Rapport Projet IMA3/4

Projet P17:
Aide au déplacement pour enfant

Ming CHEN - Jing HUA - Emilie RAOUTO
4ème année IMA - Polytech Lille

Tuteur école:
Alexandre Boé
Marion Binninger
Xavier Redon
Thomas Vantroys

Année 2019-2020

Sommaire

Introduction

Ce présent rapport a pour but de présenter les différents résultats obtenus lors du projet IMA3/4 de fin d'étude comment trouver une méthode pour aider au déplacement pour un enfant. Ce projet a été proposé par Mme Marion Binninger, ergothérapeute à l'Institut d'Éducation Motrice (IEM) de Lille pour aider un enfant ayant quelques troubles attentionnels et des réactions impulsives à être autonome dans ses déplacements. Nous avions eu le plaisir de rencontrer ce jeune garçon pour avoir plus de détail et donc répondre spécifiquement à ses besoins. Ainsi, le but est de réaliser un dispositif permettant à cet enfant et peut-être à d'autres atteint de troubles déficit de l'attention avec ou sans hyperactivité (TDAH) de pouvoir circuler librement et en toute sécurité.

Le sujet en question concerne la conception d'un prototype qui peut aider l'enfant à faire des déplacement en toute sécurité. Afin de pouvoir adapter le prototype à nos besoins, il faudra concevoir un algorithme considérant tous les cas possible dans la rue pour assurer la sécurité.

Organisé en trois parties, le projet sera tout replacé dans son contexte, en précisant les attentes et quel devrait être le résultat attendu, pour ensuite s'orienter vers les solutions envisagées, celles retenues ou non, et leur mise en place. De cette manière, ce rapport s'achève sur les résultats obtenus ,les limites et les perspectives.

I. Présentation du projet

1. Cahier de charge

Contexte

C'est le projet de IMA3/4. Le projet consiste à réaliser un dispositif permettant d'assurer la sécurité d'un enfant atteint de troubles de troubles attentionnels. Le dispositif doit permettre à l'enfant de se sentir en sécurité et de traverser la route en toute confiance.

Pendant les semestres précédents, nous avons réalisé un premier prototype avec un capteur de type ultrason et ce prototype peut détecter un objet en approche et puis faire une alerte à l'utilisateur. Ensuite nous avons aussi choisi les matériaux pour qu'on peut faire le prototype final en ce semestre.

Objectif

L'objectif est que ce dispositif réponde aux besoins de l'enfant qu'il puisse être autonome en traversant la route tant pour la sortie d'école que pour des promenades au parc. Il doit donc être avertit en cas de potentiels dangers, informé de la démarche à suivre en cas de danger.

Fonctionnement

1. Le dispositif doit être agréable à porter

Nous avons à faire à un enfant et si le dispositif est une gêne pour lui, cela occasionnera un autre trouble d'attention chez lui. Il doit donc être assez léger pour lui permettre de sentir à l'aise et en confiance.

2. Le dispositif doit servir de guide

il y aura un système audio qui permettra de transmettre les informations à l'enfant. Et Il est muni d'alertes telles que des vibreurs pour ramener son attention et prévenir du danger.

3. L'enfant ne sentira aucune vibration de son bras gauche

Dans ce cas nous devons prendre en compte son handicapée du bras gauche. Il faudra aussi faire attention à la tonalités et prendre en compte les perturbations déjà liées à l'environnement.

4.le dispositif doit s'assurer des conditions pour traverser la route

Pour faire des alertes, il faut préciser les conditions pour les activer telles que les feux de signalisation ou l'approche d'un objet en grande vitesse.

Ressources

Sous la direction de nos tuteurs Alexandre Boé, Marion Binninger, Xavier Redon et Thomas Vantroys, ce projet est réalisé par nous. Les matériaux informatique, les composants et le support technique requis dans ce projet sont fournis par l'école.

2. Choix du matériel

Webcam

Pour capter les feux de signalisation dans la route.

RPLidar A1M8

Pour scanner l'environnement et puis chercher si il y un objet(surtout une voiture) en approche avec une grande vitesse.

Raspberry Pi 3

Pour notre prototype peut traiter des images ou vidéos captés par un webcam et lire les mesures dans le lidar, il faut choisir un processeur tellement fort comme une raspberry pi.

II. Présentation du Travail

1. Partie Configuration

La première chose on doit faire avant concevoir notre algorithme est configurer la raspberry pi 3 pour que nous pouvons manipuler sur la raspberry en même temps. Donc on implémente le système Raspbian avec un desktop et puis on connecte par une liaison série comme on avait fait en cours Réseau. Ensuite on utiliser la commande "sudo raspi-config" pour autoriser la connection SSH et VNC.

