



GSTREAMER AS A FOUNDATION FOR VISION AI EDGE SOLUTIONS

TOMASZ JANCZAK (INTEL)
JAN GÓRECKI (INTEL)
STEWART TAYLOR (INTEL)

GSTREAMER CONFERENCE 2026
10-11 OCTOBER 2026 · PRAGUE, CZECH REPUBLIC

AGENDA

01

Background

5 min

02

Edge deployments

10 min

03

Vision AI evolution

10 min

04

Summary and Q&A

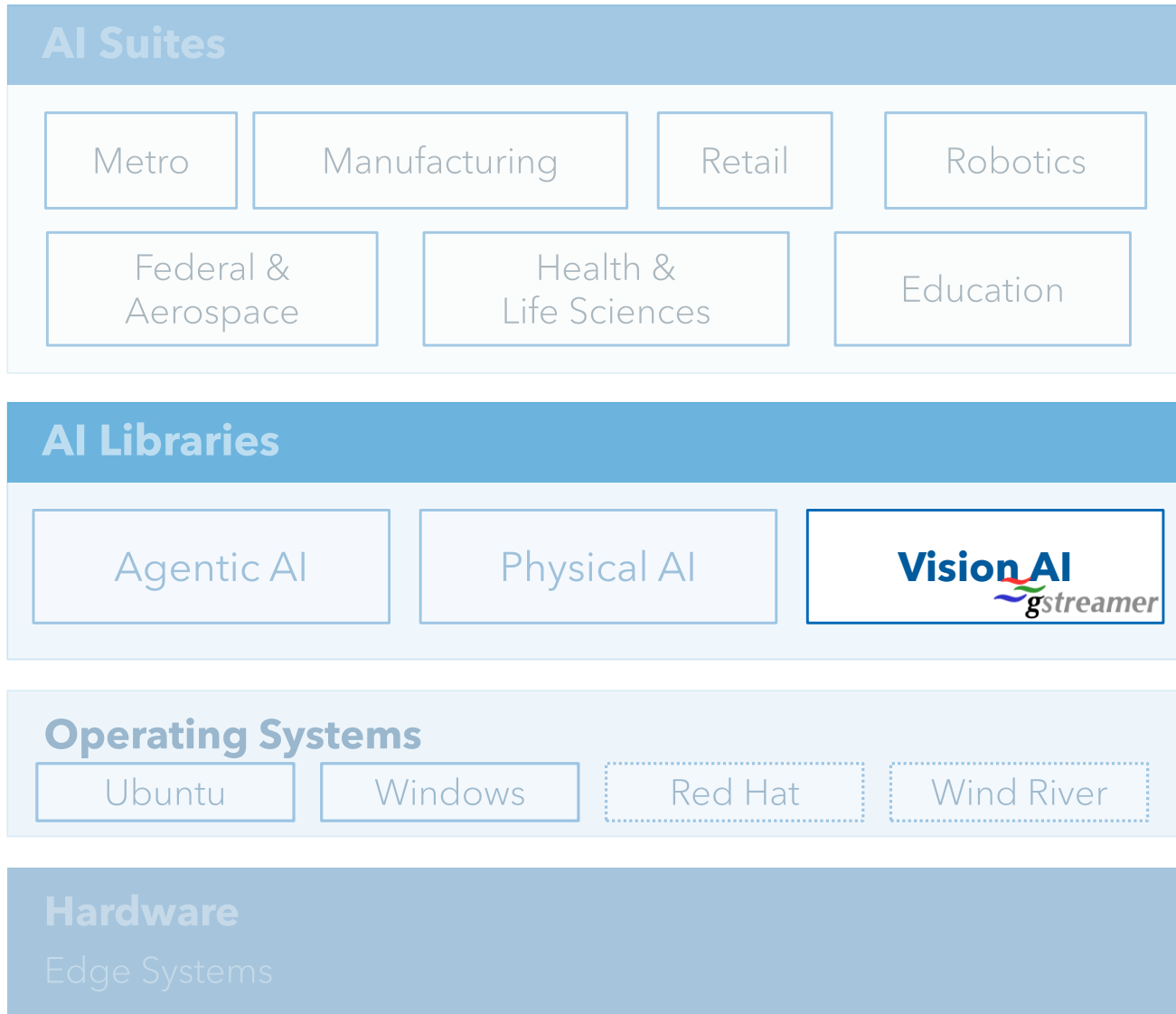
5 min

5 min

01 BACKGROUND

How Intel uses GStreamer

Intel Open Edge Platform



- GStreamer, extended by DL Streamer, is the **Vision AI foundation** in Intel's edge software.
- It powers **AI suites across metro, manufacturing, retail, smart cities and robotics** – and increasingly feeds **Agentic AI**.
- This deck is here to **share how Intel uses GStreamer at scale**, what Intel adds on top, and to **spark discussion** with the community.

GStreamer in Vision AI

Real-time video analytics



DL Streamer: 20+ elements (sources, AI inference, analytics, publishing); 10-100s of streams in parallel per device.

Offline video search



Related talks at this conference:

Computer Vision pipelines:

"Unifying GstAnalytics Metadata: Cross-Vendor Interoperability"

Performance and Prototyping:

"A GStreamer-Based Performance Evaluation Tool"

This talk: taking GStreamer / DL Streamer to production,
and evolution beyond traditional computer vision

