

# Rapport de stage

*Rapport de stage effectué du 28 mai au 5 Juillet*

## Sujet de stage

Programmation d'un robot humanoïde

Thèmes abordés :

- Programmation
- Intelligence artificielle

Tuteur de stage : DESPRES Alberic

Directrice de l'établissement : Magali Pannetier

Entreprise d'accueil : EPSI Rennes, rue Fernand Robert, 35000 Rennes

## Table des matières

<b>Remerciement .....</b>	<b>3</b>
<b>Introduction .....</b>	<b>3</b>
Présentation de l'entreprise : .....	5
Présentation des missions : .....	7
-Programmation du Yanshee (robot humanoïde) .....	7
-Tester un modèle d'IA déjà entraîné et l'implémenter sur le robot : .....	8
L'Objectif : .....	8
L'enjeu : .....	8
Problème à résoudre : .....	9
Contraintes : .....	9
<b>Démarche suivie pour effectuer ces missions .....</b>	<b>10</b>
Yanshee app : .....	10
Programmer dans le raspberry pi : .....	11
Yanshee SDK : .....	12
Open ADK : .....	13
RESTful API : .....	13
Organisation : .....	14
<b>Réalisation des missions .....</b>	<b>15</b>
<b>Programmation .....</b>	<b>16</b>
Explication détaillée du code.....	17
Fonctions créées .....	18
Exemple d'un programme et sa réponse .....	19
<b>Documentation automatique (swagger) .....</b>	<b>20</b>
<b>Autre mission : Tester un modèle d'IA déjà entraîné .....</b>	<b>22</b>
Explications.....	22
Exemples concret .....	23
<b>Combinaison des deux missions : .....</b>	<b>26</b>
<b>Conclusion .....</b>	<b>27</b>
Résumé.....	27
Technologies découvertes .....	27
Ressenti du stage.....	27
Contribution à l'EPSI .....	28
Ouverture sur la suite/projet professionnelle.....	28
<b>Annexe .....</b>	<b>29</b>

## Remerciement

Je tiens à exprimer ma profonde gratitude envers l'École Privée des Sciences Informatiques (EPSI) de Rennes pour m'avoir offert l'opportunité de réaliser ce stage. Cette expérience épanouissante a été possible grâce à l'accueil bienveillant et au soutien constant de l'équipe pédagogique de l'EPSI.

Je souhaite également remercier tout particulièrement mon tuteur de stage, Albéric DESPRES, pour ses conseils avisés et son accompagnement tout au long de cette période. Ses partages d'expérience et son expertise ont été important pour mon apprentissage et mon développement professionnel.

Merci à tous ceux qui ont contribué à rendre ce stage aussi formateur et enrichissant, cela m'a permis de gagner en expérience dans un domaine qui me plaît, et d'acquérir des connaissances qui me permettront d'aller de l'avant.

Remerciement spécial pour Alexia : merci de m'avoir emmené à l'Hospital ce jeudi 13 juin

## Introduction

EPSI, école d'ingénierie informatique de référence en France, se distingue par son approche innovante de la formation en informatique et ses partenariats avec les entreprises. Située à Rennes, l'EPSI propose des cursus axés sur les métiers du numérique et des technologies de l'information, et se positionne comme un acteur majeur dans la formation des futurs professionnels de l'IT.

Dans le cadre de ma formation en programmation et développement informatique, j'ai eu l'opportunité de réaliser un stage de 5 semaines au sein du campus, du 28 mai au 5 juillet. L'EPSI Rennes est réputée pour son environnement académique stimulant et ses projets collaboratifs, offrant aux étudiants un espace pour développer leurs compétences techniques et professionnelles.

Durant mon stage, j'ai été intégré dans l'équipe pédagogique et technique de l'EPSI. Mon projet principal consistait à programmer le robot Yanshee et ses différentes capacités. Ce robot est doté de nombreuses fonctionnalités avancées telles que la reconnaissance faciale, la synthèse vocale, mouvement articulé, etc..., ce qui en fait un outil pédagogique et technologique de haute qualité

L'objectif de ma mission principale était en trois étape : tout d'abord, comprendre et maîtriser les fonctionnalités du robot Yanshee ; ensuite, apprendre à le programmer à distance; et enfin, proposer des programmes originaux exploitant les capacités du robot.

Ce rapport consiste à détailler l'ensemble de mon expérience à l'EPSI Rennes.

#### Durée du stage :

-Durée de la mission : Du mardi 28/05 au vendredi 05/07 (5 semaines)

## Présentation de l'entreprise :

L'EPSI (École Privée des Sciences Informatiques) de Rennes est une école d'enseignement supérieur privée, spécialisée dans la formation en informatique et technologies de l'information. L'école vient du groupe « compétence et développement » qui détient aussi les écoles WIS, l'IFAG ou encore l'école 3A.

L'école est située à proximité de Villejean, sur le site de The Land.

La première école a été créée en 1961, à Paris, c'est une des premières écoles en France à offrir des formations dédiées aux métiers de l'informatique. Le campus de Rennes fait partie du réseau national EPSI, qui compte plusieurs écoles à travers le pays.



L'objectif principal de l'EPSI est de former des professionnels de l'informatique compétents et adaptables, prêts à répondre aux exigences du marché du travail en constante évolution. L'école met un accent particulier sur l'apprentissage pratique, en intégrant des projets réels et des stages en entreprise dans ses programmes.

### *Branche d'Activité :*

L'EPSI de Rennes se spécialise dans l'éducation et la formation professionnelle dans le domaine des technologies de l'information et de la communication. L'école forme des

experts en informatique, en développement logiciel, en administration réseau, en cybersécurité, en gestion de projets informatiques, et en intelligence artificielle.

#### *Offres de Formation :*

L'EPSI de Rennes propose une gamme complète de programmes de formation allant du niveau Bac+3 au Bac+5, conçus pour répondre aux besoins variés des étudiants et des entreprises.

#### *Vie Étudiante et Réseautage :*

L'école encourage l'engagement des étudiants dans des clubs et associations liés à l'informatique, à l'innovation, et à l'entrepreneuriat (comme le BDE par exemple).

Elle organise aussi régulièrement des conférences et des ateliers animés par des professionnels de l'industrie, offrant aux étudiants des opportunités de réseautage et d'apprentissage en dehors des cours traditionnels.

#### *Personnel de l'EPSI de Rennes :*

Albéric DESPRES : Coach MyDIL (s'occupe du MyDIL

Marion HUREAU : Assistante polyvalente

Nicolas ZAIMECHE : Responsable pédagogique  
Magali PANNETIER - Directrice du campus EPSI Rennes.

Danaé HENRY : Chargé des relations BtoB et développement commercial

Alexia NDEBEKA : Chargée de développement et relations entreprises

## Présentation des missions :

### -Programmation du Yanshee (robot humanoïde)

- Python
- Prise en main
- Proposition d'un programme original (possibilités multiples)



### Documents utiles :

- [Développeur Yanshee \(ubtrobot.com\)](http://ubtrobot.com) (site officiel du robot yanshee)
- [Yanshee – UBTECH Education](#)
- [Annexe 1 — Guide de l'utilisateur de l'API Yanshee Robot SDK · UBTECH/YanShee-Curriculum Wiki · Sur GitHub](#) (doc de l'API de Yanshee SDK)
- [GitHub - UBTECH/Yanshee-SDK](#) (lien github de Yanshee SDK)
- [https://github.com/UBTECH/Yan\\_SDK/blob/develop/docs/](https://github.com/UBTECH/Yan_SDK/blob/develop/docs/) (doc openADK)
- [Yanshee/Yanshee-Doc \(gitee.com\)](#) (utile pour l'API RESTful)

Les documents ci-dessus sont les plus importants pour la programmation du robot. En prenant en compte le peu d'utilisation de ce robot dans le monde, on y trouve très peu d'informations autre que ses créateurs à son sujet.

-Tester un modèle d'IA déjà entraîné et l'implémenter sur le robot :

- Détection d'objets

Document utile :

-Détection d'objet : <https://www.aranacorp.com/fr/reconnaissance-dobjet-avec-python/>

Le robot est programmable en plusieurs langues :

- Python
- Java
- C#
- Javascript
- Perl
- CURL
- PHP

### L'Objectif :

- l'objectif principal de ma mission était de me « familiariser » avec le robot Yanshee et la programmation associée. Cela impliquait une compréhension approfondie de toutes ses fonctionnalités et de son utilisation. Je devais aussi et surtout programmer le robot à distance en utilisant un des nombreux moyen existant (que j'ai découvert tout au long du stage) , permettant une interaction et un contrôle à distance. En outre, un autre objectif important était de proposer et de développer un programme original exploitant les multiples possibilités déjà intégré dans le robot. Cela comprenait la conception de nouvelles fonctionnalités et l'amélioration des capacités existantes.

### L'enjeu :

- L'enjeu principal pour l'EPSI de Rennes était de maximiser l'utilisation du robot Yanshee comme outil pédagogique innovant. En programmant de nouvelles fonctionnalités et en améliorant les capacités existantes du robot, l'école souhaitait offrir à ses étudiants des expériences d'apprentissage pratiques et interactives.



### Problème à résoudre :

Lors de mon stage et avec beaucoup de pratique du robot, j'ai pu constater plusieurs problèmes :

- **Uniquement 2 langues disponibles (en et zh) :** Le robot Yanshee ne supporte actuellement que l'anglais et le chinois, limitant son utilisation à un public restreint.
- **Manque de stabilité :** le robot peut tomber s'il marche trop vite, par exemple
- **Capacité de reconnaissance visuelle limitée :** Les taux de précision pour la reconnaissance d'objets et de couleurs sont insuffisants.
- **Manque de programmes originaux dans les API :** Les API fournies manquent de programmes originaux, limitant les possibilités d'innovation et de personnalisation par les utilisateurs.

