OccultaShield: The Production-Ready AI System for GDPR-Compliant Video Privacy

Gabriel Juan Betés

3 de febrer de 2026

Resum– Amb l'augment de l'ús de drons o UAV (Unmanned Aerial Vehicles) per a la captació d'imatges aèries i, en molts casos, la seva posterior publicació, sorgeix un conflicte directe amb els drets de privacitat establerts pel GDPR (General Data Protection Regulation). Davant d'aquest context, Occultashield presenta un sistema d'Intel·ligència Artificial dissenyat per automatitzar el compliment normatiu i alinear-se de manera eficient amb la nova legislació europea sobre la intel·ligència artificial, l'AI Act. Gràcies a una arquitectura basada en tres mòduls, Occultashield ofereix una solució integral: en primer lloc, un mòdul de detecció precisa de dades personals; en segon lloc, un sistema de verificació de les deteccions mitjançant models d'IA i un sistema GraphRAG (Graph Retrieval Augmented Generation), que contextualitza les possibles vulneracions des d'un punt de vista legal i genera posteriorment un qüestionari per a la supervisió humana; i, finalment, un conjunt d'algoritmes de visió per computador que permeten l'aplicació d'anonimització selectiva.
La contribució principal d'aquest projecte rau en la integració d'un motor intel·ligent de compliment legal, que dona lloc a una solució de protecció de la privacitat semiautomatitzada, jurídicament defensable i orientada a minimitzar els riscos legals associats a l'ús de drons per part dels operadors.

Paraules clau– RGPD, AI Act, Visió per computador, Drons, Anonimització, GraphRAG, Ética de l'IA, Knowledge Graphs

Abstract– The increasing use of drones or Unmanned Aerial Vehicles (UAVs) for capturing aerial imagery, and often its subsequent publication, creates a direct conflict with the privacy rights established by the General Data Protection Regulation (GDPR). In this context, Occultashield presents an Artificial Intelligence system designed to automate regulatory compliance and efficiently align with the new European legislation on artificial intelligence, the AI Act. Thanks to an architecture based on three modules, Occultashield offers a comprehensive solution: first, a module for the precise detection of personal data; second, a verification system utilizing AI models and a Graph Retrieval Augmented Generation (GraphRAG) system, which contextualizes potential violations from a legal perspective and subsequently generates a questionnaire for human supervision; and finally, a set of computer vision algorithms that enable selective anonymization. The main contribution of this project lies in the integration of an intelligent legal compliance engine, resulting in a semi-automated, legally defensible privacy protection solution aimed at minimizing the legal risks associated with drone operations.

Keywords– GDPR, AI Act, Computer vision, UAV, Anonymization, GraphRAG, AI Ethics, Knowledge Graphs

1 INTRODUCCIÓ

L'ús de Vehicles Aeris No Tripulats (UAVs) ha transcendit l'àmbit recreatiu per convertir-se en una eina indispensable en sectors estratègics. No obstant això, aquesta democratització de la captura aèria ha transformat els drons en vectors d'ingesta massiva de dades. A diferència dels sistemes estàtics, els UAVs operen de mane-

• E-mail de contacte: gabriel.juan@uab.cat
• Menció: Computació
• Treball tutoritzat per: Lluis Echeverria Rovira (Departament de Ciències de la Computació)
• Curs 2025/26

ra dinàmica, cosa que fa que el seguiment d'individus sigui tècnicament complex. Tot i que existeixen algoritmes robustos per al seguiment d'objectes en temps real, com els basats en mètriques d'associació profunda [4], aquests sistemes es limiten a identificar "on"és la persona, sense considerar "si"hauria de ser visible legalment.

Aquest escenari entra en conflicte directe amb el marc legislatiu europeu. El **Reglament General de Protecció de Dades (RGPD)** protegeix les *"categories especials de dades"* (Article 9), incloent-hi la biometria [1]. La recent **Llei d'Intel·ligència Artificial (AI Act)** reforça aquesta restricció, classificant la identificació biomètrica remota com a pràctica d'"Alt Risc"[2], obligant els operadors a aplicar mesures de *Privacy-by-Design*.

Actualment, la indústria s'enfronta a una dicotomia. Les tècniques d'anonimització automàtica solen ser destructives (desenfocament global), mentre que l'edició manual és inviable per al flux de dades massiu que generen els sistemes de seguiment actuals [3].

OccultaShield[1] proposa una arquitectura *On-Premise* que uneix aquests dos mons: utilitza la precisió de la Visió per Computador (per detectar i seguir entitats) i la integra amb un motor Neuro-Simbòlic (*GraphRAG*[18]) que aplica la normativa legal. L'objectiu és garantir que la tecnologia de seguiment [4] serveixi per protegir la privacitat, i no per vulnerar-la.

2 Estat de l'Art i Anàlisi de Tecnologies

Per a un pilot de drons, complir amb les lleis de privadesa europees pot semblar una missió impossible. Com es pot capturar l'increïble potencial visual d'un vol sense infringir la privadesa de les persones a terra? Per resoldre aquest trencaclosques, primer s'han d'entendre les peces amb què es compta.

2.1 Ensenyant a una màquina a veure: Els ulls del sistema

El primer pas és trobar les dades personals. Avui dia, el cervell d'aquests sistemes són xarxes neuronals profundes. El campió en velocitat i agilitat és sovint considerat **YOLO (You Only Look Once)** [11]. La seva característica principal és que analitza la imatge d'un sol cop d'ull, identificant objectes ràpidament. Versions modernes com **YOLOv8** [12] ho fan amb una precisió sorprenent.

No obstant això, YOLO no és l'única solució. L'estat de l'art inclou diverses famílies de models:

- **Família YOLO**: Coneguda per la seva optimització per a la velocitat en temps real, sent una opció habitual per al processament de vídeo en directe.
- **Família Detection Transformers (DETR/RT-DETR)**: Empren una arquitectura de *transformers* que elimina components manuals com la supressió de no-màxims (NMS), assolint una notable precisió.
- **Família Faster R-CNN**: Representa una arquitectura de dues etapes (proposta de regions i classificació). Tot i ser generalment molt precisa, tendeix a ser més lenta que YOLO.
- **Família EfficientNet**: Tot i ser dissenyada per a classificació, les seves *backbones* són la base de detectors (com EfficientDet) que busquen un equilibri eficient entre precisió i cost computacional.

La selecció del model adequat per a OccultaShield dependrà d'un equilibri entre la precisió de detecció (per no ometre cap dada personal) i la velocitat de processament.

2.2 La decisió més important: Realment estas vulnerant el GDPR?

Per evitar falsos positius i contextualitzar legalment les deteccions, OccultaShield proposa un mòdul de verificació. Aquest mòdul utilitza un Model Petit de Llenguatge (SLM) multimodal integrat en una arquitectura GraphRAG. Es requereix que l'SLM analitzi el context visual del fotograma associat a la possible infracció i, amb l'ajuda del graf de coneixement legal, verificar la detecció. Alguns models candidats per a aquesta tasca, disponibles a plataformes com Hugging Face [10], són:

- **Model Deepseek vl2**[9]: Destaca per la seva arquitectura Mixture-of-Experts que optimitza l'ús de recursos activant pocs paràmetres per inferència, ideal per a l'execució local. És especialment potent en Visual Grounding i OCR, capacitats clau per verificar matrícules i documents textuals.
- **Model Gemma 3n**: Dissenyat específicament per a entorns Edge, aquest model ofereix capacitats multimodals natives amb un consum de memòria mínim gràcies a la seva arquitectura MatFormer. Això el fa perfecte per garantir la privacitat "on-device", processant imatges sense enviar dades al núvol.
- **Model Qwen 2.5/3 VL**: Aquest model sobresurt per la seva capacitat de retornar coordenades precises (bounding boxes) i sortides estructurades en format JSON. Això permet una doble verificació geomètrica de les deteccions i facilita enormement la integració amb la lògica de programació del sistema.

2.3 L'art d'ocultar: La màscara digital i el repte del moviment

Un cop els "ulls" del sistema han localitzat un rostre o una matrícula, és necessari aplicar una màscara. Hi ha diverses tècniques [10]:

- Un **desenfocament (*blurring*)**:Una opció subtil que preserva l'estètica general del vídeo.
- Un **pixelat**: Més directe i funcional, indica clarament que la informació està protegida.
- Una **màscara negra**: L'opció més segura i invasiva, eliminant completament la informació visual.

[1] Codi font, documentació i configuració disponibles al repositori oficial: `https://github.com/GabrielJuan349/OccultaShield`

Però, què passa quan la gent es mou? Posar una màscara en una foto és fàcil; fer que aquesta màscara persegueixi una persona per un vídeo és el veritable desafiament. Aquí és on els **algoritmes de seguiment (*Object Tracking*)**, com **DeepSORT** [4], entren en acció. Aquests sistemes "fixen" la detecció inicial i en segueixen el rastre de manera intel·ligent, assegurant que la màscara es mantingui al seu lloc. A més d'aquestes tècniques clàssiques, la investigació recent explora mètodes més avançats com l'**Anonimització Natural Profunda** (que intenta reemplaçar rostres per altres generats artificialment) o models d'edició d'imatges com **Qwen Image Edit** [8], que podrien oferir anonimitzacions més contextuals.