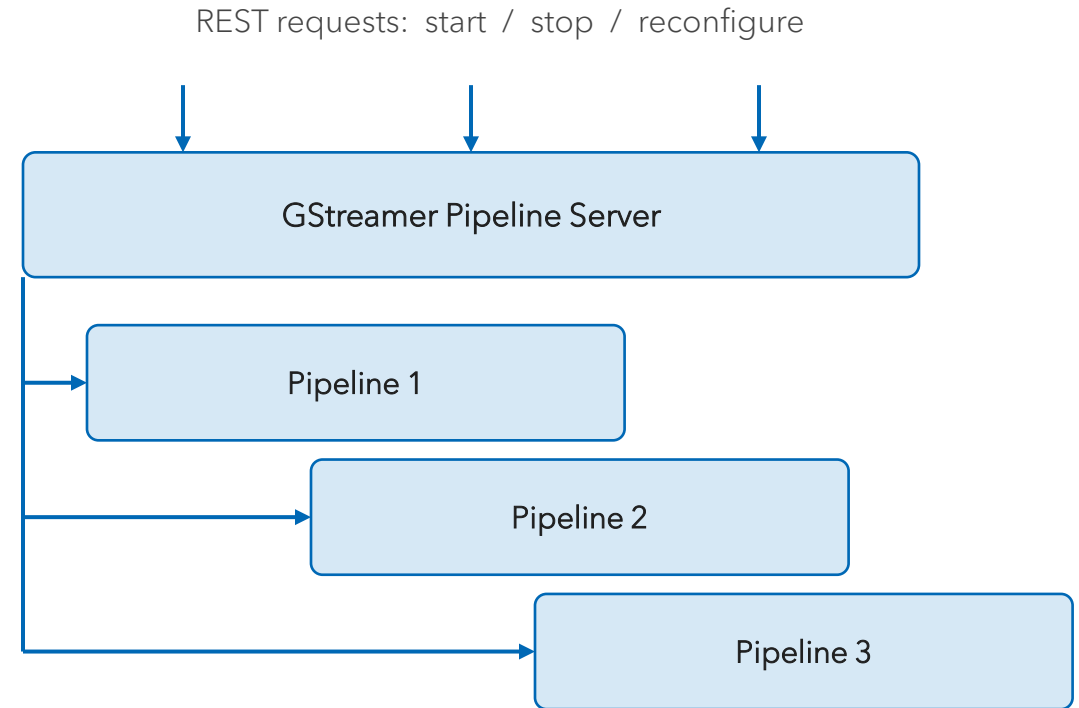
10 min

02 EDGE DEPLOYMENTS

Operating GStreamer pipelines as reliable edge services

GStreamer as a pipeline service

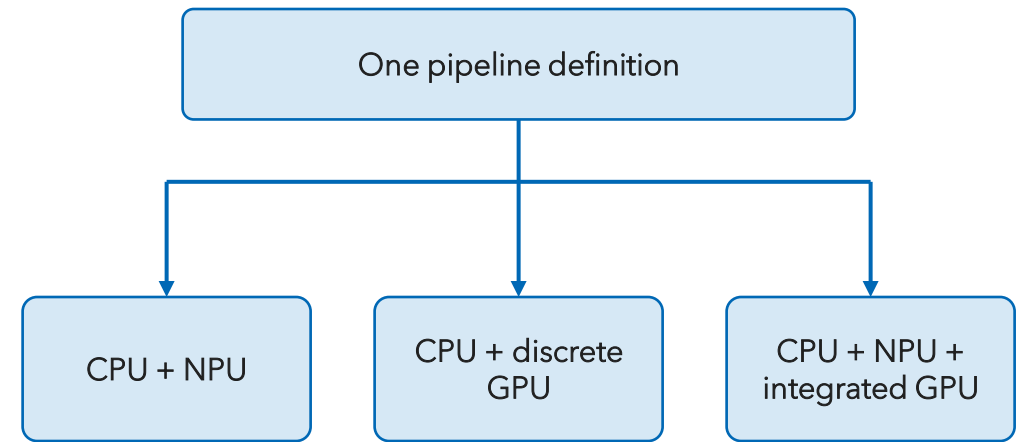
- [Problem](#): run predefined pipelines as a service, each started, stopped and reconfigured independently.
- [GStreamer today](#): gst-launch is great for single-pipeline demos; multi-pipeline deployments are custom C++ or Python apps.
- [Intel](#): DL Streamer Pipeline Server - a REST API over pipelines written in GStreamer syntax.



Question: should a pipeline-service abstraction be part of the standard distribution?

