CS M213A / ECE M202A (Fall 2025)

From Words to Shields: Agentic Privacy for Smart Sensors

Daniel Luzzatto, Sam Seban

Project Mentor: Brian Wang

Outline

- Motivation / Problem Statement
- Related / Prior Work
- Our Goals & Contributions
- Team Contributions
- Technical Method / Algorithm
- Results & Findings
- Conclusions & Future Work
- Live Demo

Motivation

- Smart environments stream raw audio/video to cloud services → privacy risk.
- Users cannot express simple privacy rules (e.g., “blur faces,” “mute medical terms”).
- Existing solutions rely on manual, rigid pipelines that don’t scale or adapt to changing policies or are not robust.
- Need a system that enforces privacy automatically from natural-language preferences.

Project Goals

- Translate user natural-language privacy requirements into executable pipelines.
- Use an LLM-based agent to build and run privacy operators for audio, video.
- Ensure consistent behavior across modalities and live/non-live inputs.
- Provide safety, verification, and auditability.

Related / Prior Work

Peekaboo (Jin et al., 2022)

- Enforces privacy using a developer-written manifest and a fixed set of detectors/transforms.
- Limitation: Static and inflexible, it cannot create new logic or tools.
- Relevance to our work: We keep the pipeline idea (detect → transform → verify) but extend it with dynamic tool generation and LLM-based planning, eliminating the static manifest bottleneck.

1-2-3 Check (Li et al., 2024)

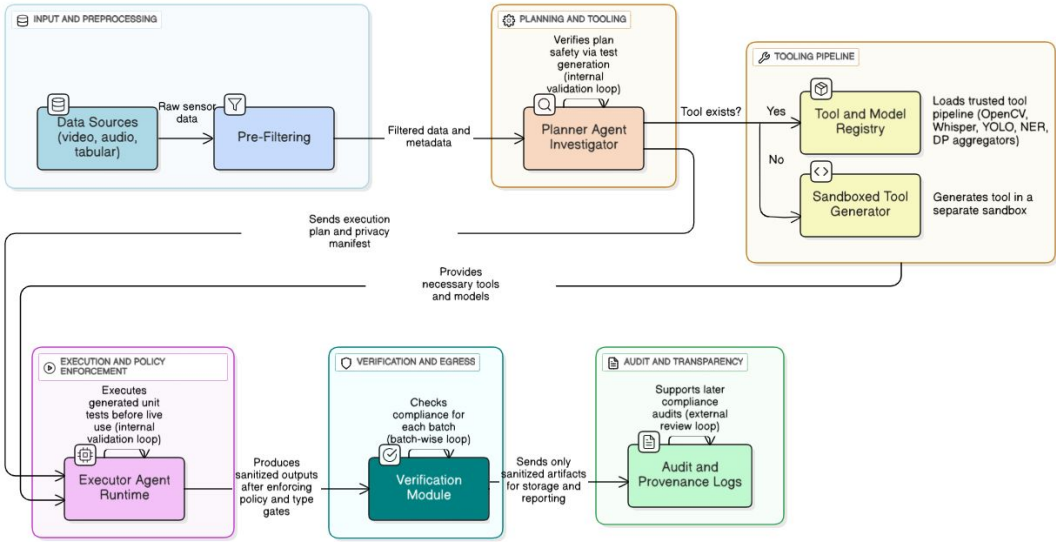
- Multi-agent system that enforces Contextual Integrity (CI) norms in text/audio.
- Uses a Checker Agent to validate outputs and reduce leakage.
- Limitation: Focuses only on contextual leakage in text/audio; no multimodal sanitization, no tool generation.
- Relevance to our work: Inspires our policy verification layer, but our system goes beyond CI checking by supporting vision + audio and generating executable sanitization tools on demand.

Our Novelty

- Autonomous tool generation when existing operators aren’t enough.
- Unified policy framework covering audio, video, and multimodal inputs.
- Dynamic verification with automatic retries and full replanning on failure.
- Full auditability via manifests and provenance logs.
- Combines LLM flexibility with strict verification, making it adaptable and trustworthy.