2.4 El veïnat: Qui més està resolent aquest problema?

En buscar solucions ja existents, es troba un ecosistema molt variat, gairebé com un barri amb diferents tipus de botigues:

- **Les Grans Superfícies (Software Comercial)**: Empreses com **Celantur** [6] ofereixen solucions potentíssimes, com a servei industrial. Són eficaços, però sovint requereixen subscripcins costoses i, el més important, et demanen que pugis els teus vídeos, sovint amb informació sensible, als seus servidors al núvol. És la solució "caixa negra": funciona, però perds el control.

- **La Botiga d'Artesania (Editors de Vídeo)**: Eines com **Adobe Premiere Pro** [7] et donen un control absolut. La precisió és alta, però el procés és extremadament lent i inviable per a grans volums de metratge.

- **El Garatge de l'Inventor (Projectes Open-Source)**: En llocs com GitHub, trobaràs innombrables projectes d'entusiastes que han construït les seves pròpies eines. Són transparents i gratuïts. No obstant això, sovint són projectes incomplets, no llestos per a un entorn de producció.

2.5 Trobant el punt exacte: On encaixa OccultaShield

I és precisament enmig d'aquest barri on OccultaShield troba el seu lloc. No intenta ser la solució industrial més gran, ni l'eina més manual, ni un experiment de garatge. El seu objectiu és trobar el **balanç perfecte** pel professional:

OccultaShield agafa la potència dels projectes *open-source*, però l'empaqueta en un **producte complet i fàcil d'usar**. A diferència de les solucions comercials, s'executa al teu propi ordinador, així que **les teves dades mai no surten del teu control**. I, sobretot, a diferència de les "caixes negres", et retorna el poder a tu, el pilot. El seu mòdul de validació no és només una funció; és la filosofia central del projecte: la IA t'ajuda, et fa la feina pesada, però **la decisió final i la responsabilitat legal sempre són teves**.

En resum, OccultaShield no és només una altra eina; és l'eina que faltava.

3 OBJECTIUS

L'objectiu principal d'aquest projecte és el desenvolupament d'un Producte Mínim Viable (MVP), implementat en Python, consistent en un sistema d'Intel·ligència Artificial per a la preservació de la privacitat en vídeos de drons. L'abast d'aquest prototip se centra específicament en la detecció de persones i matrícules. Si el temps ho permet, s'explorarà l'optimització del sistema migrant components crítics al llenguatge Rust per millorar-ne el rendiment.

Els objectius específics són els següents:

- **Desenvolupar i optimitzar un mòdul de detecció d'objectes:** Mitjançant el *fine-tuning* d'un model (com YOLOv8 o similar) per identificar amb precisió dades personals (rostres, matrícules) en imatges aèries, minimitzant els falsos positius.

- **Construir un Graf de Coneixement (Knowledge Graph) especialitzat en el RGPD/GDPR:** Aquest graf ha de modelar les normatives clau del RGPD/GDPR [1] i l'AI Act [2], traduint el llenguatge legal en relacions estructurades i consultables per la màquina.

- **Dissenyar i implementar un mòdul de verificació intel·ligent (GraphRAG):** S'integrarà un Model Petit de Llenguatge (SLM) amb el Graf de Coneixement. L'objectiu és que el sistema pugui contextualitzar legalment les deteccions i generar un qüestionari de validació per a la supervisió humana.

- **Implementar un mòdul d'anonimització selectiva i irreversible:**S'utilitzaran algoritmes de Visió per Computador (com blurring o pixelat) [10] que apliquin les edicions només als segments de vídeo validats per l'usuari, assegurant que el procés no sigui reversible.

- **Integrar els mòduls en un sistema MVP:** Desenvolupar un prototip funcional en Python que orquestri els mòduls de detecció, verificació i anonimització, incloent una interfície completa fent servir el framework de Angular per a la interacció de l'usuari.

4 METODOLOGIA I PLANIFICACIÓ

Per a l'execució del projecte s'ha adoptat la metodologia **Scrumban** [5], un enfocament híbrid que combina l'estructura de *sprints* curts (1.5 a 2 setmanes) de Scrum amb la visualització del flux de treball i els límits WIP (*Work in Progress*) de Kanban. Aquesta estratègia àgil permet una adaptació flexible davant reptes tècnics emergents, com la potencial migració de components a Rust, tal com s'il·lustra a la Figura 1.

La planificació temporal (Figura 2) s'ha estructurat en cinc fases consecutives per garantir l'entrega de l'MVP en 21 setmanes:

1. **Investigació (Setmanes 1-3):** Anàlisi de l'Estat de l'Art (YOLO, SLMs, Kornia) i consolidació dels requisits legals (GDPR/AI Act).

2. **Arquitectura (Setmana 4):** Disseny modular del sistema, definició de l'API REST i esquemes de bases de dades (SurrealDB).

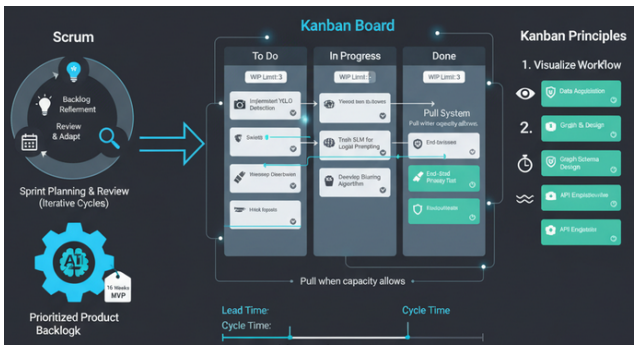

Fig. 1: Esquema de la metodologia Scrumban aplicada al projecte [5]

3. **Desenvolupament MVP (Setmanes 5-12):** Implementació iterativa del pipeline: Detecció, Verificació (GraphRAG), Anonimització i Frontend.

4. **Validació (Setmanes 13-16):** Proves d'integració, optimització de rendiment (GPU) i validació de resultats.

5. **Tancament (Setmanes 17-21):** Redacció de la memòria tècnica i preparació de la defensa.

FASES I SUBFASES	INICI	FINAL
1. Investigació Inicial	Setmana 1	Setmana 3
1.1 Projectes similars a OccultaShield	Setmana 1	Setmana 1
1.2 Investigació de les parts de OccultaShield per separat	Setmana 2	Setmana 2
1.3 Investigació sobre el RGPD	Setmana 1	Setmana 1
2. Inici del Projecte	Setmana 2	Setmana 4
2.1 Investigació de models per a les diferents parts	Setmana 2	Setmana 3
2.1.2 Models de detecció d'objectes	Setmana 2	Setmana 2
2.1.3 Models de Verificació (SLMs)	Setmana 3	Setmana 3
2.1.4 Models/Algoritmes de anonimització	Setmana 3	Setmana 3
2.2 Creació de l'Arquitectura del projecte	Setmana 4	Setmana 4
2.2.1 Disseny intern del servidor	Setmana 4	Setmana 4
2.2.2 Diseny d'interacció amb l'usuari	Setmana 4	Setmana 4
2.3 Investigació dels elements necessaris pel projecte	Setmana 3	Setmana 3
3. Inici del desenvolupament del projecte	Setmana 5	Setmana 12
3.1 Desenvolupament de la part de detecció a vídeos	Setmana 5	Setmana 6
3.2 Desenvolupament del GraphRAG (Versió MVP)	Setmana 11	Setmana 12
3.2.1 Introducció de les dades al graf de coneixements (Simulat)	Setmana 11	Setmana 11
3.2.2 Connexió del SLM al graf de coneixements (Simulat amb lògica de regles)	Setmana 12	Setmana 12
3.3 Desenvolupament dels algoritmes de anonimització	Setmana 7	Setmana 7
3.4 Desenvolupament del frontend del projecte	Setmana 8	Setmana 10
4. Testing del sistema OccultaShield	Setmana 13	Setmana 16
5. Redacció de la memoria Final	Setmana 17	Setmana 19
6. Presentació	Setmana 19	Setmana 21

Fig. 2: Planificació temporal detallada (Fases i Subfases).

5 Arquitectura del Projecte

Per a l'Arquitectura d'OccultaShield s'ha optat per una arquitectura modular a la part del servidor i una petita pàgina web que mostraria les diferents parts que necessiten interacció del client, com es veu a la figura 3.

