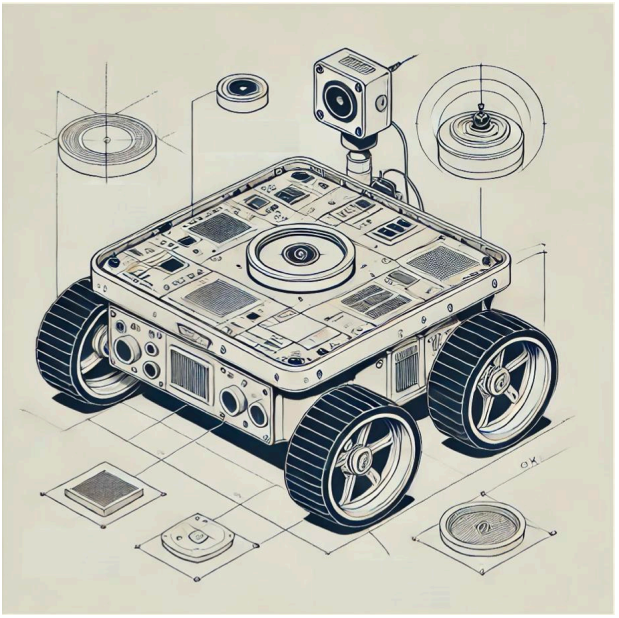
PROJET FIL ROUGE 2024-2025



Dossier de conception

Réalisé par :

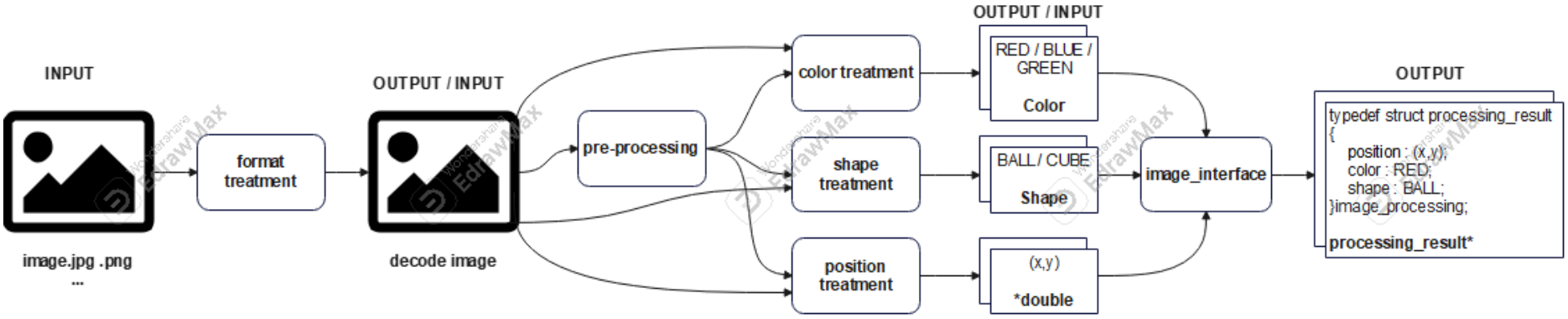
- *TAYOU MBEDE Ryan Niel*
- *WACHILL Imran*
- *BA Djibril*
- *BAKHTI Hela*
- *MEHREZ Wissal*
- *SECK Adji Aram*

I. MODULE TRAITEMENT D'IMAGES (image_processing) 2
A. ARCHITECTURE 2
B. FONCTIONNEMENT DE L'ALGORITHME 3
Cas d'un cercle 3
Cas d'un carré 4
C. Cas particulier : Plusieurs objets possédant la même couleur que celle recherché 5
Étape 1 : Segmentation 5
Epale 2 : 6
II. Commande Vocale 7
1. Module de Reconnaissance Vocale (SpeechToText) 7
2. Module d'Analyse et de Traduction des Commandes 7
a. Identification et Remplacement des Synonymes 7
b. Segmentation et Structuration du Texte 7
c. Conversion des Segments en Requêtes JSON 7
3. Module d'Exécution et Simulation 8
III. Mode IHM 8
IV. Mode Manuel - Contrôle via Clavier 8
Fonctionnement : 8
V. Mode Auto - Déplacements Autonomes 9
Fonctionnement : 9
Conclusion 9
VI. Option de Configuration : Settings 9
Description du fichier de configuration settings.json 9
Principaux paramètres et leur rôle : 10
VII. Bilans Personnelles 12
TAYOU MBEDE Ryan Niel 12
WACHILL Imran 13
BA Djibril 14
BAKHTI Hela 15
MEHREZ Wissal 15
SECK Adji Aram 16