### Contraintes :

- **Temps limité :** La mission devait être réalisée dans le temps imparti du stage, nécessitant une gestion rigoureuse du temps et des priorités pour assurer la complétion des tâches essentielles.
- **Compatibilité logicielle :** Les nouvelles fonctionnalités devaient être intégrées de manière fluide avec les systèmes existants sans causer de conflits ou de dysfonctionnements.
- **Ressources limitées :** L'accès à certaines ressources, telles que les bases de données ou les outils logiciels avancés, pouvait être limité, nécessitant des solutions alternatives ou des adaptations
- **Contraintes techniques :** Le robot Yanshee a des capacités matérielles limitées, comme la puissance de traitement et les spécifications de la caméra intégrée, ce qui impose des restrictions sur les types et la complexité des programmes pouvant être développés.

## Démarche suivie pour effectuer ces missions

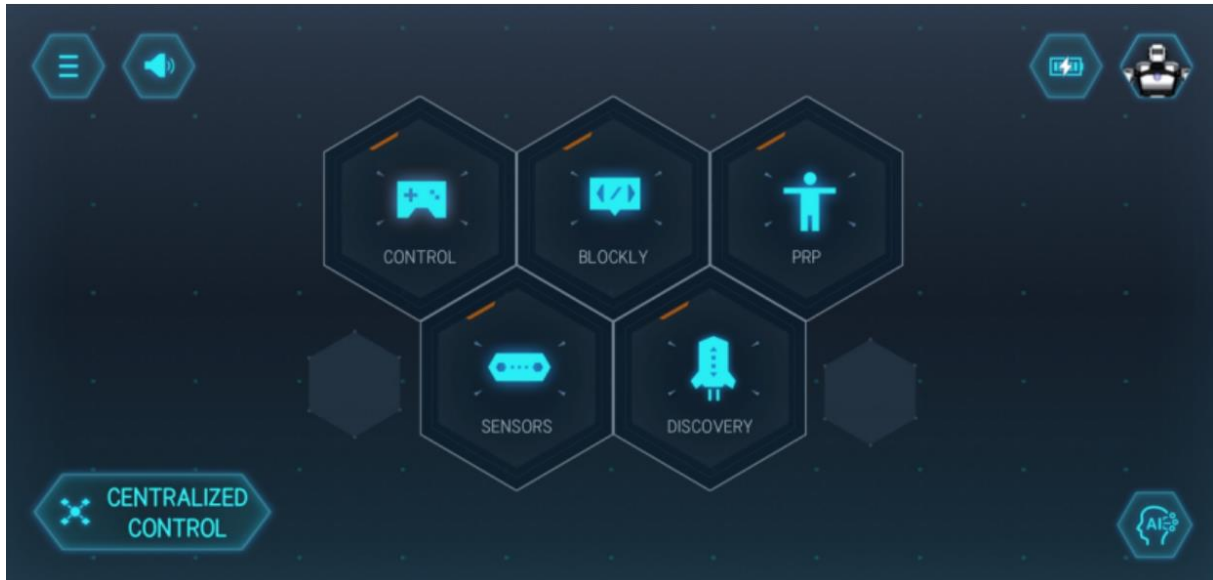
Pour programmer le robot il y a de nombreux moyens, voici ce que j'ai retenu tout au long de mon stage :

- 1- Le moyen le plus simple et rapide est via l'application Yanshee, disponible sur téléphone ou tablette.
- 2- Un autre moyen de programmer le robot est directement via son Raspberry Pi. Pour cela il faut avoir au minimum :
  - Un câble HDMI
  - Un écran
  - Une souris
  - Un clavier
- 3- Le troisième moyen de pouvoir programmer le robot est via Yanshee-SDK, qui est obtainable sur le lien github page 5 sous « document utiles : ». Il permet d'envoyer des programmes depuis votre ordinateur, à distance.
- 4- Un autre moyen de pouvoir programmer le robot à distance : grâce à l'API openADK.
- 5- La dernière méthode que j'ai utilisée pour programmer le robot est en me connectant via API RESTful.

### Yanshee app :

Cette application est très utile pour se familiariser avec le robot, on peut pratiquement tout faire dessus, c'est une bonne manière de débiter la programmation, notamment

avec « blockly », très similaire à Scratch, ce qui permet de créer des petits programmes pour mieux comprendre le robot. L'application est vraiment complète, on peut faire des choses comme faire danser le robot, lui faire faire des actions comme lever le bras, bouger la tête, frapper avec le pied ou le poing, et bien d'autres encore. On peut aussi obtenir toutes les informations sur lui, comme son IP par exemple, qui nous sera très utile par la suite.

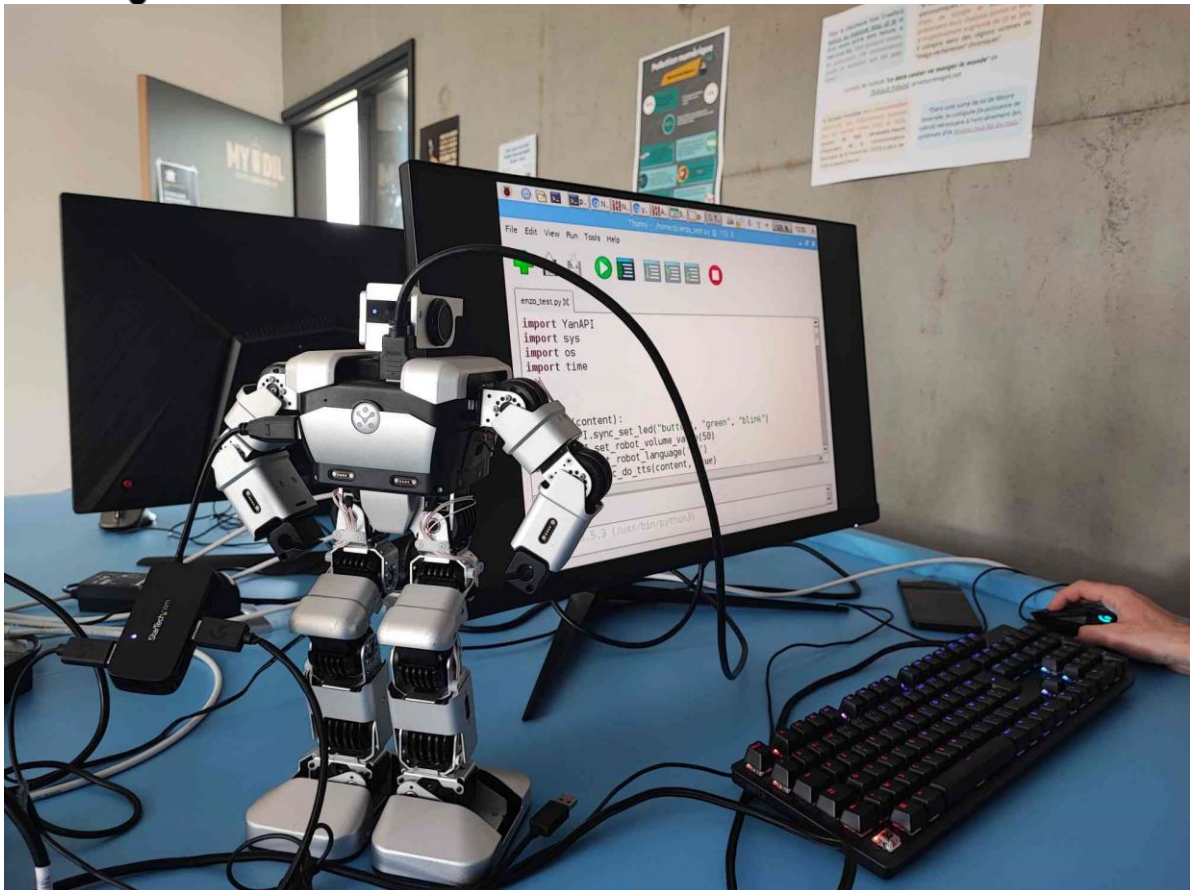


## Programmer dans le raspberry pi :

Programmer directement dans le robot est aussi une solution pour comprendre un peu mieux comment le robot fonctionne, j'ai choisi de coder en python pour plusieurs raisons :

- Cela me permet de continuer mes compétences acquises lors de cette année grâce au module Python
- Python était pour moi le choix le plus pertinent, car grâce à l'API YanAPI directement intégré, on peut faire appelle à des fonctions qui nous permettront de mieux programmer le robot.

Le seul inconvénient est qu'il faut qu'il soit constamment branché, (HDMI + souris + clavier), ce qui est encombrant car il n'y a pas beaucoup de ports sur le robot, et de plus on est limité en termes de mouvement, car le robot est bloqué avec les câbles, il risque de s'endommager ou de briser les câbles.



## Yanshee SDK :

La méthode Yanshee SDK a été ma première tentative de programmer le robot à distance, et après avoir tester d'autres méthode, je déconseille très fortement d'utiliser Yanshee-SDK pour plusieurs raisons :

- Après plusieurs jours de tentative de connexion sans réussite, même avec l'aide de mon maitre de stage, j'en ai déduit qu'utiliser cette méthode était une erreur (et heureusement que j'apprends de mes erreur). Cela a été beaucoup de temps de perdu, pour finalement quasiment aucun résultat, à chaque problème résolu, d'autre apparaissait. C'était une boucle sans arrêt, j'en ai conclu au final qu'il y avait des problèmes dans les fichiers téléchargé depuis le git (on a dû rajouter quelques lignes dans les fichiers pour pouvoir avancer) Néanmoins cela m'a permis de me familiariser avec WSL, qui m'est très utile pour le stage et qui me le sera à l'avenir (j'expliquerai plus en détail par la suite sa définition et son utilité).
- Les fonctionnalités disponible grâce à Yanshee SDK étaient moindre que les deux Prochaines méthodes que j'ai utilisées pour me connecter à distance, je n'avais donc plus aucune raison de continuer avec cette dernière.

