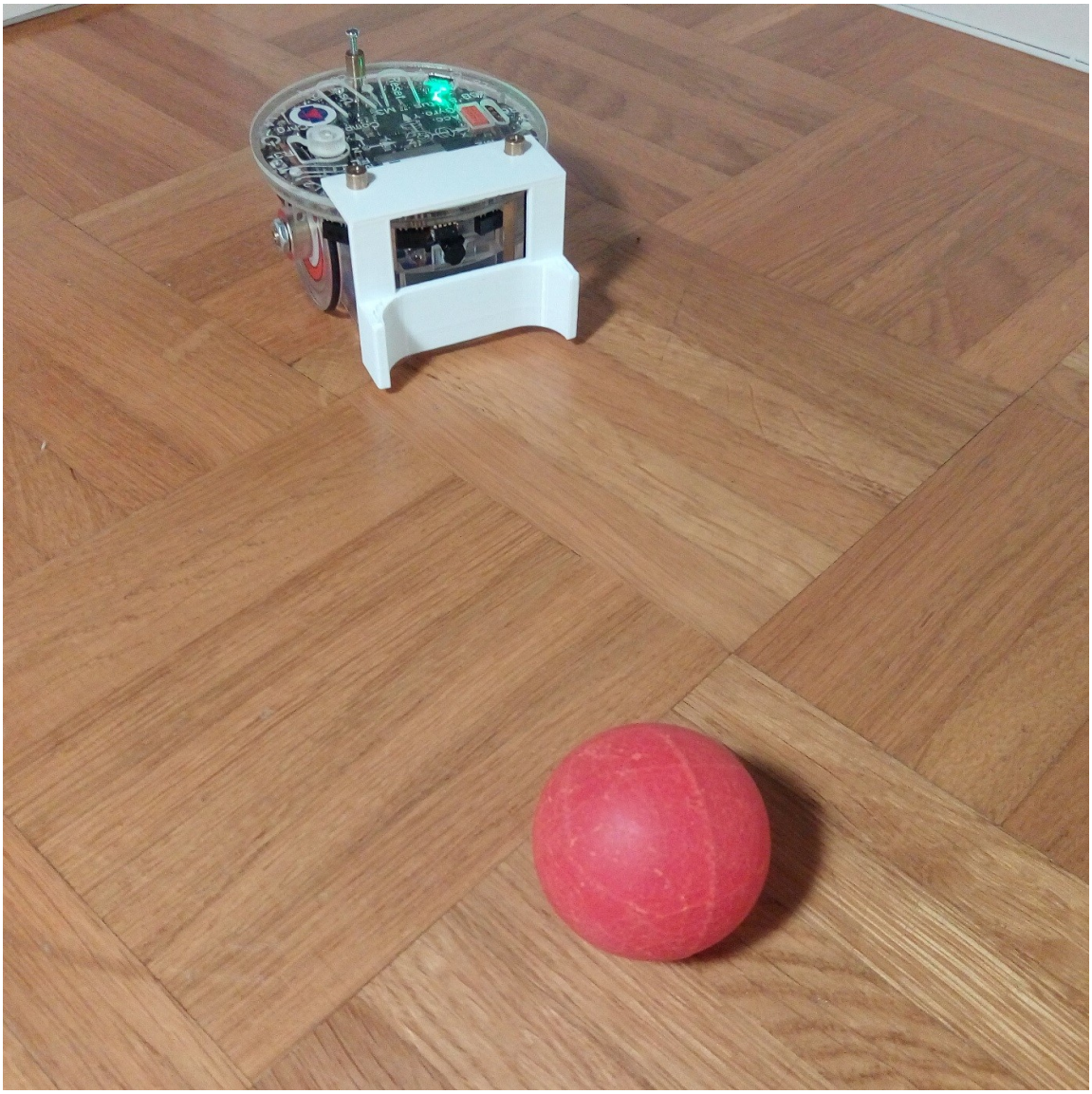
Projet de micro-informatique BA VI

Footy



Pierre Oppliger & William Galand

Semestre de printemps 2020

Table des matières

Introduction

Pour ce projet de micro-informatique, nous avons fait que notre cher e-puck se devait de repérer correctement la balle qui est une boule en plastique, la rejoindre, puis la contourner et enfin frapper dedans pour qu'elle se dirige en direction de la position originelle du robot.
Dans le présent document, nous détaillerons ce projet, ainsi que son fonctionnement.