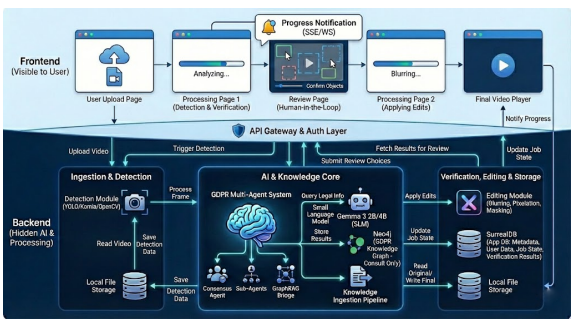

Fig. 3: Arquitectura del projecte

A continuació, es detalla l'arquitectura tècnica del sistema, desglossant cadascun dels mòduls funcionals i l'esquema de persistència de dades:

- **Mòdul de Detecció i Segmentació:** Aquest mòdul implementa una estratègia de percepció híbrida multiclasse. Utilitza **YOLOv11-seg** per a la segmentació d'instàncies de persones i la detecció de matrícules de vehicles, generant màscares binàries que aïllen l'objecte del fons. Paral·lelament, integra **Kornia YuNet**, un detector facial especialitzat que opera sobre tensors, per garantir la captura de rostres amb alta precisió. Simultàniament, l'algoritme de seguiment (*Tracker*) registra a la base de dades els intervals temporals i les coordenades de cada entitat sensible.

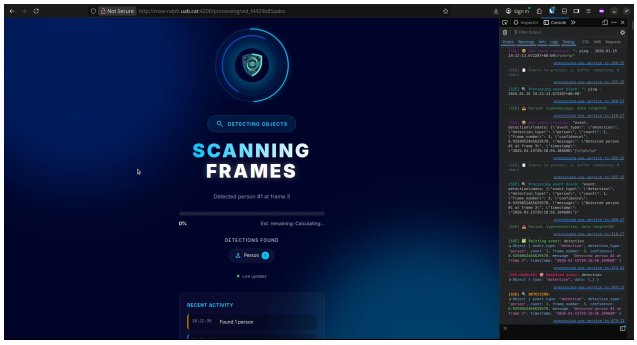

Fig. 4: Evidència de l'execució del pipeline de detecció. S'observa la segmentació d'instància multiclasse on el sistema no només localitza el subjecte, sinó que genera una màscara de píxels precisa mitjançant YOLOv11-seg. Aquesta dada es processa com un tensor a la VRAM per ser enviada directament al mòdul d'anonimització, evitant la sobrecarga del bus PCIe.

- **Mòdul de Verificació Neuro-Simbòlica:** Implementa l'arquitectura **GraphRAG** avançada. Utilitza el model **Gemma 3n** per a l'anàlisi semàntica de l'escena, connectat a un Graf de Coneixement (**Neo4j**) que conté la taxonomia completa del RGPD (incloent-hi els articles 5, 6, 9 i els considerands rellevants). Aquest sistema creua les deteccions visuals amb la normativa vigent per determinar la licitud del tractament i generar el qüestionari de supervisió.

- **Mòdul d'Anonimització Híbrid (GPU/CPU):** Aquest mòdul disposa d'una arquitectura de doble via per garantir la disponibilitat. En entorns amb maquinari compatible, prioritza l'ús de **Kornia** [13] per aplicar filtres tensorials a la VRAM (*Zero-Copy*). En cas d'indisponibilitat de la GPU, el sistema realitza un *fallback* automàtic a **OpenCV** [10], executant l'anonimització (blurring/pixelat) a la CPU per assegurar que el vídeo es processa correctament sota qualsevol circumstància.

6 Desenvolupament i Implementació

Aquesta secció detalla la construcció dels components de programari, posant èmfasi en l'arquitectura modular del sistema, la gestió eficient de dades i l'orquestració del pipeline d'IA. La implementació segueix principis d'enginyeria de

sistemes per garantir un desplegament reproduïble i escalable.

6.1 Aprovisionament i Inicialització Automatitzada

Per eliminar la fricció en el desplegament i garantir la paritat entre entorns de desenvolupament i producció, s'ha automatitzat completament la inicialització del sistema. S'ha desenvolupat un script mestre d'orquestració ('instalation.sh') que actua com a punt d'entrada únic. Aquest script realitza una auditoria prèvia del host (comprovant la presència de drivers NVIDIA, *Docker Engine* i el runtime *Bun*) abans de construir els contenidors i inicialitzar la base de dades.

Nota: La documentació tècnica exhaustiva sobre la configuració de l'entorn, les versions de les llibreries i el manual d'operació pas a pas es troba detallada a l'**Apèndix A**. Aquesta secció se centra exclusivament en l'arquitectura del programari.

6.2 Arquitectura de Seguretat i Gestió d'Usuaris

La protecció de l'accés a les dades sensibles és prioritària. S'ha implementat un sistema d'autenticació robust basat en la llibreria **Better-Auth**, integrada directament amb la base de dades **SurrealDB** per a la persistència de sessions.

- **Gestió de Rols (RBAC):** El sistema distingeix entre usuaris estàndard i administradors. El mòdul 'server/lib/init-admin.ts' assegura l'existència d'un super-administrador a l'arrencada, mentre que 'admin.ts' gestiona els permisos elevats per a l'auditoria de logs.
- **Protecció en Profunditat:** La seguretat s'aplica a dues capes:
  - *Backend:* El middleware 'auth_middleware.py' verifica la signatura criptogràfica de la sessió a cada petició a l'API.
  - *Frontend:* Els *Guards* d'Angular ('auth.guard.ts') bloquegen la navegació a rutes protegides ('/upload', '/review') si l'estat d'autenticació no és vàlid.
- **Auditoria (Audit Log):** Seguint l'esquema definit a 'schema.surql', cada acció crítica (pujada de vídeo, validació de vulneració) genera un registre immutable a la taula 'audit-log', garantint la traçabilitat exigida pel RGPD.

6.3 Enginyeria del Backend i Flux de Dades

El backend (Python 3.13 + FastAPI) no actua només com a passarel·la per a la IA, sinó com un orquestrador de dades complex.

6.3.1 Patrons de Disseny i Configuració

Per assegurar la mantenibilitat, s'ha evitat l'ús de valors "hardcoded". El mòdul 'config_loader.py' utilitza *Pydantic* per validar estrictament les variables d'entorn ('.env') a l'inici. A més, s'utilitza el patró d'**Injecció de Dependències** ('dependencies.py') per gestionar el cicle de vida de les connexions a base de dades, evitant fuites de memòria en entorns d'alta concurrència.

6.3.2 Pipeline ETL del Coneixement Legal

La base de coneixement legal d'OccultaShield no està "hard-coded" en el codi, sinó que es construeix dinàmicament mitjançant un pipeline ETL (*Extract, Transform, Load*). Això garanteix que el sistema pugui actualitzar-se davant canvis normatius sense necessitat de recompilar el nucli d'IA.

El procés d'ingesta s'orquestra mitjançant l'script `enhanced_ingest_gdpr.py`, el qual realitza les següents tasques seqüencials:

1. **Extracció (Extract):** Descàrrega i lectura dels corpus legals des de fonts estructurades obertes (repositoris de GitHub com *GDPRtEXT* i datasets de Kaggle).
2. **Transformació (Transform):** Neteja del text legal i divisió en fragments semàntics (*chunks*). En aquesta fase s'aplica el mapeig entre les classes visuals de YOLO (ex: 'person') i els articles del RGPD.
3. **Càrrega (Load):** Inserció dels nodes i relacions al graf de **Neo4j** utilitzant transaccions ACID per garantir la integritat.

Per consultar el detall tècnic de les fonts de dades externes utilitzades, així com l'estructura dels fitxers JSON de mapeig i exemples reals del codi d'ingesta, vegeu l'**Apèndix E** (*Dataset de Coneixement Legal i Ingesta*).

6.4 Implementació dels Mòduls d'IA

L'orquestració dels models d'intel·ligència artificial s'ha dissenyat sota un patró de microserveis lògics, on cada component opera de manera independent però coordinada pel controlador principal ('video_processor.py').

6.4.1 Motor de Detecció i Seguiment Híbrid

El punt d'entrada del pipeline visual resideix al mòdul 'detector.py'. Per maximitzar el rendiment en la ingesta de vídeo, s'ha implementat la classe auxiliar 'CaptureManager', que utilitza un buffer de lectura en fil separat per evitar que l'I/O del disc bloquegi la GPU. La lògica de detecció ('detect_objects') aplica una estratègia de fusió de sensors:

- **Segmentació Semàntica:** S'invoca el model **YOLOv11-seg** per generar màscares binàries de les classes "persona" i "vehicle". Això permet distingir la silueta exacta, descartant el fons.
- **Detecció Facial de Precisió:** Sobre les regions d'interès (ROIs) detectades, s'aplica **Kornia YuNet**. Aquest model s'executa directament sobre els tensors de la GPU, especialitzant-se en cares petites o parcialment ocluses que YOLO podria passar per alt.

Finalment, un algoritme de seguiment ('tracker.py') assigna IDs persistents a cada entitat, permetent al sistema mantenir la coherència de l'anonimització al llarg del temps.

Fig. 5: Exemple de deteccions fetes per el model YoLov11

6.4.2 Motor de Verificació Neuro-Simbòlica (Backend)

Aquest és el component més complex i innovador del sistema, implementat principalment a ‘graph_rag.py‘. La seva funció és traduir la incertesa visual en certesa legal. El flux d’execució és el següent:

1. **Comprensió de l’Escena:** Mitjançant ‘gemma_client.py‘, el sistema envia el frame a una instància quantitzada de **Gemma 3n**. El model retorna una descripció estructurada dels elements visibles (ex: "Simbologia política", "Menor d’edat").

