

Projet de vidéosurveillance : détection de mouvement et de personnes

SAE 202 - Semestre 2

Présenté par BEN OMRANE Yacine, CANADA Matteo,
KRUCZYNSKI Solal et RECHIDOV Soulim



Sommaire

- 1. Affichage du flux IP
- 2. Partie 1 vision par ordinateur (Filtrage, opérations morphologiques)
- 3. Partie 2 vision par ordinateur (HSV et Gestion d'états)
- 4. Partie système
- 5. Conclusion

Yacine

Affichage du flux IP

03/24



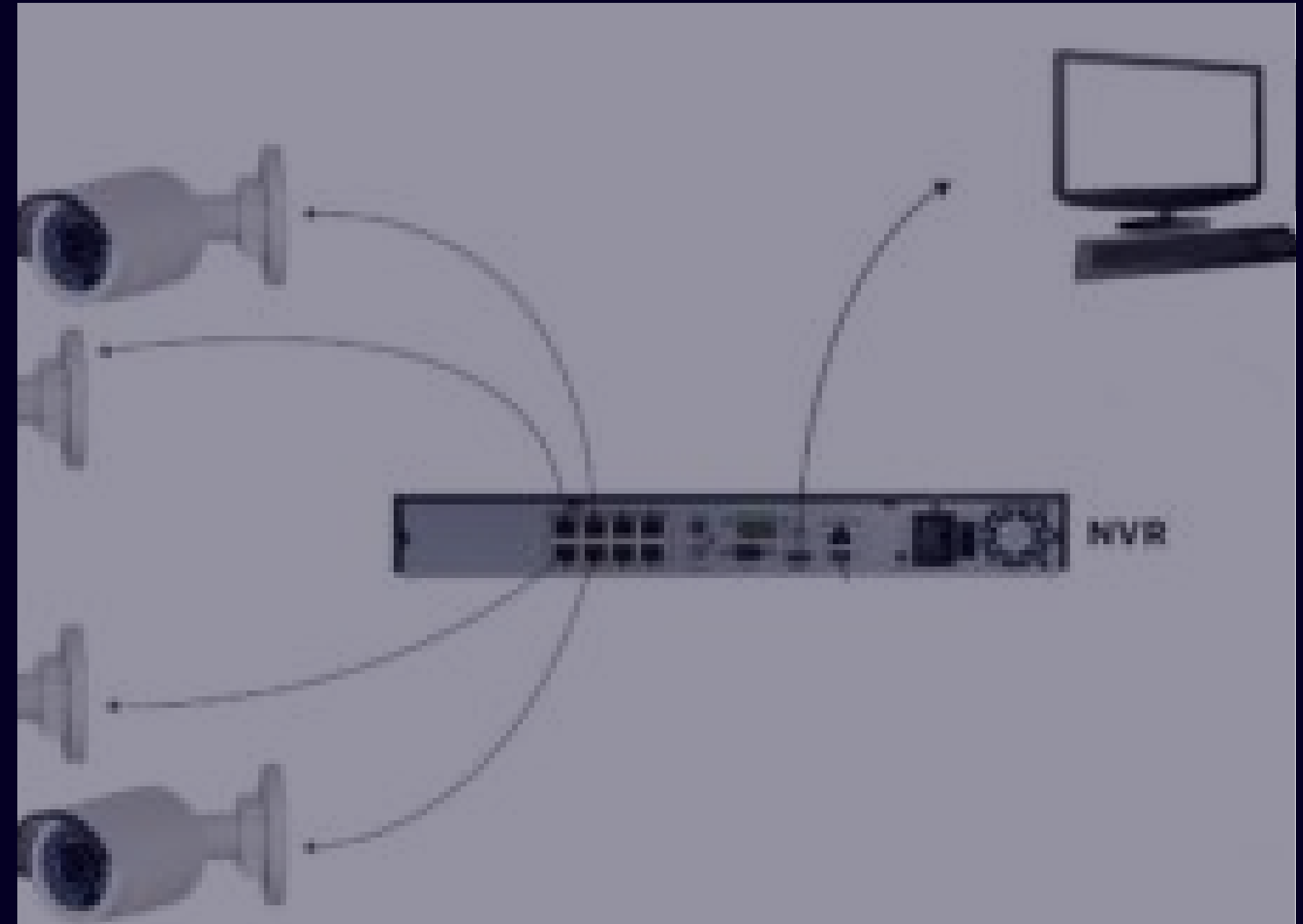
Matériel et Réseau

Caméra IP :

