Documentation générale du projet

Exterminateur de frelon asiatique

Table des matières

1. Présentation générale du projet

Ce projet consiste en la conception et la réalisation d'un dispositif embarqué de détection et de visée laser automatisée, reposant sur un Raspberry Pi 5, un système de caméras stéréoscopiques et un module laser piloté par galvanomètre. L'objectif principal est la détection visuelle de frelons asiatiques à l'aide d'un modèle d'intelligence artificielle optimisé, puis le pilotage précis d'un faisceau laser vers cette cible.

Le système combine des contraintes matérielles fortes (ressources limitées, sécurité laser, précision mécanique) et des contraintes logicielles (calibration, IA embarquée).

2. Dépôt GitHub du projet

Lien GitHub : https://github.com/janvier68/ExterminationFrelon/tree/main

Organisation du dépôt

Le dépôt GitHub regroupe l'ensemble des éléments logiciels nécessaires au fonctionnement du dispositif :

- Scripts de calibration des caméras
- Scripts de détection et de visée
- Code de pilotage du galvanomètre via DAC
- Modèles d'intelligence artificielle optimisés
- Fichiers de configuration et dépendances

Le dépôt permet ainsi de reproduire le projet, de comprendre l'architecture logicielle et d'adapter le système à d'autres cas d'usage.

4. Consignes de sécurité et utilisation du dispositif

4.1 Sécurité laser

ATTENTION – DANGER LASER

- Ne jamais se placer face au faisceau laser
- Ne jamais exposer les yeux ou la peau au laser, même brièvement
- Le laser peut provoquer des lésions irréversibles en cas d'exposition prolongée ou directe

4.2 Alimentation et commandes

Le dispositif comporte deux interrupteurs distincts :

1. **Interrupteur général** : situé sur la multiprise, il alimente l'ensemble du système
2. **Interrupteur laser** : permet d'activer ou de désactiver uniquement le module laser

Procédure d'extinction recommandée : - Éteindre le laser via son interrupteur dédié - Puis couper l'alimentation générale

Cette procédure permet d'éviter toute émission accidentelle du laser.

4.3 Lancement des scripts

Le système ne démarre pas automatiquement. Les scripts doivent être lancés manuellement depuis le terminal du Raspberry Pi.

Commandes principales :

- **Calibration des caméras**

  ```
  python setup.py
  ```

- **Système de détection et de visée**

```
python main.py
```

Toute modification de la position des caméras ou du système optique nécessite une **nouvelle calibration**.

5. Guide complet de calibration des caméras

5.1 Objectif de la calibration

La calibration permet de :

- Corriger les distorsions optiques des caméras
- Déterminer les paramètres intrinsèques et extrinsèques
- Garantir une précision maximale lors du calcul de visée du laser

Sans calibration valide, la précision du système est fortement dégradée.

5.2 Prérequis matériels

- Un damier de calibration **9 × 6 cases**
- Taille d'une case : **2,5 cm × 2,5 cm**
- Damier imprimé avec précision

Des modèles officiels sont disponibles sur le site d'OpenCV.

5.3 Interface de calibration

Lors de la calibration, un retour vidéo en direct est accessible via un serveur Flask :

- Adresse : `http://0.0.0.0:5000`

Ce retour permet de visualiser les images capturées et l'état de la calibration en temps réel.

5.4 Procédure de calibration

1. Lancer le script de calibration
2. Présenter le damier devant les caméras
3. Déplacer le damier :
    - Sur tous les axes (X, Y, Z)
    - En rotation
    - À différentes distances
4. Maintenir la manipulation pendant **2 à 3 minutes**

Une capture automatique est effectuée toutes les **0,2 secondes**.

5.5 Validation de la calibration

- Des croix vertes apparaissent sur les images
- Elles doivent être **approximativement centrées**

Si les croix sont trop décalées : la calibration est à recommencer.

5.6 Validité de la calibration

La calibration est **strictement dépendante** de la position des caméras.

Toute modification : - Déplacement - Rotation - Changement de support rend la calibration obsolète

6. Liste détaillée des composants matériels

6.1 Unité de calcul

- Raspberry Pi 5 (kit complet) ×1 (lien)
- Ventilateur actif pour Raspberry Pi 5 ×1 (lien)

6.2 Vision

- Caméra Sony IMX500 ×2 (lien)

6.3 Système laser

- Galvanomètre optique numérique 20K (380–700 nm) ×1 (lien)
- Module laser LASER TREE 450 nm – 20 W (puissance optique ~4 W) ×1 (lien)
- Pointeur laser (tests et alignement) ×1 (lien)

6.4 Électronique de commande

- DAC MCP4822 – 12 bits, 2 canaux ×1 (lien)
- Amplificateurs opérationnels TL082 ×2 (lien)
- Résistances
- Breadboard ×1
- Câbles de connexion (~30)

