Real-time Video-to-Text Summarization Using Object Detection-Guided Frame Selection and Multi-modal Language Models

Group Members and Contributions

Faris Alhammad	[Leader and Programmer]
Safwan Nabeel	[Programmer and Researcher]
Mishary Aldawood	[Product Manager and Designer]
Rayed Saidi	[Hardware Engineer]
Abdurahman Osilan	[Product Manager]

Project Mentor: Dr. Salman Khan

King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia

August 24, 2025

Project Objectives and Goals

Objective:
Design and deliver a , real-time video-to-text system that converts live or recorded video into concise summaries using object-guided frame gating (YOLOv11n), per-frame captioning (Florence-2-large), and temporal aggregation with a language model (Cohere Command-R+).

Goals:

- Enable two modes with the same core: live streaming with rolling one-minute summaries and batch processing with scene proposals.
- Reduce redundant compute by captioning only gated frames while preserving semantic coverage of important events.
- Produce faithful, grounded captions by injecting detector-aware object hints and discouraging hallucinations in aggregation.
- Generate short, coherent segment summaries that reflect the chronological evidence collected in each window.
- Expose a simple, modular web API and UI with observable artifacts at each stage (annotated frames, captions, summaries).
- Ensure robustness and privacy by avoiding training, handling LLM/network failures gracefully, and minimizing data retention.
- Keep all thresholds and cadences configurable so the system can be tuned for different hardware and scenarios without retraining.

Abstract

The exponential growth of video content has intensified the need for real-time video understanding and summarization. We present a practical system that integrates object detection-guided frame selection with multimodal captioning and large language model aggregation. The pipeline employs YOLOv11n to selectively process only frames containing semantically relevant objects, reducing computational overhead while preserving coverage. These gated frames are captioned using Microsoft's Florence-2-large vision-language model, which provides rich object-grounded descriptions. Captions and detected object counts are then accumulated over temporal windows and summarized into coherent narratives using Cohere's Command-R+ language model. This design enables real-time operation, batch video analysis, and efficient deployment in constrained environments. Unlike uniform sampling, our approach reduces redundant processing, improves semantic fidelity, and produces human-like summaries with explicit object grounding. We demonstrate applications across surveillance, accessibility, education, and scientific monitoring, highlighting a modular architecture suitable for diverse computational settings.

1 Introduction

Real-time video understanding at scale demands selective processing and robust language generation rather than uniform frame sampling. We present a training-free pipeline that operates in two modes: a live streaming path with minute-level rolling summaries, and a batch path that first proposes candidate frames via scene-change detection and then applies the same caption$\rightarrow$summary flow.

Motivation. Uniform or fixed-rate sampling wastes compute on redundant frames and often dilutes semantic coverage. Object detection provides a simple, content-aware trigger for when to spend vision-language inference. Modern vision-language models can produce rich, grounded single-image descriptions, but they do not natively maintain temporal state. A lightweight temporal layer is therefore needed to stitch per-frame descriptions into short, coherent, time-bounded summaries.

Scope. The pipeline is training-free: all components (detector, captioner, LLM) are used as-is. We do not perform task-specific fine-tuning. Temporal reasoning is handled post hoc by aggregation of single-frame captions; no video transformer is used.

Contributions. This work makes four practical contributions:

- *Object-guided gating for efficiency.* A detector-first policy that processes only semantically relevant frames.
- *Florence-2-based captioning with detector-aware grounding.* Object-aware caption generation without region-level processing.
- *LLM-based temporal aggregation.* Minute-level summarization that produces coherent segment summaries.
- *End-to-end, real-time system.* A modular implementation supporting both live streams and batch video processing.

The remainder of this paper is organized as follows. Section II reviews related work. Section III details our methodology. Section IV describes the implementation. Section V discusses applications, limitations, and future extensions. Section VI concludes.

2 Related Work

Research on video-to-text spans three threads most relevant to our system: object-guided frame selection, vision–language captioning, and language-model aggregation for temporal coherence.