- *Connectée en Wi-Fi ou Ethernet*
- *Dispose d'une adresse IP locale (ex : 192.168.0.90)*
- *Envoie un flux vidéo compressé en continu*

Client (ex: Ordinateur) :

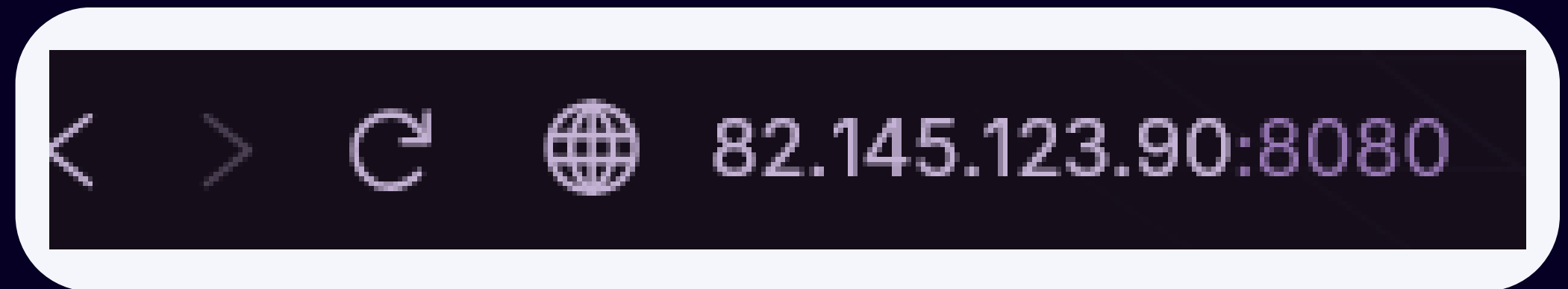
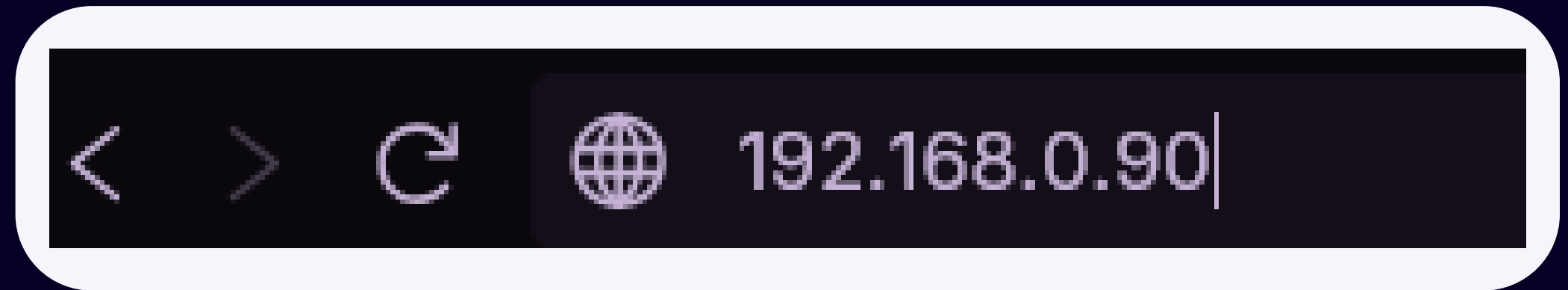
- *Doit être sur le même réseau (ou accès via IP publique)*
- *Utilise un navigateur web*



Comment on affiche le flux vidéo ?

Connexion à l'IP de la caméra

- Exemple d'URL : *http://192.168.0.90*



Transport de l'image via IP (en temps réel)

