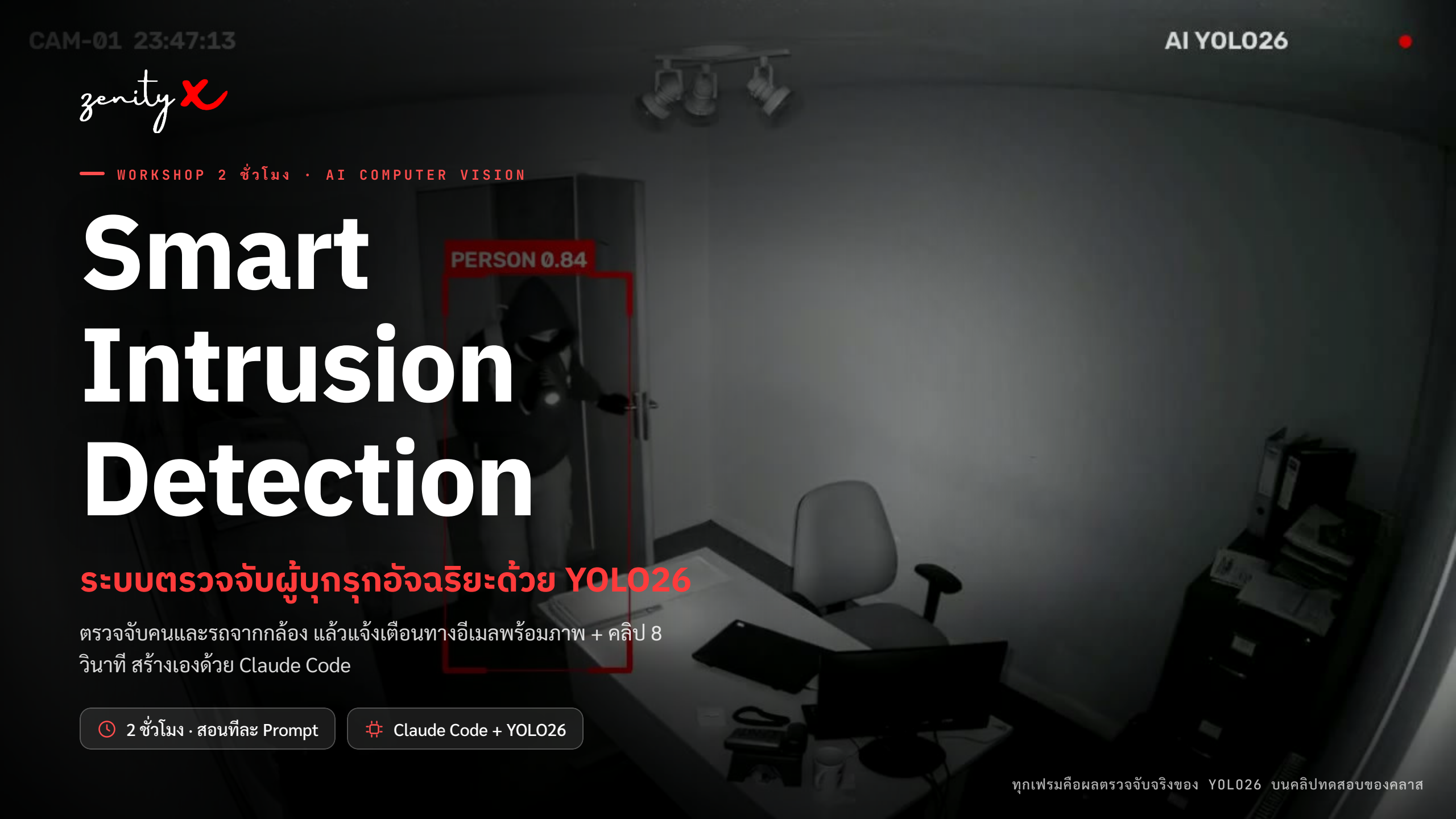


zenity x

WORKSHOP 2 ชั่วโมง · AI COMPUTER VISION

Smart Intrusion Detection

ระบบตรวจจับผู้บุกรุกอัจฉริยะด้วย YOLO26

ตรวจจับคนและรถจากกล้อง แล้วแจ้งเตือนทางอีเมลพร้อมภาพ + คลิป 8 วินาที สร้างเองด้วย Claude Code

2 ชั่วโมง · สอนทีละ Prompt

Claude Code + YOLO26

ทุกเฟรมคือผลตรวจจับจริงของ YOLO26 บนคลิปทดสอบของคลาส



ผู้สอน

เอกรัตน์ ใจชนะ

นิว · AI Instructor, ZenityX

วิทยาการคอมพิวเตอร์ · สายงานระบบตรวจจับด้วย AI และระบบสารสนเทศ

ชนะเลิศ

นำเสนอบทความวิจัยแบบบรรยาย ระดับดีเด่นอันดับ 1

Track Information Technology · AUCC 2026 การประชุมวิชาการระดับปริญญาตรีด้านคอมพิวเตอร์ภูมิภาคเอเชีย ครั้งที่ 14

ตรงกับวันนี้

ชนะเลิศ

Oral Presentation ด้านนวัตกรรม "ระบบกล้องวงจรปิดตรวจจับอัจฉริยะ"

ประกวดผลงานสหกิจศึกษาและการศึกษาเชิงบูรณาการกับการทำงาน (CWIE) ดีเด่น เครือข่าย CWIE ภาคเหนือตอนล่าง

ดีเด่น

Best of Track

Track Information Technology · การนำเสนอผลงานวิจัย

ผลงานตีพิมพ์ · RJST 2026

Home Visit Information System for Student Behavior Management — Rattanakosin Journal of Science and Technology ปีที่ 8 ฉบับที่ 2 หน้า 152–166

ปลายทางของวันนี้

ตรวจเจอผู้บุกรุก → อีเมลถึงมือถือทันที พร้อมภาพ + คลิป 8 วินาที



[แจ้งเตือน] พบ คน 1 — CAM-01 02:14:37

จาก ZenityX Intrusion Alert · ถึง เจ้าของร้าน, ยาม

INTRUSION DETECTED

ตรวจพบ	**คน 1** (ความมั่นใจ 0.87)
เวลา	12 พ.ย. 2026 · 02:14:37
กล้อง	CAM-01 · ห้องทำงาน



snapshot.jpg

clip.mp4 · 8 วินาที

RUN OF SHOW · 120 นาที

ตารางสอน 2 ชั่วโมง · ทฤษฎีสั้น ลงมือทำทีละเฟส

ความรู้ + เตรียมเครื่อง · 0:00–0:35

0:00–0:10	**เปิดคลาส** แนะนำผู้สอน · ดูผลลัพธ์ปลายทาง
0:10–0:25	**AI Detection & YOLO26** คืออะไร · ทำไม · ทำงานอย่างไร · กันขโมยอย่างไร
0:25–0:35	**เครื่องมือ + Setup** Claude Code · Python · OpenCV · YOLO26 — Prompt S0
1:50–2:00	**รันระบบเต็ม + ต่อยอด** Prompt RUN · Bonus กล้องจริง / แผงควบคุมเว็บ · Q&A

ลงมือสร้าง 5 เฟส · 0:35–1:50

0:35–0:50	**Phase 1 · ตรวจจับคน** ไฟล์วิดีโอ / webcam / กล้อง RTSP
0:50–1:00	**Phase 2 · เลือกขนาดโมเดล** เทียบ YOLO26 n / s / m · ลอง seg + pose
1:00–1:15	**Phase 3 · ช่วงเวลา + โซน** 22:00–06:00 · วาดโซนเฝ้าระวัง
1:15–1:25	**Phase 4 · กรองคลาส** คน รถยนต์ มอเตอร์ไซค์ · กันแจ้งเตือนหลอก
1:25–1:50	**Phase 5 · หลักฐาน + อีเมล** คลิป 3+5 วินาที · Google App Password

ตารางเวลาเป็นแนวทาง ผู้สอนอาจปรับเวลาแต่ละช่วงให้เหมาะกับความคืบหน้าของผู้เรียน โดยเนื้อหาและผลลัพธ์หลักยังครบถ้วน