Exemple d'utilisation

Voici un petit exemple sur la manière d'utiliser le robot avec ce projet Footy, et illustré sur la figure 1 :

1. Allumez le robot sur une surface plane. Une balle doit être disposées quelque part dans l'environnement du robot. Il est conseillé d'éviter la présence d'objet ayant une couleur similaire à celle de la balle. Vous pouvez munir l'e-puck d'un capot en imprimé 3D pour améliorer ses performances futur de frappe. Il y a un temps d'attente de 5[s] avant le passage au point 2 pour vous laisser le temps de le disposer dans son environnement.
2. Le robot va d'abord tourner sur lui-même. Il va partir dans le sens horaire. Si vous préférez un sens de rotation anti-horaire, mettez la main proche du côté droit du robot. Vous pourrez de nouveau inverser le sens de rotation du robot en mettant la main proche du côté gauche du robot.
3. Quand le robot aura localisé la balle, il s'alignera avec le centre de ladite balle. Puis, il se rapprochera et s'arrêtera devant. Pour finir, il la contournera et s'orientera vers son centre.
4. Il la frappera finalement en avance tout droit : la balle ira dans la direction de la position initiale du robot, guidée par l'éventuel capot.
5. Si vous n'interrompez pas le robot (par un reset, une extinction, etc.), il recommencera comme au point (2)

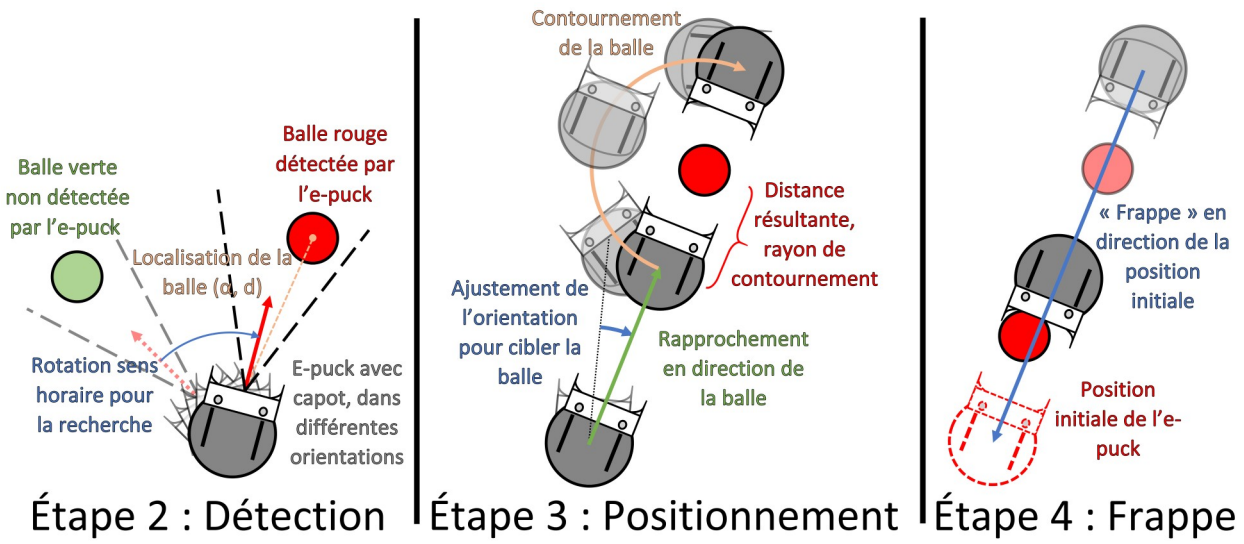


Figure 1 - Présentation schématique de la recherche de la balle en rouge par l'e-puck, en gris.

