Module : OpenCV & Computer Vision

Encadré par : M. AMMOUR Alae

Semestre : 08

Filière : 2ème année, Cycle d'ingénieur en Robotique et Cobotique

RAPPORT PROJET FIN DE MODULE

Sujet :

Réalisation d'un robot mobile autonome à l'aide de la transformée de Hough

Réalisé par :

Bouarfa LAHMAR – Mohamed BOFARHA – Hiba MOHSINE

Année universitaire : 2024/2025

SOMMAIRE

Introduction 2

2. Cahier des charges 4

2.1 Spécifications fonctionnelles 4

2.2 Spécifications techniques 4

2.3 Contraintes 4

3. Matériel et outils utilisés 5

3.1 Plateforme robotique 5

3.2 Capteur 6

3.3 Bibliothèques logicielles 6

3.4 Environnement de développement 7

4. Méthodologie 8

4.1 Architecture globale du système 8

4.2 Étapes du traitement d'image 8

4.3 Application de la transformée de Hough 9

4.4 Interprétation des résultats 9

4.5 Contrôle du robot 9

4.6 Logigramme de contrôle 10

5. Réalisation et implémentation 11

6. Résultats et analyses 13

7. Conclusion et perspectives 17

INTRODUCTION

Contexte du projet

Ce projet s'inscrit dans le cadre du module **Computer Vision**, enseigné en deuxième année du cycle d'ingénieur en **Robotique et Cobotique**. Il a pour objectif de mettre en œuvre les notions théoriques abordées pendant le cours, telles que le traitement d'image, la détection d'objets et l'analyse de scènes, dans un projet concret et expérimental.

Notre choix s'est porté sur la réalisation d'un **robot mobile autonome**, capable de suivre une trajectoire à l'aide d'une caméra embarquée. Ce projet permet d'explorer des aspects essentiels de la robotique mobile, notamment :

— l'acquisition et le traitement d'images en temps réel,

— l'interprétation visuelle de l'environnement,

— et la prise de décision autonome pour le contrôle du mouvement.

Il constitue également une excellente opportunité pour développer des compétences pratiques en programmation, intégration de capteurs, et conception de systèmes embarqués.

Problématique

Les robots autonomes jouent un rôle croissant dans les secteurs industriels, logistiques, urbains (véhicules intelligents), et même domestiques (aspirateurs intelligents, robots de surveillance). L'un des défis majeurs dans ce domaine est de permettre à ces robots de percevoir leur environnement visuellement, sans dépendre de balises externes ni d'interventions humaines constantes.

La problématique centrale que nous cherchons à résoudre est la suivante :

> *Comment permettre à un robot de suivre une trajectoire ou une route sans intervention humaine, en se basant uniquement sur la vision par ordinateur ?*

Ce défi implique plusieurs sous-problèmes techniques :

— la détection fiable d'une ligne ou d'une trajectoire dans des conditions réelles (bruit, lumière variable),

— le traitement en temps réel des images capturées par une caméra embarquée,

— la conversion des informations visuelles en commandes de déplacement adaptées.

Répondre à cette problématique demande une synergie entre la vision par ordinateur, le traitement du signal, et le contrôle moteur, tout en prenant en compte les contraintes matérielles du système embarqué.

Objectifs

Les objectifs principaux de ce projet sont les suivants :

— Mettre en pratique les concepts vus durant le module **Computer Vision** à travers un projet concret, en lien avec notre domaine d'étude, la **robotique**.

— Comprendre en profondeur les algorithmes fondamentaux de la vision par ordinateur, notamment ceux liés à la détection et au suivi de trajectoires.

— Développer une application fonctionnelle capable de traiter des images en temps réel et d'adapter le comportement du robot en conséquence.

— Acquérir des compétences pratiques en programmation, traitement d'image et contrôle embarqué.

— Explorer les différentes techniques de prétraitement d'image pour améliorer la qualité des données utilisées par le robot.

— Évaluer la performance et la robustesse du système de suivi de ligne dans des conditions variées.

Présentation générale de la solution proposée

La solution développée repose sur l'utilisation de la transformée de Hough, une méthode robuste de détection de formes géométriques, pour identifier les lignes tracées sur le sol qui matérialisent la trajectoire à suivre. Après acquisition de chaque image, un prétraitement est appliqué : conversion en niveaux de gris, réduction du bruit via un flou gaussien, binarisation pour isoler la ligne, puis un nettoyage morphologique par érosion et dilatation. Ces étapes garantissent une segmentation efficace des lignes dans des conditions d'éclairage variables et en présence de perturbations visuelles.

Les lignes détectées par la transformée de Hough sont analysées pour extraire leur position horizontale moyenne, permettant ainsi d'estimer la position relative du robot par rapport à la trajectoire centrale. En fonction de cette position, un algorithme de prise de décision détermine la commande moteur appropriée : avancer tout droit, tourner à droite ou à gauche, afin d'assurer un suivi précis et stable de la route.