ผลลัพธ์หลังเรียน

จบ 2 ชั่วโมง ได้ระบบจริงกลับบ้าน

เข้าใจ AI Detection

อธิบายได้ว่า Object Detection คืออะไร และ YOLO26 ทำงานอย่างไร

สั่ง AI สร้างระบบ

ใช้ Claude Code สร้างระบบด้วย Python + OpenCV + YOLO26 ทีละเฟส

เลือกโมเดลเป็น

เลือกขนาด YOLO26 ให้เหมาะกับเครื่องและงาน

คุมการเฝ้าระวัง

กำหนดช่วงเวลา โซน และคลาสที่ต้องการตรวจจับ

แจ้งเตือนพร้อมหลักฐาน

อีเมลพร้อมภาพ + คลิป 8 วินาที ผ่าน Gmail App Password

ต่อยอดใช้งานจริง

ต่อกล้องวงจรปิด RTSP และแผงควบคุมบนเว็บ (Bonus)

01

0:10–0:25 · ทฤษฎี 15 นาที

AI Detection & YOLO26

AI มองเห็นได้อย่างไร · YOLO26 คืออะไร ทำไมต้องใช้ ทำงานอย่างไร และช่วยกันขโมยได้อย่างไร

ปัญหาของกล้องวงจรปิดทั่วไป

กล้อง บันทึก ได้ แต่ ไม่มีใครดู



คนเฝ้าจอหลายจอพร้อมกันตลอดคืนไม่ไหว — ส่วนใหญ่รู้ตัวเมื่อเสียหายไปแล้ว แล้วค่อยย้อนดูภาพ

ทำไมต้องเป็น AI

Motion Detection เตือน ทุกอย่างที่ขยับ · AI รู้ว่า อะไร ขยับ

กล้องตรวจจับการเคลื่อนไหวแบบเดิม

ใบไม้ เงา ฝน แมว ก็เตือน
เตือนหลอกบ่อยจนคนเลิกสนใจ

ไม่รู้ว่าสิ่งที่ขยับคืออะไร
แยกคนกับรถกับสัตว์ไม่ได้

ส่งแค่ข้อความสั้น ๆ
ต้องเปิดแอปย้อนดูเอง

AI Detection (วันนี้)

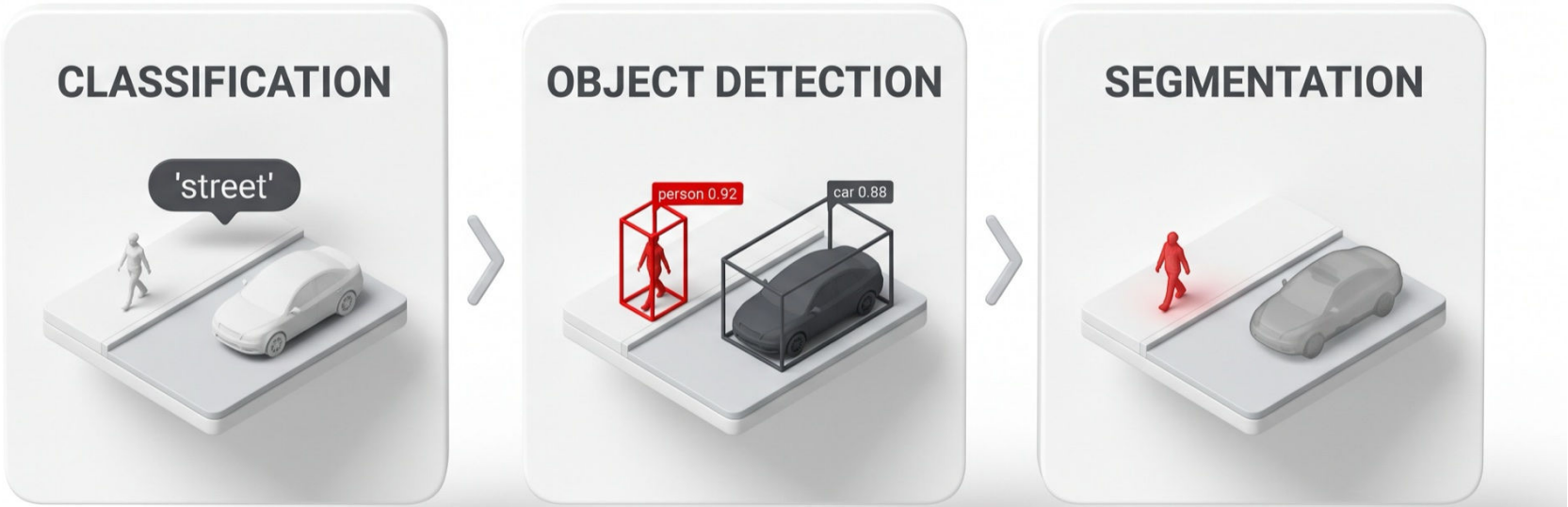
แจ้งเฉพาะคน / รถ ที่เราเลือก
80 คลาสให้เลือก ตัดสิ่งรบกวนทิ้ง

เฉพาะในโซน และเฉพาะเวลาที่ไม่ควรมีใคร
ลดการเตือนหลอกได้มาก

ส่งหลักฐานถึงมือถือทันที
ภาพ + คลิปก่อนเหตุ 3 วิ หลังเหตุ 5 วิ

AI DETECTION คืออะไร

Computer Vision 3 ระดับ — วันนี้ใช้ Object Detection



Classification บอกว่า "ภาพนี้คืออะไร" · Object Detection บอกว่า "มีอะไร อยู่ตรงไหน กี่ชิ้น" · Segmentation แยกรูปร่างทีละพิกเซล

ผลลัพธ์ของ DETECTION

1 วัตถุ = กรอบ + คลาส + ความมั่นใจ



1 **กรอบ (Bounding Box)**
ตำแหน่ง x, y และขนาดกว้าง × สูง ของวัตถุในภาพ

2 **คลาส (Class)**
วัตถุนี้คืออะไร เช่น person, car, motorcycle

3 **ความมั่นใจ (Confidence)**
0–1 ยิ่งสูงยิ่งมั่นใจ เช่น 0.90 = มั่นใจ 90%

4 **ทำซ้ำทุกเฟรม**
วิดีโอคือภาพนิ่ง 25–30 ภาพต่อวินาที AI ตรวจทุกภาพแบบ real-time

YOLO26 คืออะไร

You Only Look Once — มองภาพครั้งเดียว ทายทุกวัตถุพร้อมกัน

รุ่นล่าสุดของ Ultralytics

ออกมกราคม 2026 ออกแบบให้เร็วบนเครื่องทั่วไปและอุปกรณ์ edge

รู้จัก 80 คลาสทันที

ฝึกมากับชุดข้อมูล COCO: คน รถยนต์ มอเตอร์ไซค์ รถบรรทุก สุนัข ฯลฯ ไม่ต้องเทรนเอง

ทำได้หลายงาน

Detection · Segmentation · Pose · Classification · OBB (กรอบเอียง)

5 ขนาดให้เลือก

n · s · m · l · x — เล็กเร็ว ใหญ่แม่น เลือกตามเครื่อง

```
from ultralytics import YOLO
model = YOLO("yolo26n.pt") # ดาวน์โหลดให้เองครั้งแรก
results = model(frame) # ได้กรอบ + คลาส + confidence ทันที
```

ทำไมต้อง YOLO26

เร็ว เบา แม่น และ นำไปใช้ง่าย



End-to-end ไม่ต้องกรองกรอบซ้ำ (NMS) · เร็วบน CPU · จับวัตถุเล็กดีขึ้น · export ได้หลายแพลตฟอร์ม

ทำไมต้อง YOLO26 · รายละเอียด

4 จุดเด่นที่เหมาะกับ กล้องวงจรปิด

End-to-end ไม่ต้อง NMS

ทายกรอบสุดท้ายได้เลย ไม่ต้องมีขั้นกรองกรอบซ้ำ ขั้นตอนน้อยลง เร็วขึ้น นำไปใช้ง่าย

เร็วบน CPU

งานวิจัยของผู้พัฒนารายงานว่า YOLO26n เร็วกว่า YOLO11n บน CPU สูงสุดราว 43% — ไม่มีการ์ดจอก็รันได้

