Gregory Burns

burnsgregm@gmail.com 831-226-6907 San Francisco Bay Area
linkedin.com/in/gregburns github.com/burnsgregm burnsgregm.netlify.app/portfolio.html

PrivacyScrub V5 — One-Page Summary

Author: Greg Burns

Role: Systems Architect · ML Engineer · MLOps

Tech Stack: Python, FastAPI, Streamlit, YOLOv8-X, DeepSORT/ReID, OpenCV, FFmpeg, GCP (Cloud Run, Firestore, Cloud Tasks, GCS), Terraform

Overview

PrivacyScrub is a GPU-accelerated video anonymization service that detects, tracks, and obscures faces, license plates, and other sensitive entities in user-uploaded video. It turns raw, privacy-sensitive footage into shareable, redacted assets by combining object detection, temporal tracking, and chunked processing in a production-style, cloud-native pipeline. A Streamlit front end drives a FastAPI backend, which orchestrates work across Cloud Run workers, Firestore job state, and GCS for raw, intermediate, and final video artifacts.

Problem and Solution

Organizations increasingly rely on video—dashcam footage, workplace cameras, screen recordings—for analytics, training, and collaboration, but those assets are loaded with personally identifiable information (PII). Manual redaction is slow, inconsistent, and not scalable; naive model passes often miss frames, flicker, or fail on longer clips. PrivacyScrub solves this by providing an end-to-end anonymization pipeline: users upload a video, the system splits, processes, and reassembles it, and returns a redacted output where sensitive entities are consistently anonymized across time. The architecture is designed for long-form video, GPU workloads, job reliability, and privacy-first defaults.

How PrivacyScrub Works (High Level)

1. **Ingest**: A user uploads a video via the Streamlit UI. The FastAPI gateway stores the raw file in GCS, creates a job document in Firestore (QUEUED state), and enqueues work using Cloud Tasks / PubSub for asynchronous processing.
2. **Chunk**: An orchestrator worker pulls the job, downloads the original video, and uses FFmpeg to split it into fixed-size chunks. This allows the system to handle multi-minute videos without hitting timeouts or memory limits, while tracking progress per chunk.
3. **Detect & Track**: A GPU-enabled worker processes each chunk, using YOLOv8-X to detect faces, license plates, and optionally full-body regions. DeepSORT / re-identification embeddings are used to track entities across frames and chunks, ensuring a person or vehicle receives consistent anonymization even if they leave and re-enter the scene.
4. **Anonymize**: For each tracked entity, the worker applies configurable anonymization (blur, pixelation, or hard masking), with temporal smoothing to reduce bounding-box jitter and flicker. Redacted chunks are written back to GCS, and Firestore is updated with per-chunk completion and any errors.
5. **Stitch & Finalize**: Once all chunks complete, the orchestrator downloads the processed segments, concatenates them via FFmpeg, normalizes codecs, strips metadata, and writes the final redacted video to GCS. The job is marked COMPLETED in Firestore, and the user can download a share-safe version from the UI.

Business Impact

PrivacyScrub showcases my ability to design and ship privacy-critical, production-style ML systems, not just standalone models:

- Translating a vague requirement (“make videos safe to share”) into a concrete architecture and job lifecycle that works for long-form, GPU-heavy workloads.
- Combining detection, tracking, re-identification, and chunk-aware stitching into a cohesive pipeline that behaves reliably across retries and large files.
- Using cloud-native patterns (asynchronous workers, job state, idempotent operations, IaC with Terraform) to make the system robust, observable, and deployable.

In short, PrivacyScrub V5 demonstrates how I approach problems where computer vision, privacy engineering, and MLOps all need to come together to deliver a system that real teams could trust in a compliance-focused environment..