La capture d'images est réalisée à l'aide d'une caméra PiCamera intégrée au Raspberry Pi, garantissant une acquisition rapide et adaptée aux contraintes embarquées. Le traitement d'image et les calculs nécessaires sont effectués en temps réel grâce aux bibliothèques Python OpenCV et NumPy, ce qui permet une réactivité optimale du système. L'ensemble du système assure ainsi un déplacement autonome fluide, capable de s'adapter dynamiquement aux variations de l'environnement et aux déviations potentielles du robot.

2. CAHIER DES CHARGES

2.1 Spécifications fonctionnelles

— Le robot doit être capable de capturer en continu des images de son environnement à l'aide d'une caméra embarquée.

— Il doit détecter la ligne tracée au sol servant de guide de trajectoire.

— Il doit traiter les données visuelles en temps réel pour analyser la position de cette ligne.

— Le robot doit décider dynamiquement d'une action (tourner à gauche, à droite, avancer ou s'arrêter) en fonction de la ligne détectée.

— Il doit corriger sa trajectoire automatiquement, sans intervention humaine.

— Le système doit afficher en temps réel l'état de la direction (GAUCHE, DROITE, AVANCER, STOP) sur l'interface graphique.

2.2 Spécifications techniques

— Utilisation de la caméra PiCamera2 connectée à un Raspberry Pi pour la capture vidéo.

— Traitement d'image en Python à l'aide des bibliothèques OpenCV, NumPy et Picamera2.

— Application d'un pipeline d'image : conversion en niveaux de gris $\rightarrow$ floutage $\rightarrow$ seuillage $\rightarrow$ opérations morphologiques $\rightarrow$ transformée de Hough.

— Calcul des coordonnées moyennes des lignes détectées et prise de décision basée sur la position relative de ces lignes.

— Contrôle des moteurs via un module (motors.py) qui contrôle les roues du robot.

— Affichage graphique de l'image traitée, des lignes détectées et du masque binaire, côte à côte dans une fenêtre de visualisation.

— Arrêt d'urgence du robot en cas d'absence de ligne détectée.

2.3 Contraintes

Matérielles :

— Utilisation obligatoire de la PiCamera, compatible avec le Raspberry Pi.

— Autonomie limitée par la batterie du robot et la consommation des moteurs.

— Position de la caméra pouvant limiter certaines améliorations futures du projet.

Logicielles :

— Utilisation restreinte aux bibliothèques Python compatibles avec le matériel utilisé.

— Temps de latence très faible exigé entre la capture d'image et la réaction du robot.

— Compatibilité du code avec le système d'exploitation du Raspberry Pi.

Temps :

— Projet réalisé dans le cadre du projet de fin de module en Computer Vision, à livrer dans un délai limité.

— Intégration, tests et ajustements du code et du matériel réalisés dans un temps contraint.

3. MATÉRIEL ET OUTILS UTILISÉS

3.1 Plateforme robotique

Raspberry Pi 4

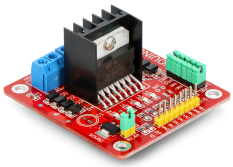
C'est le cerveau du robot. Il exécute le code Python développé pour le projet, ce qui inclut le traitement d'image en temps réel à partir des données capturées par la caméra PiCamera. Grâce à sa puissance de calcul suffisante, il analyse les images pour détecter les lignes au sol, interprète leur position et détermine la trajectoire optimale à suivre. En fonction de ces analyses, il prend des décisions instantanées telles que tourner à gauche, à droite, avancer ou s'arrêter afin d'assurer une navigation autonome précise. La carte Raspberry Pi communique ensuite avec le module de contrôle des moteurs pour envoyer des commandes adaptées, ajustant la vitesse et la direction des roues en continu. De plus, elle gère l'affichage de l'interface graphique qui permet de visualiser en direct le traitement d'image et l'état du robot, facilitant ainsi le suivi et le débogage du système.

Moteurs à courant continu (DC)

Quatre moteurs à courant continu (DC) sont utilisés pour propulser le robot, chacun étant associé à une roue. Ces moteurs fournissent la force nécessaire pour déplacer le robot dans différentes directions. Leur vitesse et leur sens de rotation sont finement contrôlés grâce au module de commande (module L298N), ce qui permet d'adapter précisément les mouvements. En modulant la vitesse des moteurs de gauche et de droite de manière indépendante, le robot peut avancer en ligne droite, tourner doucement ou effectuer des virages plus prononcés. Cette flexibilité dans le contrôle moteur est essentielle pour que le robot puisse suivre avec précision les lignes tracées au sol et corriger sa trajectoire en temps réel. De plus, les moteurs peuvent être arrêtés instantanément pour assurer la sécurité du système en cas de perte de repères visuels ou d'obstacle détecté.

Module L298N

C'est un pont en H qui permet de contrôler la vitesse et la direction des moteurs DC à partir des signaux envoyés par le Raspberry Pi. Il agit comme un intermédiaire entre la carte de commande et les moteurs. Grâce à ce module, il est possible d'inverser le sens de rotation des moteurs et de moduler leur vitesse via la modulation de largeur d'impulsion (PWM), assurant un contrôle précis et fluide des mouvements du robot.