2.1 Object-Guided Frame Selection and Scene Proposal

Uniform or fixed-rate sampling has been the default for years, sometimes aided by clustering over visual features to pick keyframes [1, 2]. Event- and object-driven policies replace blind sampling with content-aware triggers, using detectors like YOLO to decide when to spend compute [3]. Our gating follows this line: we use YOLOv11n as a first-pass filter over every frame and only caption frames when detections are present [4]. For batch processing, shot-boundary or scene-change heuristics complement object gating; we adopt an HSV-histogram Bhattacharyya distance to propose candidate frames, consistent with classic shot detection methods while remaining lightweight.

2.2 Vision–Language Models for Per-Frame Captioning

Early single-image captioners combined CNN encoders with autoregressive decoders (e.g., ViT + GPT-2) and were often used as drop-in captioners for video keyframes [5, 6]. More recent promptable vision–language models (VLMs) like BLIP-2 decouple the vision encoder and language model via lightweight adapters, improving zero-shot transfer and keeping inference costs manageable [7]. Florence-2 extends this trend with a multi-task interface that supports short and detailed captions, grounding, and OCR within a single model [8]. Unlike methods that crop and caption detected regions independently, we use detector outputs as *textual hints* to bias captions toward grounded objects while keeping a single caption per frame for speed.

2.3 Temporal Aggregation with Language Models

Video transformers (e.g., TimeSformer, ViViT) integrate temporal attention directly in the vision stack [9, 10], but they are heavier than our deployment target and require training. A practical alternative is a two-stage pipeline: generate short captions on selected frames, then aggregate them with a language model. Prior work shows that LLMs can reliably stitch short, factual snippets into coherent paragraphs when given light structure (timestamps, object summaries, or templates) [11]. We follow that paradigm and use an external LLM to produce minute-level summaries from cleaned captions plus object-frequency statistics.

2.4 Evaluation Protocols for Captioning and Summarization

Reference-based metrics from image/video captioning remain standard for benchmarking (CIDEr, BLEU, METEOR, ROUGE-L) [12–15], with MSVD and MSR-VTT commonly used datasets [16, 17]. For deployment-focused systems like ours that do not train on these datasets, reference-free indicators are useful: CLIP-based similarity between frames and generated text [18], object-mention consistency between captions and detector classes, and small-scale human ratings for coherence, faithfulness, and usefulness [19].

3 Methodology

Our pipeline has three stages: (1) *object-guided gating*, (2) *per-frame captioning with detector-aware hints*, and (3) *LLM-based temporal aggregation*. Live and batch modes share this core.

3.1 System Architecture Overview

Each incoming frame f_t is first evaluated by a real-time detector. Frames with at least one detection are queued for captioning; others are skipped. Captions and detector outputs are stored with timestamps. On a fixed schedule (one-minute sliding window in live mode, scene-based windows in batch mode), a language model receives (i) cleaned per-frame captions and (ii) an object-frequency sketch derived from detector outputs and returns a concise paragraph.