Fonctionnalités

Implémentation

Nous nous sommes mis la liste suivante des points à respecter et nous les respectons tous :

- Le robot doit pouvoir trouver la balle, la contourner et la frapper
- En tout temps, le robot s'arrête s'il y a un obstacle immédiatement devant lui
- La précision du robot doit être telle qu'un spectateur comprenne visuellement ce que le robot est en train de faire
- Le robot est parfaitement autonome d'un appareil externe (ordinateur, carte programmable, etc.)
- Le robot peut se débrouiller seul dès que l'opérateur l'a mis en position et jusqu'à ce qu'il ait frapper dans la balle
- L'opérateur peut donner une indication sur l'emplacement de la balle, uniquement pour gagner du temps dans la démonstration
- L'opérateur peut demander à ce que le robot accélère afin de remplir sa tâche plus rapidement
- Le robot aura un petit capot confectionné par nos soins pour remédier à la forme arrondie des e-pucks

La liste suivante décrit des fonctionnalités que nous avons essayé d'implémenter et que nous n'avons pas continué car elles étaient trop longues à faire et/ou trop compliquées :

- La balle se fait détecter même en «environnement hostile» : éclairage avec de la lumière colorée, la balle est sale, la balle est extrêmement proche du robot et ne rentre pas entièrement dans le champ de vision de la caméra, etc.

Détail sur l'implémentation : physique

Nous détaillerons ici l'implémentation de certaines des fonctionnalités au niveau physique. Plus de détails pour l'implémentation informatique seront donnés par la suite.

Nous respectons la donnée qui demandait l'utilisation des deux moteurs pas-à-pas, au minimum un des capteurs de distance (nous utilisons le Time-of-Flight ainsi que trois des capteurs de proximités infrarouges) et un capteur parmi ceux investigués pendant les travaux pratiques 3-5 (nous utilisons la caméra).

Déplacements

Afin de se déplacer, le robot utilise ses deux moteurs latéraux. Ils n'ont pas forcément une logique binaire (en rotation ou à l'arrêt) et leur vitesse peut varier en fonction des besoins du robot.

Ceci est particulièrement vrai pour le contournement de la balle (confer à la section «Algorithme de trajectoire»).

Arrêt immédiat

Le robot doit pouvoir s'arrête dès et tant qu'un obstacle est présent devant lui. Nous utilisons pour cela le capteur VL53L0X de type «temps de vol», qui est sur notre robot orienté de 40° vers le haut. Pendant le déplacement, la valeur de ce capteur est régulièrement contrôlé. La bibliothèque ne rafraîchit la valeur que tout les 100ms. Pour ne pas trop consommer, nous vérifions la valeur tout les 150ms. Le temps de réaction de l'e-puck est systématiquement meilleur que si son arrêt devait être commandé par un humain : les réflexes humains sont de l'ordre de 300ms[1] .

Une conséquence intéressante de cette fonctionnalité et de l'angle du détecteur est que le robot s'arrête si jamais il est mis sur le dos.

1 Cordier, F. (2017, 19 septembre), *6-Le temps de réaction, qu'est-ce que c'est ?*, Consulté le 09.05.2020 sur http://acces.ens-lyon.fr/acces/thematiques/neurosciences/outils-numeriques/temps-de-reaction-investigation-variabilite-et-traitements-statistiques-des-donnees/comprendre-1/le-temps-de-reaction-quest-ce-que-cest

Changement du sens de rotation
Durant la phase de recherche du robot, le robot tourne initialement dans le sens horaire, on peut vouloir inverser ce sens. Particulièrement si on voit que le robot aurait mieux fait de partir dans l'autre sens chercher la balle. L'opérateur peut ainsi mettre sa main, ou simplement son doigt pour le guider.
Le robot utilise pour cette détection les capteurs latéraux infrarouges gauche et droit. Il comprend ainsi la présence d'un objet sur le côté et déclenche une inversion du sens de rotation.
Le seuil de sensibilité de ce capteur a été déterminé expérimentalement pour avoir à mettre la main à environs 0.5 cm du robot.

