Rapport de projet

Equipe de projet 68 :

Guillaume Bouchex-Bellomié, Loïc Broyer,
Benoît Chahriar, Victor du Buat, Nathan Thirion

Mai 2019

Sommaire

I
Introduction

1 Contexte

Notre projet se développe en lien avec la participation du club Robotronik à l'édition 2019 de la coupe de France de robotique entre le 30 mai et le premier juin. Bien que tous les membres de notre équipe soient par ailleurs membres du club Robotronik, ce n'est pas en tant que tels que nous développons ce projet.

Dans cette compétition, deux équipes disposant chacune de un ou deux robots doivent pousser des palets disposés sur le terrain et les ranger dans des cases. Les palets représentent des atomes auxquels une couleur est associé. Les cases sont des zones colorées du terrain placées sur une extrémité du terrain figurant un tableau de Mendeleiev . Par ailleurs, les robots peuvent également pousser un palet dans un couloir placé sur un des bords du terrain. Chacune de ces actions rapporte des points à l'équipe qui sont comptabilisés en fin de partie. Un espace libre est réservé aux équipes pour les dispositifs de leur choix en haut du terrain sur un rectangle de dimensions l*L et de hauteur h (fig. 1)

Afin de pouvoir réaliser ces actions, il est impératif de pouvoir se déplacer précisément sur le terrain et donc de connaître sa position. Par ailleurs, il est impératif de savoir la position des palets pour se diriger. Jusque là, l'asservissement était assuré à l'aide d'encodeurs et le repérage des palets était réalisé en sondant l'environnement à l'aide de capteurs à ultrason disposés sur les robots. Le club Robotronik nous a commandé un système de repérage des palets et des robots basé sur le traitement d'images prises par une caméra placé en haut de l'espace réservé.

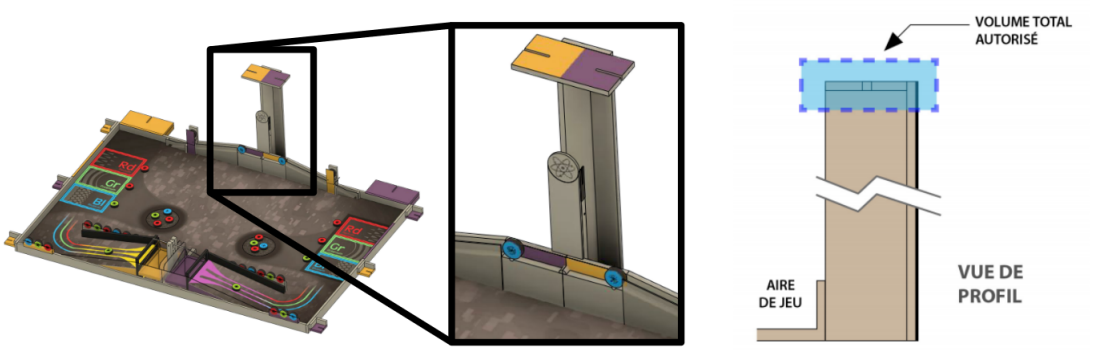

Figure 1: Représentation du terrain de jeu et de la zone utilisable pour notre projet

2 Cadre de notre participation

Ce projet nécessite de pouvoir acquérir des images du terrain, les analyser, et envoyer les données ainsi obtenues aux robots. Pour cela, il a fallu créer un support mécanique afin de pouvoir maintenir la caméra dans la même position souhaitée tout au long du match, acquérir les images, choisir le support du traitement et réaliser le logiciel l'exécutant, puis les modalités de communication.

Le support mécanique doit respecter les règles de la coupe de France c'est à dire ne pas dépasser la position et les dimensions qui lui sont dédiées.