I. MODULE TRAITEMENT D'IMAGES (image_processing)

A. ARCHITECTURE

Dans cette partie nous présenterons la structure architecturale ainsi que les différents algorithmes que nous proposons pour le module traitement d'images (**image_processing**) : formes, couleurs et position. Ses algorithmes ont été construits à partir de la méthode de décomposition par raffinage successif.



images 1 : image processing process

Cette image présente le processus de traitement d'une image.
Le module réservé au traitement d'images prendra une image dans un format classique (jpg ou png) en entrée et en sortie retourne une structure de données contenant tous les résultats de traitement sur cette image (position, couleur et forme). Raisonner de cette façon (faire transiter la communication par **image_interface**) facilite la maintenabilité du code. De plus, pour des questions de réduction de risque d'erreur, les autres modules (commande vocale et simulation) communiqueront avec les différentes fonctions du module **image_processing** (module traitement image) **uniquement via image_interface.** Donc le type que le module traitement image enverra aux modules qui le sollicitent sera **une structure (object).**

NB : Le code relatif à ‘’format treatment” sera développé au PFR2. Ici nous travaillerons avec des images non encodées.
Les types **Color** et **Shape** présents dans le schéma sont des types énumération. qui contiennent respectivement les couleurs et formes traitées. Par contre **Position** est une structure qui contient la position de l’objet trouvé.

B. FONCTIONNEMENT DE L’ALGORITHME

Nous raisonnons dans ce sens : nous allons regarder directement les pixels qui nous intéressent dans l'image. Mais la méthode est la suivante :
Supposons que nous voulons rechercher une balle bleue dans l’image.

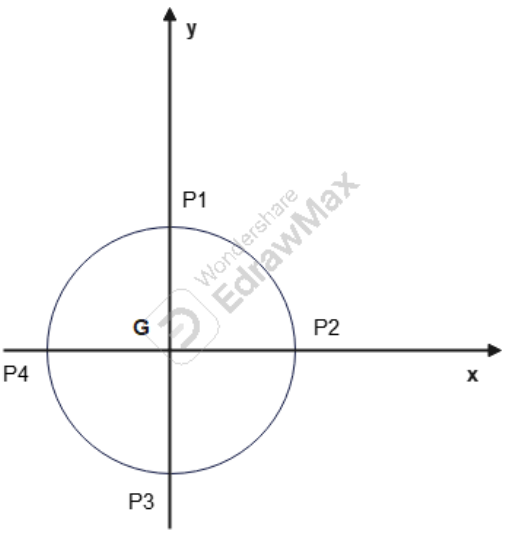
- premièrement, **“preprocessing image”** récupère l’image dans la base RGB(RED GREEN BLUE) et la convertit dans la base HSV(HUE SATURATION VALUE). l’image dans la base HSV est dans un tableau à trois dimensions (“***image hsv***”). Il est intéressant de la convertir dans une base de luminance car de telle base son moins sensible à l'éclairage ce qui est à notre avantage vue que c’est la couleur qui nous intéresse.
- **“color treatment”** récupère “***image hsv”*** et effectue la segmentation de celle-ci en fonction de la couleur qu’on recherche. Pour cela, il se base essentiellement sur la composante H (teinte) de chaque pixel. Et les seuils de “Hue” sont précisés dans le tableau ci-dessous (seuil de Hue).

  Après quoi, il indique si la couleur recherchée a été trouvée (le rouge dans notre exemple). L’image segmentée est mise dans un tableau à 2 dimensions (“binary image”).

  Il est a précisé que la segmentation est rarement parfaite il reste en général des bruits de types sel et poivre. Pour remédier à cela, nous utilisons un filtre médian. Car celui-ci est très efficace pour régler ce style de problème.
- **“shape treatment”** récupère “***binary imag***e” et vérifie si la forme correspond à celle demandée. De la manière suivante.

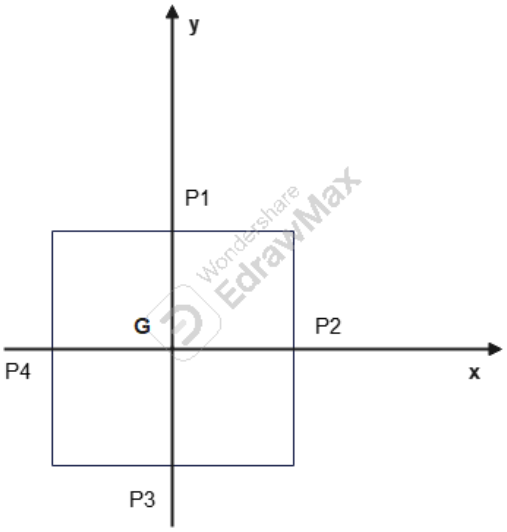
Cas d’un cercle

Un cercle peut être caractérisé par son centre et son rayon. Nous nous baserons sur ses informations pour vérifier si “binary image” contient un cercle.
Premièrement, nous déterminerons le centre de notre figure et son rayon.
Puis grâce à ses informations, nous évaluons l’aire du cercle possédant le rayon que nous avons trouvé. Puis nous comparerons l’aire(le nombre de pixel qui constitue la figure) obtenue à l'aire de la figure que contient “binary image” et nous conclurons.
<u>Procédure pour déterminer le rayon</u>
Pour le cercle de la figure ci-dessous, le rayon se détermine facilement si on a 02 points Pi distincts.