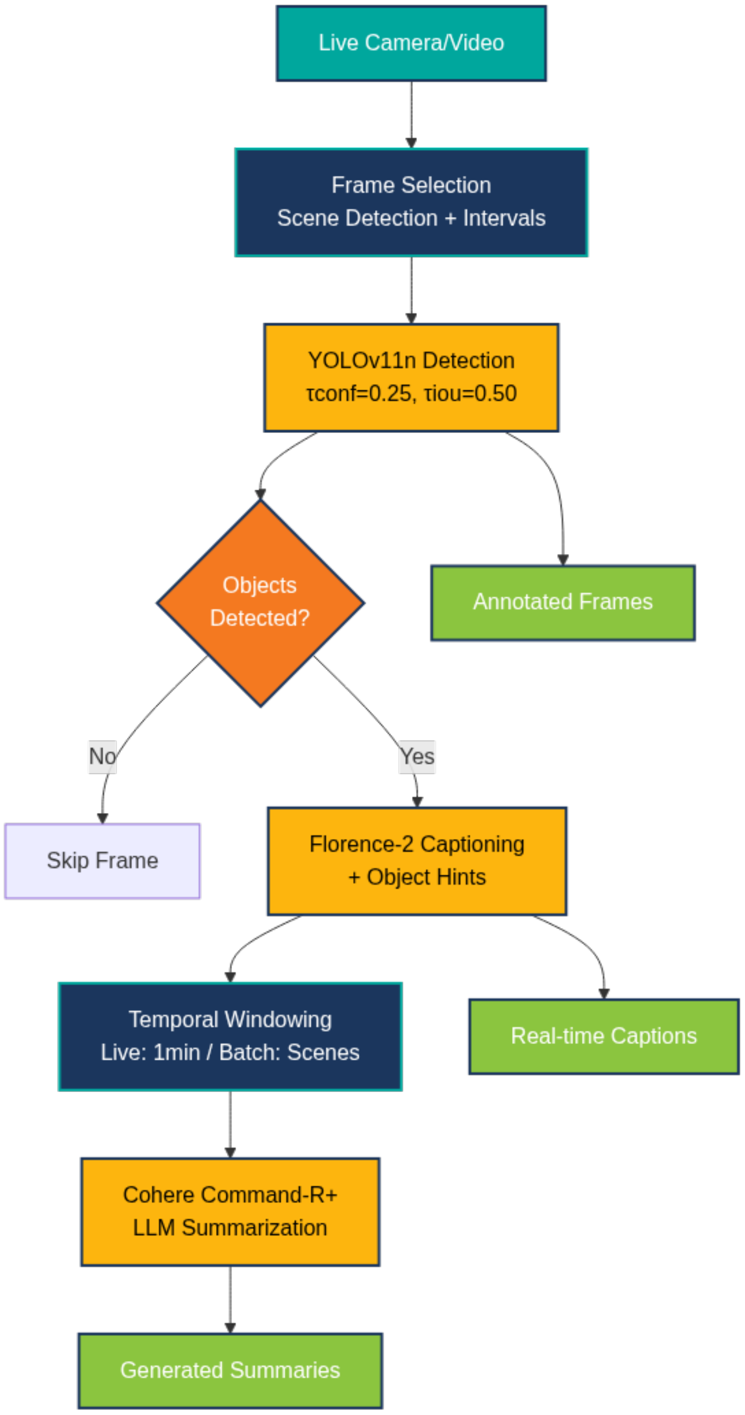


Figure 1: End-to-end pipeline. Frames are gated by YOLOv11n; gated frames are captioned by Florence-2 with detector-aware hints; cleaned captions and per-class counts are aggregated in a time window and summarized by Cohere Command-R+. Live mode uses 1-minute windows; batch mode uses scene proposals or interval fallback.

3.2 Object-Guided Gating

We use YOLOv11n [4] as a first-pass filter on every frame; the detector vocabulary is COCO (80 classes). With confidence threshold τ_{conf} and NMS IoU τ_{iou},

$$O_t \;=\; \mathrm{YOLO}(f_t;\, \tau_{\text{conf}}, \tau_{\text{iou}})\,.$$

If $O_t = \varnothing$, the frame is dropped. If $O_t \neq \varnothing$, we derive two textual artifacts: (i) a comma-joined list of detected classes and (ii) a *class:count* map. These artifacts serve as hints for captioning and as evidence for temporal aggregation.

3.3 Scene Proposal for Batch Videos

For uploaded videos, we propose candidate frames via shot-change cues. We compute HSV color histograms per frame and the Bhattacharyya distance between consecutive frames:

$$d_{\text{scene}}(t) \;=\; \sqrt{1 - \sum_i \sqrt{H_t(i)\, H_{t-1}(i)}}\,,$$

where H_t is a normalized histogram. Frames with $d_{\text{scene}}(t) \geq \tau_{\text{scene}}$ are treated as scene boundaries; we sample a subset for captioning. If no boundaries are found, we fall back to interval sampling. Object gating still applies before captioning.

