State of the Art: peligros y pizarra del aula inteligente

Ingeniería de Mantenimiento de Computadores y Redes

Propuesta por Ivan

19 de febrero de 2026

Índice

Índice

Cambio de paradigma

Objetivo

Complementar sistemas reactivos basados en *hardware* dedicado con sistemas proactivos basados en **visión artificial** (en adelante, CV por *Computer Vision*).

- ***Hardware* tradicional:** Sensores unifuncionales (humo, humedad, PIR). Entradas activas: botón, palanca, lógica con enfoque más «síncrono».
- **Enfoque *software*/IA:** Uso de cámaras CCTV (cámaras de circuito cerrado, usadas para monitoreo de vídeo de seguridad) existentes + procesamiento en el borde (*Edge Computing*) o «nube».

Índice

¿Qué riesgos hay en un aula?

- **Seguridad:**
  - Violencia física (peleas, acoso).
  - Presencia de armas o bultos de objetos contundentes.
  - Intrusos en horarios restringidos.
- **Integridad del entorno:**
  - Incendios (análisis espectral y de movimiento).
  - Inundaciones (análisis de textura en suelos).
  - Bloqueo de salidas de emergencia (análisis espacial).
  - Terremotos y actividad sísmica (temblores).
  - Desprendimientos de objetos así como lanzamiento.
  - Vandalismo (pintadas, comportamiento canalla...).
- **Salud y bienestar:**
  - Caídas y desmayos.
  - Detección de pánico o aglomeraciones.
  - Contaminación acústica (ruido estridente, chillidos...).

Índice

YOLO (*You Only Look Once*)

- **Rápido y preciso**.
- Sensibilidad a oclusiones.
- **Navaja suiza**: Detección, segmentación, seguimiento, poses...
- Código abierto (OSS, *Open Source Software*).
  `https://github.com/ultralytics/ultralytics`.
- Usado también para **atención de alumnos**:
  `https://www.mdpi.com/1424-8220/25/22/6972`
- En **tiempo real** a través de cámaras IP a servidor local.
- Usado para detectar robo, violencia, vandalismo, fuego...
- Plausible en *hardware* limitado si se relajan restricciones.

Google MediaPipe

- Rápida sturación con multitud.
- Eficiente al poder ejecutarse en **móviles** (pensado para sin GPU dedicada).
- Mejor captura de la profundidad.
- OSS.
- **Navaja suiza**.
- ¿Optimizado para Rasberry Pi? («*¿Gemini hablando bien de Google?*»).
- `https://github.com/google-ai-edge/mediapipe`.

CNN (*Convolutional Neural Networks*)

- **Extracción de características:** Aprende automáticamente a detectar bordes, texturas y formas sin programación manual.
- **Jerarquía Visual:**

$$\text{Bordes} \rightarrow \text{Formas simples} \rightarrow \text{Objetos complejos}$$

- **Invarianza:** Reconoce el objeto independientemente de su posición (x, y) en la imagen.
- El ***kernel*** (filtro) recorre la matriz de píxeles calculando el producto escalar para detectar patrones.
- Sirve para audio también.

Violencia

- La violencia es un **acto temporal** → secuencia memoria.
- **YOLO** por si solo le cuesta → YOLO-Pose. Precisamos de una TPU si hacer en el *edge*.
- **YOLO-Pose/Google MediaPipe+LSTM** (*Long Short Term Memory*).
  - Clasificador temporal.
  - *Hardware* modesto → **privacidad**.
  - **Oclusiones y OSS**.
- **SlowFast**
  - Preciso. De Facebook. OSS.
  - Computación pesada.
  - Dos modos de funcionamiento: *slow* (¿quién es?), *fast* (movimiento)
- Ejemplo detección de violencia y vandalismo
  `https://github.com/Ab-code00/SurakshaAI---Real-Time-Suspicious-Activity-Detection-Syst`

