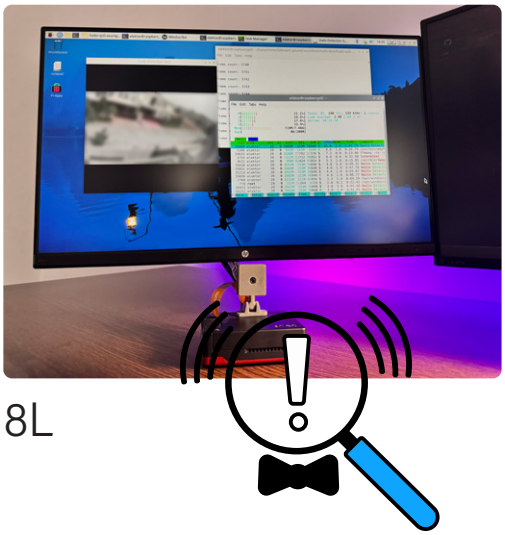
le système de sécurité IA AlertAlfred



Basé sur un Raspberry Pi 5 et le module Hailo 8L

Saad Imtiaz (Elektor)

AlertAlfred est un système de sécurité alimenté par l'IA et construit autour d'un Raspberry Pi 5 et du module Hailo 8L. Il est conçu pour identifier les individus en temps réel dans un flux de vidéo de surveillance. Ce projet montre comment configurer le système, capturer des images lorsque la présence est détectée et envoyer des alertes instantanées via Telegram - tout en assurant un traitement local des données pour préserver la confidentialité.

La sécurité domestique évolue rapidement grâce à l'IA, rendant les solutions plus abordables et accessibles que jamais. Le projet AlertAlfred vise à transformer un Raspberry Pi 5 en un hub de surveillance complet, alimenté par l'IA, capable de détecter les intrus dans un flux vidéo et de vous alerter en temps réel. Le système ne se contente pas de capturer les images des personnes détectées, mais vous envoie également des notifications immédiates via Telegram.

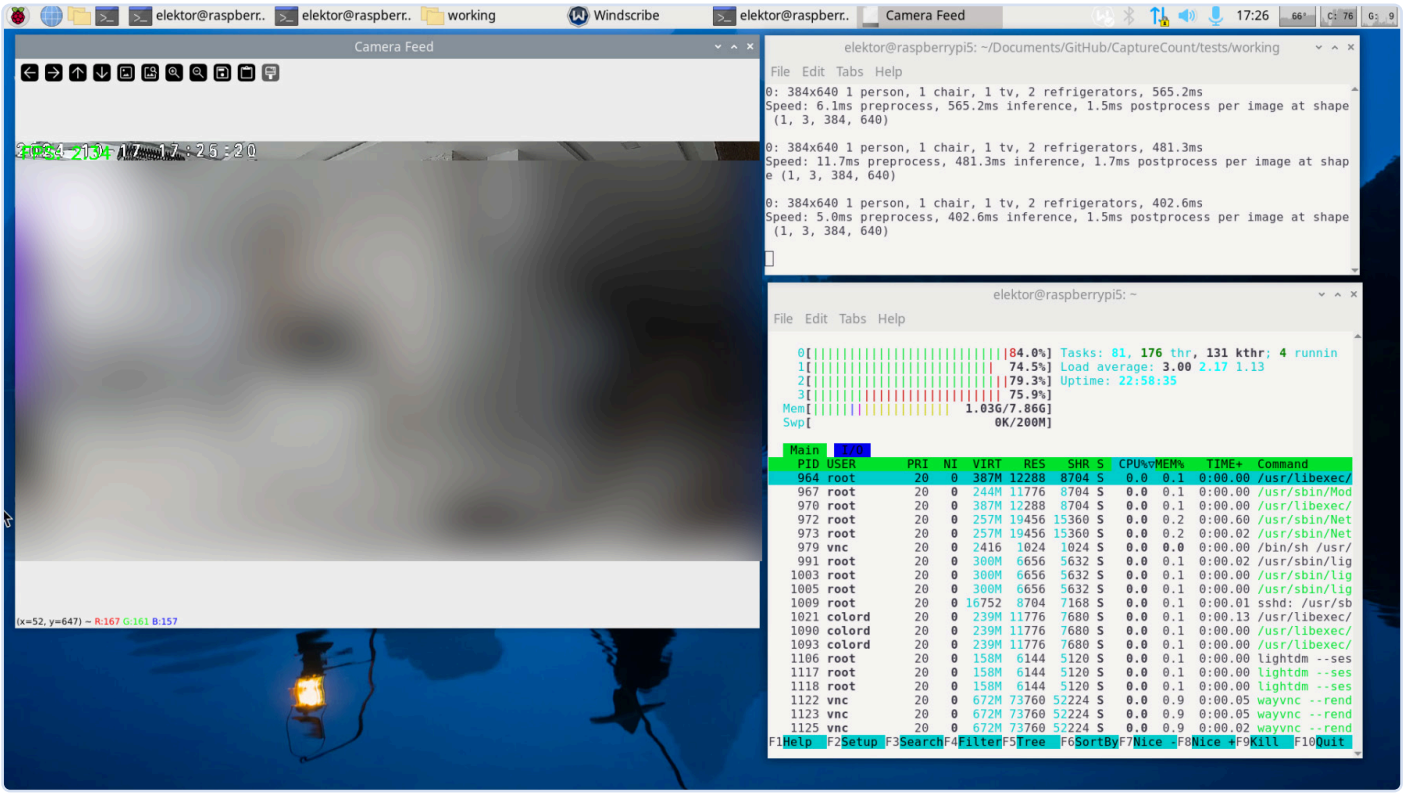


Figure 1. Performances initiales avec OpenCV et YOLOv8 sur Raspberry Pi 5, montrant des FPS limités et des contraintes de CPU.