Numériquement, nous déterminerons le point possédant la plus grande coordonnées suivantes l'axe y et le point ayant la plus petite coordonnées suivantes l'axe y. Le rayon sera la moitié de la distance entre ces deux points.

Cas d'un carré



Pour un carré, c'est typiquement le même raisonnement que pour un cercle.
Nous déterminerons le point possédant la plus grande coordonnées suivantes l'axe y et le point ayant la plus petite coordonnées suivantes l'axe y. La longueur d'un côté sera la distance entre ces deux points. Puis nous déterminons l'aire(le nombre de pixel qui constitue la figure) d'un carré ayant un côté de la longueur déterminée. Puis nous comparerons l'aire obtenue à l'aire de la figure que contient "binary image" et nous conclurons.

Remarque :

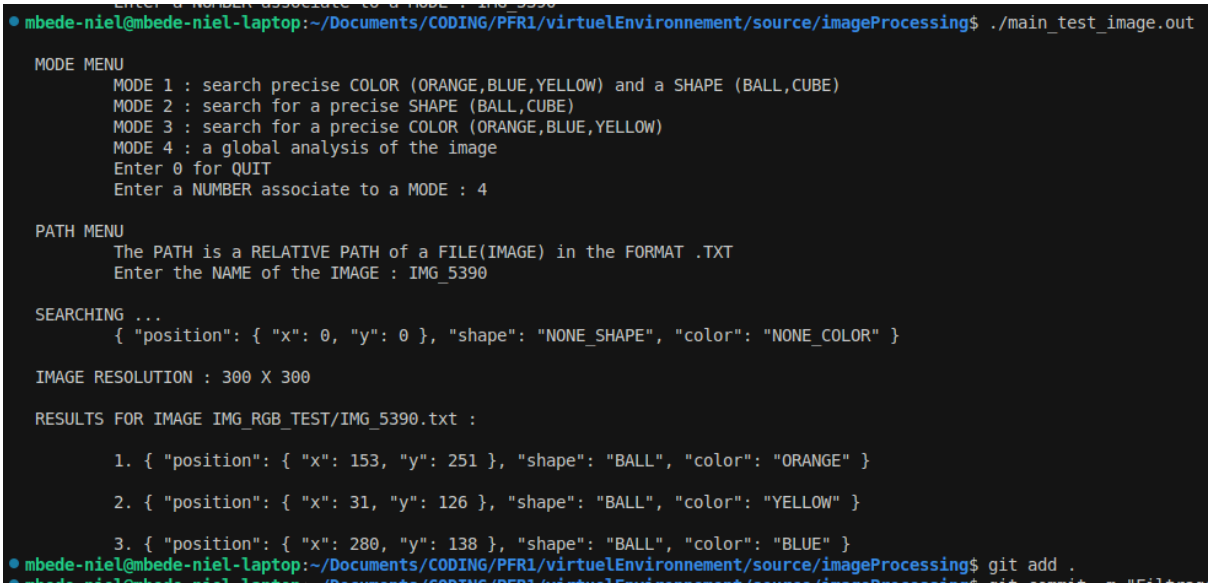
- Il ne nous est pas demandé de distinguer, un rectangle d'un carré, d'un losange. ou une forme ovale d'un cercle.
- Pour améliorer la précision de l'algorithme qui permet de déterminer les formes, Nous pouvons faire l'intersection entre les deux figures. Figure de "binary image" et figure formée à partir du rayon et du centre pour un cercle. Ou d'un côté pour une carrée.

Le resultat que nous obtenons pour l'image noir blanc ci dessous. Soit 97% de chance qu'il s'agisse d'un cercle et 65% pour d'un cube.

Fonctionnalités supplémentaires : des algorithmes supplémentaires ont été développés. Et combiné avec les algorithmes ci dessus, ils permettent de :

- réaliser une analyse complète de l'image. Ici l'utilisateur décrit ce qu'il recherche et l'algorithme ressort de l'image tout ce qui match avec sa description.
- faire une analyse plus poussée de l'image ici, l'utilisateur ne précise rien et l'algorithme ressort **tout** ce qui est contenu dans l'image en accord avec les couleurs prises en charge. Dans ce cas, même les formes qui ne sont ni des balles ni des cubes sont traitées (on génère leurs couleurs et positions).