Portable pipeline definitions

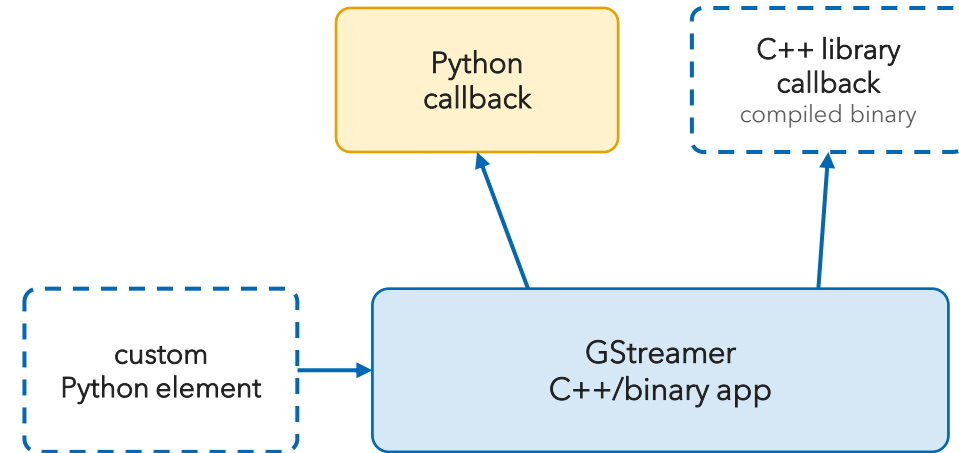
- [Problem](#): one definition across platforms – media on GPU or CPU, AI on CPU, GPU or NPU, with device, batch size and pool size differing each time.
- [GStreamer today](#): bins hide complexity, but coverage is uneven – example: good decodebin, incomplete encodebin.
- [Intel](#):
 - (1) Bin-type AI inference elements that hide XPU differences, yet full pipelines still differ across deployment devices.
 - (2) Attempted to define GStreamer pipeline templates + specializations, but on hold.