## Open ADK :

Grâce à Open ADK, j'ai pu découvrir de nombreuses fonctionnalités qui m'ont permis de vraiment mieux comprendre comment je pouvais programmer le robot à distance.

Pour pouvoir utiliser cette méthode, j'ai eu recours à WSL. Mais qu'est-ce que WSL ? :

« Le Sous-système Windows pour Linux ou Windows Subsystem for Linux (WSL) est une fonctionnalité de Windows qui vous permet d'exécuter un environnement Linux sur votre machine Windows, sans avoir besoin d'une machine virtuelle séparée ou d'un double démarrage. WSL est conçu pour offrir une expérience transparente et productive aux développeurs qui souhaitent utiliser Windows et Linux en même temps. »

Source : [Qu'est-ce que le Sous-système Windows pour Linux ? | Microsoft Learn](#)

Par la suite, après quelques commande Linux et quelques téléchargements, j'ai pus commencer à coder depuis mon IDE (vs code) en python sur mon pc, sans être branché au robot. J'ai choisi python pour les mêmes raisons que je l'ai utilisé sur le Raspberry Pi.

Pour ce faire, j'ai dû me connecter au même réseau que celui du robot, en l'occurrence « The Land ». Une fois que c'est fait, on peut aller récupérer l'IP du robot (de plusieurs façons différentes, je l'ai personnellement récupéré directement dans le Raspberry grâce à la commande « ip a » dans l'invité de commande.

Pendant toute la durée de mon stage, j'ai utilisé WSL. Sauf lorsque j'ai rencontré un problème lors de mon autre mission (voir « Tester un modèle d'IA déjà entraîné » et les implémenter sur le robot »)

## RESTful API :

Cette méthode utilise des requêtes HTTP pour communiquer avec un robot, en envoyant et recevant des données au format JSON. L'API RESTful est un moyen standardisé de permettre à différentes applications de communiquer entre elles de manière cohérente et sécurisée.

Pareil que pour les autres méthodes, j'ai toujours continué d'utiliser python, et toujours pour les mêmes raisons.

## Organisation :

Je n'avais pas vraiment d'organisation particulière au début, je touchais un peu à tout pour me familiariser avec le robot, mais petit à petit j'ai commencé à me fixer quelques objectifs à faire étape par étape, et je ne faisais pas autre chose tant que je n'avais pas réussi à faire cette étape (sauf exception avec Yanshee-SDK)

Si je devais dresser une liste des étapes que j'ai suivi tout au long de mon stage, cela donnerait ça (approximativement dans l'ordre) :

- Installer l'application Yanshee app et découvrir toutes les fonctionnalités du robot.
- Créer quelques programmes avec Blockly de l'application Yanshee app, pour avoir une première approche de « programmation » avec le robot.
- Faire les installations nécessaires sur l'ordinateur de robot, et faire quelques programmes directement dans le Raspberry Pi
- Découvrir de quoi le robot est capable en testant un maximum de fonctionnalité, entre les servos, les mouvements, la vision, l'appareil (device), etc...
- Chercher des solutions pour pouvoir programmer le robot à distance
- Une fois les solutions trouvées, comprendre le fonctionnement de ces derniers, et en déduire quelle est la meilleure solution pour moi.
- Après avoir assimiler la « meilleur » méthode pour programmer le robot à distance, je peux commencer à créer quelques programmes.
- Un problème a été rencontré : les façons d'envoyer les programmes sont long  
Solution : créer moi-même des fonctions dans un fichier « fonctions.py » que je pourrais utiliser autant de fois que je veux, aux lieux de recommencer à écrire tout manuellement.
- Enfin, tenter d'intégrer une IA qui a une reconnaissance d'objet, dans le robot.



Après 2 semaines de pratique du robot, j'en ai conclu que pour moi, la meilleure méthode pour programmer le robot était grâce à l'API RESTful pour plusieurs raisons :

- Facilité d'utilisation : La bibliothèque requests en Python est simple à utiliser et nécessite peu de configuration. Cela m'a permis de mieux me concentrer sur la logique du programme plutôt que les détails techniques des communications réseau.
- Lecture et écriture claires : Les requêtes et les réponses sont en format JSON, qui est facile à lire et à écrire.
- Compatibilité : L'API RESTful est basée sur des standards web (HTTP, URL, JSON), ce qui la rend compatibles avec de nombreuses plateformes et langages de programmation (en l'occurrence python).
- Modularité : L'API RESTful permet de découper les fonctionnalités du robot en différentes ressources accessibles par des URL distinctes. Cela rend le code plus modulaire et plus facile à comprendre et à utiliser.

J'ai donc décidé de continuer à programmer avec cette méthode, ce qui m'a beaucoup plu et qui m'a appris beaucoup de choses.

## Programmation

Voici un exemple de code qui permet de faire parler le robot (il va dire « Hello world ») :

```
import requests
import json
# URL
url = 'http://10.35.96.250:9090/v1/voice/tts'

# en tete de la requete
headers = {
    'Content-Type': 'application/json',
    'Accept': 'application/json'
}
# données envoyées au robot
data = {
    "interrupt": True,
    "timestamp": 99,
    "tts": "Hello world !"
}
#envoi de la requete
try:
    response = requests.put(url, headers=headers, data=json.dumps(data))
    # écrire la réponse obtenu, et afficher le message d'erreur si la requête n'a pas abouti
    if response.status_code == 200:
        print("Request was successful.")
        print("Response data:", response.json())
    else:
        print(f"Failed to send request. Status code: {response.status_code}")
        print("Response text:", response.text)
except requests.exceptions.RequestException as e:
    print(f"An error occurred: {e}")
```





On peut remarquer que le code est facilement coupable en « morceau ».

- 1- Import des bibliothèques :
  - a. « requests » permet d'envoyer les requêtes http en python.
  - b. « json » permet de travailler avec des données au format JSON
- 2- URL : C'est l'endroit où la requête HTTP sera envoyée, La variable `url` contient l'adresse de l'API RESTful du robot Yanshee (pour la synthèse vocale en l'occurrence, TTS = Text To Speech)
- 3- En tête de la requête : Cela indique que les requêtes et les réponses seront au format JSON.
- 4- Data : Ce dictionnaire contient les données envoyées dans le corps de la requête. Lorsque vous envoyez une requête en rapport avec une méthode (par exemple, imaginons que je veux faire bouger le robot, j'aurais donc besoins de la méthode « motion »), j'aurais besoins d'utiliser tous les paramètres nécessaires pour faire bouger le robot, dans le bon ordre, et avec les bons types, sinon il y aura un message d'erreur (voir 7)
- 5- Envoie de la requête : la variable « response » va envoyer la requête au robot, il enverra tout ce qu'on a vu précédemment.
- 6- Affichage des réponses obtenu : Si la requête a été envoyé avec succès, alors il sera affiché un message de succès. Dans certains cas, avec certaines méthodes, le robot retourne en réponse des données (comme par exemple son pourcentage de batterie ou la couleur de ses LED), c'est pour cela qu'il est important dans tous les cas d'afficher la réponses.
- 7- Afficher la ou les erreurs : La dernière partie du code sert à afficher un message d'erreur si la requête n'a pas abouti.

## Fonctions créées

En revenant sur la partie « organisation » dans « **Démarche suivie pour effectuer cette ou ces missions** », j'ai évoqué le fait de créer des fonctions pour gagner du temps et que la programmation soit plus organisée et « propre »

Voici donc une liste des fonctions que j'ai créé manuellement :

(Voir annexe pour le code des fonctions)

-fonction **tts** : permet de parler

*Paramètre : string ('Hello world !')*

-fonction **start\_motion** : permet de faire bouger le robot

*Paramètre : direction, nom du mouvement, nombre de répétition, vitesse*

*(voir « Liste des mouvements et de leurs paramètres disponibles » page 7)*

-fonction **reset** : permet de mettre le robot en position « normal »

*Note : ne pas reset le robot trop vite après un mouvement, sinon il va tendre les bras puis après se reset*

-fonction **get\_servos\_angle** : obtenir les angles des servos

-fonction **take\_photo\_age** : prend une photo et donne l'âge à l'oral

*Note : faites attention de bien rester devant la caméra le temps qu'il prenne la photo, ne soyez pas trop loin (50cm environ)*

-fonction **take\_photo** : prend une photo et la sauvegarde sur le pc, dans le dossier « photos »

-fonction **color\_recognition** : prend une photo et donne la couleur à l'oral

-fonction **put\_led** : permet de changer la couleur et le mode des led de la caméra et du bouton.

