Merged Project Report: Automating Retail Security Workflows with Generative AI, RAG, and Vision Pipelines

Team Members:

- Michael Perry Williams (mw00066@vt.edu)
- [Partner Name] (same as previous RAG assignment)

Overview

This report presents a unified solution to automate physical security workflows in retail environments using generative AI, image segmentation, and Retrieval-Augmented Generation (RAG). Drawing from three prior efforts, the merged project integrates YOLO/Mask R-CNN segmentation, vector embedding via CLIP/BLIP, and LLM-based reasoning to transform manual surveillance monitoring into a scalable AI-driven process.

Use Case Identification

Manual Business Process:

Security personnel across retail chains manually review surveillance footage and incident logs. This includes:

- Monitoring static and video feeds.
- Identifying suspicious individuals.
- Logging incidents.

Current Inefficiencies:

1. **Slow and Reactive:** Manual review delays threat detection.
2. **Inconsistency:** Personnel-dependent judgment leads to varied results.
3. **Data Silos:** Logs and video streams lack integration.
4. **Scalability Limits:** Workforce costs scale with store count.

Automation Benefits:

- **Real-Time Flagging:** AI segments and classifies visual entities.
- **Enhanced Accuracy:** Embeddings align visual features with known threats or friendly actors.
- **Interactive UI:** Staff can query incidents (e.g., "activity at back door last night?").
- **Scalable Deployment:** Single system supports hundreds of stores.

Data Preparation

Data Types and Sources:

1. **Surveillance Frames/Images** – From security cameras (static and video).
2. **Incident Logs** – Textual reports with timestamps, locations.
3. **Captions/Annotations** – AI-generated (via BLIP) or manually added descriptions.
4. **Entity Labels** – Uniformed workers (friendly) vs masked individuals (suspicious).

Preparation Approach:

- **Segmentation**: Use YOLOv8-seg, Mask R-CNN, and Grounding DINO + SAM.
- **Embedding**: Use CLIP and BLIP to vectorize visual data and scene captions.
- **Storage**: Pinecone, FAISS, or ChromaDB for scalable retrieval.

- **Synthetic Data**: Generate rare cases (e.g., armed intruders) using Stable Diffusion + ControlNet.
- **Labeling**: Use Roboflow for dataset annotation and augmentation.

Solution Design

AI-Powered Code Generation:

- **Prompted GPT-4o** to generate:
    - Python image segmentation pipeline (YOLO + CLIP).
    - Pinecone embedding/retrieval modules.
    - Flask app interface for uploading images + querying.
    - RAG inference using OpenAI, Grok, or Cohere LLMs.

Tools Used:

- **Coding**: PyCharm (backend), CodePen (frontend prototype).
- **Libraries**: PyTorch, transformers, ultralytics, Flask, Pinecone.

Workflow:

1. Image captured or uploaded.
2. Entities segmented, embedded, and upserted.
3. LLM retrieves relevant captions/entities via RAG.
4. User receives label (e.g., "Friendly") and explanation (e.g., "USPS uniform with package").

UI Features:

- Upload images.
- Ask natural-language queries.
- Get visual + textual classification results.

Prototype and Feedback

Implementation Summary:

- YOLOCameraClassifier + MaskRCNNAnalysis for segmentation.
- `ragchat` repository extended with CLIP embeddings and RAG inference.
- Flask app + CodePen frontend for interactive query interface.

Python Backend Snippet (Simplified):

```
from ultralytics import YOLO
from transformers import CLIPProcessor, CLIPModel
from pinecone import Pinecone
from flask import Flask, request, render_template

# Load models
yolo_model = YOLO('yolov8n-seg.pt')
clip_model = CLIPModel.from_pretrained('openai/clip-vit-base-patch16')
clip_processor = CLIPProcessor.from_pretrained('openai/clip-vit-base-patch16')
pc = Pinecone(api_key='API_KEY')
index = pc.Index('surveillance-images')

app = Flask(__name__)

@app.route('/', methods=['POST'])
def analyze():
    image = request.files['image']
    # segment, embed, classify...
    return render_template('result.html', result="Friendly")
```

HTML CodePen Frontend:

```html
<form action="/" method="post" enctype="multipart/form-data">
  <input type="file" name="image">
  <input type="submit" value="Analyze">
</form>
```

Challenges:

- Pinecone API setup + quota management.
- Real-time segmentation for low-res images.
- Code alignment across Tensor formats (YOLO ↔ CLIP).

Outcome:

- Classified "Person at gate" correctly as "Friendly".
- Retrieved matching captions with 100% accuracy in small test set.
- Interface received positive user feedback for clarity and utility.

Scaling and Future Development

- **Kafka + Streaming**: Ingest video in real time.
- **Live Feed Integration**: Add dashboard to monitor active feeds.
- **Predictive Analysis**: Use historical incident vectors for forecasting threats.
- **Enterprise Deployment**: Cloud-scale backend with Lambda, ECS.

Broader Applicability & Learnings

- This solution generalizes to:

    - **Healthcare**: Detect patient risks (falls, unauthorized entries).
    - **Logistics**: Monitor package theft or delivery success.
    - **Smart Cities**: Identify accidents, traffic anomalies, or public hazards.
- Generative AI reduced dev time by rapidly generating working code blocks.
- RAG with visual embeddings allowed nuanced context-aware classification.
- Simple coding tools (CodePen, Flask) made the system accessible and easy to test.

References & Resources

- Deloitte (2024). *AI in physical security: Trends and insights*.
- Radford et al. (2021). *Learning Transferable Visual Models from Natural Language Supervision*.

GitHub Repos Used:

- https://github.com/mpwusr/YOLOCameraClassifier
- https://github.com/mpwusr/MaskRCNNAnalysis
- https://github.com/mpwusr/CNNCIFAR10ImageClassifier
- https://github.com/mpwusr/ragchat