Cette version du projet est conçue pour surveiller un seul flux CCTV, capturé par une caméra IP, et pour envoyer des alertes en temps réel dès qu'une personne est détectée. Le projet présente un potentiel considérable d'évolution, avec la possibilité d'ajouter la prise en charge de plusieurs flux de caméras et la capacité de détecter divers scénarios, tels que les risques d'incendie ou les risques de sécurité pour les groupes vulnérables comme les personnes âgées ou les enfants. Il est important de noter que tous les traitements de données sont effectués localement sur le Raspberry Pi, garantissant ainsi la confidentialité et évitant la nécessité d'un stockage en cloud ou du partage externe de données. Mais laissez-moi vous dire que sa construction n'a pas été une promenade de santé. Le chemin pour le rendre opérationnel a été jonché d'obstacles, et c'est précisément là que commence la véritable histoire.

Fonctionnement du projet

Pour saisir pleinement les capacités d'AlertAlfred, il est essentiel de comprendre son fonctionnement théorique. Ce projet repose sur des modèles de vision par ordinateur et d'apprentissage automatique pour détecter des objets (spécifiquement des personnes) dans un flux vidéo. Le traitement est effectué sur un Raspberry Pi 5, qui est considérablement renforcé par l'ajout de l'unité de traitement neuronal (NPU) Hailo 8L.

Dans une configuration ordinaire, utilisant par exemple OpenCV sur un Raspberry Pi 5, le CPU est chargé de gérer tout le pipeline de traitement vidéo, du décodage des images vidéo à l'exécution de modèles d'inférence sur chaque image. Cette tâche est très gourmande en ressources CPU, comme je l'ai constaté lors de mes premiers essais, où le système ne parvenait à traiter que 2 images par seconde (FPS) sur un seul flux vidéo, et les performances se dégradaient encore lorsque j'ai ajouté un deuxième flux (**figure 1**).

C'est là que le Hailo 8L entre en jeu. Il s'agit d'une puce spécialisée conçue pour accélérer les tâches d'IA, en particulier les modèles d'apprentissage profond, comme celui utilisé pour la détection d'objets. Voici pourquoi elle est plus rapide :

Accélération du réseau neuronal : le NPU Hailo 8L est spécialement conçu pour traiter des tâches telles que la détection d'objets et la classification d'images. Capable de traiter les réseaux neuronaux à la vitesse de 13 TOPS (Tera-Opérations par seconde), il peut exécuter des trillions d'opérations par seconde. Cette performance est bien supérieure à celle du processeur d'un Raspberry Pi qui n'est pas optimisé pour ce type de tâches. Le script de détection exécuté sur la vidéo d'exemple fournie dans le dépôt d'exemples *hailo-rpi-5* [1] donnait plus de 30+ FPS ! (**figure 2**)

Traitement dédié à l'IA : Contrairement au CPU du Raspberry Pi qui est polyvalent et gère tout, du système d'exploitation aux périphériques, le Hailo 8L est exclusivement dédié à l'inférence IA. Ce partage des tâches libère le Raspberry Pi pour gérer d'autres opérations, tandis que le module Hailo s'occupe de l'inférence d'apprentissage profond, très exigent en calcul.

Efficacité : l'architecture du Hailo 8L lui permet de traiter ces tâches plus efficacement, en consommant beaucoup moins d'énergie que que ce serait le cas sur un CPU ou même un GPU. Il est donc idéal pour les applications d'IA de pointe comme AlertAlfred, où un traitement continu est nécessaire sans surcharger le système.

