounded power consumption (5.6 W), while producing interpretable, intent-aligned inspection outputs, with richer semantic insights than detection-centric baselines.

Index Terms—Edge Computing, Vision Language Models (VLM), Edge AI, UAV, Drone AI, Power Line Inspection.

I. INTRODUCTION

Critical infrastructures, such as power lines, underpin modern society, yet their inspection remains resource-intensive and operationally critical. In Europe alone, electricity distribution system operators manage over 10 million kilometers of power lines serving approximately 300 million customers [1]. Failures in transmission infrastructure have been linked to large-scale outages and wildfire incidents, emphasizing the need for scalable, reliable, and continuous inspection mechanisms. In recent years, unmanned aerial vehicle (UAV)-based inspection has significantly reduced operational risk and cost by replacing manual climbing and helicopter surveys with aerial data ac-

quisition. However, early deployments remain video-centric, where UAVs capture high-resolution imagery that is subsequently analyzed offline by human operators [2], [3]. While safer and faster than manual inspection, this paradigm remains bandwidth-intensive and depends on post hoc interpretation.

Recent advances in edge artificial intelligence have enabled onboard object detection, with You Only Look Once (YOLO)-based architectures becoming prevalent due to favorable accuracy-latency trade-offs [4]–[6]. These systems perform closed-set bounding-box detection over predefined object classes. Although effective for localization, this paradigm has two structural limitations in inspection contexts. First, closed-set detectors rely on fixed vocabularies: each defect type (e.g., corrosion, broken insulator, vegetation encroachment) must be explicitly defined and supported by annotated training data, limiting scalability as inspection objectives evolve. Second, object detection does not inherently model relational or semantic state. For example, detecting a tree and a conductor does not determine whether vegetation violates clearance constraints, nor does identifying a tower infer structural condition or environmental visibility. Detection-centric pipelines, thus, answer what is present in a scene, but not whether it satisfies a specific *operational inspection intent*. With the recent emergence of vision-language models (VLMs), large language models (LLMs), and agentic AI systems [7] capable of multimodal reasoning and task-oriented decision making, there is an opportunity to move beyond rigid object localization toward intent-conditioned inspection and semantic understanding [8], [9].

This paper introduces EVLM, an intent-driven edge vision-language framework for onboard UAV inspection. In EVLM, operator intent is expressed as a natural-language query specifying the inspection objective (e.g., defect detection or component identification). Rather than relying on predefined defect localization, EVLM formulates inspection as intent-conditioned semantic reasoning executed directly on the UAV's edge processor. Given an operator intent, EVLM (i) employs a lightweight histogram-based frame filtering to bound inference under compute and energy constraints, (ii) performs intent-conditioned multimodal inference using a domain-adapted vision language model (VLM) on selected frames, and (iii) generates structured inspection reports with a minimal set of

Table I: Comparison between closed-set object detection and intent-conditioned vision–language inference.

Dimension	Closed-set object detection (e.g., YOLO)	Vision language model
Learning formulation	Supervised classification over a fixed label space	Pretrained multimodal model with prompt-conditioned inference
Vocabulary	Closed-set; predefined defect classes	Open-vocabulary; concepts specified via natural language prompts
Output representation	Bounding boxes and class probabilities	Semantic textual responses conditioned on intent
Adaptation to new tasks	Requires retraining with annotated data	Supports zero- and few-shot reasoning via prompts
Contextual modeling	Limited to object-level detection	Models object state, spatial relations, and scene context
Communication abstraction	Frame-level or detection-level transmission	Structured semantic summaries

evidence frames, replacing continuous raw video transmission. In contrast to conventional UAV inspection pipelines tied to fixed defect classes, EVLM decouples inspection logic from static label spaces by integrating intent-conditioned reasoning, semantic validation, and evidence selection within a unified onboard pipeline. This enables bandwidth-efficient, intent-driven inspection under constrained edge resources. The primary contributions of this paper are:

1) *Edge architecture*: We design and implement a fully onboard three-stage inspection pipeline comprising deterministic frame filtering, intent-conditioned multimodal inference, and semantic packaging. The architecture ensures bounded compute, energy, and transmission costs while enabling semantic inspection directly on UAV-class edge hardware.

2) *Semantic packaging*: We develop an intent-aware consolidation algorithm that converts frame-level VLM outputs into a structured inspection report and minimal evidence subset, replacing continuous raw video transmission with compact, interoperable semantic artifacts while preserving visual evidence for verification and explainability.

3) *Parameter-efficient VLM adaptation*: We adapt a lightweight VLM (SmoVLM-256 [10]) to the power line inspection domain using parameter-efficient fine-tuning (PEFT) [11] with Low-Rank Adaptation (LoRA) [12]. This approach enables domain specialization of both visual and language representations while updating only a small set of trainable parameters, making onboard deployment feasible under UAV edge resource constraints.

4) *System deployment and validation*: We implement and deploy EVLM on an NVIDIA Jetson device representative of UAV-class edge hardware and evaluate it on 20 publicly available power line inspection videos spanning 8 environments and 5 inspection intents. Experimental results indicate a 94.8% reduction in transmitted data, from 485 kB to 25 kB per 4 s segment, corresponding to 72.75 MB versus 3.75 MB over a 10 min inspection mission. The system remains feasible on embedded hardware, maintaining moderate CPU/GPU utilization and bounded power consumption (5.6 W) while producing interpretable, intent-aligned inspection outputs.

The paper has seven sections. Section II presents the background and reviews related work. Section III presents the architecture of EVLM, followed by the details of its inspection algorithm in Section IV. Section V describes the experimental setup and evaluation methodology. Section VI reports and analyzes the empirical results. Finally, Section VII concludes the paper and outlines future directions.

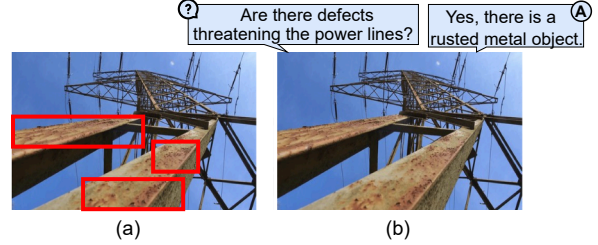


Figure 1: Example of detection vs. intent inference; (a) bounding-box output, (b) semantic response.

II. BACKGROUND AND RELATED WORK

This section contrasts detection- and intent-based inspection (Section II-A), reviews recent advances in VLMs and onboard UAV-based power line inspection (Sections II-B–II-C), and outlines how EVLM differs from prior work (Section II-D).

A. Detection-based vs. intent-based inspection

Object detection architectures such as YOLO [13]–[15] formulate perception as supervised closed-set localization and classification, predicting bounding boxes and class probabilities over a predefined label space [16], similar to Faster R-CNN [17]. These models are optimized for low-latency inference and deployed on edge platforms using lightweight backbones (e.g., MobileNet [18]), but their closed-set design couples inspection capability to annotated categories and retraining [19]. As summarized in Table I, such detectors operate at the object level and return numerical outputs, limiting their ability to reason over relational constraints, scene context, or intent-specific inspection criteria. Fig. 1a exemplifies this paradigm, where a corrosion defect is localized via bounding boxes without assessing its operational relevance.

