Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Projet de semestre 5 / 2020-2021

Filière Informatique

Eurobot - Système de localisation basé sur la vision par ordinateur

Documentation

09.11.20 – Version 0.1

Denis Rosset

Superviseurs : **Jacques Supcik**
Nicolas Schroeter

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

Table des versions

Version	Date de publication	Auteur	Description
0.1	09.11.2020	Denis Rosset	Ébauche de la documentation du premier sprint
0.1.1	09.11.2020	Denis Rosset	Mise à jour des points concernant l'analyse des caméras et des systèmes embarqués

Table des matières

1 Introduction

Cette section décrit le but du projet, la situation actuelle et les objectifs à réaliser.

1.1 Acteurs

Ce projet est suivi par les personnes suivantes :

- Jacques Supcik, *Superviseur*
- Nicolas Schroeter, Superviseur
- Denis Rosset, Etudiant

1.2 Contexte

Depuis plusieurs années, notre école participe au concours de robotique Eurobot. Le rêglement de ce concours est disponible ici :

https://www.coupederobotique.fr/wp-content/uploads/Eurobot2020_Rules_Cup_OFFICIAL_FR.pdf

Ce concours réunit des étudiants des filières informatique, génie électrique et génie mécanique, dans le but de réaliser deux robots capables d'effectuer différentes tâches. Une partie dure 100 secondes durant lesquelles les robots se déplacent sur le terrain en même temps. Aucun contact n'est permis entre les robots et aucune tentative d'entraver le bon fonctionnement des systèmes ennemis n'est autorisé. La nécessité de connaitre la position des robots (alliés et adverses) est primordiale afin de prévoir les déplacements et les actions à effectuer.

La filière informatique a déjà développé plusieurs solutions pour connaitre la position des robots sur le plateau, mais ces méthodes souffraient de limites en termes d'utilisation et de précision. Le but de ce projet est de créer un système de localisation à partir d'un système composé d'une ou plusieurs caméras posées sur un mât au-dessus du plateau (figure 1 rond rouge) et de marqueurs présents sur les robots et sur la table (figure 1 rond violet).

Figure 1 - Vue générale de l'aire de jeu

Le cahier des charges est disponible ici : https://gitlab.forge.hefr.ch/denis.rosset/sauron2021/-/blob/master/doc/cahier%20des%20charges/cahier_des_charges_v1_0.pdf

2 Sprint 1 Analyse

Le premier sprint est principalement voué à réaliser des tests technologiques, essayer différents systèmes embarqués et caméras et comprendre les fonctionnements des algorithmes de détection des ArUcos.

2.1 Analyse de risques

Une analyse de risque à été effectuée tôt dans le projet afin de se rendre compte des problèmes qui seront rencontrés et de comment les résoudre.

	Chance d'occurence			
	1%	5%	20%	50%
Système inutilisable	P5	P4, P7	P7	
Système fonctionne partiellement ou grosse perte de précision (<15cm)	P2		P3	
Système continue de fonctionner, perte de précision modéré (<5cm)		P1	P8	
Système conitnue de fonctionner, légère perte de précision (<1cm)			P6	

ID	Risque
P1	Intensité lumineuse trop faible
P2	Système ébloui par l'adversaire
P3	Déplacement involontaire du système (vibrations)
P4	Problème de communication avec le robot
P5	Problème de fonctionnement du hardware
P6	Mauvaise installation du système (légèrement décalé)
P7	Mauvaise installation du système (extrèmement décalé)
P8	Arucos mal imprimés / mal collés

Grâce à cette analyse de risques, nous pouvons nous rendre compte des problèmes auquels va être confronté le projet et ainsi les prévoir et les éviter.