Question: abstract bins that hide vendor / platform / OS differences, or pipeline templates specialized at run-time?

Pipeline extensibility: callbacks

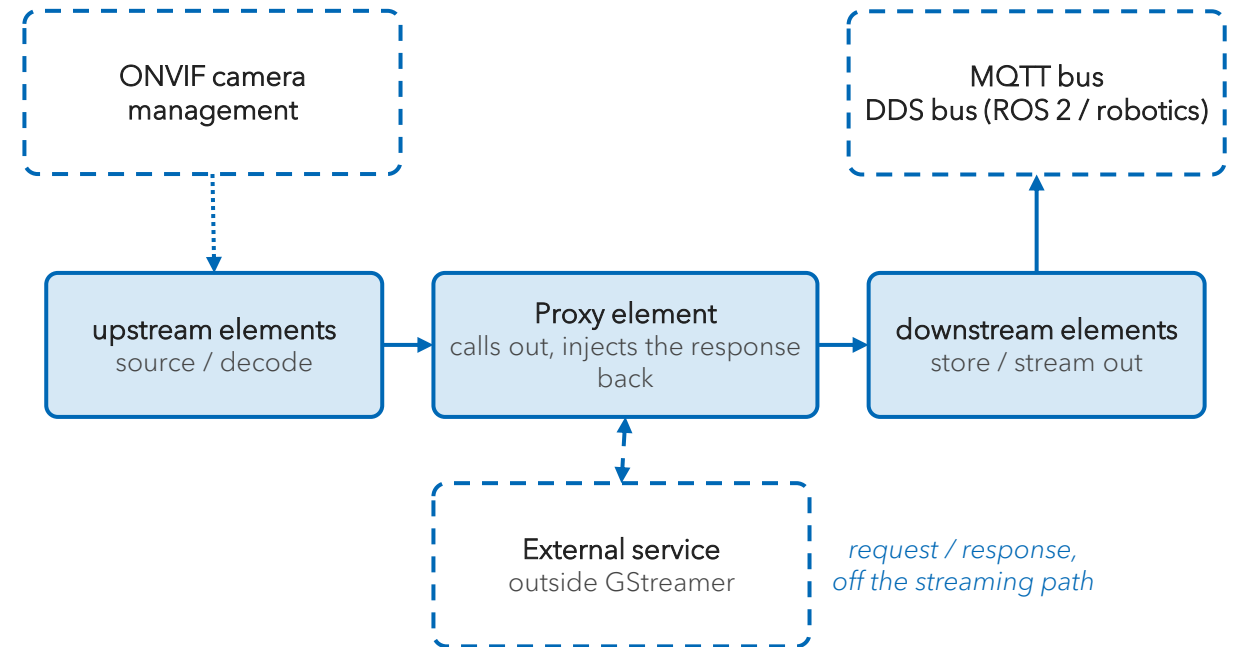
- Problem: customize one step of a pipeline without learning GStreamer internals or writing a full element.
- GStreamer today: Python elements work well, but customizing one decision point means writing a whole element - **no method for a binary element to hand a decision point to user Python**
- Intel: Python elements for custom logic, plus selective callbacks delivered as compiled C++ libraries.



Question: should a binary GStreamer element expose extension points that a Python callback can fill?

Integration with external services

- Problem: not everything belongs inside GStreamer – VLM, AI agents, camera management, message buses run as separate services.
- GStreamer today: edges covered by sources / sinks; mid-pipe calls – no standard answer.
- Intel:
 - Adds a proxy element for mid-pipe VLM, but break pipelines in other use cases
 - Custom solutions for integration with non-streaming / control elements