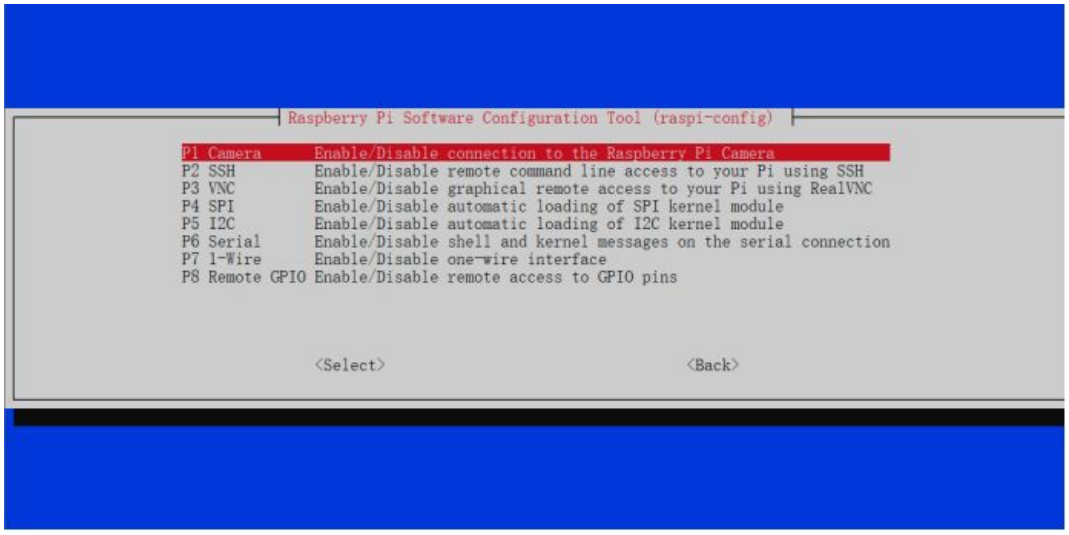

Fig.1 - Fenêtre des interfaces

Puis avec la commande "vim /etc/wpa_supplicant/wpa_supplicant.conf" pour configurer sa connexion dans un certain réseau.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

network={
        ssid="Moi"
        psk="aaabbbaaa"
        priority=5
}

network={
        ssid="Shajing"
        psk="12345678910aaa"
        priority=6
}
```

Fig.2 - Configuration du réseau

Après cela on peut réaliser une connexion SSH ou VNC entre la raspberry et l'ordinateur sous le même réseau.

2. Partie RPLidar

2.1 Un test de code sur GitHub

Avant faire l'algorithme pour scanner l'environnement et puis trouver un objet, on veut réaliser un premier essai avec l'exemple donné sur le site GitHub https://github.com/Slamtec/rplidar_ros , pour vérifier si le lidar fonctionne bien. On appilique l'application "ultra_simpl" et on obtient le résultat suivant:

```
theta: 218.83 Dist: 00265.00 Q: 47
theta: 220.22 Dist: 00271.00 Q: 47
theta: 220.23 Dist: 00277.00 Q: 47
theta: 221.03 Dist: 00000.00 Q: 0
theta: 221.28 Dist: 00271.00 Q: 47
theta: 221.67 Dist: 00000.00 Q: 0
theta: 221.75 Dist: 00271.00 Q: 47
theta: 222.80 Dist: 00273.00 Q: 47
theta: 222.97 Dist: 00000.00 Q: 0
```

Fig.3 - Une partie d'affichage de résultat

On voit ici il y a 3 valeur à afficher qui sont: l'angle en degré, la distance en mm et la qualité de ce scan. En effet il y a autre valeur qui présente le début de chaque scan qui n'est pas dans la capture. Et dans chaque tour(360 degrés), les valeurs sont stockées dans un tableau de taille 8192 pour préciser l'environnement en différents angles.

On voit aussi si la qualité est 0, cela veut dire on n'a pas réussi à récupérer la valeur de cet angle. Donc dans la suite de la conception, il faut considérer ce cas. Et en regardant le datasheet on trouve les limites pour le Lidar:

1. Faut mieux commencer récupérer les mesures dans 2 minutes depuis Lidar démarre.
2. Le température ne peut pas passer 40° et ne peut pas aller plus petit que -10°, sinon les résultats ne sont pas précis.

2.2 Les conceptions de l'algorithme pour trouver une voiture

Après réussir tester l'application donnée sur GitHub, on veut concevoir notre algorithme pour trouver une voiture dans l'environnement, donc on prend un certain angle pour simuler le cas où la voiture approche dans cet angle.

2.2.1 Simulation d'une voiture approche dans un certain angle

Dans un premier temps on veut réaliser le cas plus simple: une voiture approche directement vers l'utilisateur.

Dans ce cas on sait dans le tableau 'nodes', la distance d'un angle va diminuer successivement si la voiture est en approche. Et avec un délai d'un certain temps, et les distances correspondantes à ces 2 temps, on peut calculer la vitesse de cette voiture et si cette vitesse est positive cela veut dire la voiture est en approche.

Donc on peut proposer une limite de la vitesse qui est 20km/h dans la ville, et on le représente en 5500mm/s, si la vitesse arrive à cette limite on lance l'alerte d'un vibreur. En posant la voiture approche dans l'angle 0 degré on réalise cette simulation simple.

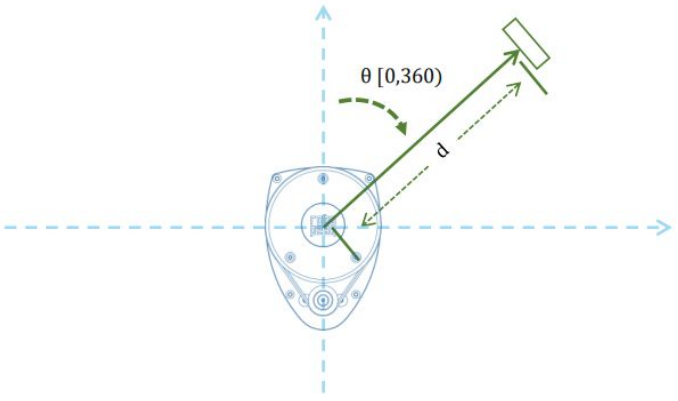

Fig.4 - Le repère de l'angle du Lidar

Puis on veut améliorer la performance à la base de cet essai, et il y 2 choses on doit réfléchir dessous telles que si la voiture approche non

directement, et si la voiture approche avec différente vitesse et différentes distance de l'utilisateur, comment établir plusieurs niveaux de l'alerte.

Avec ce cas, puis on pose différents niveaux de l'alerte avec 3 vibreurs. Comme dans l'annexe **[1]** on utilise 3 variables correspondants à une alerte normale(la voiture arrive à la vitesse limite), une alerte d'être trop proche que la voiture et une alerte de détecter une voiture avec une grande vitesse(la vitesse est beaucoup grande que la vitesse limite). Donc dans la suite on propose une autre méthode pour détecter une voiture approche non directement.

Et on ajoute aussi un fonction d'affichage pour montrer si on a arrivé à la limite de la vitesse.

