Surveillance d'une zone privée

15088

Noa PABION

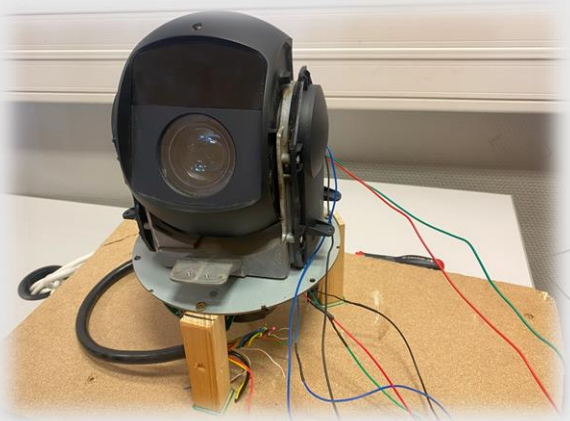
Sommaire

Introduction

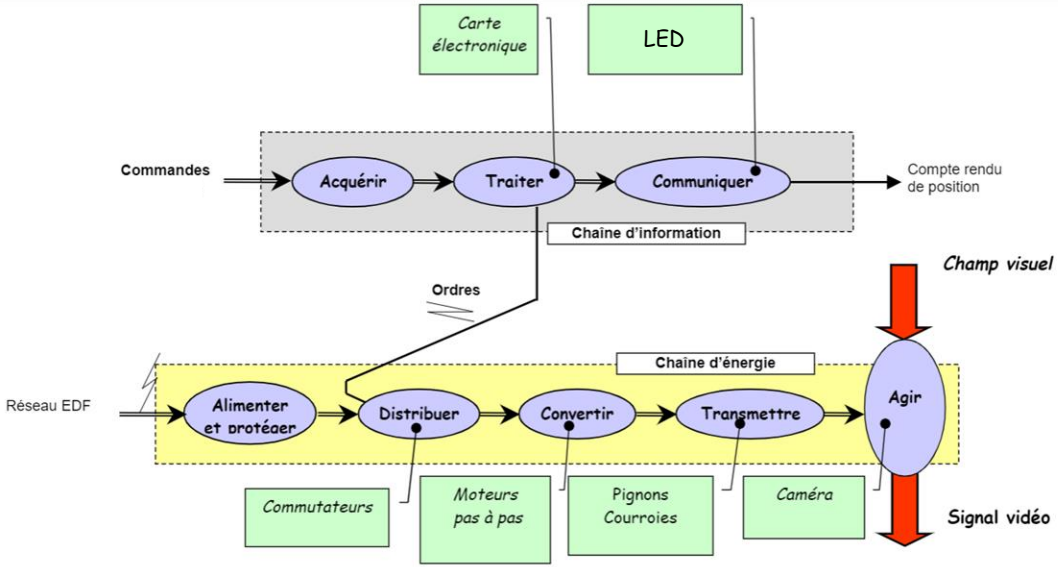
- Surveillance constante : *Prévention, sécurité, espaces publics.*
- Soutien aux forces de l'ordre : *Preuves visuelles, identification rapide.*
- Gestion urbaine améliorée : *Flux de circulation, sécurité routière.*
- Respect de la vie privée : *Responsabilité, conformité légale.*

Problématique

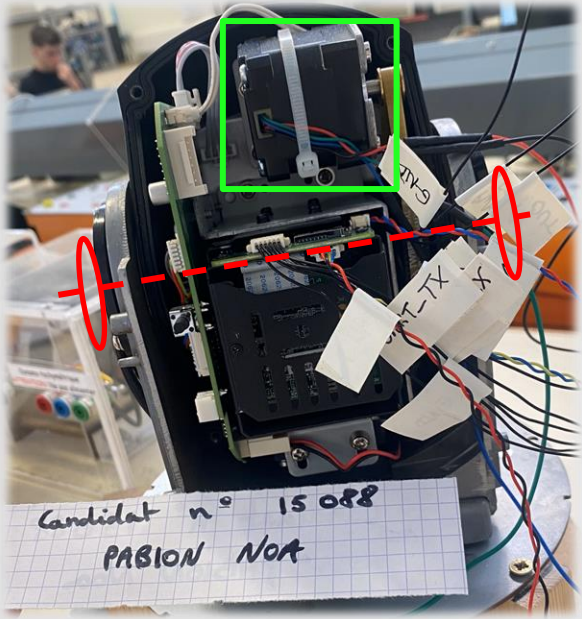
Comment piloter une caméra de sécurité par suivi de mouvement ?