Châssis imprimé en 3D

Support structurel sur lequel tous les composants (Raspberry Pi, caméra, moteurs, batteries) sont montés. Il assure la stabilité et la mobilité du robot. Conçu en impression 3D, ce châssis léger mais robuste permet une bonne répartition du poids et protège les éléments électroniques des chocs et vibrations durant les déplacements.

Piles lithium rechargeables

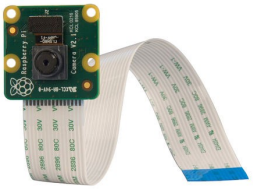
Elles alimentent les moteurs en électricité. Elles fournissent la puissance nécessaire à la locomotion du robot sans dépendre d'une source externe.

Powerbank

Fournit une alimentation électrique stable au Raspberry Pi. Cela permet à la carte de fonctionner de manière autonome sans être reliée à une prise secteur.

3.2 Capteur

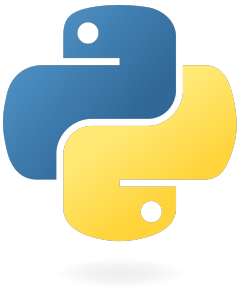
Pi Camera v2



Caméra embarquée utilisée pour capturer en temps réel les images de la route. Ces images sont traitées pour détecter les lignes au sol servant de guide au robot. Elle joue un rôle clé dans la vision du robot.

3.3 Bibliothèques logicielles

Python



Langage de programmation utilisé pour développer tout le système de commande et de traitement d'image. Il est choisi pour sa simplicité, sa large communauté et la richesse de ses bibliothèques, notamment OpenCV pour le traitement d'images. De plus, Python est parfaitement compatible avec le Raspberry Pi, facilitant ainsi le développement et le déploiement rapide du projet.

OpenCV

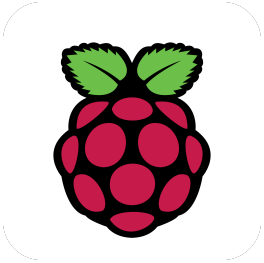
Bibliothèque spécialisée en traitement d'image. Elle est utilisée ici pour convertir les images en niveaux de gris, appliquer des filtres, des seuils, des transformations morphologiques, et surtout pour utiliser la transformée de Hough afin de détecter les lignes.

NumPy

Bibliothèque numérique utilisée pour effectuer des calculs rapides sur les matrices d'image, comme le calcul de la moyenne des coordonnées des lignes détectées.

3.4 Environnement de développement

Raspberry Pi OS (anciennement Raspbian)



Raspberry Pi OS est un système d'exploitation léger et open-source basé sur Linux, spécialement conçu pour les cartes Raspberry Pi. Il est optimisé pour les performances sur architecture ARM, ce qui le rend idéal pour les projets embarqués. Il permet de coder en Python ou en C++, d'exécuter des scripts de traitement d'image en temps réel, et de contrôler les composants du robot (moteurs, capteurs, caméra, etc.) grâce à des bibliothèques comme `OpenCV`, `Picamera2`, ou `RPi.GPIO`. Son interface graphique simple facilite également le débogage et la visualisation en direct du comportement du robot.

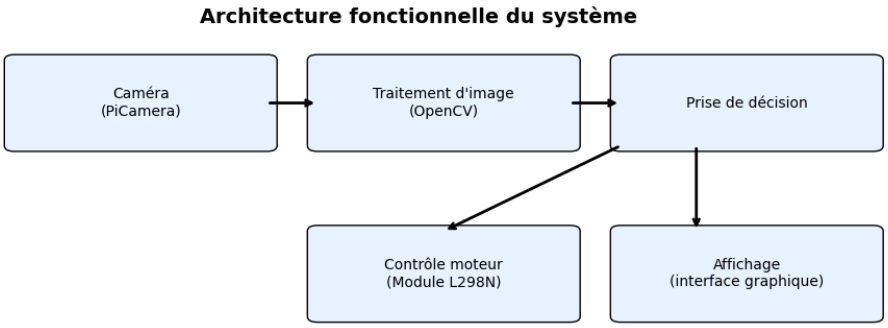
4. MÉTHODOLOGIE

4.1 Architecture globale du système

Le système repose sur une chaîne complète de capture, de traitement et d'action :

— Capture d'image avec la PiCamera.
— Traitement d'image sur le Raspberry Pi (filtrage, seuillage, détection de lignes).
— Prise de décision basée sur l'analyse de la position des lignes.
— Contrôle des moteurs pour corriger la trajectoire.
— Affichage visuel pour le suivi en temps réel.