```
theta1: 0.25 Dist1: 00407.00   theta2: 0.80 Dist2: 00393.00
Vitesse; 0.000000  mm/s si on arrive a la limite? oui cpt: 1 alerte: 0
theta1: 0.81 Dist1: 00320.00   theta2: 0.36 Dist2: 00000.00
Vitesse; 0.000000  mm/s si on arrive a la limite? oui cpt: 2 alerte: 0
theta1: 0.02 Dist1: 00130.00   theta2: 0.00 Dist2: 00000.00
Vitesse; 0.000000  mm/s si on arrive a la limite? oui cpt: 3 alerte: 1
theta1: 0.00 Dist1: 00000.00   theta2: 0.25 Dist2: 00716.00
Vitesse; 0.000000  mm/s si on arrive a la limite? non cpt: 3 alerte: 1
```

Fig.5 - L'affichage de résultat de la vitesse et l'angle correspondant d'un objet

2.2.2 Simulation d'une voiture approche non directement vers l'utilisateur

Dans ce cas je propose un algorithme non précis mais fonctionne pour un cas spécial. Si l'utilisateur et la voiture vont se rencontrer dans un certain temps, cela veut dire pour lidar, le point qui représente la voiture, ses paramètres(surtout angle et la distance) sont comme suivants:

Si le lidar est sur le bras droite qui est notre cas, selon le repère au-dessous, on pose que l'angle 0° est la direction vers laquelle l'utilisateur

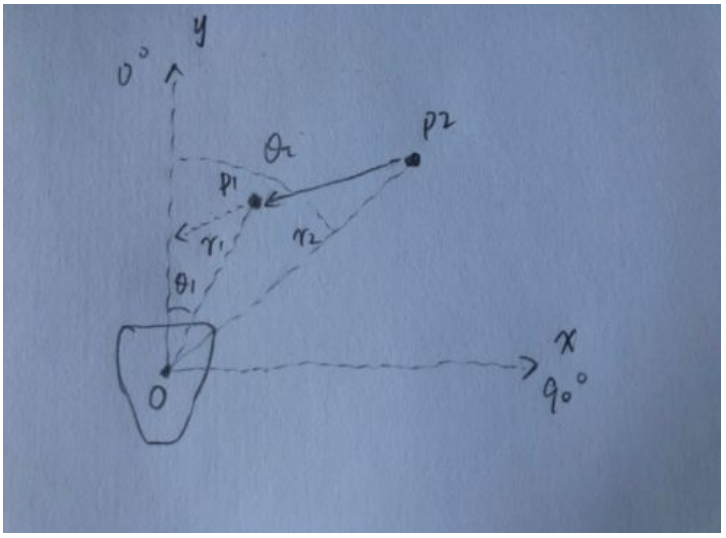

Fig.6 - Le repère du lidar

avance, donc un moment si on détecte une distance D1 qui est la distance entre la voiture et l'utilisateur dans l'angle 60°, ensuite dans un certain temps t1 on mesure la distance D2 dans l'angle 45°, on est sûr que D2 est plus petit que D1 si la voiture approche. Puis dans un certain temps t2, on mesure la distance D3 dans l'angle 30° qui serait la distance plus petit car la voiture est en approche et si ce n'est pas cas il n'y pas de danger.

Donc approximativement on peut dire il y a un danger potentiel si les 3 mesures correspondent le cas précédant. Après on implémente cet algorithme sur la raspberry.

Avec plusieurs essais de condition pour stocker la valeur de ces angles, on pose les conditions dans l'annexe **[2]** pour alerter correctement:
1. La qualité de scanner cette distance n'est pas nulle, car si cette valeur est nulle cela veut dire on n'a pas réussi à prendre cette mesure et la distance est 0, donc si on prend cette mesure quand même on va recevoir une distance qui est 0.

2. La distance est supérieur que 200mm, car si c'est inférieur que 200 mm=0.2m, il n'y a pas trop de sens(trop proche) et j'ai aussi eu une erreur qui est un moment cette valeur est négative(pour éviter cette erreur). Mais après ajouter cela, la distance = 0 quand même qui n'est pas normal, car si cette distance est inférieur à 200 mm, on l'abandonne, sûrement la valeur minimales des D1, D2, D3 sont 200mm. (200mm en effet est limité par la taille de ma chambre pour un test).

3. Cette distance est inférieur que la distance qui est stockée dans la variable D1 (initialisée par 12000 en mm car la distance maximale est 12m)

Même si on a réussi à alerter plusieurs fois avec cette méthode, elle a un grand problème de précision car elle prend seulement 3 points pour juger les conditions de l'alerte, donc dans la suite, on propose une méthode plus générale.

2.2.3 Simulation d'une voiture approche dans n'importe quel angle

Sous la direction de nos tuteurs, on trouve une autre méthode pour trouver une voiture qui approche dans n'importe quel angle qui est mieux et base du calcul mathématique.

Avant présenter la méthode en détaille, on veut définir 2 types de points, le point mobile et le point immobile. Si on calcul la vitesse de tous les points de 360°, on trouve il y a des points ont une distance changent de

temps en temps, par contre, les points qui ont une distance fixée ou changent pas beaucoup.

Donc on a 4 étapes pour avoir la vitesse et le temps avant se rencontrer:

1. Trouver des points de gravités des voitures
2. Trouver la représentation paramétrique de la droite en supposant que la voiture va droit en coordonnées polaires
3. Trouver le point le plus proche que l'utilisateur sur cette droite
4. Calculer le temps d'arriver à ce point proche et la distance entre ce point et l'utilisateur

Donc on a besoin de 2 points pour faire une droite, on garde le même repère dans la méthode précédente, et on pose 2 points P1(X1, Y1) et P2(X2, Y2), et on peut présenter P1 avec les mesures obtenus comme la figure au-dessus:

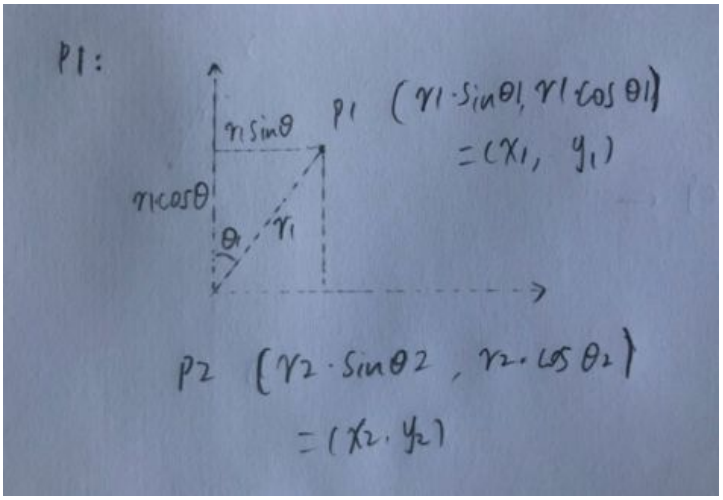

Fig.7 - La représentation du point P1 dans le repère

Avec figure 6 et 7, on peut calculer la vitesse de la voiture par une équation:

$$V = \frac{\sqrt{(y2-y1)^2 + (x2-x1)^2}}{t}$$, et puis on peut trouver la pente:

$$pente = \frac{y2-y1}{x2-x1}$$, puis on pose la droite sous forme:

$$y = \frac{y2-y1}{x2-x1} \cdot x + k$$, et donc c'est facile de calculer la valeur de k:

$$k = \frac{y2 \cdot x2 - y2 \cdot x1 - y2 \cdot x2 + x2 \cdot y1}{x2-x1} = \frac{x2 \cdot y1 - x1 \cdot y2}{x2-x1}$$,

donc la droite finales est:

$$y = \frac{y2 - y1}{x2 - x1} \cdot x + \frac{x2 \cdot y1 - x1 \cdot y2}{x2 - x1}$$, et ici on peut trouver le point le plus proche, on applique une méthode d'approximation:

On pose que la vitesse de l'utilisateur est négligeable par rapport à la voiture. Et on sait le point le plus proche est un point sur la droite.

Maintenant on sait que le point o qui a les coordonnées (0,0) est la position de l'utilisateur et supposant qu'on a 2 points de gravités avec la méthode présentée au-dessus. Aussi il y a une limite pour ce point, c'est ce point il faut se situe dans la zone où les coordonnées x et y sont tous positifs, car sinon la voiture ne va jamais se rencontrer avec l'utilisateur. Donc, on peut trouver le point le plus proche à le point o est le point sur l'axe y, aussi ce point est sur la droite en même temps:

$$(0, \frac{x2 \cdot y1 - x1 \cdot y2}{x2 - x1})$$

Puis on peut trouver la distance entre ce point et le point P1, et aussi le temps pour passer au point le plus proche:

$$dis = \sqrt{x1^2 + (\frac{y1 \cdot x2 - x1 \cdot y1 - x2 \cdot y1 + x1 \cdot y2}{x2 - x1})^2} \quad = \sqrt{x1^2 + (\frac{x1 \cdot y2 - x1 \cdot y1}{x2 - x1})^2}$$

avec $t = \frac{dis}{v}$.

Après avoir ces valeurs, on peut alors faire des alertes selon ces valeur. Si dis est petit et t est petite c'est très grave et il faut avertir vite, si dis est petite et t plus grand il faut avertir moins fort,et si dis est au-delà de 5 mètres ce n'est pas grave.

On voit le code principal dans l'annexe**[3]** avec le bibliothèque qu'on a utilisé voit dans l'annexe **[4]**,et pour certains parties on voit dans la prochaine partie "Les tests individuels".

Cependant après ajouter les fonction pour faire les racines on a eu un problème de compilation comme la figure au-dessous.

Fig.8 - L'erreur pendant la compilation

2.2.4 Les tests individuels

Comme dans la simulation précédente, on utilise des fonction mathématique, la fonction gettimeofday etc. Donc dans cette partie on montre comment on teste ces fonctions individuellement pour assurer qu'elles sont correctes dans notre code final pour bien implémenter.

Tout d'abord pour vérifier qu'on a trouver le point de gravité on ajoute une fonction d'affichage si on trouve le point de gravité, et après je mets mon main dans un certain angle pour vérifier si il y un changement de distance, est-ce qu'il peut afficher ce point avec son angle et sa distance. C'est très important car si n'a pas 2 points de gravité, on ne peut plus calculer la droite ni juger le niveau de danger. On peut voir le résultat dans la figure suivante.

Fig.9 - L'affichage de point de gravité

Ensuite car on a besoin d'un compteur de temps, donc on choisit gettimeofday, et aussi pour tester la fonction sinus on a fait un test comme la figure au-dessous.