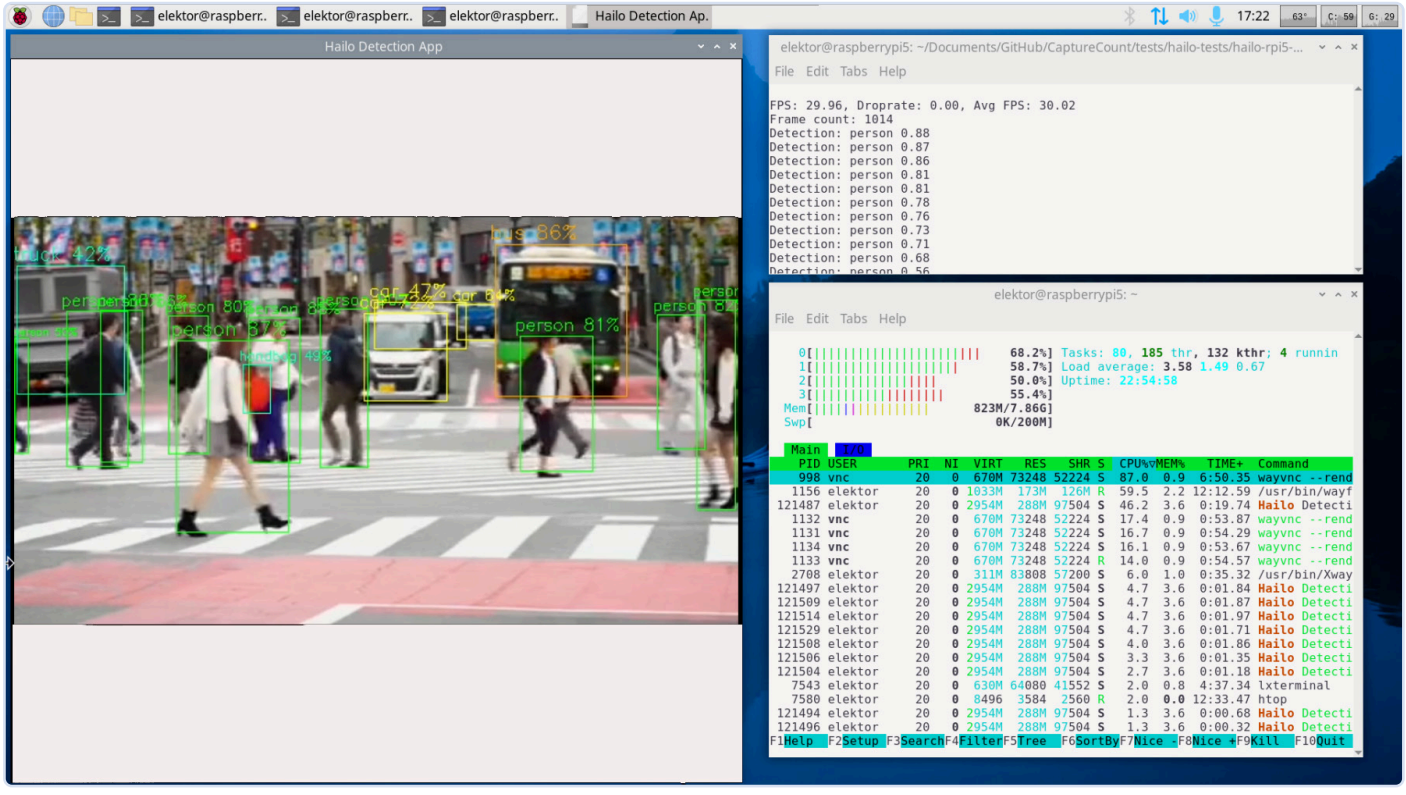


Figure 2. Amélioration des performances avec Hailo 8L, atteignant plus de 30 FPS sur la vidéo de test avec l'inférence accélérée de l'IA.

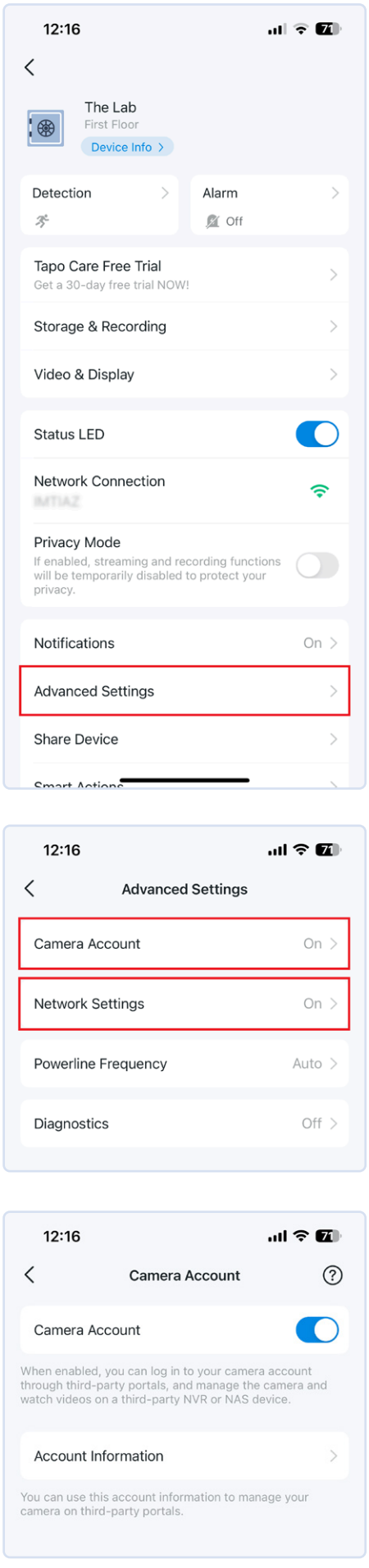
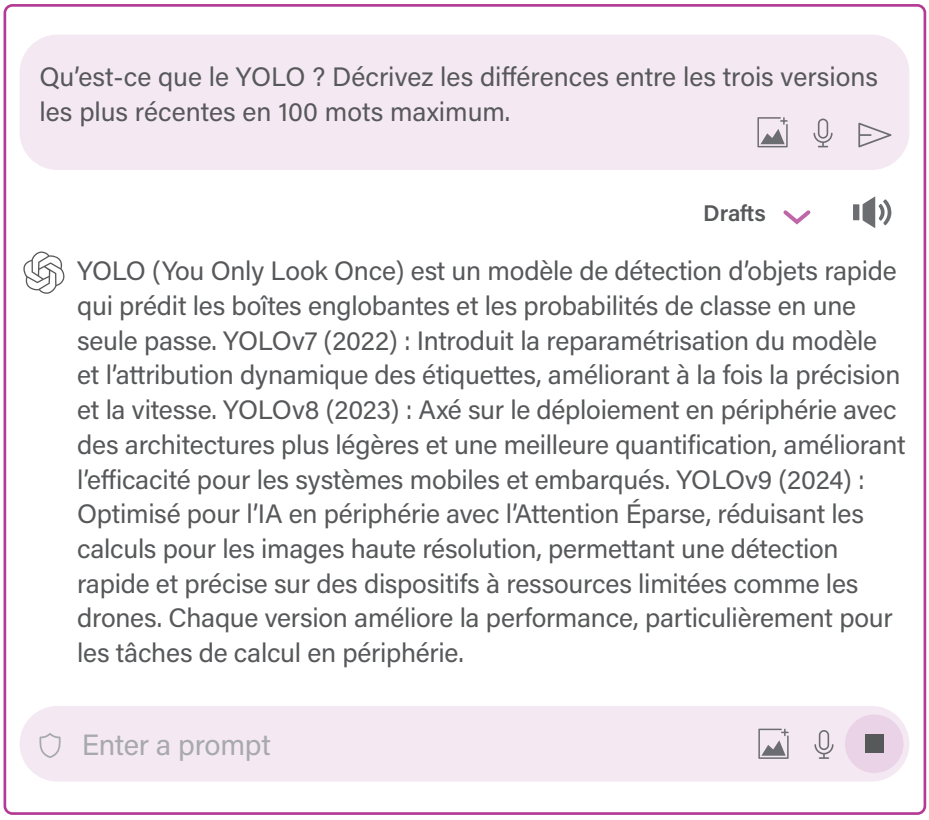