La caméra est positionnée en hauteur sur une plateforme de 20cm sur 40cm partagée entre les deux équipes, laissant un espace disponible de 20cm sur 20cm pour le support. De plus, un dépassement de cet espace de 6cm est autorisé dans toutes les directions (fig. 1) à l'exception de celle du support de l'autre équipe. La fixation à la plateforme se fait grâce à une fente de 8mm de diamètre au milieu de celle ci. Le support comportera donc un trou du même diamètre permettant d'y passer une vis serrée par un écrou papillon, maintenant le tout en place.

Le support en lui même a été conçu pour accueillir une caméra Raspberry Pi et nous avons opté pour un modèle permettant de faire varier l'angle à volonté selon 2 axes (fig. 2)

La caméra n'ayant pas un grand champs de vision, nous l'avons placée sur la plateforme afin qu'elle soit orientée principalement vers le coté du terrain dédié à notre équipe et le plus loin de celui ci. Par conséquent le support et la plateforme

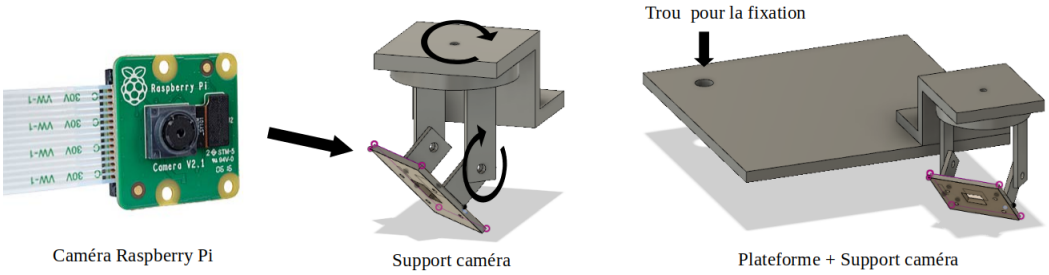

Figure 2: Représentation du système d'attache de la caméra

on été conçus pour pouvoir être séparés, car selon le côté alloué à notre équipe (que l'on ne peut pas prévoir), le montage doit être inversé pour regarder l'autre côté. Les deux axes de rotations peuvent êtres bloqués en serrant les vis des fixations. L'ensemble du montage à été conçu à l'aide du logiciel Fusion 360 de Autodesk et imprimé en 3D. Le reste de l'espace sur la plateforme sert à accueillir la Raspberry Pi en elle même. Nous avons pu tester notre prototype grâce à la réplique du terrain de la coupe de France fabriquée par l'association Robotronik.

3 Division du travail et du projet

II

Description de l'objectif

4 Cahier des charges

Mécanique		
Fonction	**Critère**	**Valeur**
a	b	c

Matériel électronique		
Fonction	**Critère**	**Valeur**
Facilité de configuration du microprocesseur	b	c
Alimentation embarquée	Tension, courant	5V, 3A
Possibilité de connexion de périphériques externes	b	c
Emettre un wifi local	b	c
Disponibilité de librairies et d'utilitaires	b	c

Informatique		
Fonction	**Critère**	**Valeur**
Lancement autonome des scripts au démarrage	b	c
Capacité à calculer l'orientation de la caméra	Précision angulaire	c
Capacité à convertir des coordonnées image en coordonnées physiques	Précision spatiale	c

5 Description théorique

Mécanique et matériel électronique

La Raspberry Pi est chargée avec une carte SD contenant toutes les librairies python utiles pour le projet (OpenCV, wifi, etc), ses codes bash et python. Une caméra est branchée sur un port dédié de la carte et l'alimentation est reliée par un pin dédié sur la carte. Celle-ci est assurée par une batterie reliée à un régulateur de tension / courant permettant d'adapter la sortie de la batterie aux demandes de la Raspberry Pi. En raison du fait qu'on n'aura pas beaucoup de temps ni d'écran lors de la Coupe de France de robotique pour tout mettre en place, la Raspberry Pi execute de manière automatique ses scripts de configuration et lance sa routine python quelques secondes après avoir été mise sous tension.