```
    struct timeval debut,fin;
    unsigned long difference;
    gettimeofday(&debut,NULL);
    delay(1000);
    int t=45;
  printf("%f\n",sin(M_PI/2));
  printf("%f\n",sin(45*PI/180.0f));
  gettimeofday(&fin,NULL);
  difference=1000*(fin.tv_sec-debut.tv_sec)+fin.tv_usec-debut.tv_usec;
  printf("la difference est %ld\n",difference);
  return(0);
```

Fig.10 - Le test du compteur du temps et la fonction sinus

3. Partie WebCam

Pour les enfants ayant des problèmes de concentration, traverser la route est sans aucun doute une chose très dangereuse. Pour ma part, je vais utiliser le Raspberry Pi et la webcam pour concevoir un appareil qui détecte les feux rouges. Cet appareil peut rapidement rappeler aux enfants la présence de feux rouges devant les enfants lorsqu'ils traversent la route, afin que les enfants puissent traverser la route en toute sécurité.

Reconnaître la lumière rouge, l'essentiel réside dans la reconnaissance des graphiques et des couleurs, après avoir vérifié les informations sur Internet. J'ai trouvé une bibliothèque de vision par ordinateur puissante, qui peut être utilisée pour développer des programmes de traitement d'image en temps réel, de vision par

ordinateur et de reconnaissance de formes. Pour mes besoins, c'est un outil puissant.

3.1 Bibliothèque OpenCV

Tout d'abord, j'ai installé la bibliothèque openCV complète (voir l'annexe pour les commandes d'installation)**[5]**

Ensuite, je comprends brièvement les principaux fichiers d'en-tête et les fonctions principales de la bibliothèque openCV, afin qu'ils puissent être utilisés correctement dans le programme.

3.2 Concevoir le programme.

3.2.1 Le fichier d'en-tête initial et la définition de deux espaces de noms cvstd