In contrast, VLM models pre-trained on large-scale image–text corpora replace closed-set classification with joint visual–textual representation learning. Given an image and a textual prompt, the model performs prompt-conditioned inference, generating semantic outputs rather than fixed class labels. As summarized in Table I, it supports open-vocabulary reasoning over object state, spatial relationships, and scene context without redefining label spaces. Fig. 1b illustrates this paradigm in power line inspection, where the model answers inspection queries (e.g., “Are there defects threatening the power lines?”) rather than returning only bounding boxes.

B. Vision language models and open-vocabulary perception

VLMs shift recognition from closed-set classification to open-vocabulary perception by aligning images with natural-language supervision [20]. Contrastive Language-Image Pre-

training (CLIP) [21] and A Large-scale Image and Noisy-text Embedding (ALIGN) [22] models established contrastive image-text pretraining for prompt-based transfer, while Bootstrapping Language-Image Pretraining (BLIP) [23] and BLIP2 [24] models unified vision-language understanding and generation. Beyond recognition, open-vocabulary localization extends language conditioning to detection and grounding, Latent Language Image Pretraining (GLIP) [25], Detic [26], Grounding DINO [27], YOLO-World [28]). Moreover, instruction tuning supports reliable prompt-conditioned multimodal reasoning (Flamingo [29], InstructBLIP [30], Qwen2-VL [31]), while compact VLMs (MobileVLM V2 [32], TinyLLaVA [33], SmolVLM [10]) reduce model and inference cost for practical on-device deployment.

C. Onboard UAV-based power line inspection

Recent years have seen significant progress in UAV-based power line inspection, particularly in autonomous navigation and visual defect detection [34]. Xing et al. [35] integrated perception with model predictive control (MPC) for obstacle-aware power line tracking. Foehn et al. [36] introduced Agilicious, a high-performance quadrotor platform with fully onboard sensing and computation for agile, vision-based inspection. On the perception side, supervised pipelines dominate, including YOLO-based real-time vegetation and component monitoring [5], Hybrid-YOLO for insulator defect detection [37], and multi-sensor fusion to improve robustness under challenging visibility [38]. Continuous raw UAV video uplink is bandwidth-intensive, motivating edge-assisted analytics [39]. Wang et al. [40] proposed a UAV video analytics architecture that reduces uplink bandwidth by combining lightweight onboard filtering with deep neural network inference on nearby edge servers, enabling drones to discard uninformative frames and transmit only relevant data for downstream processing. Maresch and Nastic introduced VATE [41], an edge-cloud video analytics system that integrates object detection at edge infrastructure, cloud-based inference, and on-device multi-object tracking, and fuses asynchronous edge and cloud detections through a dedicated edge-cloud fusion algorithm, enabling real-time processing under resource-constrained edge environments while improving detection accuracy. Complementary work, such as Geryon [42], further reduces uplink demand by transmitting only saliency-selected regions rather than full frames.

D. EVLM contributions relative to prior works

Despite progress in autonomous navigation, CNN-based detection, and bandwidth-aware offloading, existing UAV-based inspection systems have three primary limitations: 1) inspection logic remains detection-centric and closed-set, binding perception to predefined defect taxonomies; 2) outputs are low-level (bounding boxes, class scores, or cropped regions) without intent-conditioned semantic evaluation; 3) communication optimization focuses on video or region compression rather than semantic report generation. To our knowledge, EVLM is the first onboard UAV-based power line inspection

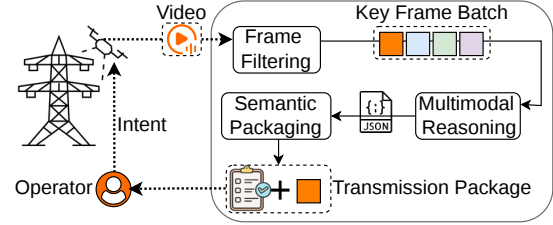


Figure 2: EVLM system architecture.

framework that unifies intent-conditioned multimodal reasoning, structured semantic report synthesis, and evidence-aware transmission. Unlike prior detection-centric or offloading-based approaches, EVLM executes intent-driven semantic evaluation entirely at the UAV and replaces raw video transmission with compact, inspection-ready semantic artifacts.

III. EVLM SYSTEM ARCHITECTURE

Fig. 2 shows the EVLM system architecture deployed on the UAV's onboard edge hardware. The system transforms a continuous aerial video stream into a compact transmission package through three tightly coupled modules: frame filtering, multimodal reasoning, and semantic packaging.

1) *Frame filtering*: Each video sequence is processed online to construct a key-frame batch for downstream inference. A lightweight inter-frame similarity analysis suppresses temporally redundant frames while preserving salient scene transitions and motion changes, retaining only frames whose dissimilarity exceeds a predefined threshold, while the rest are discarded. This module bounds VLM invocation frequency, ensuring deterministic compute and energy usage under resource constraints. Retained key frames may optionally be buffered locally for offline analysis, depending on system constraints.

2) *Multimodal reasoning*: Each retained frame in the batch is processed by the onboard VLM, guided by the operator-defined inspection objective. The model, in fact, performs multimodal inference to extract task-relevant semantics, including visible components, their conditions, contextual cues, and potential defects. The resulting frame-level semantic output is forwarded to the semantic packaging module.

3) *Semantic packaging*: This module processes the log of intent-conditioned VLM outputs for the key frames, aggregating the frame-level semantic entries into a structured inspection report and selecting a small set of representative evidence frames. The resulting transmission package, comprising the report and evidence frames, replaces continuous video with compact semantic records, reducing communication overhead while enabling interoperable outputs that can be directly consumed by downstream inspection systems and verified through the attached visual evidence.

IV. EVLM INSPECTION ALGORITHM

Alg. 1 details the EVLM's onboard inspection pipeline that maps a video stream $\mathcal{V}$ and operator intent p to a transmission package $\langle R, F^* \rangle$, where R is the structured inspection report

and F^* the subset of evidence frames selected for transmission. The algorithm is aligned with EVLM's architecture and organized into three main phases: 1) frame filtering, 2) intent-conditioned multimodal reasoning, and 3) semantic consolidation and evidence selection.

1) *Frame filtering*: First, an empty key-frame buffer $\mathcal{K}$ and reference histogram set h_{prev} are initialized (line 1). Given video duration T and its frame rate r , the algorithm computes the number of decodable frames as $N = \lfloor T \cdot r \rfloor$ (line 2), enabling deterministic bounded processing on the onboard hardware. For each frame index $i \in \{1, \dots, N\}$ (line 3), `DecodeFrame()` retrieves the RGB frame f_i (line 4). A 3D color histogram $h_i = \text{Hist}_{3D}()$ is computed (line 5), flattened, and L2-normalized (line 6) to ensure scale invariance. If no reference histogram exists, the first frame initializes both $\mathcal{K}$ and h_{prev} (lines 7-10). Otherwise, histogram correlation ρ is computed across B bins (lines 11-12). A frame is admitted as a key frame only if $\rho < \tau$ (lines 13-15), i.e., when its color distribution deviates sufficiently from the previously retained frame. This phase implements a streaming redundancy suppression mechanism with complexity $\mathcal{O}(N \cdot B)$, ensuring that downstream VLM invocation frequency is strictly bounded.