จับวัตถุเล็กดีขึ้น

เทคนิค STAL ช่วยให้เจอคนตัวเล็กในภาพกล้องมุมสูงได้ดีขึ้น

Export ได้ทุกที่

หัวทำนายเรียบง่าย (ตัด DFL) ส่งออกเป็น ONNX / TensorRT / CoreML / OpenVINO ได้ง่าย

YOLO26 ทำงานอย่างไร

ภาพเข้า → Backbone → Neck → Head → ได้ผลทันที



Backbone ดึงลักษณะเด่น (ขอบ รูปร่าง พื้นผิว) · Neck รวมลักษณะหลายขนาดให้เจอทั้งของใหญ่และของเล็ก · Head ทายกรอบ + คลาส + confidence

YOLO26 · 5 ขนาด

เล็ก = เร็ว · ใหญ่ = แม่น · กล้องวงจรปิดเริ่มที่ n

รุ่น	ความแม่น mAP	CPU (ms/ภาพ)	พารามิเตอร์
YOLO26n	40.9	38.9	2.4 M
YOLO26s	48.6	87.2	9.5 M
YOLO26m	53.1	220.0	20.4 M
YOLO26l	55.0	286.2	24.8 M
YOLO26x	57.5	525.8	55.7 M

36 FPS YOLO26n บน Apple M1 ในคลาส — ลื่นพอสำหรับกล้องสด

21 FPS YOLO26s แม่นขึ้น ยังใช้ real-time ได้

13 FPS YOLO26m เหมาะกับเครื่องที่มีการ์ดจอ

ตัวเลข mAP / CPU / พารามิเตอร์: ข้อมูลทางการของ Ultralytics (COCO 640 px) · FPS: วัดจริงบน MacBook M1 ด้วยคลิปของคลาส

ช่วยกันขโมยอย่างไร

แจ้งเฉพาะ คน/รถ ในโซน · ในเวลาที่ไม่ควรมีใคร



บ้านกลางคืน · ร้านค้าหลังปิด · โกดัง · ลานจอดรถ · ห้อง server · แปลงเกษตร — แมวนอกโซนไม่เตือน คนในโซนเตือนทันที

02

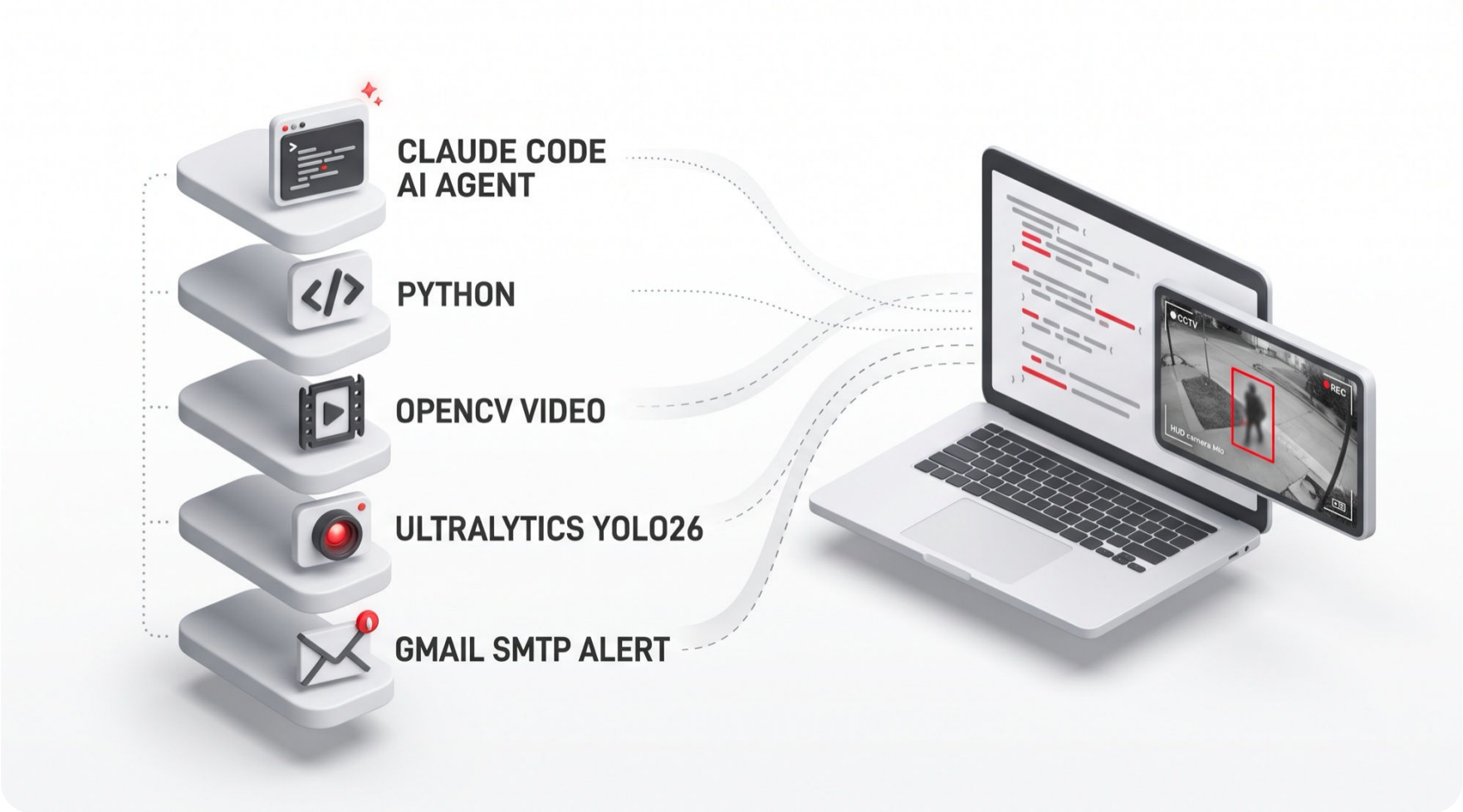
0:25-0:35 · เตรียมโปรเจกต์

เครื่องมือ + Setup

เครื่องมือที่ใช้วันนี้คืออะไร ทำไมถึงเลือก แล้วเตรียมโปรเจกต์ด้วย Prompt แรก

เครื่องมือวันนี้

AI Agent สั่งงานเป็นภาษาไทย + ชุดเครื่องมือ AI มาตรฐาน



Claude Code เขียนและรันโค้ดให้ · Python + OpenCV จัดการวิดีโอ · Ultralytics YOLO26 คือสมองที่มองเห็น · Gmail ส่งแจ้งเตือน