Le schéma suivant résume les différentes étapes :



4.2 Étapes du traitement d'image

Chaque image passe par les étapes suivantes :
a) Prétraitement

— **Grayscale (niveaux de gris)** : cette étape consiste à convertir l'image couleur en une image en niveaux de gris. Cela permet de se concentrer uniquement sur l'intensité lumineuse des pixels, en éliminant les informations de couleur inutiles pour la détection de ligne. *But : simplifier l'image et réduire les données à traiter.*

— **Gaussian Blur (flou gaussien)** : un flou gaussien est appliqué afin de réduire le bruit et les détails mineurs dans l'image. Cela rend les contours de la ligne plus homogènes et atténue les variations locales. *But : éviter les fausses détections causées par des irrégularités ou des ombres.*

— **Threshold (binarisation inverse)** : cette étape transforme l'image en noir et blanc en inversant les valeurs (ligne blanche sur fond noir devient noir sur fond blanc). Cela permet d'isoler efficacement la ligne. *But : obtenir une image binaire claire où la ligne se détache nettement du fond.*

— **Morphologie (érosion + dilatation)** : une érosion suivie d'une dilatation est appliquée pour supprimer les petits bruits et renforcer la ligne principale. Cette combinaison constitue une fermeture morphologique. *But : nettoyer l'image binaire et améliorer la continuité de la ligne détectée.*

4.3 Application de la transformée de Hough

La transformée de Hough est un algorithme utilisé pour détecter les lignes droites dans une image binaire.

a) Principe

Chaque pixel blanc (valeur 255) dans l'image peut appartenir à une infinité de lignes. La transformée de Hough transforme chaque point en une courbe dans l'espace des paramètres (,), où :

— **(rho)** : distance entre l'origine et la ligne.

— **(thêta)** : angle d'orientation de la ligne par rapport à l'axe horizontal.

L'intersection de ces courbes dans l'espace (,) indique la présence d'une ligne réelle : on obtient des pics dans un accumulateur, correspondant aux lignes dominantes de l'image.

b) Utilisation dans ce projet

— Utilisation de `cv2.HoughLinesP()` (version probabiliste).

— Filtrage des segments selon leur position, longueur et orientation.

— Détermination de la ligne principale (centrale) pour guider la trajectoire du robot.

c) Post-traitement

— Calcul de la position moyenne des extrémités des lignes.

— Comparaison avec le centre de l'image pour estimer un éventuel décalage.

— Prise de décision selon la position : tourner à gauche, droite, avancer, ou s'arrêter.

4.4 Interprétation des résultats

Condition	Action du robot
Ligne centrée	Avancer
Ligne détectée à gauche	Tourner à gauche
Ligne détectée à droite	Tourner à droite
Aucune ligne détectée	Stop (arrêt de sécurité)

Le robot réagit en temps réel selon la position relative des lignes détectées dans le champ de vision.

4.5 Contrôle du robot

Les commandes sont traduites en signaux moteurs via le script `motors.py`.

— Le Raspberry Pi contrôle les moteurs à travers le module L298N.

— Les vitesses et sens de rotation des roues sont ajustés dynamiquement.

Exemples :

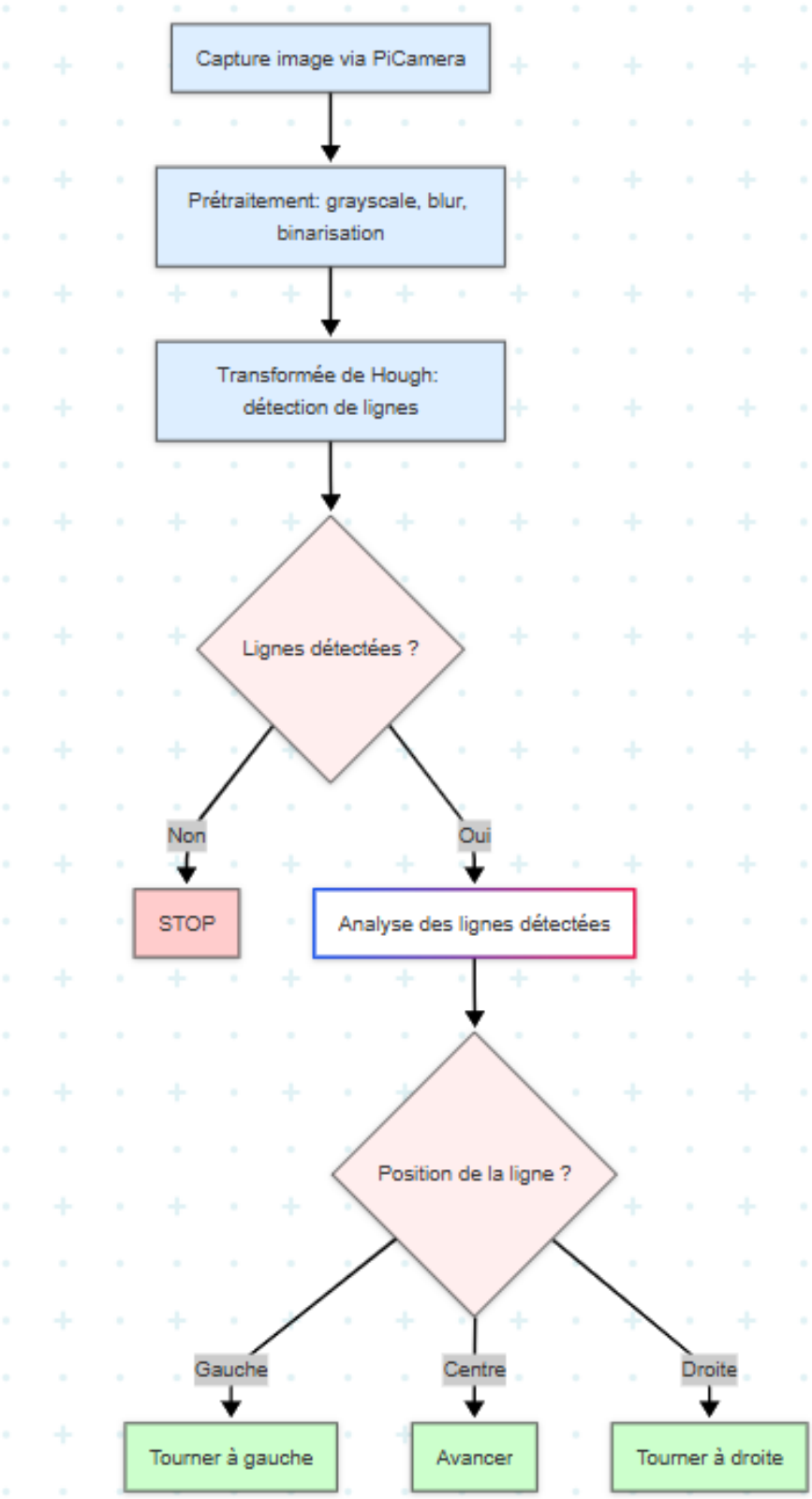
— **Avancer** : tous les moteurs tournent vers l'avant.