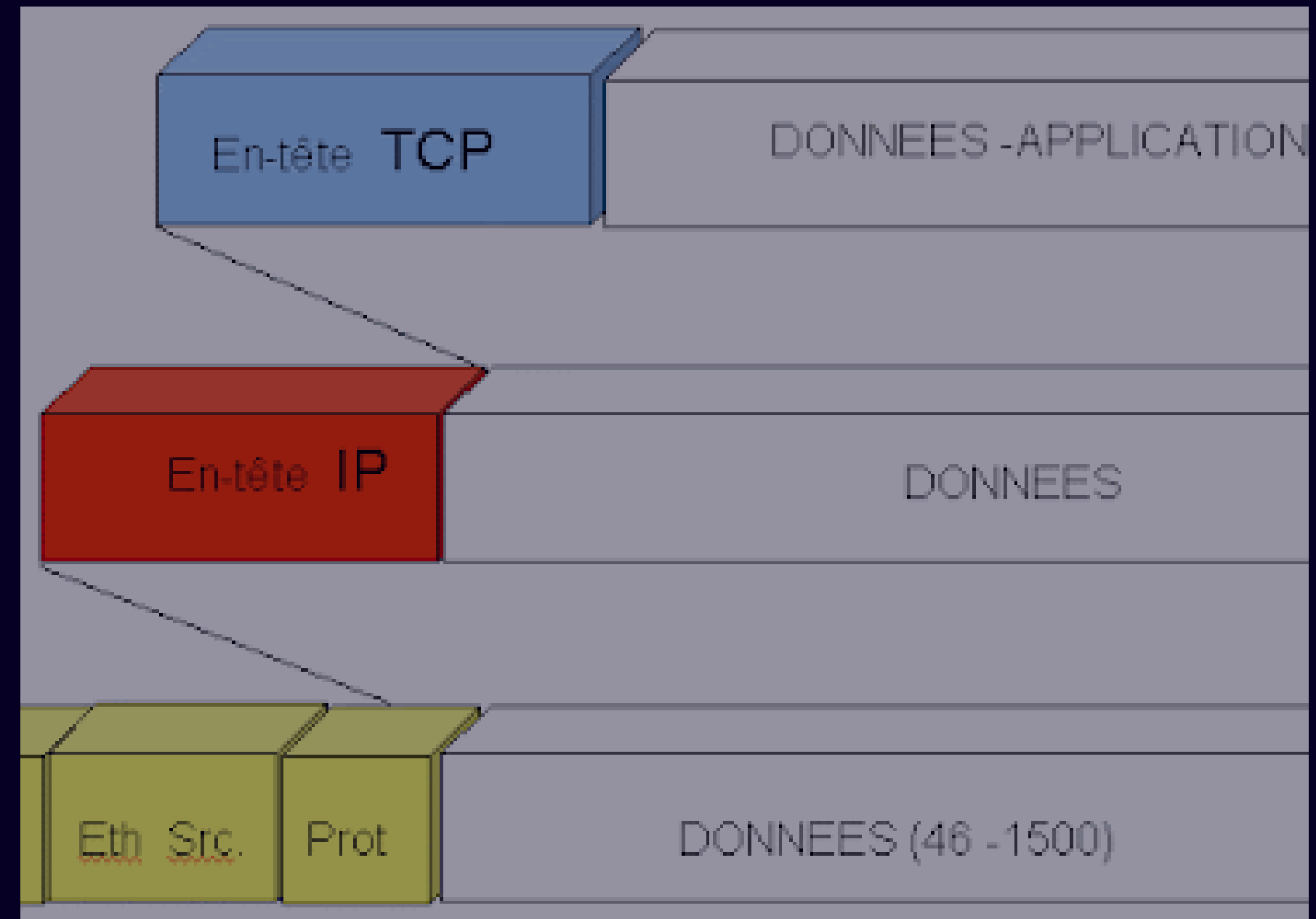
Image → Données numériques :

- *La caméra capture 25 à 30 images/seconde*
- *Chaque image est compressée (codec H.264, H.265, MJPEG)*
- *Elle devient une suite de bits (0 et 1)*



Fragmentation -> Paquets réseau

- Chaque image est découpée en petits paquets
- *Transport via :*
 - HTTP
 - *Souvent encapsulé dans TCP*



Transport IP

- Les paquets voyagent sur le réseau IP
- Chaque paquet contient :
 - Une partie du flux vidéo
 - Une adresse source et destination
 - Un horodatage pour synchroniser la lecture