Informatique

Le support mécanique de la caméra permettant d'obtenir une orientation variable sur deux axes, on doit être capable de déterminer avec précision les angles de la caméra par rapport au plan du terrain. Un code permet de determiner ces paramètres à partir de la connaissance des coordonnées physiques et sur l'image de deux points. Ensuite, on est capable de relier les coordonnées sur l'image aux coordonnées physiques en fonction de la connaissance de la hauteur de la cible (robot ou palet). (Voir Annexe Théorique 1)

Un point d'accès wifi indépendant a été préconfiguré sur la Raspberry Pi et est activé dès que la Raspberry Pi a terminé son chargement. Celui-ci n'est pas sécurisé car cela impliquerait plus de complexité et la sécurité n'est pas nécessaire pour les applications envisagées. A la fin de chaque boucle d'execution du programme principal, les données relatives aux positions des robots et des palets sont envoyées sur le réseau. On pourra également envoyer directement l'image de la caméra (brute ou modifiée) pour débuguer en l'absence d'écran.

III
Mécanique et matériel

6 Impression 3D

7 Matériel électronique

On a directement envisagé l'utilisation d'une Raspberry Pi pour effectuer toutes les opérations en raison du fait qu'elle permette d'intégrer un OS complet contrairement aux autres cartes (Arduino, STM32, etc). Ceci permet d'avoir déjà à disposition la librairie graphique OpenCV, l'interpréteur Python et d'autres utilitaires tel que le wifi. De plus, elle intègre un port dédié pour une caméra avec son logiciel. Nous avons priviligié l'utilisation de Raspbian comme OS en raison du fait qu'il intègre des pilotes dédiés pour la caméra. Pour l'alimentation en période de développement, nous avions dans un premier temps envisagé d'utiliser un PC relié par USB, cependant cela ne satisfaisait pas les demandes en courant de la Raspberry Pi, aboutissant à des corruptions de la carte SD. Nous avons donc retenu l'alimentation par des chargeurs dédiés reliés à une prise électrique. Pour ce qui est de l'alimentation embarquée, nous avons prévu d'utiliser une batterie avec un régulateur de tension / courant.

Sur la Raspberry Pi, on a réussi à compiler la librairie OpenCV utile au traitement d'image. Grâce à elle nous avons pu visualiser les images sortant de la caméra et nous assurer qu'elle fonctionnait bien. Le lancement autonome des scripts prévus au démarrage a en revanche rencontré un problème. En effet, lorsque les commandes sont lancées depuis le terminal, tout fonctionne. Mais lorsque celles-ci sont executées depuis un script bash, on rencontre des erreurs. Nous nous contenterons de démarrer la Raspberry Pi en avance et de l'initialiser à la main si nous ne trouvons pas de solution à ce problème.

IV
Informatique

8 Traitement de l'image

9 Conversion de coordonnées

En raison de la complexité des relations trigonométriques permettant de retrouver les angles en fonction de points sur l'image et sur le terrain, on exprime un angle en fonction de l'autre puis on utilise une procédure itérative de recherche du minimum de distance entre la position calculée et la position réelle. On a d'abord utilisé une résolution au centième de radian, puis on est passé au millième de radian. En effet, bien qu'en théorie le minimum est unique, il existe de nombreux points très proches du minimum et étant donné la résolution limitée, on peut le manquer et trouver un pseudo minimum avec des angles qui donneront un résultat complètement faux pour une autre position sur l'image.
On a mesuré l'orientation de la caméra et pris une image avec. Puis on a effectué des tests sur douze points de l'image en mesurant physiquement leur position sur le terrain avec la résolution du centième de radian et du millième de radian. On a observé dans tous les dcas des angles différents au centième de radian et dans tous les cas les bons angles au millième de radian. On garde donc la résolution au millième de radian.

Le code de conversion des coordonnées image vers les coordonnées physiques permet en théorie de déterminer totalement la position des cibles, à condition que la position et l'orientation de la caméra soit précisément connue. En pratique, on retrouve effectivement cette limitation, la précision des positions calculées étant très dépendante de la précision des mesures de la position de la caméra (celles-ci impactant également le code de détermination des angles). En général, on trouvait les positions à moins d'un centimètre près.