3.4 Per-Frame Captioning with Florence-2

Selected frames are captioned using Florence-2-large [8]. We request a detailed description per frame. To bias captions toward grounded objects without per-region crops, we append the detector's class hints to the caption string. Let $\tilde{c}_t$ be Florence-2's caption for f_t and $\mathcal{N}_t$ the detector hint; we store

$$c_t \;=\; \tilde{c}_t \;+\text{“ (Detected: ”}\, \mathcal{N}_t \text{ “)”}\,.$$

Before summarization, we strip the parenthetical to obtain a *clean* caption $\hat{c}_t$ and keep counts separately.

3.5 Temporal Aggregation (Sliding Window)

Captions and detector counts are accumulated in windows. In live mode we use one-minute windows aligned to wall-clock time; in batch mode we aggregate within scene windows.

$$C_W \;=\; \{\, (t_i, \hat{c}_i, O_i) \mid t_i \in W \,\}\,.$$

From $\{O_i\}$ we compute a compact object sketch using per-class *max* counts in W:

$$\mathrm{MaxCount}_W(\mathrm{obj}) \;=\; \max_{(t_i, O_i) \in W} \mathrm{Count}_{O_i}(\mathrm{obj})\,,$$

and retain a small set of representative captions (e.g., the last K unique sentences).

3.6 LLM-Based Summarization with Cohere

At each window boundary, a language model (Cohere Command-R+ [20]) receives the ordered list of cleaned captions and the object sketch and returns a short paragraph. The prompt is deterministic and instructs the model to rely only on provided evidence and avoid adding unseen objects or actions.

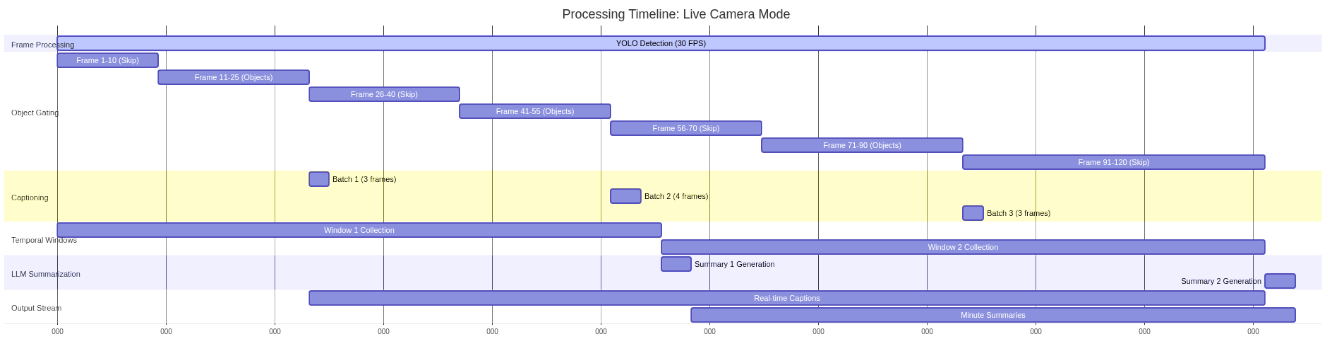


Figure 2: Processing timeline in live mode. YOLO runs on every frame; only every Nth gated frame is queued for captioning (small batches). Captions and per-class counts are accumulated in the active window; at the boundary, the LLM produces a concise summary.

Table 1: Default operational settings (tunable).

Detector confidence τ_{conf}	0.25
NMS IoU τ_{iou}	0.50
Caption cadence N (live)	10–15 (only every Nth *gated* frame)
Caption batch size	3–4 frames
Window length (live)	60 s
Windowing (batch)	scene windows; interval fallback if none
Florence-2 decoding	beam search, `num_beams=3`, no sampling
LLM template	fixed, evidence-only; deterministic decoding

4 Implementation

This section lists concrete settings and engineering details.