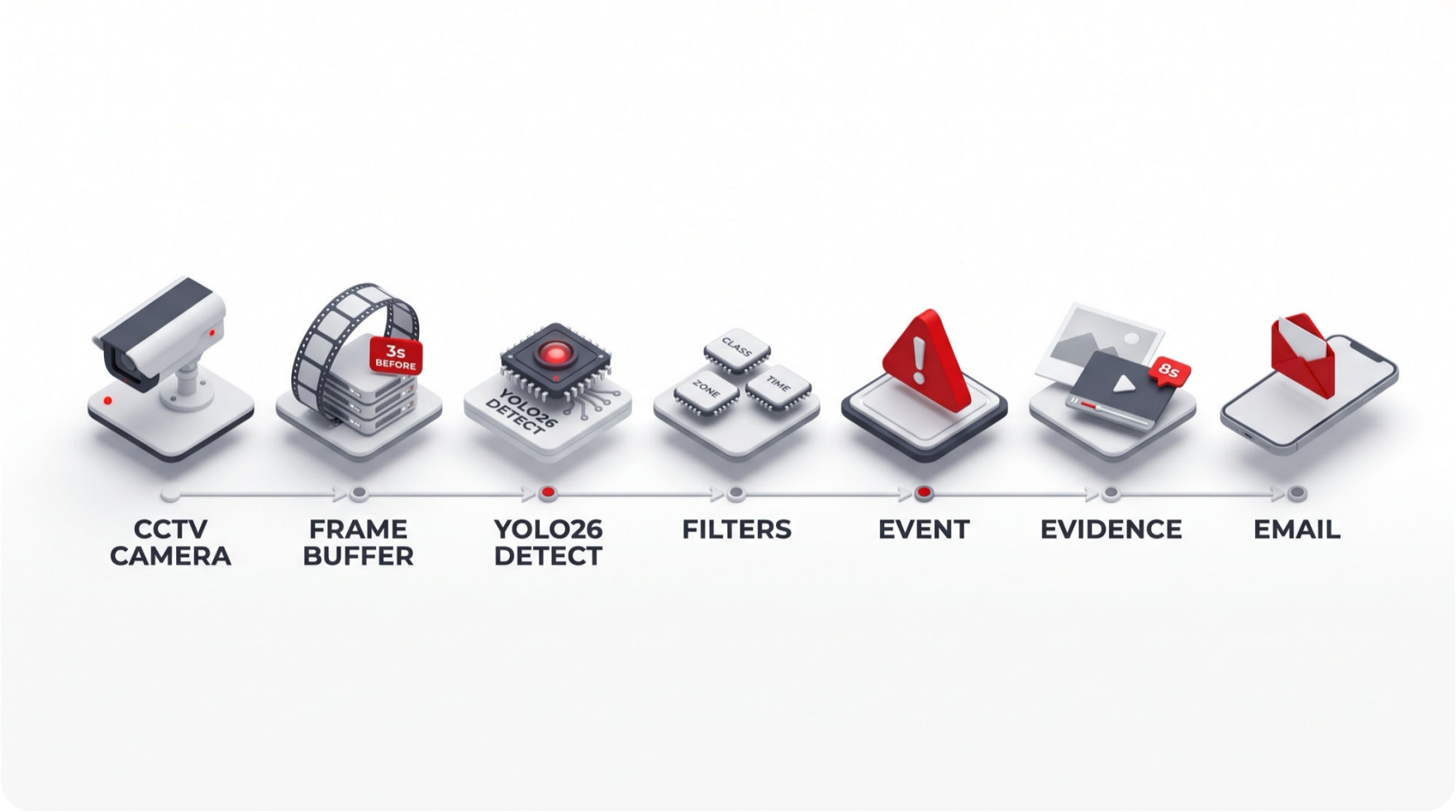
เครื่องมือแต่ละตัว · ทำไมต้องใช้

Claude Code = วิศวกรที่เราสั่งงาน · ไลบรารี = กล่องเครื่องมือของมัน

เครื่องมือ	บทบาท	ทำไมใช้
Claude Code	AI Agent เขียนโค้ด รันโค้ด แก้ error ในโฟลเดอร์โปรเจกต์	สั่งเป็นภาษาไทย ไม่ต้องเขียนโค้ดเอง ได้ระบบจริงใน 2 ชั่วโมง
Python 3.10+	ภาษาหลักของงาน AI	ไลบรารี AI ครบที่สุด
OpenCV	"ตาและมือ" อ่านวิดีโอ/กล้อง วาดกรอบ บันทึกคลิป	มาตรฐานงานภาพ ต่อ webcam และ RTSP ได้
Ultralytics + YOLO26	"สมอง" ตรวจจับวัตถุ — ติดตั้งเหมือนปลั๊กอิน	โมเดลล่าสุด เร็ว ใช้แค่ 3 บรรทัด
imageio-ffmpeg	แปลงคลิปเป็น H.264	เปิดได้ทุกเครื่องและมือถือ ไม่ต้องลง ffmpeg เอง
Gmail + App Password	"ผู้ส่งสาร" ส่งอีเมลแจ้งเตือน	ฟรี ใช้บัญชีที่มีอยู่ ปลอดภัยกว่าใช้รหัสจริง
.env (python-dotenv)	เก็บรหัสแยกจากโค้ด	รหัสไม่หลุดไปกับโค้ด

ภาพรวมระบบ

กล้อง → ตรวจจับ → กรอง → หลักฐาน → อีเมล



buffer เก็บภาพย้อนหลังตลอดเวลา · YOLO26 ตรวจทุกเฟรม · กรองด้วยคลาส โซน เวลา และต้องเจอติดกัน 3 เฟรม · ส่งภาพ + คลิป 8 วิ · พัก 60 วิ

SETUP · PROMPT S0

เปิด Claude Code ในโฟลเดอร์โปรเจกต์ แล้ววาง Prompt แรก

1. **สร้างโฟลเดอร์ yolo26-detection**
   แตก zip คลิปทดสอบ ให้ได้ yolo26-detection/videos/
2. **เปิด Terminal ในโฟลเดอร์นั้น**
   พิมพ์ `claude` แล้วกด Enter
3. **วาง Prompt S0**
   Claude Code สร้าง .venv ติดตั้งไลบรารี ตรวจคลิป
4. **กดอนุญาตเมื่อถูกถาม**
   ทุกคำสั่งที่ AI จะรัน จะขออนุญาตก่อนเสมอ

ดาวน์โหลดก่อนเรียน

คลิปทดสอบ 7 คลิป

แตกไฟล์แล้ววางโฟลเดอร์ videos/ ไว้ในโฟลเดอร์โปรเจกต์ · 22 MB

ดาวน์โหลด .zip

SETUP · S0

```
ช่วยเตรียมโปรเจกต์ "ระบบตรวจจับผู้บุกรุกด้วย YOLO26" ในโฟลเดอร์นี้ให้หน่อย
1. ตรวจว่าเครื่องมี Python 3.10 ขึ้นไปหรือยัง ถ้ายังไม่มีให้บอกวิธีติดตั้งสำหรับเครื่องฉัน (อย่าติดตั้งโปรแกรมระดับเครื่องเองโดยไม่ถาม)
2. สร้าง virtual environment ชื่อ .venv แล้วติดตั้ง ultralytics, opencv-python, pyyaml, python-dotenv, imageio-ffmpeg
3. สร้าง requirements.txt และ .gitignore (ไม่เอา .venv, .env, events/, *.pt)
4. ดูว่าในโฟลเดอร์ videos/ มีคลิปอะไรบ้าง สรุปเป็นตาราง: ชื่อไฟล์ · ความละเอียด · ความยาว · fps
5. สร้างไฟล์ CLAUDE.md บันทึกว่าโปรเจกต์นี้คืออะไร ให้ตอบเป็นภาษาไทย และรัน Python ผ่าน .venv ทุกครั้ง
เสร็จแล้วสรุปว่าพร้อมเริ่มเฟส 1 หรือยัง
```

P1

PHASE 1 · 0:35–0:50

ตรวจจับคนด้วย YOLO26

สร้าง detector.py ตัวแรก — กรอบแดงรอบคน จำนวนคน และ FPS รองรับไฟล์วิดีโอ webcam และกล้องวงจรปิด

PHASE 1 · PROMPT P1

เริ่มจาก "คน" คลาสเดียวก่อน

PHASE 1

สร้างไฟล์ detector.py ตรวจจับ "คน" จากวิดีโอด้วย YOLO26
- ใช้ ultralytics โหลดโมเดล yolo26n.pt (ระบบดาวน์โหลดให้เองครั้งแรก)
- อ่านค่าจาก config.yaml: source (เริ่มที่ videos/01_door_entry.mp4), model, confidence 0.5
- source รองรับ 3 แบบ: path ไฟล์วิดีโอ · ตัวเลข 0 = webcam · rtsp://... = กล้องวงจรปิด
- ตรวจเฉพาะคลาส person
- วาดกรอบสีแดงรอบคน พร้อมป้าย เช่น person 0.87 และแสดงจำนวนคนที่เจอ + FPS มุมซ้ายบน
- เปิดหน้าต่างแสดงผล กด q เพื่อออก
- เลือก device อัตโนมัติ: cuda > mps (Mac) > cpu
- เพิ่มตัวเลือก --no-window ให้รันแบบไม่เปิดหน้าต่าง (พิมพ์ผลเป็นข้อความแทน) และ --source ไว้เปลี่ยนแหล่งภาพจากคำสั่ง
ทดสอบเองด้วย --no-window ก่อน แล้วบอกคำสั่งให้ฉันรันเองแบบเปิดหน้าต่าง
สุดท้ายอธิบายโค้ดทีละส่วนสั้น ๆ ให้คนเพิ่งเริ่มเข้าใจ