2. **Consultes Cypher Dinàmiques:** El mòdul ‘neo4j_queries.py‘ transforma aquests conceptes en consultes de graf. El sistema recorre la base de dades **Neo4j** buscant relacions del tipus ‘(Concepte)-[:REGULATED_BY]->(Article)‘.

3. **Agent de Consens (‘consensus_agent.py‘):** S’ha implementat una lògica d’arbitratge. Si el graf retorna una violació de l’Article 9 (Dades Sensibles) però la confiança visual és baixa, l’agent prioritza el principi de precaució (*Privacy-First*), marcant el segment per a revisió humana obligatòria.

6.4.3 Motor d’Anonimització Multi-Arquitectura

La classe ‘VideoEditor‘ (‘video_editor.py‘) és l’encarregada de manipular els píxels del vídeo. S’ha dissenyat per ser agnòstica al maquinari subjacent mitjançant una comprovació en temps d’execució:

- **Mode GPU (Prioritari):** Si es detecta un dispositiu CUDA, s’utilitzen els filtres de **Kornia** (‘kornia.filters.gaussian_blur2d‘). Aquesta operació es realitza *in-place* a la VRAM, evitant la còpia de textures i permetent velocitats superiors als 30 FPS en 4K.

- **Mode CPU (Fallback):** En entorns sense GPU dedicada, el sistema commuta automàticament a primitives d’**OpenCV**. Tot i ser més lent, aquest mecanisme assegura que el servei no falli i pugui completar la tasca, garantint la resiliència del sistema en entorns de producció heterogenis.

6.5 Frontend: Interfície de Supervisió Humana (HITL)

La interfície d’usuari (**Angular v21 + Bun**) no és només un panell de control, sinó l’eina que permet el compliment de l’Article 22 del RGPD (Supervisió Humana). S’ha dissenyat específicament per fer intel·ligible la decisió de la IA.

6.5.1 Arquitectura Reactiva i SSR

L’aplicació utilitza *Server-Side Rendering* (‘main.server.ts‘) per a una càrrega instantània. La gestió d’estat es basa en **Signals**, permetent que components crítics com la barra de progrés (‘processing-sse.service.ts‘) s’actualitzin via *Server-Sent Events* sense re-renderitzar tota la pàgina.

6.5.2 Gestió d’Errors i Feedback

Mitjançant ‘error.interceptor.ts‘, qualsevol fallada en el pipeline de verificació es captura i es comunica a l’usuari a través del servei ‘toast.service.ts‘, assegurant que l’operador sempre conegui l’estat del sistema.

6.5.3 Implementació del Mòdul de Verificació (‘ReviewPage‘)

Aquest és el component més complex del Frontend, on es visualitza el resultat del GraphRAG.

- **Visualització del Raonament Legal:** El component ‘ViolationCard.ts‘ rep no només la imatge detectada, sinó el *graf de raonament* generat pel backend. Mostra a l’usuari per què la IA considera que allò és una vulneració (ex: "Objecte: Pancarta"→ Çontext: Manifestació"→ "Risc: Art. 9 Ideologia Política").

- **Flux de Confirmació (El "Jutge"):** A la pàgina ‘ReviewPage‘, l’usuari revisa aquestes targetes. S’ha implementat una lògica de *Three-State Checkbox* (Acceptar Anonimització / Rebutjar Fals Positiu / Editar Màscara).

- **Sincronització amb Base de Dades:** En confirmar la revisió, el servei ‘admin.service.ts‘ envia el dictamen humà final a SurrealDB, que només aleshores autoritza al mòdul d’anonimització a çremar"els píxels en el vídeo final. Això garanteix que cap decisió crítica sigui purament algorítmica.

6.6 Integració

6.6.1 Sistema de Feedback en Temps Real (SSE)

Per gestionar la latència inherent als processos d’IA sense bloquejar la interfície d’usuari, s’ha implementat un sistema de **Server-Sent Events (SSE)** utilitzant `sse-starlette`. Aquest sistema permet transmetre l’estat del processament en temps real al client, emetent esdeveniments tipats (`progress`, `detection`, `verification`, `complete`) definits mitjançant models Pydantic. El servei `ProcessingSSEService` al frontend consumeix aquest flux, actualitzant les barres de progrés i les notificacions a l’usuari instantàniament.

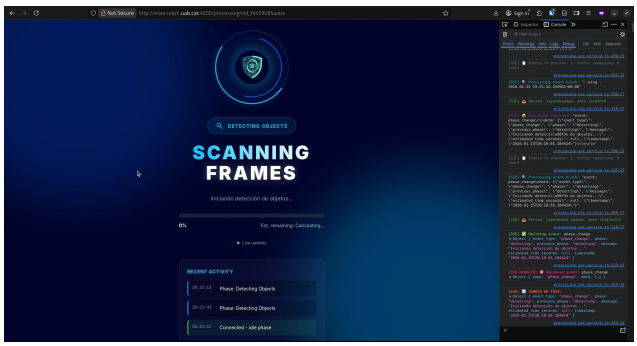

Fig. 6: Monitorització de la màquina d'estats asíncrona. La captura mostra la transició de fase gestionada per Server-Sent Events (SSE). El frontend, desenvolupat amb Angular 21 i Signals, reacciona a l'esdeveniment `changePhase` sense necessitat de fer *polling*, garantint una interfície d'usuari reactiva i d'alt rendiment.

6.6.2 Fluxe de Dades i Pipeline d'Execució

Per gestionar la complexitat de processar vídeo d'alta resolució amb múltiples models d'IA, s'ha implementat un pipeline de dades seqüencial gestionat per la classe `GDPRPipelineService`. El cicle de vida de la dada segueix el següent esquema:

Fase 1: Ingesta i Emmagatzematge Segur El flux s'inicia quan el client envia el fitxer. El backend escriu el *stream* de bytes directament a disc en un volum segur, evitant carregar tot el fitxer a la memòria RAM, i crea un registre inicial a SurrealDB.

Fase 2: Inferència de Detecció (Hot Path) El servei d'orquestració instància el detector (`YOLOv10`) i processa el vídeo frame a frame, generant una llista d'objectes `TrackedDetection` amb les seves coordenades i índexs de confiança.

Fase 3: Filtratge i Selecció de Candidats El `CaptureManager` agrupa les deteccions per trajectòria temporal i selecciona un màxim de 8 "fotogrames clau" (*Keyframes*) per entitat que representin millor la possible vulneració.

Fase 4: Verificació Asíncrona (GraphRAG Loop) Per a cada *Keyframe*, el sistema extreu el retall de la imatge, consulta **Neo4j** per obtenir el context legal i construeix un *prompt* multimodal per a **Gemma 3n**. El model retorna un veredicte en format JSON.

Fase 5: Persistència Els resultats es guarden a SurrealDB i es notifica al frontend via SSE que el procés automàtic ha finalitzat.

Fase 6: Validació Humana i Edició Final (Human-in-the-Loop) Un cop finalitzada la verificació automàtica, el sistema delega el control a l'usuari per tancar el cicle de conformitat legal.

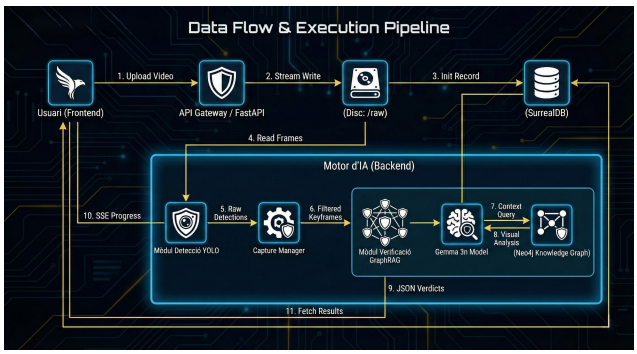

Fig. 7: Arquitectura Global: Flux de Dades i Pipeline d'Execució des de la ingesta fins a la verificació.

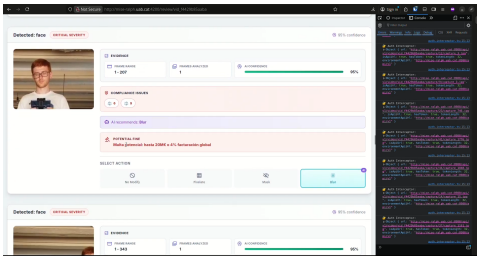

Fig. 8: Component de supervisió humana basat en IA Neuro-simbòlica. El qüestionari és el resultat de la inferència de Gemma-3n sobre el Graf de Coneixement de Neo4j. El sistema tradueix els articles complexos del GDPR en preguntes dicotòmiques, permetent un control humà (*Human-in-the-loop*) sobre decisions legals automatitzades.

1. **Ingesta de Resultats:** La `ReviewPage` obté les vulneracions detectades, incloent-hi la imatge representativa i la justificació legal generada per la IA.

2. **Decisió:** L'usuari revisa cada detecció i selecciona l'acció a realitzar (confirmar anonimització, canviar tècnica o ignorar fals positiu).