*Paramètre: mode : ['off', 'on', 'breath', 'blink']*

*Couleur : ['white', 'red', 'green', 'blue', 'yellow', 'purple', 'cyan']*

*Type : ['camera', 'button']*

*Note : Le type "camera" n'accepte que « on » et « off » comme mode.*

-fonction **alive** : bouge aléatoirement la tête, avec un intervalle de temps aléatoire, et cligne des yeux avec un intervalle aléatoire aussi aléatoire

## Exemple d'un programme et sa réponse

Exemple de code pour prendre obtenir les angles de tous les servos :

```
def get_servo_angles():
    url = 'http://10.35.96.250:9090/v1/servos/angles'
    servo_names = [
        "LeftAnkleFB", "LeftAnkleUD", "LeftElbowFlex", "LeftHipFB",
        "LeftHipLR", "LeftKneeFlex", "LeftShoulderFlex", "LeftShoulderRoll",
        "NeckLR", "RightAnkleFB", "RightAnkleUD", "RightElbowFlex",
        "RightHipFB", "RightHipLR", "RightKneeFlex", "RightShoulderFlex",
        "RightShoulderRoll"
    ]
    names_param = ','.join(servo_names)
    params = {'names': names_param}
    headers = {
        'Accept': 'application/json'
    }
    try:
        response = requests.get(url, headers=headers, params=params)
        if response.status_code == 200:
            print("Request was successful.")
            angles = response.json()
            for name, angle in angles.items():
                print(f'"{name}": {angle},')
        else:
            print(f"Failed to send request. Status code: {response.status_code}")
            print("Response text:", response.text)
    except requests.exceptions.RequestException as e:
        print(f"An error occurred: {e}")
```

Ce qui donnera comme réponse :

```
enzo@EnzoRn:/mnt/c/Users/enzor/Yanshee-SDK/output/python/example$ /bin/python /mnt/c/Users/enzor/Yanshee-SDK/output/python/example/functions_test.py
Request was successful.
"code": 0,
"data": {'LeftAnkleFB': 70, 'LeftAnkleUD': 90, 'LeftElbowFlex': 15, 'LeftHipFB': 120, 'LeftHipLR': 90, 'LeftKneeFlex': 104, 'LeftShoulderFlex': 40, 'LeftShoulderRoll': 90, 'NeckLR': 90, 'RightAnkleFB': 110, 'RightAnkleUD': 90, 'RightElbowFlex': 165, 'RightHipFB': 60, 'RightHipLR': 88, 'RightKneeFlex': 76, 'RightShoulderFlex': 140, 'RightShoulderRoll': 90},
"msg": success,
```

Dans un autre programme, on a la possibilité de modifier ces données pour faire bouger le robot « manuellement », ce qui nous ouvre une infinité de possibilités.

## Documentation automatique (swagger)

En suivant les instructions sur le lien github, on a aussi accès à swagger ui, qui permet d'avoir une doc « fais maison » avec toutes les fonctionnalités du robot.


Il suffit de rentrer cette url dans le navigateur :

<http://10.35.96.250:9090/v1/ui/>

Bien sûr, si l'IP du robot n'est pas celle-ci, il suffit juste de la changer. En revanche, il ne faut pas changer le port « 9090 ».

Si tout à été suivi à la lettre et qu'il n'y a pas eu d'erreur, on arrive sur une page avec plusieurs choix de méthodes, qui sont toute utilisable sur le robot.

voici à quoi ressemble la doc en ligne :

 **swagger**

**Explore**

### Yanshee RESTful API

**概述**

Yanshee RESTful APIs是由使用swagger-codegen基于 OpenAPI-Spec的工程。所有的API由Flask的Connexion来解释。Yanshee-ROS中的apollo提供所有的RESTful APIs服务。同时Yanshee RESTful APIs提供英文和中文两个版本:

- [英文版本](#)
- [中文版本](#)

**依赖**

Python 2.7 or 3.4+

Find out more about Swagger

<http://swagger.io>  
[Contact the developer](#)  
[Apache 2.0](#)

<b>devices : 系统参数</b>	Show/Hide	List Operations	Expand Operations
<b>media : 多媒体</b>	Show/Hide	List Operations	Expand Operations
<b>motions : 动作</b>	Show/Hide	List Operations	Expand Operations

Et voici un exemple des fonctions possible avec la photo :

visions : 计算机视觉		Show/Hide	List Operations	Expand Operations
GET	/visions	获取任务结果		
PUT	/visions	指定视觉任务停止或开始		
DELETE	/visions/photos	删除指定照片		
GET	/visions/photos	获取指定照片		
POST	/visions/photos	拍一张照片		
GET	/visions/photos/list	获取拍照列表		
DELETE	/visions/photosamples	删除上传的样本		
GET	/visions/photosamples	获取上传样本列表		
POST	/visions/photosamples	上传样本		
DELETE	/visions/streams	关闭摄像头的视频流		
POST	/visions/streams	打开摄像头的视频流		
DELETE	/visions/tags	移除指定标签或者标签和样本		
GET	/visions/tags	获取已设置标签列表		
PUT	/visions/tags	设置样本标签		

Toute les lignes ci-dessus ont des particularités et des fonctions différentes, qui sont toute utilisable sur le robot. Lorsque l'on clique sur une des lignes, on obtient plus d'informations, ce qui nous guide beaucoup plus facilement.

Voici par exemple ce qui va permettre d'activer la vision du robot :

PUT

/visions

指定视觉任务停止或开始

Implementation Notes

规则

- 规则一: 当视频流开启, 其他视觉任务不能被开启
- 规则二: 当其他视觉任务开启, 视频流任务不能被开启
- 规则三: 相同任务或者被认为是相同任务, 不能同时开启两次, 此规则适用于语音触发的视觉任务
- 规则四: 视频流任务无须遵守规则三

Response Class (Status 200)

发送成功

Model | Example Value

```
{
  "code": 0,
  "data": "{}",
  "msg": "msg"
}
```

Response Content Type

application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	(required)		body	Model   Example Value

Parameter content type:

application/json

```
{
  "operation": "start",
  "option": "face",
  "timestamp": 1551838515,
  "type": "tracking"
}
```

Try it out!

## Autre mission : Tester un modèle d'IA déjà entraîné

### Explications




Après avoir regardé le site qui proposait d'avoir sa propre IA qui reconnaît les objets en python sur son ordinateur, j'ai suivi les instructions et j'ai réussi à avoir quelques résultats.

Pour commencer, j'ai dû installer les bibliothèques suivante grâce à cette commande :

```
pip3 install opencv-python numpy imutils
```

J'ai ensuite téléchargé 2 fichiers importants pour réaliser cette mission, et j'ai créé un fichier « ObjectRecognition.py »

Voilà à quoi ressemble mon dossier où je travaille (le screen ci-dessous n'est pas le mien, il provient du site internet, mais les fichiers sont exactement les mêmes)

Nom	Modifié le	Type	Taille
 MobileNetSSD_deploy.caffemodel	25/12/2021 23:57	Fichier CAFFEMO...	22 606 Ko
 MobileNetSSD_deploy.prototxt.txt	28/06/2021 03:35	Document texte	29 Ko
 ObjectRecognition.py	26/12/2021 15:23	Python File	4 Ko

Par la suite j'ai continué de faire ce qui était demandé, j'ai regardé le code qui était proposé, je l'ai compris et j'ai pu faire mes premières tentatives de reconnaissance d'objet.

Il y a plusieurs moyens de faire de la détection :

- 1- Depuis une photo
- 2- Depuis une vidéo
- 3- Depuis la caméra de l'ordinateur
- 4- Depuis la caméra d'un Raspberry Pi

Tous les moyens ci-dessus fonctionnent différemment, et auront besoin de modification dans le code pour que ces derniers fonctionnent.

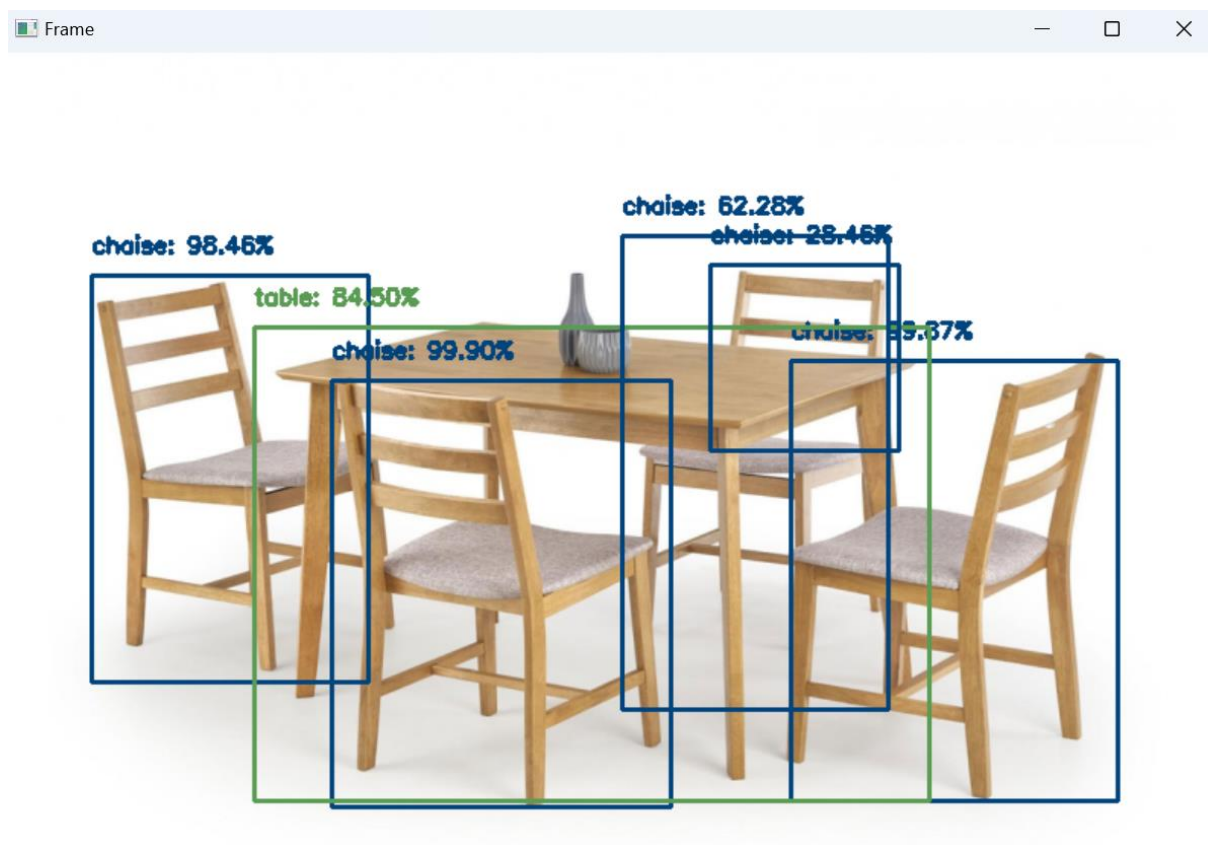
Voici ce qu'il faut modifier pour pouvoir détecter selon nos besoins.

```
#vs = VideoStream(src=0, resolution=(1600, 1200)).start()
#vs = VideoStream(usePiCamera=True, resolution=(1600, 1200)).start()
#vc = cv2.VideoCapture('./img/cars.mp4') #from video
```

## Exemples concret

### Photo :

Voici une photo enregistré dans mon PC directement après être passé par la détection d'objet.