ไฟล์วิดีโอ

source: videos/01_door_entry.mp4 — ใช้ในคลาส

Webcam

source: 0 — กล้องของโน้ตบุ๊ก

กล้องวงจรปิด

source: rtsp://... — ต่อกล้องจริงใน Bonus

รันเองแบบเปิดหน้าต่าง: `.venv/bin/python detector.py` · กด **q** เพื่อออก

PHASE 1 · ผลลัพธ์ที่ควรเห็น

กรอบแดงรอบคน + จำนวนคน + FPS มุมจอ



ภาพจริงจากระบบ · คลิป 11_cctv_office_day · YOLO26n บน Apple M1

P2

PHASE 2 · 0:50–1:00

เลือกขนาดโมเดล YOLO26

สลับ n / s / m แล้ววัดผลจริงบนเครื่องตัวเอง — เร็วแค่ไหน แม่นแค่ไหน ควรใช้รุ่นไหน

PHASE 2 · PROMPT P2

วัดจริงบนเครื่องตัวเอง แล้วให้ AI แนะนำ

PHASE 2

YOLO26 มีหลายขนาด (n, s, m, l, x) ช่วยให้ระบบสลับโมเดลได้ และเปรียบเทียบให้ดูหน่อย
- เพิ่มตัวเลือก --model ให้สลับ yolo26n.pt / yolo26s.pt / yolo26m.pt ได้จากคำสั่ง (ค่าเริ่มต้นอ่านจาก config.yaml)
- สร้าง benchmark.py รันโมเดล n, s, m กับ videos/04_office_wide.mp4 อย่างละ 150 เฟรม
  วัด FPS เฉลี่ย · จำนวนคนที่เจอต่อเฟรมเฉลี่ย · confidence เฉลี่ย แล้วสรุปเป็นตาราง
- บันทึกภาพเปรียบเทียบเฟรมเดียวกัน (เลือกเฟรมที่มีคน) ของทั้ง 3 โมเดลเรียงต่อกันเป็น benchmark.jpg
- แนะนำว่าเครื่องของฉันควรใช้ขนาดไหนสำหรับกล้องวงจรปิดแบบ real-time พร้อมเหตุผล

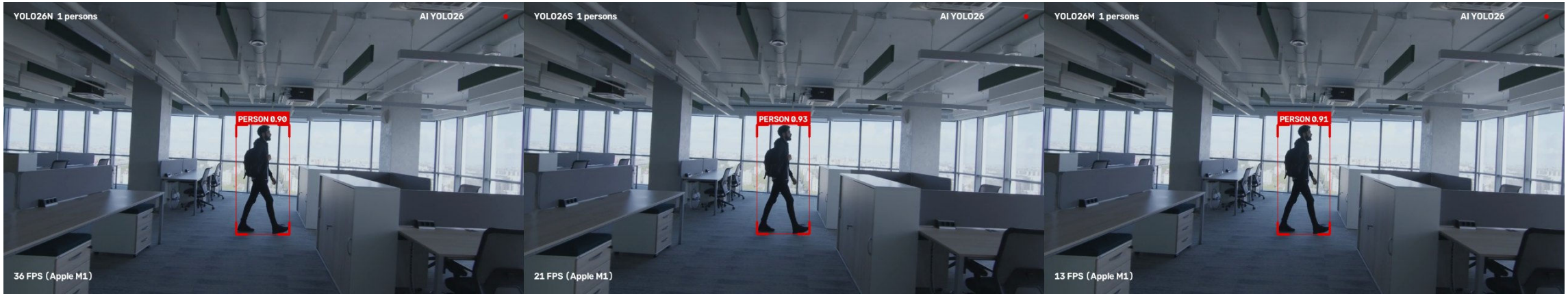
ผลที่วัดได้ในคลาส (M1)

รุ่น	FPS	ใช้เมื่อ
n	36	กล้องสด · CPU
s	21	ต้องการแม่นขึ้น
m	13	มีการ์ดจอ

โมเดล s และ m ดาวน์โหลดเพิ่มรวมราว 70 MB ในครั้งแรก

PHASE 2 · ผลลัพธ์ BENCHMARK.JPG

เฟรมเดียวกัน 3 โมเดล — เจอเหมือนกัน แต่เร็วต่างกัน 3 เท่า



ภาพจริงจากระบบ · คลิป 04_office_wide · ซ้ายไปขวา YOLO26n 36 FPS · YOLO26s 21 FPS · YOLO26m 13 FPS

PHASE 2 · เสริม · PROMPT P2B

YOLO26 ทำได้มากกว่าตีกรอบ: Segmentation + Pose

PHASE 2 · เสริม

YOLO26 ไม่ได้ทำได้แค่ตีกรอบ ลองให้ดูอีก 2 งาน
- สร้าง demo_tasks.py รัน yolo26n-seg.pt (แยกรูปร่างวัตถุทีละพิกเซล) และ yolo26n-pose.pt (จับจุดโครงร่างร่างกาย)
  กับ videos/02_hallway_door.mp4 เลือกเฟรมที่เห็นคนเต็มตัวชัดที่สุด แล้วบันทึกเป็น task_seg.jpg และ task_pose.jpg
- อธิบายสั้น ๆ ว่าแต่ละงานเอาไปใช้กับระบบรักษาความปลอดภัยแบบไหนได้

Segmentation

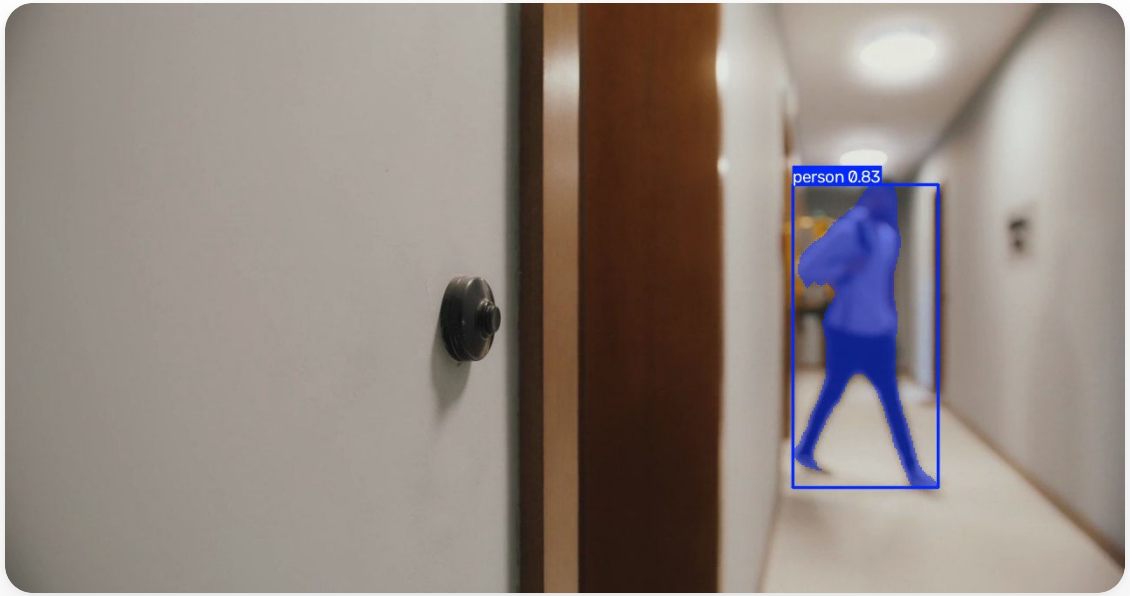
แยกรูปร่างวัตถุทีละพิกเซล — รู้ขอบเขตจริงของคน ไม่ใช่แค่กรอบสี่เหลี่ยม