4.1 Runtime and Models

Python 3.11; PyTorch with CUDA; Ultralytics YOLO; Hugging Face Transformers; OpenCV; FastAPI with Uvicorn; single-page JS frontend. Models: YOLOv11n for detection [4], Florence-2-large for captioning [8], Cohere Command-R+ for summarization [20].

```
1. POST /api/upload-image   # single-image detection + caption
2. POST /api/upload-video   # batch: scene proposals -> caption -> summary
3. WS   /ws/camera          # live camera with rolling minute summaries
4. GET  /health             # model readiness and liveness
```

4.2 Robustness and Ops

- **Queues and batching.** Queue and batch sizes cap GPU memory spikes.
- **Failures.** Camera discovery cycles through indices; LLM calls use short timeouts and retries. If summarization fails, evidence is retained and retried.

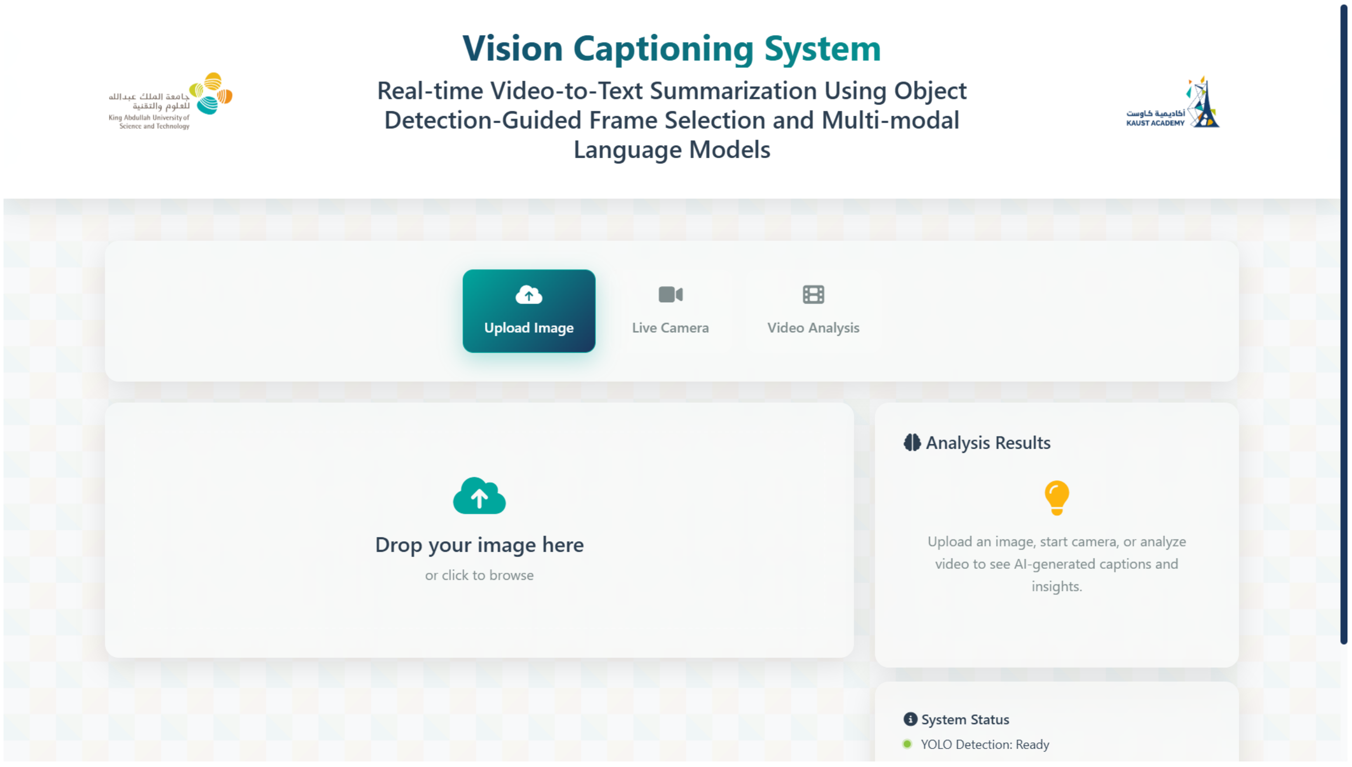
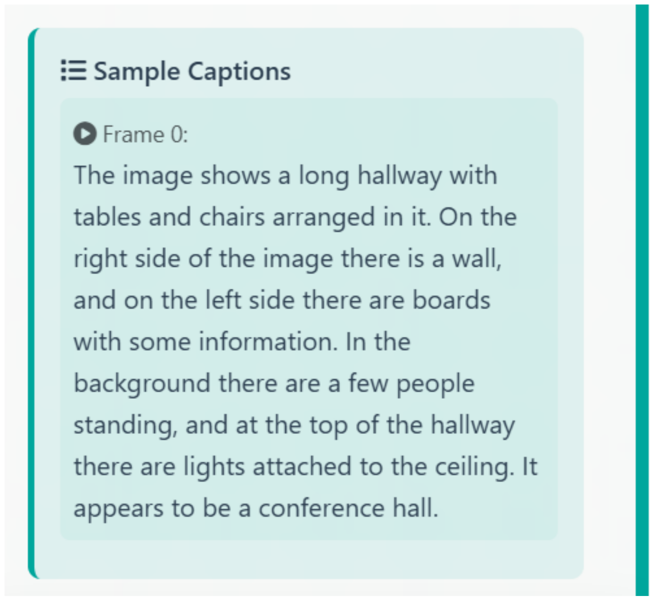