Réception + lecture

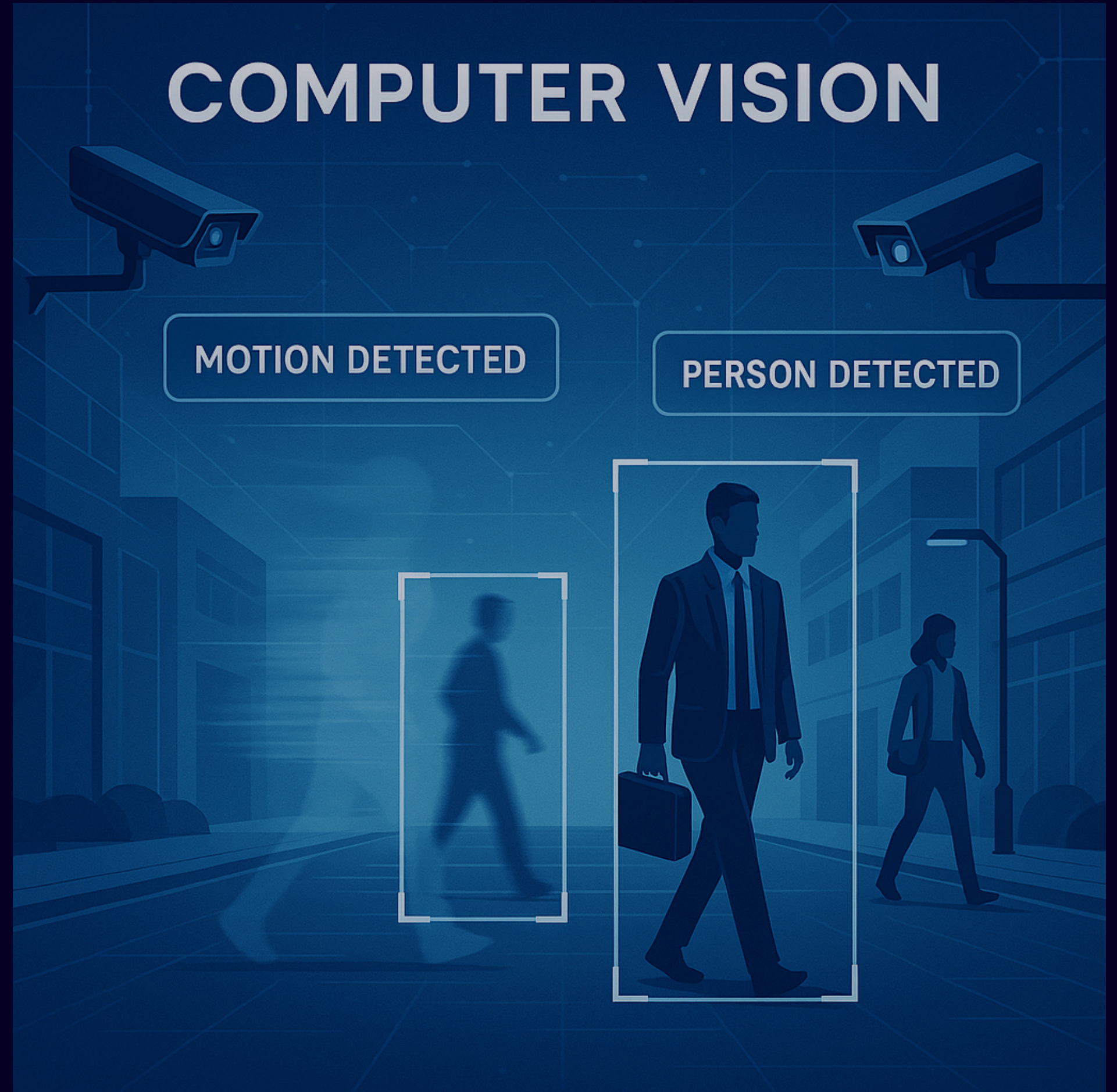
- Il les remet dans l'ordre
- Il décompresse (décode) les images
- Il les affiche en continu -> vidéo temps réel