2) *Intent-conditioned multimodal reasoning*: The retained key frames $\mathcal{K}$ are then processed individually. For each $f \in \mathcal{K}$ (line 17), the onboard VLM is invoked with the operator intent p (line 18), producing a prompt-conditioned textual output y that describes the scene state and intent-relevant observations. The tuple (f, p, y) is appended to the semantic log $\mathcal{L}$ (line 19). This phase transforms visual inputs into intent-aware semantic representations while reducing inference calls from N to $|\mathcal{K}|$, where $|\mathcal{K}| \ll N$ is under normal motion continuity.

3) *Semantic consolidation and evidence selection*: First, the operator intent p is mapped once to an operational category I via `MapIntent()` (line 20). This function applies rule-based normalization and keyword matching to assign the query to one of the five predefined categories in Section V-D. While any query is processed by the VLM, category-specific semantic rules are applied only when a mapping is established; otherwise, processing proceeds in a generic mode without intent-specific validation. Next, the report container R and evidence set F^* are initialized (line 21). For each log entry $(f, p, y) \in \mathcal{L}$ (line 22), the algorithm employs `SentenceSplit()` to segment the VLM output y into sentence units S (line 23), enabling localized semantic decisions, including negation handling and multi-claim separation. It then invokes `Detect(S)` to extract intent-relevant semantic elements from each sentence into set C (line 24), with the extracted items determined by the operational category I (e.g., components, defects, localization cues, or scene attributes). The function `EvaluateIntent()` (line 25) then applies intent-conditioned semantic rules and returns: (i) a structured summary s , (ii) a set of validated findings Φ , and (iii) a Boolean flag δ indicating whether the sentence contains intent-relevant observations (e.g., detected components or defects) that should be retained as evidence. In the next step, the record $\langle f, I, s, \Phi, C \rangle$ is appended to R (line 26), and if $\delta = 1$, the frame index f is inserted into F^*

Algorithm 1: EVLM onboard inspection algorithm.

Input: Video $\mathcal{V}$ with duration T (s) and frame rate r (fps), intent p , histogram correlation threshold τ

```

1  $\mathcal{K} \leftarrow []$ ,  $h_{\text{prev}} \leftarrow \emptyset$ 
2  $N \leftarrow \lfloor T \cdot r \rfloor$ 
3 for  $i \leftarrow 1$  to  $N$  do
4    $f_i \leftarrow \text{DecodeFrame}(\mathcal{V}, i)$ 
5    $h_i \leftarrow \text{Hist}_{3D}(f_i)$ 
6    $h_i \leftarrow \frac{h_i}{\|h_i\|_2}$ 
7   if  $h_{\text{prev}} = \emptyset$  then
8      $\mathcal{K} \leftarrow \mathcal{K} \cup \{f_i\}$ 
9      $h_{\text{prev}} \leftarrow h_i$ 
10    continue
11    $B \leftarrow \text{dim}(h_i)$ 
12    $\rho \leftarrow \frac{\sum_{b=1}^B (h_i[b] - \bar{h}_i)(h_{\text{prev}}[b] - \bar{h}_{\text{prev}})}{\sqrt{\sum_{b=1}^B (h_i[b] - \bar{h}_i)^2} \sqrt{\sum_{b=1}^B (h_{\text{prev}}[b] - \bar{h}_{\text{prev}})^2}}$ 
13   if  $\rho < \tau$  then
14      $\mathcal{K} \leftarrow \mathcal{K} \cup \{f_i\}$ 
15      $h_{\text{prev}} \leftarrow h_i$ 
16  $\mathcal{L} \leftarrow []$ 
17 foreach  $f \in \mathcal{K}$  do
18    $y \leftarrow \text{VLM}(f, p)$ 
19    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(f, p, y)\}$ 
20  $I \leftarrow \text{MapIntent}(p)$ 
21  $R \leftarrow []$ ,  $F^* \leftarrow \emptyset$ 
22 foreach  $(f, p, y) \in \mathcal{L}$  do
23    $S \leftarrow \text{SentenceSplit}(y)$ 
24    $C \leftarrow \text{Detect}(S)$ 
25    $(s, \Phi, \delta) \leftarrow \text{EvaluateIntent}(I, S)$ 
26    $R \leftarrow R \cup \{(f, I, s, \Phi, C)\}$ 
27   if  $\delta = 1$  then
28      $F^* \leftarrow F^* \cup \{f\}$ 
29 return  $\langle R, F^* \rangle$ 

```

(lines 27–28). The algorithm finally returns $\langle R, F^* \rangle$ (line 29), replacing the raw video with compact semantic artifacts.

4) *Computational complexity*: Let B denote the histogram bin count, $|\mathcal{K}|$ the number of retained key frames, and $|S|$ the average number of sentences per VLM output. Phase 1 runs in $\mathcal{O}(N \cdot B)$, Phase 2 invokes the VLM $|\mathcal{K}|$ times, and Phase 3 incurs $\mathcal{O}(|\mathcal{K}| \cdot |S|)$ due to sentence-level parsing and rule evaluation. In practice, histogram-based suppression ensures $|\mathcal{K}| \ll N$, making both filtering and semantic packaging lightweight relative to multimodal inference while guaranteeing bounded onboard resource usage.

V. EXPERIMENTAL METHODOLOGY AND SETUP

This section presents the experimental setup for evaluating EVLM, including datasets, hardware platforms, VLM configuration, intent modeling, filtering parameters, baselines, and evaluation metrics. All experiments are conducted on edge hardware representative of UAV onboard deployment.

A. Edge and training infrastructure

VLM fine-tuning is performed on a server equipped with a 128-core Intel Xeon Gold CPU and two NVIDIA Quadro GV100 GPUs. All EVLM runtime experiments are executed on an NVIDIA Jetson platform, representative of UAV-class edge hardware, featuring an 8-core CPU, 16 GB RAM, 1024 CUDA cores, and 32 Tensor cores.

B. Power line datasets

We use two publicly available UAV-based power-line inspection datasets [35], [43] covering complementary conditions. From [35], we construct 20 video sequences segmented

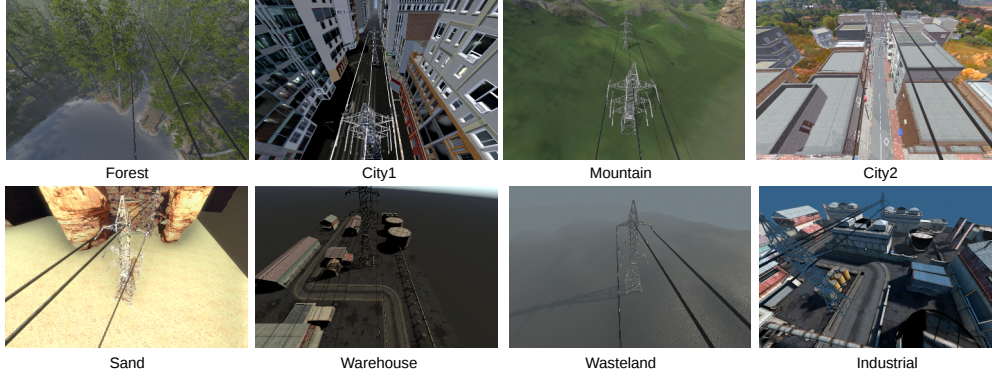


Figure 3: Sample UAV-based inspection environments across eight heterogeneous terrains [35].