Detección de intrusos

- **Objetivo**: saltar alarma si viene alguien no deseado en determinado horario.
- Cuádruple reto: **reconocimiento facial** + **horarios** + **lista blanca** + **privacidad**.
- Variantes:
  1. Naive: hay personas $\wedge$ fuera de horario $\rightarrow$ alarma.
  2. Reconocer caras $\rightarrow$ ¿Privacidad? ¿Niños? ¿Ejercer derecho?. Necesidad de una DB actualizada $\rightarrow$ lío legal.
  3. ¿Quién decide entrar y hasta dónde puede? (Absurdo no dejar entrar visita, familiares, servicios públicos, nuevo personal $\notin$ DB) $\rightarrow$ ¿1984?
- Merodeos: `https://github.com/nwojke/deep_sort`.
- `https://github.com/hectorpadin1/Network-Intrusion-Detection-System`.
- Otro enfoque, analizar la red: `https://yardenfalik.github.io/IDS-Project/`. (Esto sale en la teoría de Sistemas Distribuidos).

Incendios e inundaciones

Fuego

- **no tiene forma fija, «parpadea», brilla, reflja**.
- Puede producirse en **sitios ocultos** donde la cámara no vea.
- Versiones *custom* de YOLO.

$\Longrightarrow$ Detección por IA como complementario para anticipar.

Agua

- **No se busca encontrar líquido** sino otras características.
- **Reflejos** (da igual si es sucia o transparente).
- Detección del suelo mojado píxel a píxel (U-Net, DeepLabV3+).
- Causas naturales $\rightarrow$ otros asuntos, meteo...

$\Longrightarrow$ Interés en prevenir (tuberías rotas, fraturas...). Conocer aula concreta.

Caídas y desmayos

- Heurísticas sobre la pose del esqueleto.
  1. Cambio en la **relación de aspecto**.
  2. **Velocidad** de descenso (¿se sienta o se cae?).
  3. **Inactividad** (¿tropiezo o desmayo?).
- Otras situaciones.
  - Deporte $\rightarrow$ *fine tunning*, más movimiento.
  - *Trolling* $\rightarrow$ al menos, 18 añitos tiene la criatura.
- ¿Oclusión? Inferencia en base a lo visible.
- Tecologías a gastar: YOLO, YOLO-pose, BoT-SORT y ByteTrack.
- **OpenPifPaf**: `https://github.com/cwlroda/falldetection_openpifpaf`.
- **AlphaPose**: `https://github.com/GajuuzZ/Human-Falling-Detect-Tracks`.
- **Multicam**: `https://github.com/taufeeque9/HumanFallDetection`.

Bloqueo de salidas

Detectar objetos (mochilas, mesas) que permanecen en vías de evacuación o que impidan una salida normal. **Diferenciar un bloqueo real** de un tránsito momentáneo.

Algoritmo: IoU (*Intersection over Union*) Temporal

Sea P_{salida} el polígono de la salida, B_{bulto} la caja del objeto y T el tiempo transcurrido desde que se detecta la obstrucción:

$$\text{Solape} = \frac{\text{Área}(P_{salida} \cap B_{bulto})}{\text{Área}(P_{salida})}$$

$$(\text{Solape} > 0{,}3) \wedge (T > 60s) \rightarrow \text{Alerta}$$

Audio: gritos, auxilio, ruido estridente...

- **Clasificador de sonido de Tensorflow** YAMNet `https://www.tensorflow.org/hub/tutorials/yamnet?hl=es-419`.
- Micrófono I2S: `https://opencircuit.es/producto/fermion-i2s-mems-microphone-breakout`.

Procesamiento (YAMNet):

1. **Entrada:** Micrófono I2S (MEMS) en la Raspberry Pi.
2. **Preprocesar:** Transformada de Fourier $\rightarrow$ Mel-Spectrogram (Redes de los Computadores).
3. **Clasificación:** CNN ligera (MobileNet) entrenada en *AudioSet*.