Accélération du robot
L'opérateur peut demander au robot qu'il accomplisse plus rapidement sa tâche. En mettant la main derrière, il peut comme «pousser» le robot, sans contact bien sûr, afin qu'il aille plus vite. C'est de nouveau des capteurs infrarouge qui est utilisé, mais cette fois-ci ceux de derrière. Le fonctionnement est ensuite le même que pour le système de changement de sens de rotation.

Caméra frontale
Le capteur optique frontal est l'un des plus importants pour ce projet car il est nécessaire pour fournir la totalité des indications sur la suite des trajectoires.
Il n'est utilisé que durant la phase de recherche où on lui demande une trame de 640 x 2 pixels, pour des raisons techniques, dont seule une trame de 640 x 1 sera utilisée par la suite. Les données sont demandées en RGB 565[2].
Nous utilisons la Digital Camera Interface (DCMI) comme lien entre le matériel et notre partie du logiciel.

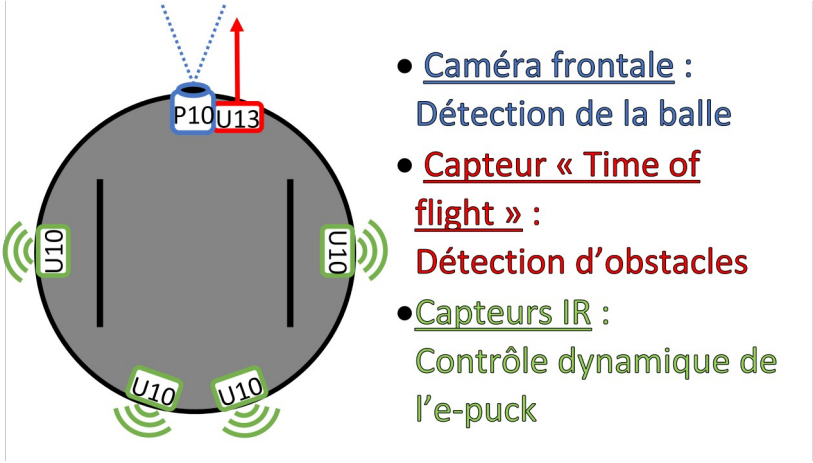


Figure 2 - Schéma récapitulatif des capteurs utilisés. En bleu la caméra, en rouge le capteur temps de vol VL53L0X et en vert les capteurs infrarouges. Les désignations sont celles du dossier électronique de l'e-puck 2.

2 Microsoft Corporation, (2018, 31 mai), *Working with 16-bit* RGB, Consulté le 09.05.2020 sur https://docs.microsoft.com/en-us/windows/win32/directshow/working-with-16-bit-rgb

Détail sur l'implémentation : informatique

Organigramme de dépendances

Nous présentons dans cet organigramme les liens d'inclusions et utilités de nos différents modules.