2.2 Algorithme

Le but principal du projet est de détecter la position d'ArUcos sur le terrain grâce à une caméra. L'algorithme qui est testé et utilisé dans les benchmarks est décrit dans ce paragraphe. Des publications scientifiques traitant du sujet ont été publiées en Juin 2018

2.2.1 Publications scientifiques

Détection des marqueurs :

https://www.researchgate.net/publication/325787310_Speeded_Up_Detection_of_Squared_Fiducial_Markers

Génération des marqueurs **:**

https://www.researchgate.net/publication/282426080_Generation_of_fiducial_marker_dictionaries_using_Mixed_Integer_Linear_Programming

Résumé :

Plusieurs filtres sont appliqués sur l'image afin de pouvoir savoir quels endroits de l'image sont des marqueurs et lesquels sont des informations qui ne nous intéressent pas.

(Figure 2) Un algorithme de seuillage est appliqué à l'image originale (a) pour arriver à l'image (b) puis une extraction des contours est faite (c). Les polygones à quatre côtés détectés sont filtrés afin de se concentrer sur les zones intéressantes (d). Une image canonique est calculée pour un des carrées (e) puis la méthode d'Otsu est appliquée (f). Une liste binaire est obtenue et permet de reconstituer l'entier qui lui correspond afin de trouver quel est l'ArUco correspondant.