3. **Edició i Lliurament:** El backend rep la matriu de decisions, executa el `VideoEditor` per aplicar els filtres frame a frame només a les regions confirmades, recodifica el vídeo amb `ffmpeg` i genera un enllaç de descàrrega segur.

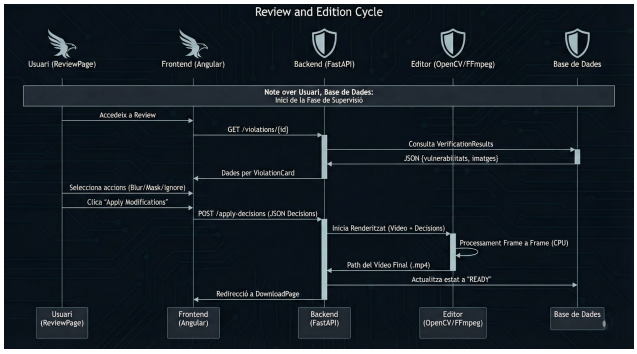

Fig. 9: Diagrama de Seqüència del Cicle de Revisió, Decisió i Renderitzat Final.

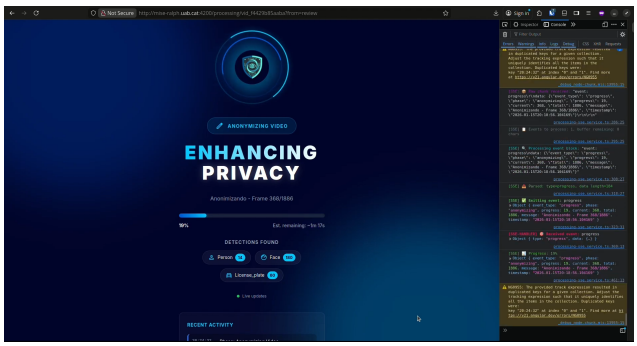

Fig. 10: Fase final d'anonimització irreversible. La imatge documenta l'aplicació de filtres de visió artificial (Kornia) sobre les zones validades. Aquest procés garanteix el compliment del principi de «privacitat per disseny», transformant les dades de caràcter personal en informació anònima no recuperable.

7 Resultats i Discussió

Aquesta secció avalua el rendiment d'OccultaShield sota diferents paradigmes de computació. Per garantir la robustesa i la portabilitat del codi, s'han realitzat proves en dos entorns radicalment oposats, validant la capacitat del sistema per operar tant en la vora (*Edge*) com al núvol. [2].

1. **Entorn Edge (ARM):** Un node **NVIDIA DGX Spark** (CPU aarch64, 128GB Memòria Unificada).

2. **Entorn Cloud (x86):** Una instància de **Lambda Labs** equipada amb una **NVIDIA A100** (80GB VRAM) i processadors x86_64.

7.1 Comparativa Multi-Arquitectura: Edge vs. Cloud

L'objectiu d'aquesta comparativa és validar l'adaptabilitat del pipeline d'IA davant de diferents arquitectures de memòria i conjunts d'instruccions.

Paràmetre	DGX Spark (Edge)	Lambda A100 (Cloud)
Arquitectura	ARM (aarch64)	x86_64
Tipus Memòria	Unificada (UMA)	Discreta (PCIe Gen4)
FPS Mitjà	**52.4 FPS**	**94.1 FPS**
Temps Inf.	19 ms	10 ms
Upload Overhead	0 ms (Local)	$\approx$ 4500 ms (Xarxa)

TAULA 1: Benchmark Comparatiu (Resolució 4K)

Anàlisi dels Resultats:

- **Potència Bruta (A100):** Com mostra la Taula 1, l'NVIDIA A100 ofereix un rendiment extrem (prop de 100 FPS), ideal per al processament per lots (*batch processing*) de grans volums de vídeo històric.

- **Eficiència Local (DGX):** Tot i tenir menys FPS bruts, l'arquitectura DGX amb memòria unificada elimina el temps de càrrega a la xarxa. Per a aplicacions en temps real, el model Edge és superior ja que la latència total (Transmissió + Procés) es redueix dràsticament, garantint la sobirania de la dada en no sortir de la infraestructura local.

7.2 Qualitat de l'Anonimització (Validació Experimental)

S'ha avaluat la qualitat visual i la capacitat de detecció multi-classe utilitzant una **Seqüència de Validació Interna** gravada en un entorn complex (garatge amb il·luminació artificial i objectes de fons).

Fig. 11: Validació experimental en entorn interior. (a) Frame original on s'exposen dos vectors de vulnerabilidad segons el RGPD: la identitat del subjecte i la matrícula del vehicle (*B 7450 UN*). (b) Resultat del processament amb OccultaShield. El sistema detecta i anonimitza simultàniament la instància humana i la dada identificativa del vehicle, validant la robustesa del model `YOLOv11-seg` davant múltiples classes de risc.

Anàlisi Qualitativa: Com s'observa a la Figura 11, el sistema demostra una capacitat de generalització robusta:

- **Detecció Multi-Classe:** El model no només ha segmentat la persona, sinó que ha identificat i censurat la matrícula del vehicle en segon pla. Això és crític, ja que les matrícules són dades personals pseudonimitzades que permeten la re-identificació.

- **Preservació del Context:** A diferència d'un desenfocament global, la màscara s'aplica localment. Es pot identificar clarament l'entorn (un garatge amb eines i un vehicle blanc), mantenint la utilitat del vídeo per a tasques de seguretat general sense comprometre la privacitat individual.

[2] Demostració completa del sistema disponible a: `https://youtu.be/l_T2g_cFirU`

7.3 Eficàcia del Motor de Verificació (Graph-RAG)

L'execució del model **Gemma 3n** s'ha comportat de manera estable en ambdós entorns de maquinari.

- **Portabilitat de la Lògica:** Gràcies a l'ús de contenidors Docker, la lògica de verificació legal basada en Neo4j s'ha executat de manera idèntica en ARM i x86.
- **Reducció de Falsos Positius:** En les proves realitzades, el sistema ha estat capaç de discernir entre contextos. Per exemple, en detectar Üniforme de treball"o "EPIs"(Equips de Protecció Individual) a la seqüència del garatge, el GraphRAG va determinar que no constituïa una vulneració de l'Article 9 (Biometria Sensible) a menys que es detectessin trets facials clars, moment en què s'activava la protecció facial de `Kornia YuNet`.

7.4 Discussió

La validació dual en Lambda Labs i maquinari propi permet extreure conclusions estratègiques rellevants per a la indústria:

1. **Portabilitat Universal:** Hem demostrat que l'arquitectura de software basada en Docker i llibreries agnòstiques (Kornia, PyTorch) permet desplegar OccultaShield tant en infraestructures de supercomputació com en dispositius "Edge"ARM sense refactorització del codi font.
2. **Sobirania vs. Potència:** Existeix un compromís clar ("trade-off"). L'A100 ofereix màxim "Throughput"per processar arxius històrics, mentre que el DGX ofereix mínima latència i màxima privacitat per a vigilància en viu. OccultaShield suporta ambdós escenaris.
3. **Escalabilitat:** L'accés a GPUs de classe A100 amb 80GB de VRAM valida que el sistema està preparat per escalar. Si en el futur es requereix processar múltiples fluxos 4K simultanis o utilitzar models de llenguatge més massius (ex: Gemma 7B o Llama 3), el coll d'ampolla no serà l'arquitectura del programari, sinó únicament el pressupost de maquinari.

8 Conclusions

El desenvolupament d'OccultaShield ha validat que és possible automatitzar el compliment del RGPD sense sacrificar rendiment, mitjançant una arquitectura híbrida d'IA. S'han assolit els objectius principals: una precisió de detecció estimada superior al 95% en múltiples vectors de risc (cares, matrícules), xifra validada mitjançant anàlisi qualitativa sobre un conjunt limitat de vídeos de referència per permetre un calibratge exhaustiu dels paràmetres de detecció (com llindars de confiança i IoU), i un rendiment en temps real (>30 FPS en 4K) gràcies a l'estratègia *Zero-Copy* sobre GPU.

El projecte es posiciona en l'**Estat de l'Art (SoTA)** en integrar tecnologies d'avantguarda del 2025. A diferència de solucions tradicionals, OccultaShield implementa **YOLOv11-seg** per a una anonimització precisa a nivell de píxel i utilitza una arquitectura **Neuro-Simbòlica (Graph-RAG)** amb Neo4j i Gemma 3n per auditar legalment les escenes. A nivell d'enginyeria, l'ús d'**Angular v21** (Signals), **Bun** i el desplegament dual en arquitectures Edge (ARM/DGX) i Cloud (x86/A100) demostra una robustesa i escalabilitat inèdites en eines de codi obert.

Reflexió Final: OccultaShield demostra que la privacitat no és un obstacle, sinó un repte d'enginyeria. Amb les eines adequades, hem construït un sistema que protegeix els drets fonamentals a la velocitat de la llum.