Pose

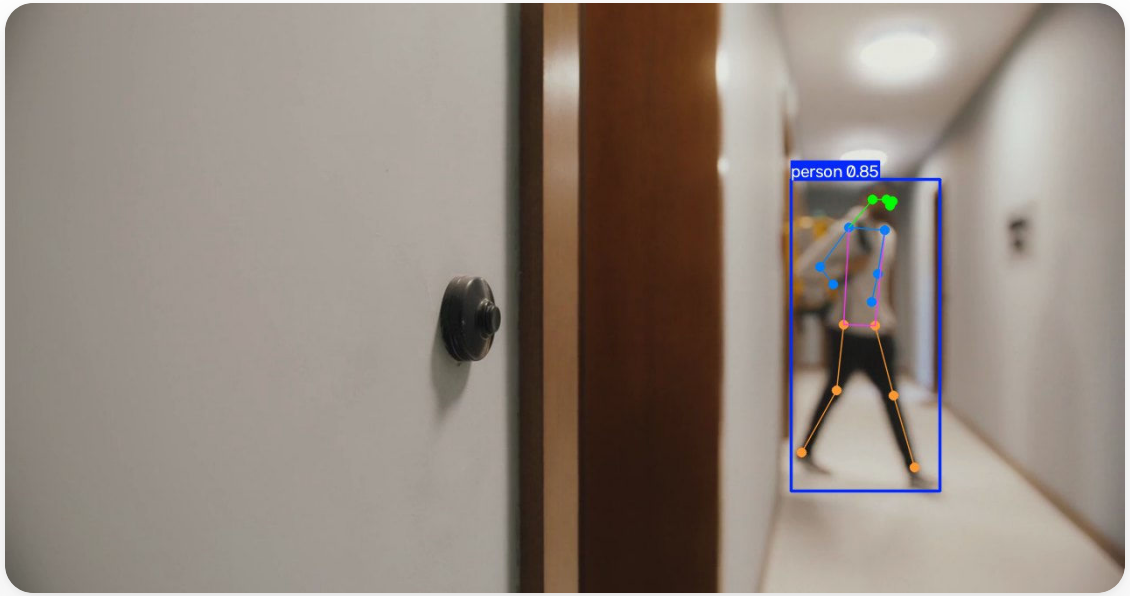
จับจุดโครงร่าง 17 จุด — ต่อยอดตรวจท่าทาง เช่น ปีนรั้ว ล้ม ยกมือ

PHASE 2 · เสริม · ผลลัพธ์

คลิปเดียวกัน 2 งาน: รูปร่าง และ โครงร่าง



Segmentation · yolo26n-seg.pt



Pose · yolo26n-pose.pt

ภาพจริงจากระบบ · คลิป 02_hallway_door

P3

PHASE 3 · 1:00–1:15

ช่วงเวลา + โซนตรวจจับ

เฝ้าระวังเฉพาะเวลาที่ไม่ควรมีใคร (เช่น 22:00–06:00) และเฉพาะพื้นที่ที่สนใจ

PHASE 3 · แนวคิด

ARMED เฉพาะช่วงเวลา · นับเฉพาะ "จุดเท้า" ที่อยู่ในโซน



ช่วงเวลาข้ามเที่ยงคืนได้

22:00–06:00 · นอกช่วงขึ้น DISARMED ยังแสดงภาพแต่ไม่นับ

วาดโซนด้วยเมาส์

--draw-zone คลิกรอบพื้นที่ Enter บันทึก · แยกโซนตามกล้อง

ใช้จุดเท้า ไม่ใช่กลางกรอบ

กึ่งกลางขอบล่างของกรอบ = ตำแหน่งที่ยืนจริงบนพื้น

จำลองเวลาได้

--now 23:00 ทดสอบตอนกลางวันได้ ไม่ต้องแก้ config

PHASE 3 · PROMPT P3

เพิ่ม schedule + zones ใน config.yaml

PHASE 3

เพิ่มการกำหนด "ช่วงเวลา" และ "โซน" ที่ระบบจะเฝ้าระวัง
ช่วงเวลา
- config.yaml เพิ่ม schedule: enabled, start "22:00", end "06:00" (ต้องรองรับช่วงที่ข้ามเที่ยงคืน)
- ในช่วงเวลา ขึ้นป้าย ARMED สีแดง · นอกช่วงเวลา ขึ้น DISARMED สีเทา ยังแสดงภาพแต่ไม่นับเป็นเหตุการณ์
- เพิ่มตัวเลือก --now HH:MM ไว้จำลองเวลา แล้วทดสอบให้ดูทั้งกรณีอยู่ในช่วง และอยู่นอกช่วง
โซน
- เพิ่มตัวเลือก --draw-zone เปิดเฟรมแรกให้คลิกเมาส์วาดรูปหลายเหลี่ยม (Enter = บันทึก, c = ล้าง, Esc = ยกเลิก)
  บันทึกพิกัดแบบสัดส่วน 0-1 ลง config.yaml แยกตามคลิป/กล้อง (zones: {ชื่อไฟล์: [...]}) ใช้ได้กับทุกความละเอียด
  แก้เฉพาะส่วนโซน อย่าลบคอมเมนต์อื่นใน config
- นับเฉพาะวัตถุที่ "จุดเท้า" (กึ่งกลางขอบล่างของกรอบ) อยู่ในโซน · ในโซนกรอบสีแดง นอกโซนกรอบสีเทา
- วาดเส้นโซนบนภาพ ถ้าคลิป/กล้องไหนยังไม่มีโซนให้ถือว่าทั้งภาพเป็นโซน
บอกคำสั่งให้ฉันลองวาดโซนเองกับ videos/13_cctv_carport_motorcycle.mp4

1 **วาง Prompt P3**

รอ AI แก้ detector.py และ config.yaml

2 **วาดโซนเอง**

`detector.py --draw-zone --source videos/13_cctv_carport_motorcycle.mp4`

3 **ทดสอบเวลา**

`--now 23:00` = ARMED · `--now 14:00` = DISARMED

P4

PHASE 4 · 1:15–1:25

กรองคลาสที่ต้องการ

เลือกจาก 80 คลาสของ YOLO26 — คน รถยนต์ มอเตอร์ไซค์ — และกันแจ้งเตือนหลอก

PHASE 4 · PROMPT P4

เลือกคลาสด้วย ชื่อ · ต้องเจอติดกัน 3 เฟรม ถึงนับเป็นเหตุการณ์

PHASE 4

ตอนนี้ตรวจแค่คน ช่วยให้เลือกคลาสได้จาก 80 คลาสของ YOLO26 (ชุดข้อมูล COCO)
- config.yaml: classes: [person, car, motorcycle] ใช้ชื่อคลาสแทนตัวเลข ถ้าพิมพ์ผิดให้แนะนำชื่อที่ใกล้เคียง
- เพิ่มตัวเลือก --list-classes แสดงทั้ง 80 คลาสพร้อมชื่อภาษาไทย
- แต่ละคลาสใช้สีกรอบต่างกัน และนับจำนวนแยกตามคลาสบนจอ
- กันแจ้งเตือนหลอก: ต้องเจอติดกันอย่างน้อย min_consecutive_frames: 3 เฟรม ถึงนับเป็น "เหตุการณ์" แล้วพักการนับ cooldown_seconds: 60 วินาที (คลิปทดสอบยาวราว 12 วินาที จึงได้ 1 เหตุการณ์ต่อการรัน)
- เมื่อเกิดเหตุการณ์ ขึ้นข้อความ INTRUSION DETECTED บนจอ และพิมพ์ log: เวลา · สิ่งที่เจอ · confidence
- ทดสอบกับ videos/14_cctv_parking_car.mp4 (รถยนต์) และ videos/13_cctv_carport_motorcycle.mp4 (มอเตอร์ไซค์)
  ลานจอดมีรถจอดอยู่แล้ว ให้ใช้โซนเฉพาะทางเข้า ไม่งั้นจะแจ้งเตือนตั้งแต่เฟรมแรก
  ถ้าตอนนี้อยู่นอกช่วงเวลา ให้ทดสอบด้วย --now 23:00 (ไม่ต้องแก้ config)