Mise à jour : Lors de la présentation, vous(le jury) avez demandé de filtrer les résultats pour qu'il soit mieux présentable, ci-dessous une image de l'affichage après cette mise à jour.



C. Cas particulier : Plusieurs objets possédant la même couleur que celle recherché

L'algorithme "**shape treatment"** présenté fonctionne correctement uniquement lorsque les couleurs des éventuelles autres objets sur l'image sont différentes de la couleur recherchée. Nous proposerons donc dans cette partie un algorithme qui permet de gérer le cas où on a plusieurs objets de même couleur.

Étape 1 : Segmentation

Pour générer plusieurs images contenant chacune un objet, nous avons réalisé une fonction **segmentation**, cette fonction prend en entrée l'image binarisé, le niveau de connexité, le seuil de pixel jugé parasite et l'adresse de la file dans laquelle chaque image contenant un objet.

Lorsque nous analysons une image binaire, nous ne savons pas à l'avance :

- **combien d'objets seront détectés,**
- **combien de pixels composent chaque objet.**

Puisque ces valeurs peuvent varier en fonction de l'image, utiliser un tableau fixe serait inefficace et limiterait notre algorithme.

Utiliser une **pile dynamique** nous permet d'ajouter autant de pixels et d'objets que nécessaire sans gaspiller de mémoire.

L'image binarisée est parcourue pixel par pixel. Dès qu'un premier pixel d'intérêt est détecté, une **pile** est créée pour représenter un nouvel objet, et les coordonnées de ce pixel y sont stockées.

Ensuite, pour chaque pixel rencontré, plusieurs vérifications sont effectuées :

1. Parcours des objets déjà détectés : On vérifie si ce pixel appartient à un objet existant en testant sa connectivité avec les pixels déjà stockés dans les piles.

2. Ajout à un objet existant : Si le pixel est voisin d'un objet, il est ajouté à la pile correspondante.
3. Fusion de plusieurs objets : Si le pixel est connecté à plusieurs objets, ces objets sont fusionnés en une seule pile.
4. Création d'un nouvel objet : Si le pixel ne peut être rattaché à aucun objet existant, une nouvelle pile est créée pour le considérer comme un objet distinct.

Ce processus se poursuit jusqu'à ce que tous les pixels de l'image aient été analysés et regroupés dans les piles appropriées. Une fois cette phase terminée, un **filtrage est appliqué** pour éliminer les objets trop petits (en dessous d'un seuil donné). Les objets restants sont ensuite transformés en images (int **) séparées et stockées dans une **file dynamique**, garantissant ainsi de pouvoir appliquer des traitements sur les objets séparés comme "**shape treatment"** qui redevient utilisable.

Exemple de segmentation :

L'image 1 est un exemple d'image que nous voulons segmenter, pour cet exemple nous cherchons les balles orange pour cela elle est binarisée pour former une nouvelle image.
A partir de cette nouvelle image nous appelons la fonction de segmentation qui remplira une file avec l'image 2 et 3 qui pourront être traitée normalement par les algorithmes vue précédemment.
Il est à noter que pour ces exemple les images sont passer en .txt puis en .jpeg, mais cela n'est <u>que pour la compréhension</u> dans le programme après le passage de l'image référence .txt en entré, tout est enregistrer dans des variable adapter int*** pour les images RGB et HSV et int ** pour les images binarisés.

Epale 2 :

L'étape 1 consiste à séparer l'image en plusieurs images. De t'elle sortent que si on analyse une image I, contenant n objets, de couleur C, on separe I en n images contenant chacune un objet de couleur Y et eventuellement d'autres objets de couleurs différentes. La difficulté réside dans le fait que le nombre d'éléments de couleurs Y contenue dans l'image n'est pas connu à l'avance. D'où l'utilisation d'une structure dynamique (une FILE de tableaux). Le second problème rencontré est le suivant, le programme ne connaît pas la taille de l'image qu'il aura à traiter à l'avance d'où l'utilisation de tableaux dynamiques. De plus, les tableaux utilisés seront échangés entre plusieurs fonctions donc si on crée des tableaux statiques dans des fonctions, la durée de vie de nos tableaux se limitera à celle de la fonction dans laquelle il a été créé.

<u>Script python :</u>

Dans le cadre de procédure de test de cette partie image. Plusieurs scripts python on été réaliser : jpegToTxt pour générer des images .txt a partir de .jpeg, txtToJpeg pour visionner des images binarisé. Et generationCroix qui avait pour but d'être implémenté dans la solution finale, mais par manque de temps être ajoutée.