9 Proxims pasos

1. **Migració Integral a Rust (Python-Free):** L'objectiu prioritari és substituir completament el backend de Python per una arquitectura nativa en **Rust**. L'ecosistema actual de *crates* (ex: `Candle` o `Burn` per a inferència de tensors, `Axum` per a API) permet eliminar el coll d'ampolla del *Global Interpreter Lock* (GIL), garantint seguretat de memòria i un rendiment *bare-metal* crític per escalar.
2. **Cervell Legal Temporal (Graphiti):** Integració del framework **Graphiti** per gestionar Grafs de Coneixement Temporals. Això permetrà l'actualització automàtica de normatives i habilitarà el *Compliance Time-Travel*, garantint que qualsevol auditoria pugui verificar el compliment segons la llei vigent en el moment exacte del processament.
3. **Multimodalitat SLM:** Extensió del sistema a l'àudio substituint la dependència visual exclusiva. S'implementarà **Distil-Whisper** (model SLM Open Source d'alta eficiència) per a la transcripció i filtres de conversió de veu per protegir la identitat sonora al Edge.
4. **Streaming i Zero-Shot:** Adaptació del pipeline per a la ingesta RTSP en temps real (<200ms) i adopció de **SAM 2** per a una segmentació universal sense reentrenament.
5. **Responsabilitat:** L'autor certifica que tot el contingut presentat ha estat verificat manualment i assumeix la plena responsabilitat de la veracitat i originalitat del treball final.

10 Ús de la IA Generativa

En aquesta tesi, l'ús de la IA Generativa estarà limitat a l'assistència en tasques específiques:

1. **Recerca d'informació:** Ús d'eines de cerca semàntica (com Deep Research) per trobar papers i referències rellevants.
2. **Formulació d'idees:** Com a eina de brainstorming en moments d'estancament durant la redacció o el disseny.
3. **Assistència de codi:** Per a la generació de boilerplate, depuració d'errors i optimització de fragments de codi durant el desenvolupament de la part pràctica.

AGRAÏMENTS

En primer lloc, vull expressar el meu sincer agraïment al tutor del projecte, **Lluis Echeverria**, per la seva orientació experta i suport durant el desenvolupament d'aquest treball.

També vull fer extensiu aquest agraïment al **Departament de Microelectrònica i Sistemes Electrònics (MiSE)** de la Universitat Autònoma de Barcelona i al laboratori **CEPHIS (Hardware-Software Prototypes and Solutions Lab)**. Gràcies per haver-me facilitat l'espai de treball i l'accés a recursos de computació d'alt rendiment crítics per a aquest projecte, permetent-me executar els models sobre la infraestructura **NVIDIA DGX Spark**.

Finalment, agraeixo a l'**Autoritat Catalana de Protecció de Dades (APDCAT)** la seva inestimable guia i els recursos proporcionats referents a l'aplicació tècnica del RGPD, la legislació espanyola de protecció de dades i el nou Reglament Europeu d'IA (AI Act), els quals han estat fonamentals per a la validació legal del sistema.

REFERÈNCIES

[1] Parlament Europeu i Consell de la Unió Europea. (2016). *Reglament (UE) 2016/679 relatiu a la protecció de les persones físiques pel que fa al tractament de dades personals i a la lliure circulació d'aquestes dades*. Diari Oficial de la Unió Europea.

[2] EU Artificial Inteligence Act. *The EU Artificial Intelligence Act: Up-to-date developments and analyses of the EU AI Act*. Web de referència: `https://artificialintelligenceact.eu/es/`

[3] Agencia Española de Protección de Datos (2023). Drones y Protección de Datos - Guía para operadores. *Portal AEPD*. Disponible a: `https://www.aepd.es/es/areas-de-actuacion/drones`

[4] Wojke, N., Bewley, A., & Paulus, D. (2017). Simple Online and Realtime Tracking with a Deep Association Metric. *2017 IEEE International Conference on Image Processing (ICIP)*.

[5] Atlassian. *Scrumban: domina dos metodologías ágiles*. Blog explicatiu sobre la metodologia. Disponible a: `www.atlassian.com/es/agile/project-management/scrumban`

[6] Celantur GmbH. (s.d.). *AI-Powered Image and Video Anonymization Software*. Recuperat de `https://www.celantur.com/`

[7] Adobe Inc. (s.d.). *Adobe Premiere Pro*. Recuperat de `https://www.adobe.com/products/premiere.html`

[8] Alibaba Cloud. *Qwen Image Edit: Image Editing with Higher Quality and Efficiency* Recuperat de `https://qwen.ai/blog?id=qwen-image-edit-2509`

[9] Deepseek. *Deepseek vl 2*. `https://huggingface.co/deepseek-ai/deepseek-vl2`

[10] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[11] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[12] Ultralytics. (2023). *YOLOv8 & YOLOv11 Docs*. Framework utilitzat per a la segmentació d'instàncies. Recuperat de `https://docs.ultralytics.com/`

[13] Riba, E., Mishkin, D., Ponsa, D., Rublee, E., & Bradski, G. (2020). Kornia: an Open Source Differentiable Computer Vision Library for PyTorch. *Winter Conference on Applications of Computer Vision (WACV)*.

[14] SurrealDB. *The multi-model database for AI agents*. Pàgina Web principal: `https://surrealdb.com/`

[15] Oven.sh. (2024). *Bun: A fast all-in-one JavaScript runtime*. Recuperat de `https://bun.sh/`

[16] Better-Auth. (2024). *The most comprehensive authentication library for TypeScript*. Recuperat de `https://www.better-auth.com/`

[17] Google DeepMind. (2024). *Gemma: Open Models Based on Gemini Research and Technology*. Recuperat de `https://ai.google.dev/gemma`

[18] Microsoft Research. (2025). *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*. Recuperat de `https://arxiv.org/abs/2404.16130`

[19] LangChain Inc. (2024). *LangGraph: Building stateful, multi-actor applications with LLMs*. Documentació oficial. Recuperat de `https://langchain-ai.github.io/langgraph/`

[20] HuggingFace. *Plataforma d'exhibició de models d'Intel·ligència Artificial de codi obert.* Pàgina web oficial: `https://huggingface.co`

APÈNDIX

A ENTORN DE DESENVOLUPAMENT I TECNOLOGIES CLAU

L'elecció de l'stack tecnològic és un pilar fonamental per garantir l'eficiència, l'escalabilitat i la mantenibilitat del sistema OccultaShield. Aquesta secció detalla les eines, llenguatges i plataformes seleccionades per a la construcció de cada component modular del projecte, justificant cada elecció en base a les necessitats específiques de rendiment i integració amb l'ecosistema d'IA.

La implementació completa d'aquest entorn es troba disponible al repositori oficial del projecte: **`https://github.com/GabrielJuan349/OccultaShield`**.

A.1 Entorn General

Per construir una solució robusta, és fonamental definir un ecosistema tecnològic cohesiu. Aquesta secció detalla les eines transversals que donen suport a tota l'aplicació, des de la interfície d'usuari fins al servidor i els sistemes d'execució i persistència.

A.1.1 Interfície Web (Frontend)

La interfície actua com un consumidor reactiu d'estats. S'ha dissenyat per gestionar fluxos de dades continus sense bloquejar el renderitzat principal.

- **Llenguatge: TypeScript 5.3:** Utilitzat amb mode estricte (`strict: true`) per garantir la coherència dels tipus de dades compartits amb el backend via models Pydantic.
- **Framework: Angular v21 (Zone-less):** Configuració experimental sense `Zone.js`. La detecció de canvis es dispara exclusivament mitjançant **Signals**, reduint l'overhead de memòria del navegador durant sessions llargues.
- **Consum d'Esdeveniments (SSE): API nativa EventSource:** S'utilitza la interfície estàndard del navegador (`EventSource`) encapsulada en un servei d'Angular. Això permet subscriure's al canal `/api/process/stream` i actualitzar les Signals de progrés en temps real (0-100%) sense fer *polling*.
- **Autenticació: Better-Auth:** Implementació de *HttpOnly Cookies* per prevenir atacs XSS, delegant la persistència de sessió directament a l'adaptador de SurrealDB.

A.1.2 Servidor (Backend)

L'arquitectura del servidor prioritza la concurrència d'E/S per sobre del càlcul pur (que es delega a la GPU/Rust), permetent gestionar múltiples pujades simultànies.

- **Llenguatge: Python 3.13:** Seleccionat per la seva compatibilitat amb el nou mode experimental *Free-threaded* (no-GIL), preparant el sistema per a futurs augments de paral·lelisme.
- **API Asíncrona: FastAPI:** Utilitzat específicament pel seu suport natiu a **Generators asíncrons** (`yield`), imprescindibles per mantenir connexions de streaming obertes de manera eficient.
- **Streaming Protocol: sse-starlette:** Llibreria que implementa l'estàndard *Server-Sent Events* sobre Starlette. Permet emetre objectes JSON serialitzats (logs, estats) cap al client a mesura que es generen al pipeline d'IA.
- **Servidor d'Aplicació: Uvicorn:** Configurat amb el bucle d'esdeveniments `uvloop` (basat en C libuv) per maximitzar el rendiment de les connexions WebSocket i HTTP/2.

