1. End-to-End System Design and Data Flow

This document outlines the end-to-end data flow and system design for a distributed edge inference pipeline
involving GigE cameras, GStreamer preprocessing, YOLOv7 inference, and cloud orchestration.

Camera Acquisition and Frame Handling

Industrial GigE cameras capture high-resolution (12MP) frames at 20 FPS, synchronized via PTP or software alignment.
SDKs manage stream initiation, buffering, and timestamping, exposing camera controls (exposure, gain, etc.).
Dedicated threads handle frame grabbing and move captured data into FIFO buffers for downstream processing.

GigE Frame Capture with aravissrc

The Aravis library and GStreamer plugin `aravissrc` are used for GigE Vision camera compatibility. Example pipeline:

gst-launch-1.0 aravissrc ! video/x-bayer,format=rggb,width=1920,height=1080,framerate=20/1 ! bayer2rgb ! videoconvert ! video/x-raw,format=RGB,width=1920,height=1080 ! queue ! v4l2sink device=/dev/videoX

GStreamer-Based Preprocessing

GStreamer orchestrates conversion (Bayer $\rightarrow$ RGB), scaling, and streaming into loopback devices for inference.
Elements include: aravissrc, bayer2rgb, videoconvert, queue, and v4l2sink. Buffers are used to prevent stalls.

YOLOv7 Deep Learning Inference

Using OpenCV to capture virtual camera frames and perform YOLOv7 inference with ONNX/TensorRT for GPU acceleration.
Includes preprocessing, inference, and postprocessing with NMS and metadata tagging.

Results Collection and Postprocessing

Detections yield class labels, bounding boxes, and confidence scores. Postprocessing filters overlapping boxes
and formats results (JSON/PB2) with metadata such as frame, timestamp, and camera ID.

ML Pipeline Orchestration and Model Management

Kubeflow Pipelines automate workflows from ingestion to retraining. Triton Server manages dynamic model versions
and canary deployments with zero-downtime reloads.

Strategies for Low Latency and High Throughput

Multithreading and async batching optimize performance. Zero-copy buffers minimize data serialization overhead.
GigE tuning (jumbo frames, QoS) and edge inference improve responsiveness.

Mitigating Frame Drops and Delays

Timestamping and buffered processing mitigate transient delays. Metrics (FPS, drop rate) are monitored via Prometheus/Grafana.

Optimizing Edge vs Cloud Inference

Edge-centric detection minimizes bandwidth; only atypical events are sent to the cloud. Quantized models maintain accuracy.

Continuous Model Retraining, Deployment, and Rollback

Kubeflow automates retraining and Triton ensures non-disruptive hot-roll updates. Versioning and rollback are handled via metadata.

Monitoring System Health and Model Drift

Operational metrics include FPS, drop rate, inference latency, and accuracy. Drift detection triggers retraining and alerts.

Stage	Key Functions	Example Tech	Metrics to Monitor
Camera/SDK	Capture, buffer, timestamp	Vendor SDK (C++/PY)	FPS, drop rate
GStreamer	Format, resize, stream	GStreamer, Aravis	Pipeline FPS, buffer health
Inference	Detection (YOLOv7/Triton)	Triton, CUDA, TensorRT	Inference time, accuracy
Orchestration	CI/CD, retrain, metadata	Kubeflow Pipelines	Workflow success, retrain logs
Monitoring	Dashboard, drift, alert	Grafana, Prometheus	Drift, FP/FN stats
Cloud Storage	Results, logs, analytics	BigQuery, Firestore, GCS	Query latency, data age