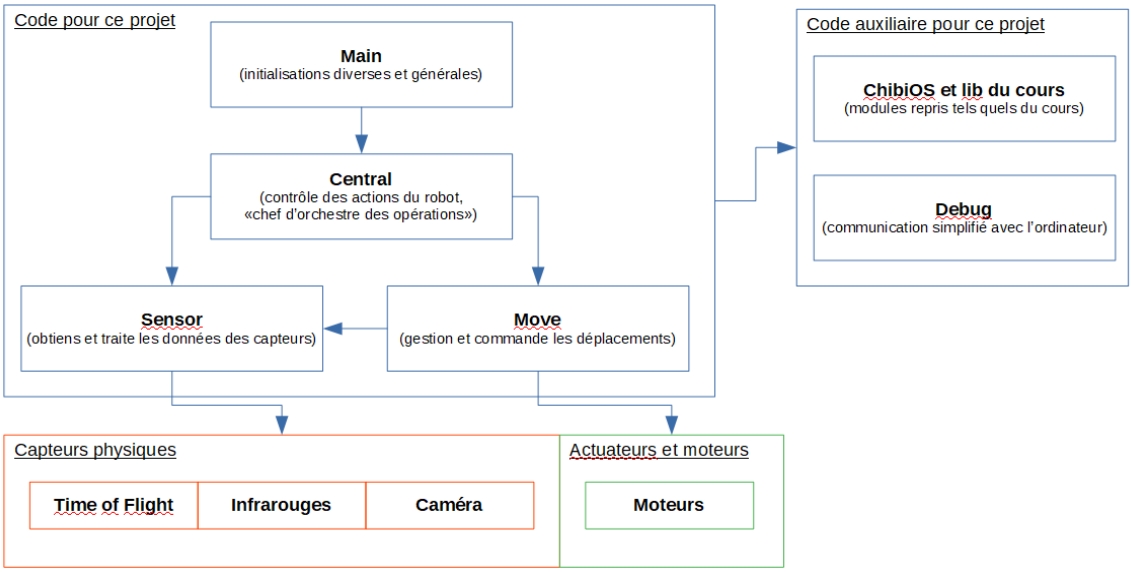


Figure 3 - Organigramme des dépendances et inclusions pour ce projet. Le module main lance les diverses initialisations. Le programme en entier est régulé par le module central qui dicte les tâches à faire. Il délègue la mesure par capteurs ainsi que la traitement des données au module sensor et la commande par actuateurs au module move. Le module move peut d'ailleurs lui-même interroger sensor si nécessaire comme pour l'arrêt immédiat. Le tout repose sur des appels à l'OS et l'utilisation de la bibliothèque du cours.

Algorithme de détection de la balle

Nous revenons maintenant sur l'un des deux «grands» algorithmes de ce projet, la détection de la balle.
Nous avions commencé par reprendre l'idée du TP CamReg du cours en cherchant une grande dérivée dans la courbe des intensités mesurées depuis la caméra. Toutefois, l'environnement est ici plus complexe qu'une ligne noire sur fond blanc : variation lumineuse, balle n'ayant pas une couleur unie, reflets sur la balle, reflets sur les objets de fonds, etc. Nous avons donc changé d'algorithme.
Nous convertissons notre image linéaire RGB565 en une image HSV dans laquelle nous recherchons une trame de pixels ayant la composante Hue dans un intervalle correspondant aux couleurs de la balle (soit $H \leq 0.05$ ou $H \geq 0.85$ pour H dans [0,1]) et vérifions la qualité de l'image par un seuil sur la composante Saturation (soit $S \geq 0.6$ pour S dans [0,1]). Certaines erreurs de détections d'image subsistent mais elles sont grandement réduites.
Afin de gagner en rapidité, pour chaque valeur possible que peut renvoyer la caméra, une table de transposition nous indique si cette valeur indiquerait la présence de la balle. L'e-puck ne fait donc jamais de conversion RGB vers HSV suivie de comparaisons mais une lecture en mémoire, plus rapide.
Dans les deux cas, plusieurs filtres comme le passe-bas avec l'algorithme de Fourier, le filtre à moyenne mobile, la valeur médiane ont été essayés sur les données capturées mais n'apporte pas un réel gain alors qu'ils rendent le traitement de chaque pixel dépendant de celui de ses voisins, ce qui complexifie

sérieusement l'utilisation de la table de transposition[3]. Un filtre à moyenne mobile a toutefois été implémenté sur les valeurs simplifiées retournées par la table de transposition, afin de diminuer les erreurs de détection.