Vision Captioning System
Real-time Video-to-Text Summarization Using Object Detection-Guided Frame Selection and Multi-modal Language Models
Upload Image
Live Camera
Video Analysis
Drop your image here
or click to browse
Analysis Results
Upload an image, start camera, or analyze video to see AI-generated captions and insights.
System Status
YOLO Detection: Ready

Figure 3: Single-page web interface with three modes: Upload Image, Live Camera, and Video Analysis. The UI displays annotated frames, per-frame captions, detected-object tags, and rolling minute summaries in live mode.

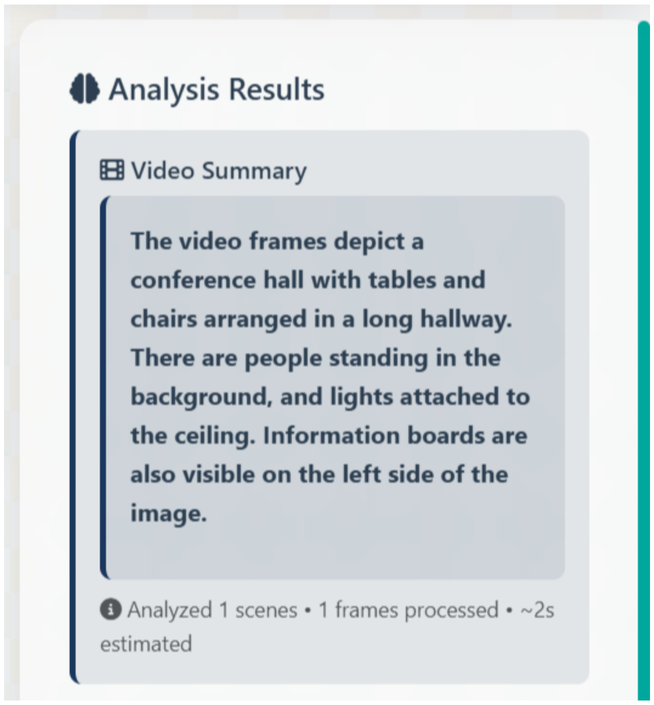
- **Security.** Secrets via environment variables; file-type checks and duration caps on uploads.