Figure 3. Activation du flux RTSP sur l'application Tapo de TP-Link pour la diffusion de séquences de vidéosurveillance.



En confiant le traitement du réseau neuronal au Hailo 8L, nous parvenons à améliorer considérablement les performances, permettant une détection de personnes en temps réel sur un flux de vidéosurveillance en direct, une tâche qui serait impossible si tout le travail était laissé au CPU du Raspberry Pi.

Flux vidéo d'une caméra IP

La création de ce projet n'a pas été simple. J'ai utilisé une caméra IP TP-Link Tapo C100 prête à l'emploi, qui diffuse un flux vidéo sur le réseau via le protocole RTSP (Real-Time Streaming Protocol). Mon premier essai pour traiter ce flux avec OpenCV et YOLOv8 sur le Raspberry Pi 5 s'est avéré décevant : je n'ai réussi à obtenir que 2 images par seconde (FPS) avec une seule caméra. L'ajout d'une seconde caméra a encore réduit le taux à seulement 1 FPS, ce qui a clairement montré que le processeur seul ne pouvait pas répondre aux exigences de la détection d'objets en temps réel.

C'est alors que je me suis tourné vers le module Hailo 8L, attiré par sa capacité de traitement de 13 TOPS pour les tâches d'intelligence artificielle, mais pour l'utiliser, il a fallu apprendre rapidement. Mon premier défi a été de modifier le code d'exemple de Hailo (fourni par Raspberry Pi) pour utiliser le flux RTSP de la caméra IP. Le protocole RTSP est couramment utilisé pour la diffusion de vidéos en temps réel, par exemple en direct de caméras de sécurité, mais sa connexion au pipeline de traitement de l'IA s'est avérée délicate. Après plusieurs tentatives infructueuses pour capturer directement le flux, j'ai opté pour une solution alternative : convertir le flux CCTV en une caméra virtuelle à l'aide d'un tutoriel disponible sur GitHub [2]. Cela m'a permis de séparer la gestion du flux du code principal.

Un dépôt GitHub a été créé pour ce projet [3]. Cet article se concentre sur les étapes clés et les idées principales de l'intégration, tandis que le dépôt GitHub offre un guide détaillé des étapes de la configuration.

Pour l'intégration de la caméra CCTV/Sécurité, j'ai utilisé la TP-Link Tapo C100. La configuration de la caméra est simple, mais une étape critique consiste à activer manuellement le flux RTSP, comme le montre la **figure 3**. Pour ce faire, naviguez vers les paramètres avancés dans l'application de la caméra et créez un compte. Veillez à définir un mot de passe robuste et à le noter pour référence ultérieure. Vous devrez également activer une IP statique pour la caméra afin d'éviter que l'adresse IP ne change à chaque réinitialisation du routeur. Cette configuration est accessible via les paramètres réseau de l'application de la caméra, comme le montre la **figure 4**.

Enfin, une fois la configuration de la caméra terminée, vous pouvez vérifier le fonctionnement du flux sur votre Raspberry Pi en utilisant la commande suivante. Assurez-vous d'ajouter l'adresse IP correcte de la caméra ainsi que le nom d'utilisateur et le mot de passe que vous avez définis précédemment.

```
gst-launch-1.0 rtspsrc location=rtsp://username:pass
```

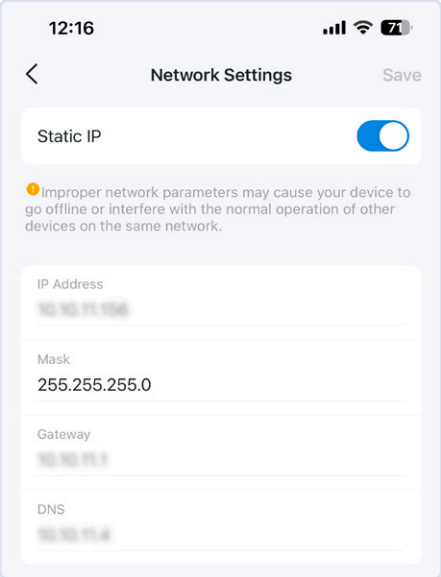


Figure 4. Définition d'une IP statique pour la caméra afin de garantir une intégration stable avec le système.

```
word@172.168.1.71:554/stream1
! rtph264depay ! avdec_h264 !
videoconvert ! autovideosink
```