High-Level Approach

- Ingest raw sensor data (video/audio) and user's natural-language privacy request
- Planner Agent interprets the request and produces a manifest (execution plan)
- Checks tool registry; reuses existing tools or calls Tool Generator to create new ones
- Executor Agent runs the tools in sequence to transform/sanitize the data
- Verification Module checks the output against the privacy policy
- All steps logged for auditability



Team Contributions

Sam

- Speech processing pipeline
- Tool generation module
- Metrics module
- Audit & logging framework
- Executor runtime

Daniel

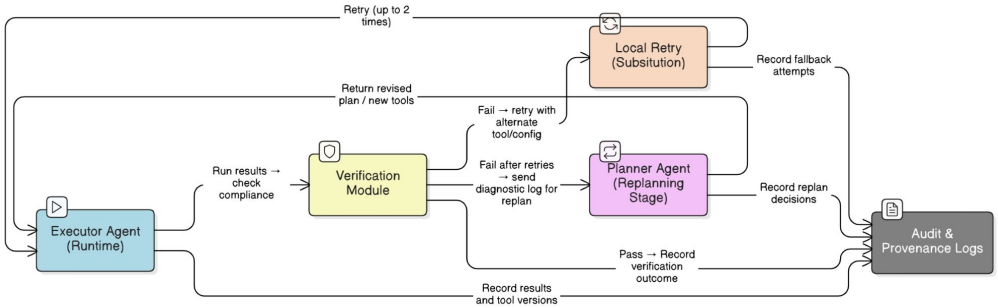
- LLM planner agent
- Face detection & blurring pipeline
- Live-streaming redaction
- Verification modules (video + audio)
- Closed-loop recovery design

Collaborative Work

- Weekly meetings
- Joint code reviews
- Shared testing & integration sessions

Algorithm - Closed-loop

- After execution, output goes through the Verification Module
- If verification passes: output is released + logged
- If verification fails: Automatic retries using fallback tools or alternative configurations; If still failing → system triggers full replanning
- Planner receives diagnostic report and updates the manifest or generates new tools
- Ensures privacy requirements are always met before data leaves the system



Algorithm - Planning

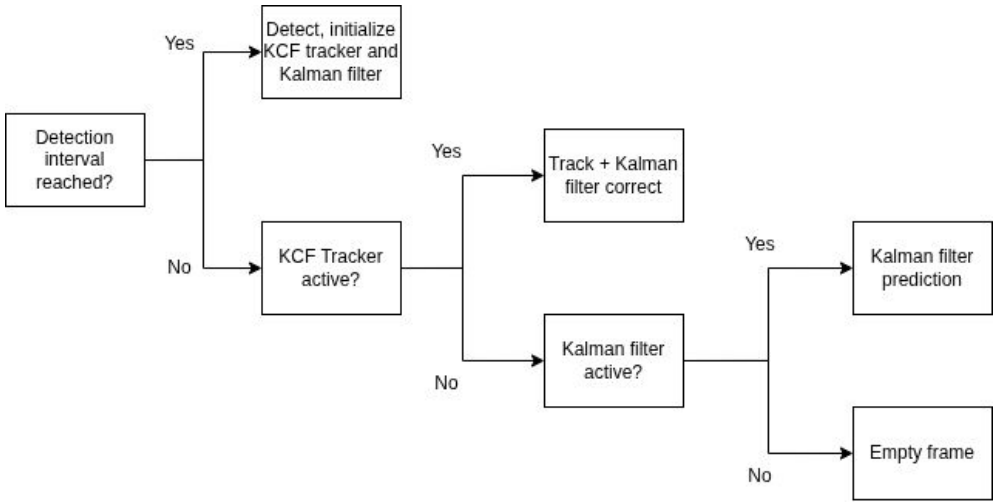
- Uses llama-3.3-70b-versatile to convert natural-language privacy requests into JSON manifests
- Chooses tools from registry (face blur, keyword mute, etc.) or proposes new tools via naming templates
- Manifest encodes a pipeline of steps with tool names + arguments
- System checks tool availability; missing tools → auto generation
- Final manifest includes metadata on generated tools and is sent to execution