```
#include <iostream>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include <wiringPi.h>
#include <cstdlib>

using namespace cv;
using namespace std;
```

3.2.2 Allumez la webcam du Raspberry Pi

Tout d'abord, j'ai choisi la webcam avec interface USB, elle est très légère et facile à transporter.Afin de détecter la position de la balle en temps réel, nous devons d'abord allumer la webcam du Raspberry Pi.

```
VideoCapture cap(0); //capture the video from web cam
if ( !cap.isOpened() ) // if not success, exit program
{
cout << "Cannot open the cam" << endl;
return -1;
}
```

Modifiez / etc / modules, ajoutez une ligne: bcm2835-v412
Comme suit:
snd-bcm2835

bcm2835-v4l2

3.2.3 Prenez des photos en temps réel depuis la webcam

```
Mat imgOriginal;
 bool bSuccess = cap.read(imgOriginal); // read a new frame from video
 if (!bSuccess) //if not success, break loop
 {
 cout << "Cannot read a frame from video stream" << endl;
 }
```

Ici, une image de type Mat imgOriginal est déclarée, puis il y a une variable booléenne nommée bSuccess, qui est utilisée pour juger si l'acquisition d'image est réussie.

3.2.4 Reconnaître le rouge

Il est difficile de trouver la gamme exacte de rouge en contrôlant le canal RVB, nous avons donc choisi l'espace HSV. HSV (Hue, Saturation, Value) est un espace colorimétrique créé sur la base des caractéristiques intuitives de la couleur,
H: Teinte S: Saturation V: Légèreté.

	Noir	Gris	Blanc	Rouge		Orange	Jaune	Vert	Cyan	Bleu	Pourpre
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34;	77	99	124	155
smin	0	0	0	43		43	43	43	43	43	43
smax	255	43	30	255		255	255	255	255	255	255
vmin	0	46	221	46		46	46	46	46	46	46
vmax	46	220	255	255		255	255	255	255	255	255

Fig.11-Table HSV en différentes couleurs

Ceci est une gamme floue, une partie du violet est classée comme rouge.

```
Mat imgHSV;
vector<Mat> hsvSplit;
cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV);
//Convert the captured frame from BGR to HSV
split(imgHSV, hsvSplit);
equalizeHist(hsvSplit[2],hsvSplit[2]);
merge(hsvSplit,imgHSV);
```

imgHSV est l'image obtenue en convertissant l'espace RVB d'imgOriginal en espace HSV, puis en séparant les trois canaux de HSV. La fonction cvEqualizeHist est utilisée pour égaliser l'histogramme, qui peut transformer une image plus claire en une image plus sombre (c'est-à-dire améliorer la luminosité et le contraste de l'image).

cvMerge: fusionnez plusieurs images monocanal en une image multicanal, afin d'avoir une image spatiale HSV. Et en ajustant la valeur de HSV, vous pouvez essentiellement filtrer toutes les zones non rouges.

3.2.5 Binarisation

La binarisation consiste à convertir une image couleur en noir et blanc pur. CvCvtColor peut convertir l'image en image en niveaux de gris, mais pas en image binaire. CvThreshold est une fonction de binarisation. La partie rouge d'origine de l'image obtenue est blanche et le reste est noir.

3.2.6 Tracez un cercle sur la zone rouge

Utilisez d'abord la transformation de cercle de Hough pour détecter le cercle: la fonction cvHougeCircles a plusieurs paramètres: la première est l'image d'entrée, la seconde est la mémoire, la troisième est la méthode et l'algorithme et la quatrième est la résolution: lorsqu'elle est définie sur 1, La résolution est la même, la résolution 2 est normale et la cinquième est la distance minimale entre deux cercles que l'algorithme peut clairement distinguer. Les deux suivants sont des seuils, suivis des rayons maximum et minimum du cercle qui peuvent être trouvés. cvPoint est un type de données dans OpenCV et représente un point en deux dimensions. centre est le centre du cercle. En fait, dessiner le centre d'un cercle équivaut à dessiner un cercle. La fonction cvCircle est utilisée, mais lorsque le rayon du cercle dessiné est 0, il devient le centre du cercle.