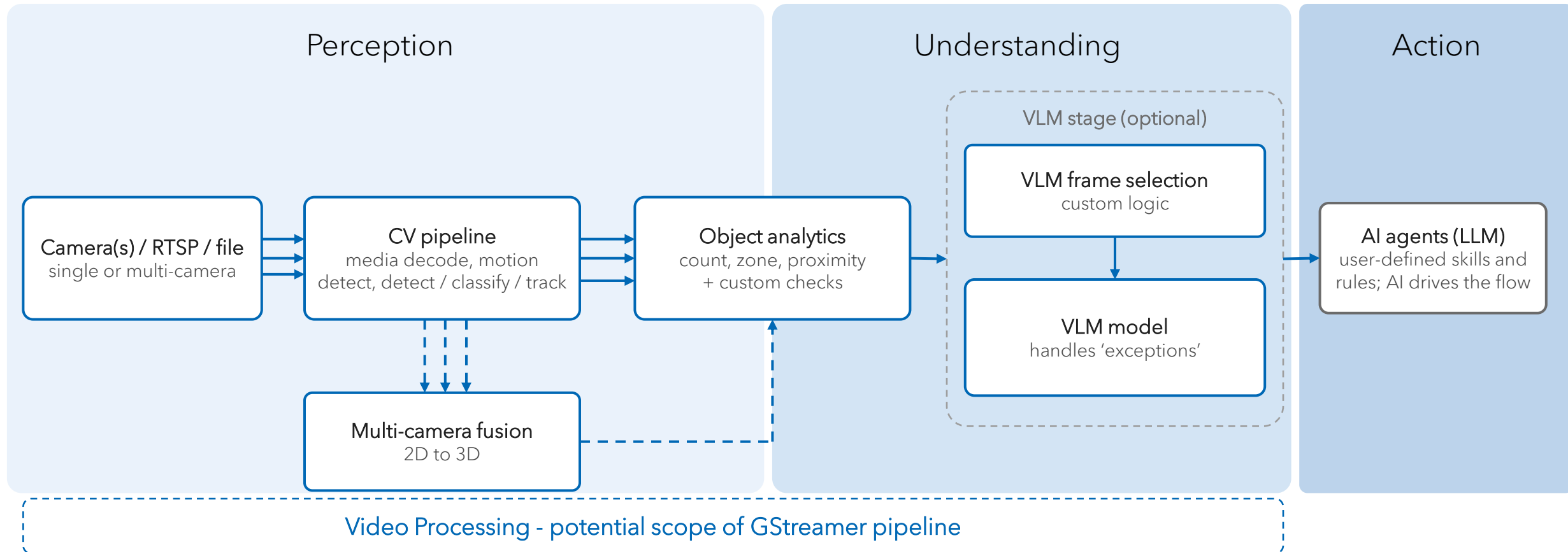
Questions: standard path for mid-pipe integration with third-party services?
"GStreamer as a Pipeline Service" for integration with non-streaming / control services?

10 min

03 VISION AI EVOLUTION

CV, vision-language models, and agentic workflows

Generalized Vision AI pipeline



Challenges:

- 1) mid-pipe integration with external services – already discussed
- 2) dynamic decisions in downstream – frame / prompt selection
- 3) cascaded pipelines or optional branches
- 4) retrospective data inspection (short-term retention)