User Request: “Blur faces and remove license plates”

```
{
            "pipeline": [
              {"tool": "blur_faces", "args": {}},
              {"tool": "blur_license_plates", "args": {}}
            ]
}
```

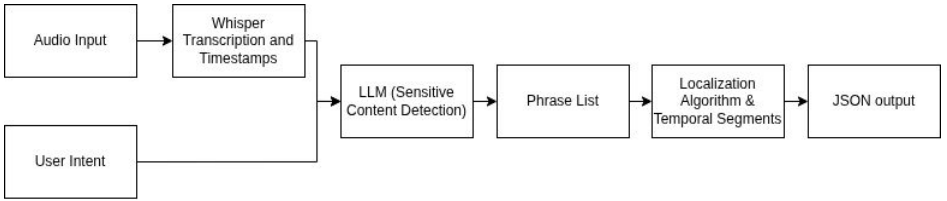
Algorithm - Face blurring

- Hybrid detection pipeline: YuNet, KCF tracker, Kalman Filter
- Cascaded strategy: detect every N frames + track in between
- Kalman model: constant velocity, 8-state filter, ~60-frame prediction cap
- Grayscale preprocessing: CLAHE + sharpening for higher contrast
- Live vs Offline:
  - Live → single-pass
  - Offline → two-pass (detect first, verify, then blur)
- Verification (offline):
  - Miss-ratio threshold (<10%)
  - Laplacian-variance blur strength (<50)



Algorithm - Speech beeping

- Whisper (base) → transcript + timestamps
- llama-3.3-70b extracts sensitive phrases from transcript based on user intent
- Sliding-window match on timestamped words with precise time intervals
- pydub mutes or inserts a beep in detected intervals
- Live streaming: FasterWhisperASR + small buffer, Real-time keyword detection before playback.
- Verification (offline):
  - Temporal integrity check (valid timestamps, no long segments, low overlap)
  - Compliance check: re-run ASR → confirm sensitive phrases no longer appear