On peut voir que les objets en premier plan ont beaucoup plus de probabilité d'être bien détecté que ceux du fond.

Mais on peut rencontrer quelques problèmes comme ci-dessous :

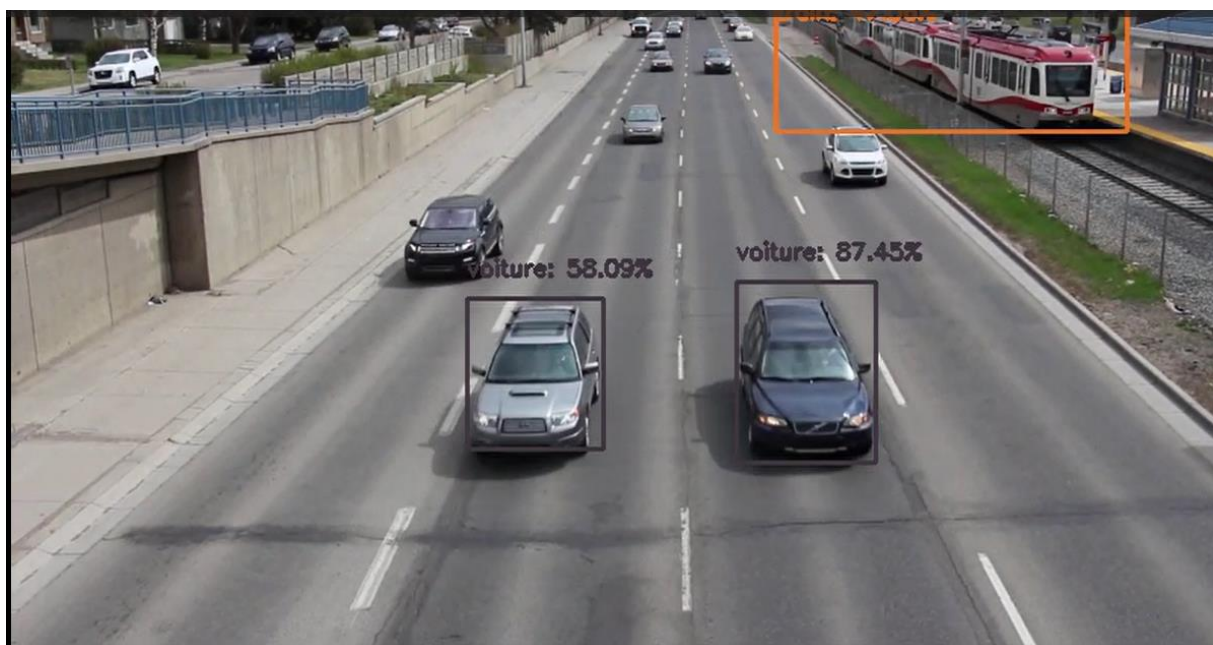




L'IA n'est pas encore assez entraînée, il est sûr à 95% que c'est un chat alors qu'il se trompe, c'est un chinchilla. On ne peut pas s'y fier à 100%.

### Vidéo :

Voici une capture d'écran de la vidéo après la détection d'objet



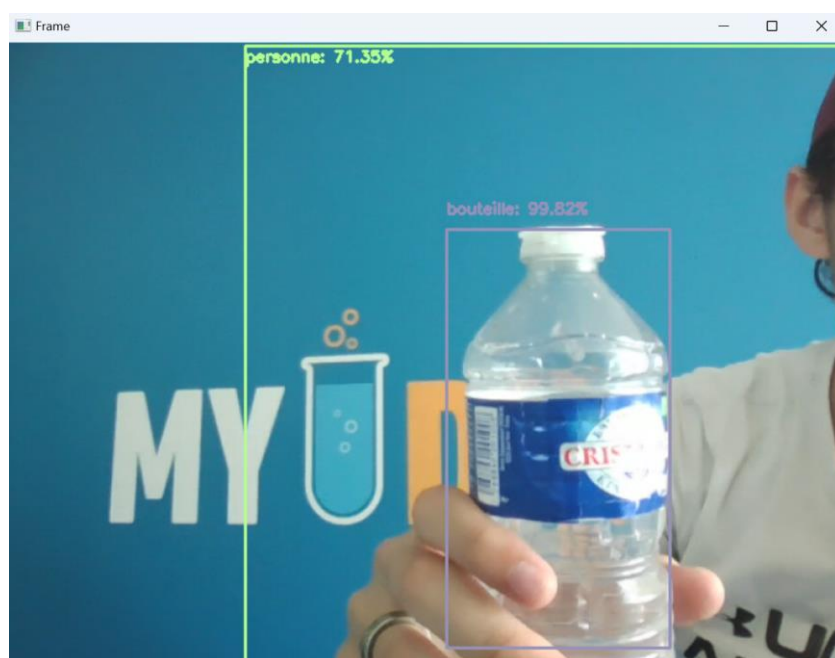


Lors de la vidéo, on voit que le carré qui entoure les entité suit cette dernière qui est en mouvement, cela réduit sa probabilité de détection, ce qui peut parfois mener à une erreur (de temps en temps, il pense qu'une voiture est une bouteille par exemple)

L'IA arrive tout de même à suivre plusieurs entité en même temps.

### Vidéo stream :

Voici une capture d'écran du stream de la caméra de mon ordinateur pendant la détection d'objet, en direct.



On remarque un fort pourcentage de détection de la bouteille et de la personne (légèrement plus faible pour la personne, car je ne suis pas totalement sur l'écran, ce qui est compréhensible).

## Combinaison des deux missions :

Après avoir réussi les deux missions, j'ai pu combiner les deux missions en une seule, c'est-à-dire que j'ai réussi à intégrer de l'intelligence artificielle au robot (de façon manuelle).

### Explications :

La première mission consiste à programmer le robot et à créer des programmes originaux, la deuxième mission consiste à découvrir et tester un modèle d'IA déjà entraîné. Alors j'ai fait en sorte d'intégrer l'IA au robot via un programme python, qui permet de prendre une photo grâce à la caméra du robot, l'IA va ensuite directement récupérer la photo qui sera stocker dans un dossier dans l'ordinateur, et va pouvoir faire la détection d'objet sur l'image qui vient d'être prise.

Dans un premier temps, la capture de photo grâce au robot :

```
PS C:\Users\enzor\OneDrive\Документы\EPSI\STAGE\AI> & C:/Users/enzor/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/enzor/OneDrive/Документы/EPSI/STAGE/AI/AI_photo.py"
Photo taken successfully.
Photo name: img_20240626_095009_8709.jpg
http://10.35.96.250:9090/v1/visions/photos?body=img_20240626_095009_8709.jpg
```

Si tout se passe bien, la photo sera prise correctement, son nom sera récupéré, et la photo sera stocké dans un dossier à proximité, avec le même nom que celle que le robot lui a donné.

```
Photo saved at: img\img_20240626_100554_2172.jpg
Load Neural Network...
Start Camera...
person
Request was successful.
Response data: {'code': 0, 'data': '{}', 'msg': 'Success'}
[INFO] elapsed time: 1.96
[INFO] approx. FPS: 15.33
PS C:\Users\enzor\OneDrive\Документы\EPSI\STAGE\AI> █
```

Après avoir stocké la photo dans le dossier, c'est l'IA qui va se charger de la suite, récupérant la photo qui vient d'être prise (avec le nom exacte), et qui va ensuite faire son opération de détection d'objet. L'image, après avoir été traité par l'IA, sera stocker dans le même dossier que celui du programme python, sous le nom de « detection.png » ou l'on pourra retrouver les informations que le robot aura détecté (images similaires aux exemples du dessus).

## Conclusion

### Résumé

J'ai commencé mon stage le mardi 28 mai, au sein de l'école EPSI de Rennes, pour effectuer une mission dans le domaine de la programmation, en python, sur un robot dénommé « Yanshee ». J'ai aussi effectué une sous-mission où je devais tester un programme d'IA déjà entraîné, et par la suite j'ai trouvé un moyen de combiner les deux missions, ce qui m'a permis d'intégrer de l'IA avec le robot.

