Arquitectura del Sistema y Estrategia de Producción

1. Arquitectura del Pipeline

El sistema se diseñó como un **pipeline secuencial optimizado**, orientado a procesamiento en tiempo casi real. Se implementaron tres módulos principales: **QC (Quality Control), Estabilidad/Vibración y Detección de Movimiento**, que trabajan en cadena sobre cada frame capturado.
La arquitectura es simple, clara y eficiente: un único loop central gestiona la lectura, el preprocesamiento y la inferencia de cada frame sin generar sobrecarga adicional.

Dado el tipo de tarea (procesamiento dependiente del siguiente frame y con operaciones ligeras), se eligió un diseño **sin multiprocessing ni threading**, ya que:

- La mayor parte del tiempo se consume en operaciones de OpenCV.
- El pipeline es estrictamente **secuencial** (las salidas de QC alimentan a movimiento y luego estabilidad).
- El overhead de manejar procesos o threads sería mayor que los beneficios.

Este diseño prioriza el **control del flujo**, la simplicidad del debugging y la replicabilidad del rendimiento.

---

2. Estrategia Teórica de MLOps y Puesta en Producción

Si este sistema debiera ejecutarse en producción en **1000 dispositivos**, la estrategia MLOps se centraría en cinco pilares:

2.1. Recolección, Almacenar y Despliegue

- El sistema se empaqueta en un contenedor Docker con dependencias controladas.
- Se distribuye como un servicio local en cada dispositivo, evitando depender de la nube para procesamiento en tiempo real.

2.2. Telemetría Ligera

Cada dispositivo enviaría métricas mínimas y agregadas:

- QC Score promedio por hora.
- Porcentaje de frames descartados por suciedad.
- Intensidad de vibración promedio (dx/dy).
- Porcentaje de tiempo con detección de movimiento.

No se envían imágenes por temas de privacidad.

2.3. Monitorización de "Data Drift" (Sucia de Cámara)

El QC Score funciona como un excelente indicador de drift.
Para 1000 dispositivos:

- Se monitorea la tendencia del QC Score en ventanas de tiempo.
- Se establecen umbrales dinámicos por dispositivo aprendidos de su historial.
- Si el QC Score cae por debajo de su rango esperado desencadena una alerta automática:
    - cámara sucia,
    - iluminación cambiante,
    - obstrucción parcial,
    - condensación/humedad,
    - desenfoque físico o vibración extrema.

A nivel central, un panel de control agrupa datos:

- porcentaje de dispositivos en “alerta de drift”,
- ranking de cámaras más deterioradas,
- tiempo promedio desde última limpieza o intervención.

---

3. Justificación de la Optimización

Para maximizar el FPS se priorizaron decisiones de diseño enfocadas en eficiencia:

3.1. Inferencia Basada en OpenCV y Procesamiento Clásico

Se eligió evitar modelos pesados de deep learning debido a:

- La tarea no lo requería (QC basado en contraste/iluminación, detección de movimiento por sustracción de fondo, vibración por diferencia de ROI).
- Permite rendimiento en hardware limitado, incluso en CPU.
- Facilita el despliegue en gran escala sin GPU.
- Minimiza el consumo energético (crítico en edge computing).

3.2. Métricas de Rendimiento

En el entorno de pruebas se alcanzaron:

- **FPS promedio:** 65–85 FPS
- **FPS mínimos:** 55 FPS bajo condiciones pesadas
- **Tiempo por frame:** ~12–15 ms
- **Procesamiento total de un video de 5106 frames:** ~70–77 segundos

---

4. Aspectos a Mejorar

A pesar de lograr el objetivo propuesto, existen aspectos que se podrían mejorar con más tiempo de ser necesario.

- **Estabilidad/Vibración:**
  El suavizado del movimiento podría mejorar incorporando un filtro temporal. Actualmente funciona bien, pero es sensible a variaciones bruscas.
- **Detección de Movimiento:**
  El método actual basado en *Background Subtraction (MOG2)* es rápido y eficiente, pero puede fallar en:
  - cambios bruscos de luz,
  - sombras,
  - vibraciones excesivas.

  Probablemente utilizando un modelo de reconocimiento de objetos estos problemas disminuirían, pero podría comprometerse el rendimiento de los resultados.

Estas mejoras no eran necesarias para cumplir el objetivo principal, pero abren opciones y posibles mejoras para lograr una versión más robusta en producción.