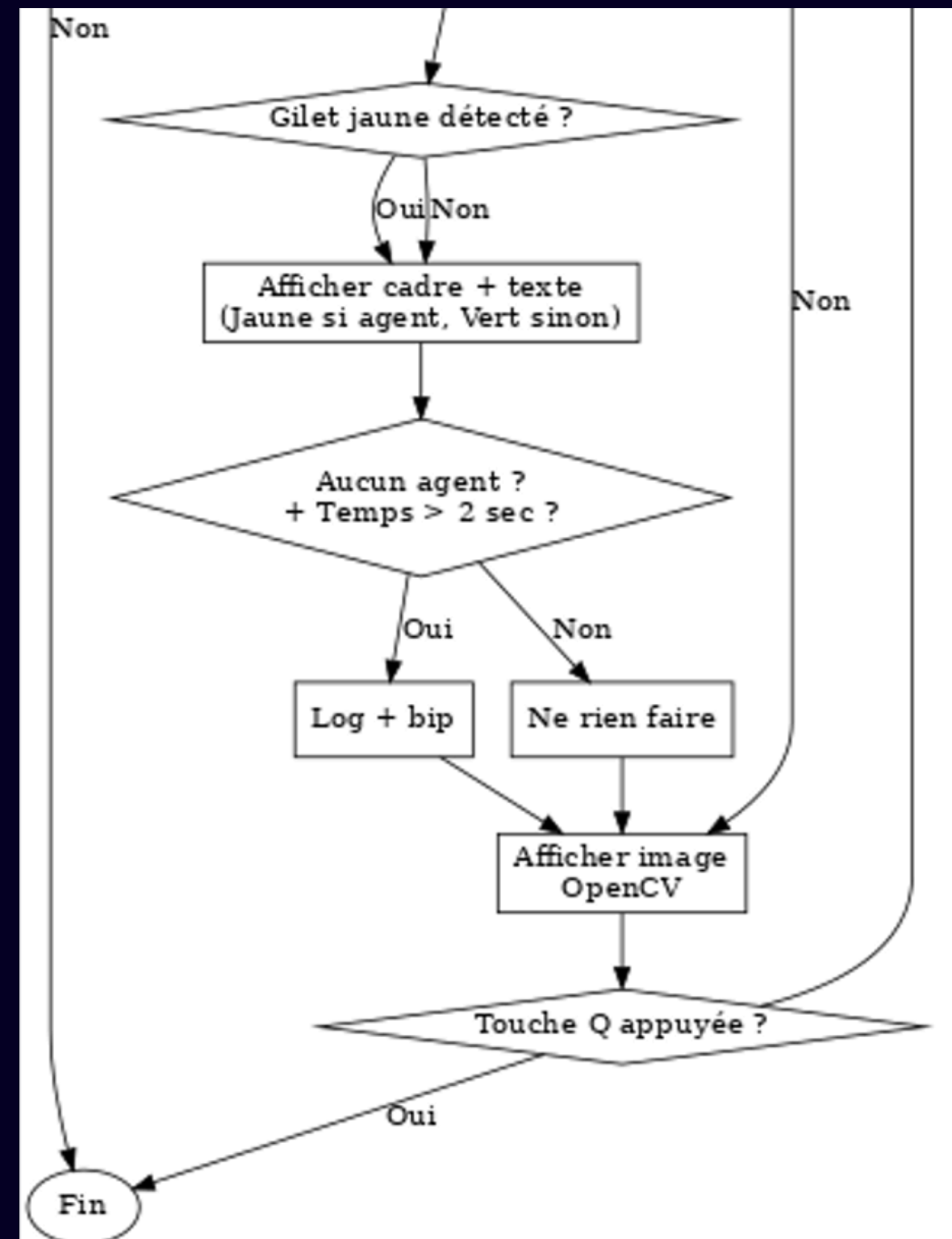
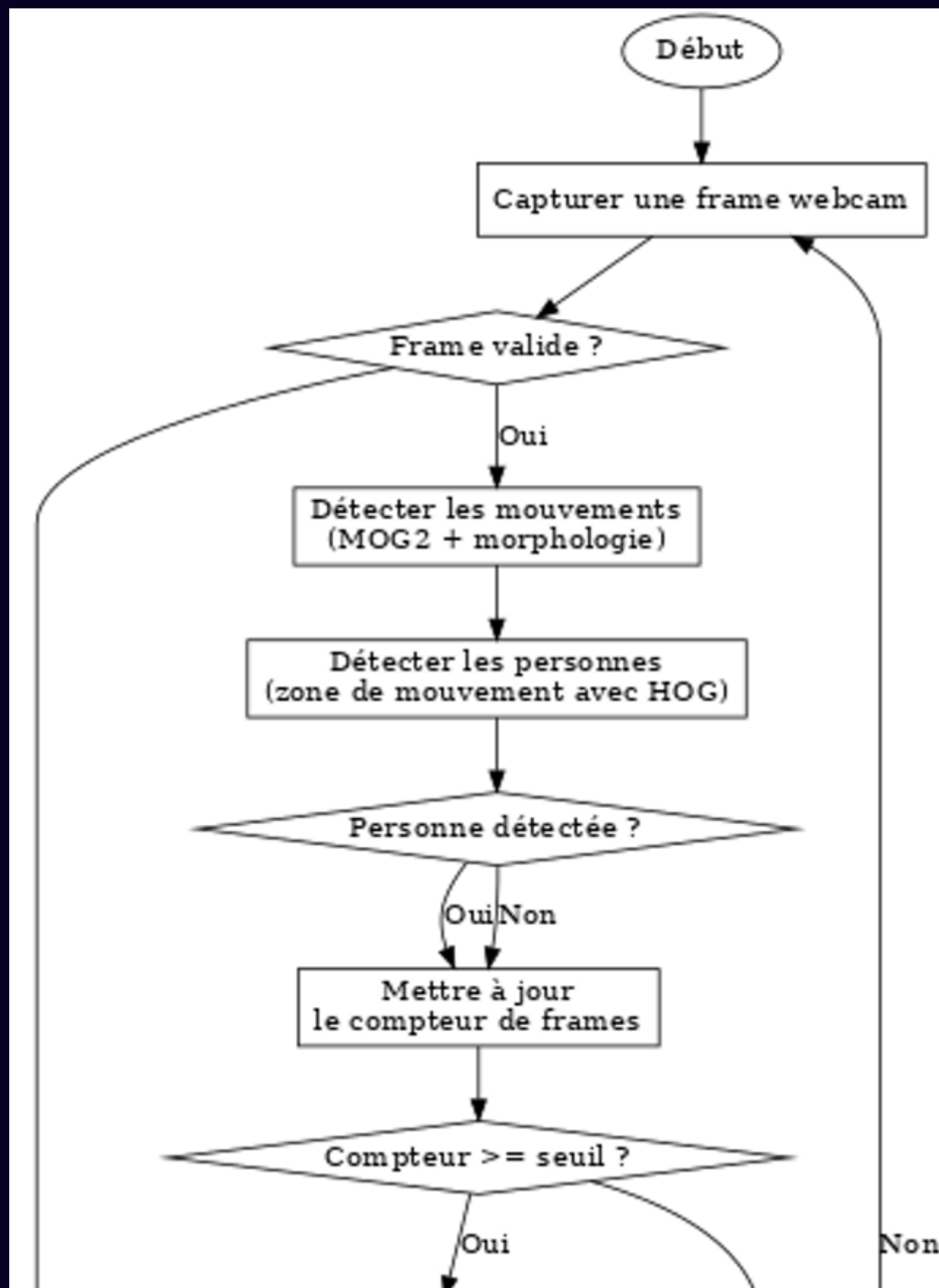
Mattéo