### Technologies découvertes

Tout au long du stage, j'ai appris à utiliser des technologies qui m'étaient alors inconnues auparavant. Cela m'a permis de découvrir de nouvelles choses et pourront m'aider dans ma carrière professionnelle. Voici les technologies que j'ai pu manipuler :

- API RESTful
- Fichiers JSON
- Swagger
- WSL

Et voici les technologies dans lesquelles j'ai pu progresser :

- Python (Bibliothèque JSON et request)

### Ressenti du stage

Pratiquer ce stage a été une bonne expérience pour moi. Cela m'a permis d'apprendre beaucoup de choses concrète et utile pour le futur, et cela m'a aussi permis de m'ouvrir sur des sujets sur lesquelles je n'étais pas forcément intéressé, en l'occurrence l'intelligence artificielle. En effet, en effectuant cette deuxième mission en rapport avec l'IA, j'ai pu comprendre un peu mieux comment fonctionne une intelligence artificielle, et cela m'a motivé à chercher plus loin et de vouloir comprendre encore plus comment elles marchent.

## Contribution à l'EPSI

Créer des programmes à distance via une API RESTful permet à l'EPSI de pouvoir utiliser le robot à des fins pédagogiques, et cela permet aussi une utilisation plus efficace.

En plus de cela, j'ai pu améliorer le robot en ajoutant une IA de reconnaissance d'objet, ce qui ouvre encore d'autres possibilités de programmation.

## Ouverture sur la suite/projet professionnelle

Comme dis précédemment dans « ressenti du stage », pratiquer l'intelligence artificielle m'a donné envie d'en savoir plus sur ce sujet. Cela m'a permis de me remettre en question et d'avoir une possibilité de m'orienter en IA. Cependant, lors de mon stage, j'ai eu la chance d'entendre parler d'une journée d'exercice de cybersécurité pour les I1. Une mise en scène a été effectuée par un des intervenants, et le but était de pratiquer la cybersécurité dans une mise en scène. J'ai trouvé ça très originale, et en m'intéressant un peu plus à ce qu'ils faisaient, cela m'a beaucoup donné envie d'en savoir plus à ce sujet qui, de base, m'intéresse pas mal.

Programme pour lancer un stream avec le robot :

```
zstream_start.py > ...
1  ∨ import requests
2  import json
3
4  url = 'http://10.35.96.250:9090/v1/visions/streams'
5  ∨ headers = {
6      'Content-Type': 'application/json',
7      'Accept': 'application/json'
8  }
9  ∨ data = {
10     'resolution': '640x480'
11 }
12
13 response = requests.post(url, headers=headers, data=json.dumps(data))
14
15 ∨ if response.status_code == 200:
16     print("Requête réussie :", response.json())
17 ∨ else:
18     print(f"Erreur lors de la requête : {response.status_code}")
19     ✨ print("Détail de l'erreur :", response.json())
20
21
22
```

Pour l'arreter il suffit de remplacer « post » par « delete » ligne 13 et relancer le programme.

```
zreset_manually.py > ...
1 import requests
2 import json
3
4 url = 'http://10.35.97.205:9090/v1/servos/angles'
5
6 headers = {
7     'Content-Type': 'application/json',
8     'Accept': 'application/json'
9 }
10
11 data = {
12     "angles": {
13         "LeftAnkleFB": 70,
14         "LeftAnkleUD": 90,
15         "LeftElbowFlex": 15,
16         "LeftHipFB": 120,
17         "LeftHipLR": 90,
18         "LeftKneeFlex": 105,
19         "LeftShoulderFlex": 40,
20         "LeftShoulderRoll": 90,
21         "NeckLR": 90,
22         "RightAnkleFB": 110,
23         "RightAnkleUD": 90,
24         "RightElbowFlex": 165,
25         "RightHipFB": 60,
26         "RightHipLR": 90,
27         "RightKneeFlex": 76,
28         "RightShoulderFlex": 140,
29         "RightShoulderRoll": 90
30     },
31     "runtime": 500
32 }
33
34 try:
35     response = requests.put(url, headers=headers, data=json.dumps(data))
36
37     if response.status_code == 200:
38         print("Request was successful.")
39         print("Response data:", response.json())
40     else:
41         print(f"Failed to send request. Status code: {response.status_code}")
42         print("Response text:", response.text)
43
44 except requests.exceptions.RequestException as e:
45     print(f"An error occurred: {e}")
```

Programme pour changer l'angle des servos :

```
zput_servos.py > ...
1  import requests
2  import json
3
4  url = 'http://10.35.96.250:9090/v1/servos/angles'
5
6  headers = {
7      'Content-Type': 'application/json',
8      'Accept': 'application/json'
9  }
10
11  data = {
12      "angles": {
13          "LeftAnkleFB": 70,
14          "LeftAnkleUD": 90,
15          "LeftElbowFlex": 15,
16          "LeftHipFB": 120,
17          "LeftHipLR": 90,
18          "LeftKneeFlex": 105,
19          "LeftShoulderFlex": 120,
20          "LeftShoulderRoll": 90,
21          "NeckLR": 90,
22          "RightAnkleFB": 110,
23          "RightAnkleUD": 90,
24          "RightElbowFlex": 165,
25          "RightHipFB": 60,
26          "RightHipLR": 90,
27          "RightKneeFlex": 76,
28          "RightShoulderFlex": 140,
29          "RightShoulderRoll": 90
30      },
31      "runtime": 500
32  }
33
34  try:
35      response = requests.put(url, headers=headers, data=json.dumps(data))
36
37      if response.status_code == 200:
38          print("Request was successful.")
39          print("Response data:", response.json())
40      else:
41          print(f"Failed to send request. Status code: {response.status_code}")
42          print("Response text:", response.text)
43
44  except requests.exceptions.RequestException as e:
45      print(f"An error occurred: {e}")
```

Programme pour prendre une photo avec le robot, et la stocker sur l'ordinateur dans un dossier :

```
zfinal_photo taking.py > ...
1 import requests
2 import os
3
4 url_take_photo = 'http://10.35.96.250:9090/v1/visions/photos'
5
6 url_get_photo_list = 'http://10.35.96.250:9090/v1/visions/photos?body='
7
8 headers = {
9     'Content-Type': 'application/json',
10    'Accept': 'application/json'
11}
12
13 photo_folder = "photos"
14
15 if not os.path.exists(photo_folder):
16     os.makedirs(photo_folder)
17
18 try:
19     response = requests.post(url_take_photo, headers=headers, json={})
20
21     if response.status_code == 200:
22         print("Photo taken successfully.")
23         response_data = response.json()
24
25         if response_data['code'] == 0 and 'name' in response_data['data']:
26             photo_name = response_data['data']['name']
27             print("Photo name:", photo_name)
28
29             photo_url = f'http://10.35.96.250:9090/v1/visions/photos?body={photo_name}'
```

```
photo_url = f'http://10.35.96.250:9090/v1/visions/photos?body={photo_name}'
print(photo_url)
photo_response = requests.get(photo_url, headers=headers)

if photo_response.status_code == 200:
    photo_path = os.path.join(photo_folder, photo_name)
    with open(photo_path, 'wb') as photo_file:
        photo_file.write(photo_response.content)

    print("Photo saved at:", photo_path)
else:
    print(f"Failed to download photo. Status code: {photo_response.status_code}")
    print("Response text:", photo_response.text)
else:
    print("Failed to take photo. Response data:", response_data)
else:
    print(f"Failed to send request to take photo. Status code: {response.status_code}")
    print("Response text:", response.text)

except requests.exceptions.RequestException as e:
    print(f"An error occurred: {e}")
```



Programme pour traquer le visage :

```
Face Tracking.py > ...
1  import requests, json, time
2
3  url = 'http://10.35.96.250:9090/v1/visions'
4
5  headers = {
6      'Content-Type': 'application/json',
7      'Accept': 'application/json'
8  }
9
10 data = {
11     "operation": "stop",
12     "option": "face",
13     "timestamp": int(time.time()),
14     "type": "tracking"
15 }
16
17 try:
18     response = requests.put(url, headers=headers, data=json.dumps(data))
19
20
21     if response.status_code == 200:
22         print("Request was successful.")
23         print("Response data:", response.json())
24     else:
25         print(f"Failed to send request. Status code: {response.status_code}")
26         print("Response text:", response.text)
27
28 except requests.exceptions.RequestException as e:
29     print(f"An error occurred: {e}")
```

Fonction pour pouvoir faire parler le robot (text to speech) :

```
def tts(text):
    url = 'http://10.35.96.250:9090/v1/voice/tts'
    current_timestamp = int(time.time())
    payload = {
        "interrupt": True,
        "timestamp": current_timestamp,
        "tts": text
    }
    headers = {
        'Content-Type': 'application/json',
        'Accept': 'application/json'
    }
    try:
        response = requests.put(url, headers=headers, data=json.dumps(payload))
        if response.status_code == 200:
            print("Request was successful.")
            print("Response data:", response.json())
        else:
            print(f"Failed to send request. Status code: {response.status_code}")
            print("Response text:", response.text)
    except requests.exceptions.RequestException as e:
        print(f"An error occurred: {e}")
```

Fonction pour faire bouger le robot :

```
def start_motion(direction, name, repeat, speed):
    current_timestamp = int(time.time())
    url = 'http://10.35.96.250:9090/v1/motions'
    payload = {
        "motion": {
            "direction": direction,
            "name": name,
            "repeat": repeat,
            "speed": speed
        },
        "operation": "start",
        "timestamp": current_timestamp
    }
    headers = {
        'Content-Type': 'application/json',
        'Accept': 'application/json'
    }
    try:
        response = requests.put(url, headers=headers, data=json.dumps(payload))
        if response.status_code == 200:
            print("Request start_motion was successful.")
            print("Response data:", response.json())
        else:
            print(f"Failed to send request. Status code: {response.status_code}")
            print("Response text:", response.text)
    except requests.exceptions.RequestException as e:
        print(f"An error occurred: {e}")
```