Eventos Críticos

- **Gritos:** Agresión o Pánico.
- **Destrucción:** Vandalismo.
- **Palabras clave:** «¡Socorro!», «¡Ayuda!» (TinyML).

Ejemplo: `https://github.com/Varun-310/SCREAM`

Privacidad garantizada. Almacenar ¿qué eventos peligrosos (*Edge*).

No todo se detecta mejor con IA

- **Fuegos, inundaciones**.
- **Sismos** (ESP32 + MPU-6050): `https://github.com/serdaraltin/earthquake-warning-system`.
- **SmartUniversity** IoT de la UA que captura diferentes parámetros de afluencia y temporal.

Demostraciones

- **Fuego**: https://github.com/MuhammadMoinFaisal/FireDetectionYOLOv8.
- **Armas**: https://www.nature.com/articles/s41598-025-07782-0, https://github.com/swatified/Weapon-Detection-System.
- **Violencia**: https://www.youtube.com/watch?v=a2xWqkFDYuU, https://www.youtube.com/watch?v=D4mjEBgAXPU, https://www.youtube.com/watch?v=qeFrjFa5Rxc
- **Caídas**: https://www.youtube.com/watch?v=vEtsmg7-fWs
- **Ataque de pánico, aglomeraciones**: https://github.com/moego0/panic-attack-detector, https://github.com/jinay-k-jain/Panic_detector_CCTV.
- **Vandalismo**: https://ieeexplore.ieee.org/document/10895139.

Índice

¿Qué tenemos a disposición?

- 600€ de presupuesto en nuevo *hardware*.
- Electrónica de otros años.
  - Placas de Arduino.
  - Rasberry Pi.
  - ESP32.
- Ordenadores del aula.
  - GPU: Nvidia GTX 1660.

¡Relajar restricciones!

1. Una cámara $\rightarrow$ un propósito.
2. Reducir FPS $\rightarrow$ no tiempo real.
3. Procesamiento centralizado (SPOF). ¿vigilancia urbanización?
4. Combiando estrategias (o aplicando divide y vencerás para el reparto de responsabilidad): *edge* detecta que hay personas, *server* averigua quienes son.

Ejecución en *hardware* limitado

No todo el *hardware* puede procesar las millones de operaciones matriciales de una CNN. Clasificación por niveles:

Dispositivo	Capacidad CNN	Tecnología	Uso
Raspberry Pi 5	**Alta**	Python, TensorFlow, PyTorch	Corre YOLO/MediaPipe para análisis complejo.
ESP32-S3	**Limitada (TinyML)**	TFLite Micro, C++	Sensor auxiliar. Solo clasificación simple (Person/No-Person).
Arduino Uno	**Nula**	-	Solo control de relés/luces. No procesa imagen.
Arduino Portenta	**Media**	OpenMV, Edge Impulse	Similar al ESP32 pero grado industrial.

Es importante la **cuantización.** Para correr en ESP32, los modelos se comprimen de `float32` a `int8`, reduciendo el tamaño del modelo de 50MB a $<$ 500KB.

¡Optimizar!

Índice

Ideas para apuntar

La pizarra es un recurso muy utilizado por el profesorado para ilustrar ideas, resolución de ejercicios y conceptos. La información de la pizarra al cabo de una sesión no está disponible. Lo cual dificulta el seguimiento de la clase para aquellos más despistados, sentados más lejos, lentitud, mala visibilidad o cualquier otra dificultad en tanto que interfiera...

1. Facilitar la **grabación y *streaming*** de clases.
2. **Seguimiento de movimiento** del profesor para el punto anterior.
3. Guardar un **historial** del trazado de la pizarra.
4. **Transcripciones de voz** de la lección.
5. **Accesibilidad** (intérpretes, traductores...).

