Keerthana bollepally

Real-Time Multi-Camera Intelligent Vision System

1. Objective

The objective of this system is to design a real-time computer vision solution that runs on edge devices such as NVIDIA Jetson or Raspberry Pi.

The system is designed by considering:

- limited memory and power
- real-time performance requirements
- reliability during failures

The focus is on practical system design and clear engineering reasoning rather than theoretical complexity.

2. Problem Overview

The task is to design a multi-camera intelligent vision system that can:

- detect objects in video streams
- understand different scene regions
- track and count objects over time
- analyze behavior and movement patterns
- work efficiently on edge hardware

The system should operate in real time and continue functioning even when network connectivity is unstable or unavailable.

3. Use Case: Smart Factory Safety Monitoring System

The proposed system is designed for a factory or warehouse environment using multiple cameras.
It helps to:

- detect people and safety equipment
- monitor movement within the workspace
- count people entering and exiting specific zones
- identify unsafe or unusual behavior

This use case is well suited for industrial and smart environments where low latency and reliability are critical.

4. Mandatory Capabilities

4.1 Object Detection

Purpose:

Detect objects such as:

- people
- helmets
- safety vests

Approach:

A lightweight object detection model is used because it:

- runs fast
- uses less memory
- works well on edge devices

This enables real-time safety monitoring.

4.2 Semantic Segmentation

Purpose:
Label each pixel of the image into regions such as:

- walking areas
- restricted zones
- machine areas

Benefit:
Helps understand where an object is located, not just what it is.

4.3 Video Analytics

The system performs:

- Tracking: assigns IDs to objects across frames
- Counting: counts people using virtual entry/exit lines
- Anomaly Detection: detects: overcrowding, entry into restricted zones, unusual movement

4.4 Image Stitching

Purpose:
Combine overlapping camera views into a single wide view.

Reason for choosing stitching:

- lower computation cost
- suitable for edge devices
- sufficient for indoor monitoring

This provides better spatial understanding without heavy computation.

4.5 Clustering / Pattern Discovery

Purpose:

- Discover patterns from movement data, such as:
- common paths
- crowded areas
- unusual behavior

Method:
Clustering algorithms group similar movement patterns without labeled data.

5. Hardware and Platform Constraints

- **Target Devices:**

1. NVIDIA Jetson (Nano / Xavier / Orin)
2. Raspberry Pi (with accelerator if available)

- **Constraints Considered**

1. limited RAM
2. power usage
3. heat
4. real-time processing needs

The system design respects these limitations to ensure stable performance.

6. NVIDIA Stack Usage

DeepStream

- manages multiple camera streams
- handles video pipelines
- enables real-time inference

Transfer Learning Toolkit (TLT)

- fine-tunes pre-trained models
- reduces training time and data needs
- lowers data requirements

TensorRT

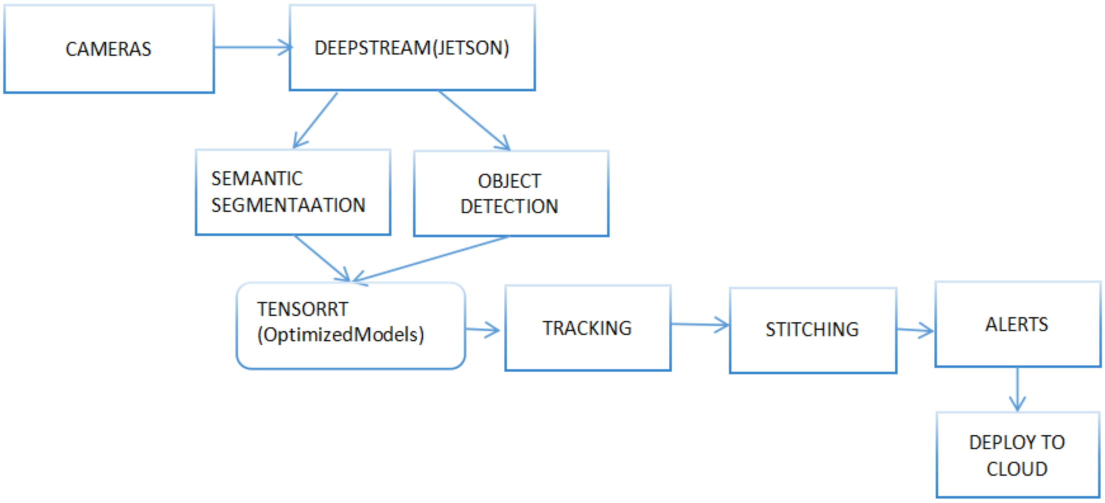
- optimizes models for faster inference
- reduces latency and improves efficiency
- improves efficiency on edge devices

7. System Architecture

Data Flow

1. Cameras capture video

2. Streams enter DeepStream pipeline
3. Detection and segmentation models run on edge
4. Tracking and analytics are applied
5. Alerts are generated locally
6. Summary data is optionally sent to the cloud



All real-time inference is performed on the edge device, while the cloud is used only for storage and analysis

8. Edge and Cloud Responsibilities

- **Edge**

1. real-time inference
2. immediate alerts
3. works without internet

- **Cloud**

1. long-term data storage
2. reports and analysis
3. model updates

9. AI Pipeline Design

- **Supervised Learning**

1. Object Detection: YOLOv8-nano (3.2M params, ~50 FPS on Jetson Nano). Pretrained on COCO, TLT fine-tuned on 1k factory images (people/helmets/vests via Roboflow). 20 epochs, batch 16, LR 0.001 on Colab GPU.
2. Semantic Segmentation: DeepLabv3-MobileNetV3. Pretrained on Cityscapes, TLT fine-tuned on factory zones (walking/restricted/machine areas).
3.

- **Unsupervised Learning**

1. Clustering: DBSCAN/K-Means on movement trajectories to discover paths, crowded areas, anomalies. No labels needed.

10. Optimization and Deployment

To ensure efficient edge deployment:

1. model size reduction (INT8 quantization)
2. lower precision where possible(FP16/INT8 precision)
3. TensorRT optimization
4. Docker-based deployment
5. remote model updates

11. Trade-Off Analysis

1. **Accuracy vs Speed:** faster models preferred for real-time use
2. **Edge vs Cloud:** edge ensures low latency, cloud supports analysis
3. **Model Size vs Power**: smaller models save power and memory

These trade-offs are required for real-world deployment.

12. Failure Scenarios

1. **Network failure:** system continues on edge
2. **Camera failure:** only affected stream stops
3. **Hardware restart:** services restart automatically
4. **Model drift:** models are updated periodically

13. Assumptions

1. cameras are correctly placed
2. edge device supports GPU acceleration
3. periodic internet access is available

14. Conclusion

This system design presents a practical and reliable approach to building a real-time multi-camera vision system for edge devices.
All required features are addressed while considering real-world limitations such as power, memory, and latency.