6.5 Environnement mécanique

- Boîtier électrique étanche (30 × 20 × 17 cm) ×1 (lien)

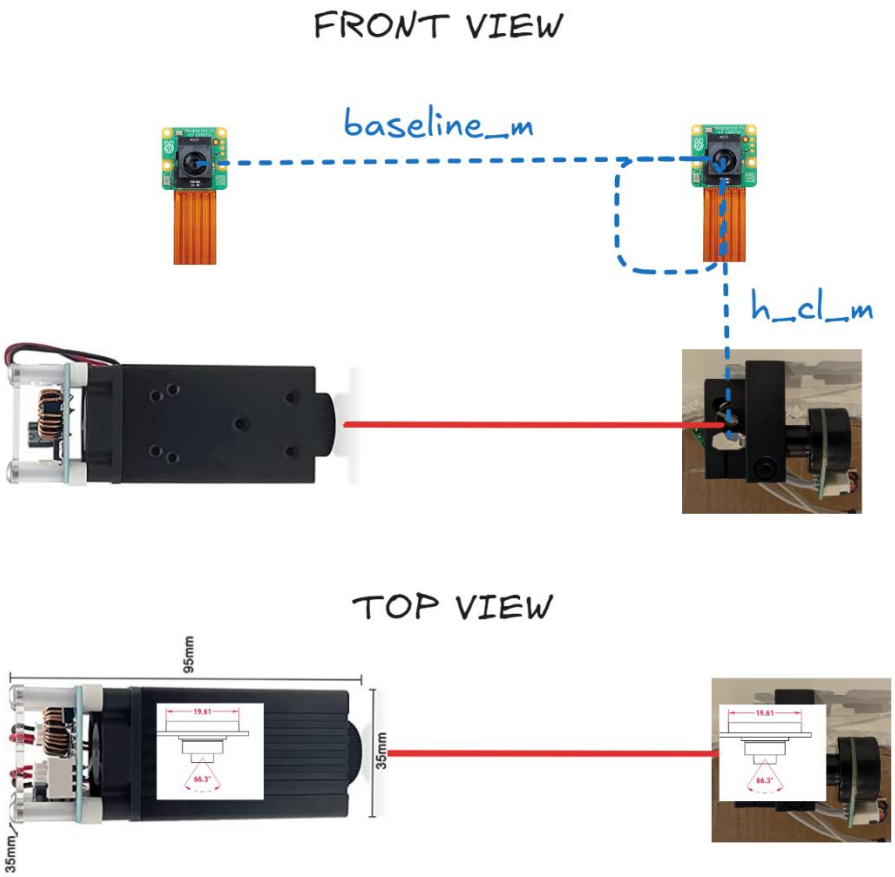
7. Reproduction du projet

7.1 Contraintes mécaniques et optiques

Pour un fonctionnement optimal :

- Les caméras et la sortie du galvanomètre doivent être **alignées en profondeur**
- Le galvanomètre doit être positionné **sous la caméras gauche**
- Le système de visée est basé sur la **caméra gauche**, qui doit être parfaitement alignée avec la sortie des miroirs
- Distance entre les centres optiques des deux caméras (baseline_m) : **10 cm**
- Les deux caméras doivent être strictement parallèles

Schéma :



Les variables *baseline_m* et *h_cl_m* sont les noms des variables dans le fichier de configuration qui peuvent être modifiées. Par défaut, nous les avons fixés tel que : *baseline_m* = 10cm et *h_cl_m* = 9.5cm.

7.2 Zone d'action du laser

- Angles d'usine : **-15° à +15°**
- Extension possible : jusqu'à **-30° à +30°**

Par manque de documentation constructeur sur le galvanomètre, les réglages d'usine ont été conservés afin de garantir la stabilité et la sécurité du système.

7.3 Contraintes liées aux performances embarquées

Le Raspberry Pi impose des limitations en termes de :

- Puissance de calcul
- Mémoire
- Débit de traitement vidéo

Pour cette raison :

- Un modèle d'IA léger et optimisé est utilisé et chargé sur les caméras
- La scène doit être simplifiée visuellement

L'ajout d'une **planche blanche** dans la zone d'action améliore fortement la détection des cibles.

7.4 Contraintes électriques

- Le câble d'alimentation du DAC est très court
- La multiprise doit être placée à proximité immédiate du dispositif

8. Installation logicielle

8.1 Système d'exploitation

- Utiliser **Raspberry Pi OS Legacy Lite**
- Installation via l'outil officiel Raspberry Pi Imager

8.2 Dépendances

- Installer l'ensemble des dépendances Python via le fichier `requirements.txt`

```
pip install -r requirements.txt
```

8.3 Remarques

- Le système est conçu pour fonctionner sans interface graphique mais
- Toute mise à jour majeure du système peut impacter la compatibilité des bibliothèques

9. Architecture logicielle et explications détaillées du fonctionnement

9.1 Organisation générale du code

Le projet repose sur une architecture logicielle modulaire afin de faciliter la compréhension, la maintenance et l'évolution du système. Des explications plus détaillées et commentées du code sont disponibles directement dans le dépôt GitHub du projet.