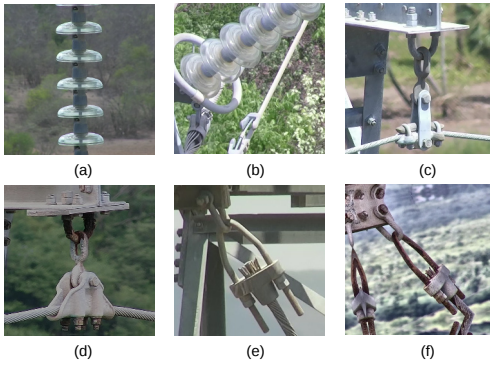


Figure 4: Component-level normal vs. defective conditions from InsPLAD dataset [43]; (a)–(b) insulator strings (normal); (c)–(d) clamp assembly with corrosion in (d); (e)–(f) cable connector with rust degradation in (f).

into fixed-length 4 s clips across 8 heterogeneous power line environments (Fig. 3). Each video has resolution 720×480 at 30 fps, yielding 120 frames per segment. We additionally use the InsPLAD dataset [43], captured with a DJI Matrice 210 V2 and Zenmuse Z30 camera (1920×1080), which provides diverse component and defect categories (Fig. 4) to extend defect coverage for fine-tuning and qualitative evaluation.

C. VLM configuration and fine-tuning

We adopt *SmolVLM-256* [10], a 256M-parameter VLM suitable for edge deployment. While pre-trained for general multimodal reasoning, the base model lacks alignment with power line inspection semantics, fine-grained defect recognition, and structured reporting requirements.

1) *Parameter-efficient adaptation*: We perform supervised fine-tuning using Parameter-Efficient Fine-Tuning (PEFT) [11] with Low-Rank Adaptation (LoRA) [12]. LoRA freezes the pre-trained backbone and injects trainable low-rank adapters into selected layers, enabling efficient domain adaptation with minimal trainable parameters. Adapters are inserted into: (i) *Language decoder*: self-attention projection and MLP layers to adapt intent-conditioned text generation; (ii) *Vision encoder*: projection layers to refine domain-specific visual rep-

resentations. Adapting both modalities jointly improves visual grounding and semantic response generation while preserving the general knowledge of the pre-trained model.

2) *Training configuration*: Model fine-tuning is conducted on the server described in Section V-A using Hugging Face TRL (SFTTrainer). We use LoRA rank $r = 8$, $\alpha = 16$, dropout 0.15, AdamW optimization with cosine learning-rate scheduling, a learning rate of 5×10^{-5} , and bf16 precision. We set the effective batch size to 8 via gradient accumulation and train for 1000 steps with a 90/10 train-validation split. Table II summarizes the hyperparameter configuration.

3) *Instruction alignment*: We further apply supervised instruction tuning to align the model with inspection-specific semantics. We transform dataset annotations (e.g., component labels and defect indicators) into descriptive instruction-response pairs using a high-capability LLM (e.g., GPT-4o) as a text-generation engine. The generation process is constrained by the underlying annotations; prompts explicitly provide the labeled components and defect attributes, and the resulting responses are restricted to describing only the annotated elements. This grounded prompting prevents unsupported content generation and ensures that the produced descriptions remain consistent with the dataset labels. To ensure quality, the generated instruction-response pairs are automatically validated against the original annotations using rule-based checks (e.g., component and defect consistency), and a subset of samples is manually inspected to verify semantic correctness. This process converts binary labels into structured, explainable descriptions, enabling the VLM to produce interpretable, context-aware inspection reports rather than simple detections.

4) *Data augmentation*: To improve robustness to viewpoint and environmental variability, training images are augmented with spatial transformations (e.g., flips, rotations, mild perspective distortions) and photometric perturbations (e.g., color jitter and grayscale conversion), yielding augmented samples per image.

D. Intent pool

We define an intent pool of 20 natural-language queries across five operational categories, used consistently during

Table II: LoRA-based fine-tuning configuration for SmolVLM-256.

<i>LoRA Configuration</i>	
Rank r	8
Scaling α	16
Dropout	0.15
Target (Language)	$q_proj, k_proj, v_proj, o_proj,$ $gate_proj, up_proj, down_proj$
Target (Vision)	$out_proj, fc1, fc2$
Bias	none
<i>Training Hyperparameters</i>	
Effective batch size	8 (2×4 accumulation)
Steps	1000
Learning rate	5×10^{-5}
Scheduler	cosine
Warmup steps	50
Optimizer	AdamW
Weight decay	0.01
Precision	bf16
<i>Data Configuration</i>	
Training samples	250, 1250 (augmented)
Test samples	100
Augmentation factor	$\times 5$
Input format	image-instruction-response
Train/val split	90%/10%

training and evaluation:

- 1) *Scene description*: High-level semantic summarization of the inspected frame (e.g., “Describe the current power line scene.”, “Summarize visible infrastructure and terrain.”).
- 2) *Component enumeration*: Identification and listing of visible infrastructure elements (e.g., “What power line components are present?”, “Are towers, or conductors visible?”).
- 3) *Context awareness*: Assessment of environmental and operational conditions, including visibility, occlusion, and weather effects (e.g., “Are visibility conditions degraded?”).
- 4) *Defect detection*: Identification of structural anomalies and risk factors such as rust, cracks, broken insulators, or vegetation contact (e.g., “Detect rust on metal parts”, “Is any component damaged?”).
- 5) *Localization*: Spatial grounding of detected components or defects within the scene (e.g., “Where is the damaged insulator located?”, “Which tower segment contains corrosion?”).

E. Frame filtering

We compute an RGB histogram (512 bins) per frame, L2-normalize it, and retain a frame when the correlation with the last retained frame falls below $\tau = 0.95$. For edge-feasibility comparison, we also evaluate (i) DCT-based perceptual hashing (pHash) [44], retaining frames with unseen 64-bit hashes, and (ii) vectorized grayscale sum of absolute differences (SAD) [45], retaining frames when the mean pixel difference exceeds $\tau_{sad} = 8000$. Unless otherwise stated, histogram correlation serves as the default filtering strategy.

F. Semantic report structure

The semantic packaging module generates a structured inspection record for each retained frame. Unlike the raw VLM output, this JSON artifact represents the post-processed, intent-evaluated report entry:

```
{
  "frame_id": 12,
  "intent": "Identify defects threatening power",
  "category": "defect_detection",
  "components": ["insulator", "tower"],
  "findings": ["rust"],
  "summary": "Rust visible on the cable",
  "connector": "

```

```
"evidence_required": true}
```

G. Resource and energy measurement

We monitor onboard resource usage via `tegrastats` [46] at a 1 ms sampling interval, recording instantaneous CPU/GPU utilization and total system power $P(t)$ (W) during inspection execution. The computational energy consumption (Wh) over an inspection interval T is modeled as:

$$E_{\text{comp}} = \int_0^T P(t) dt \approx \sum_{k=1}^M P_k \cdot \Delta t, \quad (1)$$

where Δt is the sampling period and M the number of samples within the interval T . The DJI M210 V2 UAV used to capture the InsPLAD data operates with dual TB55 batteries, providing a total energy capacity of $E_{\text{bat}} = 2 \times 174.6 = 349.2$ Wh. Under maximum takeoff weight, the nominal endurance is 24 min, corresponding to an average propulsion power of $P_{\text{flight}} = \frac{E_{\text{bat}}}{24/60} \approx 873$ W. The relative computational energy fraction is therefore $\eta = \frac{E_{\text{comp}}}{E_{\text{bat}}}$, representing the share of total flight energy consumed by onboard inspection processing. The corresponding effective flight time under computational load is thus approximated as $T_{\text{eff}} \approx 24 \cdot (1 - \eta)$ minutes.