— **Tourner à gauche** : les roues droites tournent, les roues gauches ralentissent ou s'arrêtent.

— **Stop** : tous les moteurs sont arrêtés.

4.6 Logigramme de contrôle

Voici la logique générale du système de contrôle :



5. RÉALISATION ET IMPLÉMENTATION

5.1 Architecture logicielle

La réalisation du projet repose sur une architecture logicielle modulaire divisée en plusieurs parties :

- **Acquisition d'image** : Capture vidéo en continu via la caméra PiCamera2, connectée au Raspberry Pi.
- **Traitement d'image** : Application d'un pipeline d'analyse d'image comprenant la conversion en niveaux de gris, le filtrage, le seuillage, les opérations morphologiques, puis la transformée de Hough pour détecter les lignes.
- **Interprétation des lignes détectées** : Calcul des coordonnées moyennes des lignes, évaluation de leur position relative par rapport au robot, permettant de déterminer l'orientation du robot par rapport à la trajectoire.
- **Contrôle du robot** : Envoi des commandes adaptées au module de pilotage des moteurs (module `motors.py`) pour ajuster la direction et la vitesse en fonction des résultats du traitement d'image.
- **Interface utilisateur** : Affichage en temps réel des images traitées, des lignes détectées, ainsi que de l'état décisionnel (avancer, tourner à gauche/droite, stop).

5.2 Acquisition et traitement des images

- Le module de capture utilise la bibliothèque `Picamera2` pour récupérer des frames en continu.
- Chaque image est convertie en niveaux de gris avec OpenCV (`cv2.cvtColor`).
- Un floutage gaussien est appliqué (`cv2.GaussianBlur`) pour réduire le bruit.
- Un seuillage adaptatif est ensuite effectué pour isoler la ligne sur le sol.
- Des opérations morphologiques (dilatation, érosion) permettent d'améliorer la qualité du masque binaire.
- La transformée de Hough (`cv2.HoughLinesP`) est appliquée pour détecter les segments de lignes dans l'image.

5.3 Analyse des lignes détectées

- Les coordonnées des lignes détectées sont extraites.
- Une moyenne des positions des lignes est calculée pour déterminer la trajectoire.
- En fonction de la position relative du robot à la ligne (centre, gauche, droite), une décision de mouvement est prise.
- Si aucune ligne n'est détectée, le robot s'arrête par sécurité.

5.4 Contrôle moteur

- Le module `motors.py` gère les signaux PWM envoyés au pont en H (module L298N).
- Les fonctions principales incluent :
  - `avancer()`

  - `tourner_gauche()`
  - `tourner_droite()`
  - `arreter()`
- Ces fonctions sont appelées dynamiquement en fonction de l'analyse visuelle en temps réel.
- Le contrôle est réalisé via des broches GPIO du Raspberry Pi configurées en mode PWM.