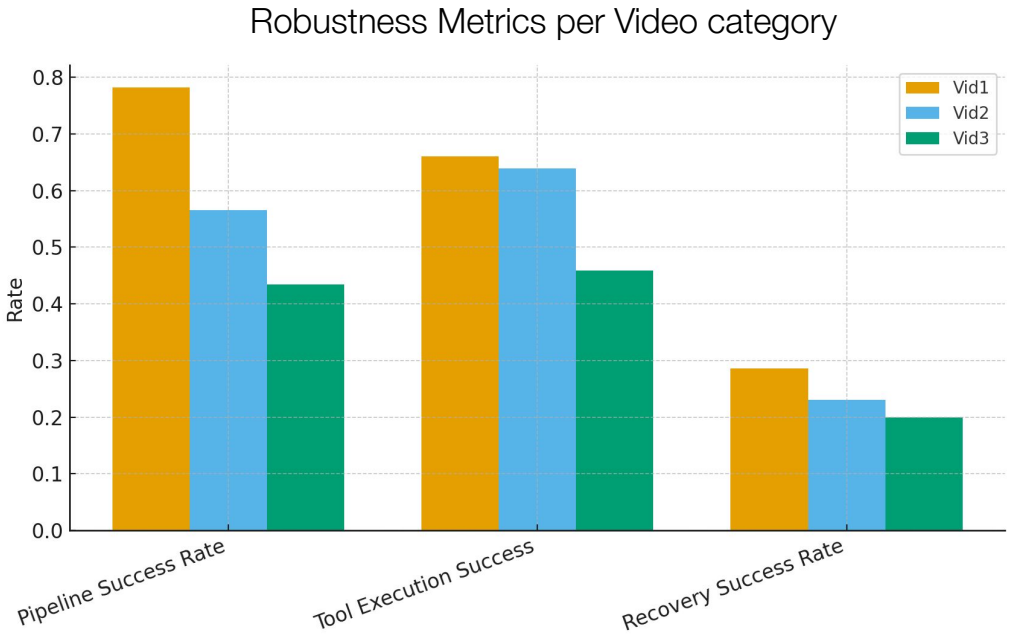
Algorithm - Tool Generation

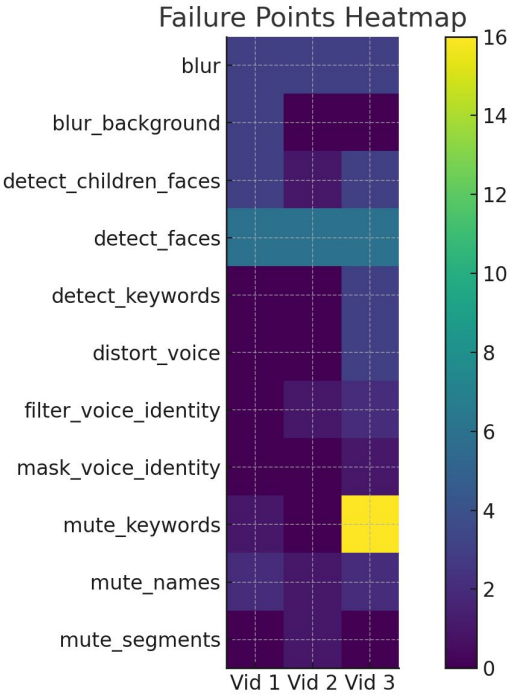
- Triggered when the Planner requests a tool not found in the registry
- Uses LLM with a specialized system prompt containing: allowed libraries whitelist, code template for a valid tool, guidelines for video/audio processing patterns. processing patterns
- LLM outputs full tool Python code
- System cleans the code and saves it under the correct category
- Tool is dynamically imported into the registry (no restart required)
- Newly generated tool becomes immediately available for execution in the pipeline

Evaluation Metrics

Category	Metrics
Correctness	Face-blur accuracy; mute/beep correctness
Robustness	Pipeline success rate; tool-generation success %; recovery success
Adaptability	Time to generate new pipeline; # of LLM steps; success rate on unseen requirements
Performance	End-to-end latency; resource usage (CPU/GPU); throughput
Security	Sandbox compliance; manifest verification; generated code safety
Auditability	Log completeness; manifest accuracy; decision traceability
Baseline Comparison	Latency overhead caused by LLM calls