H. Evaluation metrics

We evaluate EVLM at both the segment level and the full inspection mission level (comprising multiple 4 s segments) using the following metrics.

- 1) *Inspection accuracy*: Number of correctly identified components and detected defects relative to the annotated ground truth.
- 2) *Transmission data volume*: Total transmitted data (MB) per inspection mission, including compressed frames and/or structured semantic artifacts, depending on the method.
- 3) *Transmitted frames*: Number of video frames transmitted to the ground station during an inspection mission.
- 4) *Processing latency*: Total processing time, including frame filtering, VLM inference, and semantic packaging.
- 5) *Battery utilization*: Ratio between computational energy consumption and the UAV battery capacity, indicating the fraction of flight energy used for onboard inspection.
- 6) *Resource utilization*: Mean CPU and GPU utilization during segment execution, characterizing computational load and edge feasibility.

I. Evaluation baselines

We compare EVLM against representative computation and communication baselines reflecting common UAV inspection pipelines. All baselines operate on the same video segments, filtering (when applicable), and onboard hardware.

1) *Raw Video Offloading (RVO)*: All frames of each 4 s segment are AVC-compressed and transmitted to the ground station. No onboard filtering or inference is performed.

2) *Filtering-Only Transmission (FOT)*: Histogram-based frame extraction is applied onboard, and the retained frames are compressed and transmitted without semantic inference.

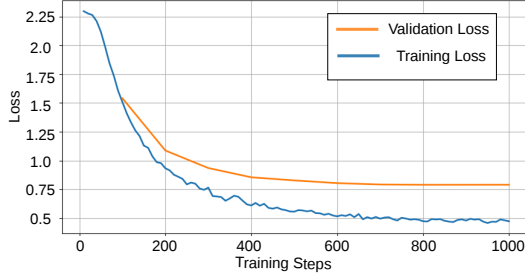
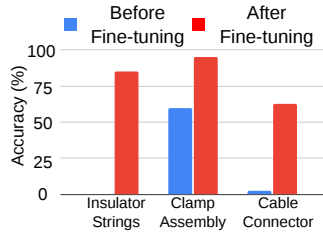
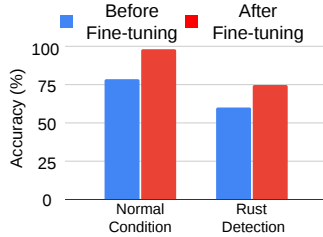


Figure 5: Training and validation cross-entropy loss during LoRA-based supervised fine-tuning of SmolVLM-256 [10] over 1000 steps.



(a) Component identification accuracy.



(b) Defect detection accuracy.

Figure 6: Inspection accuracy improvements after domain-specific instruction tuning of SmolVLM-256 [10].

3) *Detection-Centric Method (DCP)*: Following [5], each frame is processed onboard using a supervised object detector (YOLOv8n), producing bounding boxes, class labels, and confidence scores. Frames containing at least one valid detection are retained, while frames without any detections are discarded. The transmitted payload includes the compressed retained frames together with their detection metadata (bounding-box coordinates, labels, and confidence scores).

4) *VLM Per-Frame Reporting (VPR)*: The VLM is executed independently on each retained frame, and the resulting textual outputs are transmitted directly without semantic consolidation or evidence selection.

VI. EXPERIMENTAL RESULTS

This section presents the experimental evaluation of EVLM, covering training dynamics, inspection accuracy after fine-tuning, edge-device feasibility, frame-filtering efficiency, and baseline comparisons. Each experiment was repeated five times; reported results are the mean across runs.

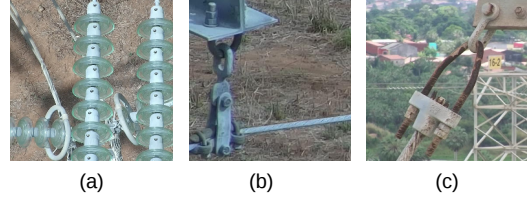


Figure 7: Representative inspection samples used for qualitative evaluation: (a) insulator string, (b) clamp assembly, and (c) cable connector exhibiting visible rust.

A. Training dynamics and convergence analysis

Fig. 5 shows the evolution of training and validation losses over 1000 supervised fine-tuning steps. The objective is token-level cross-entropy applied to the generated response sequence under LoRA adaptation, where only the injected low-rank matrices are updated while the backbone remains frozen. Both losses decrease rapidly within the first 200 steps, indicating efficient domain alignment and stable optimization. Between steps 200 and 600, the validation loss declines smoothly before converging to a plateau near 0.8. After step 600, the training loss continues to decrease while the validation loss remains nearly constant, indicating increasing specialization to the training data without corresponding improvements in generalization. To prioritize generalization and avoid unnecessary memorization, we therefore select the checkpoint at step 600 for all subsequent experiments. The smooth and stable loss curves confirm stable optimization under the chosen learning rate and LoRA configuration.

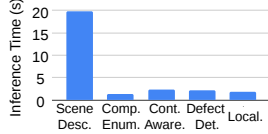
B. Inspection accuracy analysis

We evaluate the inspection accuracy of the fine-tuned VLM on two tasks, (i) infrastructure component identification and (ii) defect detection, by comparing predictions against annotated ground truth from the InsPLAD dataset. Fig. 6a shows component identification accuracy before and after domain-specific fine-tuning. The base model shows limited recognition capability, particularly for visually subtle structures such as cable connectors. After instruction tuning with LoRA adaptation, accuracy improves substantially across all categories, increasing from near-zero to approximately 85 % for insulator strings and to about 65 % for cable connectors. Clamp assembly recognition also improves substantially, increasing from 60 % to approximately 95 %.

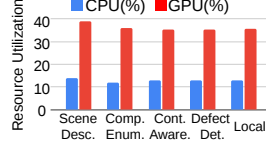
Fig. 6b reports defect detection performance. The fine-tuned model achieves nearly perfect recognition under normal conditions and improves rust detection accuracy from roughly 60 % to approximately 75 %, indicating that instruction-aligned fine-tuning enables the VLM to more reliably distinguish between intact and degraded infrastructure components. The remaining performance gap in certain categories (e.g., rust detection at around 75 %) is mainly due to the subtle visual characteristics of early-stage rust and the limited number of annotated training samples for rare defect patterns. Such defects often exhibit weak color or texture cues, making them harder to detect consistently in UAV imagery captured under varying illumination conditions and viewing angles.

Table III: Fine-tuned VLM responses to two inspection prompts for the test samples in Fig. 7(a)–(c).