5.5 Intégration et tests

- Le code a été développé et testé en plusieurs phases, en simulant d'abord la détection de lignes sur des images statiques avant d'intégrer la capture vidéo en temps réel.
- Des tests ont été effectués pour régler les paramètres de la transformée de Hough (seuils, longueur minimale des segments, etc.).
- Le système a été testé dans différentes conditions d'éclairage et sur différents types de surfaces pour valider la robustesse de la détection.
- Le contrôle moteur a été validé sur la plateforme robotique pour assurer des déplacements fluides et précis.

6. RÉSULTATS ET ANALYSES

Présentation du robot utilisé

Le robot employé pour ce projet est une plateforme mobile autonome conçue spécifiquement pour la navigation assistée par vision. Le cœur du système est une carte Raspberry Pi 4, un micro-ordinateur compact et puissant, capable de gérer le traitement d'image en temps réel grâce à son processeur quadricœur et sa mémoire RAM suffisante. Cette carte assure le traitement des données visuelles capturées par la caméra ainsi que le contrôle des moteurs. Le châssis du robot est fabriqué en impression 3D, offrant une structure légère mais robuste, adaptée à l'intégration des différents composants tout en garantissant une bonne stabilité pendant les déplacements.

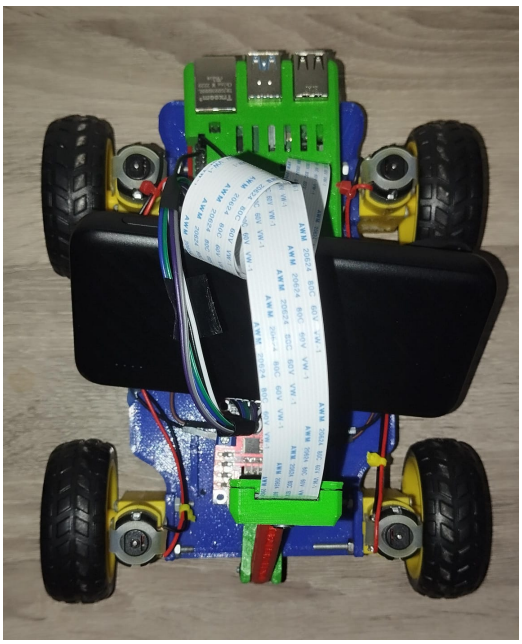
Pour la mobilité, le robot est équipé de quatre roues motrices actionnées par des moteurs à courant continu (DC). Ces moteurs sont contrôlés par un module électronique L298N, qui permet de gérer indépendamment la vitesse et la direction de chaque roue, offrant ainsi une grande précision dans les manœuvres et les ajustements de trajectoire. Cette configuration à quatre roues assure une meilleure adhérence et une stabilité accrue, même sur des surfaces légèrement irrégulières.

La perception de l'environnement est réalisée à l'aide d'une caméra PiCamera V2, un capteur caméra haute résolution spécialement conçu pour fonctionner avec le Raspberry Pi. Cette caméra capture en continu des images du sol, permettant au système de vision de détecter les lignes de guidage tracées sur le terrain. Les images sont ensuite traitées en temps réel par des algorithmes de traitement d'image développés en Python, utilisant notamment les bibliothèques OpenCV et NumPy. La PiCamera V2 offre un bon compromis entre qualité d'image et rapidité de traitement, ce qui est essentiel pour assurer une navigation fluide et réactive.

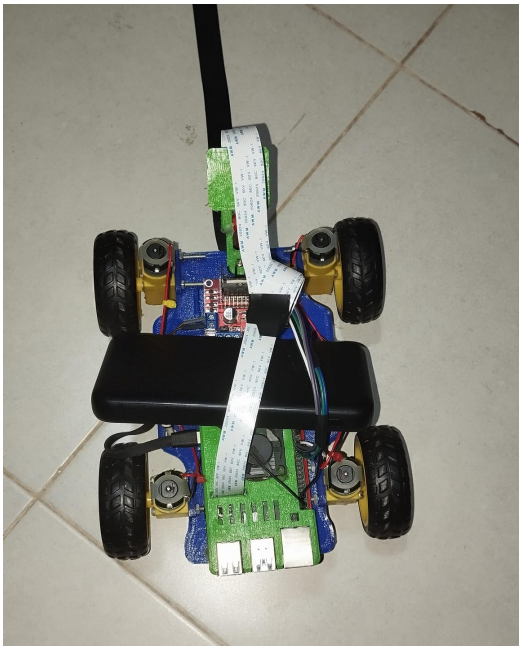
L'ensemble du matériel a été soigneusement sélectionné pour garantir une parfaite compatibilité entre les composants et pour répondre aux contraintes de performance et d'autonomie énergétique. La batterie lithium rechargeable alimente les moteurs tandis qu'une powerbank externe assure l'alimentation continue de la Raspberry Pi, ce qui permet au robot de fonctionner de manière autonome pendant une durée suffisante pour les tests et les démonstrations. Ce robot représente ainsi une plateforme efficace et flexible pour expérimenter des techniques avancées de vision par ordinateur appliquées à la robotique mobile.

