Uniform & Safety Gear Detection (RTSP → YOLO → Kafka/CMS)

Repo này chạy **nhận diện đồng phục + mũ bảo hộ** trên luồng RTSP, **vẽ overlay (ROI + bbox)** và **bắn event cảnh báo về CMS qua Kafka**. Ngoài ra có **FastAPI** để public ảnh (full/crop) cho CMS lấy về hiển thị và **MediaMTX** (tuỳ chọn) để phát lại RTSP output nhằm quan sát realtime.

> Trạng thái: code chạy được theo flow hiện tại, nhưng còn nhiều "hard-code" (Kafka broker, base URL, đường dẫn lưu ảnh, credential RTSP). Nếu dùng production thì cần dọn cấu hình + bảo mật.

Tính năng chính

- **2 model YOLO (Ultralytics)**
  - `models/hat_final.pt`: mũ bảo hộ (ví dụ: `Hardhat` / `NO-Hardhat`)
  - `models/costume_final.pt`: đồng phục (ví dụ: `uniform` / `non-uniform`)
- **ROI filter**: bỏ qua detection ở vùng trên ảnh (theo `--roi_thresh`)
- **Ensemble logic (hat vs uniform)**: dùng `Process` để thêm "logic score" dựa trên tương quan vị trí mũ và người/đồng phục
- **Debounce theo frame + cooldown**:
  - chỉ bắn event khi vi phạm kéo dài `--thresh_frames` frame
  - và cách nhau ít nhất `--cooldown_s` giây
- **Lưu ảnh + public URL**
  - lưu ảnh full + crop theo ngày: `deploy/image/YYYYMMDD/{uniform|hat}/...jpg`
  - public qua FastAPI endpoint `/image/{date}/{violation_type}/{filename}`
- **Kafka event**
  - gửi JSON vào Kafka (topic hiện tại trong code: `webhook.event`, `uniform_natmin`)

Luồng chạy chuẩn (3 bước)

1) Public ảnh bằng FastAPI

Cách A (đứng ở root repo):

```
uvicorn deploy.public_image:app --host 0.0.0.0 --port 8322
```

Cách B (cd vào folder deploy):

```
cd deploy
uvicorn public_image:app --host 0.0.0.0 --port 8322
```

Endpoint mẫu:

```
GET /image/20260108/uniform/123_full.jpg
GET /image/20260108/hat/123_target.jpg
```

> Lưu ý: `deploy/public_image.py` đang hard-code `BASE_FOLDER`. Bạn phải sửa `BASE_FOLDER` cho đúng đường dẫn `.../deploy/image`.

---

2) (Tuỳ chọn) Mở MediaMTX để phát RTSP output

MediaMTX dùng để xem lại luồng infer realtime (overlay bbox/ROI). Repo đã có binary `./mediamtx` và config `mediamtx.yml`.

```
chmod +x ./mediamtx
./mediamtx mediamtx.yml
```

Port RTSP trong `mediamtx.yml` hiện là `8555` (`rtspAddress: :8555`).
Trong code default `--output_rtsp` đang là `rtsp://127.0.0.1:8554/live` (lệch port).
Bạn chọn 1 trong 2:

- **Option 1 (khuyến nghị):** chạy infer với output đúng 8555

  ```
  python infer_origin.py --output_rtsp rtsp://127.0.0.1:8555/live
  ```

- **Option 2:** sửa `mediamtx.yml` về `:8554` nếu bạn muốn giữ 8554.

Xem stream (local):

```
ffplay -rtsp_transport tcp rtsp://127.0.0.1:8555/live
# hoặc VLC
```

Nếu server remote và bạn muốn xem ở máy local:

```
ssh -L 8555:127.0.0.1:8555 user@<server-ip>
ffplay -rtsp_transport tcp rtsp://127.0.0.1:8555/live
```

---

3) Chạy inference chính (RTSP camera → detect → Kafka + RTSP output)

Script chính: `infer_origin.py`

```
python infer_origin.py   --ip_cam <CAMERA_IP>   --port <CAMERA_PORT>   --path <RTSP_PATH>
```