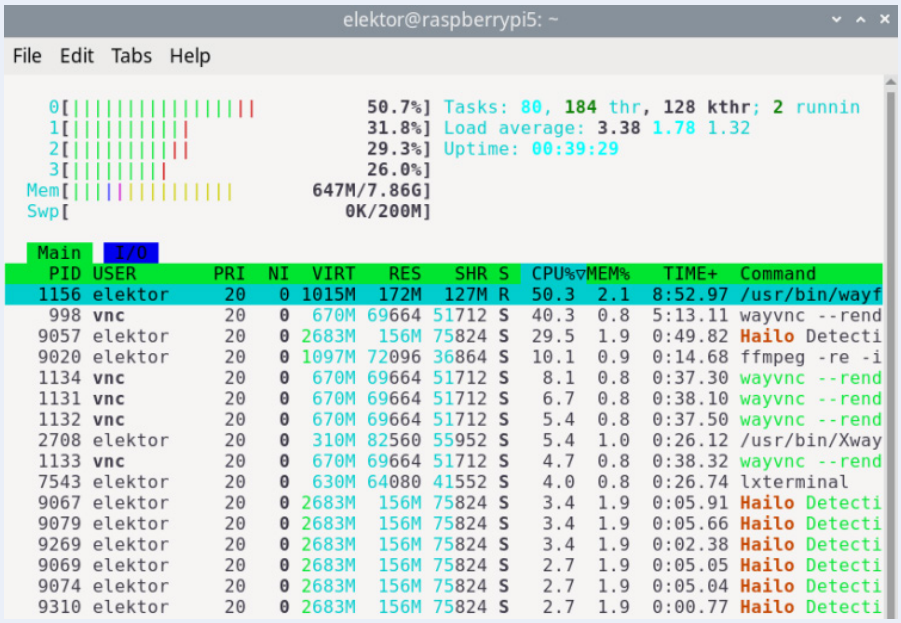
Cette commande lance un pipeline GStreamer - un cadre multimédia qui gère des tâches telles que la capture, le traitement et l'affichage de la vidéo et de l'audio. Dans cet exemple, `GStreamer` traite le flux de la caméra de vidéosurveillance. `rtspsrc` récupère le flux RTSP, `rtph264depay` décode le flux et `videoconvert` assure que le format e sortie est compatible avec l'affichage.
Une fois le bon fonctionnement du flux vidéo confirmé, vous devez transmettre ce flux CCTV à une caméra virtuelle sur votre Raspberry Pi. C'est un processus un peu long, donc pour cette étape, vous pouvez suivre le guide sur ce processus disponible dans le dépôt GitHub du projet [4].

Résolution des problèmes

Suite à la configuration de la caméra virtuelle, j'ai rencontré un autre problème : le script de détection de Hailo ne reconnaissait pas cette source. En investiguant, j'ai découvert une incompatibilité de format vidéo. La caméra virtuelle utilisait le format YUY2, qui n'était pas pris en charge par défaut dans les scripts d'exemples de Hailo. Le problème a été résolu par l'ajout d'une simple ligne de code dans le script pour prendre en charge ce format.
Voici la modification que j'ai apportée au fichier *hailo_rpi_common.py*, situé dans le répertoire *hailo-rpi5-examples/basic_pipelines*, entre les lignes 162 et 166 :

```
elif source_type == 'usb': source_
element = ( f'v4l2src device={video_
source} name={name} ! ' 'video/x-raw,
format=YUY2, width=640, height=360
! ' )
```

Performance Report for Different Raspberry Pi 5 Variants



IEn réponse aux interrogations soulevées lors de mon dernier projet d'IA, Capture Count [5], concernant les performances des différentes configurations du Raspberry Pi 5, voici un rapport succinct basé sur le test d'AlertAlfred avec une entrée CCTV. Selon mes observations, le programme fonctionne sans accroc sur un Raspberry Pi 5 doté de 2 Go de RAM, et il reste suffisamment de mémoire pour envisager l'ajout de deux flux de caméras supplémentaires. En somme, posséder plus de RAM sur votre Raspberry Pi 5 s'avère principalement bénéfique si vous planifiez de gérer plusieurs flux d'entrée. Toutefois, pour un usage se limitant à un unique flux CCTV, passer à une variante de 8 Go ne présente pas d'avantage significatif en termes de performances comparativement aux modèles de 2 ou 4 Go. L'intérêt de la RAM supplémentaire se manifeste uniquement lors du traitement de multiples flux simultanés. L'image ci-jointe illustre la consommation de RAM du projet.

Blocs du projet

Le flux de la caméra et le pipeline de détection sont maintenant correctement configurés. Avant de plonger dans le code principal de ce projet, prenons un moment pour examiner une vue d'ensemble du fonctionnement du système. Un schéma fonctionnel de haut niveau est présenté en **figure 5.** AlertAlfred utilise GStreamer pour le streaming vidéo, Hailo 8L pour l'inférence IA et OpenCV pour la capture et le traitement d'images. Voici une brève explication du fonctionnement du système de bout en bout :

Traitement des images avec GStreamer

GStreamer gère le flux vidéo provenant de la caméra de vidéosurveillance, tandis que le Hailo 8L traite chaque image pour détecter les personnes. Ce pipeline garantit que le traitement vidéo reste efficace, même à des résolutions plus élevées.

Logique de détection des personnes

Le modèle d'intelligence artificielle analyse en continu chaque image. Pour éviter les faux positifs, le système exige une détection

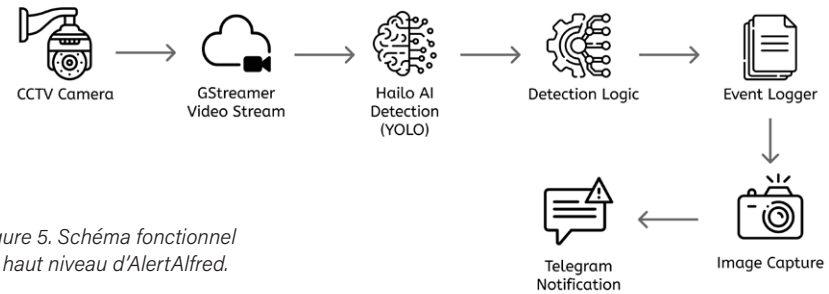


Figure 5. Schéma fonctionnel de haut niveau d'AlertAlfred.