Le code est structuré autour de plusieurs éléments principaux :

- Un **dossier de configuration**, regroupant l'ensemble des paramètres modifiables
- Des **scripts dédiés à la calibration** des caméras
- Deux scripts principaux de lancement (main), correspondant à deux modes de fonctionnement

Cette organisation permet d'isoler les paramètres sensibles, de limiter les modifications dans le code principal et de simplifier les phases de test et de débogage.

9.2 Dossier de configuration

Le dossier de configuration contient un fichier centralisant toutes les variables ajustables du projet. On y retrouve notamment :

- Les paramètres liés aux caméras (résolution, index, paramètres de calibration)
- Les constantes géométriques du système (écartement des caméras, offsets, alignements)
- Les limites angulaires du galvanomètre
- Les paramètres de conversion entre angles, tensions et valeurs envoyées au DAC
- Les seuils et paramètres utilisés par le modèle d'intelligence artificielle

Cette centralisation des paramètres permet :

- Une modification rapide du comportement du système
- Une meilleure lisibilité globale du code
- Une réduction des erreurs lors des phases de réglage expérimental

9.3 Modes de fonctionnement : avec et sans serveur Flask

Le projet propose deux points d'entrée principaux, correspondant à deux scripts main distincts :

9.3.a Mode avec serveur Flask

Dans ce mode, un serveur Flask est lancé en local (localhost). Il permet :

- D'obtenir un retour vidéo en temps réel des caméras
- De visualiser les détections effectuées par le modèle IA (bounding boxes, centres)
- De faciliter le débogage et la mise au point du système

Ce mode est principalement utilisé lors :

- Du développement
- Des phases de test
- De la calibration et des réglages optiques

9.3.b Mode sans serveur Flask

Dans ce mode, aucun serveur web n'est lancé. Le système fonctionne entièrement en tâche de fond, ce qui permet :

- De réduire la charge processeur
- D'améliorer les performances globales
- D'utiliser le dispositif en conditions réelles

Remarque importante :
L'architecture logicielle globale est identique dans les deux modes. Seule la couche d'affichage et de retour visuel diffère.

9.4 Chargement et exécution du modèle d'intelligence artificielle

Le modèle d'intelligence artificielle est chargé et exécuté sur **les deux caméras**. Cette approche permet :

- De répartir la charge de calcul
- D'exploiter la vision stéréoscopique
- D'améliorer la robustesse et la précision de la détection

Le modèle est optimisé pour fonctionner sur une plateforme embarquée à ressources limitées, comme le Raspberry Pi. Il est entraîné pour distinguer au minimum :

- Les frelons
- Les abeilles

9.5 Traitement des détections et logique décisionnelle

Pour chaque image traitée par les caméras :

1. Le modèle IA détecte les objets présents et génère des **bounding boxes**
2. Le **centre de chaque bounding box** est calculé
3. Les centres issus des deux caméras sont récupérés

Une logique de filtrage est ensuite appliquée :

- **Si la cible détectée est une abeille** : aucune action n'est effectuée
- **Si la cible détectée est un frelon** : le processus de visée est déclenché

Cette étape permet d'intégrer une logique de sécurité écologique directement au cœur du système.

9.6 Calcul des angles de visée

À partir des centres des bounding boxes détectées par chacune des caméras :

- Deux angles de visée sont calculés
- Ces angles sont exprimés en degrés

Ils représentent les angles de déplacement nécessaires pour que le faisceau laser atteigne la cible dans l'espace. Les calculs prennent en compte :

- Les paramètres issus de la calibration des caméras
- Leur position relative
- Les caractéristiques géométriques du dispositif

9.7 Conversion des angles et pilotage du galvanomètre

Les angles calculés ne sont pas directement exploitables par le matériel. Ils sont donc convertis en valeurs utilisables par le système électronique :

1. Conversion des angles en valeurs analogiques
2. Adaptation aux plages de tension acceptées
3. Envoi des valeurs au **DAC MCP4822**

Le DAC génère alors les tensions nécessaires au pilotage du galvanomètre. Les miroirs du galvanomètre s'orientent en conséquence, ce qui entraîne le déplacement précis du faisceau laser vers la cible détectée.

9.8 Documentation complémentaire de la tourelle

Le fonctionnement détaillé de la tourelle (galvanomètre, miroirs, conversion tension-angle, limites mécaniques) est décrit plus précisément dans la documentation spécifique de la tourelle.

Il est fortement recommandé de s'y référer pour :

- Comprendre les contraintes mécaniques du système
- Ajuster correctement les paramètres de pilotage
- Éviter toute détérioration du matériel