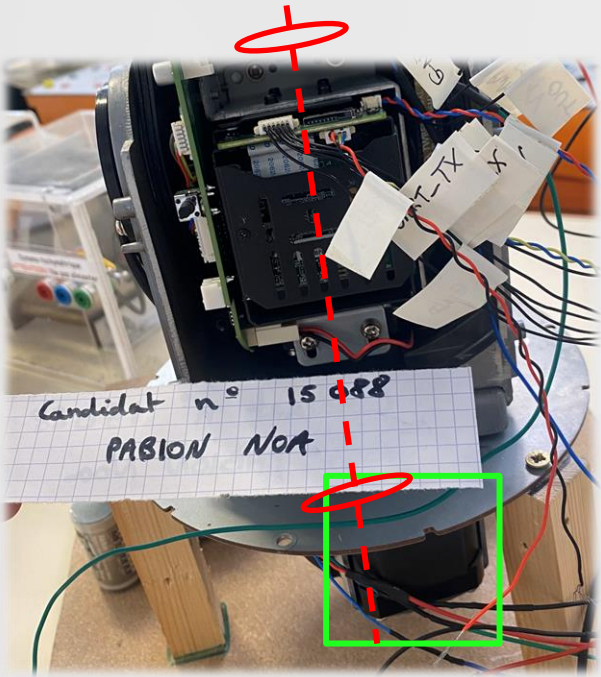


Fonctionnement du système

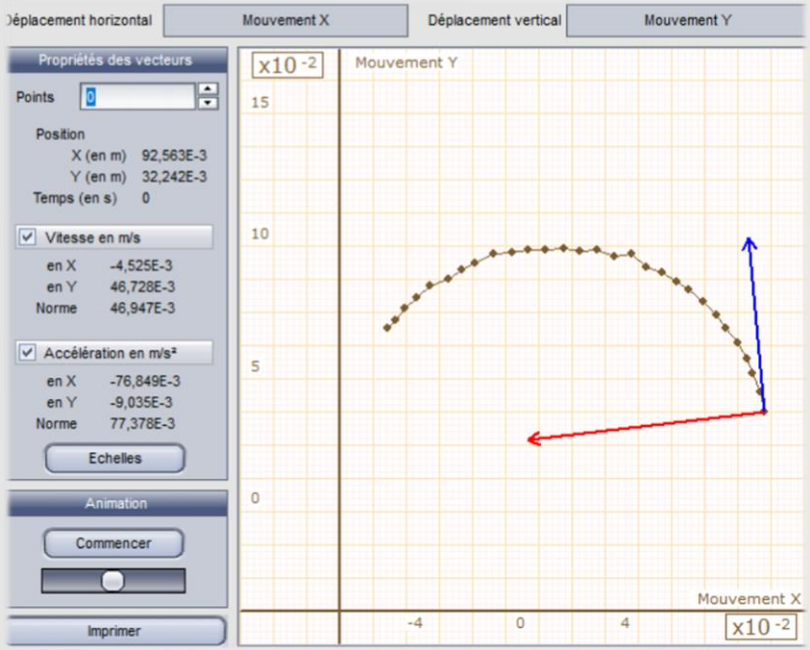


Piloter les moteurs par consigne numérique



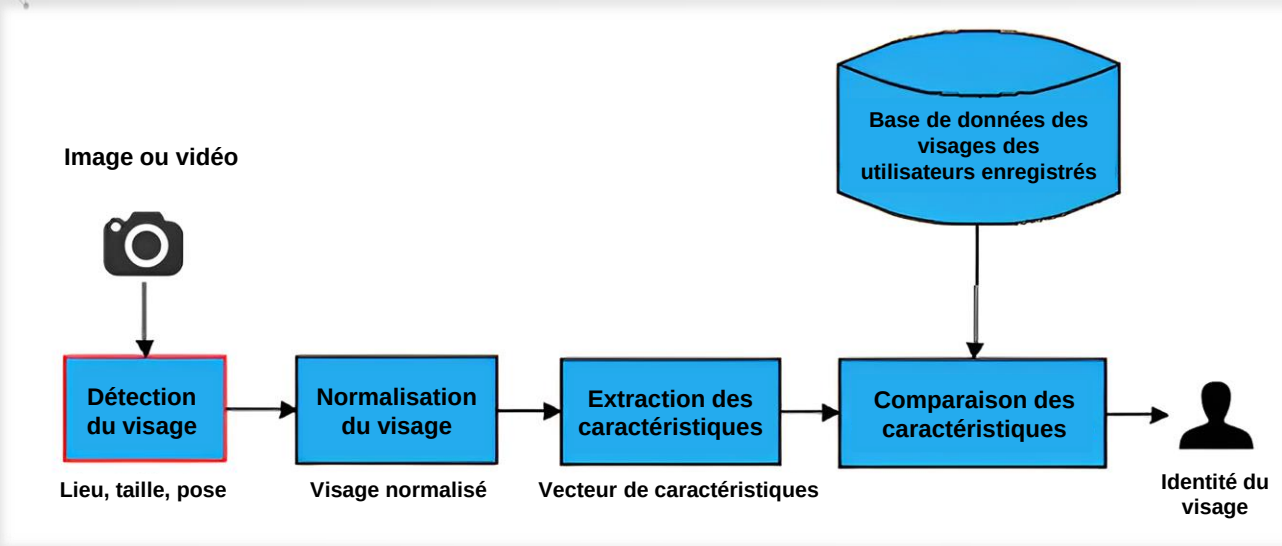


Piloter les moteurs par consigne numérique



Tracé de la position d'un point de la caméra en fonction du temps

Reconnaissance de forme sur une image