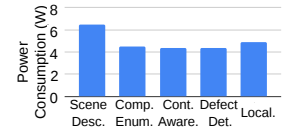
Inspection prompt	Sample Fig. 7 (a)	Sample Fig. 7 (b)	Sample Fig. 7 (c)
What is the power-line component in the image?	The component is a Glass Insulator.	The component is part of the clamp assembly.	The component is a cable connector with a twisted wire.
Is rust visible?	No rust is visible; the surface is clean and in good condition.	The component has a clean, smooth metallic surface.	Rust is visible; metal surfaces appear rough and discolored.



(a) Inference time.

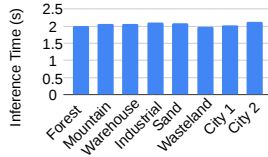


(b) CPU and GPU utilization.

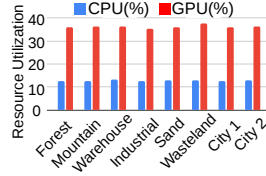


(c) Power consumption.

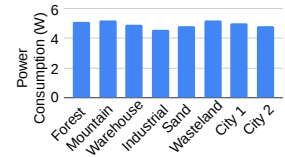
Figure 8: Average VLM runtime characteristics per inspection intent category across the entire video dataset.



(a) Inference time.



(b) CPU and GPU utilization.



(c) Power consumption.

Figure 9: Average VLM runtime characteristics per inspection environment across all prompt categories (excluding scene description).

To illustrate the system’s qualitative behavior, Fig. 7 presents representative inspection samples, while Table III reports the corresponding VLM responses for two inspection prompts: component identification and rust assessment. In Fig. 7a, the model correctly identifies a glass insulator and reports no rust. For the damaged cable connector in Fig. 7c, the model correctly recognizes the component and explicitly describes visible rust and surface degradation. These examples show that the fine-tuned model not only improves quantitative detection accuracy but also produces structured semantic inspection descriptions aligned with the defined intent prompts.

C. Edge feasibility of VLM deployment

We evaluate the feasibility of running the fine-tuned SmolVLM-256 on the UAV edge device by analyzing three runtime indicators: (i) inference time, (ii) resource utilization, and (iii) power consumption. Fig. 8 reports average values for each inspection intent category aggregated across all video segments. As shown in Fig. 8a, scene description yields the highest inference time due to longer generated responses and higher autoregressive decoding cost. In contrast, other intents, e.g., component enumeration, context awareness, defect detection, and localization, achieve lower inference times within 1.4s to 2.3s. Fig. 8b shows moderate utilization across all categories, with CPU usage around 12% to 15% and GPU utilization below 40%. Fig. 8c further shows that power

Table IV: Performance of candidate frame-filtering strategies.

Method	Latency (ms/frame)	Throughput (FPS)	Power (W)	Energy (mWh)	CPU (%)
RGB Histogram	1.9	508	3.4	0.2	37
pHash [44]	2.9	340	3.6	0.34	42
SAD [45]	5.56	178	3.5	0.43	58

consumption remains bounded within 4.4W to 6.5W, far below the UAV propulsion power budget, i.e., 873 W.

To investigate environmental sensitivity, we aggregate the same metrics per inspection environment. Fig. 9 reports averages across all prompt categories. We exclude scene description from this aggregation because its substantially higher and more variable token output length would dominate the mean and mask environment-driven effects for operational intents. Across all scenarios (forest, mountain, warehouse, industrial, sand, wasteland, city), the runtime remains stable: inference time stays close to 2s per segment (Fig. 9a), while power consumption remains around 5W (Fig. 9c). These results indicate that the fine-tuned VLM can operate efficiently on embedded UAV hardware, enabling semantic inspection at the edge with modest computational and power overhead.

D. Frame filtering analysis

We evaluate the computational efficiency of the three frame filtering strategies described in Section V-E: RGB histogram, pHash, and SAD. Table IV reports the performance metrics averaged across all video sequences in the dataset, including per-frame latency, throughput, power consumption, and CPU usage on the UAV edge device. Energy consumption

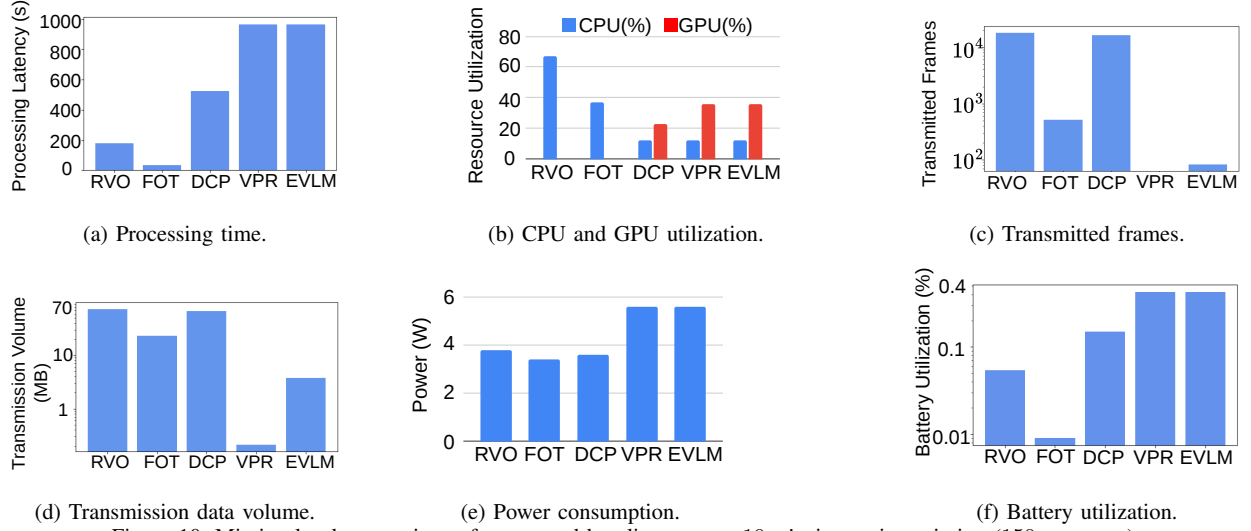


Figure 10: Mission-level comparison of EVLM and baselines over a 10 min inspection mission (150 segments).

is reported across 4s video with 120 frames. Among all, Histogram-based filtering achieves the lowest average latency (1.9 ms/frame) and the highest throughput (508 fps), indicating negligible computational overhead relative to the video decoding pipeline. In contrast, pHash requires additional discrete cosine transform operations, resulting in higher latency (2.9 ms/frame) and lower throughput (340 fps). The SAD method incurs the highest computational cost (5.56 ms/frame) and the lowest throughput (178 fps) due to the full pixel-wise difference computation required for each frame comparison.

Energy consumption follows the same trend, where histogram filtering requires only 0.2 mWh on average, compared with 0.34 mWh for pHash and 0.43 mWh for SAD. Results also confirm that CPU utilization increases with algorithmic complexity, reaching 58 % for SAD, while histogram filtering maintains lower CPU usage (37 %). Importantly, even when aggregated over full inspection video segments, the cumulative overhead remains minimal. For example, a 4 s video segment containing 120 frames introduces only ≈ 228 ms total filtering latency with the histogram method, which is negligible compared with the multi-second VLM inference stage. Considering all results, the histogram approach, with low latency, high throughput, and reduced energy cost, is well-suited for UAV inspection pipelines and is therefore adopted as the default frame filtering strategy in the EVLM system.