Experiencias previas

1. Durante la pandemia del **covid**:
   - Las reuniones a través de videollamadas.
   - Poca preparación, mal audio, aplicaciones propietarias (no FOSS).
   - Compatir pantalla, pizarra con mala visibiidad.
   - Profesores pueden emplear otras para difundir conocimiento: `https://youtu.be/_KNxz2YyhgA`.
2. El seguimiento del movimiento del profesor puede hacerse de **forma automática** desde la configuración de algunas cámaras.
   - `https://youtu.be/ZA-tUyM_y7s&t=120`.
   - Aunque siempre se deba contemplar una opción manual.
3. **Guardar algo escrito de la pizarra**.
   - Fotos a la pizarra $\rightarrow$ mala calidad al detalle $\wedge$ rostros no deseados.
   - Materiales docentes, almacenamiento en la «nube»...
4. A partir de los vídeos **es posible extraer una transcripción**.
5. Garantizar **accesibilidad** para personas ciegas (braille), sordas de lo acontecido en la sección (generar intérpretes).

Índice

Consideraciones

El sistema debe:

- Estar **a merced del profesor** como una herramienta. Y es este quién decida cuando hacerla servir.
- Garantizar la **privacidad** (GDPR). Tratamiendo de datos en *edge* o en local.
- Fácil mecanismo para **cambiar parámetros**: $ON \leftrightarrow OFF$, historial sí $\leftrightarrow$ no, movimiento sí $\leftrightarrow$ no...
- **Independiente** al tipo de pizarra (da igual si es de tiza o digital).
- Tener la posibilidad de guardar (**exportación**) el vídeo/traza de la pizarra en fichero.
- ¿Tareas paralelas? (Grabar y sacar grabados de la pizarra).

Emisiones en vivo. Grabaciones.

No es de extrañar que la mayoría de los profesores hagan servir el ordenador de clase para presentar transparencias. Donde pueden utilizarlo para configurar el directo y guardados.

Para una mayor permitir mayor flexibilidad puede:

- Tener un **mando IR** o **algo físico** que indique qué hacer a la cámara. (opción con limitaciones)
- **Aplicación web** en la que el profesor decida los parámetros.

También sería interesante conocer el horario del aula para notificar el cese del contenido y el uso que se le dé a la cámara.

Véase programas como OBS o VLC para la gestión multimedia.

Índice

¿Cuándo capturar trazas?

El sistema utiliza una cámara cenital o frontal para digitalizar la pizarra analógica automáticamente. Dos formas de hacerlo:

Disparador

La captura se activa mediante un **botón físico** solicitado por el profesor al terminar una explicación.

Detección automática

Mediante ***background subtraction***, el *software* detecta cuando la pizarra está «llena» y el profesor se aparta, disparando la foto automáticamente.

- Demo de *background substraction* con OpenCV: `https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html`.
- **Cámara PTZ con *auto-tracking***: `https://github.com/AutoPTZ/autoptz`

Privacidad y procesamiento

Proceso de anonimización:

1. **Detección de rostros:** Uso de modelos ligeros (ej. MTCNN o Haar Cascades) para localizar caras en la imagen de la pizarra.
2. **Aplicación de difuminado:** Gaussiana sobre las regiones de interés (ROIs) detectadas.
3. **Rectificación de la perspectiva:** Algoritmo de homografía para transformar la foto lateral en una imagen plana escaneada.
4. **Mejora de contraste:** Binarización adaptativa para convertir trazos de tiza/rotulador en digital nítido.
5. **Extracción en texto:** Bien uitilizar OCR para sacar texto de imagen o Whisper (OpenAI) para el audio $\rightarrow$ transcripción y subtítulos.
6. **Exportación a múltiples formatos:** Bien entre documentos (pandoc) o entre vídeo (ffmpeg).

Enlaces de interés

- **Pandoc:** `https://pandoc.org/`.
- **ffmpeg:** `https://ffmpeg.org/`.
- **Whisper:** `https://github.com/openai/whisper`.
- **Tesseract OCR:** `https://github.com/tesseract-ocr/tesseract`.
- **MTCNN:** `https://github.com/ipazc/mtcnn`.