Rapport Projet TER : Conception d'une surveillance vidéo médicale

SALVAN Corentin - MEHONG-SHIT-LI Matthieu - MARTINEZ Damien
BAIRY ONAPA Mikaël - LE LIDEC Tristan

17 avril 2021

Chapitre 1

Avant propos

L'objectif de cette unité d'enseignement est de nous préparé aux projets d'études de Master. Non content de consolider notre bagage scientifique, nous collaborons parmi un groupe de projet afin de sortir un produit fini, contexte qui pourrait se retrouver dans le milieu professionnelle également. Nous tenons avant tout à remercier Mr LAN SUN LUK Jean-Daniel pour l'aide précieuse apportée lors des points de blocages. Ce projet n'aurait pas aboutis dans le temps si nous n'avions consulter ouvrages et documents en ligne, sur lequel nous avons pu tirer informations, codes et librairies. Nous remerçions donc ces écrivains parfois anonymes.

Chapitre 2

Introduction

Chapitre 3

Description

3.1 Définition du projet

Le but de ce projet est de réaliser une surveillance vidéo avec deux composantes, la détection de chute et la reconnaissance faciale. Pour la détection de chute, lorsqu'une chute sera détecté une alarme sera activé afin d'alerter le personnel soignant. Depuis une interface graphique l'utilisateur pourra visualiser en temps réel le flux vidéo, choisir la caméra, mettre en pause le flux vidéo et désactiver l'alarme. Concernant la reconnaissance faciale, l'utilisateur connaitra le nom et le prénom de la personne si il est enregistré dans la base de données de l'établissement, ce qui permettra de détecter les éventuels intrus.

3.2 Cahier des charges

3.3 Support Logiciels/ Langage de Programmation/ Librairie

Plusieurs langages de programmation ont été discuté avant la réalisation du projet, dans un premier temps, nous avons opté pour le langage JAVA, notamment pour la programmation orienté objet qui facilite le travail en équipe lors du dévellopement. Finalement nous nous sommes decidé pour le langage Python, car il bénéfie d'une grande communauté, possède de nombreuses librairies, fonctionnalités que nous avions besoin. D'autre part c'est un langage que tous les membres de l'équipe savaient déjà programmer en Python.

Chapitre 4

Modélisation

4.1 Machine virtuelle OS Linux(Debian)

4.2 Machine virtuelle OS Raspberry Pi 2

4.3 Schéma de montage Hardware et description des composants

Sur la partie matériel nous avons choisi d'utiliser un Rasberry Pi, du fait de son aspect compact et de sa grande capacité de calcul. Concernant la partie logiciel, nous nous sommes tourné vers le langage de programmation Python car c'est un langage polyvalent avec une grande communauté et possède plusieurs librairies utiles à notre projet.

Chapitre 5

Réalisation Software-Hardware

5.1 Acquisition vidéo

Grâce à la librairie openCV, nous pouvons faire l'acquisition de tout type de caméra, en effet il suffit que la caméra soit branché à l'ordinateur pour qu'on l'utiliser. La lbrairie prend aussi en charge les caméras IP et le fichier vidéo.
A chaque appel de la fonction capture.read(), la librairie capture l'image du flux vidéo en entré, cette image va ensuite être utilisé pour faire les différents traitements et notamment la détection de chute.

5.2 Détection de chute

La détection se décompose en deux parties, la première consiste à extraire le sujet de la scène et la deuxième étudier son déplacement afin de savoir si oui ou non il y a bien eu une chute.

5.2.1 Extraction du sujet

Pour extraire le sujet deux méthodes ont été etudiés, la première methode consiste à utiliser une technique de classification. c'est une technique d'apprentissage supervisé. Il faut d'abord créer une base de données afin d'entrainer le modèle pour pouvoir reconnaitre le pattern souhaité. Cette méthode est coûteuse en ressource, mais peut être très efficace si la base de données est grande. Nous avons

5.2.2 Critère de détection de chute

5.3 Reconnaissance faciale

5.4 Interface Graphique

5.5 Commande du matériel

Chapitre 6

Bilan du projet

6.1 Difficulté du projet

6.2 Analyse du résultat confrontée au cahier des charges

6.3 Piste d'amélioration

6.3.1 Qualité du matériel

6.3.2 Résolution du système

6.3.3 Performance du code et du temps de calcul

6.4 Timeline (Diagramme de Grantt)

Bibliographie

[1] Ben Nuttall Revision, Edit on GitHub, *Librairie : GPIO zero.*
Consulté le 03/04/2021

`url = https://gpiozero.readthedocs.io/en/stable/index.html`

[2] Raspberry Pi FR, *Utiliser un lecteur RFID avec Raspberry.*
Consulté le 03/04/2021

`url = https://raspberry-pi.fr/rfid-raspberry-pi/`

Annexes

.1 Code principal du système

.2 Code librairie : RFID.py