E. EVLM vs. baselines

We compare EVLM against the four baselines defined in Section V-I using the evaluation metrics in Section V-H. The comparison considers an inspection mission defined by the intent “Is there a transmission tower in the green area?”. Fig. 10 reports aggregated results over the full mission comprising 150 video segments (10 min flight duration), while Table V presents detailed measurements for a single 4 s segment containing 120 frames.

1) *Processing latency*: Table V shows that *FOT* requires 0.228 s per segment, as it performs only histogram correlation and frame compression. *RVO* increases latency to 1.2 s since the full 120-frame segment must be encoded using the H.264 codec before transmission. *DCP* requires 3.5 s because detector inference is executed on every frame before frame retention decisions, causing computation to scale with the total number of input frames. In contrast, VLM-based pipelines (*VPR* and *EVLM*) require 6.438 s and 6.448 s per segment, respectively. Their runtime is, in fact, dominated by autoregressive token generation during multimodal reasoning, where inference cost scales with generated sequence length rather than the number of retained frames. The nearly identical latencies of *VPR* and *EVLM* indicate that semantic packaging in *EVLM* introduces negligible overhead beyond VLM inference. The mission-level results in Fig. 10a show the same latency trends across the 10 min inspection mission.

2) *Resource utilization*: Fig. 10b illustrates a separation between CPU-dominated video pipelines and GPU-driven inferences. Table V quantifies this behavior per video segment. *RVO* is CPU-bound, reaching 67 % CPU utilization due to continuous video encoding and I/O operations, while *FOT* reduces CPU load to 37 % by suppressing redundant frames early in the pipeline, but still remains CPU-only. In contrast, inference-based pipelines (*DCP*, *VPR*, and *EVLM*) shift computation to the GPU accelerator. CPU usage decreases to approximately 12 % to 15 %, while GPU utilization rises to 23 % to 36 %, reflecting the computational characteristics of neural inference workloads. Importantly, *EVLM* does not increase CPU utilization relative to *VPR*, indicating that intent evaluation and semantic packaging are lightweight operations. Consequently, the majority of computation remains confined to GPU inference, leaving sufficient CPU headroom for flight control, telemetry, and sensor processing on the UAV platform.

Table V: Segment-level comparison of EVLM and baselines.

Method	Processing latency (s)	CPU (%)	GPU (%)	Transmitted frames	Transmission volume (KB)	Power (W)	Battery usage (%)
<i>RVO</i>	1.2	67	-	120	485	3.8	3.63e-4
<i>FOT</i>	0.228	37	-	3.45	155	3.4	6.2e-5
<i>DCP</i>	3.5	15	23	110	444.58	3.6	1e-3
<i>VPR</i>	6.438	12	36	0	1.38	5.6	2.87e-3
EVLM	6.448	12	36	0.55	25	5.6	2.87e-3

3) *Transmission frames and data volume*: Fig. 10c reports the mission-level number of transmitted frames, while Fig. 10d shows the corresponding uplink data volume. Table V provides the detailed segment-level values. *RVO* transmits all 120 frames per segment by design, resulting in 485 KB of data per segment and the largest mission uplink volume (Fig. 10d). *FOT* reduces transmitted frames to approximately 3.45 per segment through histogram-based filtering, decreasing the payload to 155 KB per segment. The remaining communication cost is primarily due to transmitting compressed image frames. In contrast, *DCP* transmits nearly the entire segment (110 frames and 444.58 KB/segment) as power line inspection scenes consistently contain infrastructure elements that trigger object detections, meaning that the “retain-if-detected” policy discards few frames in practice. *VPR* minimizes communication overhead by transmitting only textual outputs (1.38 KB/segment) and no image frames, resulting in the lowest uplink volume but removing visual evidence required for inspection verification. EVLM addresses this limitation by transmitting a minimal set of evidence frames selected during semantic packaging. On average, only 0.55 frames per segment are transmitted together with the structured inspection report, yielding 25 KB/segment. Over the full 150-segment mission, this corresponds to approximately 72.75 MB for *RVO*, 66.7 MB for *DCP*, 23.2 MB for *FOT*, 0.21 MB for *VPR*, and 3.75 MB for EVLM. Thus, EVLM achieves substantial communication reduction compared with video-centric baselines while still preserving visual evidence for operator verification.

4) *Power consumption and battery impact*: Fig. 10e reports the average power consumption across the inspection mission, while Table V provides the per-segment measurements. Video-centric methods (*RVO*, *FOT*, and *DCP*) consume between 3.4 W to 3.8 W, as their workloads primarily involve CPU-based video processing. In contrast, VLM-based pipelines (*VPR* and EVLM) require approximately 5.6 W due to sustained GPU inference and transformer decoding. The battery usage shown in Fig. 10f and Table V remains small across all methods but is highest for the VLM-based pipelines, where $\eta = 2.87 \times 10^{-3} \%$ per segment. This increase reflects the longer processing time and higher power draw associated with multimodal reasoning. Using the energy model defined in Section V-G ($E_{\text{bat}} = 349.2 \text{ Wh}$ and $P_{\text{flight}} \approx 873 \text{ W}$), the computational energy fraction remains negligible relative to the UAV propulsion energy. Consequently, the effective flight-time impact $T_{\text{eff}} \approx 24 \cdot (1 - \eta)$ remains close to the nominal endurance. These results confirm that EVLM introduces only a negligible impact on the UAV’s 24 min flight time while enabling verification-ready semantic inspection outputs and

significantly reducing communication overhead.

VII. CONCLUSION AND FUTURE WORK

This paper introduced EVLM, an intent-driven onboard UAV power line inspection framework that enables vision-language semantic reasoning under strict edge compute and bandwidth constraints. EVLM combines (i) deterministic histogram-based key-frame filtering to bound VLM invocations, (ii) LoRA-based domain alignment of a lightweight VLM for intent-conditioned inspection, and (iii) semantic packaging that consolidates frame-level outputs into structured reports with a minimal set of evidence frames for operator verification. We implemented and fully deployed EVLM on an NVIDIA Jetson edge device and evaluated it on public inspection video sequences spanning 8 environments and 5 intent categories. Results show that EVLM reduces uplink volume relative to video-centric baselines while preserving interpretable, evidence-backed inspection outputs, and remains energy-feasible with negligible impact on the UAV’s propulsion-dominated energy budget. Future work will target lower end-to-end latency via VLM model optimization techniques and adaptive runtime control that tunes filtering and evidence selection based on mission objectives and available uplink capacity.

ACKNOWLEDGMENT

This work was supported by the Austrian FFG EdgeAI-Drone project, the EU ENFIELD project, and the Privacy@Edge project under the Research Council of Norway’s IKTPUS Programme.