II. Commande Vocale

Le système de **Commande Vocale** permet d'exécuter des actions à partir d'instructions données oralement. Il prend en charge à la fois des **commandes simples** (comme "avancer", "reculer", "tourner à droite", "tourner à gauche") et des **instructions complexes** (par exemple : "Aller vers la balle rouge, avancer et tourner à droite d'un angle de 60 degrés").
Ce mode fonctionne sous un **module de simulation** qui exécute les commandes et est supervisé par un **module Vocal**. Ce dernier est structuré en trois sous-modules distincts :

1. Module de Reconnaissance Vocale (SpeechToText)

Ce module est responsable de l'écoute et de la transcription des instructions vocales en texte. Il utilise la bibliothèque **SpeechRecognition** pour :

- Capturer l'entrée vocale de l'utilisateur via un microphone.
- Convertir cette entrée en texte brut compréhensible.
- Envoyer ce texte aux modules de traitement pour analyse et exécution.

2. Module d'Analyse et de Traduction des Commandes

Ce sous-module joue un rôle clé dans la compréhension et l'interprétation des instructions vocales converties en texte. Il est divisé en trois étapes principales :

a. Identification et Remplacement des Synonymes

- Analyse du texte pour détecter les mots-clés correspondant aux actions connues.
- Remplacement des synonymes pour normaliser la commande (exemple : "continuer tout droit " devient "avancer tout droit").

b. Segmentation et Structuration du Texte

- Découpage de l'instruction complète en segments logiques.
- Détection des différentes actions et de leurs paramètres associés.

c. Conversion des Segments en Requêtes JSON

- Chaque segment est transformé en une **commande JSON** compréhensible par le module de simulation.
- Exemple de conversion :
    - **Commande orale** : "Progresser de 300 et tourner à droite."
    - **Texte segmenté** : "/avancer de 300 et tourner à /droite"
    - **Commande JSON générée** :

      [{"action": "avancer", "valeur": 300}, {"action": "droite", "valeur": 90}]

- Ce format permet au module de simulation d'exécuter les actions précisément.

3. Module d'Exécution et Simulation

Ce module interprète les commandes JSON et les exécute dans l'environnement de simulation. Il assure :

- La vérification de la validité des actions.
- L'envoi des instructions à l'interface de simulation pour un rendu visuel et une exécution robotique si nécessaire.
- La gestion des erreurs ou des commandes ambiguës en demandant des précisions à l'utilisateur.
-

III. Mode IHM

Ce mode est similaire au mode vocal, sauf que le système utilise les entrées provenant de l'utilisateur via une interface homme-machine.
Le mode IHM fonctionne exactement de la même manière que le mode vocal, sauf qu'au lieu d'utiliser des instructions vocales en entrée, il prend du texte en entrée et génère les commandes associées. Ainsi, les mêmes modules utilisés dans la fonctionnalité vocale pour le traitement de la requête (segmentation, etc.) sont également utilisés dans ce module.

IV. Mode Manuel - Contrôle via Clavier

Le **mode manuel** permet à l'utilisateur de **contrôler directement** le robot via un clavier, en effectuant des déplacements et des rotations en temps réel tout en prenant en compte les obstacles et les limites de l'environnement.

Fonctionnement :

1. **Commandes Clavier :**
    - **Avancer** : Flèche **haut (UP)**.
    - **Reculer** : Flèche **bas (DOWN)** (empêché si un obstacle est détecté).
    - **Tourner à gauche** : Flèche **gauche (LEFT)** (**5°** par pression).
    - **Tourner à droite** : Flèche **droite (RIGHT)** (**5°** par pression).
    - **Quitter** : Touche **Échap (ESC)** pour revenir à l'état précédent.
2. **Gestion des Déplacements :**
    - Le robot avance/recule en fonction de sa vitesse et **vérifie** la présence d'obstacles.
    - Si un obstacle critique est détecté en arrière, **le déplacement est bloqué** et un message d'erreur s'affiche.
    - Le robot ne peut pas sortir des **limites de l'environnement**, et une alerte est déclenchée si nécessaire.
3. **Réactivité et Sécurité :**
    - **Rotation fluide** sur place pour ajuster l'orientation.

- **Contrôle en temps réel** : chaque action est exécutée immédiatement dès la pression d'une touche.
- **Messages d'erreur et de confirmation** lorsque le mouvement est bloqué par un obstacle ou une limite.

4. **Cycle d'Exécution :**
   - **Activation** : Affichage des instructions et attente des commandes clavier.
   - **Exécution** : Lecture des touches pressées et mise en mouvement du robot.
   - **Vérification des obstacles et des limites** avant chaque action.
   - **Fin du Mode** : L'utilisateur peut quitter en appuyant sur **ESC**.
   -

V. Mode Auto - Déplacements Autonomes

Le **mode automatique** permet au robot de se déplacer **de manière autonome** en prenant ses propres décisions de mouvement tout en évitant les obstacles. Son fonctionnement repose sur des algorithmes programmés qui génèrent des déplacements **aléatoires** et répétés jusqu'à l'arrêt manuel.