4.3 Front-End Interfaces

Single-page web UI with three tabs: *Upload Image*, *Live Camera*, and *Video Analysis*. It displays annotated frames, per-frame captions, detected-object tags, and rolling minute summaries.



Frame 0:
The image shows a long hallway with tables and chairs arranged in it. On the right side of the image there is a wall, and on the left side there are boards with some information. In the background there are a few people standing, and at the top of the hallway there are lights attached to the ceiling. It appears to be a conference hall.



The video frames depict a conference hall with tables and chairs arranged in a long hallway. There are people standing in the background, and lights attached to the ceiling. Information boards are also visible on the left side of the image.

Analyzed 1 scenes • 1 frames processed • ~2s estimated

Figure 4: Example outputs from the batch video path on footage from the KAUST Academy poster session. *Left:* a selected frame's caption (Florence-2). *Right:* scene-level summary (Cohere Command-R+).

5 Discussion

5.1 Strengths and Design Trade-offs

Efficiency through gating. Cadence control (caption every Nth gated frame and batch size 3–4) keeps latency predictable in live mode while reducing redundant processing.

Faithfulness via detector-aware hints. The summarizer prompt explicitly instructs against inventing objects or actions, which limits drift from the original content.

Separation of concerns. Vision remains fully stateless. Temporal coherence is handled in a small text layer that consumes cleaned captions plus object sketches. This separation simplifies debugging and isolates failures.

Mode parity. Live and batch paths share the same caption to summary core, with batch mode adding lightweight scene proposal.

Operational clarity. The system provides user-visible artifacts at each stage for diagnosing missed content and tuning thresholds.

5.2 When the Design Excels

Static cameras with intermittent activity. Object-driven gating filters idle periods, so most captioning occurs when something meaningful appears.

Scenes with visually salient nouns. Environments where target semantics are captured by COCO classes benefit from the hint mechanism and produce faithful summaries.

Deployment without training. All components run as-is, reducing engineering time and risk.

5.3 Known Limitations

Gate coverage. YOLOv11n uses a COCO 80-class vocabulary. Domain objects outside this set will not trigger captioning. Batch mode mitigates this with scene proposals and interval fallback.

Temporal reasoning is post hoc. Actions that depend on motion patterns may not be captured if per-frame captions do not surface them.

Latency and throughput ceiling. Florence-2 decoding dominates per-window cost. Increasing the caption cadence or batch size improves coverage but raises end-to-end latency and memory use.

LLM dependency. Network failures or rate limits affect only the summary stage but can delay paragraph generation. Costs are proportional to window frequency and prompt size.

Consistency assumptions. The object sketch uses per-class maxima within the window. This emphasizes peaks but may understate sustained moderate counts.

5.4 Scalability and Extensions

Gate robustness. Add a motion or frame-difference fallback that samples one frame every S seconds when the gate stays silent for T seconds.

Grounding options. For scenes with dense objects, consider passing detector boxes to Florence-2 as region prompts and composing a compact caption from region phrases.

Offline summarization. Replace the cloud LLM with a small local model for edge deployments. Keep the same prompt and evidence format to preserve behavior.

Audio as a future modality. Meeting and lecture scenarios benefit from audio cues. A simple speech-to-text channel with the same windowing can be fused at the summarizer.

Domain adaptation. If a small set of domain objects is critical, either fine-tune a lightweight detector or add a class-agnostic motion cue to increase recall without retraining.

5.5 Applications

Our system supports a diverse range of practical applications across academic, industrial, and societal domains:

- **Lecture Summarization**: Generate concise summaries of recorded classroom sessions to enhance accessibility and review.
- **Laboratory Documentation**: Provide real-time monitoring and logging of experimental procedures for reproducibility.
- **Research Video Analysis**: Support disciplines such as ecology, anthropology, and materials science by summarizing field recordings.
- **Smart Campus Monitoring**: Enable privacy-preserving surveillance for safety and security in academic environments.
- **Event Documentation**: Automatically document conferences, seminars, and academic events with scene-level summaries.
- **Museum and Cultural Heritage**: Produce recaps of exhibitions and visitor interactions for archival and operational use.