Partie 1 vision par ordinateur
(Filtrage, opérations morphologiques)

10/24



Présentation générale du programme



Notions présentes :

- Filtrage/Morphologie
- Soustraction de fond
- Vision par ordinateur
- Détection de couleur

Détection de mouvements via MOG2

(Mixture of gaussians version 2)

C'est quoi ?

Algorithme probabiliste de modélisation
d'arrière plan adaptatif

Pour chaque pixel il conserve plusieurs gaussiennes

- *une moyenne* ———> Valeur moyenne du pixel
- *une variance* ———> Variation de la valeur selon T
- *un poids* ———> Fréquence d'apparition

Lignes clés :

Dans script_detection.py [Ligne 86]

```
fgbg = cv2.createBackgroundSubtractorMOG2  
(history=500, varThreshold=50, detectShadows=False)
```

#Création de l'objet de soustraction
de fond basé sur l'algorithme MOG2

History = frames pour construire modèle de fond
⚠ (+ history grand + il est stable)

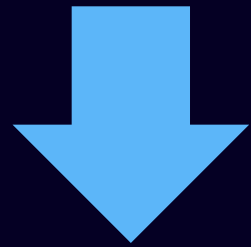
varThreshold = Seuil de variance

Filtrage et opérations morphologique

L'objectif 

Nettoyage du masque binaire
généré par MOG2

éliminer les artefacts
combler les trous
agrandir les zones pertinentes



 Robustesse / Fiabilité

Lignes de code :

Dans fonction : `detecter_mouvement(frame)`

```
fgmask = fgbg.apply(frame) #1. Soustraction de fond
```

```
fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_CLOSE, KERNEL)  
2#. Fermeture
```

```
fgmask = cv2.dilate(fgmask, None, iterations=2) #3. Dilation
```

`fgbg.apply` → Renvoie masque binaire (P = 0 ou 255)

`KERNEL` = Matrice 5x5 définition de la taille effet

`iterations = 2` pour double dilatation

Détection d'individu via HoG

Description de forme $\neq$ filtre

Pourquoi il est adapté à notre cas :

- Très fiable sur personne debout
- Léger (mémoire utilisée faible)

Enfin on obtient une liste de triangle
(x, y, w, h)