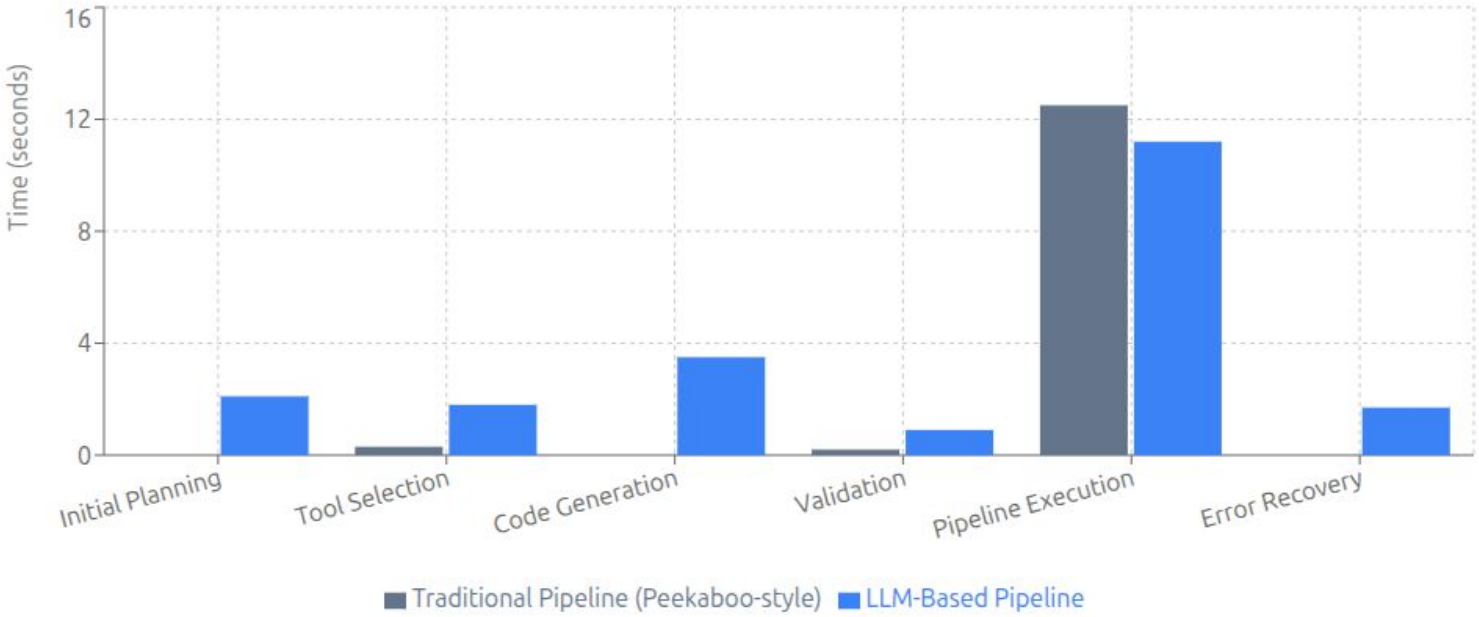
Results


Robustness Metrics per Video category
Rate
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Vid1
Vid2
Vid3
Pipeline Success Rate
Tool Execution Success
Recovery Success Rate


Failure Points Heatmap
blur
blur_background
detect_children_faces
detect_faces
detect_keywords
distort_voice
filter_voice_identity
mask_voice_identity
mute_keywords
mute_names
mute_segments
Vid 1 Vid 2 Vid 3
0
2
4
6
8
10
12
14
16

Results

Comparison of time spent in each stage between rigid traditional pipelines and adaptive LLM-based approach.



Analysis & Discussion of Results

- Overall success rate before any retry attempts is **59%**.
- When the recovery loop is triggered, an average recovery success rate of **24%** is observed. As expected, all three success rates systematically decrease as scenario complexity increases.

- In the most complex case ("Vid 3"), the mute_keywords tool struggled the most and frequently failed verification due to overlapping speech making automatic transcription more challenging.

- ~10.2s overhead compared to traditional pipelines
- More latency than traditional pipelines due to LLM calls, error recovery mechanism

Conclusions

- Built an end-to-end system where LLM agents translate natural-language privacy policies into executable pipelines.
- Achieved 69% overall success, with the retry loop recovering 24% of failed attempts.
- Strong multimodal performance:
  - Face tracking with <10% miss ratio.
  - Keyword muting/redaction across diverse privacy intents.
- Cuts implementation time from weeks of engineering → seconds of automated pipeline synthesis.

Future Work & Next Directions

Performance & Deployment

- GPU acceleration for YuNet + Whisper to reduce end-to-end latency
- Exploration of smaller/distilled models (e.g., Whisper-tiny + fine-tuning) for on-device inference
- Improved handling of multi-speaker audio via speaker diarization

Reliability & Verification

- Fine-tuning a dedicated model for privacy-tool code generation
- Human-in-the-loop verification (e.g., MTurk) to train stronger automated verifiers

Scope Expansion

- Extend beyond audio–video to arbitrary sensor modalities

References

1. Jin, H., et al. "Peekaboo: A Hub-Based Approach to Enable Transparency in Data Processing within Smart Homes." arXiv:2204.04540
2. Li, W., et al. "1-2-3 Check: Contextual Integrity-Aware Multi-Agent Privacy Reasoning." arXiv:2508.07667
3. Iravantchi, Y., et al. "PrivacyLens: On-Device PII Removal from RGB Images Using Thermal Sensing." PETS 2024
4. Zhou, J., et al. "Privacy-sensitive Objects Pixelation for Live Video Streaming." arXiv:2101.00604
5. Zhang, S., et al. "Evaluating the Efficacy of Large Language Models for Generating Fine-Grained Visual Privacy Policies in Homes." arXiv:2508.00321

Live Demo

Thank You