Fonctionnement :

1. **Déplacement Aléatoire**
   - Rotation entre **-90° et +90°** à chaque itération pour varier la direction.
   - Translation sur une distance aléatoire entre **0 et 250 unités** pour rendre le mouvement imprévisible.
2. **Contrôle du Mode**
   - Le mode **fonctionne en continu** tant que l'utilisateur ne presse pas la touche **Échap (ESC)**.
   - Avant de démarrer, le robot **désactive le mode vocal** (speak) pour éviter toute interférence.
   - Une fois arrêté, **le mode vocal est restauré** à son état initial.
3. **Cycle d'Exécution**
   - **Initialisation** : Message de bienvenue et sauvegarde des paramètres.
   - **Mouvement** : Alternance de rotations et de translations aléatoires.
   - **Arrêt** : Pression de la touche ESC, réactivation des paramètres initiaux.

Conclusion

Ce mode permet au robot d'**explorer un environnement sans intervention humaine**, ce qui peut être utile pour des applications en **automatisation** ou **cartographie**.

VI. Option de Configuration : Settings

Description du fichier de configuration settings.json

Le fichier **settings.json** est un élément clé permettant de configurer et d'adapter le fonctionnement d'un programme sans modifier son code source. Il stocke plusieurs paramètres qui influencent son comportement, son interface et son interaction avec l'utilisateur. Ce fichier est essentiel pour la personnalisation et l'optimisation du programme selon les préférences et besoins des utilisateurs.

Principaux paramètres et leur rôle :

1. **Mode Débogage (debug)**
   Ce paramètre active ou désactive l'affichage des messages de diagnostic. Lorsqu'il est activé (`true`), le programme génère des logs détaillés pour aider au développement et à la résolution d'erreurs. Désactivé (`false`), il fonctionne en mode standard sans afficher ces messages.
   - **Utilité** : Permet aux développeurs de mieux comprendre le fonctionnement interne du programme et de détecter les anomalies.
   - **Valeur par défaut** : `true` (mode débogage activé).
2. **Mode Vocal (speak)**
   Ce paramètre active l'utilisation de la synthèse vocale. Lorsque `speak` est à `true`, le programme peut lire des messages à haute voix pour l'utilisateur. Si `false`, il fonctionne uniquement avec des interfaces visuelles ou textuelles.
   - **Utilité** : Favorise l'accessibilité pour les personnes malvoyantes ou permet une interaction mains libres.
   - **Valeur par défaut** : `false` (mode vocal désactivé).
3. **Affichage (`display`)**
   Ce paramètre contrôle si l'interface graphique est activée (`true`) ou désactivée (`false`). Lorsqu'elle est désactivée, l'interaction avec le programme se fait uniquement via le terminal ou la synthèse vocale.
   - **Utilité** : Permet de choisir entre une interface graphique (GUI) ou une interaction en ligne de commande.
   - **Valeur par défaut** : `true` (affichage activé).
4. **Langue du Système (`current_language`)**
   Ce paramètre définit la langue principale utilisée par le programme pour l'affichage des menus et messages. L'utilisateur peut sélectionner une langue parmi celles disponibles dans le fichier de configuration.
   - **Utilité** : Adapte l'interface en fonction de la langue choisie pour une meilleure expérience utilisateur.
   - **Valeur par défaut** : `'fr'` (Français).
5. **Langues Disponibles (`LANGUAGES`)**
   Une liste des langues prises en charge par le programme. Cela permet de gérer les traductions et d'ajouter facilement de nouvelles langues sans modifier le code source.

- **Utilité** : Offre une expérience utilisateur multilingue et adaptable à un public international.
- **Exemple** : `["fr", "en"]` pour le français et l'anglais.

6. **Textes du Menu (`MENU_TEXT`)**
   Les textes des menus et messages sont stockés sous forme de paires clé-valeur pour chaque langue. Par exemple, `"Bienvenue dans le menu principal !"` peut être traduit en anglais par `"Welcome to the main menu!"`.
   - **Utilité** : Facilite la traduction et l'adaptation du programme à différentes langues et cultures.
7. **Programmes Externes (`PROGRAMS`)**
   Ce paramètre stocke les chemins d'accès aux logiciels externes que le programme peut exécuter. Ces logiciels peuvent être utilisés pour la synthèse vocale, l'analyse d'image ou d'autres fonctionnalités avancées.
   - **Utilité** : Permet une gestion centralisée des logiciels associés, simplifiant leur intégration et leur mise à jour.