http://biblio.univ-antananarivo.mg/pdfs/razafimandimbyManitraT_MP_MAST2_16.pdf

Reconnaissance de forme sur une image

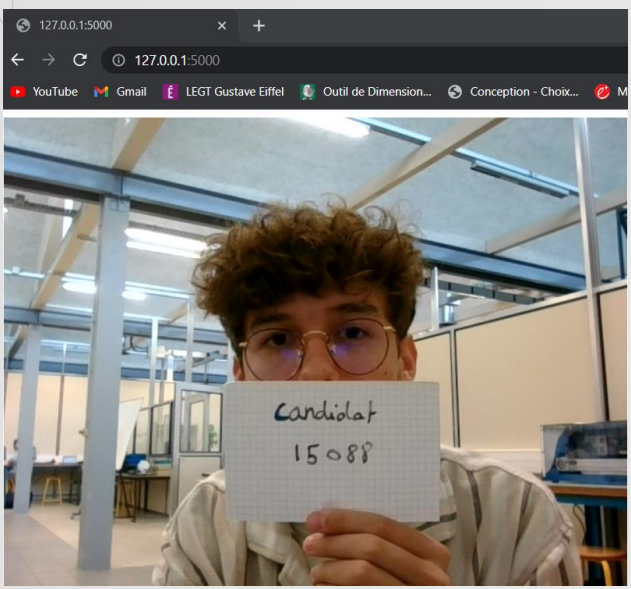
Sortir l'image de la caméra

Internet

Filaire

Bluetooth

Sortir l'image de la caméra

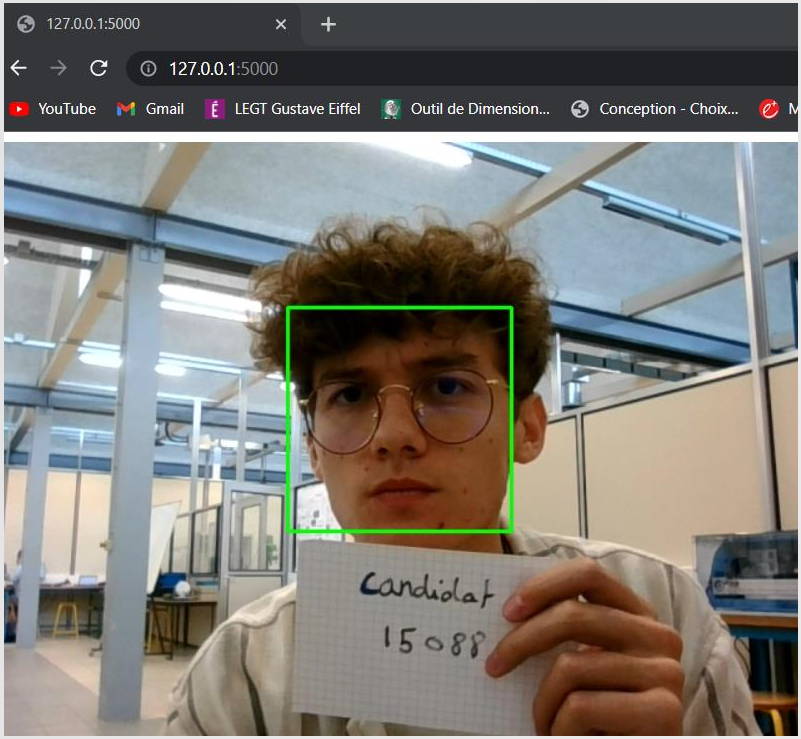


Par internet

```
AttributeError: module 'socket' has no attribute 'BTPROTO_RFCOMM'
```