Các tham số quan trọng: - `--conf_uniform`, `--conf_hat`: ngưỡng confidence cho YOLO - `--roi_thresh`: tỉ lệ ROI (0.3 nghĩa là bỏ vùng phía trên 30% ảnh) - `--w_model`, `--w_logic`, `--final_thresh`: trọng số + ngưỡng final score cho ensemble logic - `--thresh_frames`, `--cooldown_s`: debounce/cooldown event

---

Cài đặt môi trường

Yêu cầu tối thiểu

- Python 3.9+ (khuyến nghị 3.10/3.11)
- FFmpeg có trong PATH (để RTSP writer/reader chạy ổn)
- GPU CUDA (tuỳ chọn nhưng khuyến nghị cho YOLO)

Install (gợi ý)

Repo chưa có `requirements.txt` nên bạn tự pin lại sau. Tối thiểu thường cần:

```
pip install ultralytics opencv-python numpy torch fastapi uvicorn confluent-kafka
```

Nếu server không có GUI:

```
export QT_QPA_PLATFORM=offscreen
```

Chọn GPU:

```
export CUDA_VISIBLE_DEVICES=0
```

Output & Event schema

Ảnh lưu trên disk

- Full frame: `deploy/image/YYYYMMDD/{uniform|hat}/{event_id}_full.jpg`
- Crop target (nếu crop được): `deploy/image/YYYYMMDD/{uniform|hat}/{event_id}_target.jpg`

Log file (fallback)

File: `deploy/log/log_test.jsonl`
Trong `infer_origin.py` đang vừa bắn Kafka vừa append JSONL (để audit/debug).

Kafka topics

Trong `infer_origin.py`: - `producer.send_json("webhook.event", event)`
- `producer.send_json("uniform_natmin", event)`

Event payload (rút gọn)

- `triggerEventId`: id event
- `policyName`, `actionContent`, `policyType`, `msgSource`
- `deviceId`
- `bkImageUrl`, `triggerImgUrl`, `targetImgUrl`
- `attributes`: list (conf + key/value)

  Quan trọng: `IMG_BASE_URL` + `BASE_FOLDER` phải khớp nhau (ảnh lưu ở đâu thì FastAPI serve đúng ở đó).

Cảnh báo (bảo mật + vận hành)

- Bạn đang **hard-code credential RTSP** và **Kafka broker public** trong code. Đây là rủi ro lớn:
  - chuyển sang `.env` / secret manager
  - tuyệt đối không public repo kèm credential
- FastAPI đang public ảnh **không auth**. Nếu mở ra internet là tự sát:
  - đặt reverse proxy + auth
  - giới hạn IP / VPN / private network
- MediaMTX: nếu mở RTSP ra internet thì phải cấu hình auth/ACL.

Directory structure

```
nnminh322-uniformdetection/
  infer_origin.py            # main pipeline: RTSP -> infer -> RTSP out + Kafka event
  kafka_controller.py        # Kafka producer/consumer helpers (confluent-kafka)
  process.py                 # ensemble logic + ROI filtering
  utils.py                   # hardened RTSP reader/writer via FFmpeg
  deploy/
     public_image.py        # FastAPI serve ảnh
     log/
         log_test.jsonl
  models/
     costume_final.pt
     hat_final.pt
  mediamtx                   # MediaMTX binary
  mediamtx.yml               # MediaMTX config
```

Troubleshooting nhanh

- **Không mở được RTSP input**
  - thử đổi transport TCP/UDP (`OPENCV_FFMPEG_CAPTURE_OPTIONS`)
  - kiểm tra camera có cho phép stream cùng lúc nhiều client không
  - test nhanh: `ffplay -rtsp_transport tcp "<rtsp_url>"`
- **Không xem được RTSP output**
  - kiểm tra port MediaMTX (8555 vs 8554)
  - kiểm tra firewall / ssh tunnel
- **Kafka không nhận**
  - kiểm tra `bootstrap.servers` đúng không
  - broker có mở port không
  - topic có tồn tại không (hoặc broker tự tạo topic)