Fonction pour reset le robot (le mettre en position de base, mais grâce à la méthode du mouvement et non manuellement) :

```
def reset():  
    url = 'http://10.35.96.250:9090/v1/motions'  
    reset = {  
        "motion": {  
            "name": "reset",  
            "repeat": 1,  
            "speed": "slow"  
        },  
        "operation": "start",  
        "timestamp": 1  
    }  
    headers = {  
        'Content-Type': 'application/json',  
        'Accept': 'application/json'  
    }  
    response = requests.put(url, data=json.dumps(reset), headers=headers)  
    print(response.status_code)  
    print(response.json())
```

Fonction pour reconnaître l'âge de la personne :

```
functions.py > ...
129 def take_photo_age():
130     functions.tts("please put your head in front of the camera in:")
131     time.sleep(2.5)
132     functions.tts("3")
133     time.sleep(1)
134     functions.tts("2")
135     time.sleep(1)
136     functions.tts("1")
137     time.sleep(1)
138     url = 'http://10.35.96.250:9090/v1/visions'
139     headers1 = {
140         'Content-Type': 'application/json',
141         'Accept': 'application/json',
142     }
143     data1 = {
144         'operation': 'start',
145         'option': 'face',
146         'timestamp': 1551838515,
147         'type': 'age',
148     }
149     response1 = requests.put(url, headers=headers1, data=json.dumps(data1))
150     time.sleep(2)
151     url = 'http://10.35.96.250:9090/v1/visions'
152     headers2 = {
153         'Accept': 'application/json',
154     }

params2 = {
    'option': 'face',
    'type': 'age',
}
response2 = requests.get(url, headers=headers2, params=params2)
data = response2.json()
if isinstance(data.get('data'), str):
    try:
        data_dict = json.loads(data['data'])
    except json.JSONDecodeError:
        print("erreur")
        data_dict = {}
else:
    data_dict = data.get('data', {})
if 'analysis' in data_dict and 'age' in data_dict['analysis']:
    age = data_dict['analysis']['age']
    if age == 0:
        print("mettez votre tête devant l'écran")
        functions.tts("please put your head in front of the camera")
    else:
        print(age)
        functions.tts('it looks like you are around : ' + str(age) + ' years old')
else:
    print("erreur")
    functions.tts("error")
```

Fonction pour reconnaître la couleur :

```
def color_recognition():
    import requests
    import json
    url = 'http://10.35.96.250:9090/v1/visions'
    headers1 = {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
    }
    data1 = {
        'operation': 'start',
        'option': 'color',
        'timestamp': 1551838515,
        'type': 'color_detect',
    }

    response1 = requests.put(url, headers=headers1, data=json.dumps(data1))

    time.sleep(2)

    import requests
    import json

    url = 'http://10.35.96.250:9090/v1/visions?option=color&type=color_detect'

    headers = {
        'Accept': 'application/json'
```

```
def color_recognition():
    headers = {
        'Accept': 'application/json'
    }

    response = requests.get(url, headers=headers)
    print(response.status_code)
    print(response.json())
    data = response.json()

    if isinstance(data.get('data'), str):
        try:
            data_dict = json.loads(data['data'])
        except json.JSONDecodeError:
            print("erreur")
            data_dict = {}
    else:
        data_dict = data.get('data', {})

    if 'color' in data_dict:
        color_info = data_dict['color']
        if color_info:
            color_name = color_info[0]['name']
            print("couleur détecté", color_name)
            functions.tts("The detected color is:" + color_name)
        else:
            print("pas de couleur détecté")
            functions.tts("No color detected")
    else:
        print("erreur")
```

Fonction pour modifier les leds (couleur, type, mode)

```
def put_led(type, color, mode):  
    url = 'http://10.35.96.250:9090//v1/devices/led'  
    headers = {  
        'Content-Type': 'application/json',  
        'Accept': 'application/json'  
    }  
  
    data = {  
        'color': color,  
        'mode': mode,  
        'type': type  
    }  
  
    response = requests.put(url, headers=headers, json=data)  
    return response
```

Fonction pour « rendre le robot vivant »

Il va bouger aléatoirement la tête de droite à gauche, avec un intervalle de temps aléatoire, et cligner des yeux avec un intervalle de temps aussi aléatoire.

```
def alive():
    import requests
    import time
    import random
    import json
    import threading

    def led_blink():
        url = 'http://10.35.96.250:9090/v1/devices/led'
        headers = {
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        }

        while True:
            duration_on = random.randint(1, 7)

            data_on = {
                'color': 'cyan',
                'mode': 'on',
                'type': 'camera'
            }
            requests.put(url, headers=headers, json=data_on)

            time.sleep(duration_on)
            data_off = {
                'color': 'cyan',
                'mode': 'off',
                'type': 'camera'
            }
```

```
def alive():
    def led_blink():
        url = 'http://10.35.96.250:9090/v1/devices/led'
        headers = {
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        }

        while True:
            duration_on = random.randint(1, 7)

            data_on = {
                'color': 'cyan',
                'mode': 'on',
                'type': 'camera'
            }
            requests.put(url, headers=headers, json=data_on)

            time.sleep(duration_on)
            data_off = {
                'color': 'cyan',
                'mode': 'off',
                'type': 'camera'
            }
            requests.put(url, headers=headers, json=data_off)
            time.sleep(0.1)

    def move_head():
        url = 'http://10.35.96.250:9090/v1/servos/angles'
        headers = {
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        }

        while True:
            angle_tete = random.randint(70, 110)

            data = {
                "angles": {
                    "NeckLR": angle_tete,
                    "LeftAnkleFB": 70,
                    "LeftAnkleUD": 90,
                    "LeftElbowFlex": 15,
                    "LeftHipFB": 120,
                    "LeftHipLR": 90,
                    "LeftKneeFlex": 105,
                    "RightAnkleFB": 70,
                    "RightAnkleUD": 90,
                    "RightElbowFlex": 15,
                    "RightHipFB": 120,
                    "RightHipLR": 90,
                    "RightKneeFlex": 105
                }
            }
```





```
def alive():
    def move_head():
        "LeftShoulderFlex": 40,
        "LeftShoulderRoll": 90,
        "RightAnkleFB": 110,
        "RightAnkleUD": 90,
        "RightElbowFlex": 165,
        "RightHipFB": 60,
        "RightHipLR": 90,
        "RightKneeFlex": 76,
        "RightShoulderFlex": 140,
        "RightShoulderRoll": 90
    },
    "runtime": 500
}

response = requests.put(url, headers=headers)
time.sleep(random.uniform(2, 4))

# Créer et démarrer Les threads
thread1 = threading.Thread(target=led_blink)
thread2 = threading.Thread(target=move_head)

thread1.start()
thread2.start()

thread1.join()
thread2.join()
```

Fonction pour obtenir une liste de tous les mouvements possibles :

```
def get_motions_list():
    # URL de l'API
    url = 'http://10.35.96.250:9090/v1/motions/list'

    # En-têtes de la requête
    headers = {
        'Accept': 'application/json'
    }

    # Envoyer la requête GET
    response = requests.get(url, headers=headers)

    # Vérifier le statut de la réponse
    if response.status_code == 200:
        print("Request was successful.")
        # Afficher le contenu de la réponse en format JSON
        print(response.json())
    else:
        print(f"Failed to send request. Status code: {response.status_code}")
        print("Response text:", response.text)
```

Réponse :

```
Request was successful.
Response data: {'code': 0, 'data': {'system_hts_motions': [{'music': True, 'name': 'Wakawaka'}, {'music':
False, 'name': 'EnterEnergySavingSquat'}, {'music': True, 'name': '圣诞快乐'}, {'music': True, 'name': '
唱支山歌给党听'}, {'music': False, 'name': 'RightTackle'}, {'music': False, 'name': 'OneStepMoveRight'},
{'music': False, 'name': 'knee left'}, {'music': False, 'name': 'Football_LSideK'}, {'music': False, 'nam
e': 'Backward1'}, {'music': False, 'name': 'GoalKeeper2'}, {'music': False, 'name': 'ActionAging'}, {'mus
ic': False, 'name': 'ResetSwitch'}, {'music': True, 'name': '我们是共产主义接班人'}, {'music': True, 'nam
e': 'WeAreTakingOff'}, {'music': False, 'name': 'OneStepTurnLeft'}, {'music': False, 'name': 'Move_fast'}
, {'music': False, 'name': 'TurnLeft1'}, {'music': False, 'name': 'TurnR_tiny'}, {'music': False, 'name':
'GetupFront'}, {'music': False, 'name': 'Base_GetupF'}, {'music': False, 'name': 'Backward'}, {'music':
True, 'name': 'LittleApple'}, {'music': False, 'name': 'OneStepMoveLeft'}, {'music': False, 'name': 'Left
Tackle'}, {'music': False, 'name': 'Stop'}, {'music': False, 'name': 'Shutdown'}, {'music': False, 'name'
: 'LeftSidePunch'}, {'music': False, 'name': 'Fight_RHit'}, {'music': True, 'name': '从这里出发'}, {'musi
c': False, 'name': 'Leftward'}, {'music': False, 'name': 'Fight_RSideHit'}, {'music': False, 'name': 'Get
upRear'}, {'music': False, 'name': 'Left slide tackle'}, {'music': False, 'name': 'ActionAging1'}, {'musi
c': True, 'name': '光荣啊! 中国共青团'}, {'music': False, 'name': 'Base_GetupB'}, {'music': True, 'name':
'GangnamStyle'}, {'music': False, 'name': 'Base_Forward'}, {'music': False, 'name': 'ShootRight'}, {'mus
ic': False, 'name': 'TurnLeft'}, {'music': True, 'name': 'HappyBirthday'}, {'music': False, 'name': 'cali
bration_new'}, {'music': False, 'name': 'Forward1'}, {'music': False, 'name': 'TurnLeftSlowly'}, {'music'
: False, 'name': 'Football_LKick'}, {'music': False, 'name': 'TurnRight1'}, {'music': False, 'name': 'Goa
lKeeper1'}, {'music': False, 'name': 'Football_RSideK'}, {'music': False, 'name': 'Base_Leftward'}, {'mus
ic': False, 'name': 'Base_TurnR'}, {'music': False, 'name': 'Fight_LHit'}, {'music': False, 'name': 'Base
_Rightward'}, {'music': False, 'name': 'TurnRight'}, {'music': False, 'name': 'OneStepForward'}, {'music'
: False, 'name': 'Football_RKick'}, {'music': False, 'name': 'LeftHitForward'}, {'music': False, 'name':
'RightSidePunch'}, {'music': False, 'name': 'PushUp'}, {'music': False, 'name': 'RightHitForward'}, {'mus
ic': True, 'name': 'CombinationOfSongs'}, {'music': False, 'name': 'Football_LKeep'}, {'music': False, 'n
```