Par bluetooth (Programme en annexe)

Sortir l'image de la caméra



Addition du programme reconnaissance de forme avec celui de la sortie d'image par internet (voir le programme en annexe)

Conclusion

Merci pour votre attention

Annexe

Sources des différentes fonctions utilisées sur python :

- https://docs.opencv.org/4.x/index.html

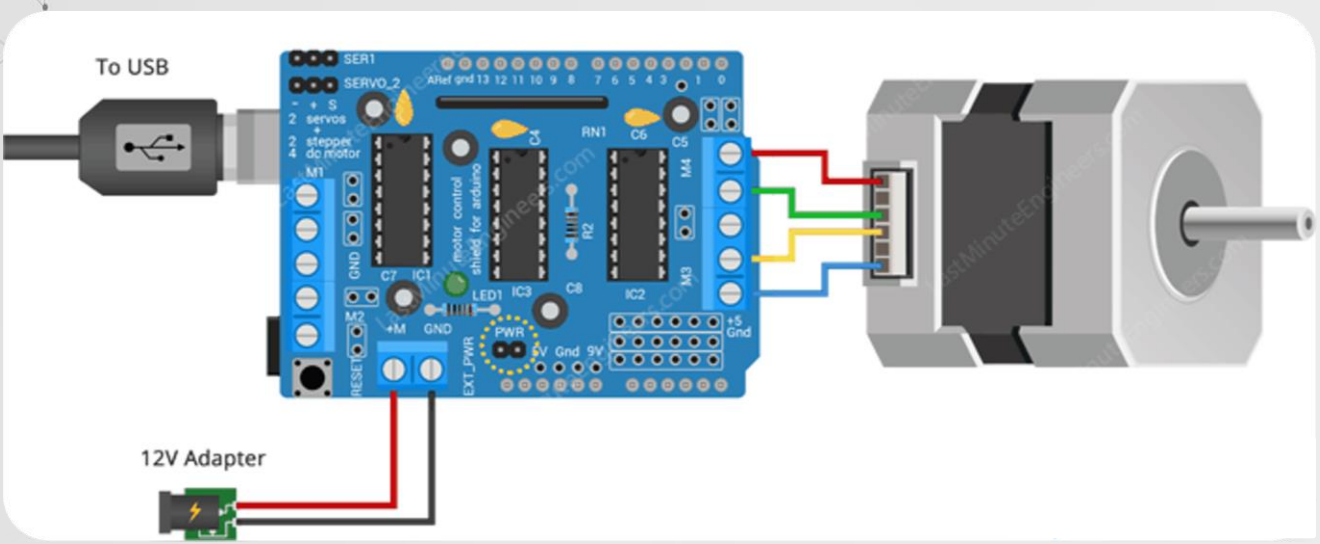
Sources pour le programme arduino :

- https://learn.adafruit.com/adafruit-motor-shield/using-stepper-motors
- https://github.com/adafruit/Adafruit-Motor-Shield-library

Annexe
Programme Arduino

```
//Importer la bibliothèque AdaFruit Moteur Shield
#include <AFMotor.h>

//200 pas par revolution pour les 2 moteurs
AF_Stepper motor1(200, 1);
AF_Stepper motor2(200, 2);

void setup() {
  Serial.begin(9600);
  Serial.println("Test");

  motor1.setSpeed(30);  // tr/min
  motor2.setSpeed(45);  // tr/min

}

void loop() {

  motor1.step(1000, FORWARD, DOUBLE); // Le nombre de pas pour tourner; Avant; Deux bobines sont alimentées à la fois pour plus de couple
  motor1.step(1000, BACKWARD, DOUBLE); // Le nombre de pas pour tourner; Arrière; Deux bobines sont alimentées à la fois pour plus de couple
  motor2.step(500, FORWARD, DOUBLE); // Pareil avec le deuxième moteur
  motor2.step(500, BACKWARD, DOUBLE); // Pareil avec le deuxième moteur

}
```

