Rapport de projet

Introduction

Ce projet vise à concevoir et déployer un système embarqué temps réel pour la reconnaissance de gestes, en exploitant l'analyse Rate Monotonic Analysis (RMA) pour la planification des tâches. Le système repose sur un **Raspberry Pi 4 Model B** comme unité de traitement principale et une caméra HD **Emeet C960** pour la capture d'images. Il s'appuie sur un traitement en parallèle des tâches critiques, notamment la capture et la classification des images, tout en garantissant le respect des contraintes de temps réel. Cette architecture permet une synchronisation précise entre les processus grâce à l'utilisation de semaphores et à l'application d'algorithmes de priorisation temps réel.

A. Architecture matérielle et logicielle

1. Matériel :
   - **Raspberry Pi 4 Model B :** Utilisé comme unité centrale de traitement, il gère l'exécution des tâches temps réel et l'hébergement des algorithmes d'analyse et de classification.
   - **Caméra HD Emeet C960 :** Source principale pour la capture vidéo, permettant de collecter des images pour la reconnaissance de gestes.
   - **Stockage local :** Les données d'images sont enregistrées au format PNG sur le stockage embarqué pour être utilisées dans le pipeline de classification.
2. Logiciel :
   a. Langages utilisés :
      - **C++ :** Dédié à la gestion des tâches en temps réel, notamment pour le contrôle de la caméra, la capture d'images, la synchronisation entre threads, et le respect des priorités des tâches critiques.
      - **Python :** Utilisé pour la conception, l'entraînement et l'exécution des modèles de classification basés sur des algorithmes de machine learning.

   b. Bibliothèques principales :
      - **OpenCV :** Implémenté pour gérer les opérations liées à la capture d'images et au prétraitement, comme la conversion en niveaux de gris.
      - **TensorFlow Lite :** Utilisé pour exécuter les modèles de classification optimisés sur un dispositif embarqué.
      - **Syslog :** Permet la journalisation des événements critiques pour le déboggage et le suivi des performances.

- **Semaphores binaires :** Implémentés pour assurer la synchronisation et la communication inter-thread.

B. Conception des modèles de classification

1. Développement et entraînement :
   - Les modèles sont entraînés en Python à l'aide de frameworks comme TensorFlow et scikit-learn, en utilisant un jeu de données annoté. Ce processus inclut des étapes de préparation des données, comme la normalisation et l'augmentation des données pour garantir la robustesse et la généralisation du modèle.
   - Les modèles développés sont testés localement pour garantir une performance optimale avant leur déploiement sur le système embarqué.
2. Optimisation et intégration :
   - Les modèles sont convertis en versions légères et optimisées à l'aide de TensorFlow Lite pour minimiser l'utilisation des ressources processeur et mémoire sur le Raspberry Pi.
   - Le pipeline de traitement comprend la capture des images en niveaux de gris, l'extraction des caractéristiques importantes, puis leur classification.

C. Planification et exécution temps réel

Les tâches principales du système, notamment la capture et la classification des images, sont orchestrées selon une logique de **Rate Monotonic Analysis (RMA)**. Cet algorithme garantit la priorité des tâches à courte période, assurant le respect des deadlines strictes imposées par l'environnement temps réel.

Les tâches sont exécutées en fonction de leur criticité :

- **Capture d'image** : Priorité maximale pour garantir un flux constant de données.
- **Classification** : Priorité intermédiaire, exécutée après la capture.
- **Séquenceur** : Priorité basse, utilisé pour synchroniser les autres tâches.

Le tableau représente l'analyse basée sur l'algorithme **Rate Monotonic Analysis (RMA)**.

Tâche	Période (T)	Fréquence	Temps d'exécution (C)	Utilité (%)	Utilité total	Limite d'utilisation	Priorité
T1	1000 ms	0.001	60 ms	6 %	77 %	77,97 %	Élevée
T2	1000 ms	0.001	700 ms	70 %			Moyenne
T3	1000 ms	0.001	1000 ms	1 %			Faible

D. Processus et logique système

1. Initialisation et configuration :
   - Le système démarre par l'initialisation du matériel, notamment la caméra HD **Emeet C960**, configurée pour fonctionner à une fréquence fixe de **30 FPS**. Cette étape garantit une synchronisation optimale avec les autres processus.
   - La priorité en temps réel est définie pour chaque tâche à l'aide de l'algorithme **SCHED_FIFO**, permettant de minimiser la latence et de respecter les contraintes de temps réel.
2. Capture et traitement d'images :
   - Un thread dédié est responsable de la capture d'images depuis la caméra. Chaque image est convertie en niveaux de gris à l'aide d'OpenCV pour réduire la complexité du traitement et est ensuite enregistrée localement au format PNG.
   - Ce processus est conçu pour être rapide et fiable, assurant la disponibilité des images pour les étapes ultérieures sans dépassement de délais.
3. Classification des images :
   - Une fois une image capturée, un autre thread se charge d'exécuter un script Python qui effectue la classification à l'aide d'un modèle de machine learning optimisé.
   - Les résultats de la classification sont renvoyés et enregistrés pour une utilisation ultérieure et affichés en temps réel.
4. Séquenceur des tâches :
   - Le séquenceur joue un rôle central dans l'orchestration des tâches. Il libère les sémaphores pour activer les threads de capture et de classification selon une séquence définie, garantissant que chaque tâche respecte ses délais spécifiques.
   - Les périodes et délais des tâches sont calculés en fonction de l'analyse RMA pour optimiser l'utilisation des ressources processeur tout en évitant les conflits.
5. Journalisation et supervision
   - Tous les événements importants, comme les erreurs ou les succès dans la capture et la classification, sont journalisés à l'aide de Syslog. Cela permet un suivi détaillé des performances et facilite le déboggage.
   - Le système est supervisé en continu, avec la possibilité de détecter et de répondre rapidement aux interruptions ou erreurs critiques.