VII. Bilans Personnelles

TAYOU MBEDE Ryan Niel

- **Implication dans le déroulement du projet**
  Dans le projet j'ai été impliqué dans toutes les tâches d'intégrations et de conceptions globale.En particulier, j'ai proposé l'architecture de communication entre les différents modules qui a été retenue pour notre projet. Mais arrivé au moment de la conception, j'ai été chargé avec Adji de concevoir et de développer le module pour le traitement d'images. Au final j'ai conçu et développé, la reconnaissance de forme, de couleurs et la détermination de la position. Puis ayant terminé beaucoup plus tôt que prévu, j'ai ajouté des fonctionnalités supplémentaires comme l'analyse complète de l'image en fonction de ce que l'utilisateur entre et la détection d'objets qui ne sont ni des balles ni des cubes. Puis je me suis penché sur la question de segmentation de l'image. Pour pouvoir traiter des images contenant plusieurs objets de même couleur. Et là j'ai travaillé avec Imran.
- **Difficultées**
  Premièrement j'avais jamais eu a faire un dossier de specification donc je ne savais pas exactement ce qui était demandé à l'intérieur. Deuxièmement, j'ai eu des difficultés pour la mise en place de l'algorithme du traitement d'image. Mais la plus grande difficultées à laquelle j'ai fait face était le traitement d'une image ''I'' contenant plusieurs objets de même couleurs ''C'' sachant que l'on cherche à analyser les objets de couleurs ''C'' dans ''I''
- **Les compétences acquises**
  - Outils : J'ai appris à utiliser des outils de projet tels que git, google docs
  - Programmation : J'ai renforcé mes connaissances en C (structure, pointeurs, FILE) et appris de nouvelles choses, comme la manipulation des tableaux dynamiques, ou encore des pointeurs de fonctions (*qsort()*). Je sais aussi désormais mieux utiliser les commandes UNIX, les filtres en particuliers.
  - Bonnes pratiques : j'ai appris à réaliser à concevoir calmement l'architecture et les algorithmes avant d'attaquer le code. Commenter suffisamment mon code. Et surtout produire un code maintenable
  - Traitement d'images : je peux désormais proposer une chaîne de traitement qui répond à un cahier de charge précis. En fonction d'un cahier de charge je suis capable de proposer une solution pour le traitement de la forme de la couleur ou de la position lorsque les conditions d'éclairage sont maîtrisées.

WACHILL Imran

- **Implication dans le déroulement du projet**
  Dans ce projet, j'ai d'abord été impliqué dans la gestion de la simulation. J'ai conçu l'environnement virtuel via turtle, comprenant la modélisation du robot et de la pièce, puis j'ai implémenté les mouvements de base, tels que les rotations et les trajectoires. Une fois cette partie réalisée, Djibril a pris le relais pour la poursuivre et l'améliorer.
  Par la suite, j'ai rejoint Niel sur la partie qui lui avait été attribuée. Je l'ai aidé à développer la lecture d'image, en apportant des améliorations et en facilitant le traitement des données. J'ai également réalisé un script Python permettant d'afficher une croix sur un pixel donné afin de représenter visuellement un objet détecté. Bien que ce script ait été fonctionnel, il n'a finalement pas été intégré à la solution finale.
  Enfin, j'ai conçu une fonction de segmentation à la demande Niel, afin d'extraire et de générer plusieurs images contenant chacune un objet à partir d'une image regroupant plusieurs objets. Cette tâche a été de loin la plus complexe, tant sur le plan technique que sur celui de l'intégration dans le projet.
- **Difficultées**
  La principale difficulté pour moi a été l'apprentissage du langage C, n'ayant jamais codé dans ce langage avant cette année. Lorsque j'ai rejoint la partie traitement d'image, j'ai dû m'adapter rapidement, car le développement était déjà bien avancé.
  Pour bien comprendre le fonctionnement du projet, j'ai dû analyser en profondeur le code existant, en naviguant entre les fichiers .c et .h, afin de saisir la structure globale et les interactions entre les différentes fonctions. Cette phase d'adaptation m'a demandé du temps et des efforts, mais elle m'a permis d'aider Niel sur cette partie.
- **Les compétences acquises**
  Utilisation de LLDB pour le débogage, permettant d'identifier et de corriger efficacement les erreurs dans du code C.
  Approfondissement du langage C, en particulier sur la manipulation des pointeurs, la gestion de la mémoire dynamique et la structuration du code en plusieurs fichiers (.c et .h).
  Consolidation des notions de C, notamment l'implémentation et l'utilisation des piles et files dynamiques pour le stockage et le traitement des objets détectés dans les images.