Programme de présentation du Yanshee :

```
Hi.py > ...
1
2 import functions
3 import time
4 import json
5 import requests
6
7 functions.reset()
8 functions.tts("Hello my friend !")
9 functions.start_motion("left", "wave", 1, "slow")
10 time.sleep(3.8)
1
2 url = 'http://10.35.96.250:9090/v1/servos/angles'
3
4 headers = {
5     'Content-Type': 'application/json',
6     'Accept': 'application/json'
7 }
8
```

```
data2 = {
    "angles": {
        "LeftShoulderFlex": 0,
        "LeftShoulderRoll": 180,
        "LeftElbowFlex": 0,
        "RightShoulderFlex": 180,
    },
    "runtime": 500
}

try:
    # Envoyer La requête PUT
    response = requests.put(url, headers=headers, data=json.dumps(data2))

    # Vérifier Le statut de La réponse
    if response.status_code == 200:
        print("Request was successful.")
        print("Response data:", response.json())
    else:
        print(f"Failed to send request. Status code: {response.status_code}")
        print("Response text:", response.text)

except requests.exceptions.RequestException as e:
    print(f"An error occurred: {e}")
```

```
functions.tts("My name is Yanshee ! It's a pleasure to meet you !")
time.sleep(3)
functions.reset()
time.sleep(2)

functions.take_photo_age()
functions.reset()
```

Programme qui combine l'IA de détection d'objet à la prise de photo du robot :

```
zfinal_AI photo.py > ...
1  import requests
2  import os
3  import fonctions
4
5  url_take_photo = 'http://10.35.96.250:9090/v1/visions/photos'
6  url_get_photo_list = 'http://10.35.96.250:9090/v1/visions/photos?body='
7
8  headers = {
9      'Content-Type': 'application/json',
10     'Accept': 'application/json'
11 }
12
13 photo_folder = "img"
14
15 if not os.path.exists(photo_folder):
16     os.makedirs(photo_folder)
17
18 try:
19     response = requests.post(url_take_photo, headers=headers, json={})
20
21     if response.status_code == 200:
22         print("Photo taken successfully.")
23         response_data = response.json()
```

```
zfinal_AI photo.py > ...
25     if response_data['code'] == 0 and 'name' in response_data['data']:
26         photo_name = response_data['data']['name']
27         print("Photo name:", photo_name)
28         photo_url = f'http://10.35.96.250:9090/v1/visions/photos?body={photo_name}'
29         print(photo_url)
30         photo_response = requests.get(photo_url, headers=headers)
31         if photo_response.status_code == 200:
32             photo_path = os.path.join(photo_folder, photo_name)
33
34             with open(photo_path, 'wb') as photo_file:
35                 photo_file.write(photo_response.content)
36
37             print("Photo saved at:", photo_path)
38         else:
39             print(f"Failed to download photo. Status code: {photo_response.status_code}")
40             print("Response text:", photo_response.text)
41     else:
42         print("Failed to take photo. Response data:", response_data)
43 else:
44     print(f"Failed to send request to take photo. Status code: {response.status_code}")
45     print("Response text:", response.text)
46
47 except requests.exceptions.RequestException as e:
48     print(f"An error occurred: {e}")
```



```
print(f"An error occurred: {e}")

import sys
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2

if len(sys.argv)==1:
    args={
        "prototxt":"MobileNetSSD_deploy.prototxt.txt",
        "model":"MobileNetSSD_deploy.caffemodel",
        "confidence":0.2,
    }
else:
    ap = argparse.ArgumentParser()
    ap.add_argument("-p", "--prototxt", required=True,
        help="path to Caffe 'deploy' prototxt file")
    ap.add_argument("-m", "--model", required=True,
        help="path to Caffe pre-trained model")
    ap.add_argument("-c", "--confidence", type=float, default=0.2,
        help="minimum probability to filter weak detections")
    args = vars(ap.parse_args())

CLASSES = ["background", "plane", "bike", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "table",
    "dog", "horse", "motorcycle", "person", "potted plant", "sheep",
    "sofa", "train", "monitor"]

COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

print("Load Neural Network...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

if __name__ == '__main__':

    print("Start Camera...")

    #vs = VideoStream(src=0, resolution=(1600, 1200)).start()
    #vs = VideoStream(usePiCamera=True, resolution=(1600, 1200)).start()
    #vc = cv2.VideoCapture('./img/cars.mp4') #from video

    time.sleep(2.0)
    fps = FPS().start()
```



```
while True:
    #frame = vs.read()
    frame = cv2.imread(os.path.join(photo_folder, photo_name)) #from image file
    #ret, frame=vc.read() #from video or ip cam
    frame = imutils.resize(frame, width=800)

    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843, (300, 300), 127.5)

    net.setInput(blob)
    detections = net.forward()

    for i in np.arange(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]

        if confidence > args["confidence"]:
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            label = "{}: {:.2f}%".format(CLASSES[idx],
                                         confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                          COLORS[idx], 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(frame, label, (startX, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
            cv2.imwrite("detection.png", frame)

    print(label)
```

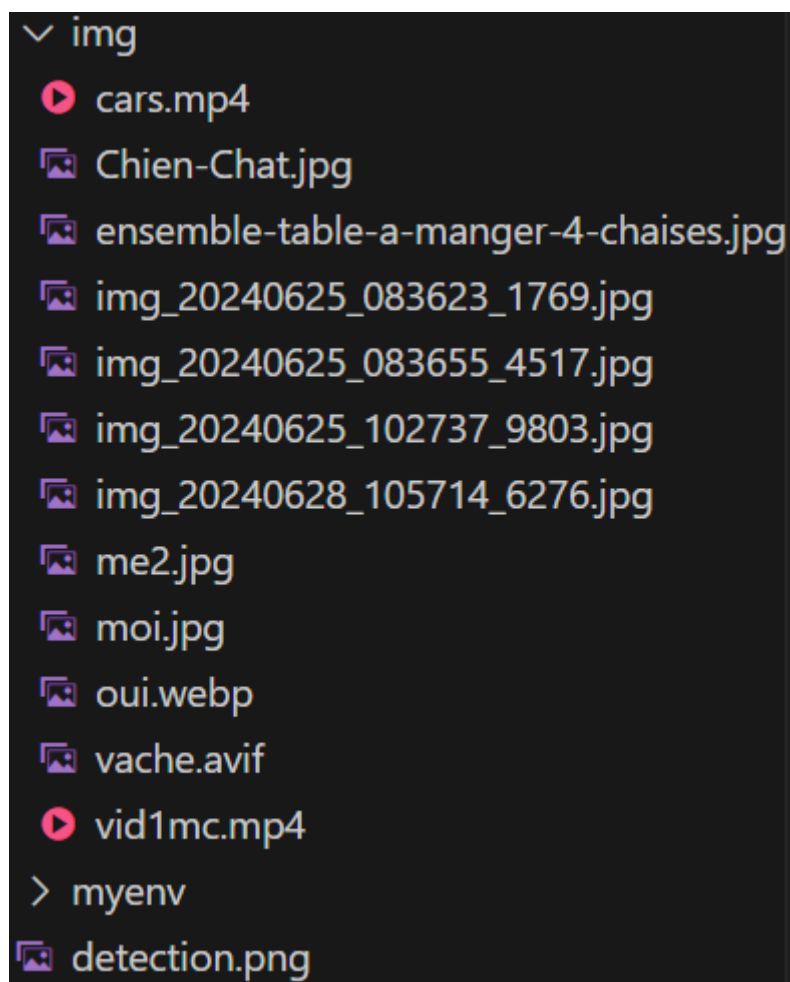
```
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF

# Exit script with Letter q
if key == ord("q"):
    break
fps.update()

fonctions.tts("this is a" + label + "sure")
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

cv2.destroyAllWindows()
#vs.stop()
#vc.release()
```

Dossier où sont stocké les photos que le robot prend en photo, mais aussi où l'intelligence artificielle vient récupérer les images/vidéo pour faire sa détection.



Les noms d'image avec une date sont les photos prises par le robot, et les autres photos et vidéos sont celles prises sur internet ou par moi-même pour faire les détections.

L'image « detection.png » est le résultat que l'on obtient après la détection.