10 Connexion Wifi

La Raspberry Pi avec Raspbian offre de nombreux utilitaires pour le wifi, donc nous nous basons sur leur utilisation. Du côté client, nous avions dans un premier temps codé un script python permettant de recevoir des données, pensant utiliser une Raspberry Pi sur les robots. Mais pour des raisons techniques, nous n'utiliserons qu'un module wifi branché à une carte STM32 sur les robots et avons codé ceci différemment. Cependant, ceci ne concerne pas strictement le projet et nous nous contenterons du code client python ici.

Le point d'accès wifi a été configuré avec succès pour se lancer au démarrage de la Raspberry Pi. De plus, un code permettant d'envoyer n'importe quel type d'objet python par wifi a été réalisé. Ainsi, nous avons pu nous assurer que tout fonctionnait bien en envoyant par wifi le flux vidéo de la caméra vers un PC distant. Cependant, nous avons pu constater une importante latence (de l'ordre de quelques secondes) entre la réception et la capture des images. On peut attribuer cela au volume des images envoyées et des tests sur des envois de données plus petites nous a donné raison. Du point de vue des clients, nous n'avons pas de problème à connecter un PC au wifi. Pour connecter les robots avec leur module wifi, cela nécessite encore du travail, mais c'est hors sujet ici.

V
Conclusion

References

Annexes

Annexe 1 : Description théorique de la conversion des coordonnées image vers les coordonnées physiques

Cette annexe a pour but d'expliquer les mathématiques derrière la conversion des coordonnées image vers les coordonnées réelles.
On considérera dans la suite la géométrie de la (REF) pour la position et l'orientation de la caméra.

On part d'une image avec des coordonnées exprimées en pixels, la position (0,0) étant en haut à gauche de l'image (REF). Il sera important dans la suite de considérer si l'image est retournée ou pas (la gauche de l'image correspondrait à la droite de la caméra).

On va chercher à convertir cette image en coordonnées pixel en coordonnées sur un plan image fictif. Pour cela, on supposera que la caméra se situe à une distance 1 du plan. Ainsi, la distance maximale au centre de la bijection de l'image sur le plan image est en $\tan(\beta_0)$ et $\tan(\gamma_0)$. Ainsi, on utilisera la conversion décrite par (1).

$$\begin{cases} x = \tan(\beta_0)(\frac{2u}{resoX} - 1) \\ z = \tan(\gamma_0)(\frac{2v}{resoY} - 1) \end{cases} \tag{1}$$

resoX étant le nombre de pixels en x et resoY respectivement en y, u et v étant les coordonnées sur l'image.

On peut interpréter la position sur le plan image comme un vecteur directeur de direction à partir de la caméra. On veut convertir cette direction dans le repère de la table et non plus celui de la caméra. Pour cela on va effectuer deux rotations successives correspondant aux angles γ_0 et β_0 de la caméra (2).

$$\vec{P_{rot}} = \begin{pmatrix} \cos(\omega) & \sin(\omega)\cos(\alpha) & \sin(\omega)\sin(\alpha) \\ \sin(\omega) & \cos(\omega)\cos(\alpha) & \cos(\omega)\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix} \quad \vec{P_{cam}} \tag{2}$$

A présent que l'on connaît la direction dans le repère du plan de jeu, on va calculer le point d'intersection de la droite partant de la caméra dans la direction $\vec{P_{rot}}$ avec le plan situé à une hauteur h de la caméra (Cette hauteur étant à adapter en fonction de la nature de la cible, robot ou palet). Ces coordonnées s'obtiennent selon 3.

$$\begin{cases} x = h * \frac{P_{rotX}}{P_{rotZ}} \\ y = h * \frac{P_{rotY}}{P_{rotZ}} \end{cases} \tag{3}$$

Il ne reste plus qu'à éventuellement tenir compte d'une translation de la caméra par rapport à l'origine fixée quelque part sur la table.

Resumé
Abstract