- **Archaeological Site Monitoring**: Record excavation progress and findings in real time using fixed cameras.
- **Performance and Arts Analysis**: Summarize theatrical or musical rehearsals and performances for archival purposes.
- **Industrial Process Monitoring**: Capture key stages of manufacturing workflows for quality control and training.
- **Wildlife and Environmental Monitoring**: Summarize animal presence, behavior, and environmental changes from long-term camera deployments.
- **Climate and Scientific Studies**: Extract meaningful events from time-lapse or continuous environmental recordings.
- **Sports and Training Analytics**: Provide short summaries of training sessions and athletic drills for coaching and review.

6 Conclusion

We presented a practical system for real-time video-to-text summarization that combines object-guided gating, strong per-frame captioning, and LLM-based temporal aggregation. Frames are first filtered by YOLOv11n so that compute is spent only when salient objects are present. Selected frames are then captioned by Florence-2-large, and cleaned captions plus detector-derived object counts are aggregated over time windows and condensed by Cohere Command-R+ into short, human-readable paragraphs. The same core applies to live streams (minute windows) and batch videos (scene-based windows).

What this contributes. The design turns uniform sampling into content-aware processing, improves faithfulness by injecting detector-informed hints into captions, and separates temporal coherence into a lightweight text stage. Each stage is observable in the UI, making the system debuggable and tunable in deployment.

Limitations in scope. The gate covers COCO's 80 classes; domain objects outside this set will not trigger captioning. Temporal reasoning is post hoc; actions not surfaced at the caption level may be underrepresented in summaries. Summarization depends on an external LLM, although failures are isolated to the paragraph stage.

Future directions. Add a motion-based fallback when the gate is silent for extended periods, optionally pass detector boxes as region prompts for denser grounding when needed, provide an offline summarizer for edge deployments, and consider audio as a complementary modality.

The system delivers selective compute, grounded single-frame descriptions, and coherent minute-scale summaries without any training, making it suitable for real-time and batch scenarios where reliability, latency, and operational clarity matter.

References

[1] B. T. Truong and S. Venkatesh, "Video abstraction: A systematic review and classification," *ACM transactions on multimedia computing, communications, and applications*, vol. 3, no. 1, pp. 3–es, 2007.

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[3] 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, pp. 779–788.

[4] Ultralytics, "Yolo11: Real-time object detection," https://github.com/ultralytics/ultralytics, 2023, accessed: 2025.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[7] J. Li, D. Li, C. Xiong, 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, pp. 19 730–19 742.

[8] B. Xiao, H. Wu, W. Xu, X. Dai, H. Hu, Y. Lu, M. Zeng, C. Liu, and L. Yuan, "Florence-2: Advancing a unified representation for a variety of vision tasks," *arXiv preprint arXiv:2311.06242*, 2023.

[9] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" in *International Conference on Machine Learning*. PMLR, 2021, pp. 813–824.

[10] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6836–6846.

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.

[12] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[14] M. Denkowski and A. Lavie, “Meteor universal: Language specific translation evaluation for any target language,” in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 376–380.

[15] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text summarization branches out*, 2004, pp. 74–81.

[16] D. Chen and W. B. Dolan, “Collecting highly parallel data for paraphrase evaluation,” in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, 2011, pp. 190–200.

[17] J. Xu, T. Mei, T. Yao, and Y. Rui, “Msr-vtt: A large video description dataset for bridging video and language,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5288–5296.

[18] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi, “Clipscore: A reference-free evaluation metric for image captioning,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 7514–7528.

[19] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “Bertscore: Evaluating text generation with bert,” in *International Conference on Learning Representations*, 2021.

[20] Cohere, “Command r+: Cohere’s flagship text generation model,” https://cohere.com/command, 2024, accessed: 2025.