BA Djibril

- Implication dans le déroulement du projet
  - ❖ Au sein de ce projet, j'ai d'abord été responsable du **traitement vocal**. Mon rôle consistait à développer la gestion des **commandes vocales simples et complexes**, ainsi que la prise en charge des **synonymes** pour améliorer la compréhension des instructions. J'ai également défini les **échanges de données** entre les modules afin d'assurer une communication fluide.
  - ❖ En parallèle, j'ai implémenté plusieurs **modes de fonctionnement** du système : le **mode vocal**, permettant un contrôle par la voix ; le **mode IHM**, offrant une interface utilisateur ; le **mode automatique**, où le robot navigue seul ; et le **mode manuel**, piloté via le clavier.
  - ❖ Une fois ces tâches achevées plus tôt que prévu, j'ai repris la gestion du **module simulation**, initialement sous la responsabilité d'Imran Wachill, lui permettant ainsi de se concentrer sur le **module image** et d'accélérer l'avancement du projet. Dans cette partie, j'ai intégré les **modules développés**, conçu les **algorithmes de navigation autonome** et mis en place la **gestion des obstacles**, incluant leur **détection des collisions et les politiques de contournement de ces obstacles** selon les trajectoires du robot.
  - ❖ De plus, j'ai pris en charge l'**intégration des différents modules du système**, veillant à la cohérence des échanges de données et à la synergie entre les différentes parties pour garantir un fonctionnement optimal du robot.
- Difficultés

  Concernant les difficultés rencontrées sur le projet, la plus grande a été de cerner précisément les problématiques liées à ce projet, de déterminer la démarche à adopter et de configurer les équipes de manière optimale pour aboutir à une solution efficace. Un autre challenge majeur a été de s'assurer que la solution trouvée répondait bien aux attentes du client. Ne sachant pas exactement si nos propositions correspondaient à ses besoins, nous avons dû organiser de nombreuses réunions avec lui pour affiner et valider nos choix.

- Les compétences acquises

  Ce projet m'a permis de m'exercer davantage en programmation et d'approfondir mes compétences en algorithmes de navigation pour robots, ce qui me sera très utile étant donné que je prévois de progresser dans le domaine de la robotique mobile.

  Cependant, l'un des points forts de ce projet est qu'il nous a rapprochés les uns des autres et nous a donné une expérience précieuse sur ce que signifie travailler en équipe.

BAKHTI Hela

Dans le cadre de ce projet, j'ai travaillé en binôme sur la mise en place de la commande vocale.

- Compétences développées

Ce projet m'a permis de renforcer mes compétences en programmation C.

- Travail en équipe

Le travail en équipe s'est globalement bien déroulé et a été bien organisé.

Cependant, certains membres ayant un niveau plus avancé progressaient rapidement, ce qui a ralenti la répartition des tâches et la contribution. Malgré cela, l'organisation claire du projet nous a permis de maintenir un bon rythme.

- Perspectives

Ce projet a été une expérience enrichissante, qui m'a permis de mieux travailler en équipe. À l'avenir, j'aimerais encore progresser en programmation pour être plus à l'aise et plus efficace.

MEHREZ Wissal

Progression en programmation

- Après la phase de PFR1, je me sens plus à l'aise avec la programmation, car je n'avais jamais codé auparavant.
- J'ai fait l'effort d'apprendre et de comprendre de nouvelles compétences en Python, notamment en traitement vocal.

Difficultés rencontrées

- Comprendre les concepts de base en Python et la structuration du code.
- Assimiler de nouvelles notions comme le traitement vocal.
- Intégrer efficacement mes connaissances dans un projet collaboratif.

Contributions au projet

- Création d'un environnement virtuel pour gérer les dépendances et assurer le bon fonctionnement du projet.
- Participation à la partie vocale du projet.

Objectifs pour la phase 2

- Surmonter les défis liés à mon apprentissage de la programmation.
- Apporter plus de contributions au projet et renforcer mes compétences techniques.

SECK Adji Aram

- **Implication dans le projet**

J’ai travaillé en binôme sur le traitement d’image, en développant les fichiers pour l’analyse de la position, couleur et forme des objets. J’ai également participé à la structuration de l’architecture du module pour assurer une bonne intégration avec le reste du projet.

- **Difficultés et progression**

Ayant redoublé, j’avais rencontré des difficultés sur cette partie l’an dernier. Cette fois, j’ai mieux compris la structuration du code, l’organisation des fichiers et la partie du traitement d’image dans le projet. Cela m’a permis d’être plus efficace et d’apporter une contribution plus aboutie.

- **Compétences acquises**

Programmation en C : meilleure gestion des structures, des fichiers et des pointeurs.

Traitement d’image : approfondissement des méthodes d’analyse de formes, couleurs et positions.

Architecture logicielle : conception d’un module structuré et fonctionnel.

Travail en équipe : meilleure coordination et intégration des différentes parties du projet.

Grâce à cette expérience, j’ai surmonté mes difficultés passées et acquis une vision plus claire du projet dans son ensemble. J’ai développé une approche plus structurée et méthodique, me permettant d’être plus efficace et de mieux collaborer avec mon équipe.