Dynamic decisions: VLM frame and prompt selection

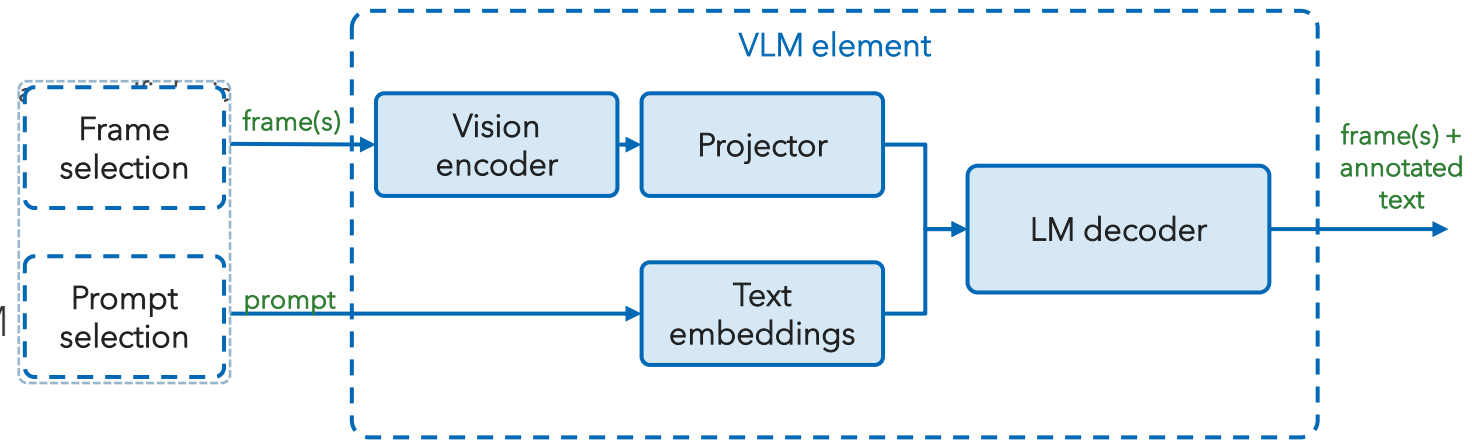
Problems:

- (1) a VLM is a graph of models; ready-made runtimes (vLLM, OpenVINO GenAI) exist
- (2) too costly on every frame – frames must be selected, e.g. from GstAnalytics metadata

Intel:

- (1) one GStreamer element covers the whole VLM graph – local or proxied
- (2) simple periodic frame selection built into the VLM element; a GStreamer Python element for custom frame and prompt selection

Challenges: how to generalize frame/prompt selection upstream and pass it downstream?



Question: new type of metadata, python callbacks, generic mechanism to set property at specific frame time?

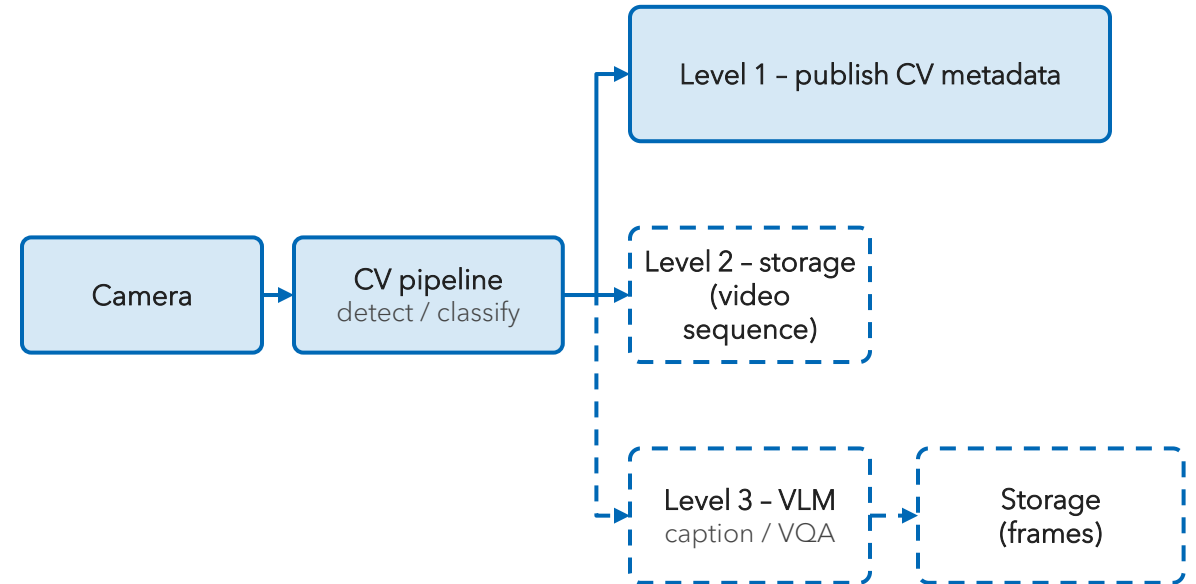
Cascaded pipelines: dynamic and triggered processing