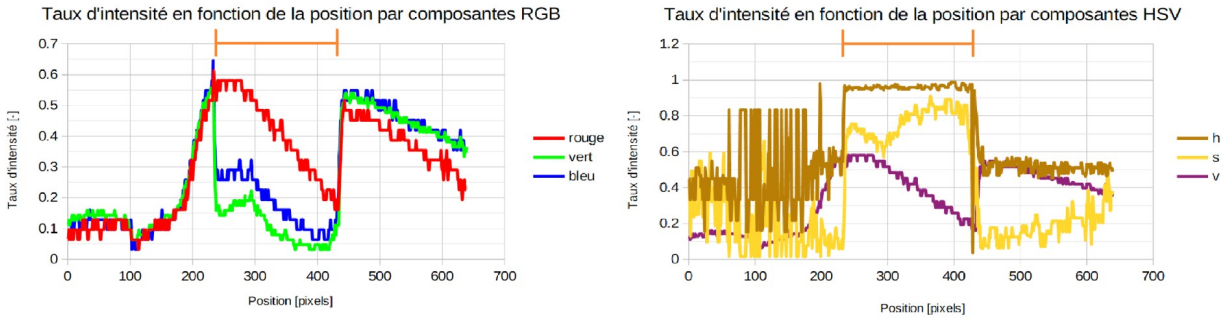


Figure 4 - Ceci est en fonction de la position en nombre de pixels (de gauche à droite) sur une ligne capturée par la caméra, les taux d'intensités par composantes RGB pour l'image de gauche et HSV pour l'image de droite en ordonnée. En orange est indiqué la zone devant être sélectionnée par l'algorithme (du pixel 236 au pixel 430). Cette image est un cas de capture des plus défavorables avec une mauvaise exposition.

Sur la figure ci-dessus on compare les performances des deux algorithmes décrits précédemment. On remarque que l'étude de pente sur le graphe RGB serait problématique pour la détection du bord gauche à cause de cette mauvaise exposition, et le bord droit qui paraît presque blanc. Par contre observer les critères sur l'images en composantes HSV, même fortement bruitée, est possible.

Algorithme de trajectoires

Les trajectoires sont déterminées de manière simplifiée, en ce que les mouvements de l'e-puck sont tous uniformes. Pour les déplacements rectiligne, le module «Move» considère la vitesse et la distance désirée et détermine le temps durant lequel les moteurs pas-à-pas doivent tourner, en prenant en compte la résolution linéaire de l'e-puck pour chaque pas ; les rotations sont établies en inversant la vitesse d'une des roues et en considérant la résolution angulaire ainsi obtenue pour un pas de chaque moteur.

La trajectoire circulaire de contournement est déterminée en fixant les vitesses des moteurs selon les rapports R_{left}/R et R_{right}/R, et selon la vitesse désirée au centre, tel que représenté sur la figure 5. La moitié de la différence entre ces vitesses est ensuite utilisée afin de calculer la durée du mouvement. En effet cet écart induit une rotation relativement au centre de l'e-puck, et l'angle de cette rotation est strictement égal à l'angle de l'arc parcouru. La formule utilisée pour la rotation pure peut donc être réutilisée.

3 Les résultats de certains de ces tests sont disponibles dans le documents etudes_couleurs.ods et echantillons d'images.ods sur notre dépôt github.

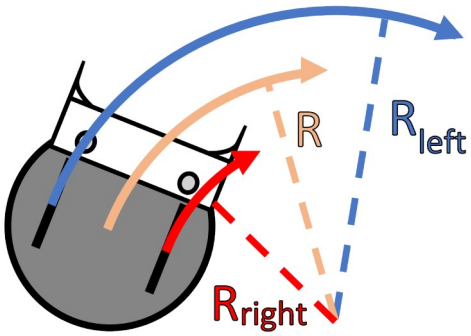


Figure 5 - Mouvement circulaire uniforme de rayon R, les vitesses du centre, de la roue gauche et de la roue droite (respectivement en orange, bleu et rouge) sont proportionnels aux rayon associés.