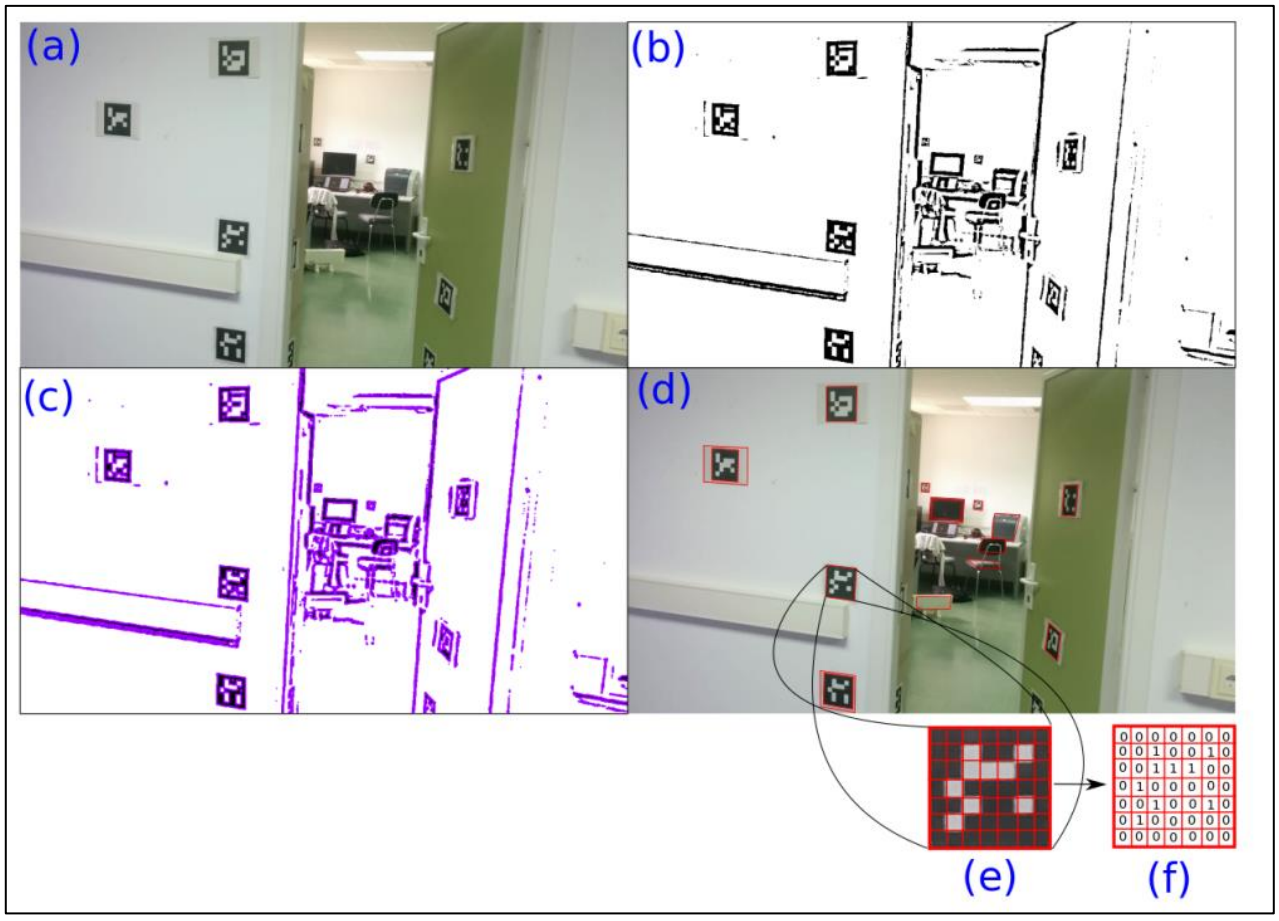

Figure 2 Etapes de traitement d'une image contenant des ArUcos

Les processus principaux pour la détection rapide et l'identification sont les suivants :

(Figure 3) L'image originale (a) est redimensionnée pour la recherche des ArUcos. Un seuillage est appliqué (c) afin de trouver les rectangles entourés en rouge (d). Une correspondance avec les marqueurs originaux (e) est faite et accélérée grâce à une Pyramide (f). Les coins des ArUcos sont ensuite utilisés pour trouver leurs localisations dans l'image originale.

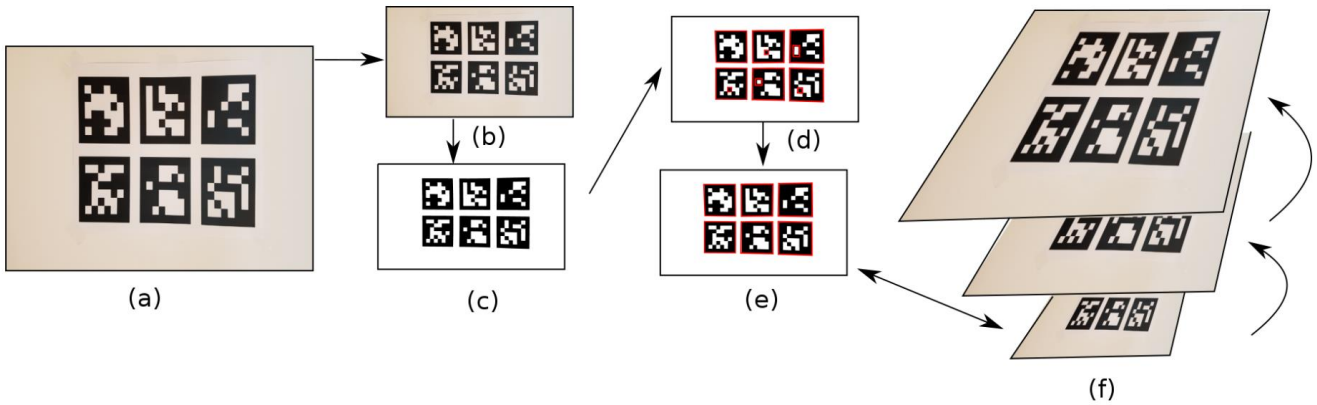

Figure 3 Processus de pipeline

(Figure 4) Avec la position des coins des ArUcos et en utilisant des projections 2d sur plan 3d, on peut, si on connaît la position d'un des marqueurs (rond bleu : marqueur déjà présent sur l'aire de jeu), retrouver la position relative des autres marqueurs (rond rouge : marqueur fictif) par rapport à lui.