ตัวอย่างจาก 80 คลาส

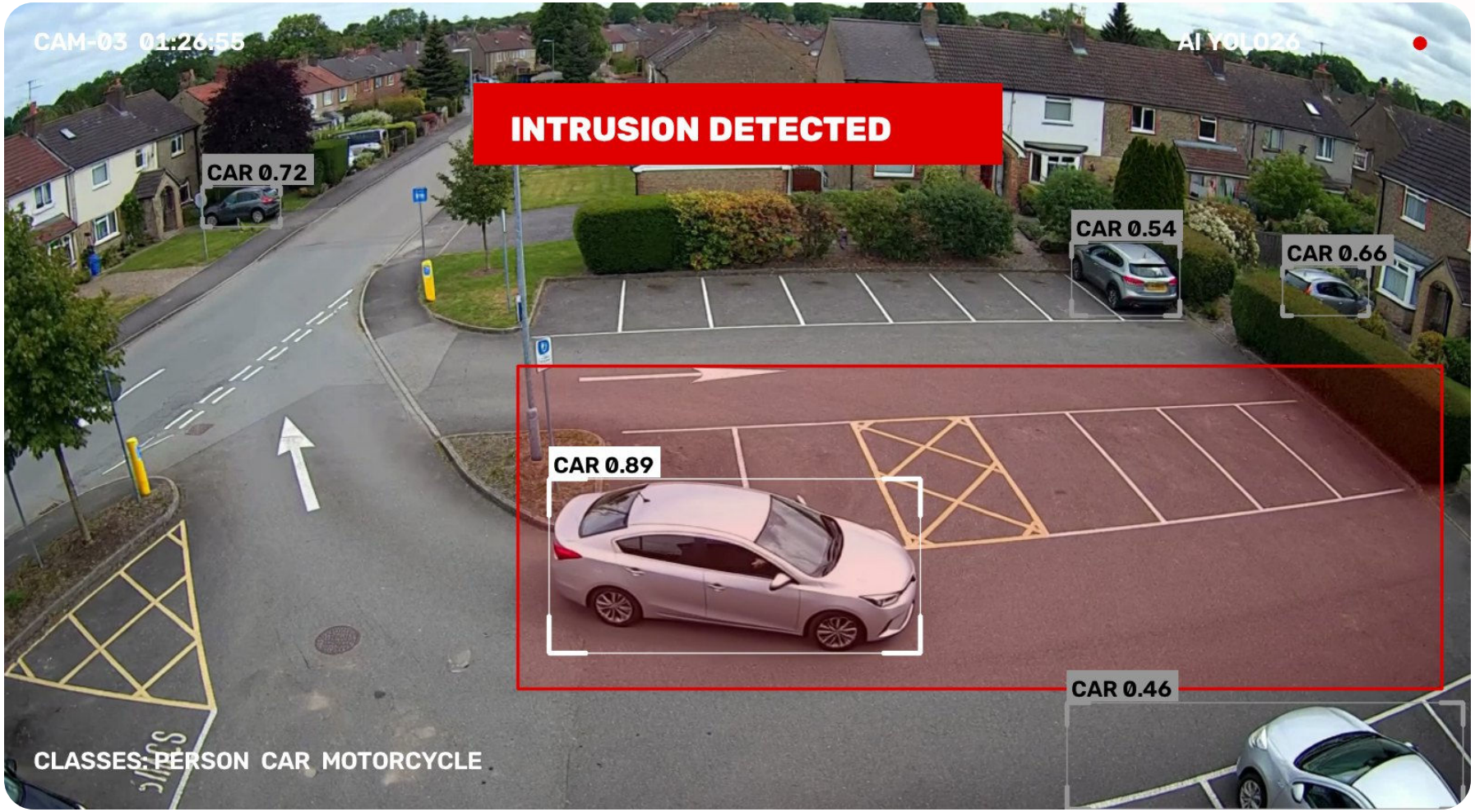
person · car · motorcycle · truck · bicycle · bus · backpack · handbag · suitcase · dog · cat · laptop

กันแจ้งเตือนหลอก

เจอติดกัน 3 เฟรม = ไม่ใช่ภาพหลอกแวบเดียว · พัก 60 วินาที หลังเตือน = ไม่ส่งอีเมลรัว

PHASE 4 · ผลลัพธ์ที่ควรเห็น

สีกรอบแยกคลาส · รถในโซนทางเข้า → INTRUSION DETECTED



ภาพจริงจากระบบ · คลิป 14_cctv_parking_car · รถที่จอดนอกโซนเป็นกรอบเทา ไม่นับ

P5

PHASE 5 · 1:25–1:50

หลักฐาน + แจ้งเตือนอีเมล

เก็บภาพ + คลิปก่อนเหตุ 3 วินาที หลังเหตุ 5 วินาที แล้วส่งอีเมลผ่าน Gmail App Password

PHASE 5 (1/2) · PROMPT P5A

buffer ย้อนหลังตลอดเวลา → คลิป 8 วินาที ทุกเหตุการณ์

PHASE 5 · หลักฐาน

เมื่อเกิดเหตุการณ์ ให้ระบบเก็บหลักฐานอัตโนมัติ
- เก็บเฟรมย้อนหลังไว้ในหน่วยความจำตลอดเวลา (buffer) เพื่อให้มีภาพก่อนเกิดเหตุ
- เมื่อเกิดเหตุ บันทึก snapshot.jpg และ clip.mp4 (ทั้งคู่มีกรอบตรวจจับ) ยาว 8 วินาที
  = ก่อนเกิดเหตุ 3 วินาที + หลังเกิดเหตุ 5 วินาที (ตั้งค่า pre_seconds / post_seconds ใน config.yaml)
- ใส่วันเวลาและชื่อกล้องบนคลิป แล้วแปลงเป็น H.264 ด้วย ffmpeg จากแพ็กเกจ imageio-ffmpeg ให้เปิดได้ทุกเครื่องและในมือถือ
- ถ้า source เป็นไฟล์วิดีโอ ให้เล่นตามความเร็วจริงและใช้เวลาของวิดีโอ คลิปจะได้ยาวถูกต้อง
- เก็บไว้ที่ events/[วันที่_เวลา]/ พร้อม event.json (เวลา · สิ่งที่เจอ · confidence · ชื่อกล้อง)
- การบันทึกคลิปต้องไม่ทำให้การตรวจจับสะดุด (ใช้ thread แยก)

ทดสอบกับ videos/11_cctv_office_day.mp4 (ถ้าอยู่นอกช่วงเวลา ใช้ --now 23:00) แล้วเช็คให้ดูว่าคลิปยาวประมาณ 8 วินาทีจริง

EVENT

3 วินาที
ก่อนเกิดเหตุ (จาก buffer)

5 วินาที
หลังเกิดเหตุ

ได้ไฟล์ต่อเหตุการณ์

events/วันที่_เวลา/
snapshot.jpg · clip.mp4 (H.264) · event.json

ทดสอบจริงแล้ว: คลิปยาว 8.00 วินาที เปิดได้ในมือถือ

PHASE 5 · ทำเองทีละขั้น

สร้าง Google App Password (รหัส 16 ตัวสำหรับโปรแกรม)

1. **เปิด 2-Step Verification**
   `myaccount.google.com/security` → การยืนยันแบบ 2 ขั้นตอน
2. **เปิดหน้า App Passwords**
   `myaccount.google.com/apppasswords` (ล็อกอินอีกครั้งถ้าถูกถาม)
3. **ตั้งชื่อแอป แล้วกดสร้าง**
   เช่น YOLO26 Alert
4. **คัดลอกรหัส 16 ตัว**
   แสดงครั้งเดียว · เว้นวรรคหรือไม่ก็ได้
5. **พิมพ์ลงไฟล์ .env เอง**
   SMTP_USER · SMTP_PASSWORD · EMAIL_TO · EMAIL_SEND=true
6. **ทดสอบ**
   `detector.py --test-email` แล้วเช็คกล่องจดหมาย

⚠ ห้ามวางรหัสในแชท AI
พิมพ์ลงไฟล์ .env ด้วยตัวเองเท่านั้น ไฟล์นี้ไม่ถูกส่งขึ้น git

ไม่เห็นเมนู App Password?
ยังไม่เปิด 2-Step หรือเป็นบัญชีองค์กร (Google Workspace) ที่แอดมินปิดไว้