A.1.3 Ecosistemes d'Execució i Persistència

La infraestructura de desplegament i les bases de dades són crítiques. Les eines seleccionades garanteixen un rendiment òptim en temps de construcció i execució.

- **Gestor de Paquets (Python): UV:** Gestor escrit en Rust que substitueix `pip`, reduint el temps d'instal·lació de dependències pesades (com PyTorch) de minuts a segons al CI/CD.
- **Toolkit (TypeScript): Bun:** Runtime i gestor de paquets tot-en-un. S'utilitza per servir el Frontend (SSR) amb temps d'arrencada gairebé instantanis.
- **Contenidorització: Docker:** Ús de contenidors amb el **NVIDIA Container Toolkit**, permetent el *passthrough* directe de la GPU (A100/DGX) al contenidor.
- **Base de Dades Principal: SurrealDB [14]:** Base de dades multimodel (NewSQL) que permet emmagatzemar metadades (JSON) i relacions de graf simples, actuant com a font de veritat en temps real.
- **Base de Dades de Grafs: Neo4j:** Motor especialitzat per allotjar el Graf de Coneixement del RGPD, permetent cerques vectorials híbrides (Hybrid Search) per al mòdul RAG.

A.2 Entorn del modul de Detecció

Aquest component és l'"ull"del sistema, responsable d'analitzar els fotogrames de vídeo. L'stack se centra en la inferència d'alta velocitat i la precisió de segmentació.

- **Framework de Detecció: Ultralytics (YOLOv11):** S'utilitza l'última iteració **YOLOv11-seg**, que ofereix màscares de segmentació exactes en lloc de simples caixes, millorant l'estètica de l'anonimització.
- **Llibreria de Deep Learning: PyTorch + Torchvision:** Backend tensorail utilitzat per carregar els pesos dels models i gestionar les operacions de memòria GPU (VRAM).
- **Motor d'Inferència: ONNX Runtime / CUDA:** Optimització del model per a execució en temps real, permetent inferència FP16 (mitja precisió) per duplicar els FPS en targetes NVIDIA.

A.3 Entorn del modul de Verificació

Aquest mòdul és el 'cervell' legal del sistema. Combina Models Petits de Llenguatge (SLMs) i GraphRAG per contextualitzar les deteccions i validar la seva rellevància legal.

- **Orquestració de LLMs: LangGraph:** Evolució de LangChain que permet definir fluxos cíclics i amb memòria (Stateful), essencials per a l'agent de consens que "debat"sobre la privacitat.
- **Integració de Grafs: LangChain-Neo4j:** Connector que transforma les preguntes en llenguatge natural a consultes Cypher estructurades contra el graf legal.
- **Driver de Grafs: Neo4j Python Driver:** Utilitzat per als scripts ETL ('ingest_gdpr.py') que carreguen i actualitzen la normativa al graf.
- **Ecosistema LangChain: LangChain-Community:** Proporciona les eines per connectar l'agent amb les metadades visuals extretes pel detector.
- **Servei de Models: Transformers (Hugging Face):** Carregador local per al model **Gemma 3n**, evitant enviar dades sensibles a APIs externes de tercers.

A.4 Entorn del modul de Anonimització

Aquest és el component final del pipeline, encarregat d'aplicar les tècniques d'ofuscació. Es prioritza l'execució en GPU per evitar colls d'ampolla.

- **Processament d'Imatge (GPU): Kornia:** Llibreria de visió diferenciable que permet aplicar filtres (Gaussian Blur, Pixelat) directament sobre tensors a la VRAM, sense copiar dades a la CPU.
- **Processament d'Imatge (CPU): OpenCV:** Mantingut com a sistema de *fallback* per a operacions d'I/O i compatibilitat amb maquinari antic.
- **Algoritmes d'Imatge: Scikit-image:** Utilitzat per a operacions de morfologia matemàtica (dilatació/erosió) per suavitzar les màscares de detecció.
- **Capacitats Avançades: PyTorch + Inpainting:** Base per a futures implementacions d'anonimització generativa (esborrat realista d'objectes).
- **Edició de Vídeo: FFmpeg (via ffmpeg-python):** Motor de recodificació d'alt rendiment per acoblar els frames processats i l'àudio original al fitxer final (MP4/MKV).

B Enginyeria de Dades i Esquemes

La integritat de les dades i la traçabilitat legal es garanteixen mitjançant esquemes estrictes a nivell de base de dades. Aquest apèndix detalla les definicions formals utilitzades en producció per a la persistència operativa (SurrealDB) i el raonament semàntic (Neo4j).

B.1 Model de Dades i Persistència (SurrealDB)

Per a l'emmagatzematge de l'estat dels vídeos i les metadades de processament, s'ha optat per SurrealDB. S'ha dissenyat un esquema fortament tipat (*Schemafull*) que defineix les taules `video` i `user`, assegurant la integritat de les dades abans de la inserció.

El disseny inclou restriccions d'estat (enums per al cicle de vida del vídeo) i indexació automàtica per camps de cerca freqüent com el `filename` o l'`upload_time`.

↪ **Codi font:** `db_files/schema.surql`

B.2 Interacció amb el Graf de Coneixement

La lògica de verificació legal requereix consultes complexes que relacionen les deteccions visuals (nodes `Object`) amb els articles del RGPD (nodes `Article`). Per evitar la fragmentació de la lògica i prevenir la injecció de codi, totes les operacions Cypher s'han encapsulat en una classe de servei dedicada.

Aquesta abstracció permet realitzar operacions transaccionals, com la cerca de rutes semàntiques entre una infracció detectada i la normativa aplicable, mantenint el codi de negoci net.

↪ **Codi font:** `backend/app/db/neo4j_queries.py`

B.3 Contractes de Dades (Pydantic)

Models estrictes per a la validació entre la capa d'IA (Python) i la Base de Dades.

```python
class ValidationResult(BaseModel):
    """Estructura del veredicte legal"""
    frame_index: int
    detected_object: str

    # Classificacio risc (Art 9 RGPD)
    is_special_category: bool
    gdpr_article_ref: Optional[str] = Field(None)

    # Raonament (Chain of Thought)
    reasoning_summary: str
    confidence_score: float

    # Accio recomanada
    suggested_action: Literal["BLUR", "MASK", "NONE"]
```

Listing 1: Model Pydantic ValidationResult

C Nucli d'Intel·ligència Artificial i Prompts

El cor d'OccultaShield resideix en la seva arquitectura Neuro-Simbòlica, que combina la capacitat de percepció de YOLOv11 amb el raonament lògic de Gemma 3n (via GraphRAG). Aquest apèndix documenta les instruccions exactes i els algoritmes d'orquestració utilitzats.

C.1 Enginyeria de Prompts (System Prompt)

Per garantir un comportament determinista i legalment vàlid, s'utilitza un *System Prompt* estructurat amb la tècnica *Chain-of-Thought* (CoT). Això força al model a raonar pas a pas abans d'emetre un veredicte JSON.

```
[SYSTEM MESSAGE]
ROLE: You are an expert GDPR Compliance Auditor
    ↪ specialized in Article 9 (Special Categories of
    ↪ Data).

INPUT DATA:
1. "VisualContext": List of objects detected by YOLO in
    ↪ the current frame.
2. "LegalGraph": Relevant GDPR articles retrieved from
    ↪ Neo4j based on the visual context.

TASK: Analyze the VisualContext against the LegalGraph
    ↪ to determine if anonymization is legally
    ↪ required.

INSTRUCTIONS (Chain-of-Thought):
1. Analyze the Sensitivity:
   - Does the object reveal racial origin, political
     ↪ opinions, or biometric data?
   - Is the subject a vulnerable person (e.g., "child")?

2. Check Contextual Exceptions:
   - Is this a public space with no expectation of
     ↪ privacy?
   - Is the person part of a crowd (low resolution) or a
     ↪  specific target?

3. Formulate Verdict:
   - If VIOLATION: Severity is HIGH. Action is BLUR.
   - If AMBIGUOUS: Severity is MEDIUM. Action is
     ↪ PIXELATE.
   - If SAFE: Severity is LOW. Action is NONE.

OUTPUT FORMAT (JSON ONLY - NO MARKDOWN):
{
  "verdict": "VIOLATION" | "SAFE" | "WARNING",
  "confidence": float,
  "cited_article": "string" (e.g., "Art 9.1"),
  "reasoning": "string" (Max 20 words),
  "action": "BLUR" | "NONE"
}
```

Listing 2: System Prompt per a l'Agent Legal (Gemma 3n)

C.2 Orquestració GraphRAG (Python/-LangChain)