Figure 4 Aire de jeu avec un ArUco fictif (rouge)

2.3 Système embarqué

Afin de réaliser le traitement d'image, et dans une nécessité de performance, des systèmes embarqués ont été comparés puis un a été choisi. Pour déterminer le choix du système embarqué, beaucoup de critères ont dû être pris en compte. Des benchmarks ainsi qu'une comparaison sous forme de tableau multicritère ont été réalisés pour simplifier ce choix.

Le choix à été fait entre les deux systèmes embarqués les plus répandus dans le domaine de la vision par ordinateur :

- Le Jetson Nano : https://developer.nvidia.com/embedded/jetson-nano-developer-kit
- Le Raspberry Pi 4 : https://www.raspberrypi.org/products/raspberry-pi-4-model-b/?resellerType=home

Bien que le Rapsberry Pi ait un CPU légèrement plus performant que le Jetson Nano, l'avantage principal du second est le fait qu'il possède un GPU. Ce GPU permet de paralléliser une partie de la charge CPU nécessaire à la détection des ArUcos.

2.3.1 Benchmarks

Le code des benchmarks est disponible dans les répertoires respectif aux systèmes embarqués à l'adresse suivante : https://gitlab.forge.hefr.ch/denis.rosset/sauron2021/-/tree/master/utilities

Les résultats des benchmarks sont disponibles ici : https://gitlab.forge.hefr.ch/denis.rosset/sauron2021/-/tree/master/doc/benchmark

Les résultats des benchmarks ont été mis en page dans le tableau ci-dessous afin de les rendre plus lisibles :

Temps de calcul en secondes	**Jetson Nano**		**Raspberry Pi 4**	
Moyenne sur 5 échantillons	1x	50x	1x	50x
Prise de vue (raspi) / copie de la mémoire (jetson)*	0.011	0.547	0.762	38.107
Algorithme de détection d'ArUcos	0.526	26.277	0.638	31.915
BackgroundSubstraction de matrice aléatoire	0.256	12.229	0.804	80.351
BackgroundSubstraction de matrice aléatoire (CUDA)	0.053	2.668	N/A	N/A

* La différence dans les tests du temps nécessaire à la prise de vue est due au fait que le Jetson Nano fait par défaut du buffering des prises de vues en parallèle alors que le Raspberry Pi 4 (avec la librairie utilisée) ne parallélise pas la prise de vue.

On peut voir dans ce tableau que les temps demandés dans le cahier des charges pour la détection des ArUcos sont réalisables avec le Jetson Nano et le Raspberry Pi. Le Jetson Nano étant celui qui pourrait recevoir le plus d'optimisation possibles grâce à son GPU le place néanmoins en tête de concours.

2.3.2 Critères de choix

Un tableau multicritère à été réalisé pour comparer les aspects externes aux systèmes embarqués.

	Poid	Jetson Nano		Rapberry pi 4	
Performances	**5**	CF Benchmark	5	CF Benchmark	2
Simplicité d'utilisation*	**2**	GPU + CUDA	2	CPU	4
Taille (installation)	**2**	70 x 45 mm	2	85,60 mm × 53,98	4
Consomation	**2**	5 volt 2 amp	3	5 volt 2 amp	3
Wi-Fi 5 GHz	**1**	5 GHz avec dongle	2	5 GHz sans dongle	4
Total	12		41		36

La simplicité d'utilisation concerne la facilité d'installation de tous les composant nécessaire à l'exécution de l'algorithme final (système embarqué, librairies, drivers, etc…). Ces points sont précisés dans le sous chapitre interfaçage.

Ce tableau à permis de se rendre compte qu'à part dans les performances, le Raspberry Pi à de très bons résultats. Etant donné la nécessité du projet à avoir un système performant et efficace, une grande pondération a été donnée pour les performances et le choix à la fin du premier sprint se portera sur Jetson Nano.

2.4 Caméra

Pour la caméra, les nécessités étaient de pouvoir voir tout le terrain en même temps avec la plus grande résolution possible et le minimum de déformation.