Listage 1. Script Python (extrait).

```
# Import necessary libraries
import gi, os, csv, time, requests, cv2, hailo
from datetime import datetime
from gi.repository import Gst
from hailo_rpi_common import get_caps_from_pad, get_numpy_from_buffer, app_callback_class
# Define a custom callback class for handling detections and alerts
class user_app_callback_class(app_callback_class):
     def __init__(self):
         ...
         self.grace_period = 2  # Grace period in seconds
         self.csv_log_path = "logs/detection_log.csv"
         os.makedirs("logs", exist_ok=True)  # Ensure logs directory exists
         with open(self.csv_log_path, 'a', newline='') as file:
             csv.writer(file).writerow(["Timestamp", "Event"])  # Initialize CSV log
         ...
     def log_event(self, event):
         # Log detection events to CSV
         ...
     def send_telegram_alert(self, image_path):
         # Send detection alert with image via Telegram
         ...
# Callback function for handling frame detections from the pipeline
def app_callback(pad, info, user_data):
     frame = get_numpy_from_buffer(info.get_buffer(), *get_caps_from_pad(pad))
     detections = hailo.get_roi_from_buffer(info.get_buffer()).get_objects_typed(hailo.HAILO_DETECTION)
     ...
     # Check if a person is detected
     if any(d.get_label() == "person" for d in detections):
         if not user_data.person_detected:
              # Log entry event
             ...
         if user_data.detection_frame_count == 10:
             user_data.send_telegram_alert("image_path.png")  # Send alert
     elif time.time() - user_data.last_detection_time > user_data.grace_period:
         user_data.person_detected = False  # Reset detection status if grace period passed
     ...
# Run the detection app
if __name__ == "__main__":
     app = GStreamerDetectionApp(app_callback, user_app_callback_class())
     app.run()
```

cohérente des personnes sur 10 images avant de déclencher une alerte. Cela garantit la fiabilité dans des conditions réelles, où des objets temporaires ou des ombres pourraient autrement déclencher une fausse détection.

Capture d'images et alertes

Une fois qu'une personne est détectée, AlertAlfred capture un instantané et envoie une notification via Telegram. Cette notification inclut l'image, permettant une vérification visuelle immédiate pour l'utilisateur.

Enregistrement des événements

Tous les événements de détection (entrées et sorties) sont enregistrés dans un fichier CSV, fournissant un enregistrement détaillé de l'activité qui peut être examiné ultérieurement.

Délai de grâce pour les notifications

Le système intègre un délai de grâce de 10 secondes, après la détection d'une personne, pendant lequel le système n'enverra pas d'autre alerte pour la même personne. Cela permet d'éviter les notifications répétées si la personne reste dans le champ de vision de la caméra. Pour déclencher une deuxième alerte, la personne doit quitter et rentrer dans le cadre une fois le délai de grâce écoulé.

Le code

Le code principal du **listage 1** gère la détection de personnes en temps réel à l'aide d'un flux de vidéosurveillance, l'enregistrement des événements et l'envoi d'alertes via Telegram. Des bibliothèques clés telles que `cv2` et `numpy` sont utilisées pour le traitement des images, tandis que `requests` gère la communication HTTP avec Telegram. Le module Hailo 8L AI est utilisé pour une détection efficace des personnes, et GStreamer (`Gst`) gère le flux vidéo de la caméra de vidéosurveillance. Dans le listage 1, un extrait du code est partagée pour référence, le code principal est fourni dans le dépôt GitHub.