Problem: multi-output pipeline, with optional stages triggered on demand (in a cascade)

Intel:

- (1) 'tee' branches followed by 'custom Python element'
- (2) explicit buffer copy in each branch to decouple different-frequency processing

Challenges: merging info from diverged branches (currently outside of GStreamer)



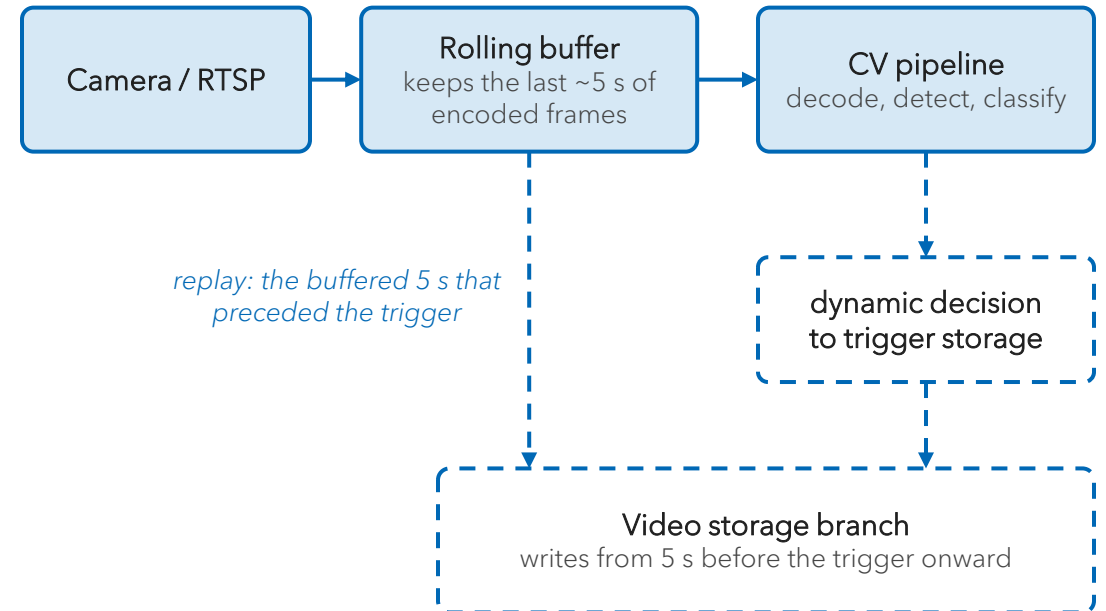
Question: dynamic branches are possible today, but they were not intended for different-frequency
Is it a 'desired' design pattern in GStreamer?

Retrospective data inspection: short-term data retention

Problem: a trigger of 'dynamic branch' may require historical data frames – a stored sequence should start 2 seconds before a person was detected

Intel: store entire video + analytics data and retrieve 'sequence of interest' outside of GStreamer

Challenges: how long to retain, and at what cost?



Question: should a rolling replay buffer be a standard GStreamer building block?

5 min

04 SUMMARY

Community discussion

[LINK](#)

Which topics would you most like to see prioritized in GStreamer?

- A. GStreamer as a pipeline service
- B. Portable pipeline definitions
- C. Python callbacks for pipeline extensibility
- D. Integration with external services
- E. VLM as a native GStreamer element
- F. Dynamic configuration from upstream
- G. Short-term data retention element
- H. No changes needed / existing mechanisms are sufficient



Choose max 3 options (A-H). Vote for your highest priority.

Results

Which topics would you most like to see prioritized in GStreamer? (pick max 3)

Integration with external services



VLM as a native GStreamer element



Dynamic configuration from upstream



GStreamer as a pipeline service



Short-term data retention element



Python callbacks for pipeline extensibility



Portable pipeline definitions



No changes needed / existing mechanisms are sufficient