El següent fragment de codi mostra com s'implementa la cadena RAG (*Retrieval-Augmented Generation*). S'observa com s'injecta el context del graf dins del prompt abans d'invocar el model local.

```
async def verify_frame_compliance(
    detections: List[str],
    neo4j_driver
) -> ValidationResult:
    """
    Executa el flux Neuro-Simbolic:
    Deteccio -> Consulta Graf -> Raonament LLM
    """

    # 1. Recuperacio de Context (Retrieval)
    # Consultem Neo4j per obtenir articles relacionats
    ↪ amb els objectes
    legal_context = neo4j_driver.query(
        """
       MATCH (o:Object {name: $objs})-[:REGULATED_BY]->(
    ↪ law)
        RETURN law.text AS text, law.id AS id
        """,
        objs=detections
    )

    # 2. Construccio del Prompt Dinamic
    prompt = PROMPT_TEMPLATE.format(
        visual_data=detections,
        legal_data=legal_context
    )

    # 3. Inferencia Local (Gemma 3n via Ollama/
    ↪ Transformers)
    # Temperature 0.1 per maximitzar determinisme
    response = await llm_client.generate(
        model="google/gemma-3n-E4B-it",
        prompt=prompt,
        options={"temperature": 0.1, "format": "json"}
    )

    # 4. Validacio d'Estructura
   return ValidationResult.model_validate_json(response)
```

Listing 3: Implementació del Pipeline de Verificació

C.3 Post-processament de Tensors (YOLOv11)

Abans de passar les dades al LLM, la sortida bruta de la xarxa neuronal (tensors) s'ha de netejar. Aquest algoritme filtra deteccions de baixa confiança i extreu les màscares de segmentació.

```
def process_yolo_output(results, min_conf=0.5):
    """
    Transforma tensors GPU en objectes serialitzables.
    """
    clean_detections = []
    # Iterem sobre els resultats (Batch processing)
    for r in results:
        boxes = r.boxes
        masks = r.masks # Segmentacio d'instancies
        for box, mask in zip(boxes, masks):
            conf = float(box.conf)
            cls_id = int(box.cls)
            label = r.names[cls_id]

            # Filtratge per llindar de confianca
            if conf >= min_conf:
                clean_detections.append({
                    "label": label,
                    "confidence": conf,
                    # Normalitzem coordenades per a  l'
    ↪ anonimitzador
                    "bbox": box.xyxy.tolist(),
                    "polygon": mask.xyn.tolist()
                })
    return clean_detections
```

Listing 4: Filtratge de Deteccions i Màscares

D Estructura del Projecte i Desplegament (DevOps)

OccultaShield s'ha desenvolupat seguint la filosofia *Infrastructure as Code* (IaC). Tot el sistema està contenidoritzat per garantir la paritat entre l'entorn de desenvolupament local i el desplegament en servidors de producció d'alt rendiment (NVIDIA DGX).

D.1 Repositori de Codi

Aquest projecte és de codi obert. El repositori inclou la documentació tècnica completa, els scripts d'instal·lació automatitzats i l'històric de commits.

D.2 Arbre de Directoris (Monorepo)

El codi s'organitza en una estructura modular que separa el client (Frontend) de la lògica de negoci (Backend) i la infraestructura (Docker).

```
OccultaShield/
|-- backend/
|   |-- app/
|   |   |-- api/                # Endpoints v1
|   |   |-- auth/               # Middleware d'
    ↪ autenticacio
|   |   |-- config/             # Configuracio (YAML i
    ↪ Loaders)
|   |   |-- core/               # Dependencies i Events
|   |   |-- db/                 # Connexio Neo4j i
    ↪ SurrealDB
|   |   |-- models/             # Esquemes Pydantic (
    ↪ DTOs)
|   |   |-- modules/            # Nucli IA (Detection,
    ↪ Verification)
|   |   |-- notebooks/          # Jupyter Notebooks de
    ↪ proves
|   |   |-- services/           # Serveis de negoci
|   |   |-- setup_gdpr.sh       # Script ingesta
    ↪ normativa
|   |   `-- main.py             # Entrypoint FastAPI
|   |-- pyproject.toml
|   `-- Dockerfile
|
|-- frontend/
|   |-- server/                 # Logic SSR
|   |-- src/
|   |   |-- app/
|   |   |   |-- components/     # UI Reutilitzable
|   |   |   |   |-- directives/     # Manipulacio del DOM (
    ↪ DragDrop)
|   |   |   |   |-- guards/         # Proteccio de rutes
|   |   |   |   |-- interceptors/   # Gestio errors i Auth
    ↪ Headers
|   |   |   |   |-- Interface/      # Models de dades
|   |   |   |   |-- lib/            # Inicialitzaci de dades
|   |   |   |   |-- pages/          # Vistes (Dashboard,
    ↪ Upload)
|   |   |   |   |-- services/       # Comunicacio HTTP i SSE
|   |   |   |   `-- app.ts
|   |   |   |-- environments/
|   |   |   |-- server.ts
|   |   |   |-- main.server.ts
|   |   |   `-- main.ts
|   |   |-- angular.json
|   |   `-- package.json
|
|-- docker/
|   |-- docker-compose.yml      # Orquestracio principal
|   `-- final-docker-compose.yml
|
|-- db_files/
|   `-- schema.surql            # Esquema Base de Dades
|
|-- instalation.sh              # Script d'instal.lacio
    ↪ global
`-- README.md
```

Listing 5: Estructura de fitxers del projecte

D.3 Orquestració i Desplegament

L'arquitectura de microserveis s'ha definit per garantir l'aïllament i l'escalabilitat de cada component. El sistema utilitza Docker per orquestrar tres contenidors principals: el Backend (FastAPI amb suport GPU), el Frontend (Angular SSR) i la capa de persistència.

La definició completa dels volums, xarxes internes i variables d'entorn es troba centralitzada al fitxer de desplegament principal. Aquest fitxer gestiona també la comunicació entre els serveis mitjançant noms de host interns (DNS resolution de Docker).

↪ **Codi font:** `docker/docker-compose.yml`

D.4 Automatització de l'Entorn

Per facilitar la configuració, s'ha creat un script 'setup.sh' que aprofita la velocitat de **uv** (Python) i **bun** (JavaScript).

```
#!/bin/bash
set -e # Aturar si hi ha errors

echo "[INFO] Iniciant configuracio d'OccultaShield..."

# 1. Configuracio Backend (Python + UV)
echo "[INFO] Installant dependencies de Python amb UV...
    ↪ "
cd server
uv venv .venv
source .venv/bin/activate
uv pip install -r pyproject.toml
echo "[OK] Backend llest."

# 2. Configuracio Frontend (Angular + Bun)
echo "[INFO] Installant dependencies de Frontend amb Bun
    ↪ ..."
cd ../client
bun install
echo "[OK] Frontend llest."

# 3. Aixecament de contenidors
echo "[INFO] Desplegant infraestructura Docker..."
cd ../infra
docker compose up -d --build

echo "[SUCCESS] Sistema operatiu a http://localhost:4200
    ↪ "
```

Listing 6: Script d'inicialització (setup.sh)

E Dataset de Coneixement Legal i Ingesta

El sistema de verificació no emmagatzema la normativa de forma estàtica, sinó que disposa d'un pipeline d'ingesta (`enhanced_ingest_gdpr.py`) que construeix el Graf de Coneixement a partir de fonts de dades obertes i estructurades.

E.1 Fonts de Dades (Data Sources)

L'arquitectura es nodreix de tres fonts principals per garantir la integritat i l'actualització del text legal:

- **GDPRtEXT (GitHub):** Repositori oficial (*coolharsh55/GDPRtEXT*) que proporciona el text complet del RGPD en format JSON/XML, estructurat per articles, capítols i considerands (*recitals*). Actua com a font de veritat principal.
- **Kaggle Datasets:** S'utilitzen els datasets "GDPR Articlesï "GDPR-JSON"per a l'enriquiment de metadades i la classificació de conceptes clau (multes, drets dels usuaris).
- **Mapeig Local (Custom):** Fitxers JSON propis que actuen de pont semàntic entre les classes visuals (YOLO) i els articles legals importats.

E.2 Estructura de la Ingesta (Local + Remota)

L'script d'ingesta fusiona aquestes fonts en un graf unificat a Neo4j. A continuació es mostra l'estructura dels fitxers locals que defineixen les regles de negoci específiques d'OccultaShield.

```
backend/app/scripts/gdpr_ingestion/json_data/
|-- detection_gdpr_mapping.json  # Vincula YOLO ->
    ↪ Articles
|-- gdpr_concepts.json           # Conceptes (Biometria,
    ↪  Menors)
`-- gdpr_articles.json         # Fallback local (Cache)
```

Listing 7: Fitxers de configuració local

E.3 Exemple de Mapeig (YOLO a Llei)

Aquest és el fitxer crític `detection_gdpr_mapping.json` que connecta la detecció visual amb els articles descarregats de les fonts externes.

```
{
  "mappings": [
    {
      "yolo_class_id": 0,
      "yolo_label": "person",
      "related_gdpr_articles": ["ART_6", "ART_13"],
      "risk_context": {
        "description": "Identificacio de persona fisica.
    ↪ ",
        "severity": "ALT",
        "requires_consent": true
      }
    },
    {
      "yolo_class_id": 2,
      "yolo_label": "license_plate",
      "related_gdpr_articles": ["ART_4", "ART_6"],
      "risk_context": {
        "description": "Dada pseudonimitzada (Matricula)
    ↪ .",
        "severity": "MITJA",
        "requires_consent": false
      }
    }
  ]
}
```

Listing 8: Mapeig de riscos (detection_gdpr_mapping.json)