✕ ใช้รหัส Gmail ปกติไม่ได้
ต้องเป็น App Password · ยกเลิกได้ทุกเมื่อที่หน้าเดิม

PHASE 5 (2/2) · PROMPT P5B

ส่งอีเมล HTML ภาษาไทย ภาพในตัว + ไฟล์แนบ

PHASE 5 · อีเมล

```
เมื่อเก็บหลักฐานเสร็จ ให้ส่งอีเมลแจ้งเตือนผ่าน Gmail
- ใช้ smtp.gmail.com พอร์ต 465 (SSL) อ่านค่าจากไฟล์ .env:
  SMTP_USER · SMTP_PASSWORD (App Password 16 ตัว) · EMAIL_TO (หลายคนคั่นด้วยจุลภาค) ·
EMAIL_SEND=false/true
- สร้าง .env.example ให้ด้วย ห้ามใส่รหัสจริงในโค้ด และห้ามแสดงรหัสในแชท
- หัวเรื่อง เช่น [แจ้งเตือน] พบ คน 1 · รถยนต์ 1 — CAM-01 22:15:08
- เนื้อหา HTML ภาษาไทย: สิ่งที่ตรวจพบ · เวลา · ชื่อกล้อง/สถานที่ · ภาพ snapshot แสดงในตัวอีเมล
- แนบไฟล์ snapshot.jpg และ clip.mp4
- ถ้า EMAIL_SEND=false ให้บันทึกเป็น email.eml ในโฟลเดอร์เหตุการณ์แทน (ไว้ทดสอบโดยไม่ส่งจริง)
- ส่งอีเมลใน thread แยก ถ้าส่งไม่สำเร็จให้บอกสาเหตุเป็นภาษาไทย (เช่น รหัสผิด ยังไม่เปิด 2-Step) แต่การ
ตรวจจับต้องทำงานต่อ
- เพิ่มตัวเลือก --test-email ส่งอีเมลทดสอบ 1 ฉบับ
ทดสอบแบบ EMAIL_SEND=false ให้ดูก่อน ฉันจะใส่รหัสใน .env เอง แล้วบอกขั้นตอนที่ฉันต้องทำต่อ
```

- **ทดสอบก่อนด้วย EMAIL_SEND=false**
  ได้ไฟล์ email.eml ในโฟลเดอร์เหตุการณ์ เปิดดูหน้าตาได้
- **ส่งจริงด้วย EMAIL_SEND=true**
  Claude Code จะขออนุญาตก่อนรัน — กดอนุญาต
- **ส่งไม่ผ่าน ระบบไม่หยุด**
  บอกสาเหตุเป็นภาษาไทย แต่ยังตรวจจับต่อ

PHASE 5 · ผลลัพธ์ที่ควรเห็น

อีเมลแจ้งเตือนพร้อม snapshot + คลิป 8 วินาที



[แจ้งเตือน] พบ คน 1 — CAM-01 02:14:37

จาก ZenityX Intrusion Alert · ถึง เจ้าของร้าน, ยาม

INTRUSION DETECTED

ตรวจพบ	**คน 1** (ความมั่นใจ 0.87)
เวลา	12 พ.ย. 2026 · 02:14:37
กล้อง	CAM-01 · ห้องทำงาน



snapshot.jpg

clip.mp4 · 8 วินาที

GO

1:50–2:00 · PROMPT RUN + BONUS

รันระบบเต็ม + ต่อยอด

รวมทุกเฟสกับคลิปกลางคืน แล้วต่อยอดสู่กล้องจริงและแผงควบคุมบนเว็บ

รันระบบเต็ม · PROMPT RUN

ทุกอย่างทำงานพร้อมกัน — ช่วงเวลา · โซน · คลาส · หลักฐาน · อีเมล

RUN

```
รันระบบเต็มกับ videos/12_cctv_office_night_ir.mp4 โดยใช้ทุกอย่างที่ทำมา (ช่วงเวลา · โซน · คลาส · หลักฐาน · อีเมล)
ถ้าตอนนี้อยู่นอกช่วงเวลาที่ตั้งไว้ ให้ใช้ --now 23:00 · ถ้าคลิปนี้ยังไม่มีโซน ให้ใช้ทั้งภาพ
สรุปผล: เกิดเหตุการณ์กี่ครั้ง · ไฟล์อยู่ที่ไหน · ส่งอีเมลสำเร็จหรือไม่
แล้วอัปเดต CLAUDE.md ว่าระบบทำอะไรได้บ้าง และรันด้วยคำสั่งอะไร
```

- **เกิดเหตุการณ์ และมีโฟลเดอร์ใน events/**
  snapshot.jpg · clip.mp4 · event.json · email
- **อีเมลเข้ากล่องจดหมาย**
  หัวเรื่องไทย ภาพในตัว ไฟล์แนบเปิดได้
- **CLAUDE.md อัปเดตแล้ว**
  เปิด session ใหม่ AI ก็รู้ว่าระบบทำอะไรได้

BONUS · ทำต่อเองได้

ต่อ กล้องจริง และทำ แผงควบคุมบนเว็บ

BONUS · กล้อง RTSP

ฉันจะต่อระบบกับกล้องจริง ช่วยเตรียมให้หน่อย
- อธิบายรูปแบบ URL RTSP ของกล้องยอดนิยม (Hikvision, Dahua, TP-Link Tapo) แบบ main stream และ sub stream
  และแนะนำว่าควรใช้ sub stream เพราะอะไร
- ให้ config.yaml อ่าน source จาก .env (CAMERA_URL) เพื่อไม่ให้รหัสกล้องอยู่ในไฟล์ config
- ถ้ากล้องหลุด ให้ต่อใหม่อัตโนมัติ แต่ถ้าต่อไม่ได้ตั้งแต่แรกให้แจ้ง error ทันที
- ทดสอบกับ webcam ของเครื่องนี้ (source: 0) ให้ดู
กล้องของฉัน: [ยี่ห้อ] · IP [192.168.1.xx] · user [admin] (รหัสผ่านฉันจะใส่ใน .env เอง)

BONUS · แผงควบคุมเว็บ

สร้างแผงควบคุมบนเว็บด้วย Flask (webapp.py) ที่ใช้ detector.py ตัวเดิม
- เลือกคลิป/webcam/RTSP · เลือกคลาสจากรายการ · เลือกขนาดโมเดล · ปรับ confidence
- ดูภาพสดพร้อมกรอบตรวจจับ (MJPEG stream) · ปุ่มเริ่ม/หยุด
- รายการเหตุการณ์ล่าสุด กดดูภาพ คลิป และตัวอย่างอีเมลได้
- ดีไซน์เรียบแบบหน้าจอห้องควบคุมกล้อง ใช้ไอคอน SVG ห้ามใช้อีโมจิ รองรับมือถือ
เปิดที่ http://127.0.0.1:8000 แล้วบอกวิธีใช้

ผู้สอนโชว์แผงควบคุมที่ทำเสร็จแล้วเป็นตัวอย่าง

แก้ปัญหาที่พบบ่อย

ติดตรงไหน ดูตารางนี้ก่อน หรือถาม Claude Code ได้เลย

อาการ	สาเหตุ	วิธีแก้
ModuleNotFoundError: ultralytics	รันด้วย Python นอก .venv	ใช้ .venv/bin/python (Windows .venv\Scripts\python)
FPS ต่ำมาก	โมเดลใหญ่บน CPU	ใช้ yolo26n · ลด imgsz เป็น 480
เตือนหลอกบ่อย	confidence ต่ำ / ไม่มีโซน	confidence 0.6 · วาดโซน · เพิ่มเป็น 5 เฟรม
ไม่มีเหตุการณ์เลย	อยู่นอกช่วงเวลา (DISARMED)	รันด้วย --now 23:00
แจ้งเตือนตั้งแต่เฟรมแรก	มีรถจอดอยู่ในภาพแล้ว	วาดโซนเฉพาะทางเข้า
535 Username and Password not accepted	ใช้รหัส Gmail ปกติ / พิมพ์ผิด	สร้าง App Password ใหม่ ใส่ใน .env
webcam เปิดไม่ได้บน Mac	ยังไม่ให้สิทธิ์กล้อง	System Settings → Privacy → Camera
RTSP ต่อไม่ได้	URL/รหัสผิด หรือคนละวง LAN	ทดสอบ URL ใน VLC ก่อน · ใช้ sub stream

zenityx

สรุปเนื้อหา · ถาม-ตอบ

Q&A

ขอบคุณครับ

ทุกคำถามยินดีตอบ — ทั้งการติดตั้งใช้งานจริง และการต่อยอดระบบตรวจจับของคุณ

โทร
062-262-9644

อีเมล
admin@zenityxai.com