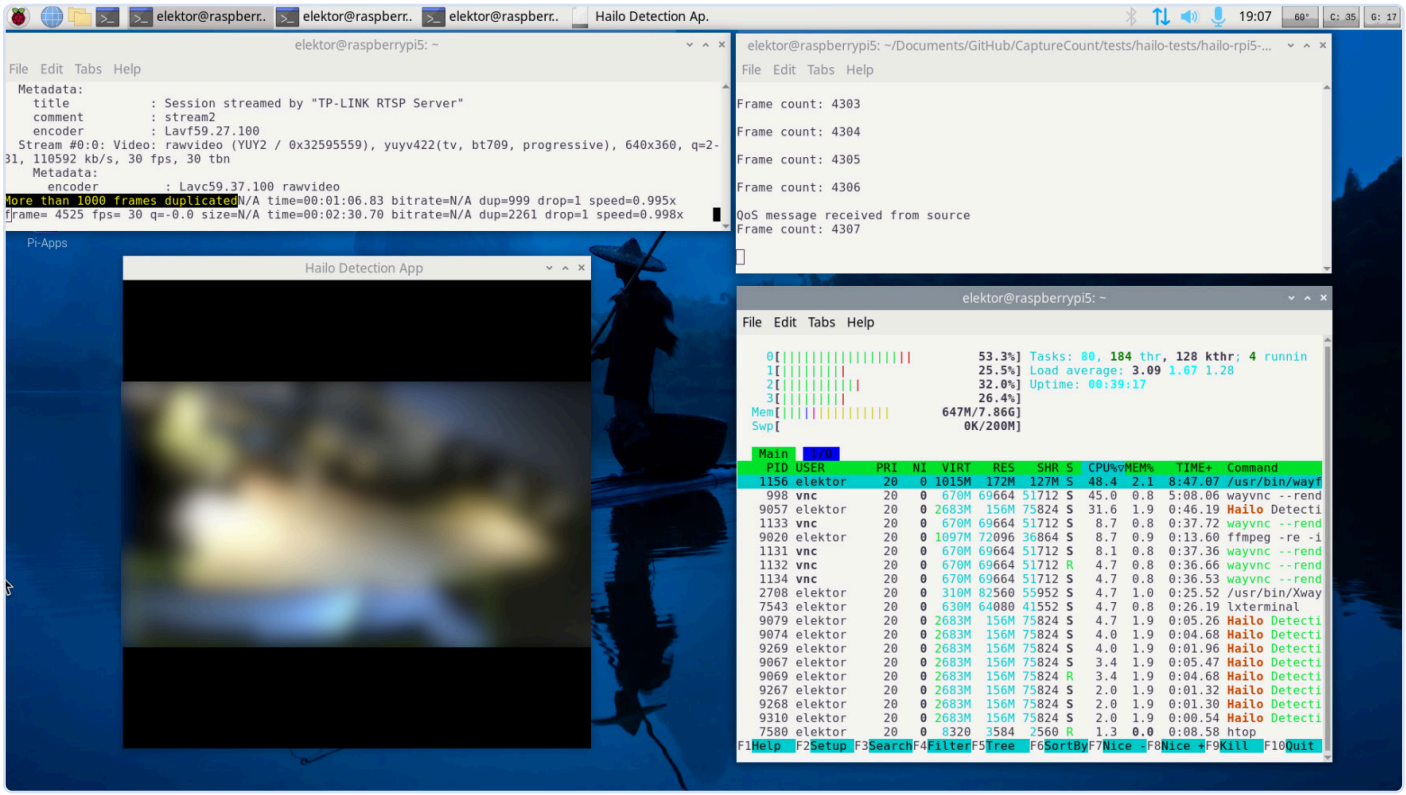


Figure 6. Flux CCTV en direct traité par AlertAlfred avec détection de personnes en temps réel sur le Raspberry Pi.

La logique de détection est encapsulée dans la classe `user_app_callback`, qui suit les événements de détection, les enregistre et envoie des notifications. Des variables clés comme `self.person_detected` permettent de savoir si une personne est actuellement détectée, et `self.detection_frame_count` garantit que le système ne déclenche des alertes qu'après 10 images consécutives afin d'éviter les faux positifs. La `grace_period` de 2 secondes empêche les alertes répétées pour la même personne, tandis que self.csv_log_path définit l'endroit où les événements de détection sont enregistrés. L'intégration de Telegram est gérée via `self.telegram_bot_token` et `self.telegram_chat_id`, qui permettent au système d'envoyer des alertes en utilisant l'API Telegram.

La méthode `setup_csv_log` s'assure de l'existence d'un répertoire de logs et d'un fichier CSV, tandis que `log_event` enregistre les événements d'entrée et de sortie de la personne. La méthode `send_telegram_alert` construit et envoie un message avec les détails de la détection et une image d'accompagnement de la personne détectée via Telegram.

La fonction `app_callback` est le cœur du pipeline de détection, déclenché lorsque de nouvelles données vidéo sont disponibles. Elle extrait l'image et la traite à l'aide du module Hailo 8L, en vérifiant les détections de « personnes ». Si une personne est détectée de manière cohérente sur 10 images, une image est capturée, sauvegardée et envoyée via Telegram. Si aucune personne n'est détectée après le délai de grâce, l'état de détection est réinitialisé et un événement de sortie est enregistré.

Enfin, l'exécution principale crée une instance de la classe de rappel, initialise l'application de détection et exécute le système. Cette configuration intègre efficacement la détection de personnes, l'enregistrement d'événements et les alertes en temps réel, alimentée par le module Hailo 8L pour l'accélération de l'IA et le traitement local axé sur la confidentialité.

AlertAlfred en action

Pour déployer AlertAlfred sur votre Raspberry Pi 5, vous devez d'abord disposer de quelques composants matériels essentiels. Le système nécessite un Raspberry Pi 5, le kit Raspberry Pi AI (livré avec le M.2 HAT+ et le module Hailo 8L), une caméra CCTV compatible et une carte microSD chargée avec Raspberry Pi OS.

Pour la configuration du logiciel, vous devrez d'abord cloner le dépôt Hailo, qui sert de base à la construction du pipeline de détection. Cette opération peut être réalisée avec la commande :

```
git clone https://github.com/hailo-ai/hailo-rpi5-examples.git
```

Une fois le dépôt cloné, vous devrez installer les dépendances nécessaires, y compris OpenCV, NumPy, et GStreamer. Vous pouvez le faire en suivant simplement les instructions dans le fichier *doc/basic-pipelines.md*, dans le dépôt *hailo-rpi5-examples*. Après avoir configuré votre environnement, configurez Telegram en créant un bot via BotFather et en obtenant le jeton du bot et l'ID du chat. Ces détails seront ajoutés au script pour activer des alertes en temps réel. Enfin, collez le script *alert-alfred.py* dans le dossier *basic_pipelines* du dépôt *hailo-rpi5-examples*.