Détermination de la distance de la balle

La distance de la balle au robot est calculée en fonction de l'angle perçu sur le détecteur, plus la balle est éloignée plus cet angle sera faible. L'expérience a montré que, en raison de la qualité de la détection qui diminue avec la distance, l'application d'un facteur correctif ayant une relation quadratique avec la distance, comme sur la figure 6, menait aux meilleurs résultats. L'effet d'un tel facteur sur des distances déterminées est montré en figure 7.

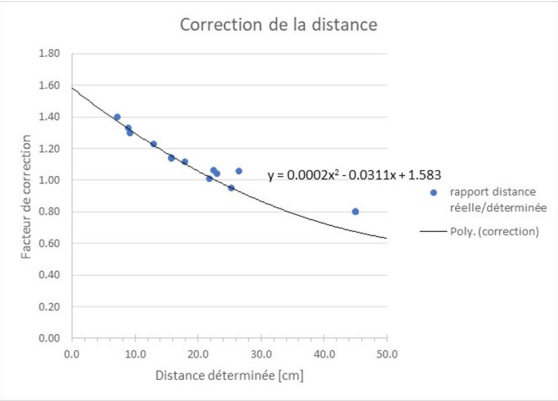


Figure 6 - Terme correctif de la distance. En bleu les relations entre distance réelle mesurée et distance détectée. En noir le terme correctif appliqué en fonction de la distance.

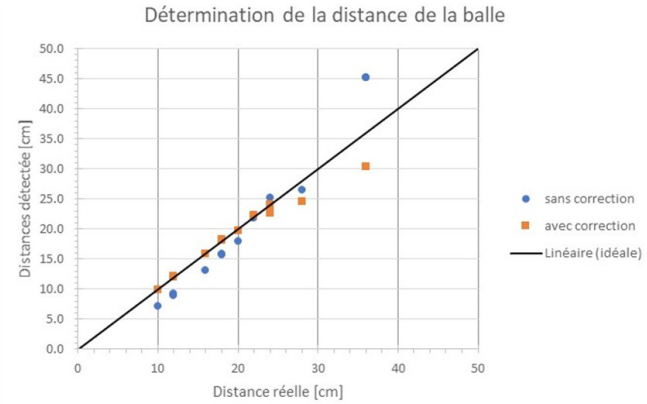


Figure 7 - Exemple de correction sur des distances détectées en fonction de distances réelles mesurées. En bleu les résultats sans correction, et en orange avec. En noir une droite de pente 1 représentant le cas idéal.

Conclusion

Nous avons fort apprécié de faire ce projet. C'était la première fois pour nous que nous faisions un vrai projet de robotique dans le cadre d'un cours académique.

Nous avions tous deux déjà fait un peu de robotique en hobbyiste mais cette fois-ci nous avons pu implémenter ce projet avec la rigueur d'un «vrai projet». Accessoirement, nous avons pu jouer avec une multitude de capteurs que nous avions déjà vu dans d'autres cours et développer en groupe sur Github, ce qui est une expérience différente de celle de développer seul.

Nous remercions le professeur et l'ensemble de ses assistants pour leurs efforts mis dans le projet mais également dans le cours, dans la rapidité et la clarté des réponses à nos questions, surtout dans les conditions imposées par la situation actuelle en Suisse.

Bibliographie

MONDADA, Francesco, (semestre de printemps 2020), *Microinformatique (pour MT)* [notes et slides prises pendant le cours], EPFL
DI SIRIO, Giovanni, (2017), *ChibiOS/RT 3.0The Ultimate Guide*, documentation de ChibiOS
DELANNOY, Claude, (2008), *Apprendre le C++*, Eyrolles, ISBN-13: 978-2212124149
DAVID, H., (2017, 3 août), *Algorithm to convert RGB to HSV and HSV to RGB in range 0-255 for both*, Consulté le 09.05.2020 sur https://stackoverflow.com/questions/3018313/algorithm-to-convert-rgb-to-hsv-and-hsv-to-rgb-in-range-0-255-for-both