Annexe
Schéma du branchement du moteur



https://forums.adafruit.com/viewtopic.php?f=31&t=184821

Annexe
Programme Python - Image

```python
#Importer bibliotheque OpenCV
import cv2

#Charger l'image
image = cv2.imread("gens.jpg")

#Creer un detecteur de visages et de corps
Detecte_Visages = cv2.CascadeClassifier(r"C:\Users\PABION Noa\AppData\Roaming\Python\Python38\site-packages\cv2\data\haarcascade_frontalface_default.xml")
Detecte_Corps = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_fullbody.xml")

#Convertir l'image en niveaux de gris pour faciliter la detection
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#Detecter les visages et les corps sur l'image
faces = Detecte_Visages.detectMultiScale(gray, scaleFactor=1.04, minNeighbors=11, minSize=(30, 30))
bodies = Detecte_Corps.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=3, minSize=(150, 150))


#Dessiner des rectangles autour des visages detectes
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 4)


#Dessiner des rectangles autour des corps detectes
for (x, y, w, h) in bodies:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)


#Afficher l'image avec les visages et corps detectes
cv2.imshow("Visages detectes", image)
cv2.waitKey(0)
cv2.imwrite('image.png', image)
```

Annexe
Programme Python - Vidéo

```python
#Importer bibliothèque OpenCV
import cv2

#Créer un détecteur de visages
face_detector = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

#Créer un détecteur de corps
body_detector = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_fullbody.xml")

#Ouvrir la webcam
video = cv2.VideoCapture(0)
#video = cv2.VideoCapture("Video.mp4")

while True:
    #Lire une image à partir de la webcam
    ret, frame = video.read()

    #Convertir l'image en niveaux de gris pour faciliter la détection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #Détecter les visages sur l'image
    faces = face_detector.detectMultiScale(gray, scaleFactor=1.15, minNeighbors=8, minSize=(30, 30))

    #Détecter les corps sur l'image
    bodies = body_detector.detectMultiScale(gray, scaleFactor=1.05, minNeighbors=7, minSize=(100, 200))

    #Dessiner des rectangles autour des visages détectés
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    #Dessiner des rectangles autour des corps détectés
    for (x, y, w, h) in bodies:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

    #Afficher l'image avec les visages et les corps détectés
    cv2.imshow("Detection de visages et de corps", frame)

    #Quitter le programme si la touche 'q' est pressée
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

#Libérer la webcam et fermer la fenêtre d'affichage
video.release()
cv2.destroyAllWindows()
```

Annexe
Programme - Bluetooth avec un appareil

```python
import cv2
import bluetooth
import socket

#Adresse MAC du périphérique Bluetooth
mac_address = "88:A9:B7:46:0F:53"

#Initialisation de la connexion Bluetooth
Connexion = bluetooth.BluetoothSocket(socket.AF_BLUETOOTH, socket.SOCK_STREAM, socket.BTPROTO_RFCOMM)
Connexion.connect((mac_address, 1))

#Capture d'une image de la caméra
cap = cv2.VideoCapture(0)
ret, frame = cap.read()

#Encodage de l'image en format JPEG
img_bytes = cv2.imencode('.jpg', frame)[1].tobytes()

#Envoi des données de l'image sur la connexion Bluetooth
Connexion.sendall(img_bytes)

#Fermeture de la connexion Bluetooth et de la caméra
Connexion.close()
cap.release()
```

Annexe
Programme Python

```python
import cv2
from flask import Flask, Response

#Initialisation de l'application Flask
app = Flask(__name__)

#Créer un détecteur de visages
face_detector = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

#Créer un détecteur de corps
body_detector = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_fullbody.xml")

#Fonction pour accéder à la caméra
def get_frame():
    camera = cv2.VideoCapture(0) #Indiquez le numéro de la caméra à utiliser
    while True:
        success, frame = camera.read()
        ret, frame = camera.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_detector.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5, minSize=(30, 30))
        bodies = body_detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        for (x, y, w, h) in bodies:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

        if not success:
            break
        else:
            #Convertir le frame en bytes pour l'affichage sur le site web
            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
    camera.release(frame)


#Chemin pour afficher le flux vidéo sur le site web
@app.route('/video_feed')
def video_feed():
    return Response(get_frame(), mimetype='multipart/x-mixed-replace; boundary=frame')

#Chemin pour afficher la page web
@app.route('/')
def index():
    return '<html><body><img src="/video_feed"></body></html>'

if __name__ == '__main__':
    app.run(debug=True)
```

FIN