Voici quelques images du robot en fonctionnement :

Robot vue de face



Vue de dessus

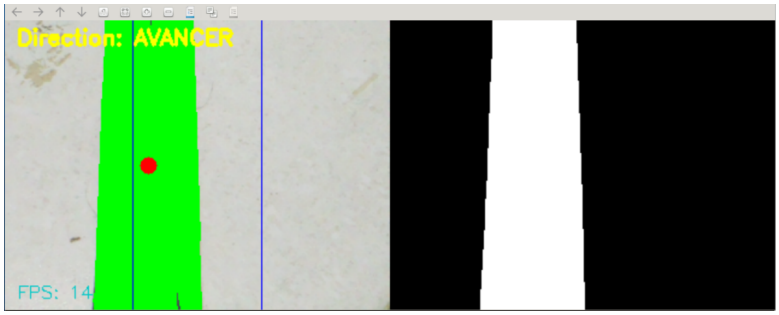


Robot suivant la ligne

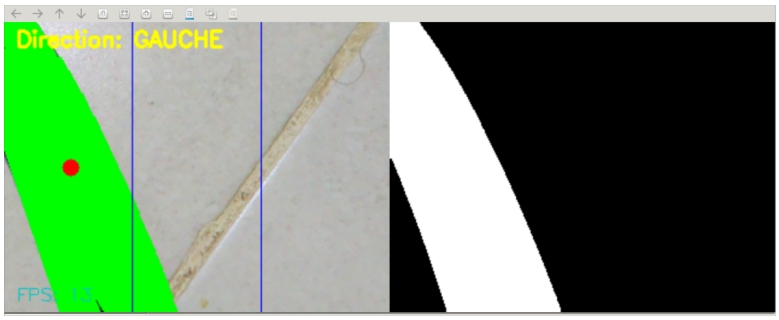
Robot vu de face

Captures d'écran des résultats

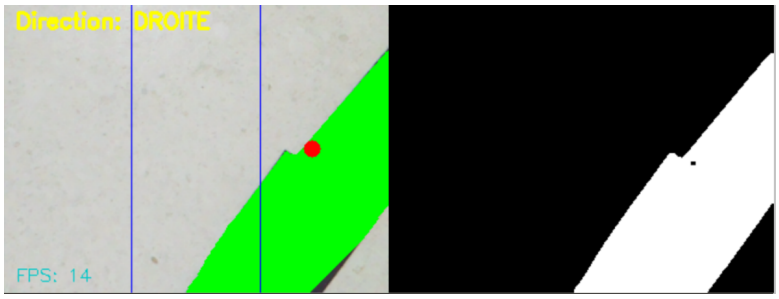
Les captures d'écran présentées ci-dessous montrent la fenêtre d'affichage du système en temps réel, qui reflète les quatre principaux comportements du robot : avancer, tourner à gauche, tourner à droite et s'arrêter. Cette interface graphique affiche non seulement l'image capturée par la caméra, mais aussi le résultat du traitement d'image, notamment la détection des lignes au sol qui servent de guide pour la navigation autonome. Chaque image illustre clairement comment le robot interprète son environnement visuel pour prendre des décisions instantanées et ajuster sa trajectoire en conséquence. Cette visualisation est essentielle pour le suivi et le débogage du système, car elle permet d'observer en direct la qualité de la détection des lignes et la réactivité du robot face aux différents scénarios rencontrés. Elle met également en évidence la robustesse du traitement d'image et la précision du contrôle moteur, assurant ainsi une navigation fluide et sécurisée. Ces résultats confirment que le robot répond efficacement aux commandes, même en présence de variations dans le tracé ou les conditions du sol.



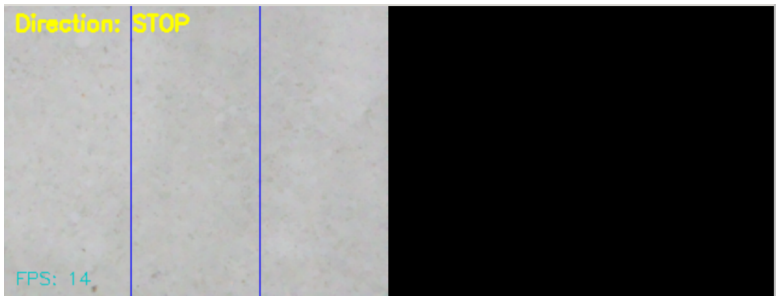
Robot avançant



Robot tournant à gauche



Robot tournant à droite



Robot à l'arrêt

Comparaison entre attentes et résultats réels

Le robot a globalement répondu aux attentes définies en début de projet. Il détecte correctement la ligne tracée dans la majorité des situations, avec un temps de réaction rapide et satisfaisant, conforme aux spécifications initiales. Les virages, qu'ils soient à gauche ou à droite, sont exécutés avec précision, assurant une trajectoire fluide et maîtrisée. De plus, l'arrêt automatique en cas d'absence de ligne a fonctionné de manière fiable lors des tests.