Lignes de code :

#Initialisation du descripteur

```
hog = cv2.HOGDescriptor()  
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

#Dans la boucle principale

```
frame_masked = cv2.bitwise_and(frame, masque)  
frame_upscaled = cv2.resize(frame_masked, None, fx=UPSCALE, fy=UPSCALE)  
personnes, _ = hog.detectMultiScale(frame_upscaled, winStride=(8, 8),  
padding=(8, 8), scale=1.05)
```

#SVM modèle d'approche structurelle pré-entraîné

Solal

Partie 2 vision par ordinateur
(HSV et Gestion d'états)

15/24

HSV et Gestion d'états

Gilet jaune
détecté =

Agent
de sécurité



HSV

Pourquoi HSV ?

Parce qu'il est plus robuste pour reconnaître des couleurs spécifiques, comme le jaune, même si l'éclairage change.

HSV

- On extrait une zone de l'image
- Puis on convertit cette région de l'espace BGR (Bleu-Vert-Rouge) vers HSV :
- HSV = Hue (Teinte), Saturation (intensité de la couleur), Value (luminosité)
- Ensuite, on crée un masque : on garde seulement les pixels dont la teinte est comprise entre [25, 70, 140] et [40, 255, 255] — typique du gilet jaune.
- On calcule la proportion de jaune dans la zone, et si elle dépasse SEUIL_JAUNE, on considère que la personne porte un gilet jaune

```
def contient_gilet_jaune(region):  
    hsv = cv2.cvtColor(region, cv2.COLOR_BGR2HSV)  
    bas_jaune = np.array([25, 70, 140])      # ✅ Plage ajustée pour ton gilet (i  
    haut_jaune = np.array([40, 255, 255])  
    masque = cv2.inRange(hsv, bas_jaune, haut_jaune)  
    proportion = cv2.countNonZero(masque) / (region.shape[0] * region.shape[1])  
    return proportion >= SEUIL_JAUNE
```

Gestion d'états

- Le programme ne déclenche pas une alerte immédiatement dès qu'il voit une silhouette.
- Il attend que la détection soit confirmée pendant 3 images consécutives (variable `VALIDATION_FRAMES`). Ça évite les faux positifs.
- Une fois validée, il regarde si un agent de sécurité (gilet jaune) est détecté :
 - S'il y en a un : pas d'alerte.
 - S'il n'y en a pas : bip et envoi de log.
- Pour éviter de spammer les alertes : on impose un délai de 2 secondes (`INTERVALLE_SUPPRESSION`) entre chaque enregistrement.

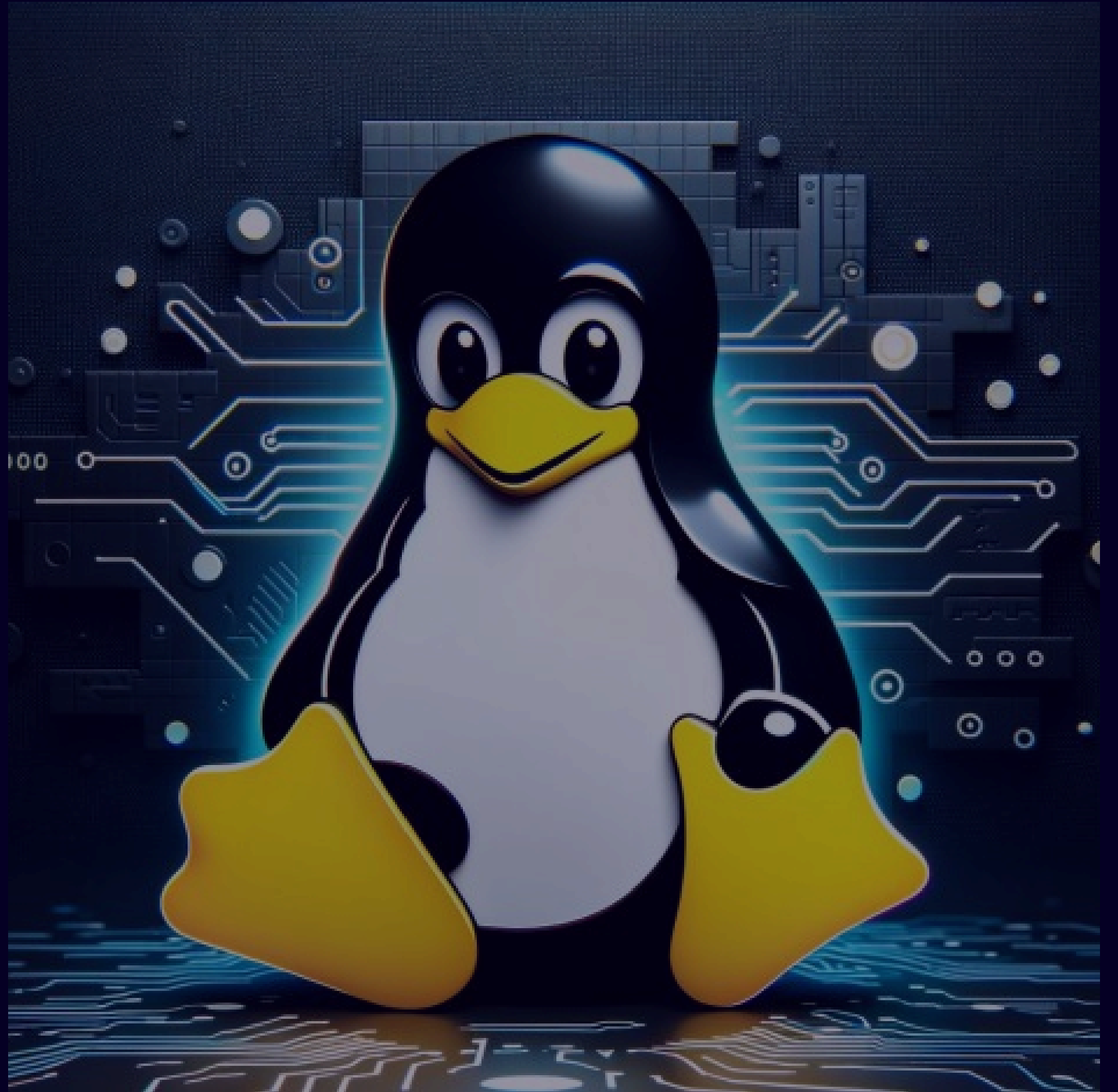
```
personne_detectee_frames = 0  
last_detection_time = 0
```