Le choix de la caméra se portait sur deux candidats potentiels :

La arducam imx219 wide angle : https://www.pi-shop.ch/arducam-imx219-wide-angle-camera-module-drop-in-replacement-for-raspberry-pi-v2-and-jetson-nano-camera

La raspberry pi camera fisheye : https://www.pi-shop.ch/raspberry-pi-camera-i-fisheye-lens-fischauge

Un test de qualité à été réalisé pour les deux caméras (figure 5) :

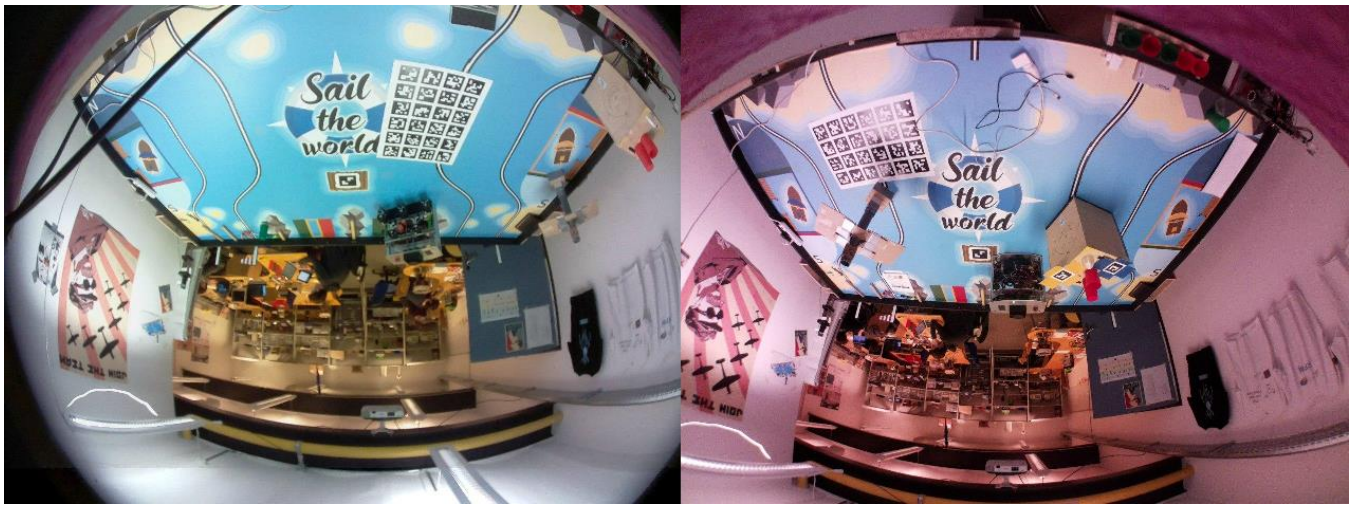

Figure 5 Test de résolution des caméras. A gauche Raspberry pi cam fisheye. A droite Arducam imx219.

Les différences de résolutions ne sont pas visibles sur les images entières. De meilleures images devraient être prises pour mieux illustrer la différence. Les différences de couleurs ne sont pas importantes car l'algorithme fonctionne en noir et blanc.

Un tablea multicritère a aussi été réalisé pour les caméras :

	Poid	Raspberry pi camera fisheye		Arducam imx219 wide angle	
Résolution	**5**	2K	3	4K	5
Angle	**2**	125°	3	130°	4
Version	**5**	1	1	2	5
Total			26		58

La Arducam imx219 wide angle est objectivement meilleure sur tous les points comparés. Elle sera donc choisie pour le prochain sprint.

2.5 Installation de l'environnement de développement pour les tests technologiques

Différents logiciels et librairies sont nécessaires au bon fonctionnement des systèmes embarqués.

2.6 Interfaçage entre la caméra et le système embarqué

Pour permettre aux différents systèmes embarqués d'utiliser les caméras, des manipulations hardware doivent être réalisées.

3 Sprint 2 prototype sans communication