REFERENCES

- [1] Eurelectric, “Distribution Grids in Europe: Facts and Figures,” 2020. [Online]. Available: <https://www.eurelectric.org/wp-content/uploads/2024/06/dso-facts-and-figures-11122020-compressed-2020-030-0721-01-e.pdf>
- [2] O. B. Schofield, N. Iversen, and E. Ebeid, “Autonomous Power Line Detection and Tracking System Using UAVs,” *Microprocessors and Microsystems*, 2022.
- [3] Y. He, Z. Liu, Y. Guo, Q. Zhu, Y. Fang, Y. Yin, Y. Wang, B. Zhang, and Z. Liu, “UAV-Based Sensing and Imaging Technologies for Power System Detection, Monitoring and Inspection: A Review,” *Nondestructive Testing and Evaluation*, 2025.
- [4] L. Lu, Z. Chen, R. Wang, L. Liu, and H. Chi, “YOLO-Inspection: Defect Detection Method for Power Transmission Lines Based on Enhanced YOLOv5s,” *Journal of Real-Time Image Processing*, 2023.
- [5] S. Rong, L. He, S. F. Atici, and A. E. Cetin, “Advanced YOLO-based Real-time Power Line Detection for Vegetation Management,” *IEEE Transactions on Power Delivery*, 2025.
- [6] M. Liu, Z. Li, R. Tao, Y. Li, and G. Sheng, “Self-Supervised Small Object Detection YOLO for Power Transmission Line Inspection,” *Transactions on Instrumentation and Measurement*, 2025.

- [7] Z. Azimi Ourimi, R. Farahani, R. Prodan, and C. Timmerer, "ELLMPEG: An Edge-based Agentic LLM Video Processing Tool," in *Proceedings of the ACM Multimedia Systems Conference 2026*, 2026.
- [8] L. Lovén, R. Farahani, I. Murturi, S. Sigg, and S. Dustdar, "Agentic Edge Intelligence: A Research Agenda," in *Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing*, 2025.
- [9] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "UAVs Meet Agentic AI: A Multidomain Survey of Autonomous Aerial Intelligence and Agentic UAVs," *arXiv preprint arXiv:2506.08045*, 2025.
- [10] A. Marafioti, O. Zohar, M. Farré, M. Noyan, E. Bakouch, P. Cuenca, C. Zakka, L. B. Allal, A. Lozhkov, N. Tazi *et al.*, "SmolVLM: Redefining Small and Efficient Multimodal Models," *arXiv preprint arXiv:2504.05299*, 2025.
- [11] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey," *arXiv preprint arXiv:2403.14608*, 2024.
- [12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," *Iclr*, 2022.
- [13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [14] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.
- [16] R. Farahani, Z. Azimi, C. Timmerer, and R. Prodan, "Towards AI-assisted Sustainable Adaptive Video Streaming Systems: Tutorial and Survey," *arXiv preprint arXiv:2406.02302*, 2024.
- [17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances in neural information processing systems*, vol. 28, 2015.
- [18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.
- [19] A. Zareian, K. D. Rosa, D. H. Hu, and S.-F. Chang, "Open-Vocabulary Object Detection Using Captions,"
- [20] J. Zhang, J. Huang, S. Jin, and S. Lu, "Vision-Language Models for Vision Tasks: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning Transferable Visual Models from Natural Language Supervision," in *International conference on machine learning*. PMLR, 2021.
- [22] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, "Scaling Up Visual and Vision-Language Representation Learning with Noisy Text Supervision," in *International Conference on Machine Learning*. PMLR, 2021.
- [23] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping LLanguage-Image Pre-Training for Unified Vision-Language Understanding and Generation," in *International Conference on Machine Learning*. PMLR, 2022.
- [24] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping Language-Image Pre-Training with Frozen Image Encoders and Large Language Models," in *International conference on machine learning*. PMLR, 2023.
- [25] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang *et al.*, "Grounded Language-Image Pre-Training," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.
- [26] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting Twenty-Thousand Classes using Image-Level Supervision," in *European conference on computer vision*. Springer, 2022.
- [27] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su *et al.*, "Grounding Dino: Marrying Dino with Grounded Pre-Training for Open-Set Object Detection," in *European conference on computer vision*. Springer, 2024.
- [28] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, "YOLO-World: Real-Time Open-Vocabulary Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 901–16 911.
- [29] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: A Visual Language Model for Few-Shot Learning," *Advances in Neural Information Processing Systems*, 2022.
- [30] W. Dai, J. Li, D. Li, A. Tiong, J. Zhao, W. Wang, B. Li, P. N. Fung, and S. Hoi, "InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning," *Advances in Neural Information Processing Systems*, 2023.
- [31] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge *et al.*, "Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution," *arXiv preprint arXiv:2409.12191*, 2024.
- [32] X. Chu, L. Qiao, X. Zhang, S. Xu, F. Wei, Y. Yang, X. Sun, Y. Hu, X. Lin, B. Zhang *et al.*, "MobileVLM V2: Faster and Stronger Baseline for Vision Language Model," *arXiv preprint arXiv:2402.03766*, 2024.
- [33] B. Zhou, Y. Hu, X. Weng, J. Jia, J. Luo, X. Liu, J. Wu, and L. Huang, "TinyLLaVA: A Framework of Small-scale Large Multimodal Models," *arXiv preprint arXiv:2402.14289*, 2024.
- [34] Y. Luo, X. Yu, D. Yang, and B. Zhou, "A Survey of Intelligent Transmission Line Inspection Based on Unmanned Aerial Vehicle," *Artificial Intelligence Review*, 2023.
- [35] J. Xing, G. Cioffi, J. Hidalgo-Carrió, and D. Scaramuzza, "Autonomous Power Line Inspection with Drones via Perception-Aware MPC," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023.
- [36] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio *et al.*, "Agilicious: Open-Source and Open-Hardware Agile Quadrotor for Vision-based Flight," *Science robotics*, 2022.
- [37] B. J. Souza, S. F. Stefenon, G. Singh, and R. Z. Freire, "Hybrid-YOLO for Classification of Insulator Defects in Transmission Lines Based on UAV," *International Journal of Electrical Power & Energy Systems*, 2023.
- [38] T. Santos, T. Cunha, A. Dias, A. P. Moreira, and J. Almeida, "UAV Visual and Thermographic Power Line Detection using Deep Learning," *Sensors*, 2024.
- [39] Q. Zhang, Y. Luo, H. Jiang, and K. Zhang, "Aerial Edge Computing: A Survey," *IEEE Internet of Things Journal*, 2023.
- [40] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan, "Bandwidth-Efficient Live Video Analytics for Drones via Edge Computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018.
- [41] M. Maresch and S. Nastic, "Vate: Edge-Cloud System for Object Detection in Real-Time Video Streams," in *2024 IEEE 8th International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2024.
- [42] K. Deng, D. Zhao, Q. Han, S. Wang, Z. Zhang, A. Zhou, and H. Ma, "Geryon: Edge-Assisted Real-Time and Robust Object Detection on Drones via mmWave Radar and Camera Fusion," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2022.
- [43] A. L. B. Vieira e Silva, H. de Castro Felix, F. P. M. Simões, V. Teichrieb, M. dos Santos, H. Santiago, V. Sgotti, and H. Lott Neto, "InsPLAD: A Dataset and Benchmark for Power Line Asset Inspection in UAV Images," *International Journal of Remote Sensing*, 2023.
- [44] "What is a Perceptual Hash?" <https://www.phash.org/>, accessed: 2026-02-20.
- [45] M. Jakubowski and G. Pastuszak, "Block-Based Motion Estimation Algorithms—A Survey," *Opto-Electronics Review*, 2013.
- [46] NVIDIA, "Tegrastats Utility," https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvlib_docs/index.html#page/DRIVE_OS_Linux_SDK_Development_Guide/Utilities/util_tegrastats.html, accessed: 2026-02-20.