```
vector<Vec3f> circles;
HoughCircles( imgThresholded, circles, CV_HOUGH_GRADIENT,1.5, 10, 200, 100,
0, 0 );
for( size_t i = 0; i < circles.size(); i++ )
{
Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
int radius = cvRound(circles[i][2]);
circle( imgThresholded, center, 3, Scalar(0,255,0), -1, 8, 0 );
circle( imgThresholded, center, radius, Scalar(155,50,255), 3, 8, 0 );
}
```

3.2.7 Interface de contrôle

En raison des différentes conditions d'éclairage à différents moments, il est préférable d'avoir une interface de contrôle pour ajuster le HSV en temps réel afin que l'effet d'identification du rouge soit le meilleur. OpenCV a une interface graphique dédiée pour écrire des fonctions, ce qui est également très pratique pour écrire.

3.2.8 Démarrez le vibreur

Afin de rappeler l'enfant plus rapidement, j'ai activé le vibreur à chaque fois que le programme détectait une lumière rouge, afin qu'il puisse se concentrer et éviter le danger.**[6]**

3.3 Résultats du programme

Enfin, le résultat de ce programme est le suivant**:**

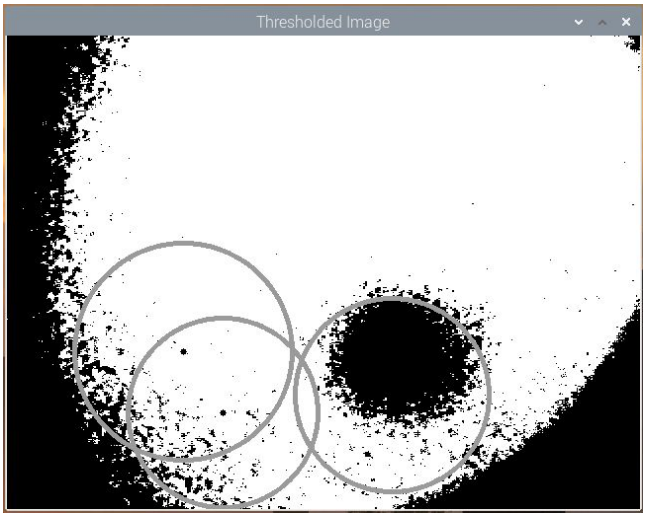

Fig.12 - le résultat du webcam

4. Partie Actionneurs

Concernant la partie actionneurs, nous utilisons 4 moteurs vibreurs et un haut parleur pour le son. Dans le but d'amplifier le son de sortie, nous avons opté pour un amplificateur constitué d'un LM386. En effet, le LM386 est un amplificateur audio à faible voltage qui est fréquemment utilisé dans les appareils musicaux à piles comme les radios, les guitares, les jouets, etc. La plage de gain est de 20 à 200 . Ici nous avons augmenté le gain à 200 en utilisant une résistance et un condensateur entre les PIN 1 et 8.

Pour rendre l'alerte du danger proportionnelle aux vibrations, nous activons l'un des vibreurs ou les 4 en même temps pour bien signifier le rapprochement du danger vers l'enfant. Pour le son, nous avons la fonction "omxplayer" qui appliqué directement en ligne de commande avec comme paramètre "leFichierSon.mp3" fait joué le son jusqu'à la fin. Pour le code alors nous faisons deux scripts shell, tel que l'un prend en compte l'audio de l'avertissement et le second l'audio du danger. Dans

un programme en C, nous utilisons la fonction "execl" qui prend en paramètre: le chemin d'accès prog, prog, les arguments et NULL.

```
int pid;
pid=fork();
if(pid==0){
execl("/home/IMA4/projet/programmes/code1", " ", "code1",NULL)
_exit(0) }
else{
wait();}
```

5. Résultat obtenu

Comme nous avons réalisé certains tests séparément, et à la fin on veut faire une liaison entre différentes parties, à cause du coronavirus on ne peut que faire la liaison entre le lidar et le webcam.

Fig. 13-le test avec le lidar et le webcam

Cette fois on envoie un photo avec une balle rouge pour simuler le feu rouge à un moment au hasard, et puis on implémente la première méthode de la simulation d'une voiture en approche, pour tester le fonctionnement, on trouve que soit une voiture en approche avec une vitesse supérieur que la consigne ou le moment où on reçoit un image d'une balle rouge, les vibreurs peuvent être activé pendant certain temps, selon le niveau de la vitesse d'un objet qui est une boîte noir dans notre test pour simuler une voiture.

6. Limites et Perspectives

6.1 Limites

Même si on a réussi un essai avec lidar et webcam, il y encore des limites de notre prototype.

1. Le temps de réponse pour le webcam est un peu long qui sera danger dans la vie.
2. Nous n’avons pas considéré le cas où il y a plusieurs voitures.
3. Nous n’avons pas réussi la fin de troisième méthode.
4. Le prototype est lourd à porter pour un enfant, à cause du lidar, la raspberry et la batterie.

6.2 Perspectives

Correspondant à nos limites, nous pouvons chercher un lidar plus léger mais moins lourd, pareil pour la raspberry et la batterie. Cependant, pour bien précis la situation de l'environnement dans la roue, il faut prendre assez de mesures et donc on a besoin un processeur tellement fort pour qu’il peut faire ces calculs, aussi à cause de sa performance, il a besoin plus d’énergie en même temps, donc si on change une batterie moins lourd et avoir moins d’énergie, il faut bien charger la batterie chaque fois qui sera non pratique.

Donc pour résoudre ces problème, il faut choisir un capteur, un processeur moins lourd mais avec même performance, aussi pour une batterie moins lourd mais suffisant d'alimenter le prototype. Aussi le coût sera un problème si on veut ce prototype fait face à ces type de clients comme cet enfant Souleyman.

CONCLUSION

Ce semestre nous a permi de faire une très grande partie du projet, de répondre au cahier de charge pour satisfaire au mieux les besoins de l'enfant, malgré certaines limites. Nous avons trouvé aussi le projet vraiment intéressant parce que c'est une vrai application à un enfant, c'est vraiment pour résoudre un problème dans la vie. Sauf cela, il nous permet aussi de mettre en pratique les connaissances acquises en cours de notre filière comme la raspberry qu'on a vu en cours de réseau, le langage C++ qu'on a vu en cours de OCSA etc… Et la direction des tuteurs nous permet d'avoir une plus grande perspective quand on traite un problème, comment réfléchir comme un ingénieur, par exemple dans notre projet nous devons choisir une solution pour avoir une forte performance et être léger en même temps.

C'est le projet le plus long qu'on a eu, donc nous avons également appris comment organiser un projet de long temps, comment répartir le travaux dans un groupe, aussi la capacité d'auto-apprentissage.

Après tout, nous voulons bien évidemment nommer nos encadrants dans ce travail, Monsieur Alexandre Boé, Madame Marion Binninger, Monsieur Xavier Redon et Monsieur Thomas Vantroys. Leurs idées, indications techniques et suggestions sont vraiment utiles pour le projet.

Annexe

[1] Le code avec différents niveaux de l'alerte