```
if personne_detectee:  
    personne_detectee_frames += 1  
else:  
    personne_detectee_frames = 0
```

```
if personne_detectee_frames >= VALIDATION_FRAMES:  
    ...  
    current_time = time.time()  
    if not agent_detecte and (current_time - last_detection_time >= INTERVALLE_SUPPRESSION):  
        log_detection("Personne detectee")  
        winsound.Beep(FREQUENCE_BIP, DUREE_BIP)  
        last_detection_time = current_time
```

Soulim

Partie système



Problématique : Comment savoir si quelqu'un est passé dans la nuit ?

Notre solution : Des fichiers de log sur un serveur Linux

Pourquoi ce choix :

- Faire participer les connaissances de l'année dans cet SAE
- Serveur Debian → Très sécurisé

Les principales étapes

- Créer une VM Debian en mode bridge 
- Connexion SSH via paire de clés privées/publiques sans mot de passe
- Modifier ce fichier pour refuser les connexions ssh basiques `/etc/ssh/sshd_config`
- Gestion des permissions sur le fichier et le répertoire
- Utilisation de Fabric pour la connexion SSH via python



Les principales étapes

```
1 # === LOGGING ===
2 def log_detection(texte):
3     from datetime import datetime
4     from fabric import Connection
5     import paramiko
6
7     # 1) Enregistrement local dans le fichier de log
8     timestamp = datetime.now().strftime("[%Y-%m-%d %H:%M:%S]")
9     ligne_log = f"{timestamp} {texte}\n"
10    with open("detection_log.txt", "a") as log_file:
11        log_file.write(ligne_log)
12
13    # 2) Envoi de la même ligne sur le serveur distant
14    rsa_key = paramiko.RSAKey.from_private_key_file(r"C:\Users\matte\.ssh\id_rsa")
15    cnx = Connection(
16        host='172.23.99.178',
17        user='sae202',
18        connect_kwargs={
19            "pkey": rsa_key,
20            "look_for_keys": False,
21            "allow_agent": False,
22            "disabled_algorithms": {"pubkeys": ["ssh-dss"]}
23        }
24    )
25    commande = f"echo '{ligne_log.strip()}' >> /home/sae202/fichier_log"
26    cnx.run(commande, hide=True)
27
```

TRÈS IMPORTANT !

CONCLUSION

**MERCI DE NOUS
AVOIR ÉCOUTÉS.**