Avant d'exécuter le script, vous devez lancer le processus dans un terminal séparé pour commencer à capturer les séquences CCTV sur votre caméra virtuelle. Vous pouvez le faire avec la commande suivante (veuillez suivre le guide [4] avant d'exécuter cette commande) :

```
ffmpeg -re -i rtsp://username:password@172.168.1.71:554/stream1 -r 30 -f v4l2 -vcodec rawvideo -pix_fmt yuyv422 /dev/video10
```

Après avoir commencé à capturer les images CCTV sur votre caméra virtuelle, vous pouvez exécuter le script en utilisant la commande suivante via le terminal, alors que vous êtes dans le dossier *hailo-rpi5-examples* :

```
python basic_pipelines/alert-alfred.py --input /dev/video10
```

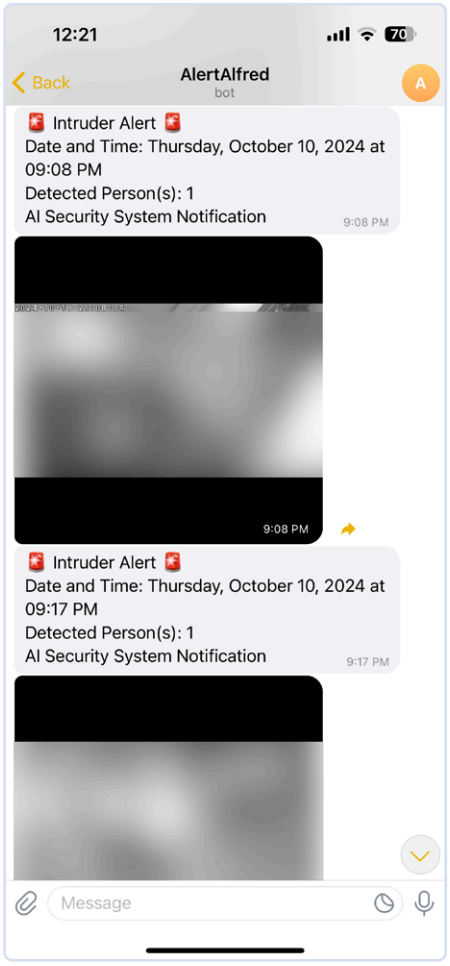


Figure 7. Notification d'alerte Telegram montrant la personne détectée, l'horodatage et une image du flux de vidéosurveillance.

En supposant que l'entrée */dev/video10* du script transmet correctement vos images CCTV, le système commencera à traiter le flux pour la détection de personnes. Si tout fonctionne comme prévu, le système devrait ressembler à la **figure 6**, qui montre le flux en direct en cours de traitement pour les détections.

Lorsque le système identifie une personne, il capture une image et envoie une notification en temps réel via Telegram, y compris l'horodatage et le nombre de personnes détectées, comme le montre la **figure 7**. Cette notification fournit une confirmation visuelle instantanée, garantissant que vous restez informé de toute activité détectée.

Possibilités futures

Bien que la version actuelle d'AlertAlfred se concentre sur la détection de personnes à l'aide d'un seul flux CCTV et l'envoi d'alertes via Telegram, le projet a le potentiel d'évoluer vers un système de surveillance plus robuste et polyvalent. Les améliorations futures pourraient inclure la prise en charge de plusieurs flux de vidéosurveillance, ce qui permettrait de surveiller des zones plus vastes ou des sources multiples. En outre, le système pourrait être adapté pour identifier d'autres types de scénarios critiques tels que les risques d'incendie, les chutes de personnes âgées, ou pour renforcer la sécurité dans divers contextes. Le fonctionnement entièrement local sur le Raspberry Pi assure une plateforme fiable pour les applications qui mettent l'accent sur la confidentialité des données.

Les contributions de la communauté peuvent aider à faire avancer ce projet. Que ce soit par l'amélioration des algorithmes de détection, l'expansion des scénarios identifiables, ou l'ajout de nouvelles fonctionnalités, chaque contribution est une opportunité d'enrichir le système. Toute contribution ou suggestion visant à améliorer le système sera grandement appréciée par le développeur et l'ensemble de la communauté.

240474-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (saad.imtiaz@elektor.com), ou contactez Elektor (redaction@elektor.fr).

Contribuez à ce projet

Si vous souhaitez contribuer à ce projet, vous pouvez poster vos idées sur la plateforme en ligne Elektor Labs : www.elektormagazine.fr/labs.

À propos de l'auteur

Saad Imtiaz, ingénieur senior chez Elektor, est spécialisé en mécatronique et possède une solide expérience dans les systèmes embarqués et le développement de produits. Tout au long de sa carrière, il a collaboré avec un large éventail d'entreprises, des startups novatrices aux multinationales bien établies, en pilotant des projets de prototypage et de développement à la pointe de la technologie. Avec un parcours significatif dans l'industrie aéronautique et à la tête d'une startup technologique, Saad apporte à Elektor une combinaison unique de compétences techniques et d'esprit entrepreneurial. Il contribue au développement de projets dans les domaines du logiciel et du matériel.

Produits

- **Raspberry Pi 5 (8 GB RAM)** www.elektor.fr/20599
- **Raspberry Pi AI Kit** www.elektor.fr/20879
- **Raspberry Pi AI Camera** www.elektor.fr/20953
- **Argon NEO 5 BRED Case for Raspberry Pi 5** www.elektor.fr/20788

LIENS

[1] Exemples avec Hailo Raspberry Pi 5 | Dépôt Github : https://github.com/hailo-ai/hailo-rpi5-examples
[2] RTSP.to-webcam | Dépôt Github : https://github.com/apple-fritter/RTSP.to-webcam
[3] AlertAlfred: AI Security System | Dépôt Github : https://github.com/ElektorLabs/Alert-Alfred
[4] Guide to stream the CCTV camera on the virtual camera | Dépôt Github : https://github.com/ElektorLabs/Alert-Alfred/blob/main/docs/cctv-to-virtualcam-guide.md
[5] Saad Imtiaz, "détecteur et compteur d'objets basé sur le Raspberry Pi 5," Elektor 3-4/2024: https://elektormagazine.fr/230749-04