```
printf("si on arrive a la limite? %s ",( (int)Vitesse > 5000)?"oui":"non");
//if(sec_distance!=0 && pre_distance!=0 && Vitesse>1000)
if(Vitesse > double(5000))
{
        cpt = cpt+1;
}
else if(sec_distance!=0 && pre_distance!=0 && Vitesse<1000)
{
        cpt=0;
}
if(cpt>2)
{
        alerte=1;
}
if(Vitesse > 25000)
{
        alerte_vite=1;
}
if(pre_distance < 200)
{
        alerte_proche=1;
}
printf("cpt: %d alerte: %d \n", cpt,alerte);
```

[2] Le code avec différents niveaux de l'alerte

```
if(60<(nodes[pos].angle_q6_checkbit >> RPLIDAR_RESP_MEASUREMENT_ANGLE_SHIFT)/64.0f &&  (nodes[pos].angle_q6_checkbit >> RPLIDAR_RESP_MEASUREMENT_ANGLE_SHIFT)/64.0f<61) {
    if( nodes[pos].sync_quality >> RPLIDAR_RESP_MEASUREMENT_QUALITY_SHIFT!=0 && nodes[pos].distance_q2/4.0f>1000 &&nodes[pos].distance_q2/4.0f<distance_30)
    {
        distance_60=nodes[pos].distance_q2/4.0f;
    }
```

[3] Le code principal

```
198     // fetech result and print it out...
199     //op_frequence = drv->getFrequency();
200     while (1) {
201     presence_voiture=0;
202         rplidar_response_measurement_node_t nodes[8192];  //on prend le max point pour la precision
203     rplidar_response_measurement_node_t new_nodes[8192];  //on prend un deuxieme mesure pour trouver des points de gravites
204         size_t   count = _countof(nodes);
205
206     op_result = drv->grabScanData(nodes, count);
207
208     if(alerte==1) {                 //condition de l'alerte
209        digitalWrite(7,HIGH);
210        if(alerte_vite==1)
211        {
212            digitalWrite(0,HIGH);
213        }
214       if(alerte_proche==1)
215        {
216            digitalWrite(2,HIGH);
217        }
218        delay_lidar(1000);
219        digitalWrite(7,LOW);
220        digitalWrite(0,LOW);
221        digitalWrite(2,LOW);
222        alerte=0;
223        alerte_vite=0;
224        alerte_proche=0;
```

```
    }

  //on trouve les points de gravite d'abord
        if (IS_OK(op_result)) {
            drv->ascendScanData(nodes, count);
        }

    op_result = drv->grabScanData(new_nodes, count);
    if (IS_OK(op_result)) {
        drv->ascendScanData(new_nodes, count);
        for (int pos2 = 0; pos2 < 2000  ; pos2=pos2+1) {
            if( ((nodes[pos2].angle_q6_checkbit >> RPLIDAR_RESP_MEASUREMENT_ANGLE_SHIFT)/64.0f<90) && ((new_nodes[pos2].angle_q6_checkbit >> RPLIDAR
            {
            presence_voiture=1;
            num2=pos2;
            gettimeofday(&debut,NULL);
            dis2= (nodes[pos2].distance_q2/4.0f>new_nodes[pos2].distance_q2/4.0f)?new_nodes[pos2].distance_q2/4.0f:nodes[pos2].distance_q2/4.0f;
            angle2=(nodes[pos2].angle_q6_checkbit >> RPLIDAR_RESP_MEASUREMENT_ANGLE_SHIFT)/64.0f;
            x2 = dis2 * sin(angle2*PI/180.0f);
            y2 = dis2 * cos(angle2*PI/180.0f);
            break;
            }
            else
            {
            presence_voiture=0;
            }
        }
```

```
        delay_lidar(500);
        drv->ascendScanData(nodes, count);
        drv->ascendScanData(new_nodes, count);
        for (int pos1 = 0; pos1 < 2000  ; pos1=pos1+1) {
            if( presence_voiture && ((nodes[pos1].angle_q6_checkbit >> RPLIDAR_RESP_MEASUREMENT_ANGLE_SHIFT)/64.0f<angle2) && ((nodes[pos1].angle_q6
            {
            num1=pos1;
            gettimeofday(&fin,NULL);
            dis1= (nodes[pos1].distance_q2/4.0f>new_nodes[pos1].distance_q2/4.0f)?new_nodes[pos1].distance_q2/4.0f:nodes[pos1].distance_q2/4.0f;
            angle1=(nodes[pos1].angle_q6_checkbit >> RPLIDAR_RESP_MEASUREMENT_ANGLE_SHIFT)/64.0f;
            x1 = dis1 * sin(angle1*PI/180.0f);
            y1 = dis1 * cos(angle1*PI/180.0f);
            break;
            }
        }