Cependant, certaines limites ont été observées, notamment lors de la navigation sur des surfaces très irrégulières ou présentant des obstacles imprévus. Dans ces conditions, le robot manifeste quelques instabilités et des écarts par rapport à la trajectoire idéale. Ces observations soulignent la nécessité d'optimiser davantage la stabilité du système et d'intégrer des mécanismes d'adaptation plus robustes.

Par ailleurs, bien que la détection des panneaux routiers soit fonctionnelle, l'intégration de techniques plus avancées pourrait améliorer la précision et la rapidité de reconnaissance, surtout dans des environnements plus complexes ou avec des variations d'éclairage importantes.

Ainsi, les résultats obtenus confirment la validité de la conception initiale tout en ouvrant la voie à des améliorations futures pour accroître la fiabilité et la polyvalence du robot.

7. CONCLUSION ET PERSPECTIVES

Bilan du projet

Le projet a permis de concevoir et développer un robot capable de détecter des lignes et obstacles avec une bonne réactivité. Les objectifs principaux ont été atteints dans la majorité des cas, notamment la reconnaissance des panneaux et la navigation autonome.

Cependant, certaines limitations ont été observées, telles que des difficultés sur des surfaces irrégulières ou en conditions d'éclairage défavorables. Ces contraintes n'ont pas permis d'exploiter pleinement le potentiel du système, mais elles constituent une base précieuse pour de futures optimisations.

Analyse des résultats

Les tests effectués ont montré une détection fiable et un comportement réactif du robot, conforme aux exigences initiales. La précision dans l'exécution des virages et des arrêts est satisfaisante, bien que perfectible sur des scénarios complexes.

Ces résultats valident la méthode choisie et démontrent la faisabilité du concept, tout en soulignant la nécessité d'améliorations pour garantir une performance robuste en conditions réelles variées.

Améliorations

Ce projet constitue une base solide pour de nombreuses améliorations et extensions futures. Plusieurs pistes peuvent être envisagées afin d'optimiser ses performances et d'enrichir ses fonctionnalités :

- Intégrer d'autres types de caméras, comme des caméras thermiques ou des caméras stéréo, pour améliorer la détection et la reconnaissance des panneaux routiers dans des conditions variées (faible luminosité, intempéries, etc.).
- Appliquer des techniques d'intelligence artificielle, notamment des réseaux de neurones convolutionnels (CNN), afin d'améliorer la précision et la robustesse de la détection par le robot.
- Utiliser la méthode de détection par mélange de Gaussiennes (MOG2) pour une meilleure identification et suivi des obstacles en mouvement dans l'environnement du robot.
- Développer un système de fusion de données multi-capteurs (caméras, LIDAR, ultrasons) pour obtenir une perception plus complète et fiable de l'environnement.
- Mettre en place un algorithme d'apprentissage en ligne permettant au robot d'adapter ses paramètres en temps réel selon les conditions environnementales changeantes.
- Étendre les capacités du robot à la détection et au suivi dynamique des piétons et autres usagers de la route afin d'améliorer la sécurité.

Ces améliorations permettront d'enrichir les fonctionnalités du robot, de le rendre plus autonome et efficace, et d'ouvrir la voie à des applications plus avancées dans le domaine de la robotique mobile et de la vision artificielle.

Problèmes rencontrés et solutions apportées

Position de la caméra : L'emplacement actuel limite la possibilité d'ajouter d'autres capteurs ou améliorations, ce qui restreint les évolutions du projet.

Contraintes de temps : Le délai imparti a limité la profondeur des tests et l'intégration de fonctionnalités avancées.

Assemblage matériel : Des difficultés ont été rencontrées lors du montage du châssis imprimé en 3D, notamment pour fixer correctement les moteurs et la caméra.

Erreurs de programmation : Plusieurs bugs initiaux liés au traitement d'image et au contrôle moteur ont été corrigés via des tests itératifs et des ajustements des paramètres du pipeline d'image.

Conclusion

Ce projet a constitué une véritable opportunité pour mettre en pratique l'ensemble des connaissances théoriques et techniques acquises durant notre cursus. Il nous a permis d'appliquer des concepts abstraits dans un cadre concret, favorisant ainsi une meilleure compréhension des enjeux liés à la conception, au développement, et à la gestion de projet.

Au-delà de l'aspect purement technique, cette expérience nous a aussi appris à organiser notre travail, à respecter les délais, et à collaborer efficacement au sein d'une équipe. Nous avons été confrontés à des difficultés réelles, ce qui nous a encouragés à développer notre capacité d'analyse, de résolution de problèmes, et d'adaptation face aux imprévus.

Par ailleurs, la réalisation de ce projet nous a permis d'utiliser des outils et des méthodes professionnelles, renforçant ainsi notre préparation à intégrer le monde du travail avec un savoir-faire opérationnel. Ce travail a également favorisé l'acquisition d'un sens accru de la rigueur et de la responsabilité.

En somme, ce projet a été une étape essentielle de notre formation, combinant théorie et pratique, et nous préparant efficacement aux défis futurs que nous rencontrerons dans notre carrière professionnelle.