        //maintenant on a toutes les mesures qu'on a besoin pour calculer le droite
        temps=(fin.tv_sec-debut.tv_sec)+fin.tv_usec-debut.tv_usec;    //le temps en second
        Vitesse=sqrt( (y2-y1)*(y2-y1) + (x2-x1)*(x2-x1) )/temps;
        dis=sqrt( x1*x1 + pow( (x1*y2-x1*y1)/(x2-x1) ,2) );
        t=dis/Vitesse;

    if(Vitesse > double(5000))
    {
        alerte=1;
        }

```

```
    if(Vitesse > 15000)
    {
        alerte_vite=1;
        }
    if(dis < 2000)
    {
        alerte_proche=1;
        }
    if(dis > 5000)
    {
        alerte=0;
        }

        if (ctrl_c_pressed){
            break;
        }

    }         //parenthese pour if
}   //parenthese pour while
    drv->stop();
    drv->stopMotor();
    // done!
on_finished:
    RPlidarDriver::DisposeDriver(drv);
    drv = NULL;
    return 0;
}
```

[4] Les bibliothèque qu'on a appliqué pour lidar:

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <wiringPi.h>
#include <cstdlib>
#include <iostream>
#include <cmath>
#include <math.h>
#include <sys/time.h>

#include "rplidar.h"

[5] Installation d'OpenCV et ses Bibliothèques:

sudo apt-get install libopencv-dev
sudo apt-get install python-opencv

sudo apt-get install build-essential git cmake pkg-config -y
sudo apt-get install libjpeg8-dev -y
sudo apt-get install libtiff5-dev -y
sudo apt-get install libjasper-dev -y
sudo apt-get install libpng12-dev -y
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev -y
sudo apt-get install libgtk2.0-dev -y
sudo apt-get install libatlas-base-dev gfortran -y

wget https://github.com/Itseez/opencv/archive/3.4.0.zip
wget https://github.com/Itseez/opencv_contrib/archive/3.4.0.zip

cd /home/pi/Downloads
unzip opencv-3.4.0.zip
unzip opencv_contrib-3.4.0.zip

cd /home/pi/Downloads/opencv-3.4.0
mkdir build
cd build

[6] Le code du parti webcam

```
 #include <iostream>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include <wiringPi.h>
#include <cstdlib>

using namespace cv;
using namespace std;
int main( int argc, char** argv )
{
if(-1==wiringPiSetup())
{
    cerr<<"setup error\n";
    exit(-1);
}
pinMode(7,OUTPUT);
VideoCapture cap(0); //capture the video from web cam
if ( !cap.isOpened() ) // if not success, exit program
{
cout << "Cannot open the cam" << endl;
return -1;
}
namedWindow("Control", CV_WINDOW_AUTOSIZE);
//create a window called "Control"
int iLowH = 156;
int iHighH = 180;
int iLowS = 43;
int iHighS = 255;
```

```
int iHighS = 255;
int iLowV = 46;
int iHighV = 255;
//Create trackbars in "Control" window
cvCreateTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)
cvCreateTrackbar("HighH", "Control", &iHighH, 179);
cvCreateTrackbar("LowS", "Control", &iLowS, 255);
//Saturation (0 - 255)
cvCreateTrackbar("HighS", "Control", &iHighS, 255);
cvCreateTrackbar("LowV", "Control", &iLowV, 255);
//Value (0 - 255)
cvCreateTrackbar("HighV", "Control", &iHighV, 255);
while (true)
{
Mat imgOriginal;
bool bSuccess = cap.read(imgOriginal);
// read a new frame from video
if (!bSuccess) //if not success, break loop
{
cout << "Cannot read a frame from video stream" << endl;
break;
}
Mat imgHSV;
vector<Mat> hsvSplit;
cvtColor(imgOriginal, imgHSV, COLOR_BGR2HSV);
//Convert the captured frame from BGR to HSV
//因为我们读取的是彩色图，直方图均衡化需要在HSV空间做
split(imgHSV, hsvSplit);
equalizeHist(hsvSplit[2],hsvSplit[2]);
```

```
equalizeHist(hsvSplit[2],hsvSplit[2]);
merge(hsvSplit,imgHSV);
Mat imgThresholded;
inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), imgThresholded); //Threshold the image
/* //开操作 (去除一些噪点)
Mat element = getStructuringElement(MORPH_RECT, Size(5, 5));
morphologyEx(imgThresholded, imgThresholded, MORPH_OPEN, element);
//闭操作 (连接一些连通域)
morphologyEx(imgThresholded, imgThresholded, MORPH_CLOSE, element);
*/
vector<Vec3f> circles;
HoughCircles( imgThresholded, circles, CV_HOUGH_GRADIENT,1.5, 10, 200, 100, 0, 0 );
//依次在图中绘制出圆
for( size_t i = 0; i < circles.size(); i++ )
{
Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
int radius = cvRound(circles[i][2]);
//绘制圆心
circle( imgThresholded, center, 3, Scalar(0,255,0), -1, 8, 0 );
//绘制圆轮廓
circle( imgThresholded, center, radius, Scalar(155,50,255), 3, 8, 0 );
digitalWrite(7,HIGH);
delay(1000);
digitalWrite(7,LOW);
    }
imshow("Thresholded Image", imgThresholded); //show the thresholded image
imshow("Original", imgOriginal); //show the original image
char key = (char) waitKey(300);
if(key == 27)
break;
}
```